

**DAW 1**  
**CURSO 2021-2022**

**Bíceps**



**Antonio Miguel Martel**



**ULPGC**

# Índice

1. Introducción.....	3
1.1 Problemática.....	3
1.2 Motivación.....	3
2. Desarrollo.....	4
2.1 Análisis de requisitos y diseño del sistema.....	4
2.2 Instalación de las herramientas necesarias para instalar Tailwind.....	4
2.2.1 Instalación de la herramienta npm.....	4
2.2.2 Laboriosa y complicada instalación de Tailwind css.....	7
2.2.3 Configuración de Tailwind (De aquí pa bajo es importante).....	8
2.2.4 Configuración de postcss para tailwind.....	12
2.2.5. Uso de postcss en nuestro proyecto.....	14
3. Conclusión.....	16
Bibliografía.....	17

# 1. Introducción

## 1.1 Problemática

Es un hecho conocido el que las aplicaciones necesitan de los medios adecuados para crecer y desarrollarse con sencillez. La página web de g2Babies no es una excepción, ya que necesita de dichos medios para funcionar y evolucionar adecuadamente.

Es por ello que necesitamos de clases en nuestros ficheros .css, pues así podemos reducir el impacto que sufren las teclas Ctrl, C y V de nuestro teclado hasta un 85% con este nuevo cambio innovador.

## 1.2 Motivación

La idea de instalar Tailwind en nuestro proyecto surge en Octubre de 2021 de la mano de Fireship, un youtuber random de habla anglosajona dedicado a la producción de breves videotutoriales de índole educativa, destinados a la creciente masa de wannabes ñoclosos (en los que me incluyo) que aspiran a algo más en la vida que barrer patios.

Se buscaba, por unas causas u otras, la implementación de las clases de Tailwind, aprovechando ficheros .css y permitiendo que dichos ficheros se apoyasen en las utilidades que ofrece nuestro dios salvador Tailwind.

Texto de ejemplo uwu.

## 2. Desarrollo

### 2.1 Análisis de requisitos y diseño del sistema

Si jaja

### 2.2 Instalación de las herramientas necesarias para instalar Tailwind

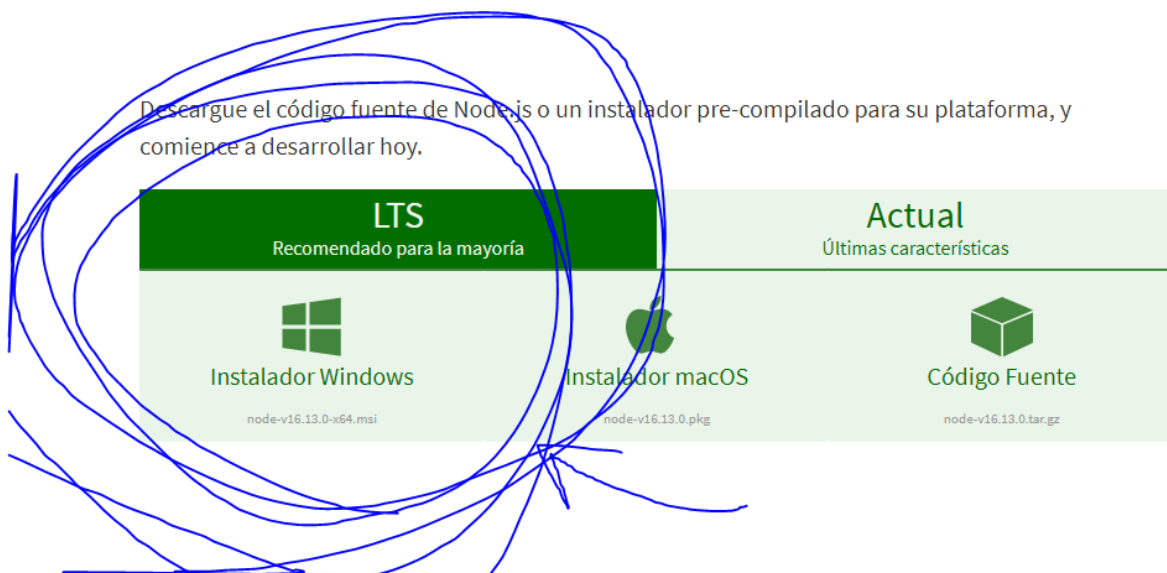
#### 2.2.1 Instalación de la herramienta npm

Como bien no sabemos, npm es un sistema de gestión paquetes que surgió a principios del año 2010. Es el sistema defacto para Node.js (un entorno de ejecución para JavaScript muy poco conocido).

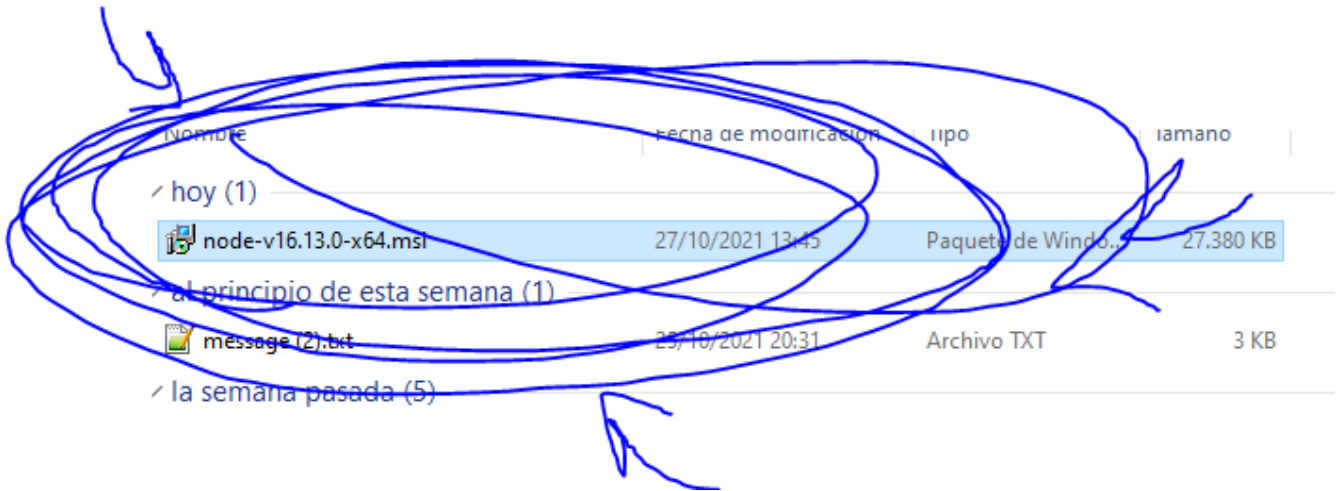
En linux y Mac viene instalado pero como aquí somos todos unos cromaños que todavía usan Windows tendré que explicar cómo se instala (Pasos complicados tomen apuntes porfi)

Te vas a la página principal de Node.js: <https://nodejs.org/es/download/>

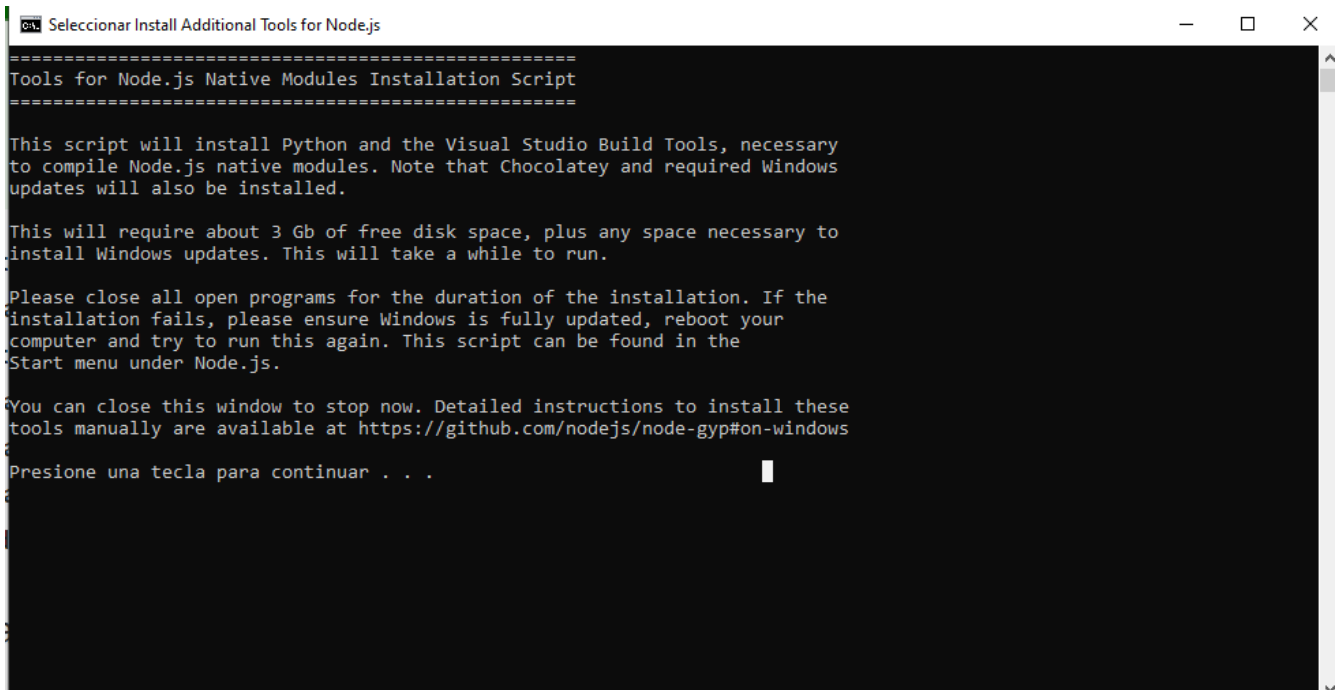
Pulsas el puto botón:



Abres el puto fichero



Le das que puto si a todo

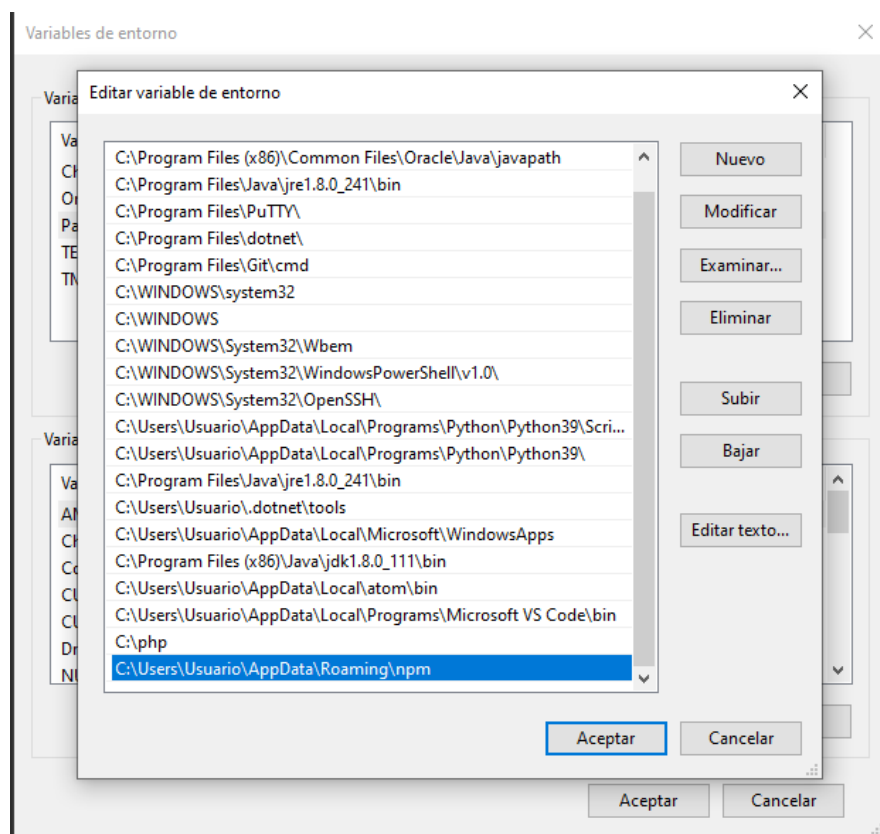


Pues ya tenemos Node.js (nos chupa el huevo) y npm (nos cae bien)

Espera medio mes a que se instale todo.

Yo me voy a cagar de hecho, brb

Si quieres creerte guay como yo puedes irte a las variables de entorno y verificar que se ha añadido una entrada a la variable \$PATH con la ruta al ejecutable de npm



## 2.2.2 Laboriosa y complicada instalación de Tailwind css

Cierras y abres visual studio (Es como reloadear la terminal que tiene)

Vas a la terminal y pones:

```
>> npm install -D tailwindcss@latest postcss@latest autoprefixer@latest
```

Haces lo que te diga la terminal

Enhorabuena! Has instalado Tailwind en tu proyecto!

## 2.2.3 Configuración de Tailwind (De aquí pa bajo es importante)

Ahora tenemos un fichero de configuración creado por npm que nos dice que coño tenemos en nuestro proyecto.

```
proyecto-daw1 > {} package.json > ...  
1  {  
2    "devDependencies": {  
3      "autoprefixer": "^10.3.7",  
4      "postcss": "^8.3.11",  
5      "tailwindcss": "^2.2.17"  
6    }  
7  }  
8
```

El otro fichero llamado package-lock.json que es dice las dependencias de lo que hemos instalado

Para usar Tailwind a lo crema necesitaremos su fichero de configuración.

```
PS C:\Users\Usuario\Documents\Proyectos\DAW\proyecto-daw1> npx tailwindcss init  
Created Tailwind CSS config file: tailwind.config.js  
PS C:\Users\Usuario\Documents\Proyectos\DAW\proyecto-daw1>
```

Tiene que estar en la "root dir". Es decir, en la carpeta principal del proyecto.



Crea esto en el root dir:

```
proyecto-daw1 > JS tailwind.config.js > <unknown>
1  module.exports = {
2    purge: [],
3    darkMode: false, // or 'media' or 'class'
4    theme: {
5      extend: {},
6    },
7    variants: {
8      extend: {},
9    },
10   plugins: [],
11 };
12
```

No me gusta como nos lo deja así que lo voy a tocar un poco

```
module.exports = {
  mode: "jit", // Build más rápido
  purge: ["./Template/*.html"], // Quita lo que no usamos
  darkMode: "class", // or 'media' or 'class'
  theme: {
    extend: {
      fontSize: {
        "8xl": ["6.5rem", { lineHeight: "1" }],
      },
    },
  },
  variants: {
    extend: {},
  },
  plugins: [],
};
```

El purge sirve para quitar las clases de tailwind que no usamos. Es necesario para que jit funcione bien.

El darkMode está puesto en modo "class". Significa que los hijos van a heredar los atributos del padre a la hora de pasarse a dark mode. (A lo mejor no lo implementamos así pero por

ahora es una opción). Recomiendo ver esto para que vean un poco lo que tengo en mente:  
<https://css-tricks.com/a-complete-guide-to-dark-mode-on-the-web/>

El tema de theme: extend: ... es para poder modificar las clases que nos da tailwind por defecto. En este caso cambié el de "8xl" para el tamaño de la fuente. Tmb puedes cambiar colores, paddings y demás:

Como se usa: <https://tailwindcss.com/docs/theme>

Opciones: <https://github.com/tailwindlabs/tailwindcss/blob/v1/stubs/defaultConfig.stub.js>

Para poder usar las clases que tenemos de tailwind primero hay que pasarlas a css.

Cuando lo importabamos con el href nos venían todas las clases compiladas de tailwind

```
<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
```

Pues ahora las vamos a compilar porque vamos a usar clases dedicadas.

Comenzaremos creando un par de clases css con tailwind y css normal (esto es muy bonito xd)

```
Template > styles > # article.css
...
1 | @import "tailwindcss/base";
2 | @import "tailwindcss/components";
3 | @import "tailwindcss/utilities";
4 |
5 | @layer components {
6 |
7 |   .contenedor-head {
8 |     @apply max-w-full m-0 flex items-center flex-wrap flex-col text-5xl;
9 |     padding: 0 0 10vh 0;
10 |    height: 70vh;
11 |   }
12 |
13 |   .contenedor-cards {
14 |     @apply flex items-center flex-col max-w-full m-0;
15 |     background: linear-gradient(to top, #dadada 97%, rgba(0, 0, 0, 0) 0);
16 |   }
17 |
18 |   #particles-js {
19 |     @apply absolute w-full h-screen bg-gray-100 bg-no-repeat bg-cover ml-auto mr-auto;
20 |     z-index: -1;
21 |   }
22 |
```

Esto no es css y si lo usamos a pelo no va a funcionar. Los 3 imports de arriba del todo incluyen todo lo que nos trae tailwind al instalarlo. El @layer acopla las clases que pongamos dentro de él en el paquete de componentes de tailwind (components). Gracias a ello podemos crear "clases de clases" y llamarlas tranquilamente desde el html como hacíamos con el tailwind que nos venía por defecto.

Usaremos la CLI de tailwind que nos permite compilar este código a clases css. Más adelante usaremos postcss, que añade más funcionalidades junto a esto.

```
>> npx tailwindcss -i ./Template/styles/article.css -o ./Template/styles/compiled/main.css
```

Y con un href a main.css tendremos dichas clases y todo lo que tenemos en tailwind.

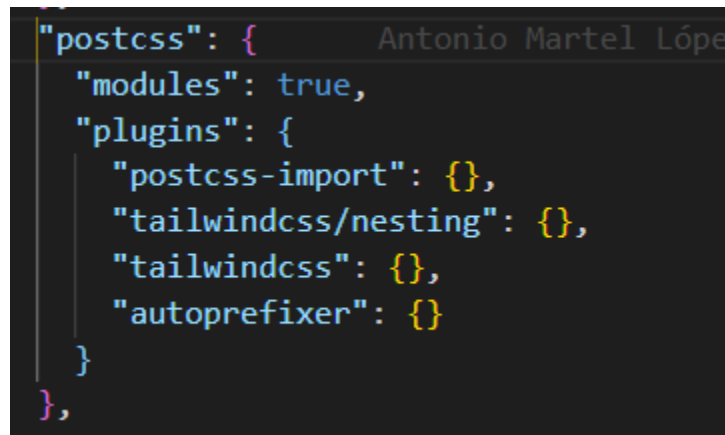
PERO tenemos un problema gordo. Y es que solo sirve con un fichero. Todos sabemos que un solo fichero css para toda una aplicación web es una prank gigante. Lo ideal sería tener un fichero .css para cada archivo .html o tener el css integrado dentro de cada .html (como se hace en la programación orientada a componentes como en Vue o Angular). En nuestro caso, interesa tener un archivo único que compilaremos que reúna todos los archivos que componen el estilo de nuestra página.

## 2.2.4 Configuración de postcss para tailwind

Pues para solucionar el problema anterior usaremos postcss. Nos venía en la instalación por defecto de tailwind. De hecho, tailwind nos pide que lo usemos con postcss porque reduce los tiempos de compilación (build times) y te permite añadir más virguerías (veremos las que más me llamaron). Instalaremos una utilidad de postcss que nos permite importar ficheros css en los ficheros que compilaremos.

```
>> npm install postcss-import
```

Y lo añadiremos en la lista de plugins que usa postcss. Esto está en el fichero packages.json



```
"postcss": {
  "modules": true,
  "plugins": {
    "postcss-import": {},
    "tailwindcss/nesting": {},
    "tailwindcss": {},
    "autoprefixer": {}
  }
},
```

Es importante saber que el orden importa y se aplica de arriba a abajo a la hora de compilar un fichero. En el fichero css que compilaremos se irá aplicando cada plugin en dicho orden. postcss-import lee los @import "dir", coje todo el código que haya en dicha dir y lo pega en el fichero donde está el import. tailwindcss/nesting es una utilidad que nos permite añadir los @imports dentro de la directiva @layer sin que hayan errores. Luego se aplica tailwind, que lee todas nuestras clases y las compila a código css y por último autoprefixer compila el css y le añade los prefijos que sean necesarios.

Los prefijos en el css se usan para que el css que hagamos sea compatible para todos los navegadores posibles. Usa los prefijos que hay en Can I Use, que es una página que se actualiza semanalmente para probar la compatibilidad de una página web con diferentes navegadores.

Página de Can I Use: <https://caniuse.com/>

Página de autoprefixer: <https://autoprefixer.github.io/>

Y ahora nos queda compilar el fichero a lo que tiene que ser. En este caso compilaremos el archivo main.css a uno nuevo llamado styles.css

Con postcss el comando es casi igual que en tailwind:

```
>>postcss ./Template/styles/compiled/main.css -o ./Template/styles/compiled/styles.css --watch
```

El --watch sirve para que cada vez que cambies cualquier clase se compile todo de nuevo automáticamente sin que tengas que volver a ejecutar el comando.

Nuestras clases tendrán un link al fichero de styles.css

```
<link rel="stylesheet" href="styles/compiled/styles.css"/>
```

## 2.3. Nuestro proyecto

### 2.3.1 Título genérico 1

Nosotros en el proyecto tenemos:

En la carpeta de styles/compiled:

El fichero main.css. Aquí estarán todos los ficheros .css que importaremos (ahora solo está article.css). De aquí se compila el código a true css usando postcss.

```
Template > styles > compiled > # main.css
    You, a minute ago | 1 author (You)
 1  @import "tailwindcss/base";
 2  @import "tailwindcss/components";
 3  @import "tailwindcss/utilities";
 4  @import "../article.css";
 5
 6
 7  /* Para poder usar los nuevos estilos
 8  |   usa el comando:
 9  |   npm run css-compile */      You, an hour
10
```

¿Porqué ahora usamos npm run css-compile en vez del rollo de "postcss ./Template/styles/compiled/main.css -o ./Template/styles/compiled/styles.css --watch"?

Porque npm te permite configurar scripts para que trabajes a tu bola. Añadiríamos lo siguiente en la carpeta de package.json:

```
"scripts": {  
  "css-compile": "postcss ./Template/styles/compiled/main.css -o ./Template/styles/compiled/styles.css --watch"  
},
```

Con Rails sera muy diferente (en ese caso será con webpack). Haré otro documento si hace falta explicando como funciona.

Para compilar el código css escribe:

```
>> npm run css-compile
```

Y todo funcionará solo. Lo puedes parar con Ctrl + C y no pasa nada.

Y el fichero styles.css. Esto es el estilo, esta vez compilado, que teníamos en main.css. Nuestras clases apuntarán a este fichero.

```
Template > styles > compiled > # styles.css  
You, seconds ago | 1 author (You)  
1  /*! tailwindcss v2.2.19 | MIT License | https://tailwindcss.com */  
2  
3  /*  
4  Document  
5  =====  
6  */  
7  
8  /**  
9  Use a better box model (opinionated).  
10 */  
11  
12 *,  
13 ::before,  
14 ::after {  
15   box-sizing: border-box;  
16 }  
17  
18 /**  
19 Use a more readable tab size (opinionated).  
20 */  
21  
22 html {  
23   -moz-tab-size: 4;  
24   -o-tab-size: 4;  
25   tab-size: 4;
```

Es css random que nadie entiende pero que hace lo que tiene que hacer.

En la carpeta de styles:

Los ficheros css que importaremos en styles/compiled/main.css

```
Template > styles > # article.css
You, seconds ago | 1 author (You)
1 @layer components { You, an hour ago • feat: Tailwind + postcss configured
2 |
3   .contenedor-head {
4     @apply max-w-full m-0 flex items-center flex-wrap flex-col text-5xl;
5     padding: 0 0 10vh 0;
6     height: 70vh;
7   }
8
9   .contenedor-cards {
10    @apply flex items-center flex-col max-w-full m-0;
11    background: linear-gradient(to top, #dadada 97%, rgba(0, 0, 0, 0) 0);
12  }
13
14  #particles-js {
15    @apply absolute w-full h-screen bg-gray-100 bg-no-repeat bg-cover ml-auto mr-auto;
16    z-index: -1;
17  }
18
```

### 3. Conclusión

Pretty crema klk

### Bibliografía

[1] Dirección General de Modernización Administrativa, Procedimientos e Impulso de la Administración Electrónica, Magerit versión 3.0: Metodología de análisis y gestión de riesgos de los Sistemas de Información. Libro I: Método. 2012.

[2] Dirección General de Modernización Administrativa, Procedimientos e Impulso de la Administración Electrónica, Magerit versión 3.0: Metodología de análisis y gestión de riesgos de los Sistemas de Información. Libro II: Catálogo de elementos. 2012.



- [3] Dirección General de Modernización Administrativa, Procedimientos e Impulso de la Administración Electrónica, Magerit versión 3.0: Metodología de análisis y gestión de riesgos de los Sistemas de Información. Libro III: Guía de Técnicas. 2012.
- [4] A. Ocón Carreras y C. Rosa Remedios, Apuntes de Emergencias Tecnológicas, 2ª. Ed. Las Palmas De Gran Canaria, 2019.
- [5] Cisco Systems, Inc.: "Cisco - Cisco Intrusion Detection", <http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/>, 2002.
- [6] Cisco Systems, Inc.: "Configuring SPAN and RSPAN", [http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/sw\\_6\\_3/config\\_gd/span.htm](http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/sw_6_3/config_gd/span.htm), 2002.
- [7] Enterasys Networks, Inc.: "Intrusion Detection Solutions", <http://www.enterasys.com/ids/>, 2002.
- [8] Garfinkel, S., Spafford, G.: "Practical UNIX & Internet security, 2nd Ed.". O'Reilly & Associates, Inc. 1997.
- [9] Garfinkel, S., Spafford, G.: "Web security and commerce". O'Reilly & Associates, Inc. 1997.
- [10] Guttman, B., Bagwill, R.: "Internet Security Policy: A technical guide", NIST Specail Publication, Julio de 1999.
- [11] IETF: "IDWG - Intrusion Detection Working Group", <http://www.ietf.org/html.charters/idwg-charter.html>