

Diseño de un sitio web

Desarrollo de Aplicaciones Web

Autores:

Pazos Parada, Pablo

Raposeiras Canabal, David Manuel

Índice

Índice.....	2
Diseño.....	3
¿Qué es trustip?.....	3
Funcionamiento y apariencia en detalle.....	3
Inventario del contenido.....	5
Arquitectura de la información.....	6
Mapa de navegación.....	7
Interfaces.....	8
Paleta de Colores.....	8
Welcome Page.....	8
Globe Page.....	9
Leaderboard Page.....	10
Contribute Page.....	11
Login Page.....	12
Storyboards.....	13
Consultar una IP/país.....	13
Registrarse en la web.....	14
Estructura de ficheros.....	15
Implementación.....	16
HTML.....	16
Header & Footer.....	17
Welcome Page.....	18
Globe Page.....	19
Leaderboard Page.....	21
Contributors Page.....	22
Login Page.....	23
Estructura de ficheros actualizada.....	24
CSS.....	25
Header & Footer.....	27
Welcome Page.....	28
Multicol.....	29
Flex-wrap.....	29
Aspecto final.....	30
Globe Page.....	31
Flex Container.....	31
Solapamiento de elementos.....	31
Responsividad para móvil.....	32
Aspecto final.....	33
Leaderboard Page.....	34
Float.....	34

Responsividad.....	35
Aspecto final.....	35
Contributors Page.....	36
Grid.....	36
Aspecto final.....	37
Login Page.....	38
Bootstrap.....	38
Aspecto final.....	40
Estructura de ficheros actualizada.....	40
JavaScript.....	41
Globe Page.....	41
Three.js.....	42
Tween.js.....	42
📁 libs/.....	42
JS orbit-controls.js.....	42
JS trackball-controls.js.....	43
JS perlin-noise.js.....	43
JS THREE.MeshLine.js.....	43
📁 scripts/globe/.....	43
JS config.js.....	43
JS shaders.js.....	44
JS app.js.....	44
JS setup.js.....	45
JS globe.js.....	48
JS points.js.....	49
JS marker.js.....	49
JS markers.js.....	50
JS utils.js.....	51
JS scripts/globe.....	51
Carga dinámica de scripts.....	51
preload().....	51
Despliegue de tarjetas.....	53
Eventos para el buscador.....	55
Leaderboard page.....	56
Contributors Page.....	58
Login page.....	60
Estructura de ficheros actualizada.....	64
Modificaciones.....	66

Nota: La página se puede visitar directamente en el siguiente enlace: trustip

Diseño

¿Qué es trustip?

trustip es una innovadora página web diseñada para ser el referente mundial en la consulta de seguridad y localización de direcciones IP. Nuestro objetivo es proporcionar a los usuarios una plataforma interactiva y accesible que les permita explorar datos clave sobre la actividad de IPs maliciosas en todo el mundo, ayudando a construir un entorno digital más seguro.

Una de las características más destacadas de *trustip* es su globo terráqueo en 3D, que permite a los usuarios interactuar con los países y visualizar en tiempo real el número de IPs maliciosas reportadas en cada región. Esta herramienta intuitiva facilita una comprensión visual del panorama global de seguridad cibernética.

Además, *trustip* cuenta con funcionalidades avanzadas como la búsqueda de IPs, que permite a los usuarios determinar el país de origen de una IP específica y si ha sido marcada como maliciosa. También incluye un ranking de países, donde se destaca a las naciones con menor cantidad de IPs maliciosas, promoviendo la transparencia y el reconocimiento de buenas prácticas en ciberseguridad.

Por último, *trustip* incorpora un sistema de usuarios que permite a las personas registrarse para reportar IPs sospechosas. Este enfoque colaborativo fomenta una comunidad activa de usuarios comprometidos en la identificación y prevención de amenazas digitales.

trustip no solo es una herramienta para consultar datos, sino un esfuerzo global para empoderar a los usuarios y promover una red más segura y confiable para todos.

Funcionamiento y apariencia en detalle

En cualquier página de *trustip* habrá siempre dos factores comunes; el encabezado y el pie de página. El encabezado será un menú con los siguientes elementos:

- El logo de *trustip*. Si se pulsa, nos llevará a la Welcome Page.
- Un símbolo de globo terráqueo. Si se pulsa nos llevará a la Globe Page.
- Un símbolo de ranking. Si se pulsa nos llevará a la Leaderboard Page.
- Un símbolo de reporte. Si se pulsa nos llevará a la Contribute Page.
- Un símbolo de usuario. Si se pulsa nos llevará a la Login Page.

El pie de página será una simple barra en la que figurará información de contacto para el usuario.

Al entrar en *trustip*, se mostrará la Welcome Page. En esta página encontraremos una pequeña descripción sobre el proyecto *trustip* y una llamada a la actuación del usuario que visite la web para que contribuya aportando información sobre IPs maliciosas sin registrar. Además, también figurará un apartado en el que aparecerán los autores de este proyecto a modo de mención. Al final de esta página habrá un botón que nos llevará a la Globe Page.

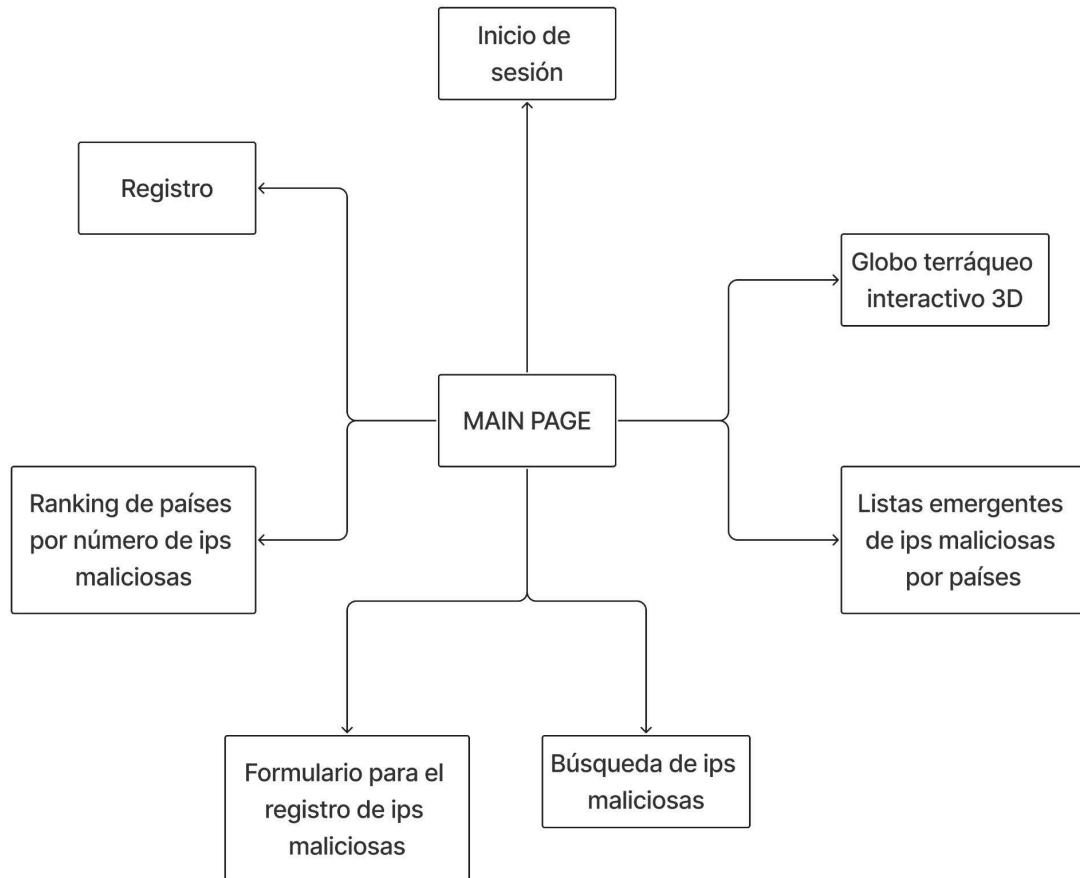
La Globe Page mostrará un globo terráqueo en 3D dividido en países. Cada país tendrá un color específico (del verde al rojo) que indicará la cantidad de IPs maliciosas que residen en el mismo (más rojo cuantas más haya) con respecto a la cantidad del resto. Al pulsar en cualquiera de estos países se desplegará un popup con información sobre el país y las IPs maliciosas registradas en su territorio. Cada IP estará linkeada de tal forma que al pulsar en ella se nos abrirá una web externa con más información sobre la misma. En esta misma página también encontraremos un menú de búsqueda, si en él buscamos una IP, se desplegará desde el globo terráqueo el popup del país al que corresponde y un pequeño mensaje indicando si la IP es maliciosa o no.

En la Leaderboard Page nos encontraremos un ranking de países ordenados de más seguro a menos según el número de IPs maliciosas detectadas en su territorio.

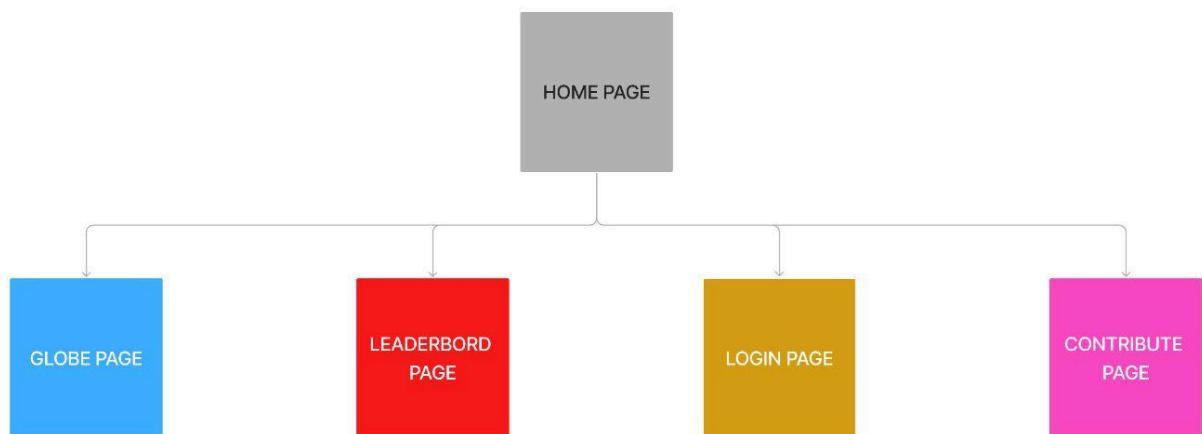
La Contribute Page servirá como entrada para contribuir al proyecto. En ella habrá un formulario que cubrir para enviarlo como reporte de una IP maliciosa. La IP figurada en el reporte será evaluada y añadida más tarde al globo terráqueo y contabilizada en el ranking si el reporte es acertado.

Finalmente, la Login Page servirá para iniciar sesión o crear una cuenta con la que poder reportar incidentes en la Contribute Page.

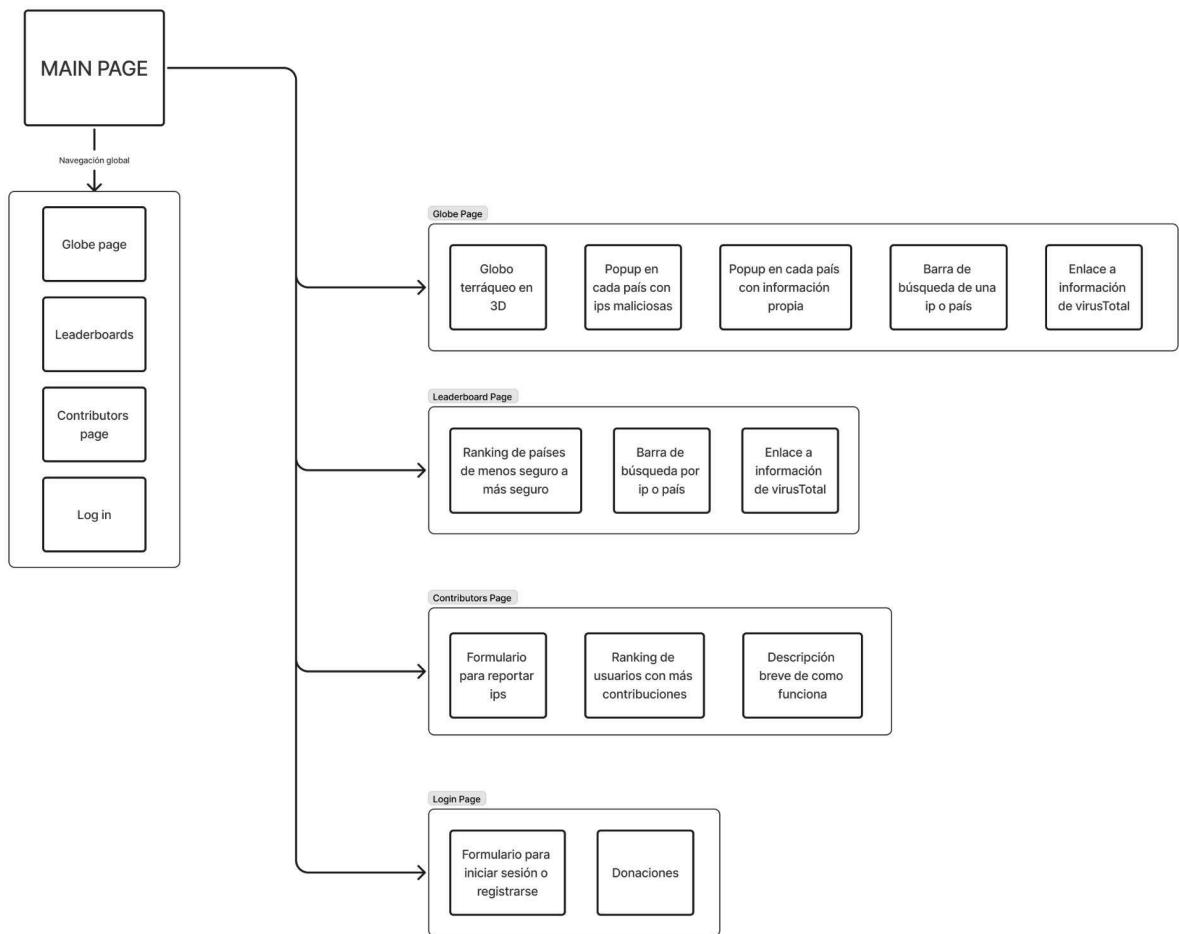
Inventario del contenido



Arquitectura de la información



Mapa de navegación



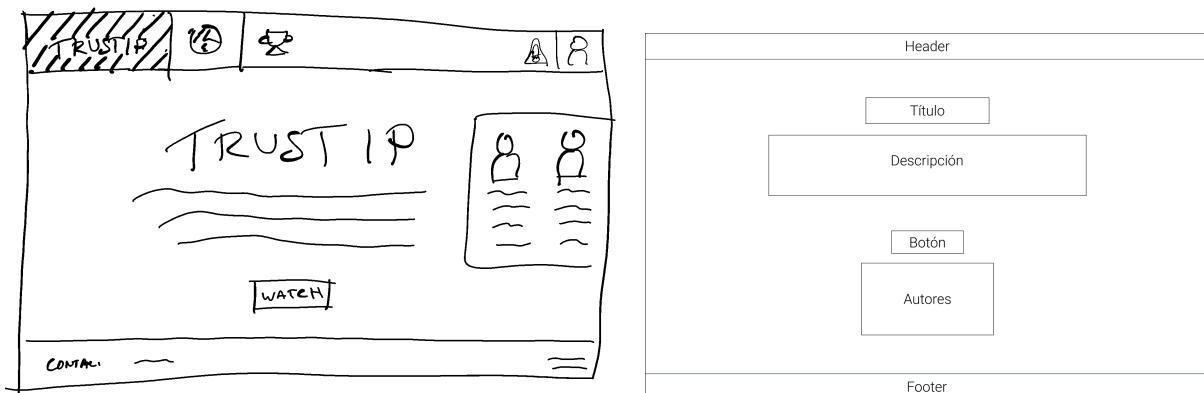
Interfaces

Paleta de Colores

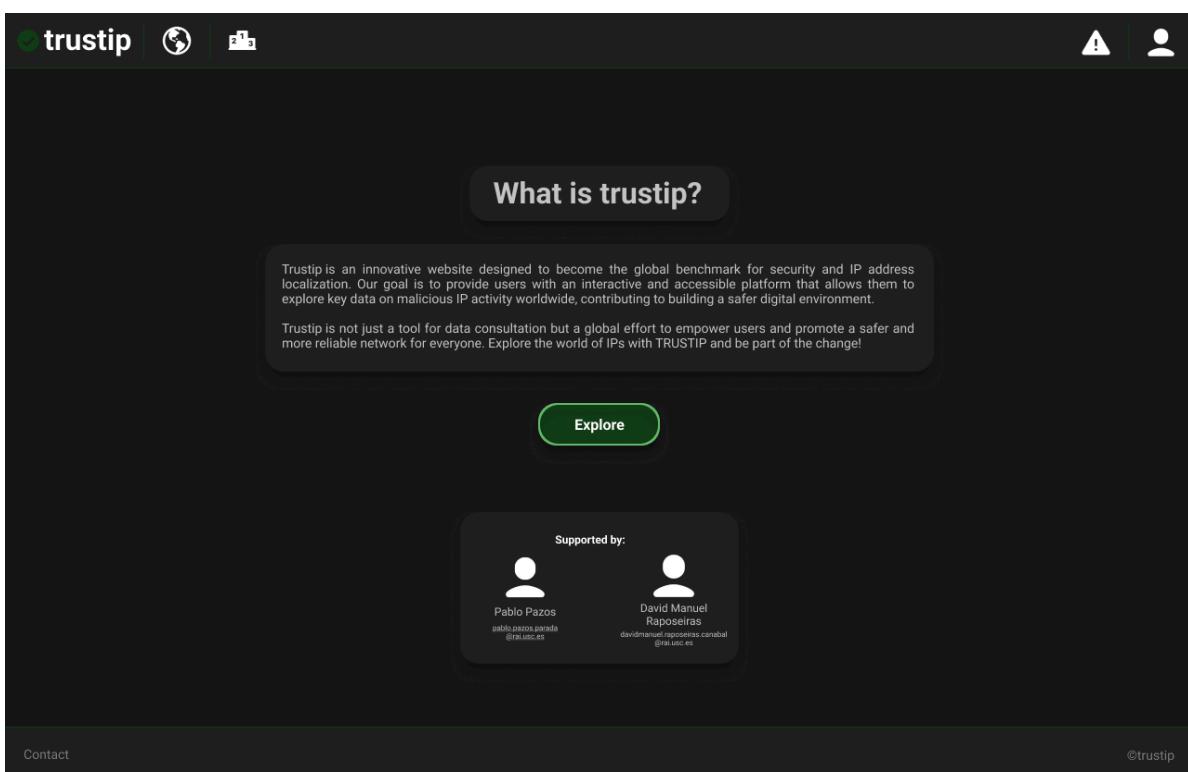


Esta paleta transmite seriedad y enfoque técnico gracias a los tonos oscuros, mientras que los verdes refuerzan la idea de seguridad y validación. Ideal para una web centrada en ciberseguridad y análisis de IPs maliciosas.

Welcome Page

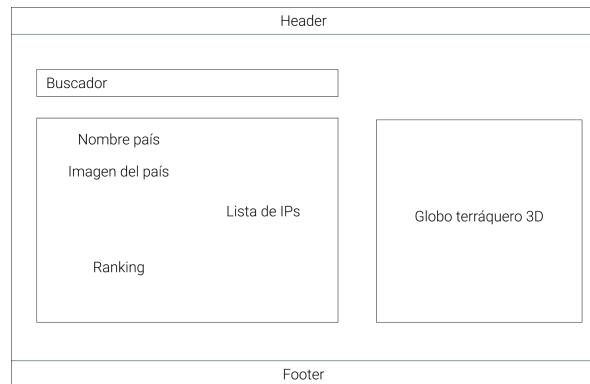
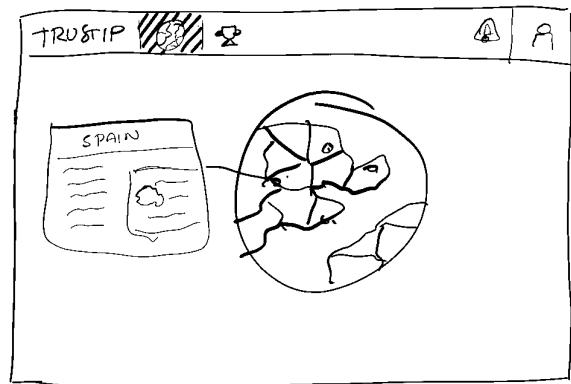


The image shows a hand-drawn sketch of a website header and a wireframe of a landing page. The sketch includes a logo, navigation icons, and a search bar. The wireframe shows a header section with 'Título' and 'Descripción', a central content area with 'Botón' and 'Autores', and a footer section.



The screenshot of the Trustip website features a dark theme. At the top, there is a navigation bar with the Trustip logo, a globe icon, and a search icon. Below the navigation, a large button labeled 'What is trustip?' is centered. A text box provides a brief description of the platform's purpose. A prominent green 'Explore' button is located in the center. At the bottom, a 'Supported by:' section lists two individuals: Pablo Pazos and David Manuel Raposeiras, each with their names and contact information. The footer contains links for 'Contact' and '©trustip'.

Globe Page

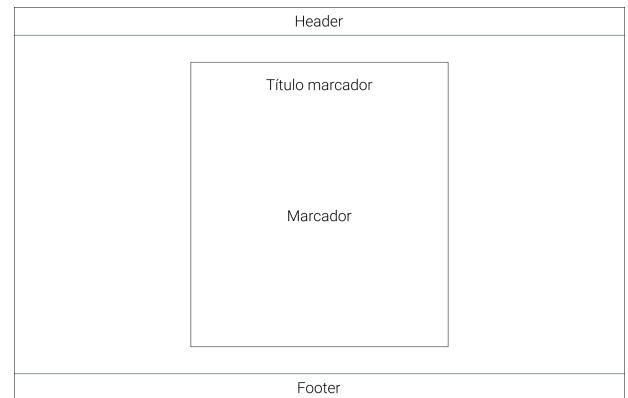
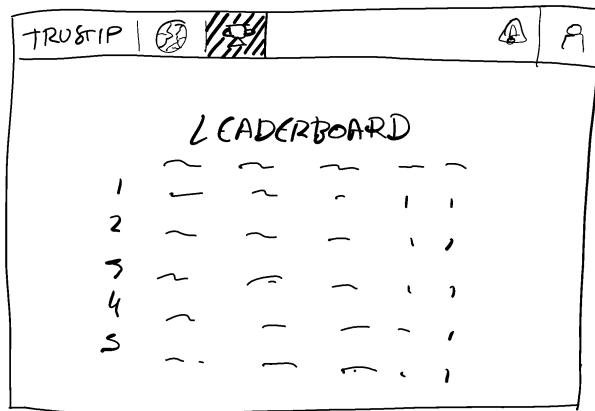
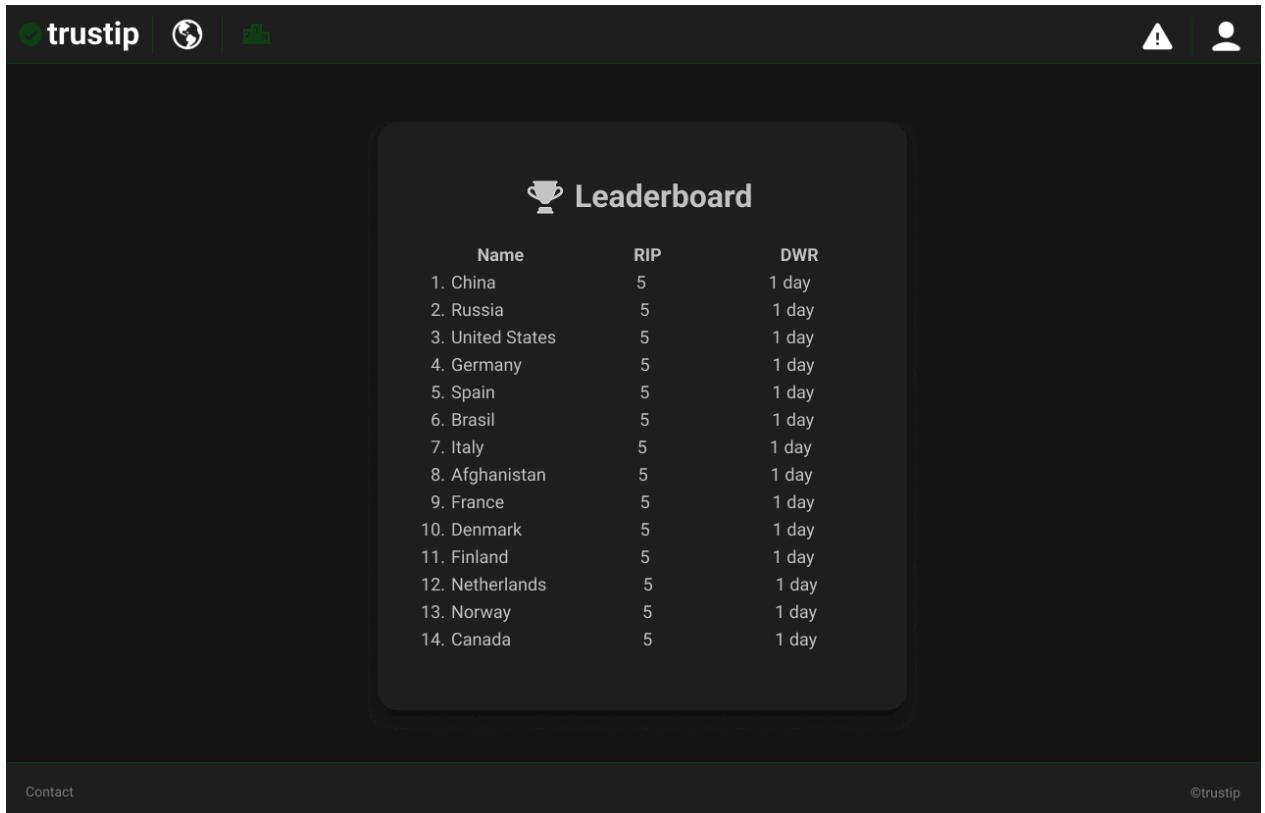


Rank: 5

IPs reported last Month: 15
IPs reported in total: 210

Contact ©trustip

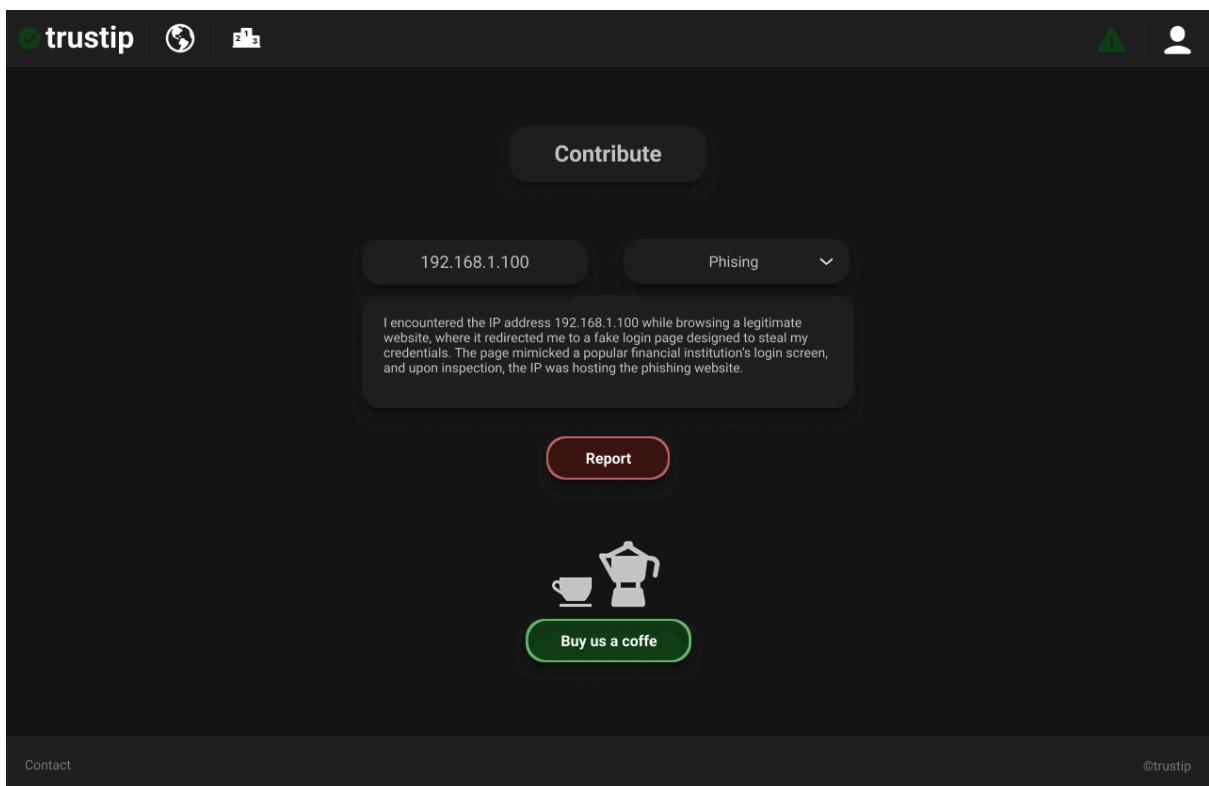
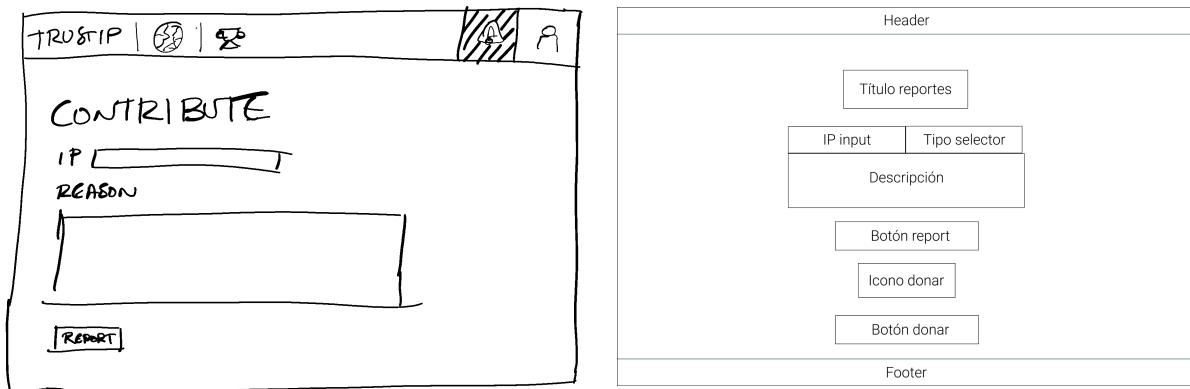
Leaderboard Page

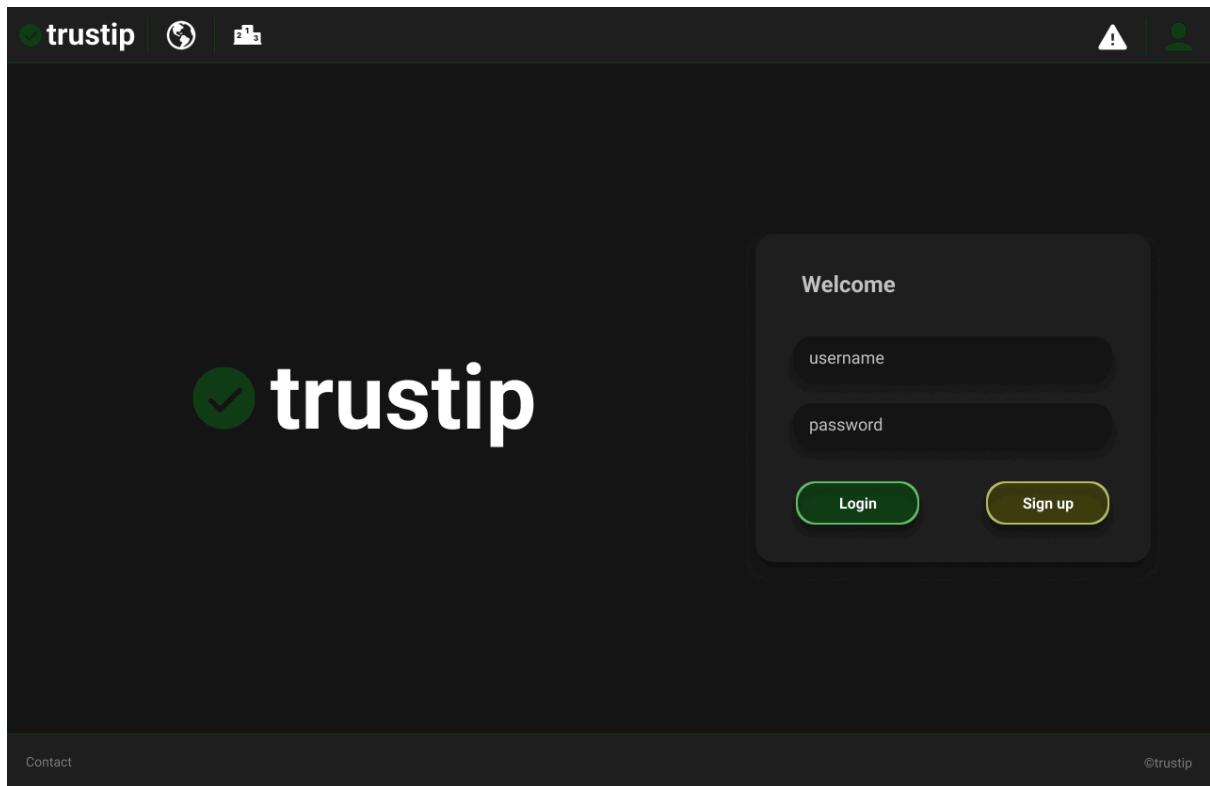
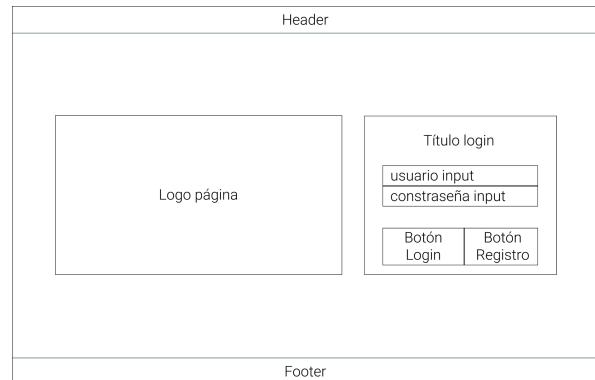
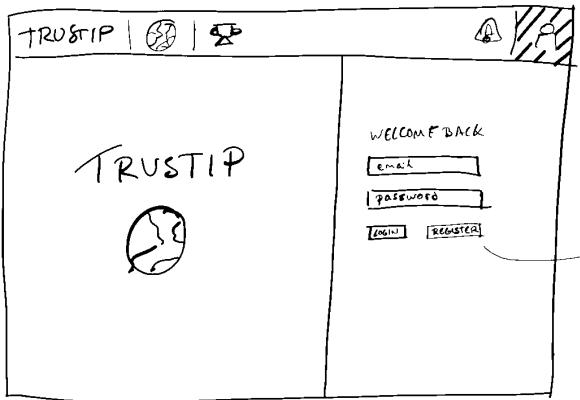
A screenshot of a mobile application's 'Leaderboard' page. The page has a dark background and features a title 'Leaderboard' with a trophy icon. Below the title is a table listing 14 countries with their names, R1P values (all 5), and DWR values (all 1 day). At the bottom of the screen are navigation links for 'Contact' and '©trustip'.

Name	R1P	DWR
1. China	5	1 day
2. Russia	5	1 day
3. United States	5	1 day
4. Germany	5	1 day
5. Spain	5	1 day
6. Brasil	5	1 day
7. Italy	5	1 day
8. Afghanistan	5	1 day
9. France	5	1 day
10. Denmark	5	1 day
11. Finland	5	1 day
12. Netherlands	5	1 day
13. Norway	5	1 day
14. Canada	5	1 day

Contribute Page



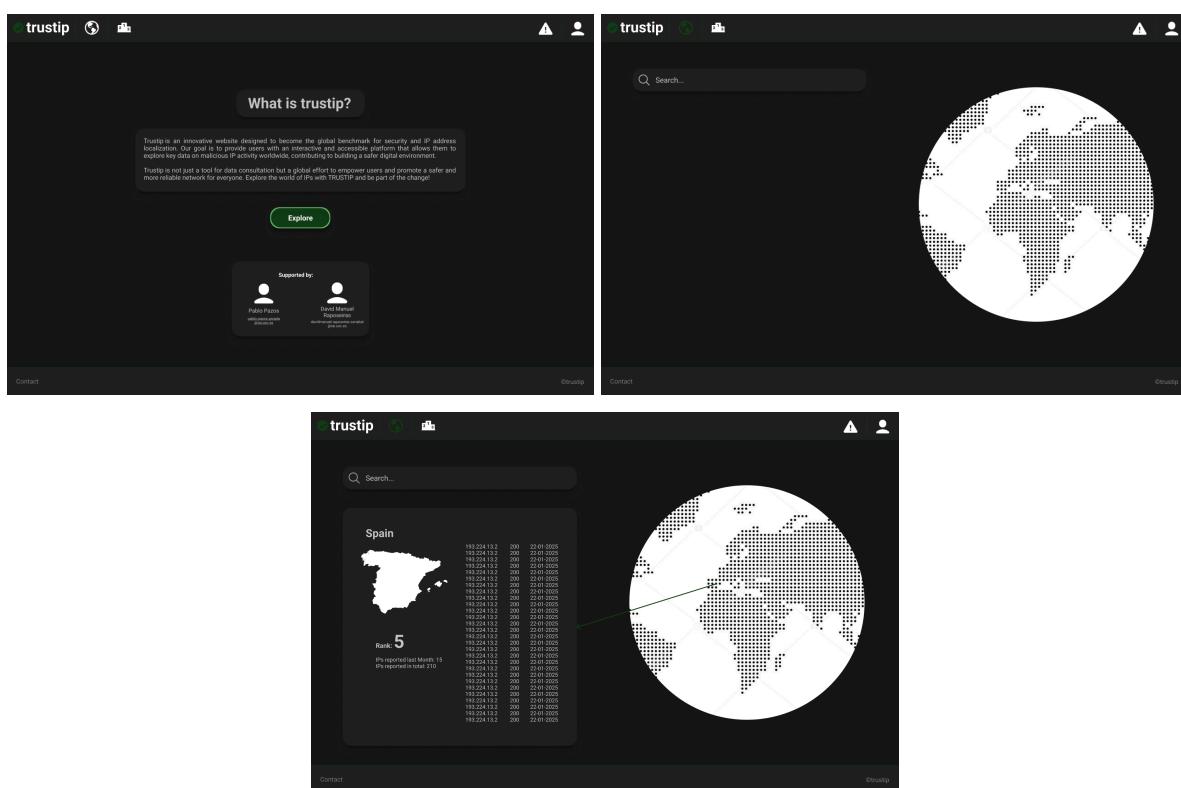
Login Page



Storyboards

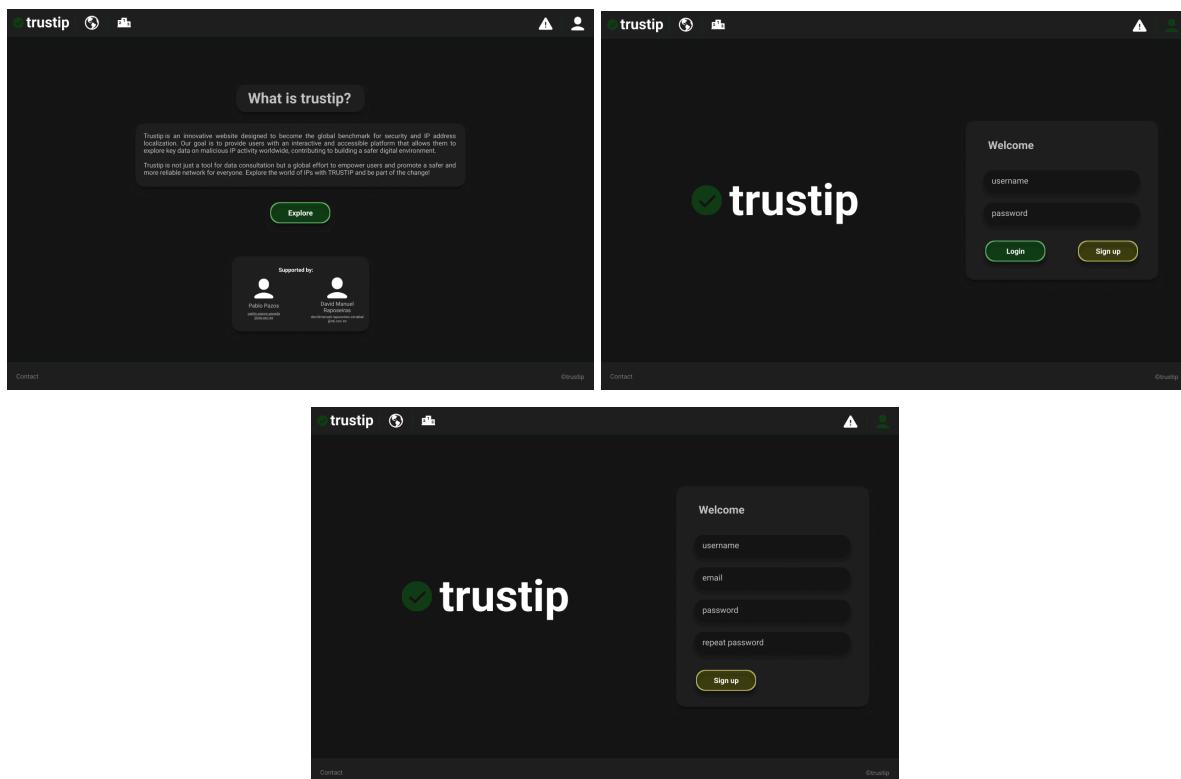
Consultar una IP/país

Para consultar una IP/país, se debe acceder a la Globe Page desde la página principal (la Welcome Page) pulsando el botón de “Explore” o el icono del globo terráqueo en la Top Bar. Una vez en la Globe Page se podrá buscar en la barra de búsqueda la IP o país a chequear y automáticamente el globo girará para posicionarse en ese país (el que se busca o al que pertenece la IP) y aparecerá un popup con la información del mismo.



Registrarse en la web

Para registrarse en *trustip*, se debe acceder a la Login Page desde la página principal (la Welcome Page) pulsando el icono del usuario en la Top Bar. Una vez en la Login Page habrá que pulsar el botón de “Sign up” para que aparezcan el resto de casillas a cubrir para poder registrarse. Una vez cubierto el formulario podremos registrarnos pulsando otra vez el botón de “Sign up”.



Estructura de ficheros

```
trustip/
├── css/
│   └── styles.css
├── data/
│   ├── coordinates.json
│   └── ips.json
└── js/
    ├── contributors.js
    ├── globe.js
    ├── leaderboard.js
    ├── login.js
    └── <> contributors.html
        ├── globe.html
        ├── index.html
        ├── leaderboard.html
        └── login.html
```

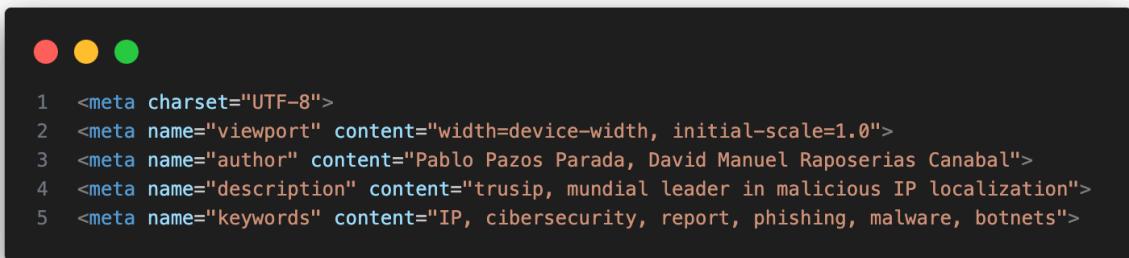
Implementación

HTML

La implementación de HTML en este proyecto se ha hecho siguiendo el concepto de HTML semántico, es decir, utilizando las tags del lenguaje para reforzar el significado de la información en las páginas más que para simplemente organizar esta información. Esto ha sido posible gracias a la utilización de HTML 5.0, que incluye nuevas etiquetas para hacer precisamente esto.

Aunque se ha priorizado siempre este estilo de programación, no siempre ha sido posible. A lo largo de este apartado se muestran los mapas y árboles de etiquetas de las interfaces descritas anteriormente. En estos mapas, podemos ver como el uso de divs ha sido necesario de vez en cuando debido a la ausencia de una etiqueta que representase mejor la información mostrada sin abusar de la etiqueta section.

Otro de los puntos a destacar en esta implementación es el buen uso de las etiquetas meta, las cuales hemos añadido en el head de cada página para reforzar la cohesión y sentido del proyecto. En la siguiente imagen se muestran las etiquetas mencionadas:



```
1 <meta charset="UTF-8">
2 <meta name="viewport" content="width=device-width, initial-scale=1.0">
3 <meta name="author" content="Pablo Pazos Parada, David Manuel Raposeras Canabal">
4 <meta name="description" content="trusip, mundial leader in malicious IP localization">
5 <meta name="keywords" content="IP, cibersecurity, report, phishing, malware, botnets">
```



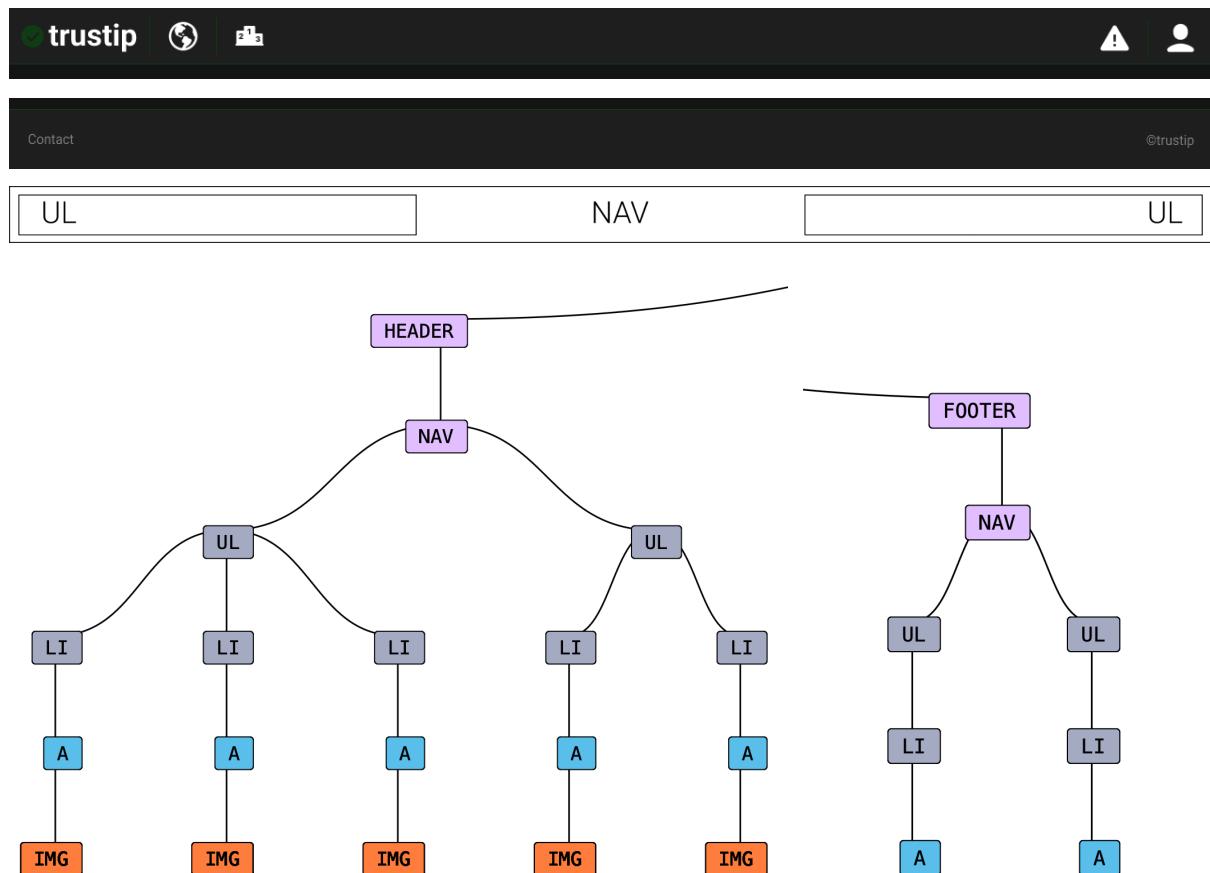
```
1 <!DOCTYPE html>
2 <html lang="en">
```

Por supuesto, al principio de cada página también se incluye la etiqueta <!DOCTYPE html>, indicando que vamos a utilizar, como se explicó antes, HTML 5. Además, tampoco nos olvidamos de añadir el lenguaje de nuestra página, en este caso inglés (mediante lang="en" en la etiqueta <html>).

El conjunto de estas buenas prácticas en el uso de etiquetas hace que nuestro código, además de ser más sostenible y escalable, también sea indexado de manera más correcta por los navegadores, generando, al final, un mayor número de visitas a nuestra página.

Header & Footer

A lo largo de la página web, hay dos elementos que siempre están presentes: el header y el footer. Para no añadirlos a todos los mapas y árboles de etiquetas, a continuación se muestran sus propios mapas:



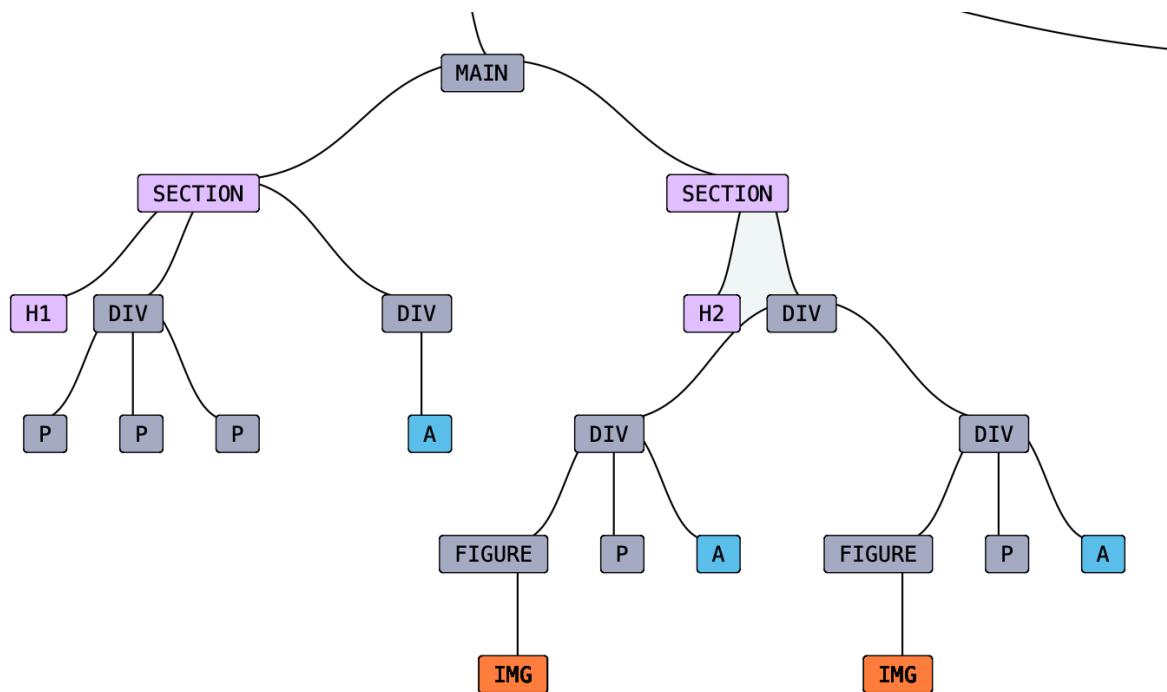
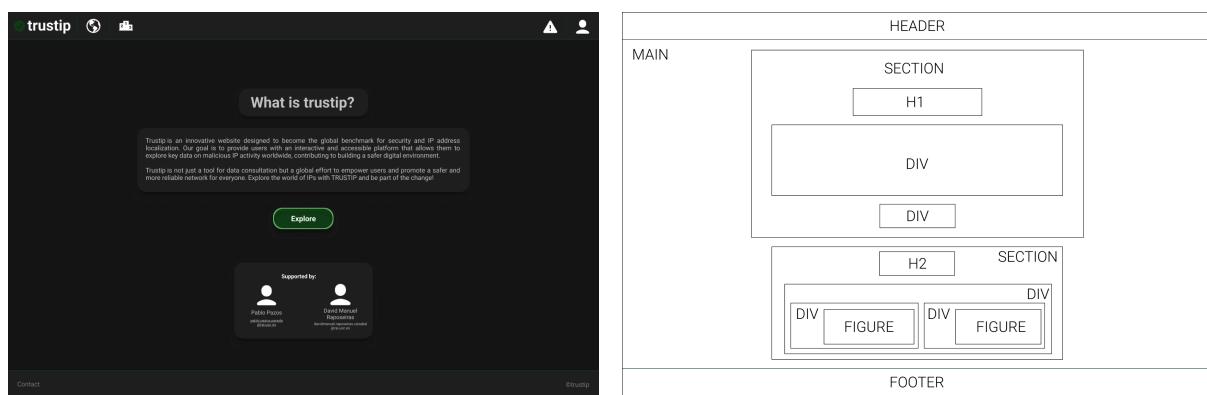
Como podemos ver, la implementación de ambos es prácticamente idéntica y muy sencilla; se componen simplemente de dos listas no numeradas. El objetivo es que una de ellas esté pegada al margen izquierdo del componente mientras que la otra lo esté al derecho (aunque esto no será posible hasta que implementemos CSS), para generar una sensación de división temática y también cubrir lo máximo posible el ancho de la pantalla.

La principal diferencia entre estos componentes radica en los elementos que conforman sus listas. El header utiliza imágenes que actúan como iconos, lo cual tiene como objetivo captar la atención del usuario y facilitar la navegación. En cambio, el footer se compone únicamente de texto, ya que su propósito es ofrecer información complementaria sin distraer del contenido principal de la página.

Welcome Page

Para la página de bienvenida, tenemos una disposición bastante sencilla formada por dos etiquetas section. En el primero de ellos nos encontramos con un simple heading y dos etiquetas div, el primero para una descripción y el segundo a modo de botón. Sería totalmente razonable usar la etiqueta button en vez de div para este último caso.

En el section de más abajo, nos encontramos de nuevo un heading y un div, pero esta vez el div tiene un estructura más compleja; cuenta con dos divs más con un figure cada uno.



Una buena práctica que hemos aplicado y que no se ha mencionado previamente, es el uso estricto de etiquetas de línea siempre contenidas dentro de etiquetas de

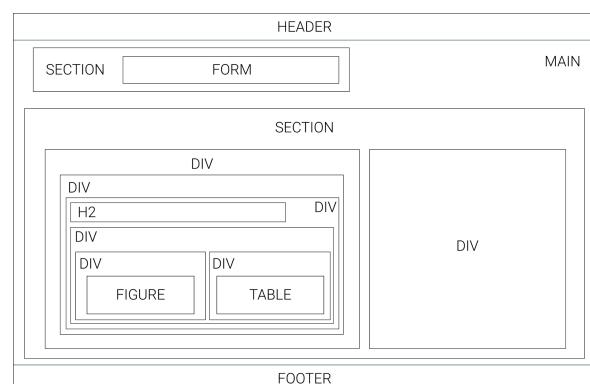
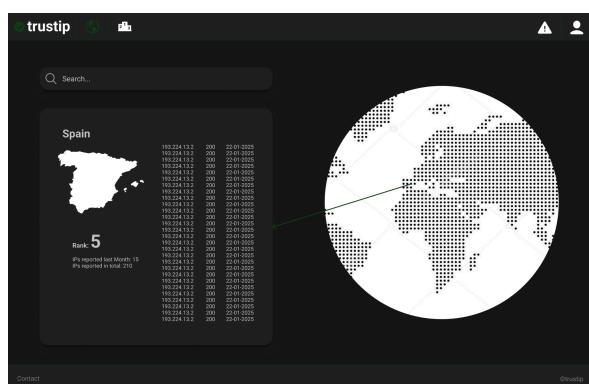
bloque. Esta decisión contribuye a mantener una estructura del código más limpia, coherente y predecible. Esta característica puede apreciarse tanto en el mapa de etiquetas actual como en los que se presentan a continuación.

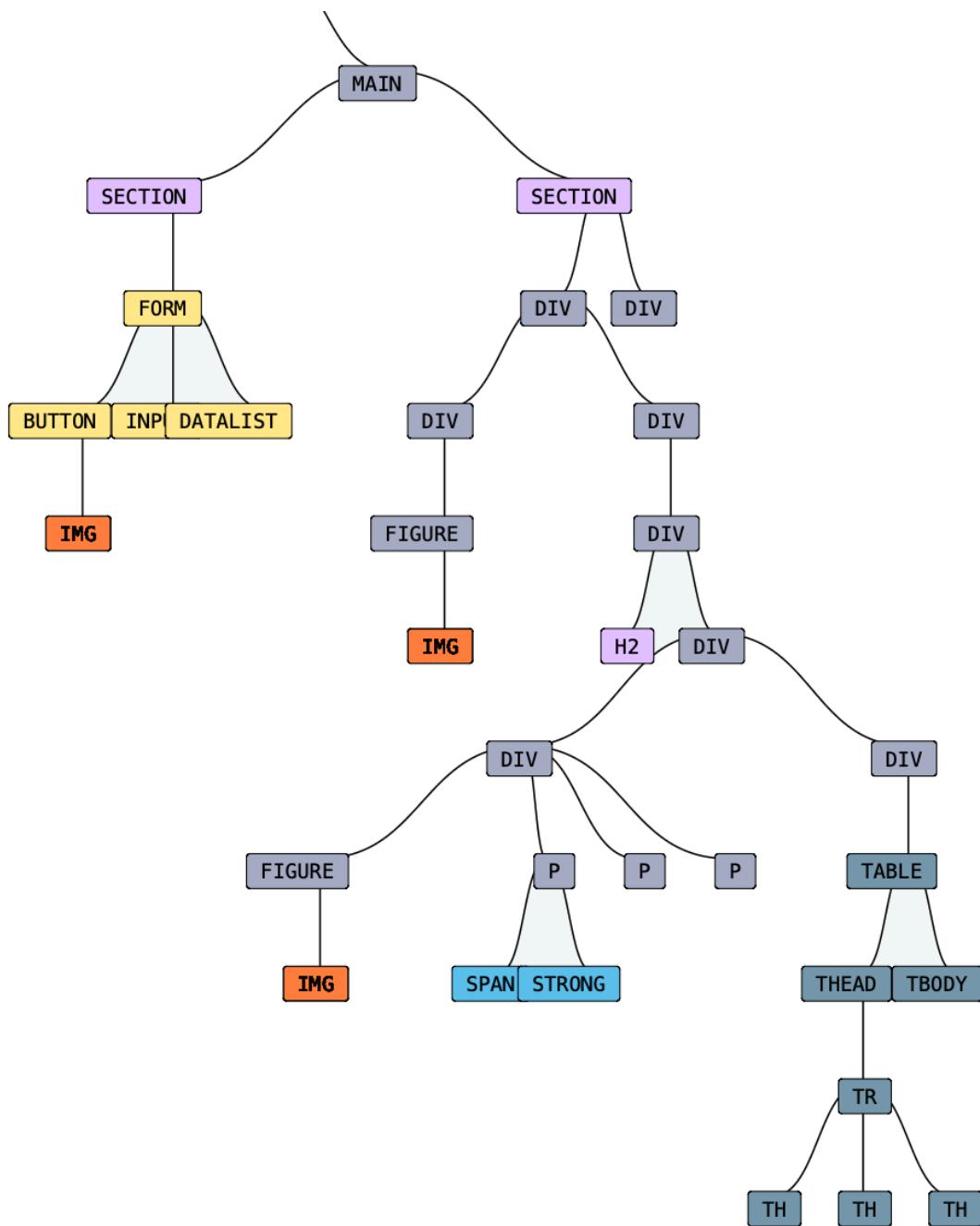
Globe Page

Continuando, nos encontramos con la página estrella de este proyecto; la página del globo terráqueo en 3D. Por desgracia, solo utilizando HTML no vamos a conseguir mucho, no será hasta que lleguemos a JavaScript que esta página comenzará a tener sentido.

El layout HTML se compone de dos etiquetas section, uno pequeño para contener el form de la barra de búsqueda, y otro principal que se dividirá a su vez en dos etiquetas div. Uno de ellos mostrará el propio globo 3D y el otro información de cada país con el que el usuario interactúe en el globo. Este último contiene, entre otras cosas, una tabla HTML en la que visualizar las IPs reportadas del país seleccionado..

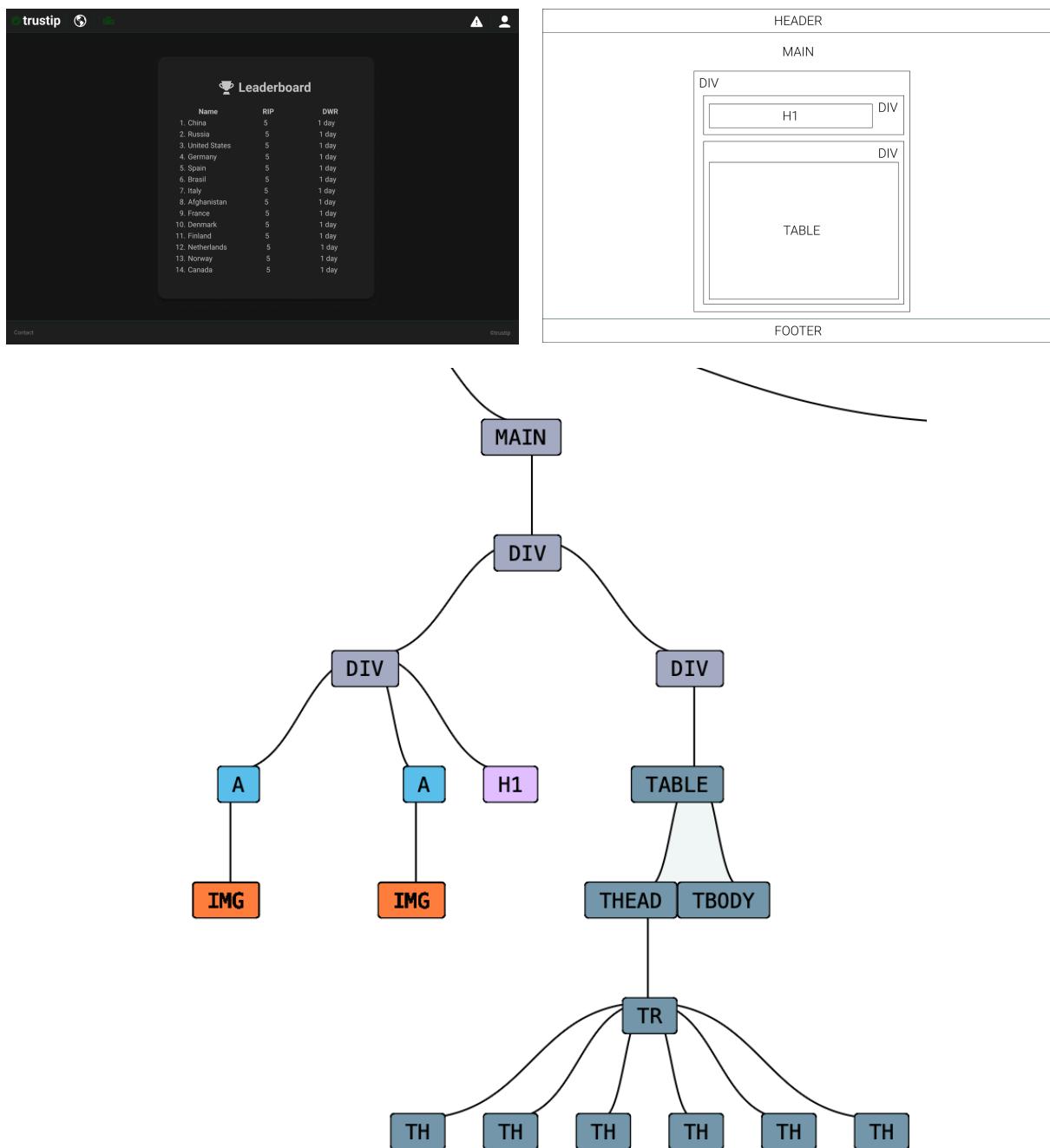
Tal y como se aprecia en el mapa de etiquetas que se presenta más adelante en esta sección, el <form> de la barra de búsqueda incorpora un elemento <datalist>. Esta etiqueta resulta especialmente útil, ya que permite ofrecer opciones de autocompletado al usuario a medida que escribe. En este caso concreto, se utilizará para almacenar los nombres de los países representados en el globo 3D, lo que facilitará una búsqueda más rápida, eficiente y accesible al proporcionar sugerencias automáticas basadas en los datos disponibles.





Leaderboard Page

La página siguiente contiene la tabla de clasificación. Su diseño es sencillo y minimalista, priorizando el enfoque del usuario en la propia tabla e intentando no sobrecargar la interfaz de información innecesaria. Para ello, se utiliza un div, que podría ser perfectamente un section, y que a su vez se divide en otras dos etiquetas div, uno con el heading y un par de iconos (se ha añadido uno más con respecto al mockup por cuestiones estéticas) y el otro con la tabla, el componente principal de esta página.

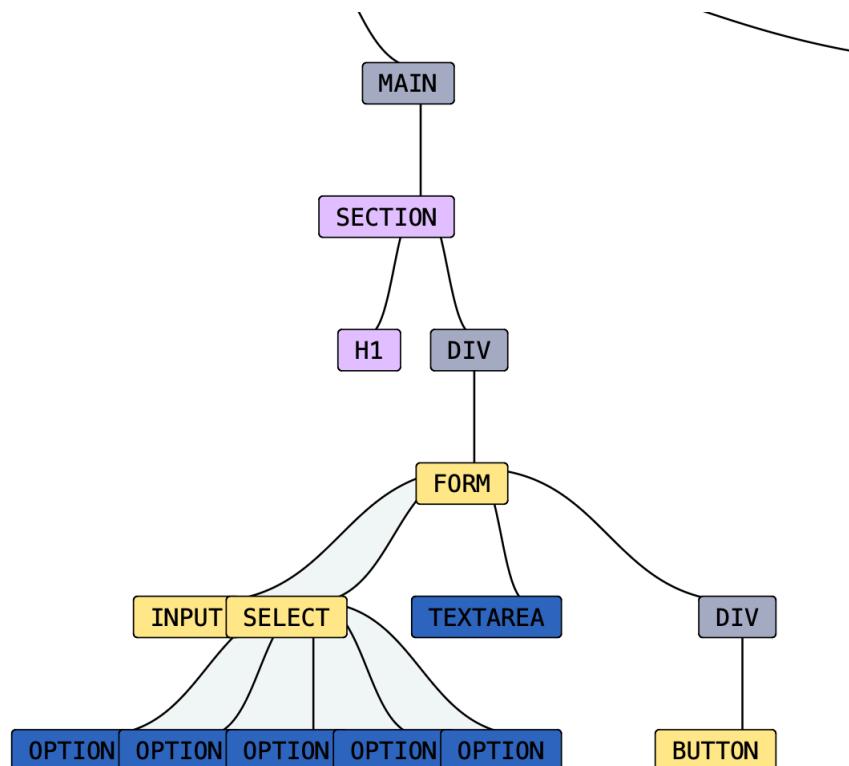
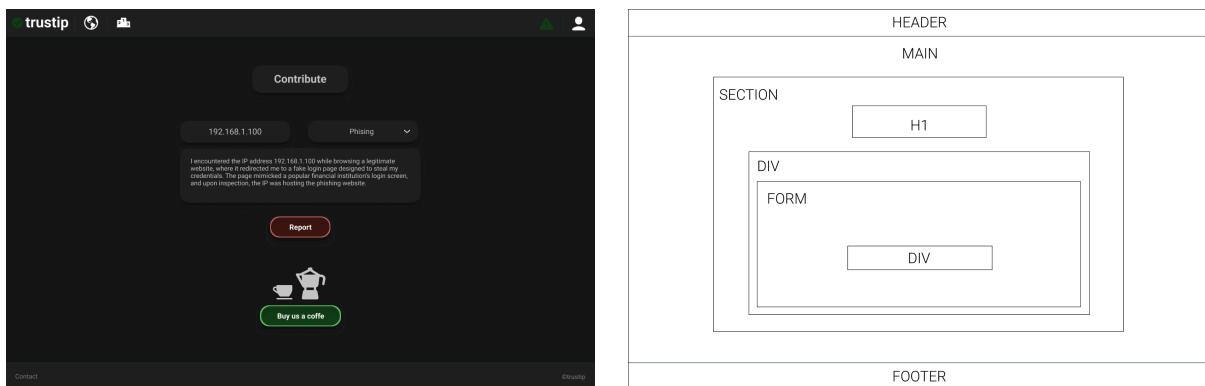


Contributors Page

Volvemos a tener aquí una arquitectura simple pero efectiva, esta vez utilizando solo un section, que contiene un heading y un div con el form entero dentro, además de un div en forma de botón para enviar el propio form.

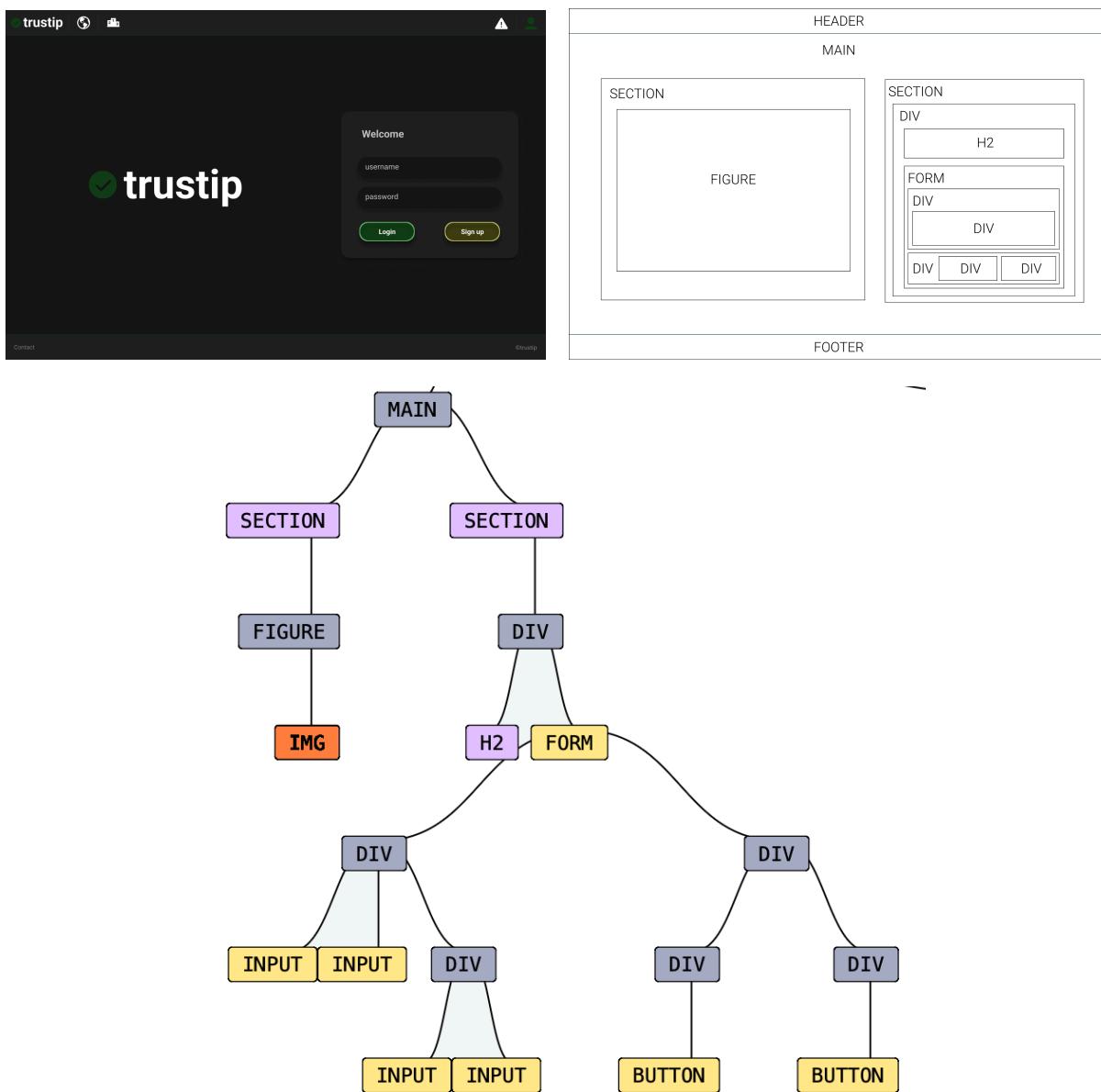
Algo a destacar de esta pantalla es el uso del elemento select con varias opciones acopladas. Gracias a esto, el usuario puede especificar el tipo de ataque que está reportando de manera cómoda.

En el apartado de modificaciones al final de este documento vienen recogidos los cambios que hemos hecho sobre esta página al empezar a codificar el HTML y darnos cuenta de que el mockup tenía partes innecesarias.



Login Page

Por último, llegamos a la Login Page. La disposición se basa en dos etiquetas section, una a modo de placeholder y otra que contiene todas las funcionalidades de la página. La primera de ellas simplemente contiene el logo de la Web y tiene fines estéticos. La segunda etiqueta section mencionada es la más importante, en él encontramos un div que contiene un heading y un form, conteniendo este último dos divs, que a su vez se dividen en uno y en dos respectivamente.



Aunque el form parezca que tiene una construcción excesivamente anidada por ese uso de un div dentro de otro y con inputs en ambos lugares, lo cierto es que lo que estamos haciendo aquí es adelantarnos a los acontecimientos. Como se ha mostrado en este documento previamente, uno de los casos de uso de esta web es el registro de un usuario. Este registro se hace pulsando el botón de "Signup", que

automáticamente extenderá el section de login, mostrando los campos extra que completar para poder registrarse. Ese primer div que vemos dentro del grande, es el encargado de hacer aparecer o desaparecer esos campos extra; más adelante, concretamente en la implementación de CSS, nos encargaremos de esto.

Estructura de ficheros actualizada

```
📁 trustip/
  └── assets/
    └── icons/
      ├── globe.png
      ├── logo.png
      └── user.png
      ...
    └── ...
    └── <> contribute.html
    └── <> globe.html
    └── <> index.html
    └── <> leaderboard.html
    └── <> login.html
```

CSS

Para la implementación del CSS de la web hemos decidido modularizar el código lo máximo posible. Para ello, hemos creado primero un archivo *main.css* que recogerá los estilos principales de la web, los cuales se reutilizarán en todas o casi todas las páginas.

```
● ● ●  
1 * {  
2     margin: 0;  
3     padding: 0;  
4     box-sizing: border-box;  
5 }
```

Lo primero de todo, como siempre se hace en este tipo de proyectos, es reiniciar los márgenes y paddings y fijar que el tamaño de los elementos incluya su padding y sus bordes.

Más adelante en este mismo fichero nos encontramos con la clase *elevated-card*, que define en gran parte el estilo de la web. Genera un estilo de tarjeta elevada, utilizando sombras, colores oscuros y bordes redondeados. Este tipo de elementos son muy utilizados en páginas webs modernas debido a su fácil reutilización, su esencia de minimalismo y su sensación de autoorganización.

```
● ● ●  
1 .elevated-card {  
2     background-color:  
3         var(--primary-color);  
4     padding: 16px 24px;  
5     border-radius: 16px;  
6     box-shadow: 6px 6px 12px  
7         rgba(0, 0, 0, 0.4);  
8     text-align: center;  
9     width: fit-content;  
10    margin: 16px auto;  
11 }
```

```
● ● ●  
1 :root {  
2     --primary-color: #123d15;  
3     --secondary-color: #3d3d12;  
4     --background-dark: #1E1E1E;  
5     --background-light: #aaa;  
6     --background-general: #181818;  
7     --text-light: #ffffff;  
8     --dark-red: #3d1712;  
9     --dark-green: #1A5D2B;  
10    --light-green: #66BB6A;  
11    --light-yellow: #B5BB66;  
12    --light-red: #bb6666;  
13 }
```

Si nos fijamos bien en el código anterior, nos encontraremos con que el color del fondo no está especificado como suele estarlo en un CSS; ya sea con hex, rgb o cualquier otra manera. Esto es porque, siguiendo nuestros principios de modularidad, hemos creado variables globales para poder reutilizar los colores en todos los documentos y poder realizar cambios sobre ellos de una manera rápida y sencilla.

El *body* y el *main* también reciben un estilo predeterminado. En el *body* se define el color de fondo, es estilo de letra y el padding necesario para que los elementos no se superpongan o antepongan al header y footer. El *main*, por su parte, recibe un estilo orientado a la organización de elementos en el mismo; un flex centrado de tipo columna con un espaciado adecuado entre elementos.



```

1 body {
2   font-family: Arial, sans-serif;
3   background-color:
4     var(--background-general);
5   color: var(--text-light);
6   text-align: center;
7   padding-top: 72px;
8   padding-bottom: 72px;
9 }
10

```

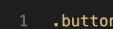


```

1 main {
2   display: flex;
3   flex-direction: column;
4   align-items: center;
5   gap: 20px;
6 }

```

Otro elemento que recibe estilos ya en esta página, son los botones. Al tener varios a lo largo de la Web, es necesario definir unas pautas comunes; están son las nuestras:



```

1 .button {
2   margin-top: 10px;
3   text-decoration: none;
4   font-size: 18px;
5   font-weight: bold;
6   color: var(--text-light);
7   padding: 10px 32px;
8   border-radius: 24px;
9   display: inline-block;
10  transition: background 0.3s, transform 0.2s;
11  border-width: 2px;
12  border-style: solid;
13 }
14
15 .button :hover {
16   transform: scale(1.05);
17 }
18

```



Si los instanciamos en cualquier otra página y le damos colores, obtendremos el efecto representado en las imágenes de la derecha cuando dejemos el ratón sobre cualquier de ellos.



```

1 @media screen and (max-width: 600px){
2   .content-card{
3     max-width: 100%;
4   }
5   html,body{
6     overflow: scroll;
7     padding-right: 8px;
8     padding-left: 8px;
9   }
10  main{
11    gap: 0px;
12  }
13  .elevated-card{
14    margin: 0;
15 }
16

```

En esta página también definimos un pequeño ajuste responsive para la vista desde móviles; aumentando tamaños, reduciendo paddings y aplicando alguna que otra modificación más para que la experiencia desde estos dispositivos más pequeños no sea inferior.

En las siguientes secciones se incluirán imágenes comparativas donde podremos ver estos pequeños detalles en acción.

Header & Footer

Por último, en main.css también se encuentra el estilo del header y el footer. Como ambos van a ser utilizados en todas las páginas, y esta hoja de estilos es la que marca las líneas generales, tiene mucho sentido que los estilos de estos dos elementos se definan aquí.

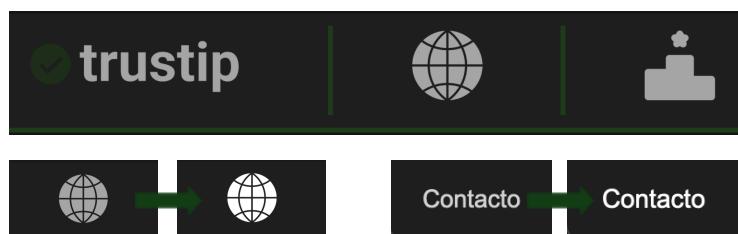
Como pudimos ver en el apartado de HTML, y como podremos ver en las imágenes que se muestran más abajo, la estructura y estilo del header y footer es muy similar.

```
● ● ●
1 header, footer {
2   background-color:
3     var(--background-dark);
4   padding: 20px;
5   display: flex;
6   align-items: center;
7   height: 48px;
8   width: 100%;
9   position: fixed;
10  left: 0;
11  z-index: 1000;
12 }
13
14 header {
15   border-bottom: 2px solid var(--primary-color);
16   top: 0;
17 }
18
19 footer {
20   border-top: 2px solid var(--primary-color);
21   bottom: 0;
22 }
23
24 header nav, footer nav {
25   display: flex;
26   justify-content: space-between;
27   width: 100%;
28   align-items: center;
29 }
30
31 .left-header, .right-header {
32   list-style: none;
33   display: flex;
34   gap: 2vw;
35   height: 36px;
36 }
37
38 .left-footer, .right-footer{
39   list-style: none;
40   height: 28px;
41 }
42
43 .right-header, .right-footer {
44   margin-left: auto;
45 }
46
47 header ul li a img {
48   height: 32px;
49   width: auto;
50   padding: 2px;
51   opacity: 0.6;
52   transition: opacity 0.3s ease-in-out;
53 }
54
55 header ul li a img.selected, header ul li a img:hover {
56   opacity: 1;
57 }
58
59 footer ul li a {
60   text-decoration: none;
61   color: var(--text-light);
62   font-size: 14px;
63   opacity: 0.8;
64   transition: opacity 0.3s ease-in-out;
65 }
66
67 footer ul li a:hover {
68   opacity: 1;
69 }
70
71 header ul li:not(:last-child),
72 footer ul:not(:last-child) {
73   border-right: 2px solid var(--primary-color);
74   padding-right: 2vw;
75 }
```

Se utiliza un fondo oscuro para integrarse con el resto del diseño y se fija una altura y posición constante. Con z-index nos aseguramos que los elementos están por encima del resto de contenido. Aseguramos que los elementos internos estén alineados verticalmente con el uso de *display: flex* y *align-items: center*.

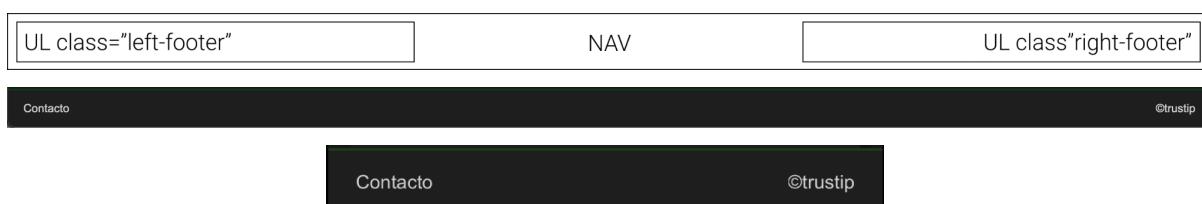
Tanto el header como en el footer está formado por un elemento *nav* con dos listas *ul*, una a la izquierda y otra a la derecha. Los *nav*, las listas y los elementos de navegación tienen todos su propio estilo acorde con el diseño global de la página y que podemos ver en la imagen adjunta.

Además, se añaden elementos separadores entre los elementos de cada una de las listas y animaciones tipo *hover* para mejorar la interacción con el usuario.



Por último, con la clase *selected*, hacemos que el efecto del hover se mantenga en el ícono que represente la página en la que estamos.

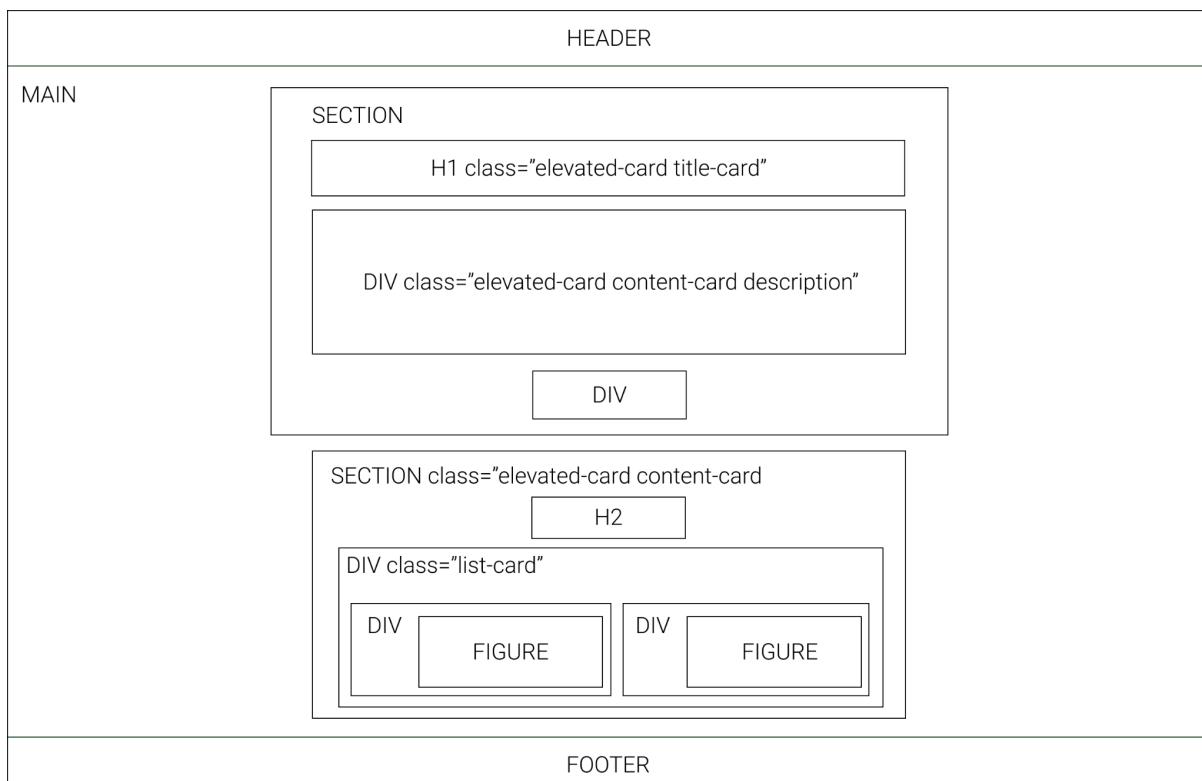
El resultado final del header y footer en ordenador y móvil es el siguiente:



A partir de aquí, explicaremos página por página los puntos más destacados de sus respectivas hojas de estilo.

Welcome Page

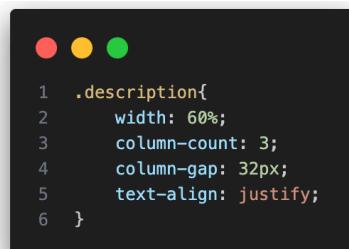
A continuación se muestran las clases asignadas a cada uno de los elementos mencionados para esta página en el apartado de HTML:



Multicol

Si nos fijamos en el primer *div* del primer *section*, veremos que contiene la clase *description*. Esta clase utiliza CSS multicol para la responsividad del texto que contiene. Este se presenta en tres columnas, que se ajustan según el ancho de la pantalla, y que mejoran la legibilidad del contenido. Además, este elemento se fija con una anchura del 60% de la pantalla para que exista un margen responsivo horizontal y que el texto no tenga una altura demasiado baja.

El problema de usar estas tres columnas, es que en móvil la experiencia de usuario será muy mala, ya que cabrá aproximadamente una palabra por columna; por lo que nos interesa revertir este elemento a solo una columna. Esto lo haremos gracias a *media query*:



```

1 .description{
2   width: 60%;
3   column-count: 3;
4   column-gap: 32px;
5   text-align: justify;
6 }

```

TRUSTIP is a cutting-edge platform aimed at becoming the global standard for IP address localization and cybersecurity. It offers users an intuitive and engaging interface to explore critical data on malicious IP activity across the world. TRUSTIP is not just a tool for data consultation but a global effort to empower users and promote a safer and more reliable network for everyone. Explore the world of IPs with TRUSTIP and be part of the change! TRUSTIP leverages real-time intelligence and community-driven reporting to provide a transparent view of global IP threats. With a focus on accessibility and precision, it enables users to identify, track, and understand potentially harmful activity online. Whether you're a cybersecurity expert or a curious user, TRUSTIP empowers you to take action, contribute to a safer web, and stay informed about the evolving digital landscape.



```

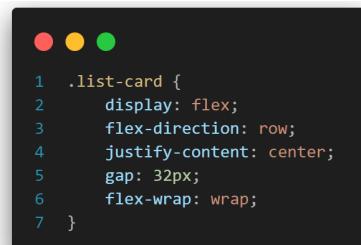
1 @media screen and (max-width: 600px){
2   .description{
3     column-count: 1;
4   }
5
6   .elevated-card{
7     width: 100%;
8   }
9 }

```

TRUSTIP is a cutting-edge platform aimed at becoming the global standard for IP address localization and cybersecurity. It offers users an intuitive and engaging interface to explore critical data on malicious IP activity across the world. TRUSTIP is not just a tool for data consultation but a global effort to empower users and promote a safer and more reliable network for everyone. Explore the world of IPs with TRUSTIP and be part of the change! TRUSTIP leverages real-time intelligence and community-driven reporting to provide a transparent view of global IP threats. With a focus on accessibility and precision, it enables users to identify, track, and understand potentially harmful activity online. Whether you're a cybersecurity expert or a curious user, TRUSTIP empowers you to take action, contribute to a safer web, and stay informed about the evolving digital landscape.

Flex-wrap

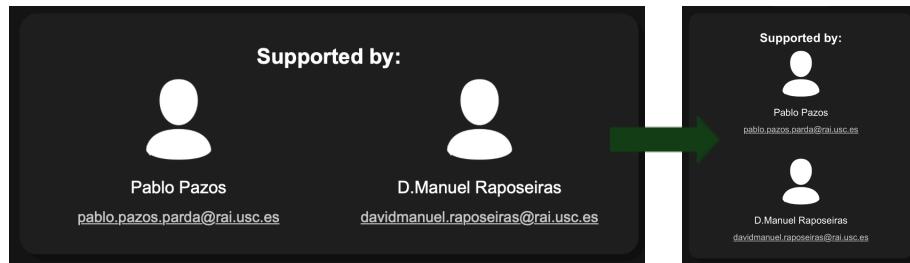
Dentro del único *div* presente en el segundo *section*, se utiliza la clase *list-card*, la cual aplica un diseño de tipo *flex* en dirección horizontal (*flex-row*). Esto permite que los elementos hijos se distribuyan de forma uniforme a lo largo del eje horizontal, logrando una disposición ordenada y alineada visualmente. Añadiéndole el atributo *flex-wrap: wrap* a esta clase, conseguimos que cuando la pantalla no sea lo suficientemente grande como para mostrar los elementos en fila, los muestre en columna; aportando la responsividad que buscamos.



```

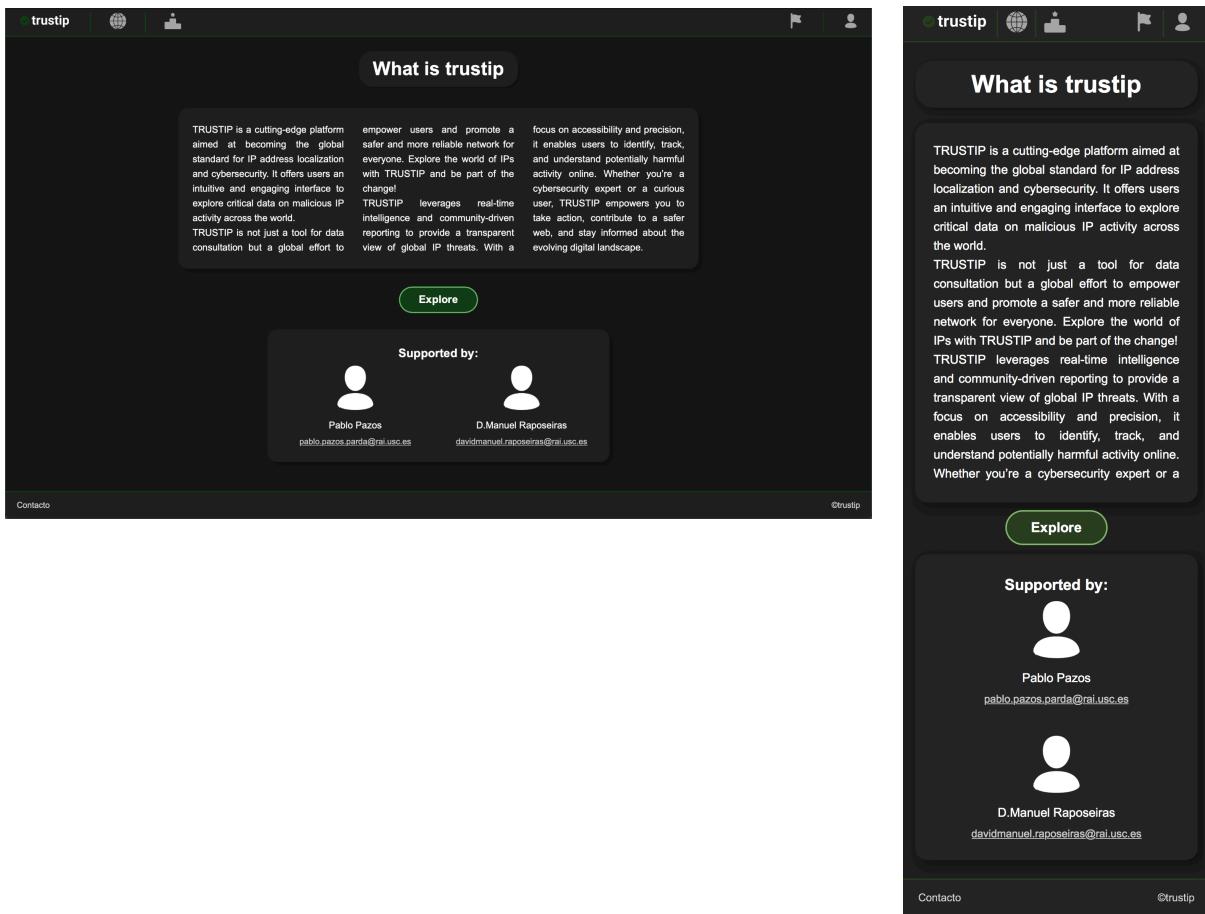
1 .list-card {
2   display: flex;
3   flex-direction: row;
4   justify-content: center;
5   gap: 32px;
6   flex-wrap: wrap;
7 }

```



Aspecto final

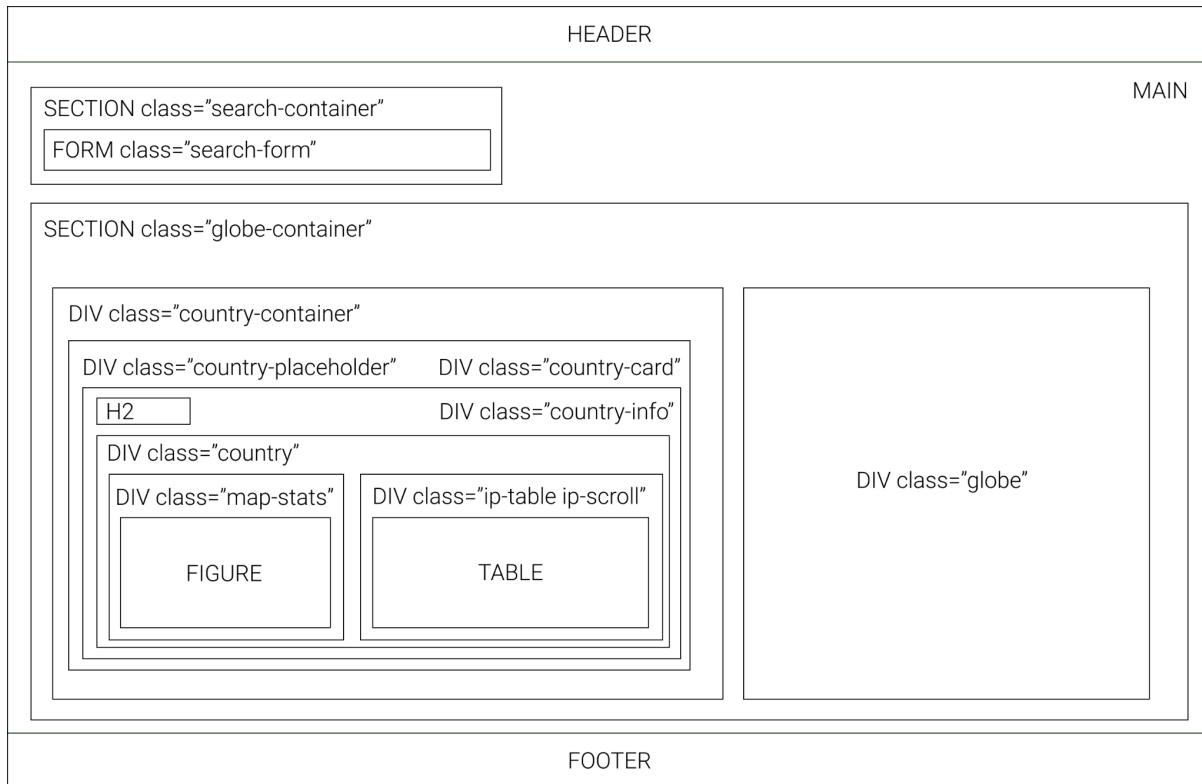
Con estos añadidos, el aspecto final de la página es el siguiente:



The screenshot shows the final design of the trustip website. The 'What is trustip' section on the left contains a detailed description of the platform's purpose and features, with a prominent 'Explore' button. The 'Supported by' section on the right lists the two individuals and their contact information. The overall layout is clean and professional, reflecting the changes made to the page.

Globe Page

De nuevo, las clases de cada uno de los elementos del apartado de HTML:



Flex Container

La página se apoya fuertemente en Flex Container para mantener una estructura ordenada. Las clases `.globe-container`, `.country`, `.country-placeholder`, `.map-stats`, `.search-icon`, `.search-form` y `.globe` utilizan `display: flex` para organizar los elementos en fila o columna, según el caso. Además, mediante el uso de propiedades como `gap`, `flex-direction`, `height` y `align-items`, se asegura que el contenido se mantenga bien distribuido incluso con cambios de resolución. Aunque pueda parecer excesivo, estas propiedades permiten que el diseño se ajuste de forma controlada y elegante en pantallas más estrechas, como ya hemos demostrado anteriormente.

Solapamiento de elementos

En esta página añadimos el uso de `grid` en la clase `.country-container`. Esto nos permite superponer la tarjeta de país sobre su placeholder. Conseguimos esto haciendo que ambos estén en la misma celda del grid, por lo que no tienen más remedio que superponerse. De esta forma, conseguimos un placeholder para cuando entremos en la página y aún no hayamos hecho ninguna búsqueda.

Por supuesto, la transición de placeholder a tarjeta se hace mediante una animación, así como la transición de una tarjeta de país a otra.

```

1 .country-container {
2   display: grid;
3   grid-template-columns: 1fr;
4   grid-template-rows: 1fr;
5   max-width: 600px;
6 }
7
8 .country-placeholder,
9 .country-card{
10   grid-column: 1 / -1;
11   grid-row: 1 / -1;
12 }
13
14 .country-placeholder {
15   z-index: 1;
16
17   max-width: 600px;
18   height: 100%;
19
20   display: flex;
21   align-items: center;
22   justify-content: center;
23 }
24
25
26 .country-card {
27   z-index: 2;
28
29   opacity: 0;
30   pointer-events: none;
31   padding: 32px;
32   transition: transform 0.5s ease, opacity 0.5s ease;
33   background-color: var(--background-dark);
34   border-radius: 16px;
35   box-shadow: 6px 6px 12px rgba(0, 0, 0, 0.4);
36   color: white;
37   max-width: 600px;
38   height: 100%;
39 }

```

```

1 .country-card.show {
2   opacity: 1;
3   pointer-events: auto;
4   animation: scaleUp 0.5s ease-out forwards;
5 }
6
7 .country-info.show {
8   visibility: visible;
9   animation: fadeInSlideUp 0.5s ease-out forwards;
10 }

```



```

1 @media screen and (max-width: 600px) {
2   html, body {
3     overflow: scroll;
4   }
5
6   .globe-container {
7     flex-direction: column;
8     padding: 0;
9   }
10
11   .ip-scroll {
12     overflow-x: auto;
13   }
14
15   .ip-table table td {
16     word-break: break-word;
17   }
18
19   .country-card{
20     display: flex;
21   }
22
23   .country-placeholder {
24     display: none;
25   }
26
27   .country {
28     flex-direction: column;
29   }
30
31   .search-container{
32     width: 100%;
33   }
34
35   .country-card{
36     width: 100%;
37     margin-top: 16px;
38   }
39
40   .globe{
41     display: none;
42   }
43 }

```

Responsividad para móvil

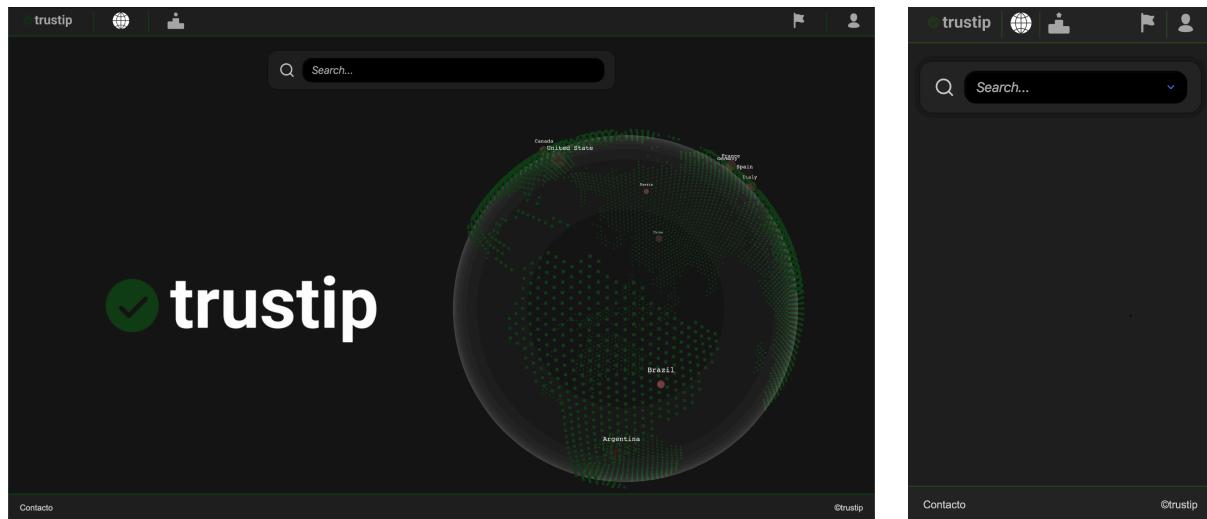
Como también sucede en la página anterior, se utiliza una *media query* para adaptar el diseño a dispositivos móviles. El poco espacio en las pantallas móviles y el alto contenido de nuestras páginas provoca que tengamos que realizar grandes cambios; el primero de ellos, permitir el desplazamiento (*overflow: scroll*).

El globo terráqueo y el placeholder de la tarjeta del país se eliminan desde un primer momento, dejando a la barra búsqueda como único elemento de la pantalla. Los contenedores principales de la página (.globe-container y country-container), también sufren cambios, ya que la disposición de sus elementos pasa de fila a columna para ajustarse a pantallas más pequeñas.

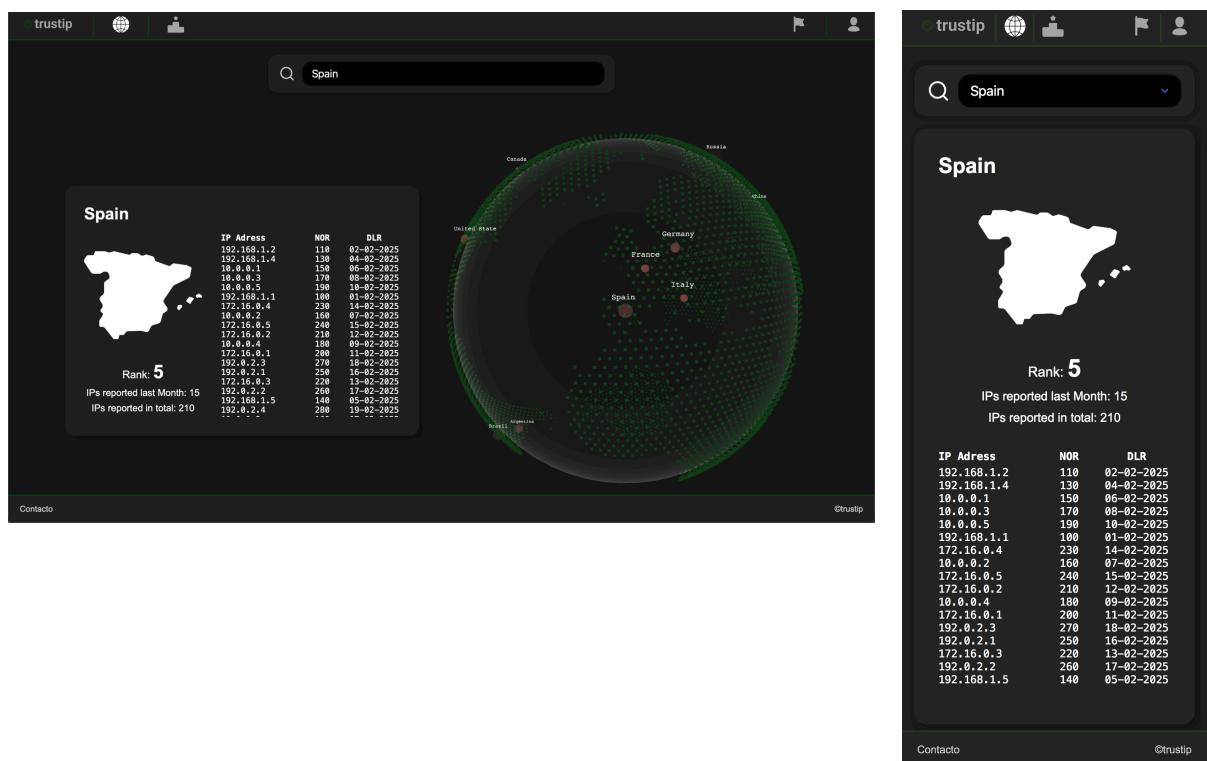
Por último, *.country-card* y *.search-container* pasan a ocupar todo el ancho de la pantalla (*width: 100%*), optimizando el espacio útil.

Aspecto final

Tras esta implementación, así nos encontraremos la página en cada visita:

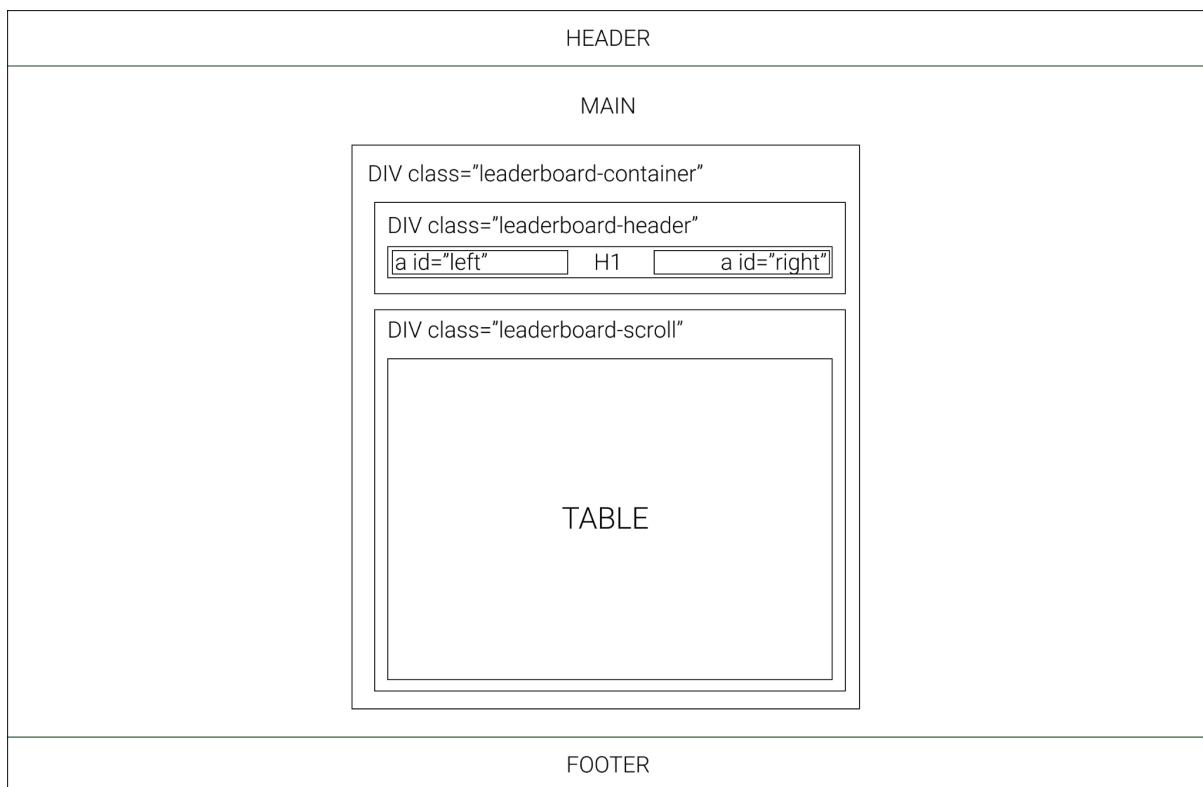


Y así tras la búsqueda de un país (ya sea mediante el globo o la barra de búsqueda):



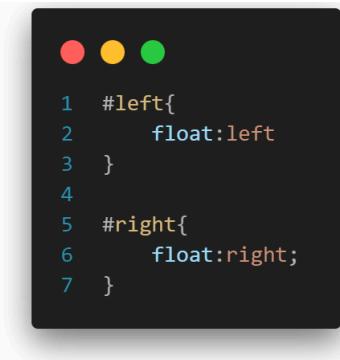
Leaderboard Page

Una vez más, las clases de la estructura HTML:



Float

Dentro de la clase *leaderboard-header* se incluyen dos elementos identificados con los ids *left* y *right*, los cuales contienen los iconos de los trofeos. Para lograr que estos iconos se posicionen a ambos lados del título central, se utiliza la propiedad *float*. Con *float: left;* se coloca el ícono de la izquierda, mientras que con *float: right;* se posiciona el ícono de la derecha, permitiendo que el título quede centrado en el espacio intermedio. Este enfoque facilita la disposición visual de los elementos en línea y contribuye a una presentación equilibrada y atractiva en la cabecera del leaderboard, tal como se observa en la imagen adjunta.



```
1 #left{  
2     float:left  
3 }  
4  
5 #right{  
6     float:right;  
7 }
```



Responsividad

Al igual que en las páginas anteriores, también se ha adaptado el diseño para pantallas móviles. El contenedor principal se ajusta para ocupar todo el ancho y alto de la pantalla, aprovechando así el espacio disponible. La cabecera de la tabla pasa a abarcar el ancho completo, mientras que el tamaño de la fuente y el espaciado de las celdas se reducen ligeramente para mejorar la legibilidad en dispositivos con pantallas más pequeñas.

Todos estos son pequeños cambios pero que, como se verá a continuación, mejoran considerablemente la experiencia de usuario en dispositivos móviles.

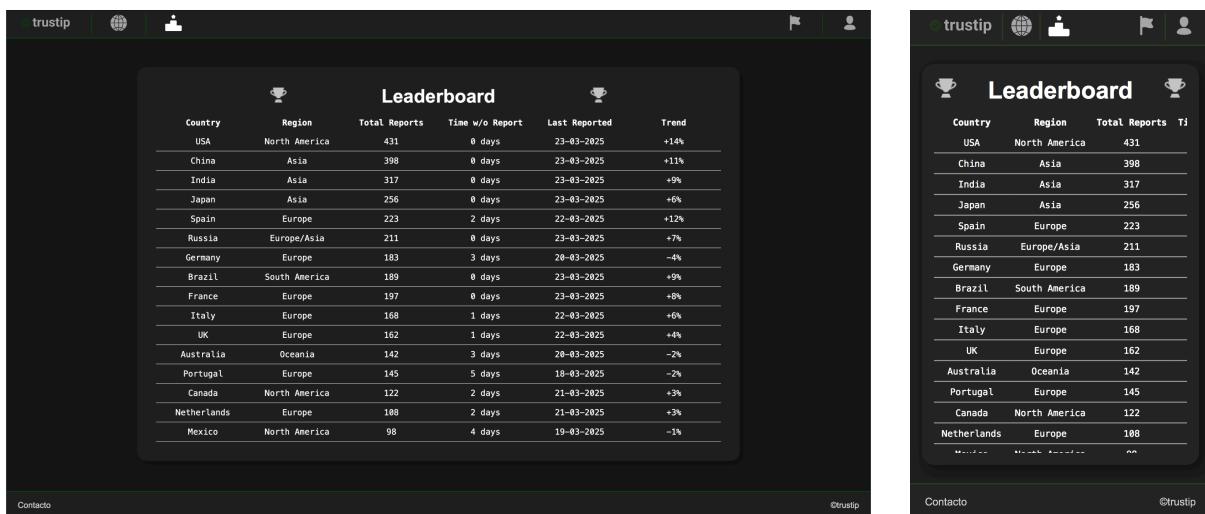
```

1 @media screen and (max-width: 600px) {
2   .leaderboard-container {
3     padding: 16px;
4     width: 100%;
5     height: 100%;
6     margin-top: 0;
7   }
8   .leaderboard-header {
9     width: 100%;
10  }
11  .leaderboard-scroll table {
12    font-size: 12px;
13  }
14  th {
15    width: auto;
16    white-space: nowrap;
17  }
18  td, th {
19    padding: 6px;
20  }
21  body{
22    padding-bottom: -20px;
23  }
24 }

```

Aspecto final

Este es el aspecto estético final de la página:

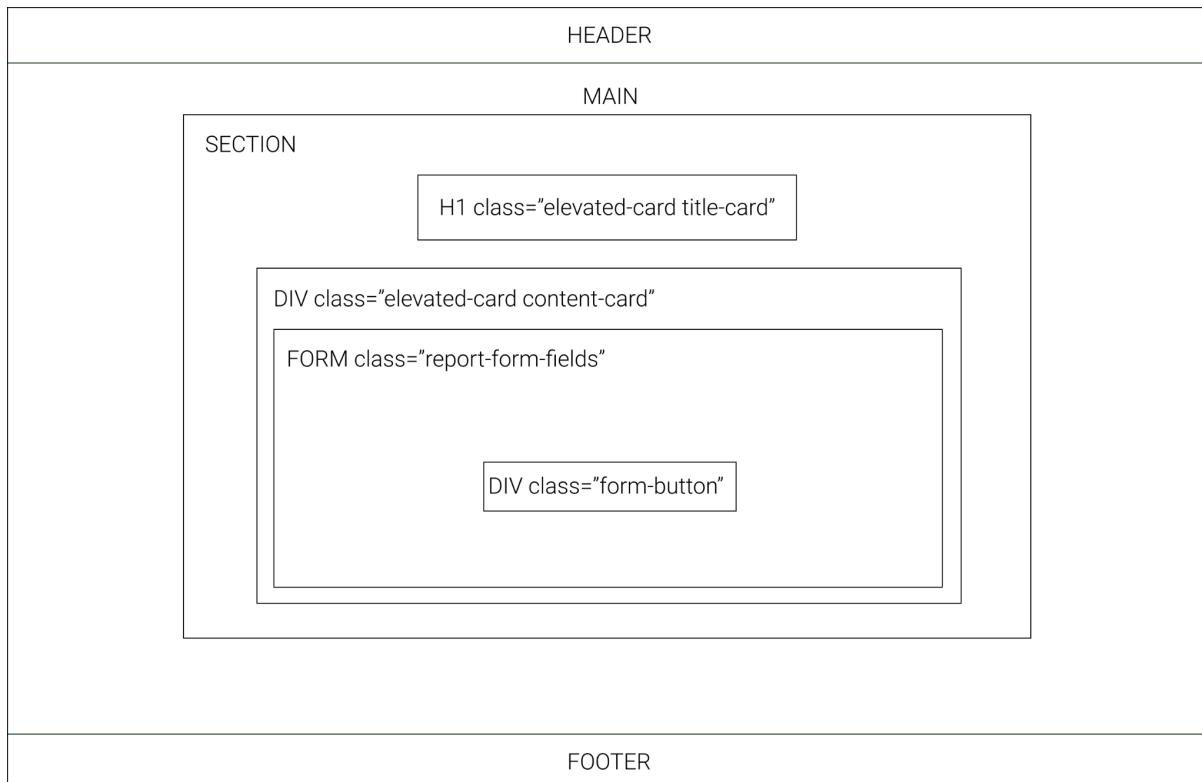


Leaderboard					
Country	Region	Total Reports	Time w/o Report	Last Reported	Trend
USA	North America	431	0 days	23-03-2025	+14%
China	Asia	398	0 days	23-03-2025	+11%
India	Asia	317	0 days	23-03-2025	+9%
Japan	Asia	256	0 days	23-03-2025	+6%
Spain	Europe	223	2 days	22-03-2025	+12%
Russia	Europe/Asia	211	0 days	23-03-2025	+7%
Germany	Europe	183	3 days	20-03-2025	-4%
Brazil	South America	189	0 days	23-03-2025	+9%
France	Europe	197	0 days	23-03-2025	+8%
Italy	Europe	168	1 days	22-03-2025	+6%
UK	Europe	162	1 days	22-03-2025	+4%
Australia	Oceania	142	3 days	20-03-2025	-2%
Portugal	Europe	145	5 days	18-03-2025	-2%
Canada	North America	122	2 days	21-03-2025	+3%
Netherlands	Europe	108	2 days	21-03-2025	+3%
Mexico	North America	98	4 days	19-03-2025	-1%

Por supuesto, la tabla es desplazable verticalmente en ordenador y vertical y horizontalmente el móvil; por lo que es 100% funcional en ambos dispositivos.

Contributors Page

Como siempre, las clases de los elementos HTML de esta página:



Grid

El elemento principal de esta página es un formulario que, para ser responsive, emplea *grid*. En pantallas grandes se organiza en tres columnas y dos filas, permitiendo que los campos de IP, categoría y botón estén alineados horizontalmente en la primera fila, mientras que el campo de descripción ocupa toda la fila inferior.



```
1 .report-form-fields {  
2   width: 100%;  
3   display: grid;  
4   gap: 16px;  
5   grid-template-columns: 2fr 2fr 1fr 1fr;  
6   grid-template-areas:  
7     "ip category . button"  
8     "textarea textarea textarea textarea";  
9 }
```



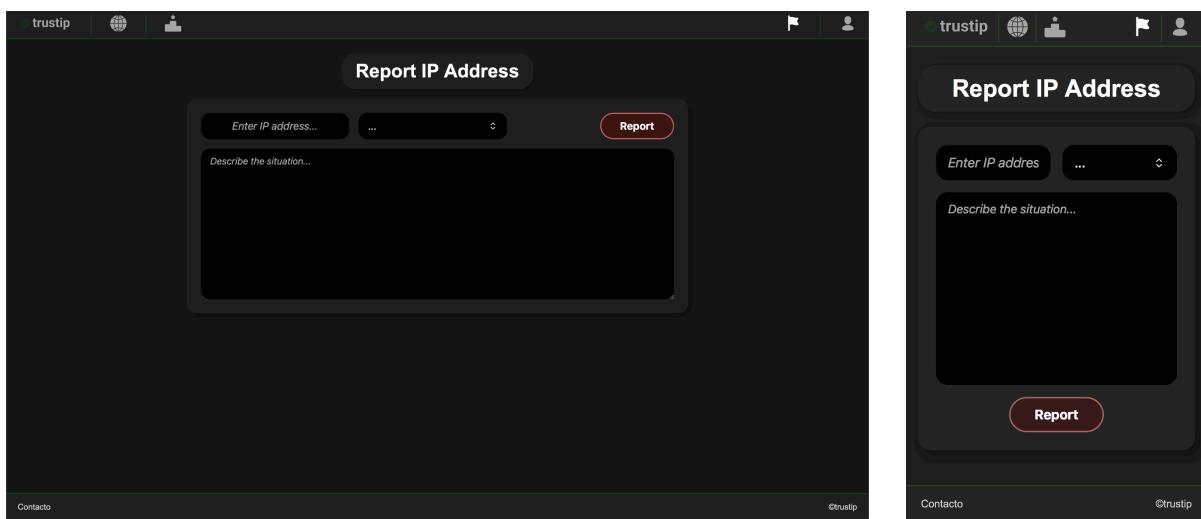
```
1 @media screen and (max-width: 600px) {  
2   .report-form-fields{  
3     grid-template-columns: 1fr 1fr;  
4     grid-template-areas:  
5       "ip category"  
6       "textarea textarea"  
7       "button button";  
8     }  
9     ...  
10 }
```

Sin embargo, para móviles, intercambiamos el número de columnas por el número de filas para conseguir un diseño más adaptado a las estrechas pantallas de los dispositivos móviles. Ahora nos encontramos con los campos de IP y categoría en la primera fila, el campo descripción en la segunda y el botón de reporte en la última.

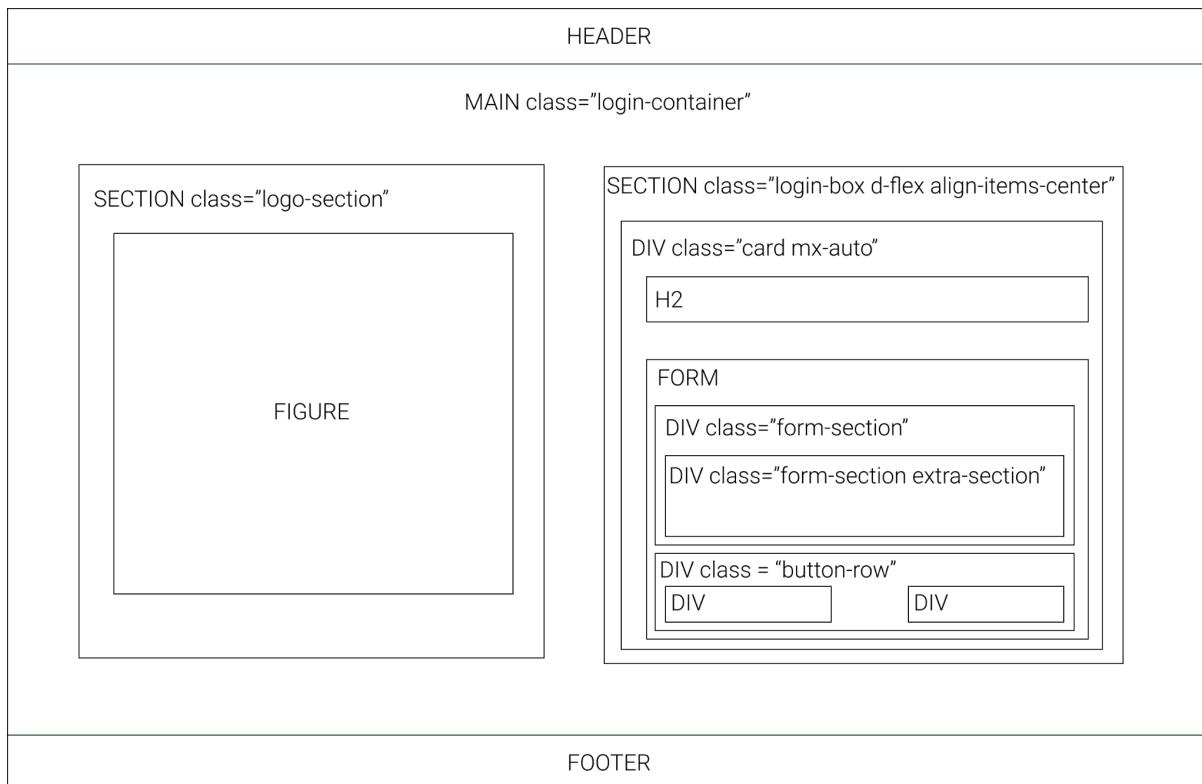
Para ambos dispositivos, la disposición de los elementos del formulario se gestiona mediante *grid-area*, lo cual se refleja tanto en la definición de columnas como en las áreas asignadas (*grid-area: ip, textarea, category, button*). Esta técnica permite un control detallado de la distribución de los elementos del formulario, y facilita la adaptación del diseño sin alterar la estructura HTML.

Aspecto final

Gracias al uso de *grid* en esta página, el resultado es el siguiente:



Login Page



Bootstrap

Esta página destaca por el uso de bootstrap para el diseño responsivo. Importamos la hoja de estilos de bootstrap desde la cabecera junto con su script y luego usamos las diferentes clases que proporciona bootstrap en nuestro archivo HTML junto con las clases de css para el estilo que queremos, para obtener el diseño responsivo.

Cabe destacar las siguientes clases usadas:

- **d-flex**: Para convertir contenedores en cajas flexibles. Se usa en header, footer y en el contenido principal para alinear los elementos horizontalmente.
- **align-items-center**: Alinea verticalmente los elementos dentro de un contenedor flex.
- **justify-content-between**: Distribuye el espacio entre los elementos de manera uniforme, colocándolos en los extremos.
- **ms-auto**: Margen automático al lado izquierdo (margin-start) para alinear elementos a la derecha.
- **mx-auto**: Centra elementos horizontalmente.
- **px-0, py-0, mt-2, pt-4, pb-3, etc.**: Clases de espaciado (padding y margin) propias de Bootstrap.

A continuación se adjunta una imagen del código HTML del header con las clases de Bootstrap añadidas:

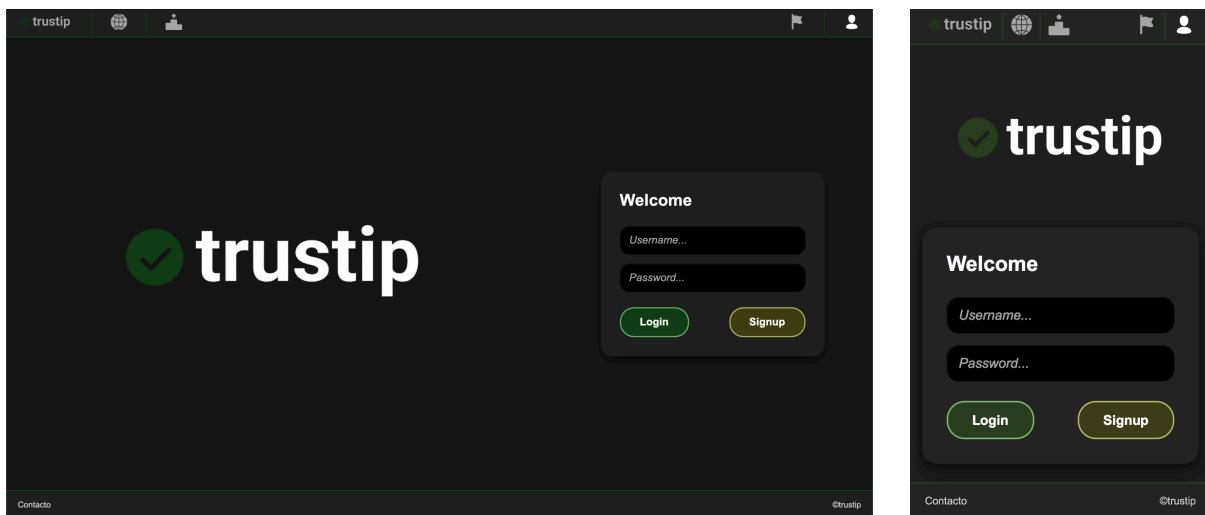
```
1 <header class="d-flex pt-4 pb-3">
2     <nav class="d-flex align-items-center justify-content-between">
3         <ul class="left-header d-flex px-0 mt-2">
4             <li>
5                 <a href="index.html">
6                     
7                 </a>
8             </li>
9             <li>
10                 <a href="globe.html">
11                     
12                 </a>
13             </li>
14             <li>
15                 <a href="leaderboard.html">
16                     
17                 </a>
18             </li>
19         </ul>
20         <ul class="right-header ms-auto d-flex px-0 mt-2">
21             <li>
22                 <a href="contributors.html">
23                     
24                 </a>
25             </li>
26             <li>
27                 <a href="login.html">
28                     
29                 </a>
30             </li>
31         </ul>
32     </nav>
33 </header>
```

A diferencia de las demás páginas, en este caso se utiliza una media query con `max-width: 1200px` específicamente diseñada para mejorar la visualización en dispositivos móviles y pantallas más pequeñas. Cuando se cumple esta condición, el contenedor principal `.login-container` cambia su disposición de fila a columna, posicionando el logo sobre el formulario. Además, el logo incrementa su ancho hasta el 80% para mantener su visibilidad y relevancia en pantallas reducidas. Finalmente, la tarjeta `.card` se ajusta para ocupar todo el ancho disponible del contenedor, asegurando una mejor adaptación al entorno móvil.

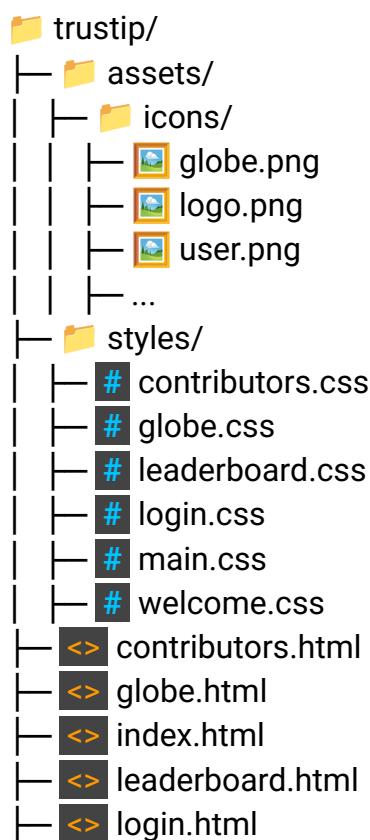
```
1 @media screen and (max-width: 1200px) {
2     .login-container{
3         flex-direction: column;
4     }
5
6     .logo-section img{
7         width: 80%;
8     }
9
10    .card{
11        width: 100%;
12    }
13
14 }
```

Aspecto final

Este es el aspecto final de la página:



Estructura de ficheros actualizada



JavaScript

Globe Page

Hemos llegado al elemento central de nuestra aplicación: la representación del globo terráqueo en 3D. Para esta parte del proyecto, decidimos ir un paso más allá de los requisitos establecidos, con el objetivo de desarrollar una funcionalidad que nos permitiera destacar de forma clara frente a otros proyectos y aportar un componente visual e interactivo verdaderamente diferencial.

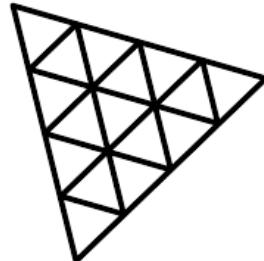
Si exploramos los archivos JavaScript vinculados a esta página, encontraremos la siguiente estructura:

```
libs/
├── orbit-controls.js
├── perlin-noise.js
├── THREE.MeshLine.js
└── trackball-controls.js
scripts/
└── globe/
    ├── app.js
    ├── config.js
    ├── globe.js
    ├── marker.js
    ├── markers.js
    ├── points.js
    ├── setup.js
    ├── shaders.js
    └── utils.js
    └── globe.js
```

Cada uno de estos archivos cumple un rol específico en el funcionamiento del globo 3D, y en los apartados siguientes se detalla su propósito. No obstante, antes de entrar en esos detalles, es fundamental comprender la tecnología base sobre la que se construye toda esta funcionalidad.

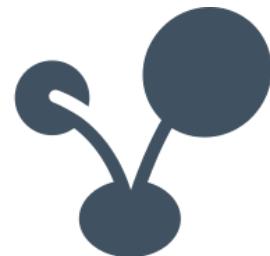
Three.js

Three.js es una librería de JavaScript liberada en GitHub en 2010 por el español Ricardo Cabello y que permite crear y renderizar gráficos 3D en el navegador utilizando WebGL, sin necesidad de escribir código gráfico complejo en bajo nivel. Ofrece una API sencilla e intuitiva para trabajar con escenas tridimensionales, cámaras, luces, materiales, geometrías, y animaciones. Gracias a Three.js, es posible desarrollar experiencias interactivas en 3D de manera accesible y altamente personalizable. En el caso de nuestra aplicación, esta tecnología es la base que hace posible representar el globo terráqueo y sus elementos visuales (como puntos, líneas, marcadores y animaciones) con un rendimiento fluido y atractivo.



Tween.js

Tween.js es una librería de JavaScript que permite crear transiciones suaves entre valores numéricos a lo largo del tiempo, ideal para animaciones personalizadas en aplicaciones web. Aunque es independiente, se utiliza frecuentemente junto a Three.js para animar propiedades como posiciones de cámara, escalas de objetos o efectos visuales en entornos 3D. En nuestra aplicación, se emplea para mover la cámara suavemente hacia el país seleccionado en el globo, aportando una navegación más fluida y agradable para el usuario.



libs/

La carpeta `libs/` contiene un conjunto de archivos JavaScript esenciales para la interacción 3D dentro de la página del globo terráqueo. Estos ficheros no han sido desarrollados desde cero, sino que son librerías externas o adaptaciones pensadas para complementar y extender las funcionalidades nativas de Three.js. A continuación, se detalla el propósito de cada uno:

orbit-controls.js

Este archivo permite controlar la cámara del entorno 3D mediante el ratón. Gracias a él, el usuario puede rotar, hacer zoom y desplazar la vista alrededor del globo con movimientos suaves e intuitivos. Es ideal para facilitar la exploración del contenido geográfico de la escena.

JS *trackball-controls.js*

Funciona como una alternativa a *orbit-controls.js*, proporcionando un estilo de navegación tipo trackball, con mayor libertad de movimiento en rotaciones. Este tipo de control puede ofrecer una experiencia más fluida o natural en dispositivos táctiles o interfaces interactivas avanzadas.

JS *perlin-noise.js*

Implementa algoritmos de ruido Perlin y Simplex, usados comúnmente para generar efectos visuales suaves y orgánicos, como animaciones de puntos, variaciones de color o simulaciones de movimiento natural en la superficie del globo.

JS *THREE.MeshLine.js*

Extiende Three.js añadiendo soporte para líneas con grosor variable. Mientras que Three.js solo permite dibujar líneas básicas sin grosor, esta librería permite representar trayectorias, conexiones o rutas en el globo de forma mucho más clara y visualmente atractiva, lo cual es clave para mostrar flujos entre IPs o regiones.

 **scripts/globe/**

En esta carpeta se encuentra el código fuente principal encargado de renderizar e interactuar con el globo terráqueo en 3D. Para mantener una arquitectura clara, escalable y fácil de mantener, el código se ha dividido en 9 scripts independientes, cada uno con una responsabilidad específica dentro del sistema.

Esta organización modular permite que cada archivo se encargue de una tarea concreta, como la configuración del entorno, la carga de datos, la creación de puntos, líneas, animaciones, entre otros aspectos clave del comportamiento del globo.

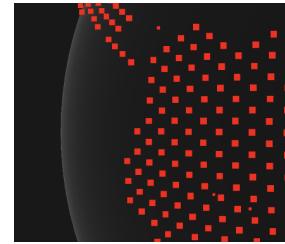
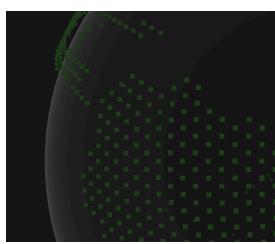
A continuación, se explicará cada uno de estos ficheros siguiendo el flujo de ejecución real: es decir, el orden en el que se van invocando o utilizando desde que el usuario abre la página del globo. De esta manera, se podrá comprender no sólo el propósito de cada script, sino también cómo colaboran entre sí para construir la experiencia interactiva final.

JS *config.js*

Define todas las constantes y estructuras de configuración del globo: rutas de texturas, tamaños, colores, escalas, comportamiento de rotación, elementos visibles, etc. Sirve como punto de referencia común para todos los scripts posteriores.

Por ejemplo, en este fragmento de código se puede observar cómo se asignan los colores a los puntos del globo, a los marcadores y a sus destellos. Si modificamos el atributo “globeDotColor” obtendremos este cambio en el globo:

```
1 colors: {  
2     globeDotColor: 'rgb(18, 61, 21)',  
3     globeMarkerColor: 'rgb(61, 23, 18)',  
4     globeMarkerGlow: 'rgb(187, 102, 102)',  
5 }
```



JS **shaders.js**

Contiene los shaders personalizados en GLSL para el globo, su atmósfera y los puntos. Estos shaders definen cómo se renderiza visualmente el globo, incluyendo efectos como brillo atmosférico y coloración dinámica.

- El shader del globo combina una textura base con un efecto de atmósfera difusa, calculando la intensidad según el ángulo de visión (usando la normal del vértice).
- El shader de la atmósfera crea un halo sutil alrededor del planeta, simulando el efecto de dispersión de la luz.
- El shader de puntos permite colorearlos individualmente y aplicarles una textura circular para lograr un acabado más natural en su representación.

JS **app.js**

La clase App actúa como componente central y coordinador del sistema de visualización 3D basado en Three.js. Se encarga de inicializar los elementos fundamentales de la escena, incluyendo:

- La creación de una instancia de THREE.Scene.
- La configuración del *renderer*, que define cómo se renderiza el contenido en la pantalla.
- La inicialización de la cámara con perspectiva y su posicionamiento inicial.
- La activación de los *OrbitControls*, que permiten al usuario interactuar con el globo mediante rotación (aunque en este caso con limitaciones definidas).

Además, implementa un bucle de animación personalizado utilizando `requestAnimationFrame`, en el que se controla el tiempo transcurrido entre frames para lograr una animación fluida, constante y desacoplada del framerate del sistema (uso de `deltaTime`). También incorpora un método `handleResize` para ajustar la cámara y el renderizado cuando cambia el tamaño de la ventana.

En resumen, App es el núcleo del flujo de ejecución del globo 3D: prepara el entorno, lanza la animación, y gestiona dinámicamente las interacciones y actualizaciones visuales necesarias durante toda la experiencia.

JS **setup.js**

Es el punto de entrada principal del sistema de visualización del globo 3D. Su función es orquestar la inicialización y configuración de la escena, así como el comportamiento interactivo y animado del entorno. Es aquí donde se instancia la clase App, que centraliza el renderizado y la animación:

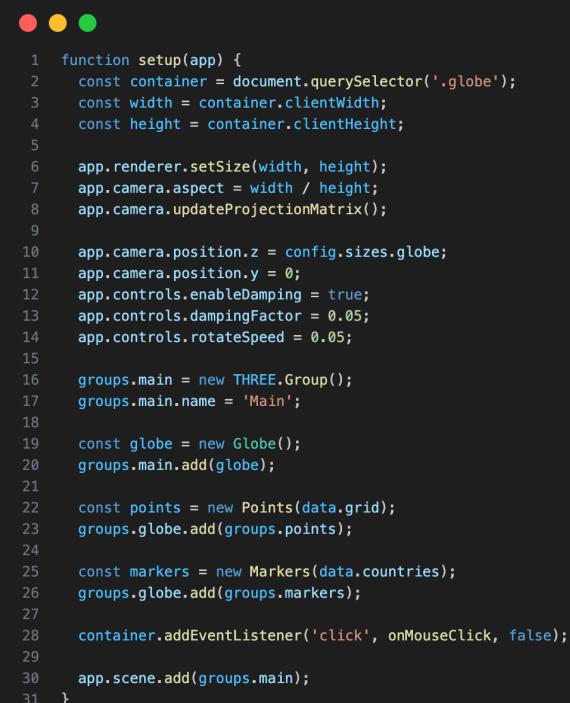


```
1 const app = new App({ setup, animate, preload });
2 window.onload = app.init;
3 window.onresize = app.handleResize;
```

Este archivo define tres funciones clave que se pasan como argumentos al constructor de App:

setup(app) → Esta función se ejecuta al inicio y es responsable de:

- Ajustar el tamaño del renderer y la cámara.
- Posicionar la cámara en el eje Z para enfocar el globo.
- Aplicar suavizado al movimiento de cámara (enableDamping).
- Crear los grupos 3D principales como *Main*, *Globe*, *Points* y *Markers*.
- Enlazar el evento de clic sobre los marcadores.



```
1 function setup(app) {
2   const container = document.querySelector('.globe');
3   const width = container.clientWidth;
4   const height = container.clientHeight;
5
6   app.renderer.setSize(width, height);
7   app.camera.aspect = width / height;
8   app.camera.updateProjectionMatrix();
9
10  app.camera.position.z = config.sizes.globe;
11  app.camera.position.y = 0;
12  app.controls.enableDamping = true;
13  app.controls.dampingFactor = 0.05;
14  app.controls.rotateSpeed = 0.05;
15
16  groups.main = new THREE.Group();
17  groups.main.name = 'Main';
18
19  const globe = new Globe();
20  groups.main.add(globe);
21
22  const points = new Points(data.grid);
23  groups.globe.add(groups.points);
24
25  const markers = new Markers(data.countries);
26  groups.globe.add(groups.markers);
27
28  container.addEventListener('click', onMouseClick, false);
29
30  app.scene.add(groups.main);
31 }
```

Al final de setup se llama añade como listener del container principal del globo 3D la siguiente función:

```
● ● ●  
1  function onMouseClick(event) {  
2      event.preventDefault();  
3  
4      const container = document.querySelector('.globe');  
5      const rect = container.getBoundingClientRect();  
6      const mouse = new THREE.Vector2();  
7      mouse.x = ((event.clientX - rect.left) / rect.width) * 2 - 1;  
8      mouse.y = -((event.clientY - rect.top) / rect.height) * 2 + 1;  
9  
10     const raycaster = new THREE.Raycaster();  
11     raycaster.setFromCamera(mouse, app.camera);  
12     const intersects = raycaster.intersectObjects(groups.markers.children, true);  
13  
14     if (intersects.length > 0) {  
15         let markerObject = intersects[0].object;  
16         while (markerObject && markerObject.name !== 'Marker') {  
17             markerObject = markerObject.parent;  
18         }  
19         const countryName = markerObject ? markerObject.userData.countryName : null;  
20         if (countryName) {  
21             selectCountry(countryName);  
22         }  
23     }  
24 }
```

Esta función de evento es especialmente interesante para nosotros. Es la responsable de que cuando hagamos clic en uno de los marcadores del globo, se obtenga el nombre del país representado por ese marcador.

En primer lugar, la función bloquea el comportamiento por defecto del clic para evitar interferencias con la interacción 3D. Luego, calcula la posición exacta del clic dentro del contenedor del globo, transformándola en coordenadas normalizadas, que son necesarias para trabajar con el sistema de detección de objetos de Three.js.

Una vez obtenidas estas coordenadas, la función lanza un “rayo invisible” desde la cámara hacia la escena, pasando por el punto donde el usuario ha hecho clic. Este rayo se utiliza para detectar si ha colisionado con alguno de los objetos interactivos, en este caso, los marcadores del globo.

Si el rayo colisiona con un marcador, la función recorre su jerarquía hasta encontrar el grupo principal que lo representa (ya que un marcador está compuesto por varios subelementos visuales como el punto, el brillo o la etiqueta). Al encontrar el

marcador principal, se extrae el nombre del país que tiene guardado en sus datos personalizados.

Finalmente, si se ha podido identificar correctamente un país, se llama a una función externa encargada de gestionar la selección. Esta función despliega por pantalla una tarjeta mostrando información del país seleccionado y además llama a la siguiente función:



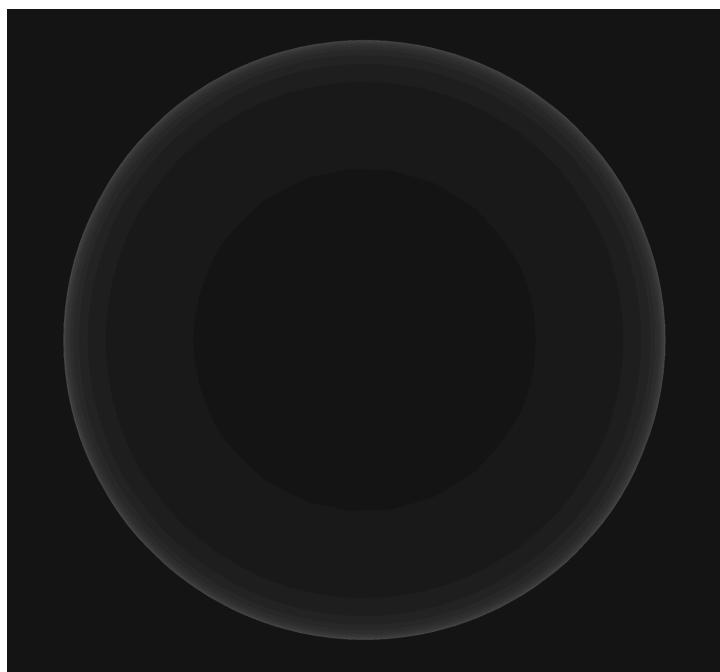
```
1 function goToCountry(info) {
2     animations.rotateGlobe = false;
3
4     const lat = parseFloat(info.latitude);
5     const lon = parseFloat(info.longitude);
6
7     const currentDistance = app.camera.position.length();
8
9     const targetPos = latLngToCameraPosition(lat, lon, currentDistance);
10
11    const startPos = {
12        x: app.camera.position.x,
13        y: app.camera.position.y,
14        z: app.camera.position.z
15    };
16
17    new TWEEN.Tween(startPos)
18        .to({ x: targetPos.x, y: targetPos.y, z: targetPos.z }, 1500)
19        .easing(TWEEN.Easing.Quadratic.Out)
20        .onUpdate(() => {
21            app.camera.position.set(startPos.x, startPos.y, startPos.z);
22            app.controls.update();
23        })
24        .start();
25
26    app.controls.target.set(0, 0, 0);
27    app.controls.update();
28}
```

goToCountry se encarga de mover suavemente la cámara hasta la posición correspondiente al país seleccionado previamente. Para lograr esto desactiva temporalmente la rotación automática del globo, calcula la posición de destino a partir de las coordenadas geográficas, y anima la transición de la cámara desde su posición actual hasta esa nueva ubicación mediante la librería *Tween.js*, logrando un efecto de desplazamiento fluido. Finalmente, actualiza el control de cámara para mantener el enfoque centrado en el globo.

animate(app) → La función `animate` se encarga de gestionar todas las actualizaciones visuales que deben aplicarse en cada fotograma del renderizado del globo 3D. Se ejecuta de forma continua dentro del bucle de animación y se apoya en la librería `Tween.js` para mantener las transiciones suaves entre estados. En ella se actualizan aspectos visuales como el tamaño, color y visibilidad de los puntos, líneas y marcadores, en función de la configuración actual. Además, si la rotación automática del globo está activada, esta función también se encarga de mover la cámara alrededor de la esfera. Gracias a esta función, se garantiza que el estado visual del globo esté siempre sincronizado con la interacción del usuario y con los parámetros de configuración definidos.

JS globe.js

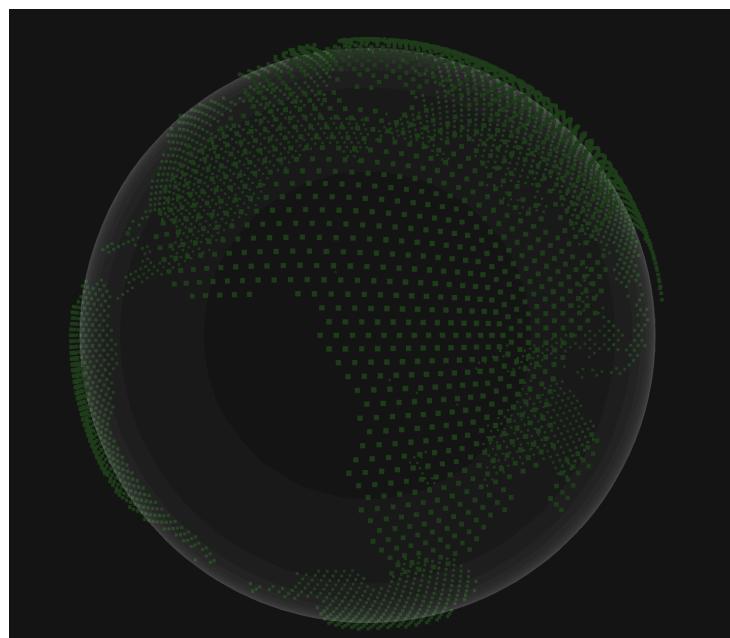
Este archivo crea el objeto principal del globo 3D utilizando una esfera generada con `Three.js`. A dicha esfera se le aplica un material personalizado definido mediante shaders (contenidos en `shaders.js`), lo que permite un acabado visual sofisticado y altamente configurable. Además, incorpora una atmósfera envolvente que simula un halo de luz alrededor del planeta, también generado mediante shaders específicos.



En la imagen anterior, puede verse el globo vacío renderizado con su atmósfera, sirviendo de base para el resto de elementos.

JS *points.js*

Este componente se encarga de representar una nube de puntos estáticos distribuidos sobre la superficie del globo. Cada punto se coloca a partir de coordenadas geográficas (latitud y longitud) y se transforma a coordenadas tridimensionales usando funciones matemáticas personalizadas. Los puntos se renderizan mediante THREE.Points, y cada uno incluye propiedades visuales como color y tamaño que pueden ajustarse dinámicamente.

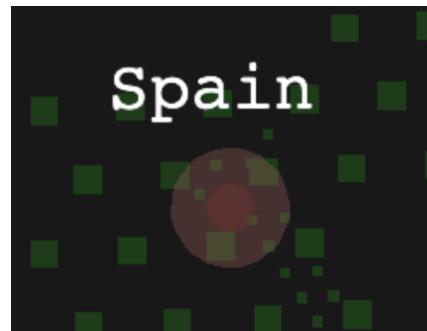


En la imagen asociada, se aprecia el globo cubierto por una capa uniforme de puntos, aportando textura y profundidad visual.

JS *marker.js*

Este archivo define la clase Marker, utilizada para construir un marcador individual que representa un país o una ubicación relevante. Cada marcador está compuesto por un punto, un efecto de brillo (glow) y una etiqueta visible con el nombre del país. Los elementos se agrupan y se posicionan en la escena según sus coordenadas geográficas, lo que permite su posterior interacción y animación.

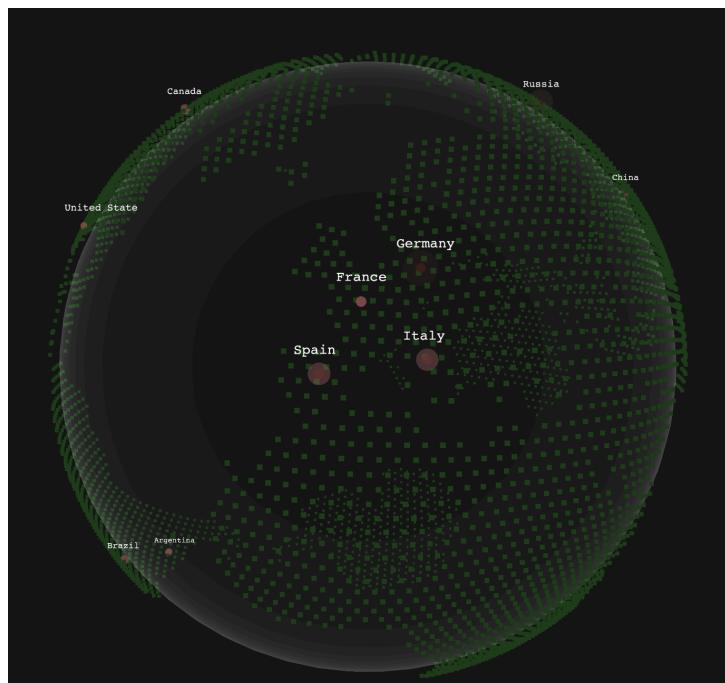
A continuación se muestra un marcador individual sobre el globo, mostrando su etiqueta y su brillo característico.



JS *markers.js*

Este archivo gestiona la creación masiva de marcadores sobre el globo. Utiliza los datos cargados previamente en la función preload (explicada posteriormente en script/globe.js) para colocar un marcador por cada país con latitud y longitud definidas. Cada marcador se crea usando la clase Marker y se añade al grupo global groups.markers. Esta clase centraliza y automatiza la colocación de todos los elementos interactivos que representan países.

La imagen siguiente muestra el globo poblado por múltiples marcadores, cada uno ubicado en su país correspondiente, listos para ser seleccionados o animados.



JS **utils.js**

Este archivo (`utils.js`) contiene un conjunto de funciones utilitarias destinadas principalmente a realizar conversiones de coordenadas geográficas (latitud y longitud) a coordenadas tridimensionales en el contexto del globo 3D. Estas funciones permiten transformar puntos geográficos en posiciones sobre la esfera, posicionar la cámara en función de coordenadas, y gestionar su rotación automática alrededor del globo. También incluye funciones auxiliares para limitar valores (`clamp`) e imprimir la posición actual de la cámara. Todas estas herramientas contribuyen a una representación y navegación precisa y fluida en la escena.

JS **scripts/globe**

Este archivo actúa como punto de entrada y coordinador de la interfaz interactiva de la página del globo 3D en el proyecto *trustip*. Es el encargado de cargar datos, gestionar eventos de búsqueda e interacción, y lanzar la carga secuencial de scripts necesarios para que el globo funcione correctamente. Se podría considerar como el archivo que une la lógica de “backend” (datos) con la presentación visual (renderizado en *Three.js*).

Carga dinámica de scripts

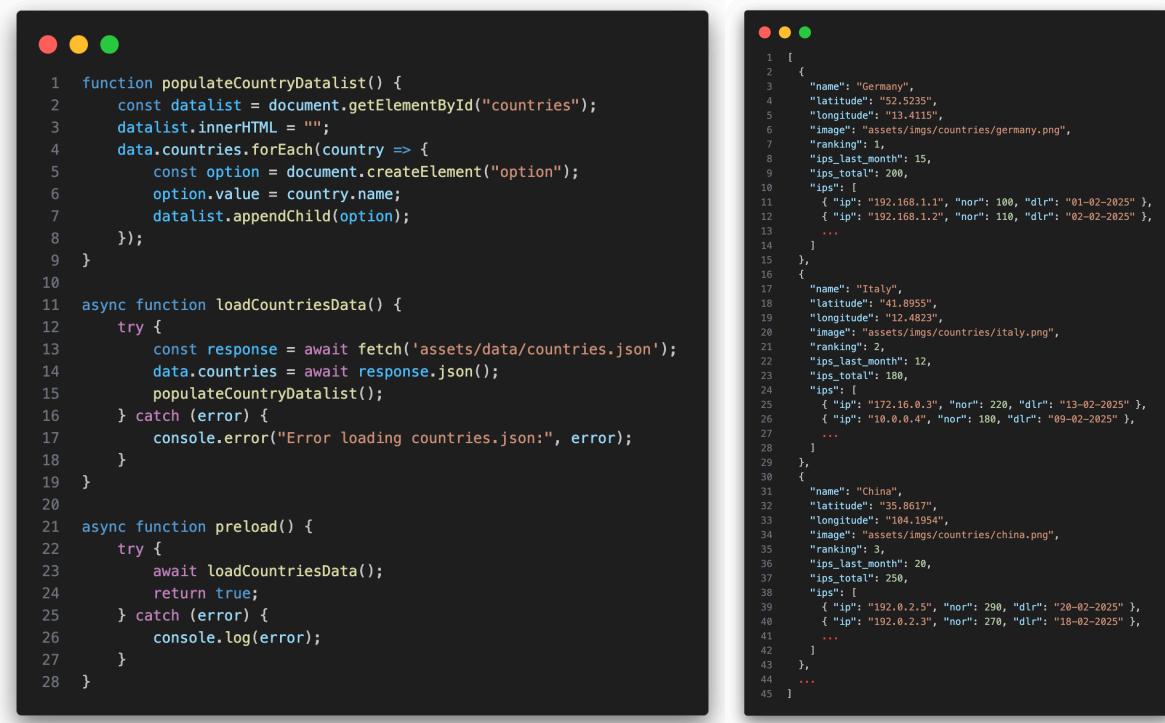
En primer lugar, se define un array llamado `scriptUrls`, que contiene las rutas de todos los scripts que deben ser cargados secuencialmente. Este array incluye librerías externas como *Three.js* o *Tween.js*, y varios scripts locales del propio sistema.

La función `loadScriptsSequentially()` se encarga de cargar esos scripts uno por uno en orden, garantizando que cada uno esté completamente cargado antes de proceder al siguiente. Esto es importante porque muchos scripts dependen unos de otros y deben respetar un orden lógico de ejecución.

Esta carga solo se activa si el ancho de la pantalla es mayor a 600 píxeles, es decir, en dispositivos no móviles; ya que en los móviles el globo terráqueo no se desplegará por motivos de imprecisión para la selección de países.

`preload()`

Se define y se llama a la función `preload()`, esta se encarga de cargar los datos de los países que se mostrarán sobre el globo terráqueo 3D y en las tarjetas. Para ello, se apoya en la función `fetch()` de la API nativa de JavaScript, conocida como Fetch API, que permite realizar peticiones HTTP de forma asíncrona y sencilla.



```

1  function populateCountryDatalist() {
2      const datalist = document.getElementById("countries");
3      datalist.innerHTML = "";
4      data.countries.forEach(country => {
5          const option = document.createElement("option");
6          option.value = country.name;
7          datalist.appendChild(option);
8      });
9  }
10
11 async function loadCountriesData() {
12     try {
13         const response = await fetch('assets/data/countries.json');
14         data.countries = await response.json();
15         populateCountryDatalist();
16     } catch (error) {
17         console.error("Error loading countries.json:", error);
18     }
19 }
20
21 async function preload() {
22     try {
23         await loadCountriesData();
24         return true;
25     } catch (error) {
26         console.log(error);
27     }
28 }

```

```

1 [
2   {
3     "name": "Germany",
4     "latitude": "52.5235",
5     "longitude": "13.4115",
6     "image": "assets/imgs/countries/germany.png",
7     "ranking": 1,
8     "ips_last_month": 15,
9     "ips_total": 200,
10    "ips": [
11      { "ip": "192.168.1.1", "nor": 100, "dlr": "01-02-2025" },
12      { "ip": "192.168.1.2", "nor": 110, "dlr": "02-02-2025" },
13      ...
14    ],
15  },
16  {
17    "name": "Italy",
18    "latitude": "41.8955",
19    "longitude": "12.4823",
20    "image": "assets/imgs/countries/italy.png",
21    "ranking": 2,
22    "ips_last_month": 12,
23    "ips_total": 180,
24    "ips": [
25      { "ip": "172.16.0.3", "nor": 220, "dlr": "13-02-2025" },
26      { "ip": "10.0.0.4", "nor": 180, "dlr": "09-02-2025" },
27      ...
28    ],
29  },
30  {
31    "name": "China",
32    "latitude": "35.8617",
33    "longitude": "104.1954",
34    "image": "assets/imgs/countries/china.png",
35    "ranking": 3,
36    "ips_last_month": 28,
37    "ips_total": 250,
38    "ips": [
39      { "ip": "192.0.2.5", "nor": 290, "dlr": "20-02-2025" },
40      { "ip": "192.0.2.3", "nor": 270, "dlr": "18-02-2025" },
41      ...
42    ],
43  },
44  ...
45 ]

```

Dentro de la función, se realiza una petición al archivo `countries.json`, ubicado en la carpeta `assets/data/`. Este archivo contiene un array de objetos, donde cada objeto representa un país y almacena diversos datos como el nombre, latitud, longitud, imagen representativa, ranking, número de IPs detectadas y una lista de direcciones IP asociadas, como puede verse en la imagen correspondiente.

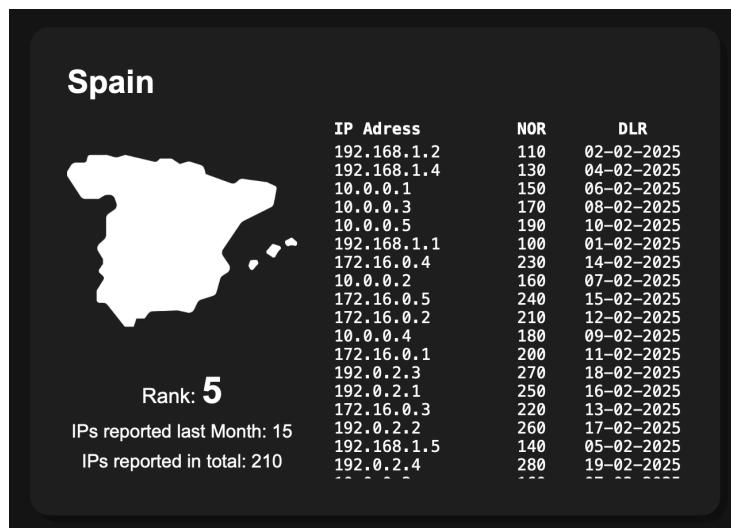
Una vez que los datos han sido cargados correctamente, se almacenan en `data.countries`. A continuación, se llama a la función `populateCountryDatalist()`, que se encarga de poblar dinámicamente el elemento `<datalist>` del HTML. Esta función recorre todos los países del JSON y crea una opción (`<option>`) por cada uno de ellos, utilizando su nombre como valor. Estas opciones se insertan en el `datalist`, que está vinculado al campo de búsqueda de la página, tal y como se explicó en el apartado de HTML de este documento.

Gracias a esta lógica, se implementa un sistema de búsqueda con autocompletado, donde el usuario puede comenzar a escribir el nombre de un país y recibir sugerencias en tiempo real, facilitando así la navegación y mejorando notablemente la experiencia de uso.

Despliegue de tarjetas

Una vez seleccionado un país, ya sea a través del buscador o haciendo clic sobre un marcador en el globo, entra en acción la función `selectCountry`, una de las piezas clave del archivo. Esta función localiza el país correspondiente dentro del objeto `data.countries` y actualiza dinámicamente la interfaz con la información asociada. En la tarjeta informativa se reemplazan el nombre del país, su imagen representativa, el ranking actual, el número total de IPs detectadas y otros detalles relevantes.

Paralelamente, la función `updateIpTable` se encarga de actualizar la tabla de direcciones IP ubicada en la parte inferior de la tarjeta. Esta tabla se genera dinámicamente a partir del listado de IPs del país y muestra, para cada una, su dirección, el número de reportes recibidos (nor) y la fecha del último registro (dlr). Esto proporciona al usuario una visión rápida y clara de la actividad asociada a ese país, enriqueciendo la experiencia interactiva con datos específicos y relevantes.



En caso de que sea la primera vez que se visualiza esta tarjeta, se aplica una animación CSS para mostrarla de forma gradual, aportando una transición visual agradable. Además, se invoca la función `goToCountry(info)`, definida en otro archivo, que, como se ha explicado anteriormente, se encarga de animar la cámara del globo para centrar la vista exactamente sobre la posición geográfica del país seleccionado. Esto permite que la interacción entre los datos y el modelo 3D sea completamente fluida y coherente.

```
1  function updateIpTable(ips) {
2      const tbody = document.querySelector('.ip-scroll table tbody');
3      tbody.innerHTML = "";
4      ips.forEach(entry => {
5          const tr = document.createElement('tr');
6          tr.innerHTML = `
7              <td>${entry.ip}</td>
8              <td>${entry.nor}</td>
9              <td>${entry.dlr}</td>
10         `;
11         tbody.appendChild(tr);
12     });
13 }
14
15 function selectCountry(countryName) {
16     const info = data.countries.find(c => c.name.toLowerCase() === countryName.toLowerCase());
17     if (info) {
18         console.log("Selected Country:", info.name);
19
20         const card = document.querySelector('.country-card');
21         const countryInfo = document.querySelector('.country-info');
22         const countryTitle = document.querySelector('.country-card h2');
23
24         countryTitle.textContent = info.name;
25         const countryImage = document.querySelector('.map-stats img');
26         countryImage.src = info.image;
27         countryImage.alt = info.name;
28         document.querySelector('.map-stats .rank span').textContent = "Rank: ";
29         document.querySelector('.map-stats .rank strong').textContent = info.ranking;
30         document.querySelector('.map-stats p:nth-child(3)').textContent =
31             "IPs reported last Month: " + info.ips_last_month;
32         document.querySelector('.map-stats p:nth-child(4)').textContent =
33             "IPs reported in total: " + info.ips_total;
34         updateIpTable(info.ips);
35
36         if (!cardAnimated) {
37             card.classList.add('show');
38             card.addEventListener('animationend', function handler() {
39                 countryInfo.classList.add('show');
40                 card.removeEventListener('animationend', handler);
41                 cardAnimated = true;
42             });
43         } else {
44             countryInfo.classList.remove('show');
45             void countryInfo.offsetWidth;
46             countryInfo.classList.add('show');
47         }
48
49         goToCountry(info);
50         document.querySelector('.search-form input[name="search"]').blur();
51
52     } else {
53         console.log("Country not found:", countryName);
54     }
55 }
```

Eventos para el buscador

Finalmente, el archivo también gestiona los eventos asociados al campo de búsqueda, permitiendo al usuario interactuar de manera flexible con el sistema.



```
1 const searchForm = document.querySelector('.search-form');
2 searchForm.addEventListener('submit', function(event) {
3     event.preventDefault();
4     const searchInput = document.querySelector('.search-form input[name="search"]');
5     const query = searchInput.value.trim();
6     if (query) {
7         selectCountry(query);
8     }
9 });
10
11 const searchInput = document.querySelector('.search-form input[name="search"]');
12 searchInput.addEventListener('change', function(event) {
13     const query = event.target.value.trim();
14     if (query) {
15         selectCountry(query);
16     }
17 });
18
19 searchInput.addEventListener('keydown', function(event) {
20     if (event.key === 'Tab') {
21         event.preventDefault();
22         const currentText = this.value.trim().toLowerCase();
23         const datalist = document.getElementById("countries");
24         let candidate = null;
25         if (datalist) {
26             for (let i = 0; i < datalist.options.length; i++) {
27                 const optionValue = datalist.options[i].value;
28                 if (optionValue.toLowerCase().startsWith(currentText)) {
29                     candidate = optionValue;
30                     break;
31                 }
32             }
33             if (candidate) {
34                 this.value = candidate;
35             }
36         } else if (event.key === 'Enter') {
37             event.preventDefault();
38             selectCountry(this.value.trim());
39         }
40     }
41 });


```

Se configuran tres tipos de eventos principales:

- Evento de envío del formulario (submit): se activa cuando el usuario pulsa Enter o hace clic en el botón de búsqueda, y lanza inmediatamente la búsqueda del país escrito.
- Evento de cambio (change): se ejecuta cuando el usuario selecciona una opción del autocomplete del datalist, lo que permite detectar rápidamente la selección sin necesidad de confirmar manualmente pulsando enter o el botón de búsqueda.

- Evento de pulsación de teclas (keydown): si el usuario pulsa la tecla `tab`, se completa el campo de búsqueda con la primera coincidencia disponible del `datalist`, facilitando una escritura más rápida.

Todos estos eventos finalizan llamando a la función `selectCountry(query)`, explicada justo en el apartado anterior.

Leaderboard page

En esta página implementamos dos efectos visibles usando JQuery y JES6. Además cargamos los datos de la tabla de un archivo XML.

```

● ● ●
1 $(document).ready(() => {
2   const tbody = $('#leaderboard-body');
3
4   $.ajax({
5     type: 'GET',
6     url: 'assets/data/countries.xml',
7     dataType: 'xml',
8     success: (xml) => {
9       $(xml).find('country').each((index, element) => {
10         const name = $(element).find('name').text();
11         const region = $(element).find('region').text();
12         const total = $(element).find('total').text();
13         const time = $(element).find('time').text();
14         const last = $(element).find('last').text();
15         const trend = $(element).find('trend').text();
16
17         const row = $(
18           <tr class="animated-row">
19             <td>${name}</td>
20             <td>${region}</td>
21             <td>${total}</td>
22             <td>${time}</td>
23             <td>${last}</td>
24             <td>${trend}</td>
25           </tr>
26         );
27
28         setTimeout(() => {
29           $('#leaderboard-body').append(row);
30           index * 100);
31
32         row.on('animationend', () => {
33           row.removeClass('animated-row');
34         });
35       });
36     },
37     error: (xhr) => {
38       console.error('Error loading XML data', xhr.status, xhr.statusText);
39     }
40   });
41 });
42 });

```

El primero de los métodos es el que podemos ver en la imagen. Se encarga de cargar los datos del XML mediante el uso de JQuery Ajax. Se busca en el archivo con `.find` los campos de cada una de las entradas de la tabla, se genera el código HTML asociado a esa fila y por último se carga en el DOM con `.append`. En este caso al cargarlo aplicamos una animación que explicaremos más adelante.

Para este método usamos los estándares definidos en JES6. Un ejemplo en este código sería el uso del tipo de variable `const` o el uso de Arrow functions.

Además también se tuvieron que realizar modificaciones en el código del HTML donde anteriormente aparecía declarada toda la información de la tabla, ahora solo aparece la cabecera declarada ya que el body se cargará dinámicamente desde el archivo de javascript.

```

● ● ●
1 <table>
2   <thead>
3     <tr>
4       <th>Country</th>
5       <th>Region</th>
6       <th>Total Reports</th>
7       <th>Time w/o Report</th>
8       <th>Last Reported</th>
9       <th>Trend</th>
10      </tr>
11    </thead>
12    <tbody id="leaderboard-body">
13    </tbody>
14 </table>

```

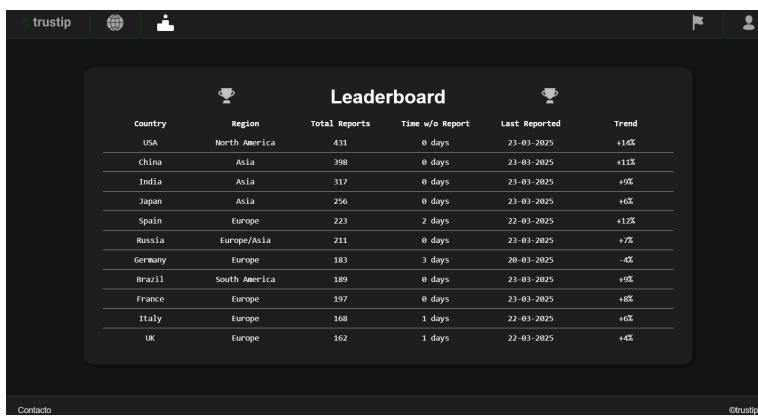
A continuación vamos a explicar el código del efecto que mencionamos anteriormente. Consiste en que en vez de cargar todas las filas de golpe al entrar en la página, irán apareciendo una a una hacia abajo. Para esto modificamos el archivo css añadiendo el estilo de la animación desde ahí y en nuestro archivo de js lo llamamos desde una función desde una Arrow function. También hay que destacar que una vez terminada la animación se elimina el efecto para que no haya conflictos con los otros efectos de la página.

```

● ○ ●
1 @keyframes slideFadeIn {
2   from {
3     opacity: 0;
4     transform: translateY(15px);
5   }
6   to {
7     opacity: 1;
8     transform: translateY(0);
9   }
10 }
11
12 .animated-row {
13   opacity: 0;
14   animation: slideFadeIn 0.4s ease forwards;
15 }

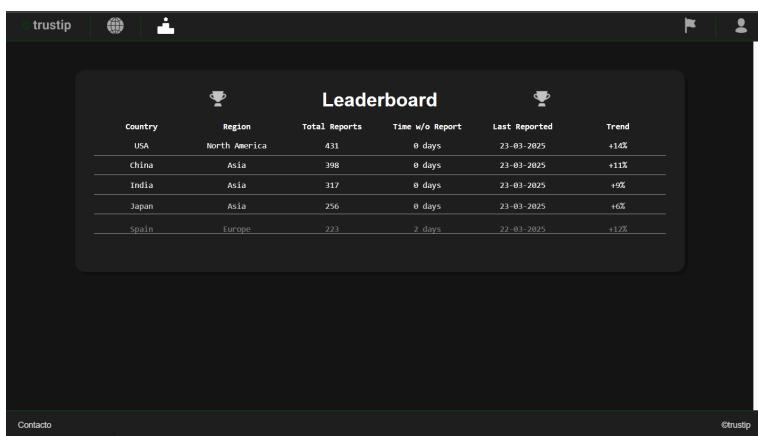
```

Antes del efecto, al entrar en la página se veía la tabla completamente cargada tal como aparece en la imagen a continuación.



Leaderboard					
Country	Region	Total Reports	Time w/o Report	Last Reported	Trend
USA	North America	431	0 days	23-03-2025	+10%
China	Asia	398	0 days	23-03-2025	+11%
India	Asia	317	0 days	23-03-2025	+9%
Japan	Asia	256	0 days	23-03-2025	+6%
Spain	Europe	223	2 days	22-03-2025	+12%
Russia	Europe/Asia	211	0 days	23-03-2025	+7%
Germany	Europe	183	3 days	20-03-2025	-4%
Brazil	South America	189	0 days	23-03-2025	+9%
France	Europe	197	0 days	23-03-2025	+8%
Italy	Europe	168	1 days	22-03-2025	+6%
UK	Europe	162	1 days	22-03-2025	+4%

Con el efecto cada las filas van apareciendo una a una. En la siguiente imagen podemos observar el estado intermedio de la animación.



Leaderboard					
Country	Region	Total Reports	Time w/o Report	Last Reported	Trend
USA	North America	431	0 days	23-03-2025	+10%
China	Asia	398	0 days	23-03-2025	+11%
India	Asia	317	0 days	23-03-2025	+9%
Japan	Asia	256	0 days	23-03-2025	+6%
Spain	Europe	223	2 days	22-03-2025	+12%

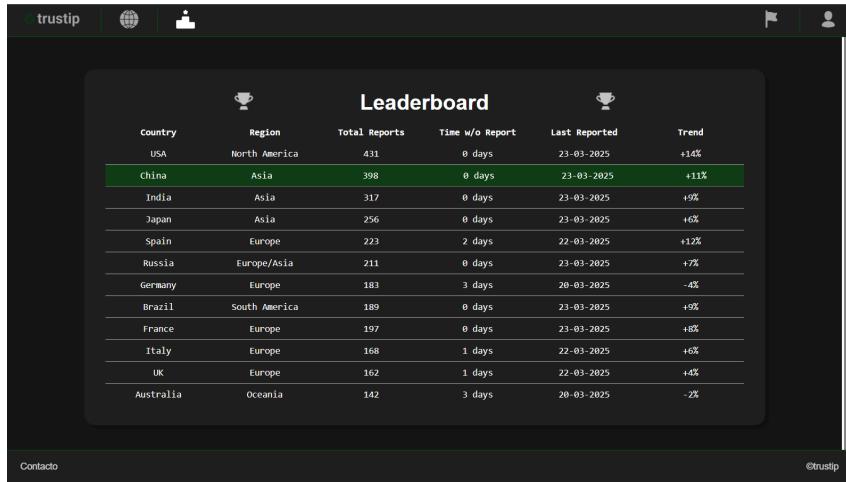
```

1  $(document).ready(() => {
2      const highlightColor = '#123D15';
3
4      $('#leaderboard-body').on('mouseenter', 'tr', function () {
5          $(this).css({
6              'background-color': highlightColor,
7              'transform': 'scale(1.02)',
8              'transition': 'all 0.2s ease',
9              'z-index': 1,
10         });
11     });
12
13     $('#leaderboard-body').on('mouseleave', 'tr', function () {
14         $(this).css({
15             'background-color': '',
16             'transform': 'scale(1)',
17             'transition': 'transform 0.2s ease, background-color 0.2s ease',
18             'z-index': '',
19         });
20     });
21 });

```

El segundo efecto se encuentra en una nueva *Arrow function*. Cuando el usuario pasa el ratón por encima de cualquier fila de la tabla, esta se resalta visualmente con un cambio de fondo y un pequeño efecto de zoom. Al cambiar el *focus* del ratón, vuelve a su estado original.

Este efecto se consigue mediante el uso de los eventos `mouseenter` y `mouseleave`, donde se modifican los estilos de cada fila. Podemos observar la animación en la siguiente imagen.



Leaderboard					
Country	Region	Total Reports	Time w/o Report	Last Reported	Trend
USA	North America	431	0 days	23-03-2025	+14%
China	Asia	398	0 days	23-03-2025	+11%
India	Asia	317	0 days	23-03-2025	+9%
Japan	Asia	256	0 days	23-03-2025	+6%
Spain	Europe	223	2 days	22-03-2025	+12%
Russia	Europe/Asia	211	0 days	23-03-2025	+7%
Germany	Europe	183	3 days	20-03-2025	-4%
Brazil	South America	189	0 days	23-03-2025	+9%
France	Europe	197	0 days	23-03-2025	+8%
Italy	Europe	168	1 days	22-03-2025	+6%
UK	Europe	162	1 days	22-03-2025	+4%
Australia	Oceania	142	3 days	20-03-2025	-2%

Contributors Page

En el script de esta página se introduce un efecto visual para los campos del formulario y una validación dinámica.

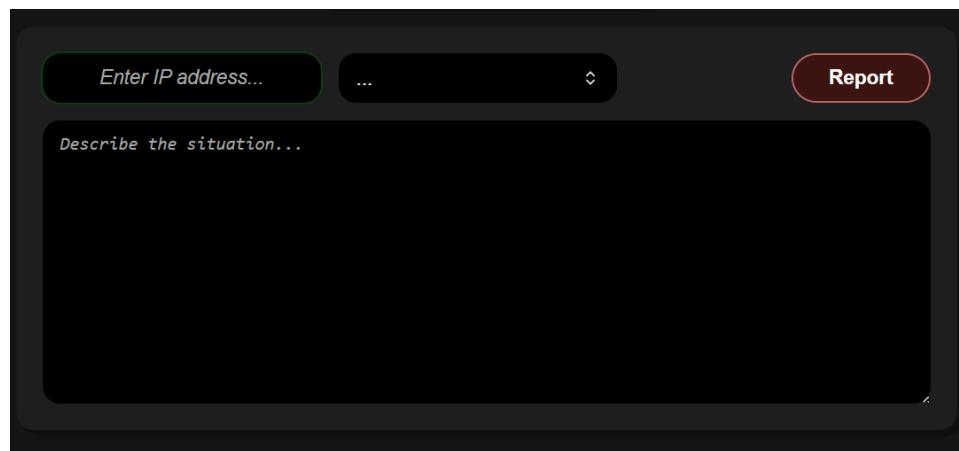
Cuando un usuario hace click en cualquiera de los campos del formulario se resaltan con un contorno verde. Al salir del campo el efecto se elimina. Este efecto guía visualmente al usuario al indicar en

```

1 ipInput.addEventListener('focus', function () {
2     ipInput.style.outline = '2px solid #123D15';
3 });
4 ipInput.addEventListener('blur', function () {
5     ipInput.style.outline = '';
6 });

```

qué campo está escribiendo. Conseguimos esto mediante el uso de los eventos focus (para cambiar el estilo al seleccionar) y blur (para cambiar el estilo cuando dejemos de seleccionar).

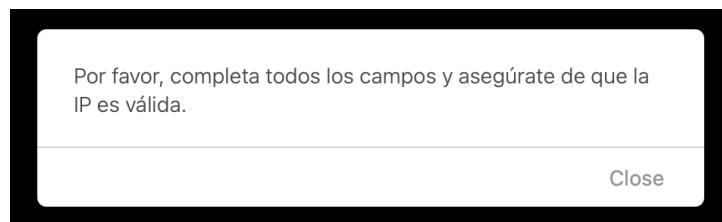


Además, al realizar click sobre el botón de reporte, el sistema comprueba que todos los campos estén completos y que la IP tenga un formato válido. Si algún campo está vacío se muestra una alerta informando al usuario. El objetivo es evitar envíos incorrectos. Para ello comprobamos que todos los campos estén completos y que la ip sea válida dentro de un EventListener al hacer click en el botón de reporte.



```
1 reportButton.addEventListener('click', function (e) {
2     var ip = ipInput.value.trim();
3     var category = categorySelect.value;
4     var description = descriptionArea.value.trim();
5
6     var ipRegex = /^(25[0-5]|2[0-4]\d|1\d\d|[1-9]\?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]\?\d)){3}$/;
7
8     if (!ipRegex.test(ip) || category === "" || description === "") {
9         e.preventDefault();
10        alert("Por favor, completa todos los campos y asegúrate de que la IP es válida.");
11    }
12 });
13});
```

En el caso de que no sea válida se envía una aviso con alert(). En la imagen podemos ver este mensaje.



Además cabe destacar el uso de los 5 métodos diferentes de acceso al DOM que podemos encontrar en esta página.

```
1 var form = document.forms[0];
2 var ipInput = document.getElementsByName('ip')[0];
3 var categorySelect = document.getElementsByName('category')[0];
4 var descriptionArea = document.getElementsByTagName('textarea')[0];
5 var reportButton = document.querySelector('.button.report');
```

Login page

El archivo login.js introduce efectos dinámicos en la página de acceso que mejoran la experiencia del usuario al interactuar con el formulario. Estos efectos incluyen la alternancia entre los modos de login y registro, resaltado de campos, y validación dinámica de los campos en el registro.

Cuando el usuario hace clic en cualquiera de los campos del formulario estos se resaltan automáticamente con un contorno verde. Al salir del campo, el contorno se elimina.

Al hacer click en el botón de reporte, se realiza una validación completa de los campos, mostrando alertas si se detectan errores (email inválido, contraseñas distintas, nombre de usuario corto, etc.).

Estos dos efectos son los mismos que los usados en la página anterior por lo que no los volveremos a explicar para no ser redundantes. Sin embargo, cabe destacar la importancia de incluirlos también en esta página para mantener una coherencia en los estilos ya que ambos son formularios.

Además, al pulsar el botón Signup, el formulario se expande automáticamente mostrando dos nuevos campos: el correo electrónico y la confirmación de la contraseña. Este efecto visual transforma el formulario de inicio de sesión en un formulario de registro sin necesidad de recargar la página ni redirigir al usuario. De igual forma, al pulsar el botón Login, el formulario vuelve a su estado original, ocultando los campos adicionales y mostrando solo los necesarios para iniciar sesión.

Este comportamiento se implementa mediante la asignación de un listener a cada botón, con acciones distintas dependiendo de si la clase expanded está presente o no. Dicha clase controla la animación que muestra el formulario de registro (signup).

Cuando la clase expanded está activa, significa que el formulario de registro está visible. En ese caso, el botón de login actúa como interruptor para volver al formulario de inicio de sesión, mientras que el botón de signup se encarga de validar los campos del formulario y, si todo es correcto, enviar el formulario con un submit.

Por el contrario, si la clase expanded no está presente (es decir, el formulario de inicio de sesión está activo), el botón de login realiza la validación de los campos y ejecuta el submit en caso de éxito, mientras que el botón de signup simplemente cambia a su respectivo formulario.

```
1  signupBtn.addEventListener("click", (e) => {
2      e.preventDefault();
3      if (card.classList.contains("expanded")) {
4          const email = form.querySelector('input[name="email"]');
5          const username = form.querySelector('input[name="username"]');
6          const password = form.querySelector('input[name="password"]');
7          const confirmPassword = form.querySelector('input[name="confirmPassword"]');
8
9          let valid = true;
10         let errorMessage = "";
11         const emailRegex = /^[^@\s]+@[^\s@]+\.\[^@\s]+\$/;
12
13         if (!emailRegex.test(email.value.trim())) {
14             valid = false;
15             errorMessage += "- Introduce un email válido.\n";
16         }
17         if (username.value.trim().length < 4) {
18             valid = false;
19             errorMessage += "- El nombre de usuario debe tener al menos 4 caracteres.\n";
20         }
21         if (password.value.trim().length < 4) {
22             valid = false;
23             errorMessage += "- La contraseña debe tener al menos 4 caracteres.\n";
24         }
25         if (password.value !== confirmPassword.value) {
26             valid = false;
27             errorMessage += "- Las contraseñas no coinciden.\n";
28         }
29
30         if (!valid) {
31             alert(`Errores en el formulario:\n\n${errorMessage}`);
32         } else {
33             console.log("Signup submit triggered");
34             form.requestSubmit();
35         }
36     } else {
37         card.classList.add("expanded");
38         console.log("signupBtn clicked: expanding form for signup");
39     }
40 });

});
```

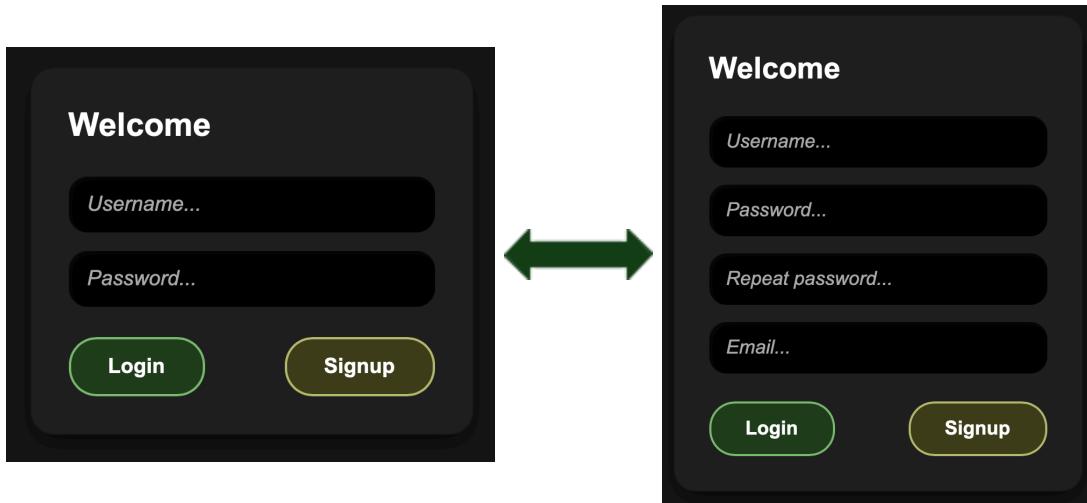


```
1 loginBtn.addEventListener("click", (e) => {
2   e.preventDefault();
3   if (card.classList.contains("expanded")) {
4     card.classList.remove("expanded");
5     console.log("loginBtn clicked: collapsing form (switching to login mode)");
6   } else {
7     const username = form.querySelector('input[name="username"]');
8     const password = form.querySelector('input[name="password"]');
9
10    let valid = true;
11    let errorMessage = "";
12
13    if (username.value.trim().length < 4) {
14      valid = false;
15      errorMessage += "- El nombre de usuario debe tener al menos 4 caracteres.\n";
16    }
17    if (password.value.trim().length < 4) {
18      valid = false;
19      errorMessage += "- La contraseña debe tener al menos 4 caracteres.\n";
20    }
21
22    if (!valid) {
23      alert(`Errores en el formulario:\n${errorMessage}`);
24    } else {
25      console.log("Login submit triggered");
26      form.requestSubmit();
27    }
28  }
29});
```

En las siguientes imágenes se puede observar el comportamiento dinámico del formulario de acceso. En la imagen de la izquierda se muestra el estado inicial de la página, donde únicamente se solicitan el nombre de usuario y la contraseña.

Al pulsar el botón Signup, se activa una animación que expande el formulario, añadiendo los campos de confirmación de contraseña y correo electrónico, como se muestra en la imagen de la derecha. Esta transición se realiza de forma fluida, sin recargar la página.

Si el usuario vuelve a pulsar el botón Login, el formulario regresa a su estado original, ocultando los campos adicionales y volviendo a la vista mostrada en la primera imagen.



Para la realización del cambio de formulario llevamos a cabo cambios en login.css y en login.html . Modificamos la clase form-section tanto en HTML como en CSS:

→ HTML:

- ◆ Añadimos los campos de registro como input.

→ CSS:

- ◆ En el estado inicial, el contenedor .form-section tiene una altura máxima de 120 píxeles y overflow: hidden, lo que oculta los campos adicionales del formulario. Al añadir la clase expanded, el max-height se incrementa a 300 píxeles, permitiendo que los nuevos campos se muestren.
- ◆ Con transition: max-height 1s ease-in-out evitamos que el cambio sea brusco y que la animación sea fluida

```

1  .form-section {
2    max-height: 120px;
3    overflow: hidden;
4    transition: max-height 1s ease-in-out;
5  }
6
7  .form-section.expanded {
8    max-height: 300px;
9  }

```

```

1  <div class="form-section">
2    <input type="text" name="username" placeholder="Username..." required>
3    <input type="password" name="password" placeholder="Password..." required>
4    <input type="password" placeholder="Repeat password..." />
5    <input type="email" placeholder="Email" />
6  </div>

```

Estructura de ficheros actualizada

```
trustip/
├── assets/
│   ├── data/
│   │   ├── countries.json
│   │   ├── countries.xml
│   │   ├── grid.js
│   │   └── processing.js
│   ├── icons/
│   │   ├── globe.png
│   │   ├── logo.png
│   │   └── user.png
│   ...
│   ├── imgs/
│   │   ├── countries/
│   │   │   ├── france.png
│   │   │   ├── germany.png
│   │   │   └── spain.png
│   │   ...
│   │   ├── clouds.jpg
│   │   ├── disc.jpg
│   │   ├── textures/
│   │   │   ├── earth_dark.jpg
│   │   │   ├── earth_day.jpg
│   │   │   ├── earth_night.jpg
│   │   │   ├── map_indexed.png
│   │   │   └── map_outline.png
│   │   └── libs/
│   │       ├── orbit-controls.js
│   │       ├── perlin-noise.js
│   │       ├── THREE.MeshLine.js
│   │       └── trackball-controls.js
│   └── scripts/
│       ├── globe/
│       │   ├── app.js
│       │   ├── config.js
│       │   ├── globe.js
│       │   ├── marker.js
│       │   ├── markers.js
│       │   ├── points.js
│       │   └── setup.js
```

```
|- |
|   |   |- JS shaders.js
|   |   |- JS utils.js
|   |   |- JS contributors.js
|   |   |- JS globe.js
|   |   |- JS leaderboard.js
|   |   |- JS login.js
|   |- Folder styles/
|       |- # contributors.css
|       |- # globe.css
|       |- # leaderboard.css
|       |- # login.css
|       |- # main.css
|       |- # welcome.css
|       |- <> contributors.html
|       |- <> globe.html
|       |- <> index.html
|       |- <> leaderboard.html
|       |- <> login.html
```

Modificaciones

A lo largo del proyecto se han ido realizando ciertas modificaciones sobre la idea original que nos ha parecido importante mencionar y justificar en esta sección para que quede reflejada la evolución del proyecto a lo largo del tiempo y la diferencia entre el diseño original y el resultado final.

- *"Cada país tendrá un color específico (del verde al rojo) que indicará la cantidad de IPs maliciosas que residen en el mismo (más rojo cuantas más haya) con respecto a la cantidad del resto"*
Hemos decidido eliminar esta funcionalidad porque no aporta nada extra al proyecto más que el hecho de ser repetitivo. Tanto en la Leaderboard Page como en el popup de cada país, se muestra su ranking.
- *"Cada IP estará linkeada de tal forma que al pulsar en ella se nos abrirá una web externa con más información sobre la misma"*
Esta funcionalidad también ha sido eliminada ya que un link a una página externa no aporta ningún valor real.
- *"En esta misma página también encontraremos un menú de búsqueda, si en él buscamos una IP, se desplegará desde el globo terráqueo el popup del país al que corresponde y un pequeño mensaje indicando si la IP es maliciosa o no"*
Esta funcionalidad se ha modificado a la búsqueda de países en vez de IPs. No hay motivo justificado de este cambio, la dificultad es la misma para ambas búsquedas por lo que es irrelevante si se buscan IPs o países.
- *"La IP figurada en el reporte será evaluada y añadida más tarde al globo terráqueo y contabilizada en el ranking si el reporte es acertado"*
Al no disponer de backend, esta funcionalidad ha sido imposible de desarrollar.
- Cambios con respecto al mockup de la Contributors Page
Durante el proceso de creación del HTML de esta página hemos llegado a la conclusión de que el apartado de donaciones era innecesario e irrelevante, ya que su funcionalidad iba a ser un simple link a una página externa, lo cual, una vez más, no aporta ningún tipo de valor.
- Cambios en el mapa de navegación
El mapa de navegación está realizado antes de los cambios mencionados anteriormente. Es decir, los elementos del mapa que coinciden con las modificaciones anteriores no se corresponden con el diseño actual.

