|  |  |
| --- | --- |
| Meld | |
|  | |
| 2024PROYECTO FINAL - DAW | PABLO BELLÓI.E.S. LEONARDO DA VINCI |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Meld** es un proyecto que fusiona el diseño gráfico con tecnologías web modernas como Next.js, Tailwind CSS y Three.js. Su objetivo es demostrar lo que es posible hacer con las herramientas de hoy en día y crear experiencias web innovadoras y visualmente impactantes. | |  |

ÍNDICE

[Estudio del sector 4](#_Toc167377368)

[Estructura funcional empresarial 4](#_Toc167377369)

[Objetivos 6](#_Toc167377370)

[Planificación 6](#_Toc167377371)

[Requisitos 7](#_Toc167377372)

[Viabilidad 8](#_Toc167377373)

[Primeros pasos 10](#_Toc167377374)

[Guía de instalación 10](#_Toc167377375)

[Funcionamiento del entorno 10](#_Toc167377376)

[Componentes de React 10](#_Toc167377377)

[Hooks de React 11](#_Toc167377378)

[Componentes de cliente y servidor 12](#_Toc167377379)

[Carga asíncrona 13](#_Toc167377380)

[Memoria de creación del proyecto 13](#_Toc167377381)

[Inicialización de Next.js 14](#_Toc167377382)

[Inicialización de NextUI 17](#_Toc167377383)

[Uso de librerías gráficas 19](#_Toc167377384)

[Implementación de modo oscuro 22](#_Toc167377385)

[Modelos 3d 24](#_Toc167377386)

[Partículas 26](#_Toc167377387)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Estudio del sector Cualquier empresa grande siempre va a seguir una línea en su diseño gráfico. Un diseño uniforme que vaya a definir sus ideales y su propia visión de su empresa.  Toda empresa necesita captar la atención del usuario para conseguir que este consuma lo que esta ofrezca. Esto es en general, ya que el sector que más nos interesaría es evidentemente el de las tecnologías de la información (IT). En este sector es especialmente solicitado el diseño gráfico. Cada día aparecen más startups de tecnología, cada día aparecen nuevos proyectos de código abierto y se desarrollan más facilidades para los diseñadores: es un sector extremadamente demandado actualmente. Específicamente, las empresas cuya operación se encuentra en aplicaciones web o escritorio, además de marketing, son las más interesadas Estructura funcional empresarial Muchas empresas grandes normalmente subcontratan a otras empresas o individuos para que gestionen su diseño. En este caso, supondremos que vamos a ser contratados por Landor, una empresa consultora de branding. Esta empresa ha trabajado con firmas muy grandes como Nike, Apple, Renault, etc. Además, forma parte de una empresa mucho más grande, WPP plc. |  |  |
|  | Esta empresa tiene varios equipos dependiendo del trabajo a realizar. Normalmente se suele dividir entre branding o puramente el diseño específico de un trabajo.  El branding es el diseño de la imagen de una marca, ya sea su logo, línea estética o imagen pública.  Un diseño específico puede ser, por ejemplo, el rediseño de la página web de una empresa. Normalmente suele haber un líder para el proyecto que dirige a un equipo (de los muchos que puede haber simultáneamente), este se encarga de la asignación y división de las tareas de cada miembro. El tamaño de un equipo puede ser de entre 10 hasta cientos de personas (por supuesto, este con una jerarquía más compleja). Landor tiene unos 5000 empleados, por lo que el área de operación de esta es global, realizando trabajos desde el logo de FedEx, en Estados Unidos, hasta el logo de BP en Inglaterra o el logo turístico de la ciudad de Madrid de hace unos años. La facturación de la empresa la controla WPP, por lo que no se sabe mucho, pero WPP es de las mayores empresas consultoras del mundo, por lo que este volumen debe ser alto. Los empleados de Landor normalmente disfrutan de beneficios como la capacidad de teletrabajar, horario flexible, y un buen salario dependiendo de la empresa que les solicite. | |  |

Este esquema muestra a un único jefe de diseño, pero puede haber uno para cada departamento. Este supervisa a los diversos líderes de equipo para cada encargo. La ubicación de un equipo no tiene que ser presencial, normalmente el equipo no está en la misma ubicación. La mayoría de la gente teletrabaja y acude a la empresa para las reuniones semanales. Puede haber gente en el mismo equipo que no esté en la misma ciudad o incluso país.

A graphic of a company structure

Description automatically generated

1Esquema de organización de los equipos de diseño

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Objetivos Como dije antes, el proyecto se encargará de enseñar visualmente como puede llegar a ser una aplicación web moderna, con todos los componentes de diseño gráfico que las empresas piden, con herramientas modernas.  Evidentemente, al ya existir estas herramientas hace un tiempo, no soy el primero en aplicarlas al diseño gráfico web, por lo que voy a aplicar las buenas prácticas del diseño técnico ya aplicadas e innovar en la medida de lo posible.  El objetivo principal es proporcionar una demostración práctica del potencial de las tecnologías modernas, como Next.js, Tailwind y Three.js, aplicadas al diseño gráfico en aplicaciones web. El proyecto aspira mostrar visualmente cómo estas herramientas pueden integrarse y aprovecharse en un ámbito profesional completamente real, a empresas que potencialmente podrían requerir de estos servicios.  Otros objetivos:   * Innovar en el Diseño Gráfico Web * Integrar tecnologías modernas * Aportar ideas a gente o empresas   Al cumplir con estos esto, busco brindar inspiración y orientación a empresas que puedan estar interesadas en adoptar tecnologías modernas para mejorar sus aplicaciones web. |  |  |
|  | Planificación  1. Investigación:    * 1. Investigar tendencias actuales en diseño web.      2. Estudiar tecnologías relevantes (Next.js, Tailwind, Three.js, etc.).      3. Identificar necesidades y demandas del sector empresarial objetivo. 2. Desarrollo de la Aplicación Web:    * 1. Implementar la estructura del proyecto y backend utilizando Next.js.      2. Integrar estilos y componentes de interfaz utilizando Tailwind y más.      3. Incorporar elementos gráficos y efectos visuales con Three.js.      4. Documentar el proceso de desarrollo, incluyendo capturas de pantalla y explicaciones correspondientes. 3. Pruebas y Depuración:    * 1. Realizar pruebas de usabilidad y rendimiento para que la aplicación tenga una correcta adaptabilidad.      2. Identificar y corregir errores de funcionalidad y diseño. 4. Entrega y Documentación:    * 1. Finalizar y optimizar la aplicación web para su lanzamiento.      2. Terminar la documentación sobre el proceso de desarrollo y las decisiones tomadas. 5. Defensa del proyecto    * 1. Crear una presentación visual y del proyecto para su exposición. | |  |
|  | Requisitos Para poder implementar todo esto, no necesitaremos nada del otro mundo: lo normal para desarrollar una aplicación web.  En caso de querer alojar nuestra página web, necesitaríamos un servidor para ello. En él necesitaríamos alojar la página en cualquier servicio que soporte Node.js, la tecnología backend con la que Next funciona. En caso contrario, podemos recurrir a hostings online si no quisiéramos alojarlo nosotros. Estos servicios normalmente se pagan mensualmente, y el costo varía dependiendo de las especificaciones que necesitemos.  También necesitaríamos un dominio para nuestra aplicación en caso de que queramos. Este se puede comprar fácilmente desde muchos proveedores como Cloudflare. No es estrictamente necesario, ya que muchos hostings te permiten alojar la página con uno de sus dominios, aunque no siendo este exactamente el que nosotros queramos.  También, naturalmente, necesitaríamos un equipo para desarrollar la propia aplicación en un IDE. Voy a utilizar VS Code para llevar a cabo el proyecto, aunque cualquier otro que soporte JavaScript es válido. Voy a trabajar solo. | |  |
|  | Viabilidad Para ser fiel a un trabajo real, vamos a crear una factura de presupuesto con nuestra supuesta empresa Landor.   |  |  |  | | --- | --- | --- | | Concepto | Descripción | Costo Estimado (EUR) | | Desarrollo de la Aplicación Web | Honorarios por el desarrollo del proyecto | 3000 | | Servidor | Costo mensual por el alojamiento de la página web en un servidor Node.js | 50 / mes | | Dominio | Costo anual por la compra del dominio para la aplicación web | 15 / año | | Herramientas de Desarrollo | En el caso de no ser gratis, se incluirían aquí. En nuestro caso, VSC es gratis. | - | | Tarifa de Desarrollador (por hora) | Tarifa por hora de trabajo del desarrollador | 25/hora | | Marketing Digital | Publicidad en línea para promocionar la aplicación web en redes sociales y buscadores | 500/mes | | Gastos Varios | Costos adicionales como servicios de diseño gráfico, licencias de software, etc. | 300 | | Total | **3300 + 550 / mes + 15 / año (dependiendo del tiempo de desarrollo y marketing)** | | | |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Documentación y notas de desarrollo  Meld  Ahora vamos a pasar a detallar el funcionamiento completo de la web. Además, se explicarán muchas decisiones y procesos tomados hasta llegar al producto final. |  |  |

##### Documentación

### Primeros pasos

#### **Guía de instalación**

En el caso de querer [descargar el proyecto](https://github.com/mzzdev/meld.git), deberás hacer uso de git con el comando:



Después, nos situaremos en el directorio de este con:



Una vez dentro, deberemos de instalar sus dependencias de node, con el comando:



Después de que se instalen, ya podremos iniciar el proyecto en un servidor local, con el comando:



Y accederemos a el en una ventana del navegador mediante la dirección

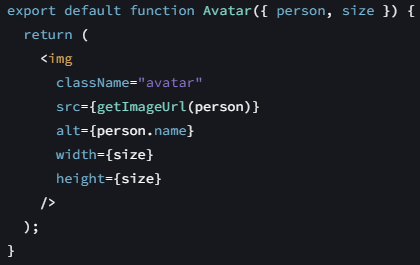
localhost:3000

### Funcionamiento del entorno

En esta sección explicaremos principalmente los fundamentos básicos de los que haremos uso en el proyecto, de Nextjs, el cual usa React para funcionar.

#### **Componentes de React**

Los componentes son el corazón de React. Se tratan de estructuras de interfaz modulares que si las juntamos tendríamos una página completa. Esto nos permite reutilizar código para ahorrar tiempo.



Un componente tiene varias partes:

* Props (propiedades): Son los atributos que recibe el componente, como si de una función normal se tratara. Estos valores pueden ser usados en cualquier parte del componente.
* Parte superior (encima del return): Aquí podremos designar lógica con JS normal o usar hooks (se verán en el siguiente apartado).
* Return: Aquí se devuelve código HTML normal, que será mostrado al usar el componente en otros lugares de la aplicación. Un return solo puede devolver un único bloque padre, en este caso una imagen. Esto no supone un problema ya que un bloque puede tener tantos hijos como sea necesario. El padre puede ser un <div> o <main> normal, o también puede ser <> con su cierre </>. Este markup es añadido por React en el caso de que no queramos usar ningún otro.

Los componentes siempre deben empezar con una letra en mayúsculas y deben ser exportados, en el caso de que queramos usarlos en otros archivos. Este componente también es exportado con default, esto indica que es el componente principal y el que se usará en los demás ficheros. Solo puede haber un default por archivo, por lo que crear otros componentes no default nos sirve para usarlos dentro del que sí lo es.

#### **Hooks de React**

Los hooks de React son funciones que siempre empiezan con la palabra ‘use’. Se usan para gestionar estados, por ejemplo, aumentar el contador de seguidores de una cuenta.



En este ejemplo, count es el valor cuyo estado queremos actualizar, y setCount es la función que hará la acción que nosotros queramos llevar a cabo. useState es un hook de React que se encargará exactamente de que esto funcione, ahora se le está pasado ‘0’, que será el valor por defecto. Los hooks siempre deberán ser declarados dentro de componentes, en la parte superior (fuera del return) de estos. Este componente usará el hook para crear un botó, en el cual al ser pulsado, llamará a handleClick, que actualizará el contador de clicks con la función designada en el hook. Después devolverá un botón que mostrará el valor que hemos actualizado.

A computer screen shot of code

Description automatically generated

Ahora podemos usar los componentes MyButton que queramos, y cada valor se actualizará por separado.

A black screen with white text

Description automatically generated

A close up of a text

Description automatically generated

Gracias a React, nos ahorramos muchas líneas de código en comparación a como lo haríamos en JS vanilla.

Muchas librerías externas incluyen sus propios hooks y además nosotros también podemos crear los nuestros propios. En este proyecto usaremos hooks a menudo.

#### **Componentes de cliente y servidor**

En Next.js, podemos especificar en que parte se renderiza cada fichero, en cliente o servidor mediante la etiqueta



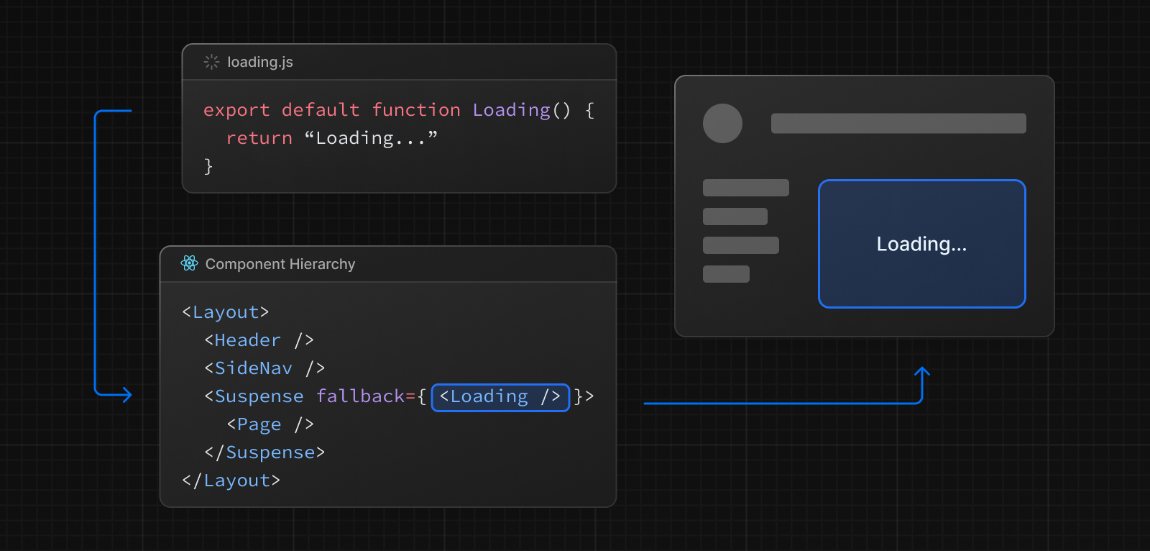
o “use server”, respectivamente, al principio de cada archivo. Si no escribimos ninguna, por defecto se renderizará en el servidor. Deberemos usar ambos, dependiendo de lo que hagamos en ese fichero. Por ejemplo, si usamos hooks, ese componente deberá renderizarse de lado del cliente, si no, no funcionará. Esto se debe a que en el servidor no existen los estados.También podremos usar estados o APIs, que en el servidor no se podría.

Por otro lado, renderizar de lado del servidor nos aporta otras cosas, como seguridad extra, ya que el cliente no tiene acceso al código, para realizar tareas de fetching en bases de datos, almacenar variables secretas para nuestro hosting. El servidor tendrá un mejor rendimiento, uniforme, ya que el cliente puede estar conectado con una mala conexión o hardware; también podremos realizar tareas de caching, streaming o SEO.

#### **Carga asíncrona**

Con ayuda de Next y React, podremos hacer que la carga asíncrona sea más sencilla de manejar. Supongamos que queramos traer datos de una base de datos a un componente específico de nuestra web. Si no lo hiciéramos de manera asíncrona, nuestra web no cargaría hasta que los datos no llegaran, aunque haya componentes que ya hayan terminado de cargar porque no necesitan hacer fetch de datos. Con ayuda de Next, podemos hacer lo siguiente: crear un archivo loading.tsx en junto con el page.tsx de la página que estemos. La interfaz que devuelva este archivo (por medio de un componente), será mostrada automáticamente cuando haya una petición en curso. Esto no nos sirve demasiado, ya que la página seguiría inutilizable mientras que la petición siga en curso, pero con ayuda de Suspense, de parte de react podemos gestionar de manera granular que componentes reciben datos de manera asíncrona.

Suspense es un componente que nos permite envolver código para marcarlo como asíncrono, pasándole un fallback, componente que se mostrará de manera granular en el espacio del código que siga cargando:



En el ejemplo tenemos un componente asíncrono rodeado de otros estáticos como el header o la barra de navegación. Al envolverlo, se la pasa el fallback del componente dentro del archivo loading.tsx, el cual se mostrará en lugar de paralizar toda la página, hasta que se reciban los datos.

### Memoria de creación del proyecto

En esta sección se detallarán los pasos que fueron seguidos para crear el proyecto desde cero.

#### **Inicialización de Next.js**

Para poder crear el proyecto necesitaremos tener instalado Node. Podemos hacerlo mediante su página web:

<https://nodejs.org/>

Para el proyecto estoy utilizando la versión 21.7.3, por lo que necesitarás como mínimo esa versión o superiores. Es posible elegir una versión en específico desde la página de descargas (<https://nodejs.org/en/download>) o utilizando herramientas como [nvm](https://github.com/nvm-sh/nvm) o [nvm-windows](https://github.com/coreybutler/nvm-windows) , que nos permiten tener más de una versión de Node instalada a la vez y cambiar entre ellas cuando queramos.

Ahora ya podemos crear un proyecto desde cero. He creado una carpeta dedicada para ello, aunque el instalador nos creará otra con el nombre que digamos:



Situados en una línea de comandos en esa carpeta, podemos ejecutar el instalador de Next mediante el comando:



Ante cualquier permiso de instalación decimos que sí.

A black background with white text

Description automatically generated

Una vez lanzado, el instalador nos hará preguntas de los ajustes que queramos aplicar a nuestro proyecto:

A computer screen with blue and white text

Description automatically generated

Vamos a explicar estas opciones:

* **Nombre de proyecto: Meld**
* **¿Quieres usar TypeScript?: sí**

En este proyecto voy a utilizar TypeScript, un superconjunto de JavaScript (cualquier código escrito en JS será perfectamente compatible) que nos permite usar tipado estático en nuestro código. Esto quiere decir que las variables, valores que aceptan y devuelven las funciones, etc. Deberán siempre tener un tipo, ya sea String, number, arrays, boolean, tipos personalizados que nosotros podemos crear

A screen shot of a computer

Description automatically generated

, o incluso el tipo :any, que permite cualquiera. Que exista un tipado estático nos permitirá ayudar a detectar errores más rápidamente y a mejorar la escalabilidad del código. TS será usado en lugar de JS si marcamos esta opción.

* **¿Quieres usar ESLint?: sí**

Un linter es una herramienta que analiza nuestro código en busca de problemas, errores de formato, o malas prácticas. Nuestro código se examina sin tener que ser ejecutado. Ejemplo:

A screenshot of a computer

Description automatically generated

La variable prueba no se usa, por lo que saldrá un error hasta que esta se use. ESLint se integrará automáticamente si marcamos esta opción.

* **¿Quieres usar Tailwind CSS?: sí**

Tailwind es un framework de CSS que nos permitirá construir más rápido nuestra aplicación. Los estilos de Tailwind se aplican directamente en las clases que se aplican a un elemento HTML, en lugar de tener que crear nosotros los estilos. Ejemplo:



Más adelante explicaré muchos de los conceptos de Tailwind, por ahora, marcaremos la opción para que Tailwind se instale con Next.

* **¿Quieres usar el directorio src/?: sí**

La estructura de carpetas de Next puede llenarse rápidamente de archivos de configuración y carpetas, por ello, si usamos el directorio src, nos será más fácil encontrar nuestras carpetas y archivos de código.



Por el momento solo tenemos la carpeta app, pero puede que creemos más en el futuro.

* **¿Quieres usar enrutador App?: sí**

En Next, hay dos tipos de enrutadores de páginas (cómo se gestionan las distintas páginas de nuestra web), Pages y App. Ejemplo:

****

YouTube tiene una página llamada feed, y dentro de esta tiene una llamada history. En nuestro proyecto vamos a usar App. Esto hará que podamos cambiar la ruta de nuestra app de la siguiente manera:

A screenshot of a computer

Description automatically generated

Next, por defecto buscará archivos llamados page, y estos contendrán todo el HTML de cada página. Si creamos una carpeta dashboard con su archivo page dentro, nuestra tendrá la ruta prueba.com/dashboard , y este page contendrá su HTML. Por defecto, Next nos crea un archivo page dentro de la carpeta App, el cual servirá de raíz de nuestra web.

A screenshot of a computer

Description automatically generated

* **¿Quieres personalizar el alias de importación por defecto?: no**

Esto hace que @/ se utilice como alias de importación. Esto hace que las rutas relativas se simplifiquen. Por defecto, si usamos src como es nuestro caso, @/ es un acceso directo para decir /src/

Usaremos esto:

A close up of a computer screen

Description automatically generated

En lugar de esto:



#### **Inicialización de NextUI**

Ahora procedemos a instalar la librería gráfica que vamos a utilizar, [NextUI](https://nextui.org/) (no confundir con Next.js). Esta librería nos permitirá acceder a componentes de React personalizables. Para ello cambiamos al directorio de nuestro proyecto e instalamos las dependencias.

****

Procedemos a seguir las instrucciones de la [instalación manual](https://nextui.org/docs/frameworks/nextjs#manual-installation) de la librería. Hacemos la manual y no la automática porque no queremos la plantilla de Next.js que nos crea esta.

1. Instalamos las dependencias



1. Añadimos las dependencias al archivo de configuración de Tailwind (Tailwind.config.js)

A computer screen with many colorful text

Description automatically generated

1. Creamos el archivo /app/providers.tsx

A screen shot of a computer code

Description automatically generated

1. Modificamos layout.tsx para que nuestra página sea envuelta en el proveedor de NextUI

A computer screen shot of text

Description automatically generated

También incluiremos las librerías clsx, tailwind-merge y mini-svg-data-uri para su posterior uso.



Con estas librerías podremos hacer uso de herramientas para crear muchos efectos visuales.

Una de estas librerías es Aceternity, para la cual también deberemos crear el archivo cn. Este archivo hace que mediante el uso de la función cn se haga uso de Tailwind-merge y clsx a la vez. CLSX sirve para aplicar estilos condicionales, por ejemplo, cambiar el color del fondo al pulsar un botón. twMerge hace que no haya conflictos entre clases de Tailwind.

**cn.ts**

A screen shot of a computer code

Description automatically generated

#### **Uso de librerías gráficas**

Las dos que principalmente vamos a utilizar son NextUI y Aceternity. La primera es para elementos de la UI técnicos, como botones, interruptores, separadores, etc. Y la segunda para elementos más gráficos, como efectos diversos y una gran cantidad de utilidades. Por supuesto todos estos componentes nos vendrán con unos estilos de serie, los cuales nosotros modificaremos. Incluso cambiaremos algunas de sus funcionalidades.

Para empezar, haremos uso de un componente de Aceternity llamado Background Gradient Animation, para crear el fondo de una parte de la landing page del proyecto. Ya hemos configurado antes sus dependencias, ahora deberemos crear el componente en nuestro proyecto. El código se puede encontrar en [su página](https://ui.aceternity.com/components/background-gradient-animation).

Los componentes de nuestra aplicación van dentro de la carpeta src, junto a app, en una carpeta que debemos crear para ello (no es necesaria, pero es recomendable para mantener el orden):

A screenshot of a computer

Description automatically generated

Además, dentro de la carpeta de componentes he creado ‘ui’, una carpeta solo para componentes gráficos (User Interface). Aquí crearemos un archivo tsx (React con TypeScript), preferiblemente con el nombre de este, y colocaremos su código.

A screen shot of a computer program

Description automatically generated A screen shot of a computer program

Description automatically generated

Cualquier componente se implementa así:



A black screen with colorful text

Description automatically generated

Se vería así:

A screenshot of a computer

Description automatically generated

En cada componente podemos cambiar lo que queramos para que se adapte a nuestras necesidades. Por ejemplo, podríamos modificar el CSS de este con Tailwind para que en lugar de ocupar un contenedor con la altura y anchura de la pantalla entera:

A screen shot of a computer screen

Description automatically generated

Que ocupe la mitad vertical y horizontal:

A computer screen shot of a code

Description automatically generated

A screenshot of a computer

Description automatically generated

O podemos cambiar los colores:

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

A colorful circle on a purple background

Description automatically generated

Las posibilidades son infinitas. Con estas herramientas trabajaremos durante toda la landing page.

#### **Implementación de modo oscuro**

Para la gestión de un modo oscuro, voy a usar la librería [next-themes](https://github.com/pacocoursey/next-themes):



Tendremos que modificar el archivo providers.tsx que previamente creamos para NextUI:

A computer screen shot of text

Description automatically generated

He añadido el proveedor ThemeProvider, que he renombrado para no confundirlo. Esto es para que los hooks que utilicemos funcionen correctamente. También he designado que el tema por defecto sea el del sistema.

Ahora, crearé el componente ThemeProvider, el cual hará uso del hook useTheme que nos da next-themes. Este componente devolverá un botón switch de NextUI el cual al ser pulsado comprobará cual es el tema actual, y se cambiará mediante el hook.

A screen shot of a computer program

Description automatically generated

En la página principal me encargo de la colocación del interruptor. También he tenido que configurar sus iconos dependiendo del estado:

A blue and black background with white circles

Description automatically generated

Además, he añadido una clase al CSS global. Sirve para crear una pequeña transición al cambiar entre los temas, apuntando a body, div y p y añadiendo una animación de ¼ s al cambiar sus colores y colores de fondo.

A black background with white text

Description automatically generated

Tema oscuro de Meld:

A screen shot of a computer screen

Description automatically generated

Tema claro de Meld:

A screenshot of a computer

Description automatically generated

Por supuesto que esta librería nos facilita el acto de cambiar de tema con un hook, pero nosotros aún deberemos programar los estilos para ambos temas mediante CSS o Tailwind. Ejemplo:



Este texto por defecto será de color blanco, y en modo oscuro lo será de color blanco.

#### **Modelos 3d**

Para conseguir esto usaremos Three.js, React-three-fiber y Drei:



Una vez instalados, crearemos un componente que nos permita cargar modelos 3D convencionales mediante THREE.

A screen shot of a computer code

Description automatically generated

Usaremos el hook useLoader de Fiber, que precargará los modelos en caché. AA este le pasaremos GLTFLoader, de parte de THREE, junto con la dirección de nuestro modelo 3D, que de primeras lo estamos poniendo arbitrariamente dentro del componente (lo cambiaré para pasarle la URL como prop). Mediante el hook useFrame podremos añadir efectos varios (de momento añado una rotación del modelo para probar). Una vez inicializados estos hooks, el componente devolverá el primitive, que es básicamente nuestro modelo cargado. Ahora bien, lo que queremos es que nuestro componente devuelva una vista, que contenga nuestro modelo cargado, y que nosotros podamos añadir efectos o lo que queramos. De esto se encarga Canvas, el marco donde se verá todo. Se puede observar el uso de MeshComponent, que contiene nuestro modelo cargado, junto a otros efectos que iré cambiando.

A blue atom with a white center

Description automatically generated

#### **Partículas**

También instalaremos la librería [tsParticles](https://github.com/tsparticles/tsparticles), para crear muchos efectos relacionados con partículas. Explicaré esto cuando lo usemos.



#### **Animaciones**

Utilizaremos la librería tailwind-animations, para añadir más animaciones CSS de manera nativa a Tailwind:



#### **i18n (internacionalización)**

#### **Diseño responsive**