M4.252 - HTML y CSS

PEC 3

Aníbal Santos Gómez

PARTE 1 - TEORÍA

1. **Explica los principales pros y contras de Grid, Flex y floats en cuanto al maquetado de layouts web. Indica para qué es mejor cada una de estas técnicas.**

   Float no está diseñada para maquetar layouts, pese a utilizarse para eso. El uso de float se limita a especificar que un elemento flotará a la derecha o a la izquierda del contenedor y el resto del contenido de este flotará a su alrededor. Sin embargo, el uso de floats para posicionar elementos en nuestro layout puede dar lugar a una serie de extraños bugs que requieren de parches para solucionarlos.

   Ventajas:

   - Método popular.
   - Los fallos que se producen están muy bien documentados y se han desarrollado muchos métodos para corregirlos.

   Desventajas:

   - Necesitan ser reseteados (clearfix). Si se utilizan media queries para personalizar la apariencia en función del dispositivo será necesario hacer un clear para cada adaptación.
   - No se pueden alinear en vertical.
   - Las alturas de las capas no son iguales.
   - Dependen del orden en que aparezcan las capas en el HTML.

   Flexbox permite maquetar layouts que se adaptan al tamaño de pantalla y que nos libran de las desventajas de usar float, como el ajuste de nuestros elementos en ciertas medidas de pantalla. Una de las desventajas de flexbox es que solamente

nos permite organizar nuestros elementos en horizontal o en vertical, pero no ambos a la vez, lo que limita enormemente nuestras posibilidades.

Flexbox funciona muy bien en una sola dirección ya sea vertical o horizontal, pero si queremos trabajar en un layout posicionando elemento en ambas direcciones nos enfrentamos a otro de los problemas comunes, no solamente en flexbox sino también usando floats.

Otro inconveniente a la hora de usar tanto flexbox como float es el hecho de tener que agrupar nuestros elementos en cajas, ya que necesitamos que un grupo concreto de elementos se posicione conjuntamente en una misma zona. En el caso de flexbox, suponiendo que estamos trabajando en un layout horizontal deberíamos agrupar todos los elementos que queramos posicionar verticalmente en un nuevo tag padre, que contendrá todos los elementos a posicionar verticalmente y especificar allí la dirección vertical. Hacer esto nos obliga a añadir tags de html innecesarios y que no aportan ningún valor semántico.

Ventajas:

- Independencia del orden en que aparezcan las capas en el HTML. Elimina la necesidad de utilizar JavaScript para esto.

- Ofrecen alineación vertical.

- Permiten crear capas con la misma altura.

- Permiten la estructura en filas o columnas de manera sencilla.

- Ofrecen una gran flexibilidad en cuanto a las opciones a utilizar.

- Las cajas pueden ocupar automáticamente el espacio disponible, crecer o menguar a petición.

Desventajas:

- La sintaxis es algo compleja.
- Es necesario utilizar prefijos para soportar todos los navegadores.
- No es compatible con IE9 y anteriores.

CSS Grid que no nos fuerza a usar parches para lograr el comportamiento esperado de nuestro css y nos ofrece la posibilidad de maquetar un layout en ambas direcciones sin la necesidad de añadir elementos extras e innecesarios para lograrlo.

Otra ventaja de usar grid es el hecho de que consigue separar nuestro html del css, ya no necesitamos modificar o adaptar nuestro html para que se comporte bien con nuestro css.

CSS Grid funciona mediante un container que engloba una serie de elementos sobre los que se aplican las propiedades de grid.

Ventajas:

- Eliminación total de la necesidad de definir capas contenedoras.
- Independencia del orden en que aparezcan las capas en el HTML.
- Independencia completa entre HTML y presentación visual.
- No necesitamos escribir grandes cantidades de código, con pocas líneas de código podemos colocar los mismos elementos de HTML en diferentes posiciones en función del estilo de cada página.
- Es muy amigo de los móviles, utilizando media queries es muy sencillo convertir un layout de escritorio en un layout de móvil sin necesidad de cambiar nada de nuestro marcado HTML.

En resumen la tendencia es utilizar CSS Grid, y complementariamente Flexbox, ya que si simplemente tenemos un listado de elementos en horizontal o vertical, que tienen características similares y que siempre guardan patrones parecidos, utilizaremos Flex.

Si por el contrario existen elementos diferentes, que pueden colocarse en diferentes posiciones según el tamaño de la pantalla, o si el layout es demasiado complicado como para realizarlo con Flex, utilizaremos Grid.

2. **Explica en breves palabras qué es el diseño responsivo (responsive design) y enumera al menos cuatro características y/o técnicas que ayudan a realizar un diseño responsivo. Explica brevemente qué se entiende por "mobile first".**

El responsive design es una configuración en la que el servidor siempre envía el mismo código HTML a todos los dispositivos y se usa CSS para modificar el procesamiento de la página en el dispositivo.

Es una técnica de diseño y desarrollo web que mediante el uso de estructuras flexibles (contenedores flexibles, imágenes y video flexibles) y junto con Media Queries especificados en CSS, logran adaptar un sitio web al entorno del dispositivo en el que se encuentre. Con ésta técnica se consigue que el contenido del diseño de un sitio se vea bien y pueda ser legible para los usuarios sobre el dispositivo con el que éste interactúe.

Características más importantes del diseño responsive

- Adapta la web al ancho del dispositivo. El diseño normalmente pasa de ser un diseño horizontal a un diseño vertical, adaptándose al ancho del dispositivo. El usuario no tiene que desplazarse horizontalmente.

- Reorganiza los distintos elementos de la web. Los contenidos de la web se disponen de tal forma que puedan verse correctamente en todas las

pantallas. Se utilizan botones de mayor tamaño, para facilitar la interacción táctil, se reajustan el tamaño y la separación de la letra, para que sea más legible y se usan imágenes adaptables para cada dispositivo.

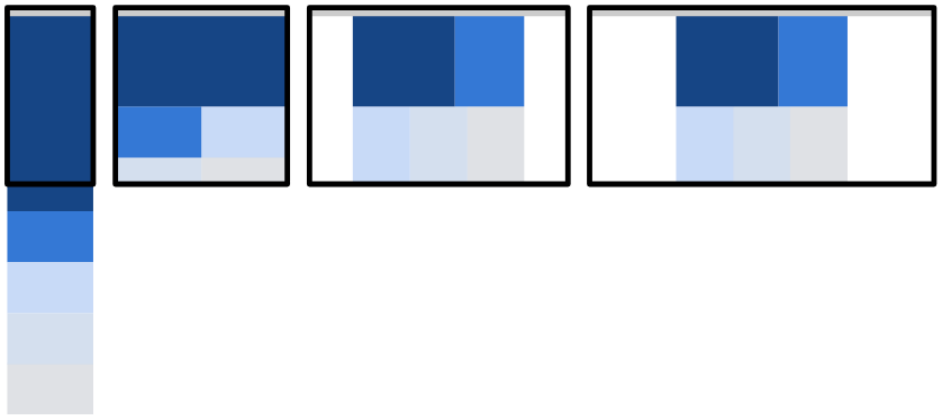
- Simplifica la web. Se priorizan los elementos, solo se utilizan los necesario eliminando objetos gráficos estéticos y que sólo tienen sentido en la versión de escritorio. También se eliminarán los elementos que tardan mucho en cargarse.

- Cambio de apariencia de algunos elementos. Como por ejemplo el menú de navegación, que pasa a ser desplegable, lo que ahorra espacio de pantalla.

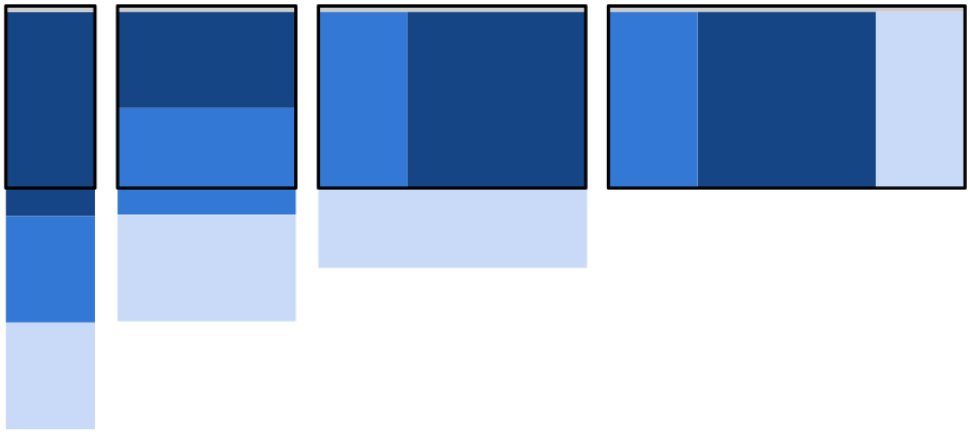
Como técnicas o patrones de diseño responsive, tendremos las siguientes, que son las más conocidas:

- Mostly fluid consiste en una cuadrícula fluida. En las pantallas grandes o medianas se mantiene el mismo tamaño y simplemente se ajustan los márgenes en las más anchas.

  En las pantallas más pequeñas, la cuadrícula fluida genera el reprocesamiento del contenido principal, mientras que las columnas se apilan verticalmente.



- Drop Column, en el caso de los diseños con varias columnas de ancho completo, durante el proceso de colocación de columnas éstas únicamente se colocan de forma vertical debido a que el ancho de la ventana es demasiado reducido para el contenido. En un momento dado, todas las columnas se apilan verticalmente.



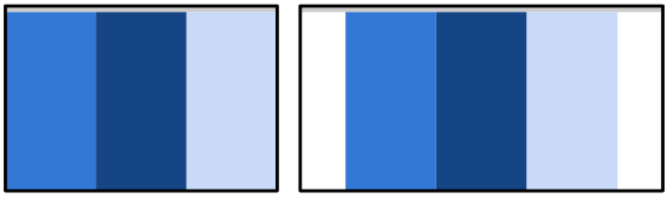
- Layout shifter es el más adaptable, ya que posee varios puntos de interrupción en diferentes anchos de pantalla.

  Debido a las diferencias significativas entre cada punto de interrupción principal.

- Tiny tweaks permite realizar pequeños cambios en el diseño, como ajustar el tamaño de la fuente, cambiar el tamaño de las imágenes o desplazar el contenido de maneras muy poco significativas.

  Funciona correctamente en diseños con una sola columna, como los sitios web lineales de una sola página y los artículos con mucho texto.

- Off canvas coloca contenido menos usado (tal vez menús de navegación o de apps) fuera de la pantalla y solo lo muestra cuando el tamaño de la pantalla es suficientemente grande. En las pantallas más pequeñas, el acceso al contenido es posible con solo a un clic.



Mobile first.

Es una forma de mejora del diseño responsive progresivo, que se enfoca en la priorización del diseño y el desarrollo para dispositivos móviles por encima del diseño y desarrollo para pantallas de escritorio.

Básicamente este tipo de diseño prioriza la construcción del diseño primero en dispositivos pequeños para abordar después los intermedios o más grandes. Es decir, primero dipositivos móviles y después los de escritorio.

PARTE 2 - INFORME

A continuación se realiza un informe documental para el trabajo sobre el sitio web trabajado en esta práctica. Se describen en él los siguientes puntos:

1. Descarga e instalación del repositorio.
2. Organización de ficheros y archivos.
3. Arquitectura del proyecto.
4. Gestión y organización de clases HTML y CSS.
5. Diseño gráfico y maquetación.
6. Despliegue.

1. Descarga e instalación del repositorio.

Este proyecto se encuentra alojado en el siguiente repositorio en la platafora GitHub, para descargarlo debemos seguir las siguientes instrucciones:

1. Descargar o clonar el repositorio:

   Podremos descargar el repositorio en la siguiente dirección:

   https://github.com/ansango/Master-Websites-Applications-UOC.git

   O bien clonarlo mediante el sistema de control de versiones Git, ejecutando el siguiente comando en terminal:

   git clone https://github.com/ansango/Master-Websites-Applications-UOC.git

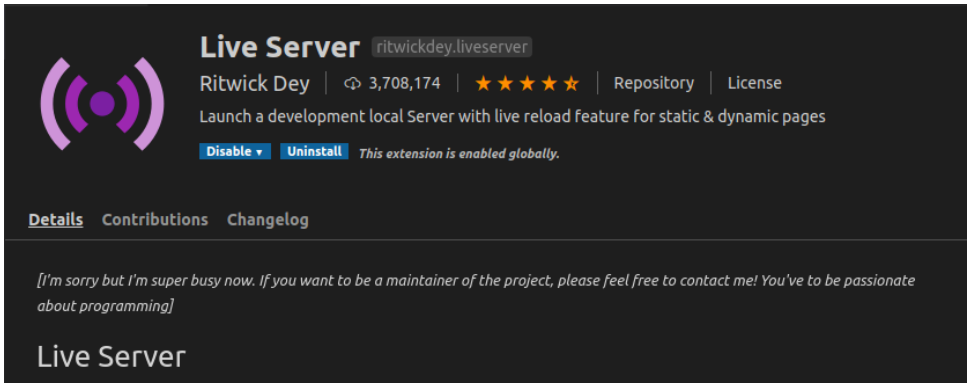
   En este repositorio encontraremos la documentación del trabajo correspondiente a esta práctica y el resto de proyectos presentados en el Máster de desarrollo de sitios y aplicaciones web cursado en la UOC.

2. Para acceder al proyecto una vez descargado el repositorio, debemos seguir la siguiente ruta:

   Master-Websites-Applications-UOC/01 - HTML, CSS/02 - PECS/03 - PEC3/

   Dentro de esta carpeta encontraremos dos carpetas, Materiales Pec 3 y Site. Site es el directorio del proyecto.

3. Para instalarlo no necesitamos ningún gestor de paquetes, simplemente abrir la carpeta Site en un editor de código como Visual Studio y utilizar algún plugin para crear un servidor virtual que vaya actualizando nuestros cambios en tiempo real, sin necesidad de guardar y refrescar el navegador, como es el caso de la extensión “Live Server”, que podremos encontrar en extensiones en Visual Studio Code.



2. Organización de ficheros y archivos.

Los archivos que se trabajan en este proyecto tienen la siguiente distribución:

- **Site**

  En Site encontraremos todos los archivos que conforman el proyecto y sobre los cuales debemos trabajar.

- **css**
  - **estilos.css**

  En el directorio css encontraremos el archivo estilos.css donde se encuentra la estructura de maquetación css que formatea el estilo de todo el proyecto. Cada acción que se realiza se encuentra comentada en el archivo bajo el sistema de comentarios que soporta css y que se explicaran en punto cuarto de este epígrafe.

- **img**
  - **Archivos .png y .svg**

  En img encontraremos las imágenes que se tratan en en el proyecto, tanto las imágenes ejemplificativas, pictogramas y los logotipos.

- **contacto.html**

  El archivo de contacto.html representa el documento html del sitio que establece el método de contacto con nuestro sitio web.

- **index.html**

  El archivo de index.html representa el documento html del sitio que establece el índice de acceso a nuestro sitio web.

- **objetivo-1-fin-pobreza.html**

  El archivo de objetivo-1-fin-pobreza.html representa el documento html del sitio que establece el acceso al primer objetivo relacionado con la prelación de objetivos recogidos en objetivos.html.

- **objetivos.html**

  El archivo de objetivos.html representa el documento html del sitio que establece los objetivos informativos de nuestro sitio web.

3. Arquitectura del proyecto.

En cuanto a la arquitectura del proyecto, nos encontramos ante una arquitectura web plana horizontal. Este tipo de arquitectura tiene normalmente menos niveles de profundidad, es decir, los usuarios deben hacer un menor número de clicks para encontrar cualquier página en el sitio web.

La arquitectura es la siguiente:

- Inicio.

  Página principal de acceso con enlaces al resto de niveles de la arquitectura.

  - Objetivos.

    Página de acceso a los objetivos principales.

    - Objetivo-1 fin de pobreza.

      Objetivo a desarrollar como primero de la lista anterior.

  - Contacto

    Página de contacto.

  Dentro de esta arquitectura encontramos una serie de elementos genéricos adaptativos a diferentes dispositivos:

  - Barra de navegación.

    La barra de navegación se encuentra presente en todos los documentos html para navegar entre los mismos.

  - Pie de página.

El pie de página se encuentra presente dentro de todos los elementos haciendo referencia a los créditos de autor, así como diferente información secundaria como las redes sociales, política de privacidad, cookies, etc...

Continuación de la arquitectura:

La arquitectura desarrollada en este proyecto es básica creada en HTML y CSS plano, la línea de desarrollo avanzada a seguir consistiría en adaptar este proyecto, mediante un framework frontend como Angular, Vue o React para la estructura del proyecto siguiendo por ejemplo un Modelo Vista Controlador mediante un modelo de Single Page Application.

Además podríamos utilizar algún framework estable que nos permita agilizar el desarrollo CSS como por ejemplo Bootstrap o Bulma CSS, dependiendo de la libertad y los requisitos posteriores del proyecto y el cliente.

4. Gestión y organización de clases HTML y CSS.

- HTML:
    - Elementos comunes:
        - <!DOCTYPE html>: Declaramos el tipo de documento, nos sirve para saber que nuestro documento está siguiendo la estructura determinada por un DTD, que determina los atributos que pueden tener las etiquetas.
        - <html lang="es">: Utilizamos esta etiqueta para declarar el documento html y en él, el idioma que está escrito, “es”.
        - <head>: declaramos la cabecera del documento, donde quedan reflejadas etiquetas de título de página, y los metas, además de la importación de los estilos css mediante la etiqueta link.

- <meta>: aportamos información del documento, como la descripción de cada documento, la compatibilidad con navegadores, el tipo de caracteres soportados, si soporta viewport. Todo esto se utiliza para el posicionamiento en motores de búsqueda.

- <title>: describe el título del documento.

- <link>: es un elemento que carga un recurso externo, en este caso cargamos nuestros estilos css haciendo referencia mediante el atributo rel a stylesheet y con href indicamos la ruta del archivo. Además con él importamos la fuente de Google Open Sans.

■ <body>: representa el contenido de un documento HTML.

- <header>: representa un grupo de ayudas introductorias o de navegación. Hemos definido la barra de navegación con una clase para modificar los estilos de forma sencilla y genérica para todos los documentos.

    - <nav>: representa una sección de una página cuyo propósito es proporcionar enlaces de navegación. Aquí hemos utilizado para componer la navegación una lista no ordenada <li> que incluye una etiqueta <a> para referirnos a cada documento mediante su atributo href.

- <main> utilizamos main para definir los bloques de contenido, debe de ser uno por documento, después se explicarán en cada uno de los documentos, ya que son diferentes como decimos.

- <footer>: esta etiqueta hace referencia al pie de página donde figuran los créditos así como los diferentes enlaces a redes

sociales y otros enlaces informativos acerca de la privacidad, sobre nosotros o aviso legal. Se forma mediante un genérico o contenedor, un <p> para los créditos de autor y otro para crear la flecha que nos lleva a la parte superior de cada documento.

- Inicio:
  - <main class="content">: en este caso tenemos un main formado por dos <article>:

    - <article class="index-article-1">: agrupa un h1 que es el título principal de la página, un párrafo explicativo y un contenedor del iframe que presentamos:

      - <h1>: utilizamos h1 como elemento de título de primer nivel único para cada <main> de cada documento, donde indicamos el título del mismo.

      - <p>: utilizamos p para distribuir los textos de los diferentes documentos.

      - <iframe>: para insertar el video de youtube, utilizamos la etiqueta iframe, que nos embebe un video de la fuente correspondiente.

    - <article class="index-article-2">: agrupamos h2 y un contenedor de las imágenes de cada objetivo.

      - <h2>: utilizamos h2 como subtítulo de este segundo <article>.

      - : este div agrupa cada link y cada imagen correspondiente a cada objetivo.

        - <a>: utilizamos <a> para linkear a otros sitios web externos o internos dentro de nuestros documentos.

            - <img>: para insertar las imágenes en el documento utilizamos el tag img.

- Objetivos:

    - <main class="objetives-content">: en este caso tenemos un main formado por un <ul> y un <article>:

        - <ul class="breadcrumb">: esta lista no ordenada indica que estamos ante la navegación de breadcrumb, en este caso formada por dos <li> y sólo uno de estos <li> con un <a>.

            - : nos sirve para determinar cada elemento de la lista o bien ordenada o sin ordenar.

            - <a>: utilizamos <a> para linkear a otros sitios web externos o internos dentro de nuestros documentos. En este caso linkeamos hacia el documento index.html

        - <article class="objetives-container">: este <article> agrupa el el contenido principal del main, donde encontraremos un único <h1> como título del documento, y 17 <section>, que contienen dos tipos de clases, class=”pair” y class=”unpair”:

            - <h1>: utilizamos h1 como elemento de título de primer nivel único para cada <main> de cada documento, donde indicamos el título del mismo.

            - <section>: representa una sección genérica de un documento. Sirve para determinar qué contenido

corresponde a qué parte de un esquema. En este caso sirve para identificar elementos pares e impares para posteriormente maquetarlos con cada clase anteriormente mencionada.

- : es un contenedor genérico para maquetar las imágenes
    - <img>: para insertar las imágenes en el documento utilizamos el tag img.
- : es un contenedor genérico para maquetar el texto y el link a cada objetivo.
    - <h2>: utilizamos h2 como subtítulo cada <section>.
    - <p>: utilizamos p para distribuir los textos de los diferentes documentos.
    - <a>: utilizamos <a> para linkear a otros sitios web externos o internos dentro de nuestros documentos. En este caso linkeamos hacia cada objetivo concreto (solo el primero (objetivo-1-fin-pobreza.html) tiene enlace, el resto están vacíos).

- Objetivo-1 Fin de pobreza:

  - <main class="objetives-content">: en este caso tenemos un main formado por un <ul> y un <article>:

    - <ul class="breadcrumb">: esta lista no ordenada indica que estamos ante la navegación de breadcrumb, en este caso formada por dos <li> y sólo uno de estos <li> con un <a>.

      - : nos sirve para determinar cada elemento de la lista o bien ordenada o sin ordenar.

      - <a>: utilizamos <a> para linkear a otros sitios web externos o internos dentro de nuestros documentos. En este caso linkeamos hacia el documento index.html y objetivos.html.

    - <article class="objetive-subcontent">: en este caso, el <article> agrupa dos :

      - : este div supone el contenedor de un submenu, compuesto por:

        - : este dive es un contenedor de la imagen del pictograma primero.

        - : supone una lista no ordenada compuesta por <li> y <a> que utilizamos como submenú para desplazarnos dentro del contenido del documento objetivo-1-fin-pobreza.html.

      - : este contenedor integra un <h1> y tres <section>:

- <h1>: utilizamos h1 como elemento de título de primer nivel único para cada <main> de cada documento, donde indicamos el título del mismo.

- <section>: en este caso cada <section> hace referencia a data, goals y links con un identificador como selector cada uno está compuesto por lo siguiente:

    - <h2>: utilizamos h2 como subtítulo cada <section>.

    - : una lista desordenada de <li> <p> en el caso de id=”data” y una lista desordenada de <li> <a> en el caso de id=”links”.

    - : una lista ordenada de <li> <p> en el caso de id=”goals”.

- Contacto:
    - <main class="contact-content">: en este caso tenemos un main formado por un <ul> y un <article>:

        - <ul class="breadcrumb">: esta lista no ordenada indica que estamos ante la navegación de breadcrumb, en este caso formada por dos <li> y sólo uno de estos <li> con un <a>.

            - : nos sirve para determinar cada elemento de la lista o bien ordenada o sin ordenar.

- <a>: utilizamos <a> para linkear a otros sitios web externos o internos dentro de nuestros documentos. En este caso linkeamos hacia el documento index.html.

- <article class="form-container">: en este caso, se agrupan un <h1> y un <form>

  - <h1>: utilizamos h1 como elemento de título de primer nivel único para cada <main> de cada documento, donde indicamos el título del mismo.

  - <form>: con form declaramos el bloque que contendrá el formulario con sus labels, inputs y otro tipo de etiquetas de contenido. Está compuesto por:

    - <section class="input-container">: tenemos dos sections compuestos el primero por:

      - <h2>: utilizamos h2 como subtítulo cada <section>.

      - <label>: representa una etiqueta para la interfaz que enlazamos con un input mediante los atributos for para label e id para input.

      - <input>: con input registramos los datos del formulario que posteriormente serán enviados al servidor y almacenados en una base de datos. Contamos con atributos de tipo, como texto o fecha, un id asociado al for de un label un name y un required.

El segundo por:

- <h2>: utilizamos h2 como subtítulo cada <section>.

- <select>: representa un control que muestra un menú de opciones. Las opciones contenidas en el menú son representadas por elementos <option>, es usado en el formulario contenido en el documento contacto.html dos veces para pedir datos al usuario.

- <option>: lo utilizamos para representar los ítems dentro del select en el formulario de contacto.html.

- <p>: utilizamos p para distribuir los textos de los diferentes documentos.

- : contenedor genérico para los siguientes elementos:

  - <input>: input de tipo checkbox acerca de la política de protección de datos.

  - <a>: enlace referente acerca de la política de protección de datos.

- : contenedor genérico para maquetar el botón de envío de los datos del formulario, que contiene lo siguiente:

  - <input>: input de tipo submit.

- CSS:
    - Elementos comunes:
        - /* GENERAL */

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
  font-family: 'Open Sans', sans-serif;
}
```

Aquí hacemos un reset de todos los estilos que puedan ser aplicados por el navegador u otros agentes externos, quitando márgenes, paddings y configurando la fuente predeterminada del documento Open Sans.

        - /* COLORS */

```
:root {
  --text: #000000;
  --text-hover: #ffffff;
  --current: #306291;
}
```

Mediante custom variables fijamos los colores que utilizaremos durante el resto de la plantilla para no tener que indicar el color de forma individual y así poder reutilizar estas variables.

- /* FONTS */

```
h1 {
  font-size: 30px;
  line-height: 35px;
  text-align: center;
}

h2 {
  font-size: 22px;
  line-height: 25px;
  text-align: center;
}

footer, article {
  font-size: 18px;
}
```

En esta parte definimos que h1 tendrá un tamaño de 30 px, interlineado a 35px y estará centrada, al igual h2, pero con un tamaño de 22 px e interlineado a 25px. El tamaño de la fuente para cada footer y cada article será de 18px.

- Mobile: a continuación se explica la estructura por defecto que corresponde a móviles porque el resto están generadas por media queries adaptando la estructura por defecto mencionada.

  - /* NAV */

```
.main-nav img {
  width: 40px;
  padding: 20px 0 20px 0;
}
```

Con esta clase y el selector img, fijamos el tamaño de la imagen a 40px y un padding superior e inferior de 20px.

```
.main-nav a {
  text-decoration: none;
  color: var(--text);
}
```

Con esta clase y el selector a, quitamos el subrayado y fijamos el color del enlace.

```
.main-nav a:hover {
  color: var(--text-hover);
}
```

Con esta clase y el selector a:hover, fijamos el color del hover.

```
.main-nav>ul>li {
  padding: 10px;
  text-align: center;
  list-style: none;
  font-weight: bold;
}
```

Con esta clase y los selectores ul>li, fijamos un padding general de 10px, centramos el texto, quitamos los bolos de la lista y fijamos el ancho de la fuente en negrita.

```
.main-nav>ul>li>.current {
  color: var(--current);
}
```

Con esta clase y el selector ul>li y la clase current, fijamos el color de cada enlace en el cual nos encontramos por documento como activo.

- /* FOOTER */

```
.main-footer {
  width: 100%;
  padding: 0 25px 0 25px;
}
```

Aplicamos con la clase a todo el footer un ancho del 100% con un padding derecho e izquierdo de 25px.

```
.main-footer a {
  text-decoration: none;
  color: var(--text);
}
```

Quitamos el subrayado y fijamos el color a cada enlace contenido en la clase seleccionada.

```
.links-group, .links-social {
  padding: 25px 0 25px 0;
}
```

Para estos contenedores fijamos un padding superior e inferior de 25px.

```
#arrow a {
  float: right;
  padding: 20px 0 20px 0;
}
```

Con el selector por id y a, seleccionamos el posicionamiento flotante a la derecha y establecemos un padding superior e inferior de 20px.

```
.main-footer a:hover {
  color: var(--text-hover);
}
```

Fijamos el color del hover de cada enlace con la clase y el selector a:hover.

```
.main-footer ul>li {
  list-style: none;
  text-align: center;
  font-weight: bold;
  padding: 5px;
}
```

Quitamos los bolos de la lista, centramos el texto, fijamos el ancho de la fuente a negrita y establecemos un padding genérico de 5px con la clase y el selector ul>li.

```
.main-footer p {
  text-align: center;
  padding: 0 15px 0 15px;
}
```

Centramos y establecemos un padding derecho e izquierdo de 15px a todos los párrafos de la clase seleccionada.

```
.main-footer img {
  width: 40px;
}
```

Con el contenedor y el selector img, fijamos el ancho de la imagen a 40px.

- /* MAIN */

```
main {
  width: 100%;
  padding: 0 20px 0 20px;
}
```

Fijamos que todos los main de cada documento estén al 100% de ancho del documento con un padding derecho e izquierdo de 20px por defecto.

```
main>article>h1 {
  padding: 30px 0 30px 0;
}
```

Cada título principal (h1) contenido en el article de cada main, tendrá un padding de 30 px superior e inferior.

```
ul.breadcrumb {
  margin-top: 40px;
  list-style: none;
  text-align: center;
}
```

Para la lista desordenada de clase breadcrumb fijamos un margen superior de 40px, sin los bolos y centramos el texto.

```
ul.breadcrumb li {
  display: inline;
  font-size: 18px;
}
```

Cada elemento de la lista desordenada de la clase breadcrumb tendrá un display en línea con un tamaño de fuente de 18px.

```
ul.breadcrumb li+li:before {
  padding: 8px;
  color: var(--text);
  content: "/\00a0";
}
```

Cada elemento anterior además tendrá un padding de 8px de separación, así como un color fijo y el contenido de cada uno se setea antes a / mediante la propiedad content.

```
ul.breadcrumb li a {
  color: var(--current);
  text-decoration: none;
}
```

Cada elemento anterior que sea un enlace tendrá otro color fijo y además quitaremos el subrayado.

```
ul.breadcrumb li a:hover {
  color: var(--current);
  text-decoration: underline;
}
```

Respecto del selector anterior, fijamos un hover con un color determinado y un subrayado como efecto visual.

- /** INDEX **/

```
.index-title-description {
  text-align: center;
  padding-bottom: 30px;
}
```

Fijamos el texto descriptivo alineado al centro y establecemos una separación mediante un padding inferior de 30px.

```
.embed-container {
  position: relative;
  padding-bottom: 56.25%;
  height: 0;
  overflow: hidden;
}
```

Respecto del contenedor del iframe fijamos una posición relativa, una separación inferior mediante un padding al 56,25 % que supondrá la adaptabilidad del contenedor, un alto de 0 y ocultaremos el excedente con overflow hidden.

```
.embed-container iframe {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  padding: 15px;
  border: 0;
}
```

Respecto del elemento iframe, fijamos una posición absoluta respecto del contenedor anterior, fijando el punto superior e izquierdo a cero, con un ancho del 100% y una altura también del 100%, establecemos también un padding general de 15px y quitaremos los bordes.

```
.index-article-1 {
  width: 100%;
  padding-top: 30px;
}
```

Respecto del contenedor del primer article, fijamos el mismo al 100% con una separación superior mediante padding de 30px.

```
.index-article-2 {
  width: 100%;
  padding: 50px 10px 0px 10px;
}
```

Respecto del contenedor del segundo article, fijamos el mismo al 100% con una separación superior de 50px, y derecha e izquieda de 10px.

```
.index-article-2 h2 {
  padding: 20px 0 20px 0;
}
```

Separamos los h2 del segundo article de index superior e inferiormente con un padding de 20px.

```
.index-img-container img {
  width: 100%;
  height: auto;
  padding: 25px;
}
```

Fijamos cada imagen al 100% de ancho respecto del contenedor anterior, seteamos la altura de forma automática y establecemos un padding general de 25px.

- /** OBJETIVES **/

```
.objetives-content {
  max-width: 850px;
  padding: 0px 30px 0px 30px;
}
```

Fijamos el contenedor a una anchura máxima de 850px con una separación con padding derecha e izquierda de 30px. Esto afectará al breadcrumb y al article con la clase objetives-container siguiente.

```
.objetives-container {
  padding-top: 25px;
}
```

Para el article con esta clase establecemos un padding superior de 25px.

```
section img {
  width: 100%;
  height: auto;
  padding: 50px 100px 50px 100px;
}
```

Respecto a cada imagen de cada section fijamos un ancho del 100%, una altura auto y un padding derecho e izquierdo de 100px y superior e inferior de 50px.

```
.text-container p {
  padding: 10px 0 15px 0;
  text-align: justify;
}
```

Los textos de cada section tendrán un padding superior e inferior de 15px y un el texto estará justificado.

```
.text-container a {
  margin-top: 20px;
  color: var(--text);
  text-decoration: none;
}
```

Cada enlace de cada contenedor de cada texto tendrá un margen superior de 20px, un color determinado y no llevará decoración de subrayado.

```
.text-container a:hover {
  color: var(--current);
  text-decoration: underline;
}
```

En cambio cuando hagamos hover sobre lo anterior, fijamos un color diferente, y establecemos como decoración del texto en hover un subrayado.

- /** OBJETIVES 1 **/

```
.objetive-submenu {
  padding: 10px;
}
```

Fijamos para el submenú un padding general de 10px.

```
.objetive-submenu img {
  width: 100%;
  height: auto;
  padding: 30px 50px 30px 50px;
}
```

Para la imagen del submenú fijamos un ancho del 100%, una altura auto, y un padding superior e inferior de 30px así como derecho e izquierdo de 50px.

```
.objetive-submenu>ul {
  padding: 5px;
  text-align: center;
  list-style: none;
  font-weight: bold;
}
```

Para la lista desordenada del submenu, fijamos un padding de 5px, centramos el texto, quitamos los bolos y establecemos el ancho de la fuente a negrita.

```
.objetive-submenu>ul>li {
  padding: 5px;
}
```

Para cada elemento de la lista fijamos un padding general de 5px.

```
.objetive-submenu>ul>li>a {
  color: var(--current);
  text-decoration: none;
}
```

Para cada enlace de cada elemento quitamos el subrayado y fijamos un color.

```
.objetive-container {
  padding: 10px;
}
```

Para el contenedor del objetivo fijamos un padding de 10px.

```
.objetive-container h1 {
  padding: 10px 0 10px 0;
}
```

Para cada h1 del contenedor del objetivo fijamos un padding superior e inferior de 10px.

```
.objetive-container h2 {
  padding: 10px 0 10px 0;
  text-align: start;
}
```

Respecto a cada subtítulo (h2), fijamos un padding superior e inferior de 10px

```
.objetive-container>section>ul>li {
  margin: 20px;
  text-align: justify;
}
```

Para cada elemento de cada lista desordenada de cada section del contenedor anterior fijamos un margen general de 20px y justificamos el texto.

```
.objetive-container>section>ol>li {
  padding: 10px 0 10px 0;
  text-align: justify;
  margin: 20px;
}
```

Para cada elemento de la lista ordenada de cada section del contenedor anterior fijamos un padding superior e inferior de 10px, justificamos el texto y un margen general de 20px.

```
.objetive-container>section>ul>li>a {
  color: var(--current);
  text-decoration: none;
}
```

Para cada enlace de la lista desordenada final quitamos el subrayado y fijamos un color.

- /** CONTACT **/

```
.form-container {
  padding: 20px 0 10px 0;
  width: 100%;
}
```

Fijamos para el contenedor un ancho del 100% con un padding de 20px y 10px superior e inferior.

```
form section>h2 {
  text-align: start;
  padding-top: 10px;
  padding-bottom: 10px;
}
```

Para cada subtitulo de cada section de form, alineamos el texto al principio y con un padding superior e inferior de 10px.

```
.input-container {
  padding: 10px 0 10px 0;
}
```

Respecto a esta clase fijamos un padding superior e inferior de 10px.

```
.input-container p {
  padding: 10px 0 10px 0;
  font-size: 1rem;
}
```

Cada párrafo tendrá un padding superior e inferior y un tamaño de fuente de 1rem.

```
.input-container a {
  font-size: 1rem;
  text-decoration: none;
  color: var(--current);
}
```

Cada enlace tendrá un tamaño de fuente de 1rem y un color fijo, no tendrá subrayado.

```
input[type=text] {
  width: 100%;
  padding: 5px;
  margin-top: 5px;
  margin-bottom: 5px;
  background-color: var(--text-hover);
  border: 1px solid var(--text);
  border-radius: 4px;
  font-style: italic;
}
```

Cada input de tipo texto tendrá un ancho del 100%, un padding de 5px, un margen superior e inferior de 5px, un fondo de color dijo, un borde de 1px solid junto con el color anterior, un borde redondeado de 4px y un estilo de fuente itálica.

```
input[type=checkbox] {
  margin-left: 10px;
}
```

Cada input de tipo checkbox tendrá un margen izquierdo de 10px.

```
.input-submit-container {
  max-width: 150px;
  margin-left: auto;
  margin-right: auto;
  padding: 20px 0px 20px 0px;
}
```

La clase corresponde al contenedor del botón de envío del formulario, que tendrá un máximo de ancho de 150px, un margen izquierdo y derecho auto para centrar el botón en el contenedor y un padding superior e inferior de 20px.

```
input[type=submit] {
  background-color: var(--current);
  color: var(--text-hover);
  padding: 12px 50px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  text-transform: uppercase;
  font-weight: bold;
}
```

El input de tipo submit tendrá un fondo de color y un color fijos, un padding de 12px 50px, sin bordes con un radio de 4px, fijamos un puntero cuando se ponga el cursor por encima, el texto del botón irá en mayúscula y el ancho del mismo será en negrita.

```
.visuallyhidden {
  display: none;
}
```

Con esta clase ocultamos de la vista los label del formulario, que deben estar presentes para la accesibilidad, pero no para nuestro diseño.

```
select {
  width: 100%;
  padding: 5px;
  margin-top: 5px;
  margin-bottom: 5px;
  border-radius: 4px;
  background-color: var(--text-hover);
  font-weight: bold;
  border: 1px solid var(--text);
}
```

Fijamos el ancho del select al 100%, con un padding de 5px, un margen superior e inferior de 5px, un radio para el borde de 4px, un fondo fijo, un ancho para la fuente en negrita y un borde de 1px solid con un color fijo también.

```
textarea {
  width: 100%;
  height: 200px;
  border-radius: 4px;
  outline: none;
  resize: none;
  overflow: auto;
  border: 1px solid var(--text);
}
```

En cuanto al textarea, fijamos un ancho del 100%, con una altura de 200px, un radio del borde de 4px, que no tenga outline, que no puede redimensionarse, cuyo rebase es auto y con un borde de 1px solid con color fijo.

- Other devices: para crear la adaptación responsive hemos utilizado media queries y display flex en las diversas situaciones que así lo requerían, de la siguiente manera:

  - @media (min-width: 655px): a partir de un ancho de 655 px las reglas genéricas cambiarán a las siguientes:

    - /* NAV */

```
.main-nav {
  width: 650px;
  margin-left: auto;
  margin-right: auto;
}
```

El contendor tendrá un ancho de 650px, con un margen izquierdo y derecho auto.

```
.main-nav img {
  width: 60%;
}
```

La imagen del contenedor tendrá un ancho al 60%.

```
.main-nav>ul {
  display: flex;
  flex-direction: row;
  justify-content: space-around;
  align-items: center;
}
```

Para la lista desordenada, establecemos un display flex, cuya dirección será row, y el espacio entre cada elemento será con espacio alrededor, centramos por último los elementos en el contenedor.

- /* FOOTER */

```
.links-container {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  align-items: center;
}
```

Para los elementos de este contenedor fijamos un display flex, dirección row, el justificado será espacio entre cada elemento y centraremos los elementos.

```
.links-group>ul {
  display: flex;
  flex-direction: row;
  align-items: center;
}
```

Para los elementos de este contenedor fijamos un display flex, dirección row y centraremos los elementos.

```
.links-social ul {
  display: flex;
  flex-direction: row;
  align-items: center;
}
```

Para los elementos de este contenedor fijamos un display flex, dirección row y centraremos los elementos.

```
.links-social ul>li {
  padding: 0 10px 0 10px;
}
```

Fijamos para cada elemento de la lista de esta clase un padding derecho e izquierdo de 10px.

```
.links-group ul>li {
  padding: 0 10px 0 10px;
}
```

Fijamos para cada elemento de la lista de esta clase un padding derecho e izquierdo de 10px.

- /** INDEX **/

```
.index-article-1 {
  max-width: 650px;
  margin-left: auto;
  margin-right: auto;
  padding: 50px 0 20px 0;
}
```

El contendor tendrá un ancho de 650px, con un margen izquierdo y derecho auto y un padding superior de 50px y otro inferior de 20px.

```
.index-article-2 {
  padding: 100px 0 100px 0;
}
```

Para este contenedor fijamos un padding superior e inferior de 100px.

```
.index-img-container {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  max-width: 1924px;
  margin-left: auto;
  margin-right: auto;
}
```

Respecto al contenedor de las imágenes, fijamos un display flex, que tenga wrap, justificamos el contenido el centro, con un ancho máximo 1924 px y un margen izquierdo y derecho auto.

```
.index-img-container a {
  flex-shrink: 0;
  flex-basis: 300px;
  display: flex;
  align-items: center;
  justify-content: center;
  padding: 1px;
}
```

Para cada enlace establecemos un flex shrink a 0, con un flex basis de 300px, un display flex, centramos los ítems al centro y también los justificamos al centro con un padding de 1px.

- /** OBJETIVES **/

```
.objetives-content {
  max-width: 850px;
  margin-left: auto;
  margin-right: auto;
  padding: 50px 0 20px 0;
}
```

El contendor tendrá un ancho de 850px, con un margen izquierdo y derecho auto y un padding superior de 50px y otro inferior de 20px.

```
.objetives-content .breadcrumb {
  padding-left: 20px;
  text-align: start;
}
```

El contenedor del breadcrumb tendrán un padding izquierdo de 20px y el alineado será inicial.

```
.unpair {
  display: flex;
  justify-content: flex-start;
}
```

Cada clase impar tendrá un display flex y una justificación inicial.

```
.pair {
  display: flex;
  justify-content: flex-end;
}
```

Cada clase par tendrá un display flex y una justificación final.

```
section img {
  padding: 50px 0px 50px 0;
}
```

Cada imagen dentro de cada section, tendrá un padding superior e inferior de 50px.

```
.img-container {
  flex-grow: 0;
  width: 33%;
  height: auto;
  padding: 20px;
  align-self: center;
}
```

El contenedor de cada imagen tendrá un flex grow a cero, con un ancho del 33%m una altura auto, un padding general de 20px y un alineado al centro.

```
.text-container {
  flex-grow: 0;
  width: 33%;
  height: auto;
  padding: 20px;
  align-self: center;
}
```

El contenedor de cada texto tendrá un flex grow a cero, con un ancho del 33%m una altura auto, un padding general de 20px y un alineado al centro.

```
.text-container h2 {
  padding-top: 0px;
}
```

Cada h2 de cada contenedor de texto tendrá un padding superior cero.

```
.text-container p {
  padding: 10px 0 15px 0;
  text-align: justify;
}
```

Cada párrafo de cada contenedor de texto tendrá un padding superior de 10px e inferior de 15px, el texto estará justificado.

```
.text-container a {
  margin-top: 20px;
}
```

Cada enlace de cada contenedor de texto tendrá un margin superior de 20px.

- /** OBJETIVE-1 **/

```
.objetive-content {
  max-width: 850px;
  margin-left: auto;
  margin-right: auto;
  padding: 50px 0 20px 0;
}
```

El contendor tendrá un ancho de 850px, con un margen izquierdo y derecho auto y un padding superior de 50px y otro inferior de 20px.

```
.objetive-content .breadcrumb {
  text-align: start;
  padding-left: 20px;
}
```

El contenedor del breadcrumb tendrán un padding izquierdo de 20px y el alineado será inicial.

```
.objetive-subcontent {
  display: flex;
  flex-wrap: wrap;
  padding-top: 30px;
  padding-bottom: 30px;
}
```

El contendor tendrá un display flex con wrap y con un padding superior e inferior de 30px.

```
.objetive-submenu {
  width: 30%;
  padding-left: 20px;
}
```

El ancho será de 30% con un padding izquierdo de 20px.

```
.objetive-container {
  width: 70%;
  padding-left: 100px;
}
```

El ancho será de 70% con un padding izquierdo de 100px.

```
.objetive-submenu>ul {
  text-align: start;
  padding: 0;
}
```

La lista desordenada del contenedor tendrá un alineado inicial con un padding a cero.

```
.objetive-submenu>ul>li {
  text-align: start;
  padding: 10px 0px 10px 0;
}
```

La elemento de la lista desordenada del contenedor tendrá un alineado inicial con un padding superior e inferior de 10px.

```
.objetive-submenu img {
  padding-left: 0;
  padding-right: 40px;
}
```

La imagen del contenedor tendrá un padding izquierdo a 0 y un padding derecho a 40px.

```
.objetive-container h1 {
  text-align: start;
}
```

El título principal estará alineado al inicio.

- /* CONTACT */

```
.contact-content {
  max-width: 850px;
  padding: 50px 0 20px 0;
  margin-left: auto;
  margin-right: auto;
}
```

El contendor tendrá un ancho de 850px, con un margen izquierdo y derecho auto y un padding superior de 50px y otro inferior de 20px.

```
.contact-content .breadcrumb {
  text-align: start;
  padding-left: 20px;
}
```

El contenedor del breadcrumb tendrán un padding izquierdo de 20px y el alineado será inicial.

```
.form-container {
  width: 70%;
  margin-left: auto;
  margin-right: auto;
  padding: 50px 0 70px 0;
}
```

El contenedor del formulario tendrá un ancho al 70%, margen izquierdo y derecho auto y un padding superior a 50px e inferior a 70px.

- @media (min-width: 950px): a partir de un ancho de 950 px las reglas genéricas cambiarán a las siguientes:

    - /* NAV */

```
.main-nav {
  width: 950px;
  margin-left: auto;
  margin-right: auto;
}
```

El contendor tendrá un ancho de 950px, con un margen izquierdo y derecho auto.

5. Diseño gráfico y maquetación.

Respecto a las reglas de diseño gráfico y maquetación se han seguido los siguientes wireframes en baja fidelidad para la colocación de cada elemento, que anteriormente hemos ido situando con los clases y selectores CSS.

En última instancia ilustraremos captura por captura el diseño final tanto la adaptación para dispositivos móviles como la adaptación para el resto de formatos para otros dispositivos.

Expondremos en primer lugar los wireframes en baja fidelidad para dispositivos móviles:

index.html (phone)

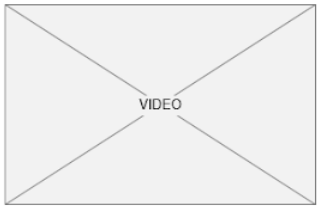
Header

- item 1
- item 2
- item 3
- item 4
- item 5

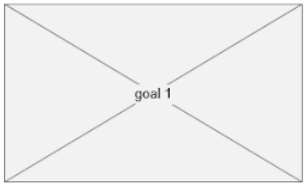
Main

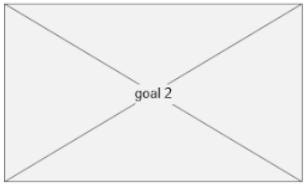
Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet. Proin gravida dolor sit amet lacus accumsan et viverra justo commodo. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet. Proin gravida dolor sit amet lacus accumsan et viverra justo commodo.

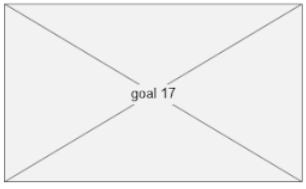


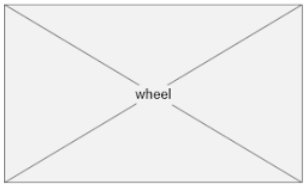
Titular





■■■





Footer

- item 1
- item 2
- item 3

- item 1
- item 2
- item 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod
Lorem ipsum dolor sit amet, consectetur adipiscing elit.

objetivos.html (phone)

Header

- item 1
- item 2
- item 3
- item 4
- item 5

Main

item 1 / item2

Titular

pictograma goal 1

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

pictograma goal 2

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

pictograma goal 3

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

pictograma goal 4

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

■■■

pictograma goal 17

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

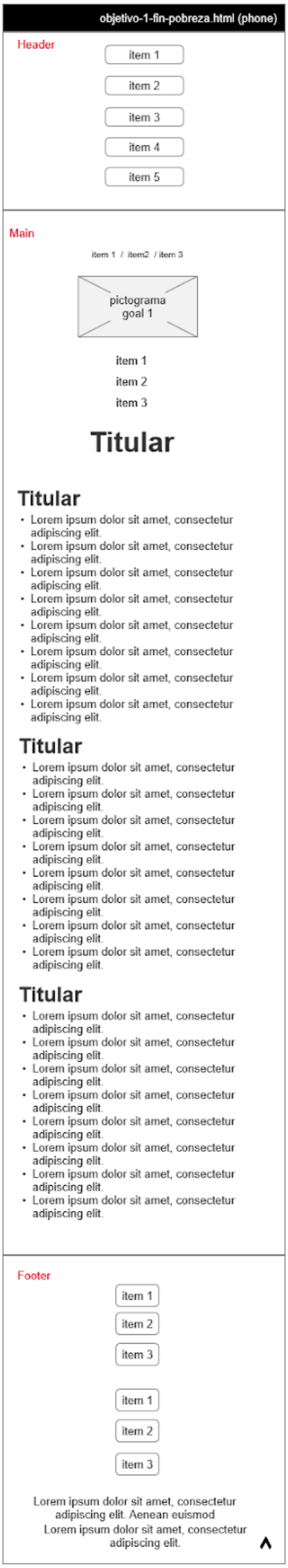
+ link

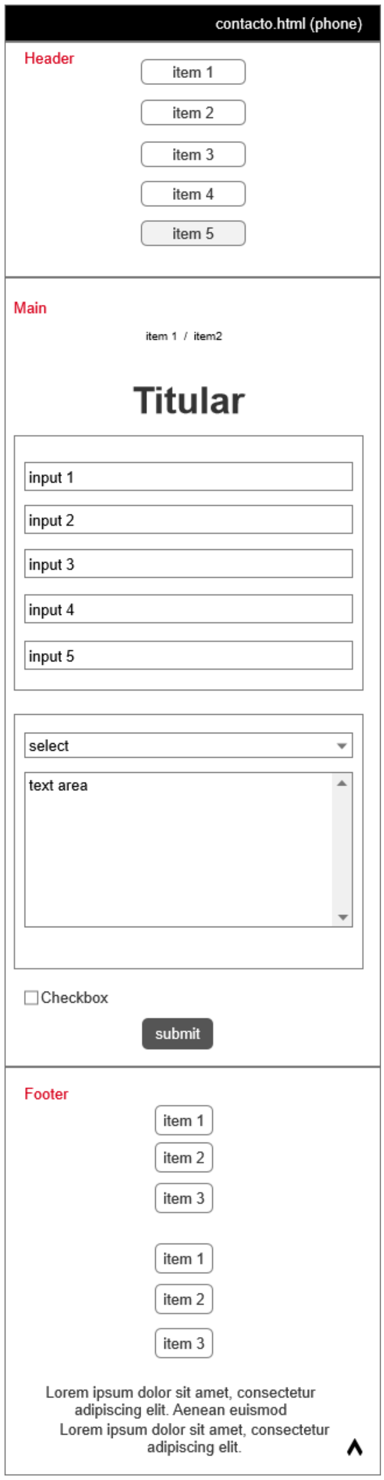
Footer

- item 1
- item 2
- item 3

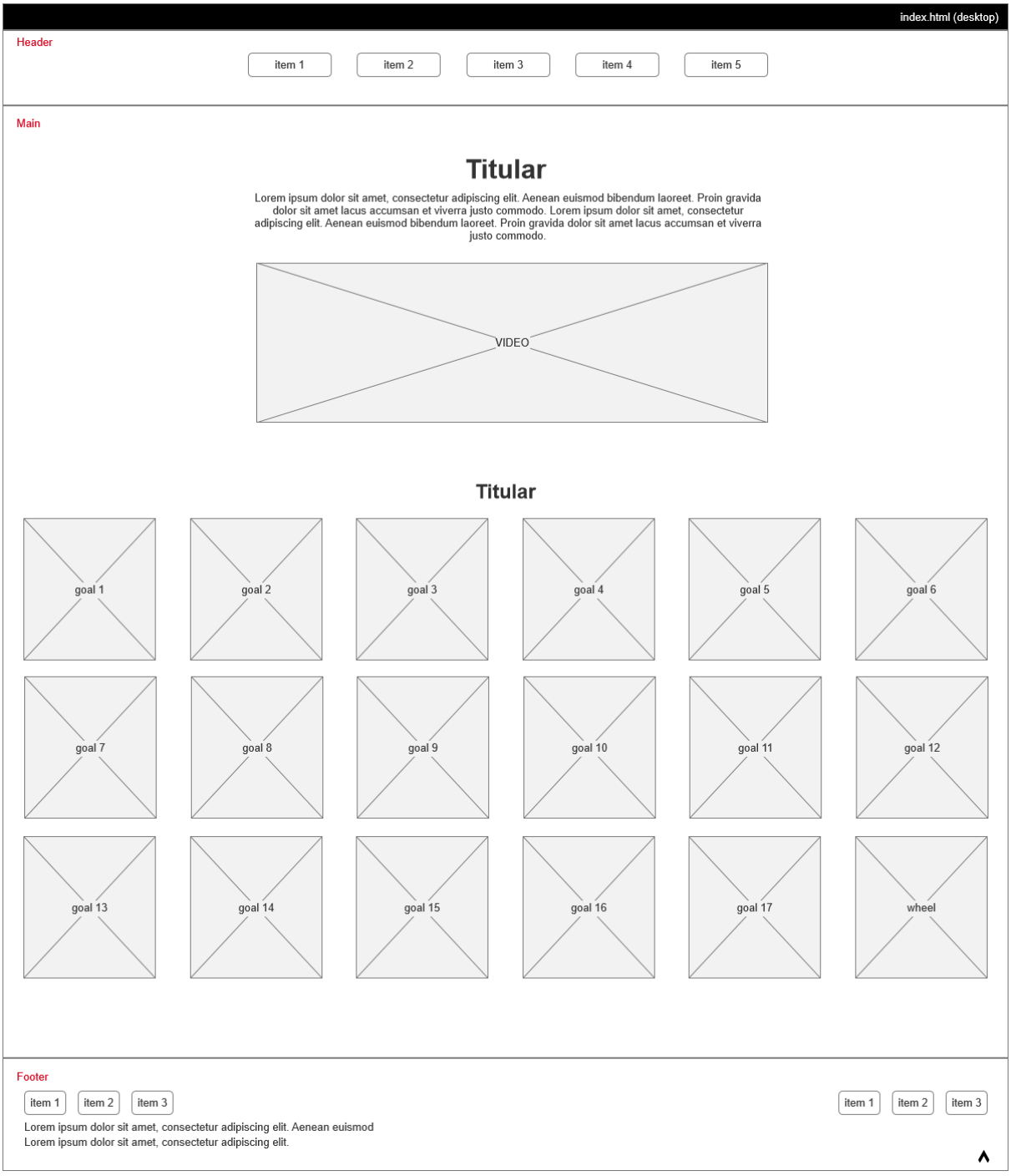
- item 1
- item 2
- item 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod
Lorem ipsum dolor sit amet, consectetur adipiscing elit.


objetivo-1-fin-pobreza.html (phone)
Header
item 1
item 2
item 3
item 4
item 5
Main
item 1 / item2 / item 3
pictograma goal 1
item 1
item 2
item 3
Titular
Titular
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Titular
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Titular
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Footer
item 1
item 2
item 3
item 1
item 2
item 3
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod
Lorem ipsum dolor sit amet, consectetur adipiscing elit.


contacto.html (phone)
Header
item 1
item 2
item 3
item 4
item 5
Main
item 1 / item2
Titular
input 1
input 2
input 3
input 4
input 5
select
text area
Checkbox
submit
Footer
item 1
item 2
item 3
item 1
item 2
item 3
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod
Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Wireframes en baja fidelidad para el resto de dispositivos:



objetivos.html (desktop)

Header

item 1 | item 2 | item 3 | item 4 | item 5

Main

item 1 / item2

Titular

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

■■■

Titular

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.

+ link

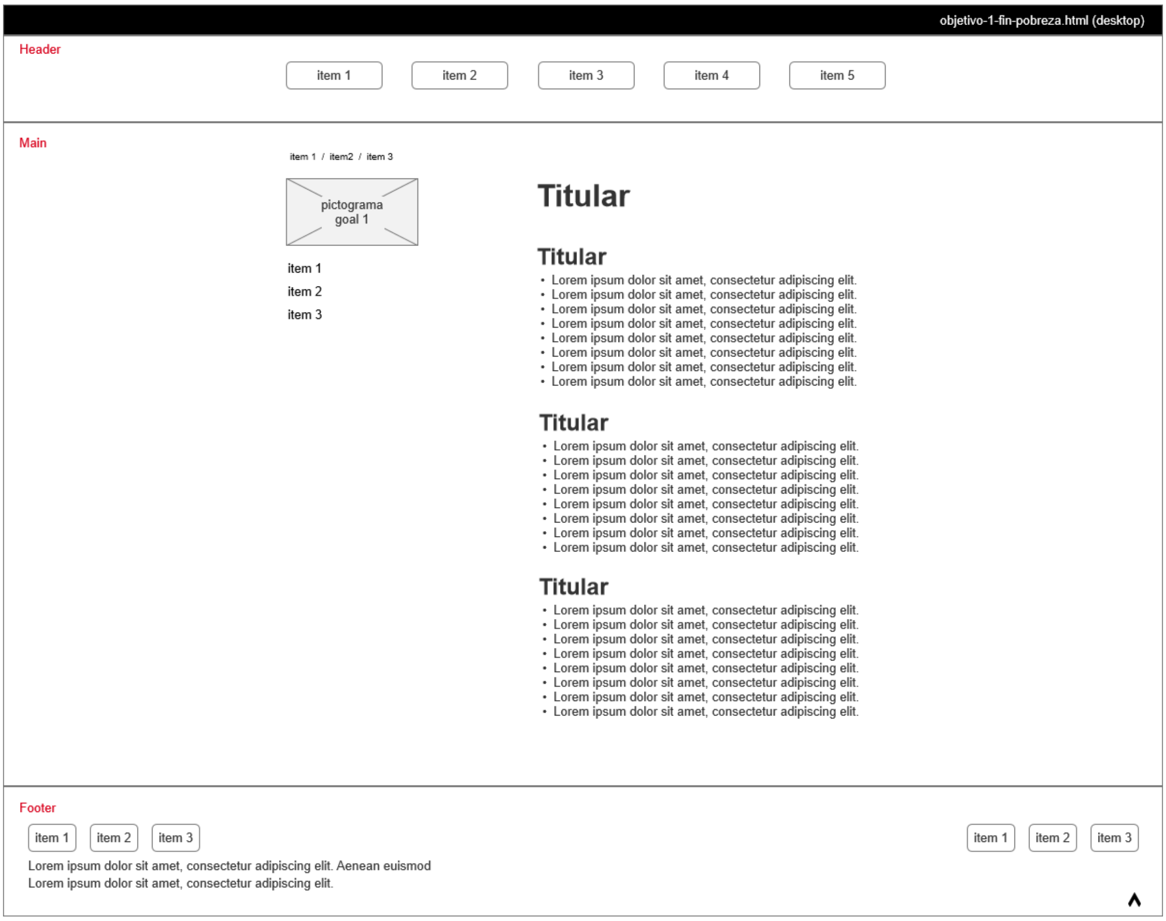
Footer

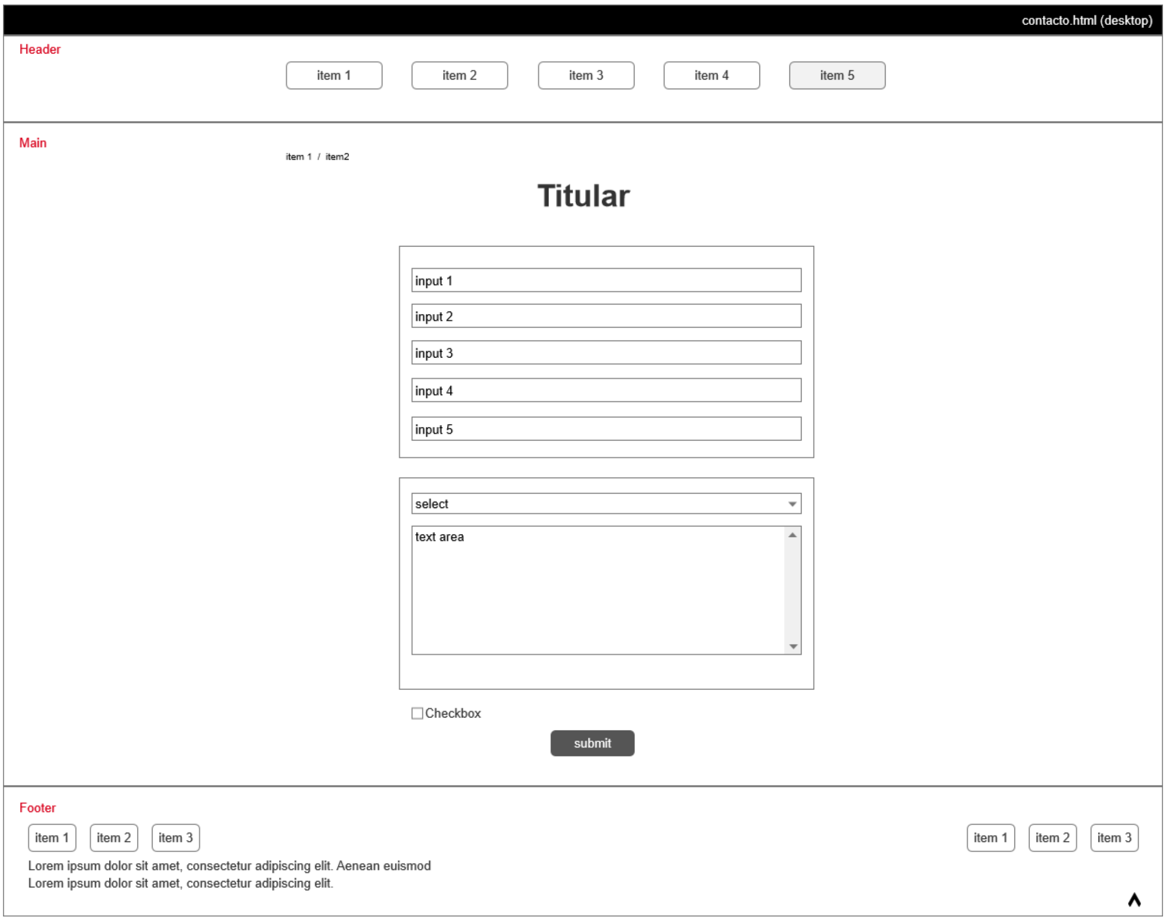
item 1 | item 2 | item 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod
Lorem ipsum dolor sit amet, consectetur adipiscing elit.

item 1 | item 2 | item 3

^


objetivo-1-fin-pobreza.html (desktop)
Header
item 1
item 2
item 3
item 4
item 5
Main
item 1 / item2 / item 3
pictograma goal 1
item 1
item 2
item 3
Titular
Titular
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Titular
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Titular
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Footer
item 1
item 2
item 3
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
item 1
item 2
item 3


contacto.html (desktop)
Header
item 1
item 2
item 3
item 4
item 5
Main
item 1 / item2
Titular
input 1
input 2
input 3
input 4
input 5
select
text area
Checkbox
submit
Footer
item 1
item 2
item 3
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
item 1
item 2
item 3

Por último mostraremos las capturas del resultado final en dispositivos móviles y en el resto de dispositivos:

index.html:

objetivos.html

17 objetivos para transformar el mundo

1. Fin de la pobreza

Más de 700 millones de personas, el 10% de la población mundial, aún viven en

básicas tales como la salud, la educación y el acceso al agua y al saneamiento, por citar algunas...

+ sobre Objetivo 1

2. Hambre cero

Es el momento de repensar cómo crecemos, compartimos y consumimos nuestros alimentos. Si se hace bien, la agricultura, la silvicultura y la pesca pueden proporcionar alimentos nutritivos para todos y generar ingresos decentes...

+ sobre Objetivo 2

17. Alianzas para lograr los objetivos

Un programa exitoso de desarrollo sostenible requiere de alianzas entre los gobiernos, el sector privado y la sociedad civil...

+ sobre Objetivo 17

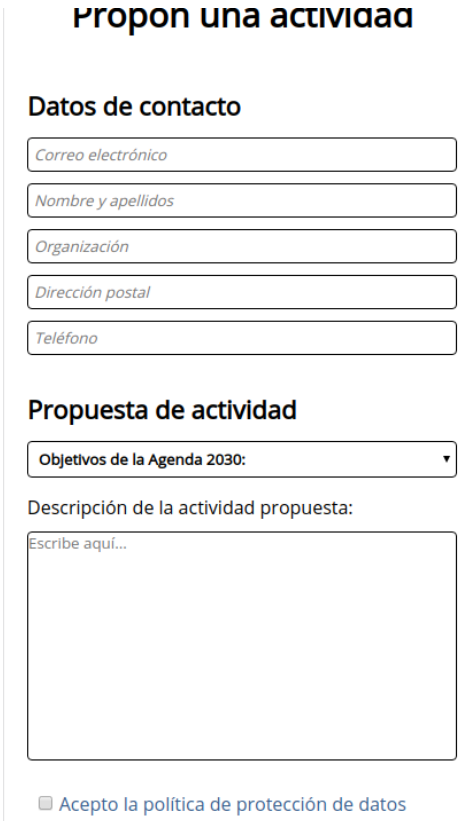
objetivo-1-fin-pobreza.html

Objetivo 1. Fin de la pobreza

Datos destacables

- Unos 783 millones de personas vive por debajo del umbral de pobreza internacional, con 1,90 dólares diarios.
- En 2016, menos del 10% de los trabajadores de todo el mundo vivían con sus familias con menos de 1,90 dólares diarios por persona.
- En el mundo existen 122 mujeres, entre los 25 y 34 años, viviendo en extrema pobreza por cada 100 hombres del mismo grupo de edades.
- La mayoría de las personas que viven por debajo del umbral de pobreza viven en dos regiones: Asia meridional y África subsahariana.
- Las altas tasas de pobreza se encuentran a menudo en los países pequeños, frágiles y afectados por conflictos.
- Uno de cada cuatro niños menores de cinco años, en todo el mundo,

contacto.html



index.html


Objetivos
Actualidad
Actividades
Contacto
Objetivos de desarrollo sostenible
En 2015, la ONU aprobó la Agenda 2030 sobre el Desarrollo Sostenible, una oportunidad para que los países y sus sociedades emprendan un nuevo camino con el que mejorar la vida de todos, sin dejar a nadie atrás.
Do you know all 17 SDGs?
SUSTAINABLE DEVELOPMENT GOALS
17 objetivos para transformar el mundo
7 ENERGIA ASSEQUIBLE I NO CONTAMINANT
8 TREBALL DECENT I CREIXEMENT ECONÒMIC
9 INDÚSTRIA, INNOVACIÓ I INFRAESTRUCTURA
10 REDUCCIÓ DE LES DESIGUALTATS
11 CIUTATS I COMUNITATS SOSTENIBLES
12 PRODUCCIÓ I CONSUM RESPONSABLES
13 ACCIÓ PEL CLIMA
14 VIDA SUBMARINA
15 VIDA D'ECOSISTEMES TERRESTRES
16 PAU, JUSTÍCIA I INSTITUCIONS SÒLIDES
17 ALIANCES PER A ASSOLIR ELS OBJECTIUS
Sobre nosotros
Aviso legal
Política de privacidad
© 2019-2020 Aníbal Santos Gómez. Trabajo realizado para la asignatura HTML y CSS. Primer semestre del curso 2019-2020

objetivos.html

17 objetivos para transformar el mundo

1. Fin de la pobreza

Más de 700 millones de personas, el 10% de la población mundial, aún viven en situación de pobreza extrema y luchan por satisfacer las necesidades más básicas tales como la salud, la educación y el acceso al agua y al saneamiento, por citar algunas...

+ sobre Objetivo 1

2. Hambre cero

Es el momento de repensar cómo crecemos, compartimos y consumimos nuestros alimentos. Si se hace bien, la agricultura, la silvicultura y la pesca pueden proporcionar

16. Paz, justicia e instituciones sólidas

Las amenazas de homicidio intencional, la violencia contra los niños, la trata de personas y la violencia sexual, son temas importantes que deben ser tratados para crear sociedades pacíficas e inclusivas...

+ sobre Objetivo 16

17. Alianzas para lograr los objetivos

Un programa exitoso de desarrollo sostenible requiere de alianzas entre los gobiernos, el sector privado y la sociedad civil...

+ sobre Objetivo 17

objetivo-1-fin-pobreza.html

Objetivo 1. Fin de la pobreza

Datos destacables

- Unos 783 millones de personas vive por debajo del umbral de pobreza internacional, con 1,90 dólares diarios.
- En 2016, menos del 10% de los trabajadores de todo el mundo vivían con sus familias con menos de 1,90 dólares diarios por persona.
- En el mundo existen 122 mujeres, entre los 25 y 34 años, viviendo en extrema pobreza por cada 100 hombres del mismo grupo de edades.
- La mayoría de las personas que viven por debajo del umbral de pobreza viven en dos regiones: Asia meridional y África subsahariana.
- Las altas tasas de pobreza se encuentran a menudo en los países pequeños, frágiles y afectados por conflictos.
- Uno de cada cuatro niños menores de cinco años, en todo el mundo, tiene una estatura inadecuada para su edad.
- En 2016, solo el 45% de la población mundial estaba

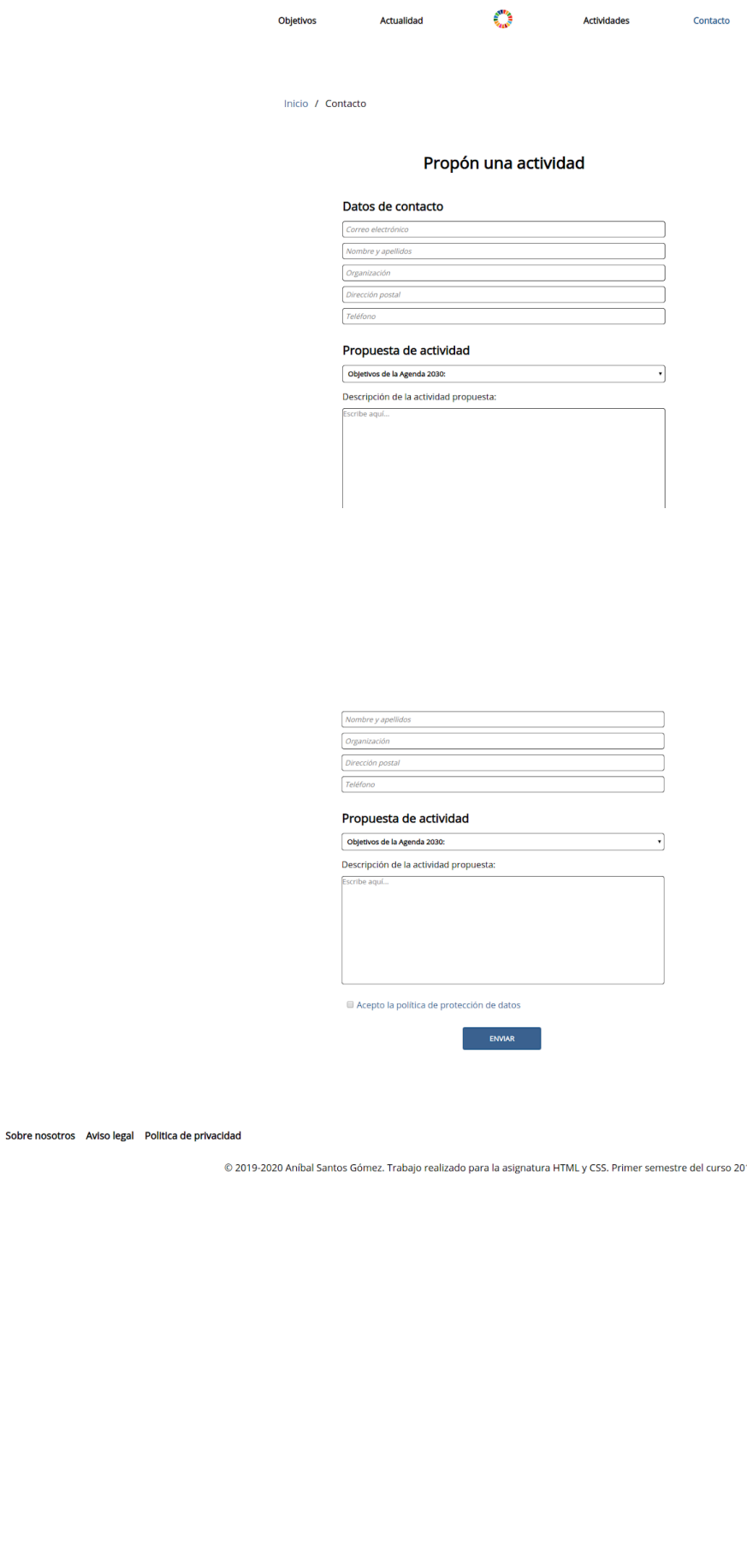
menos adelantados, para poner en práctica programas y políticas encaminados a poner fin a la pobreza en todas sus dimensiones.

7. Crear marcos normativos sólidos en los planos nacional, regional e internacional, sobre la base de estrategias de desarrollo en favor de los pobres que tengan en cuenta las cuestiones de género, a fin de apoyar la inversión acelerada en medidas para erradicar la pobreza.

Enlaces

- Programa de las Naciones Unidas para el Desarrollo
- Banco Mundial
- Fondo de las Naciones Unidas para la Infancia
- Fondo Monetario Internacional
- Pacto mundial
- UNESCO
- Oficina de las Naciones Unidas para la Reducción del Riesgo de Desastres

contacto.html



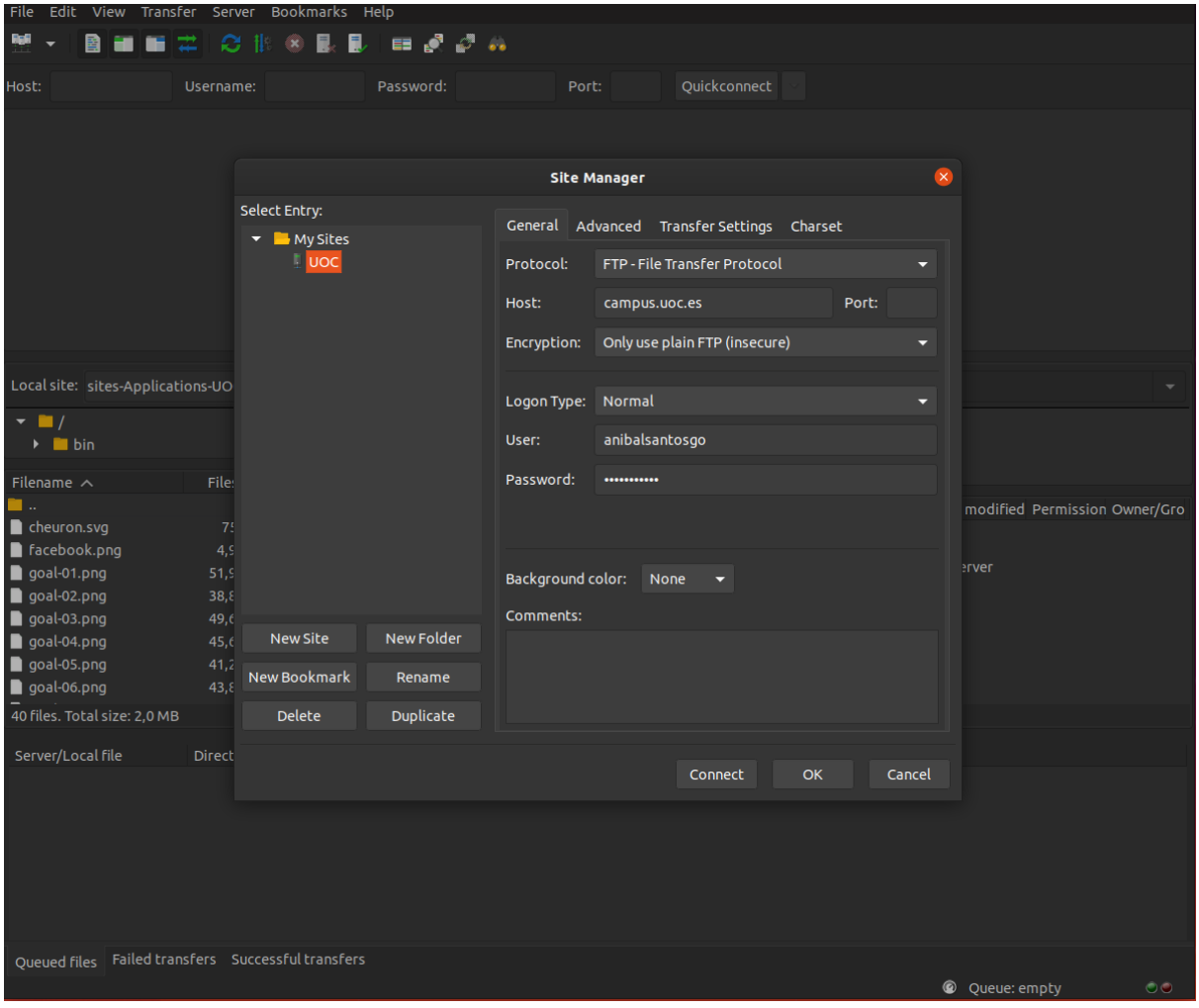
6. Despliegue.

Para finalizar este informe explicamos como realizar el despliegue o deploy del sitio web creado. Debemos tener en cuenta que el sitio web que hemos creado es estático.

1. En primer lugar descargamos la aplicación opensource Filezilla, disponible para Windows, Mac y distribuciones de Linux:

   https://filezilla-project.org/

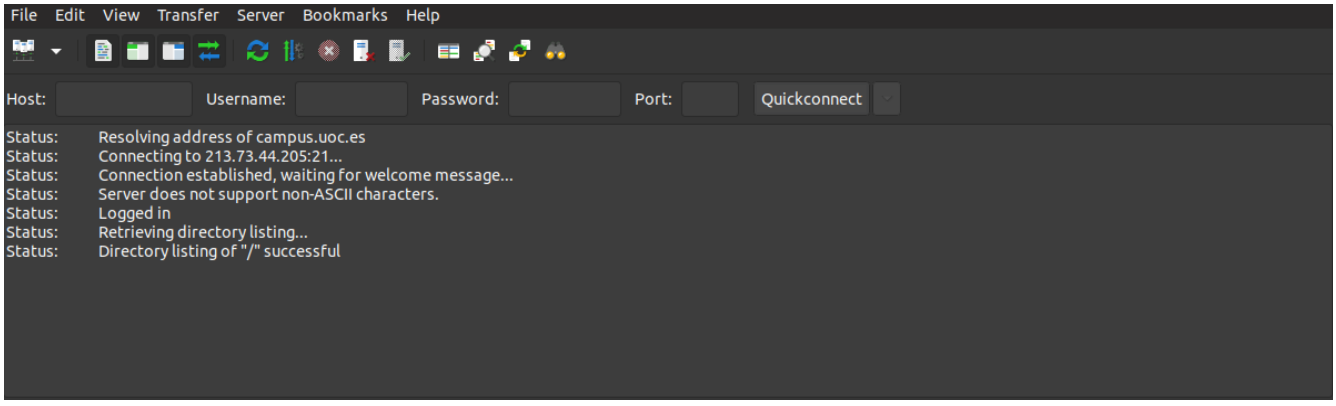
2. Una vez descargada y abierta, procedemos a configurar el servidor a través de File > Site Manager:



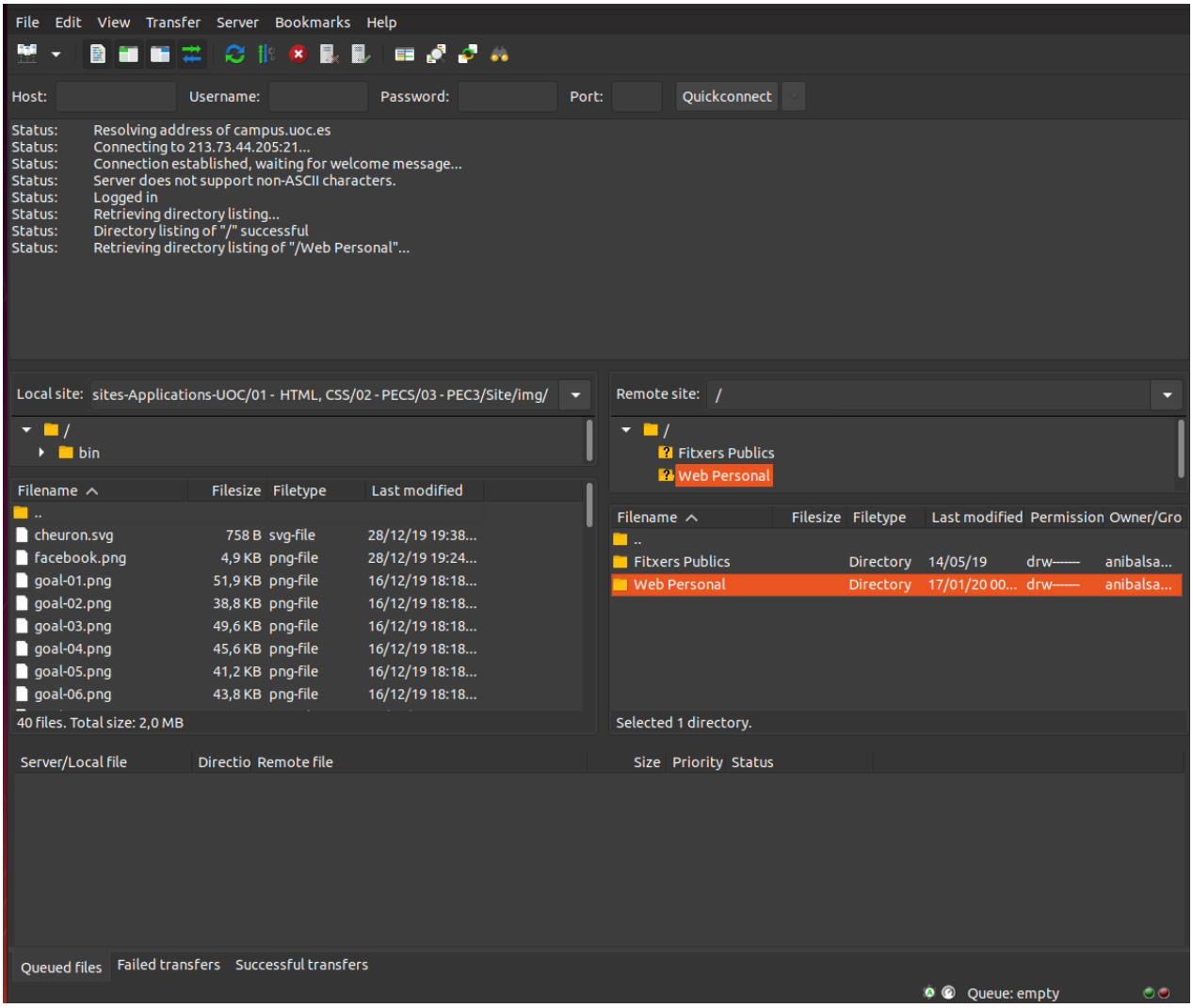
Introducimos la siguiente configuración:

- Protocol: FTP - File Transfer Protocol
- Host: campus.uoc.es
- Encryption: Only use plain FTP
- Logon type: Normal
- User: nuestro usuario en la UOC.
- Password: nuestro password en la UOC

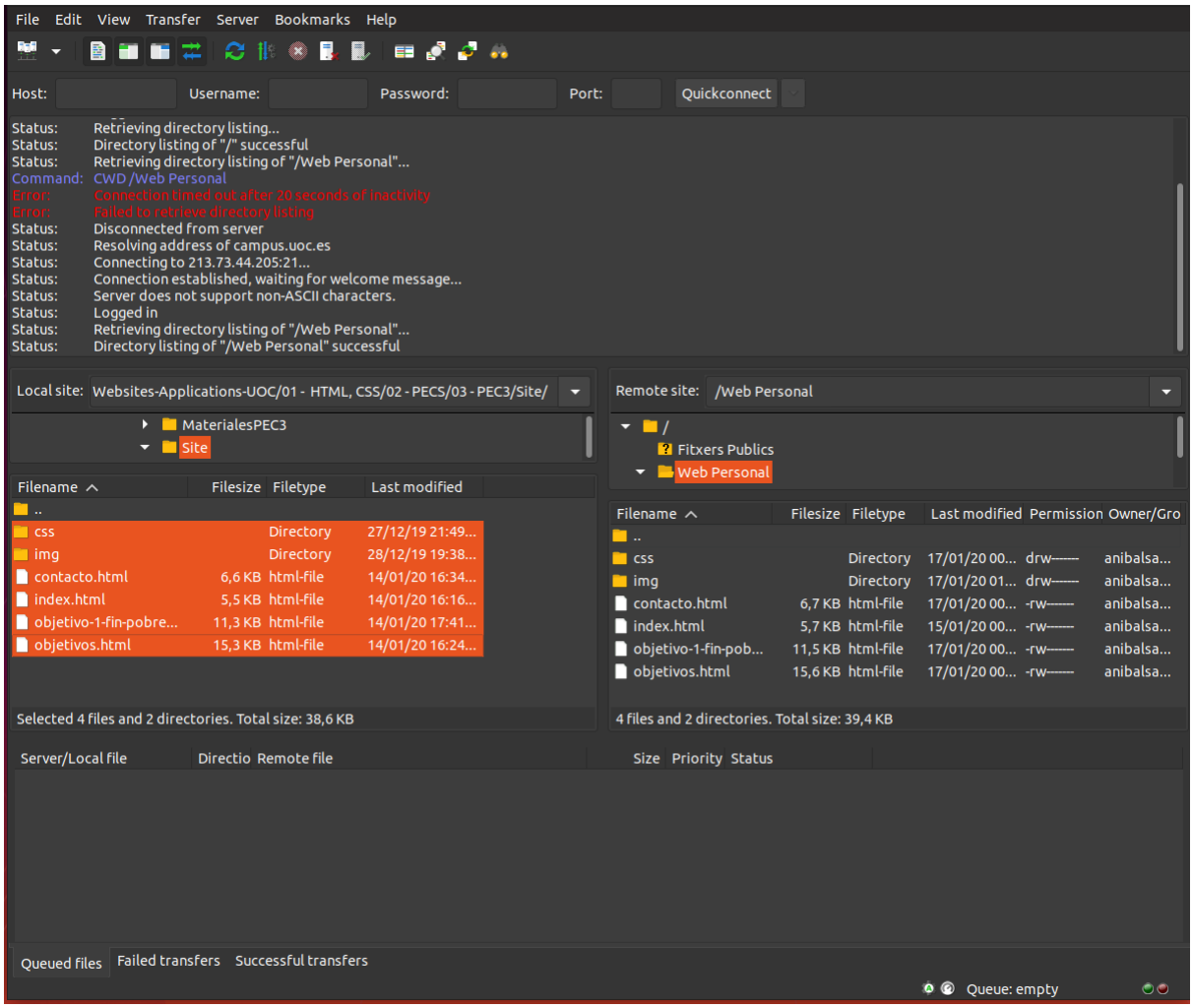
Le damos a conectar y esperamos a que se conecte recibiendo el siguiente mensaje de confirmación:



3. Conectados al servidor procederemos a transferir los archivos y directorios del proyecto a nuestro servidor. Para ello seleccionaremos el directorio “Web Personal” dentro del servidor de la UOC:



4. Seleccionada la carpeta, en la parte izquierda tendremos nuestros directorios en local de nuestro disco duro. Seleccionaremos todos los archivos que queremos transferir a la carpeta “Web Personal” del servidor de alumno de la UOC y presionando en el botón derecho del ratón, le daremos a la opción de upload:



5. Por último comprobaremos que todos los archivos han sido cargados en el servidor de forma satisfactoria.

6. Para aplicar cualquier cambio sobre el proyecto, necesitaremos volver a subir los archivos modificados del proyecto.

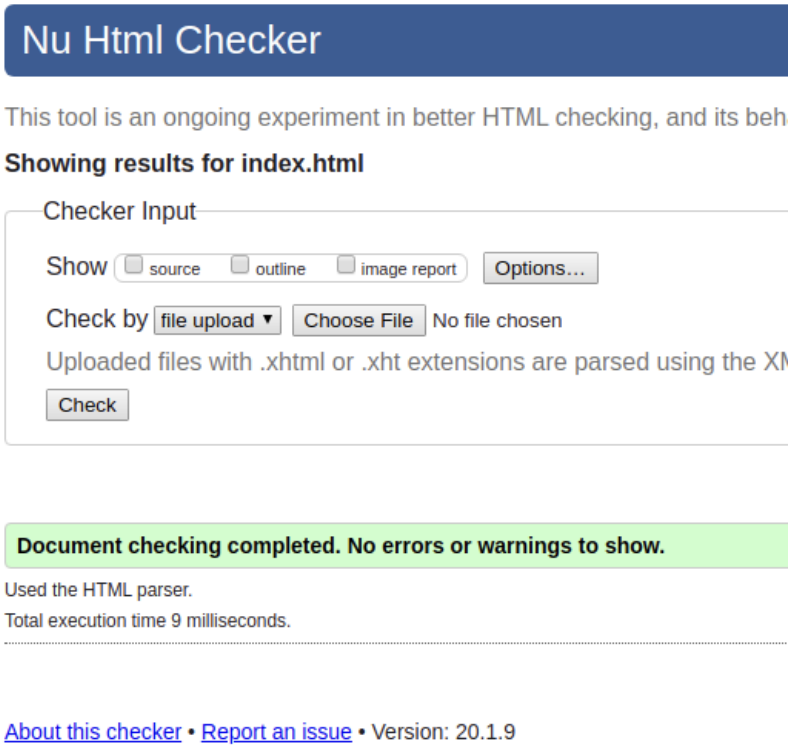
7. Por último, para acceder a la publicación del proyecto, accederemos en nuestro navegador a la siguiente dirección (dependerá de cada usuario):

   http://cv.uoc.edu/~anibalsantosgo/index.html

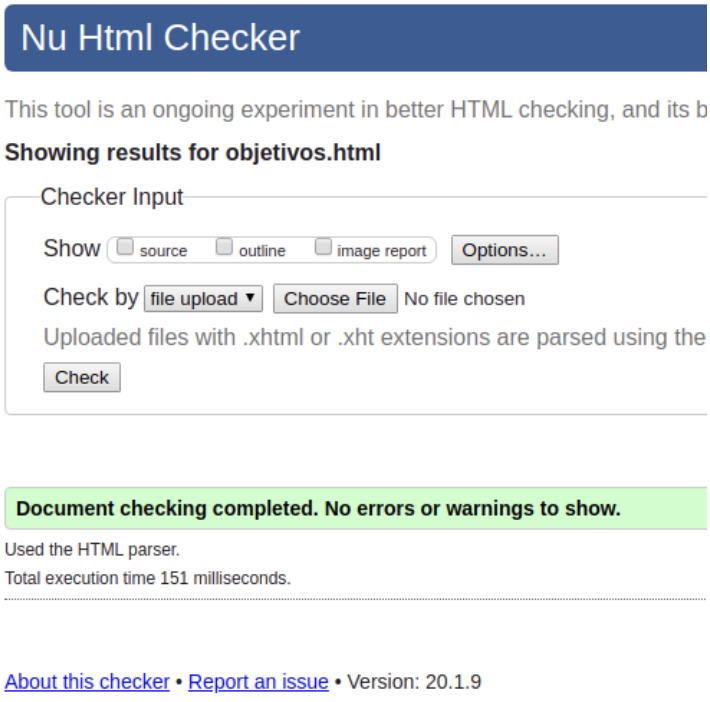
PARTE 3 - VALIDACIÓN

Validación https://validator.w3.org/

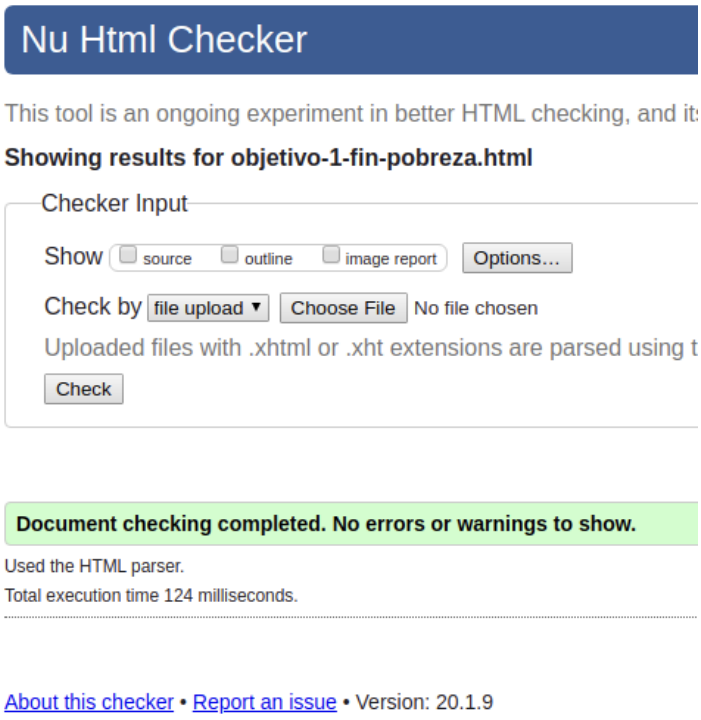
1. **index.html**



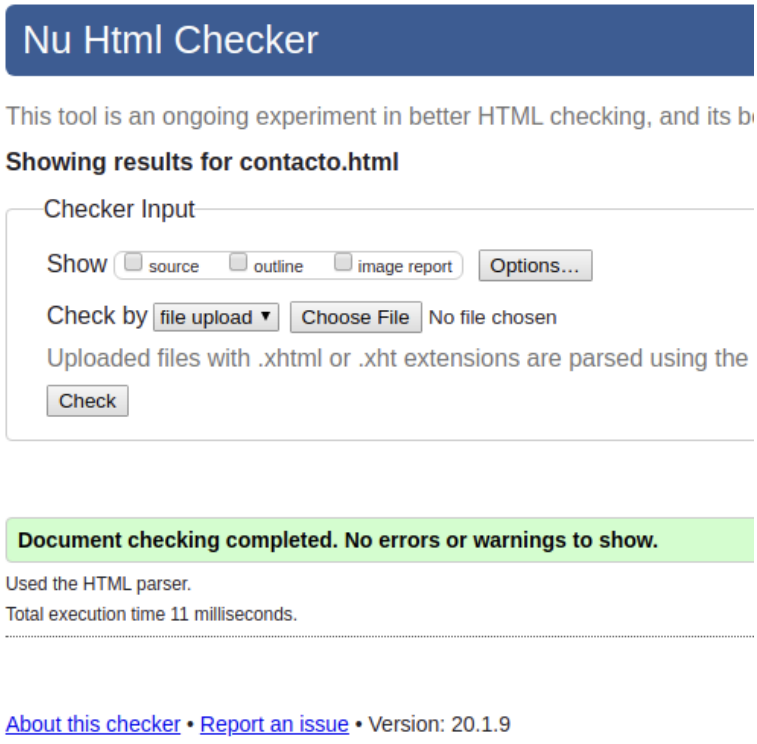
2. **Objetivos.html**



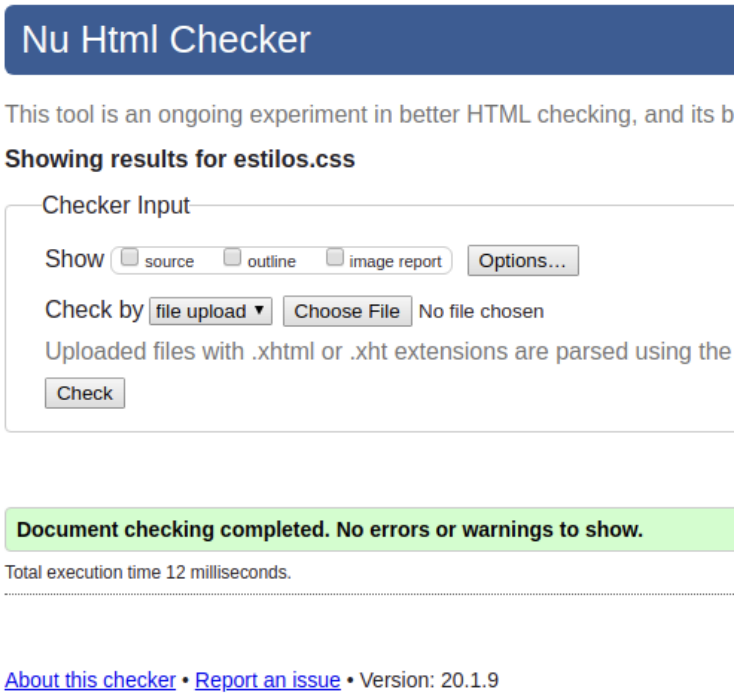
3. **objetivo-1-fin-pobreza.html**



4. **contacto.html**



5. estilos.css



Resultados del Validador CSS del W3C para estilos.css (CSS versión 3 + SVG)

Disculpas! Hemos encontrado las siguientes errores (24)

URI : estilos.css

13	:root	Error de análisis sintáctico `--text: #000000;`
14	:root	Error de análisis sintáctico `--text-hover: #ffffff;`
15	:root	Error de análisis sintáctico `--current: #306291;`
16	:root	Error de análisis sintáctico `}`
47	.main-nav a	Propiedad no válida : color Error de análisis sintáctico `var(--text)`
51	.main-nav a:hover	Propiedad no válida : color Error de análisis sintáctico `var(--text-hover)`
62	.main-nav > ul > li > .current	Propiedad no válida : color Error de análisis sintáctico `var(--current)`
74	.main-footer a	Propiedad no válida : color Error de análisis sintáctico `var(--text)`
87	.main-footer a:hover	Propiedad no válida : color Error de análisis sintáctico `var(--text-hover)`
130	ul.breadcrumb li + li:before	Propiedad no válida : color Error de análisis sintáctico `var(--text)`
135	ul.breadcrumb li a	Propiedad no válida : color Error de análisis sintáctico `var(--current)`
140	ul.breadcrumb li a:hover	Propiedad no válida : color Error de análisis sintáctico `var(--current)`
212	.text-container a	Propiedad no válida : color Error de análisis sintáctico `var(--text)`
217	.text-container a:hover	Propiedad no válida : color Error de análisis sintáctico `var(--current)`
245	.objetive-submenu > ul > li > a	Propiedad no válida : color Error de análisis sintáctico `var(--current)`
274	.objetive-container > section > ul > li > a	Propiedad no válida : color Error de análisis sintáctico `var(--current)`
303	.input-container a	Propiedad no válida : color Error de análisis sintáctico `var(--current)`
311	input[type="text"]	Propiedad no válida : background-color Error de análisis sintáctico `var(--text-hover)`
312	input[type="text"]	Propiedad no válida : border Error de análisis sintáctico `var(--text)`
329	input[type="submit"]	Propiedad no válida : background-color Error de análisis sintáctico `var(--current)`
330	input[type="submit"]	Propiedad no válida : color Error de análisis sintáctico `var(--text-hover)`
349	select	Propiedad no válida : background-color Error de análisis sintáctico `var(--text-hover)`
351	select	Propiedad no válida : border Error de análisis sintáctico `var(--text)`
361	textarea	Propiedad no válida : border Error de análisis sintáctico `var(--text)`

Encontramos estos fallos por las pseudoclases de estilo de color pseudo-clase :root