

**INSTITUTO DE EDUCACIÓN SECUNDARIA CAMPANILLAS**  
TÉCNICO SUPERIOR DE *DESARROLLO WEB*

# **Proyecto Final de Ciclo**

## **Singularity**

Realizado por  
**Alan Gallardo Clemente**

**DEPARTAMENTO DE FAMILIA PROFESIONAL DE**  
**INFORMÁTICA Y COMUNICACIONES**

MÁLAGA, JUNIO DE 2021



## Contents

1.	Introducción .....	5
1.1	Presentación y objetivos.....	5
1.2	Planteamiento del problema .....	5
1.3	Estructura de la memoria.....	5
2.	Requisitos .....	6
2.1	Propósito .....	6
2.2	Requisitos profesorado y opciones escogidas .....	6
2.2.1	Módulo DWEC .....	6
2.2.2	Módulo DWES.....	7
2.2.3	Módulo DIW .....	8
2.2.4	Módulo DAW.....	8
2.3	Descripción general.....	9
2.3.1	Perspectiva .....	9
2.3.2	Funciones .....	9
2.3.4	Restricciones y posibles dependencias.....	11
3.	Análisis.....	11
3.1	Introducción .....	11
3.2	Diagrama de clases.....	11
3.3	Diagrama de casos de uso .....	13
3.3.1	Casos de uso (usuario no registrado).....	13
3.3.2	Casos de uso (usuario registrado).....	14
3.3.3	Casos de uso (usuario premium) .....	15
3.4	Diagrama entidad-relación .....	16
4.	Diseño .....	18
4.1	Introducción .....	18
4.2	Planificación de la interfaz.....	19
4.3	Técnicas de composición .....	20
4.3.1	Colores .....	20
4.3.2	Fuentes .....	21
4.3.3	Accesibilidad y usabilidad .....	22
5.	Implementación.....	25
5.1	Introducción .....	25
5.2	Lado Cliente.....	27
5.2.1	Introducción .....	27
5.2.2	Tecnologías y librerías .....	28
5.2.2.1	React.....	28

5.2.2.2	Redux .....	28
5.2.2.3	Axios .....	29
5.2.2.4	Material-UI.....	29
5.2.2.5	Moment .....	29
5.2.2.6	Google-Login .....	29
5.2.3	Estructura .....	30
5.3	Lado Servidor.....	31
5.3.1	Introducción .....	31
5.3.2	Tecnologías y librerías .....	31
5.3.2.1	NodeJS .....	31
5.3.2.2	Express .....	32
5.3.2.3	Mongoose.....	32
5.3.3	Estructura .....	33
6.	Despliegue.....	33
6.1	Hosting .....	33
6.1.1	Introducción .....	33
6.1.2	Despliegue de la parte cliente.....	33
6.1.3	Despliegue de la parte servidor .....	34
6.2	Documentación.....	35
6.3	Control de versiones.....	36
7.	Validaciones .....	37
7.1	Introducción .....	37
7.2	Validación de enlaces.....	37
7.3	Validación de estilos .....	37
7.4.	Validación de la resolución .....	38
8.	Conclusiones .....	38
8.1	Retrospectiva del trabajo realizado .....	38
8.2	Posibles ampliaciones .....	39
9.	Bibliografía .....	40
10.	Anexo .....	42

# 1. Introducción

## 1.1 Presentación y objetivos

La siguiente documentación técnica describe y analiza el trabajo realizado en el proyecto final de ciclo de desarrollo de aplicaciones web en IES Campanillas.

El proyecto consiste en el desarrollo de una web de artículos de ámbito científico en el que los usuarios podrán consultar una gran diversidad de artículos e información relacionada con el mundo de la ciencia.

Todos los artículos son accesibles para los usuarios, tanto los registrados como los no autenticados. Los usuarios registrados en la plataforma podrán puntuar los artículos o abrir un debate en un foro. Los usuarios registrados premium podrán gozar de todo lo anterior además de poder redactar sus propios artículos.

Además, la web cuenta con un foro donde los usuarios registrados podrán realizar preguntas que otros usuarios responderán.

## 1.2 Planteamiento del problema

El principal planteamiento del problema ha sido la construcción de un sitio web para alojar artículos de ciencia. A grandes rasgos la funcionalidad de la aplicación debía ser:

- Mostrar todos los artículos en la página principal.
- Permitir a todos los usuarios, registrados o no, consultar dichos artículos.
- Para un usuario premium poder crear, editar y eliminar su propio artículo.
- Permitir a un usuario registrado editar su perfil, así como añadir un método de pago.
- Permitir a los usuarios registrados realizar preguntas en un foro que serán contestadas por otros usuarios registrados.

## 1.3 Estructura de la memoria

La presente documentación está dividida en una serie de apartados que corresponden al proceso de desarrollo del proyecto. Dichos apartados, de forma resumida, son:

- **Requisitos.** Este apartado abarca el propósito y una descripción general del proyecto, repasando las distintas funcionalidades.
- **Análisis.** En este apartado se realiza un análisis del proyecto en forma de diagrama de clases y diagrama de casos de uso, ayudando a visualizar la construcción de la aplicación.
- **Diseño.** En este apartado se toman en cuenta los distintos requisitos del diseño de la interfaz, el diseño de un prototipo, abarcando las distintas técnicas de composición: paleta de colores, pesos visuales, fuentes, accesibilidad y usabilidad.
- **Implementación.** Partiendo del análisis y del diseño se empieza con la implementación del proyecto. En este apartado se describen las distintas tecnologías y librerías que han formado parte de la web.

- **Despliegue.** Todo lo abarca el despliegue de la web en un hosting externo, además de la documentación de la web y el control de versiones que se ha ido teniendo durante el desarrollo de la misma.
- **Validaciones.** Este apartado se centra en la comprobación del correcto funcionamiento de la web desarrollada mediante una serie de pruebas.
- **Retrospectiva.** En este apartado se echa la vista atrás en lo que se ha realizado, llegando a conclusiones de lo aprendido, dificultades que han tenido lugar y posibles ampliaciones de cara al futuro.

## 2. Requisitos

### 2.1 Propósito

El propósito de este apartado es definir los requerimientos impuestos de la web a desarrollar, así como las distintas funcionalidades y las restricciones que se pueden encontrar.

### 2.2 Requisitos profesorado y opciones escogidas

#### 2.2.1 Módulo DWEC

En el módulo Desarrollo Web Entorno Cliente, Sergio Banderas Moreno envió los siguientes requisitos:

- **Opción 1. Backend PHP + Frontend Javascript/JQuery Ajax**
  - *Javascript, ES6, JQuery*
  - *Validación de formularios*
  - *Borrar fila con fadeout*
  - *Paginación con Ajax*
  - *Buscador con autocompletado*
  - *Jquery UI: Calendario, Ventanas modales.*
- **Opción 2. Backend NodeJS Socket.io + Frontend Javascript/JQuery Ajax**
  - *Javascript/Typescript, ES6, LocalStorage*
  - *Validación de formularios*
  - *Ajax-Jquery comunicación con servidor*
- **Opción 3. Backend NodeJS + Frontend VueJS**
  - *VueJS uso de components, routers*
  - *Validación de formularios*
  - *Autenticación Firebase.*

Me decanté por la opción 3 adaptándola con ReactJS en vez de VueJS y MongoDB en vez de Firebase ya que quería que el backend fuese con NodeJS. La decisión de usar ReactJS fue por haberlo estado usando durante mi periodo de prácticas en la empresa y porque me parece más completo que VueJS.

Aunque la opción 3 tenga solo tres requisitos, he implementado otros de las anteriores opciones, tales como la paginación o el buscador.

Más adelante se explica cómo se ha implementado toda la parte del frontend.

### 2.2.2 Módulo DWES

En el módulo Desarrollo Web Entorno Servidor, Antonio José Sánchez Bujaldón envió los siguientes requisitos:

- **Opción 1. Proyecto backend PHP**

*La implementación del proyecto deberá basarse obligatoriamente en el patrón de diseño MVC (Modelo-Vista-Controlador) utilizando la versión 8 del framework Laravel. Se utilizará preferentemente Laravel UI con Bootstrap como Starter Kit.*

*La aplicación será de temática libre y deberá estar constituida por tres secciones totalmente diferenciadas: una vista genérica para cualquier visitante, otra para usuarios registrados y la de administración. A las dos últimas se accede mediante nombre de usuario y contraseña, debiendo realizarse en todo momento un correcto control de sesiones y protección de rutas a través de middlewares. Además, deberá incluir todo lo estudiado en clase (herencia de plantillas, componentes dinámicos, etc.), además de paginación con Laravel y aquellas características adicionales que se deseen añadir.*

*El código del proyecto, que deberá seguir los estándares de estilo para PHP, se subirá obligatoriamente a un repositorio de Github y se actualizará periódicamente cada semana.*

*La aplicación finalizada se desplegará obligatoriamente en un hosting externo y, además deberá estar convenientemente documentada, debiéndose justificar en todo caso las decisiones tomadas.*

*Se utilizarán los motores MariaDB o MySQL. Sea como sea, la base de datos deberá contar, como mínimo con 5 tablas, entre las que debe encontrarse la correspondiente a los usuarios. En el repositorio del proyecto deberá incluirse obligatoriamente el esquema E/R.*

*La base de datos se deberá crear utilizando las herramientas que proporciona el framework (migraciones, seeders y factorías) y no directamente sobre PHPMysqlAdmin o desde consola.*

*Se valorará positivamente la utilización de Breeze (con Tailwind CSS) o Jetstream (con Tailwind CSS o Bootstrap).*

*Se podrá completar la aplicación utilizando componentes dinámicos basados en Vue.js y Axios.js para la comunicación asíncrona en background.*

- **Opción 2. Proyecto backend Node**

*Utilizando NodeJS o PHP se deberá construir un sistema de logeo a la aplicación, realizando en todo momento un control de sesiones. En función de la aplicación realizada se le añadirá alguna funcionalidad desarrollada en el back, como una tabla de puntuaciones, perfil de usuario con posibilidad de edición, un sencillo chat para conversar con otros jugadores, etc. Esto es, se deberá desarrollar una mínima funcionalidad con un lenguaje de servidor.*

*Si se opta por PHP, el código seguirá en todo momento estándares de estilo para PHP y deberá documentarse convenientemente. En todo caso, no se podrá utilizar Laravel u otro framework similar.*

La elección del backend fue la opción 2, con Node, pero implementando la mayoría de requisitos de la primera opción, tales como, el patrón MVC, control de sesiones, middleware, 5 tablas en la base de datos, Axios, etc...

Más adelante se explica cómo se ha implementado toda la parte del backend.

### 2.2.3 Módulo DIW

En el módulo Diseño de Interfaces Web, Moisés Martínez Gutiérrez envió los siguientes requisitos:

*Diseñar un mapa de navegación y un prototipo de la interfaz con Adobe XD, explicando las técnicas de composición aplicadas: paleta de colores, posicionamientos, pesos visuales, tipos de fuentes, elementos de accesibilidad y usabilidad, rejilla utilizada...*

*Utilizar la hoja de estilos CSS3 para dar formato a la interfaz. En caso de utilizar algún Framework tipo Bootstrap, Materialize, UIKit... Crear una landing page solo con CSS3*

*El diseño de la interfaz deberá ser "Responsive" utilizando Media Queries, FlexBox y Grid layout.*

*Utilizar el preprocesador SASS para estructurar los archivos css en un único main.css (main.scss) con @import a los demás scss (colores, cabecera, pie, cuerpo...).*

*Insertar un elemento multimedia de cada tipo: vídeo, sonido, canvas y SVG.*

*Se podrán utilizar librerías tipo: Chartjs, D3.js, KoolChart, Snap.svg... para incluir el elemento multimedia (gráficas, animaciones, galería de imágenes).*

*Utilizar un software tipo Inkscape para crear el logo de la página al cual se le podrá aplicar animación con Snap.svg o anime.js.*

En el proyecto se cumplen todos los requisitos a excepción del uso de SASS, ya que React implementa el uso de CSS a través de JavaScript.

Más adelante se explica cómo se ha implementado toda la parte del diseño.

### 2.2.4 Módulo DAW

En el módulo Despliegue de Aplicaciones Web, Francisco Javier Jurado Jurado envió los siguientes requisitos:

*Despliegue local o remoto en un Servidor web (o Servidor de Aplicaciones).*

*Documentar todo el código fuente (propio) del Proyecto (JavaScript, TypeScript, Java, Php u otros) de forma adecuada (se indicarán unos mínimos).*

*Generar de forma automática la Documentación Técnica del mismo en PDF y HTML (Uso obligatorio de una Herramienta de Generación Automática tipo JavaDoc, PhpDocumentor, JsDoc, o similar).*

*Acreditar uso de un SCV. Se recomienda el uso de GIT/GITHUB o similar*

*Acreditar un seguimiento de versiones. Ramas.*

*Acreditar un despliegue automatizado, si es posible.*

*Acreditar uso de SSH y/o FTP en el desarrollo y/o despliegue, si es posible.*



*Entrega como Anexo o incorporado al Repositorio del Proyecto una Carpeta con la Documentación generada de forma automática a partir de las fuentes de la Aplicación Web Desarrollada.*

*Ficheros de Configuración acreditativos (del Servidor usado para el despliegue local y de la Aplicación Web).*

El despliegue se ha realizado en remoto, usando un servicio de hosting, se ha generado la documentación del proyecto y se ha utilizado GitHub como sistema de control de versiones.

Más adelante se explica cómo se ha implementado toda la parte del despliegue.

## 2.3 Descripción general

### 2.3.1 Perspectiva

El desarrollo del sitio web tiene como objetivo ofrecer contenidos para los usuarios interesados en la ciencia. Dichos contenidos serán distintos artículos con noticias recientes que usuarios premium crearán o un foro para responder preguntar o abrir debates solo para usuarios autenticados.

### 2.3.2 Funciones

#### Usuario no registrado

- Visualizar artículos: Un usuario no registrado podrá tener acceso completo a un artículo a excepción de puntuarlo.
- Visualizar autores: Un usuario no registrado podrá visualizar la lista de autores de la plataforma, así como los distintos artículos que hayan publicado.
- Filtrar por búsqueda: Un usuario no registrado podrá realizar búsquedas en un buscador para dar con un artículo en concreto.
- Inicio de sesión: Un usuario no registrado podrá iniciar sesión si posee ya una cuenta registrada.
- Registrarse: Un usuario no registrado podrá rellenar un formulario para poder registrarse en la web y poseer una cuenta.

#### Usuario registrado

- Visualizar artículos: Un usuario registrado podrá tener acceso completo a un artículo.
- Puntuar artículo: Un usuario registrado podrá puntuar un artículo dándole “Me gusta”.
- Visualizar autores: Un usuario registrado podrá visualizar la lista de autores de la plataforma, así como los distintos artículos que hayan publicado.

- Filtrar por búsqueda: Un usuario registrado podrá realizar búsquedas en un buscador para dar con un artículo en concreto.
- Perfil: Un usuario registrado podrá acceder a su perfil, donde podrá consultar y modificar cosas como, su información, artículos creados y su método de pago. Un usuario autenticado por Google no podrá hacer cambios en su perfil, a excepción de su método de pago.
- Foro: Un usuario registrado podrá acceder a un foro donde encontrará diversidad de preguntas hechas por otros usuarios, podrá responderlas o crear una pregunta.
- Cerrar sesión: Un usuario registrado podrá cerrar sesión y pasar a ser un usuario no registrado.

### **Usuario premium**

- Visualizar artículos: Un usuario premium podrá tener acceso completo a un artículo.
- Puntuar artículo: Un usuario premium podrá puntuar un artículo dándole “Me gusta”.
- Crear un artículo: Un usuario premium podrá crear un artículo, haciéndolo visible para los demás usuarios. También podrá modificarlo una vez creado o borrarlo.
- Visualizar autores: Un usuario premium podrá visualizar la lista de autores de la plataforma, así como los distintos artículos que hayan publicado.
- Filtrar por búsqueda: Un usuario premium podrá realizar búsquedas en un buscador para dar con un artículo en concreto.
- Perfil: Un usuario premium podrá acceder a su perfil, donde podrá consultar y modificar cosas como, su información, artículos creados y su método de pago. Un usuario autenticado por Google no podrá hacer cambios en su perfil, a excepción de su método de pago.
- Foro: Un usuario premium podrá acceder a un foro donde encontrará diversidad de preguntas hechas por otros usuarios, podrá responderlas o crear una pregunta.
- Cerrar sesión: Un usuario premium podrá cerrar sesión y pasar a ser un usuario no registrado.

### 2.3.4 Restricciones y posibles dependencias

Para poder ejecutar el proyecto solo se necesitará un ordenador o dispositivo móvil / tablet con un navegador y una conexión a Internet.

Se ha tenido en cuenta el navegador Chrome para la realización del diseño de la interfaz, por lo que en otros navegadores (Firefox, Opera...) el resultado podría verse ligeramente afectado.

## 3. Análisis

### 3.1 Introducción

Se ha realizado el análisis de la web siguiendo los estándares definidos por UML (Unified Modeling Language).

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema, incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

UML cuenta con varios tipos de diagramas, en este caso se hará uso del diagrama de clase, diagramas de casos de uso y diagrama entidad-relación.

### 3.2 Diagrama de clases

En este apartado se ha decidido modelar el funcionamiento del diagrama de clases para adaptarlo a React (más adelante se profundizará en que es React y cómo funciona), ya que, aunque no son clases como tal funcionan de manera parecida.

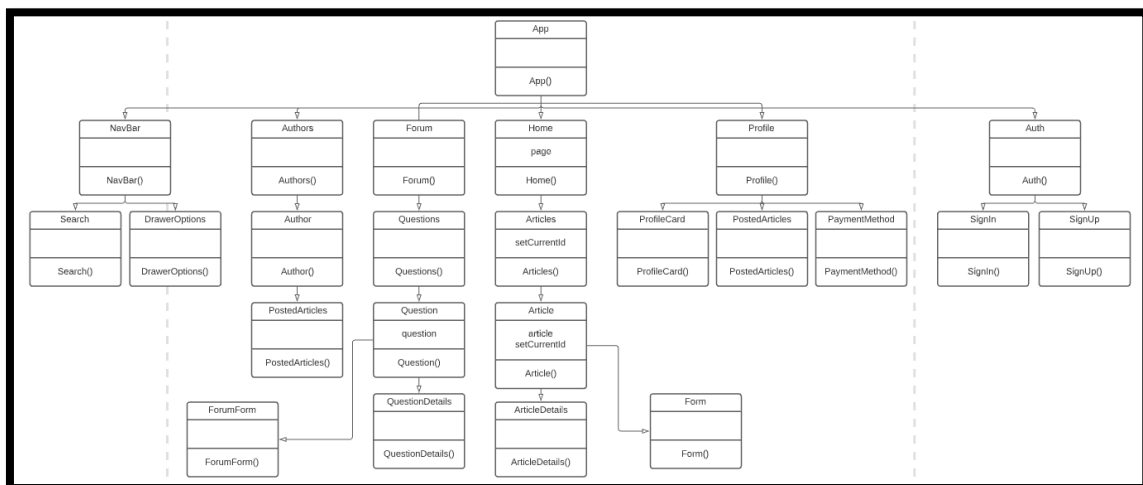


Figura 1. Diagrama de clases

- **App.** Es el componente raíz, de él cuelgan todos los demás componentes.
- **Home.** La página principal de la web.
- **Articles.** La lista de artículos.
- **Article.** Un solo artículo.
- **ArticleDetails.** El componente que muestra los detalles de un artículo.
- **Forum.** El foro.

- **Questions.** Las preguntas del foro.
- **Question.** Una sola pregunta.
- **QuestionDetails.** El componente que muestra los detalles de una pregunta.
- **Profile.** El perfil.
- **ProfileCard.** El componente que muestra los datos del usuario.
- **PostedArticles.** El componente que muestra los artículos publicados.
- **PaymentMethod.** El componente que muestra el método de pago del usuario.
- **Authors.** El componente que muestra los autores.
- **Author.** Un solo autor.
- **PostedArticles.** Los artículos publicados por un autor.
- **Auth.** El componente de la autenticación.
- **SignIn.** Autenticarse.
- **SignUp.** Darse de alta.
- **NavBar.** La barra de navegación.
- **Search.** La barra de búsqueda.
- **DrawerOptions.** El menú deslizante.
- **Form.** El formulario para crear un artículo.
- **ForumForm.** El formulario para crear una pregunta.

### 3.3 Diagrama de casos de uso

#### 3.3.1 Casos de uso (usuario no registrado)

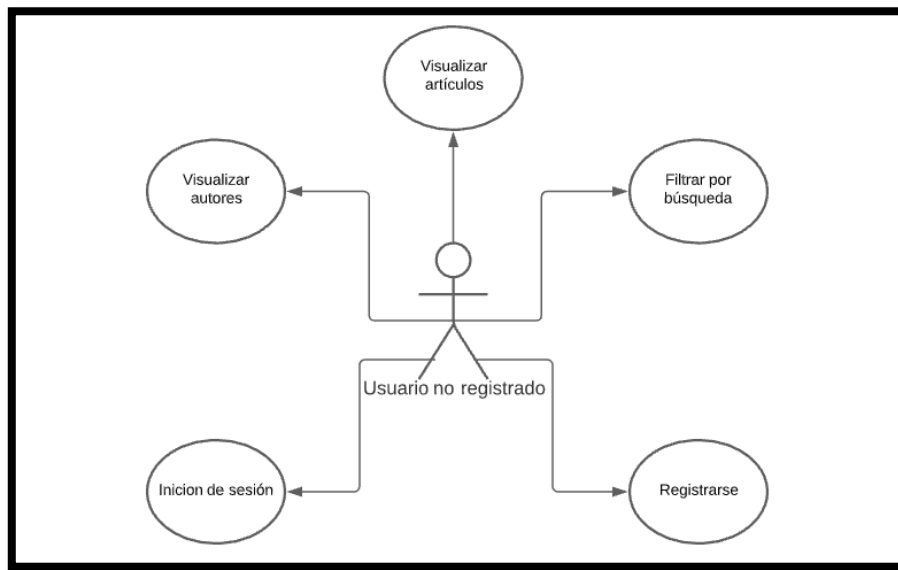


Figura 2. Caso de uso (Usuario no registrado)

#### Usuario no registrado

- Visualizar artículos: Un usuario no registrado podrá tener acceso completo a un artículo a excepción de puntuarlo.
- Visualizar autores: Un usuario no registrado podrá visualizar la lista de autores de la plataforma, así como los distintos artículos que hayan publicado.
- Filtrar por búsqueda: Un usuario no registrado podrá realizar búsquedas en un buscador para dar con un artículo en concreto.
- Inicio de sesión: Un usuario no registrado podrá iniciar sesión si posee ya una cuenta registrada.
- Registrarse: Un usuario no registrado podrá rellenar un formulario para poder registrarse en la web y poseer una cuenta.

### 3.3.2 Casos de uso (usuario registrado)

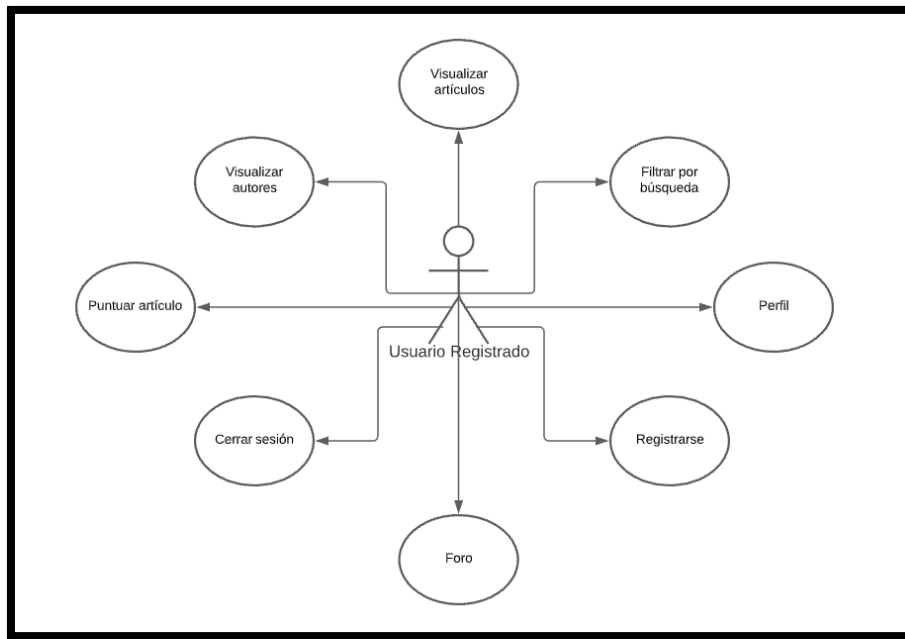


Figura 3. Caso de uso (Usuario registrado)

#### Usuario registrado

- Visualizar artículos: Un usuario registrado podrá tener acceso completo a un artículo.
- Puntuar artículo: Un usuario registrado podrá puntuar un artículo dándole “Me gusta”.
- Visualizar autores: Un usuario registrado podrá visualizar la lista de autores de la plataforma, así como los distintos artículos que hayan publicado.
- Filtrar por búsqueda: Un usuario registrado podrá realizar búsquedas en un buscador para dar con un artículo en concreto.
- Perfil: Un usuario registrado podrá acceder a su perfil, donde podrá consultar y modificar cosas como, su información, artículos creados y su método de pago. Un usuario autenticado por Google no podrá hacer cambios en su perfil, a excepción de su método de pago.
- Foro: Un usuario registrado podrá acceder a un foro donde encontrará diversidad de preguntas hechas por otros usuarios, podrá responderlas o crear una pregunta.
- Cerrar sesión: Un usuario registrado podrá cerrar sesión y pasar a ser un usuario no registrado.

### 3.3.3 Casos de uso (usuario premium)

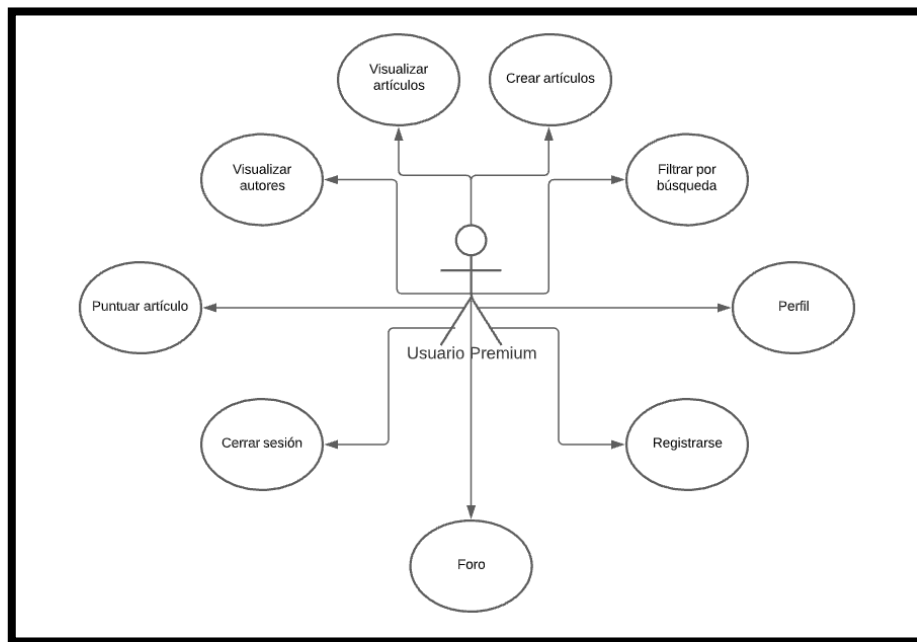


Figura 4. Caso de uso (Usuario premium)

#### Usuario premium

- Visualizar artículos: Un usuario premium podrá tener acceso completo a un artículo.
- Puntuar artículo: Un usuario premium podrá puntuar un artículo dándole “Me gusta”.
- Crear un artículo: Un usuario premium podrá crear un artículo, haciéndolo visible para los demás usuarios. También podrá modificarlo una vez creado o borrarlo.
- Visualizar autores: Un usuario premium podrá visualizar la lista de autores de la plataforma, así como los distintos artículos que hayan publicado.
- Filtrar por búsqueda: Un usuario premium podrá realizar búsquedas en un buscador para dar con un artículo en concreto.
- Perfil: Un usuario premium podrá acceder a su perfil, donde podrá consultar y modificar cosas como, su información, artículos creados y su método de pago. Un usuario autenticado por Google no podrá hacer cambios en su perfil, a excepción de su método de pago.
- Foro: Un usuario premium podrá acceder a un foro donde encontrará diversidad de preguntas hechas por otros usuarios, podrá responderlas o crear una pregunta.

- Cerrar sesión: Un usuario premium podrá cerrar sesión y pasar a ser un usuario no registrado.

### 3.4 Diagrama entidad-relación

Para el desarrollo de este proyecto se ha usado MongoDB (más adelante se definirá que es), resumidamente, MongoDB es un sistema de base de datos NoSQL, orientado a documentos.

NoSQL como la propia palabra indica, trata de un sistema que difiere del modelo clásico de los sistemas de gestión de bases de datos relacionales, siendo el más destacado que no usa SQL como lenguaje principal de consultas. Los datos almacenados no requieren estructuras fijas como tablas.

A continuación, se ha simulado como sería el esquema entidad-relación y el paso a tabla si tratásemos con una base de datos relacional.

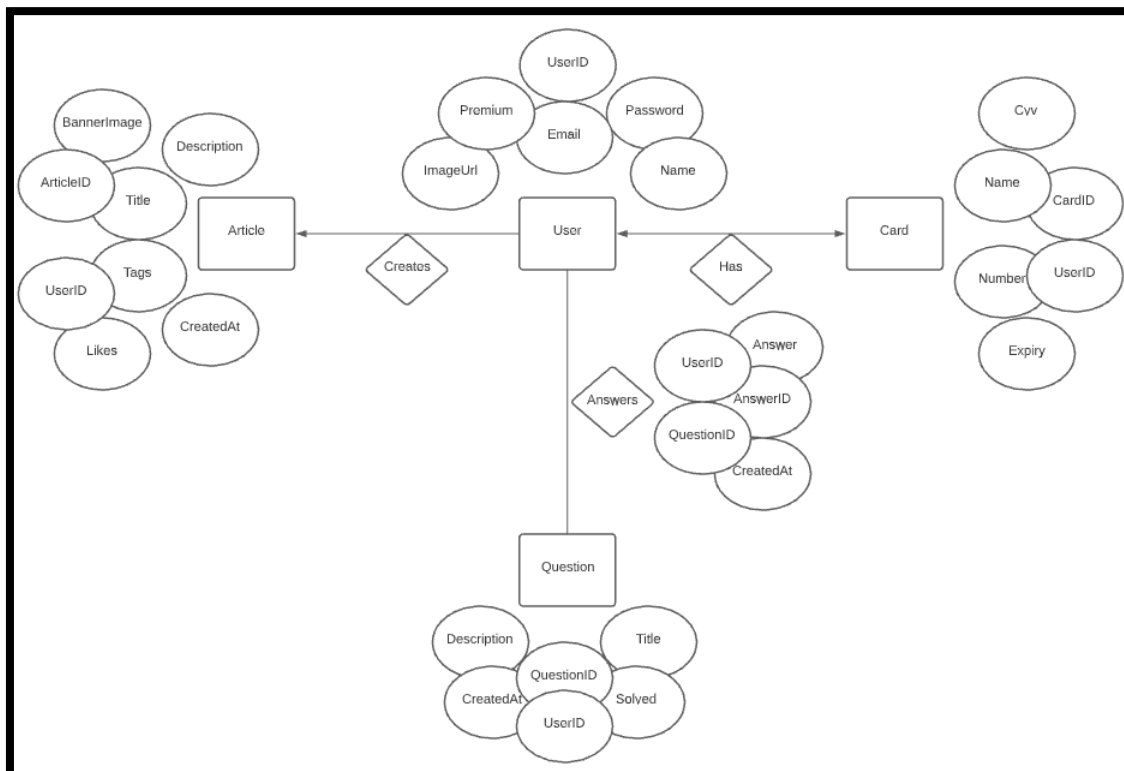


Figura 5. Diagrama entidad-relación

#### User

UserID	La clave primaria del usuario
Name	El nombre completo del usuario
Email	El email del usuario
Password	La contraseña del usuario
Premium	Si el usuario es premium o no
ImageUrl	La imagen de perfil del usuario



**Article**

ArticleID	La clave primaria del articulo
UserID	La clave foránea del usuario
Title	El título del articulo
Description	La descripción del articulo
BannerImage	La imagen del encabezado del articulo
Tags	Las etiquetas del articulo
Likes	Los me gusta del articulo
CreatedAt	La fecha de creación del articulo

**Card**

CardID	La clave primaria de la tarjeta
UserID	La clave foránea del usuario
Name	El nombre de la tarjeta
Number	El número de la tarjeta
Expiry	La fecha de expiración de la tarjeta
Cvv	El cvv de la tarjeta

**Question**

QuestionID	La clave primaria de la pregunta
UserID	La clave foránea del usuario
Title	El título de la pregunta
Description	La descripción de la pregunta
Solved	Si la pregunta está resuelta o no
CreatedAt	La fecha de creación de la pregunta

**Answer**

AnswerID	La clave primaria de la respuesta
UserID	La clave foránea del usuario
QuestionID	La clave foránea de la pregunta
Answer	La respuesta
CreatedAt	La fecha de creación de la respuesta

## 4. Diseño

### 4.1 Introducción

Para que un diseño web sea efectivo, debe lograr que los usuarios del sitio puedan acceder con facilidad a los contenidos, interactuar con eficacia con todos los componentes y sentirse cómodo en forma permanente, y todo ello sin siquiera pensarlo.

El elemento que consigue que esto sea posible es la interfaz, en cuya preparación y diseño se debe poner especial atención.

Una parte importante de los éxitos o fracasos de los sitios depende en gran medida de este elemento fundamental del diseño. Además, el diseño de interface debe perseguir hacer webs usables.

La finalidad de un buen diseño de interfaz es proporcionar un marco de uso que permita realizar las tareas de la mejor forma posible. Por eso, los objetivos de un diseñador de interfaces deben ser dos: la simplicidad y la coherencia.

La simplicidad con que se desarrolle esta interfaz es crucial para determinar que un usuario se sienta satisfecho y desee regresar a un sitio.

El hecho de que una persona deba realizar una extensa navegación por el sitio para hallar lo que busca en él es totalmente contraproducente. Por el contrario, si alguien que visita un sitio cuenta con varias herramientas que le permiten acceder rápidamente a aquello que le interesa, seguramente volverá.

La simplicidad está dada por varios factores a tener en cuenta. El primer concepto importante es que los elementos gráficos o textuales que componen la interfaz deben ser claros y de fácil identificación.

## 4.2 Planificación de la interfaz

Por si queda alguna duda, la interfaz de usuario, también conocida como UI (User Interface) es el medio por el cual los usuarios interactúan con una máquina o cualquier otro dispositivo tecnológico. Es decir, incluye los puntos de contacto entre una persona y el equipo.

Para realizar la planificación de la interfaz se ha construido un prototipo de web usando AdobeXD.

Adobe XD se centra en el diseño de interfaces digitales para aplicaciones móviles, ya sean para PC o para Mac, o sitios web. Y a través de este programa es mucho más sencillo crear imágenes preliminares de la interfaz de un proyecto.

Además, te permite ahorrar tiempo porque puedes realizar cambios a partir de esas imágenes preliminares para que todo quede optimizado antes de llegar a la etapa de programación.

Adobe XD te brinda la posibilidad de crear prototipos interactivos para que puedas presentar el “boceto” de lo que será la web, volviendo mucho más tangible tu trabajo y aterrizándolo en un plano mucho más real.

Cuenta con una gran variedad de herramientas, además de plugins que se pueden descargar en la store de adobe.

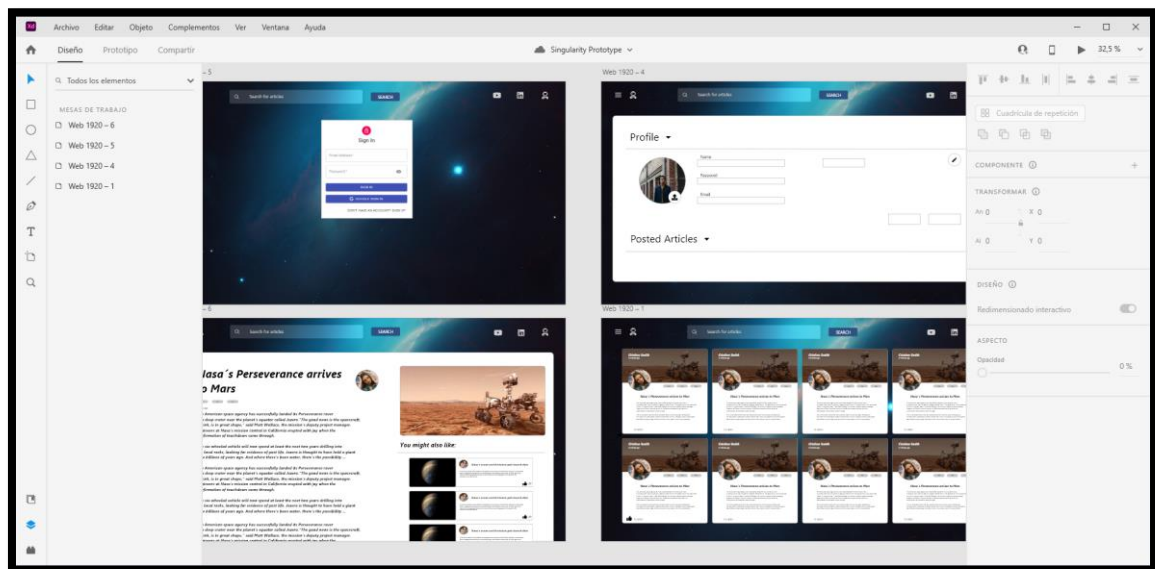


Figura 6. Prototipo AdobeXD

Estas son algunas de las pantallas que se han ido creando y que se han tenido en cuenta a la hora de realizar el diseño de la interfaz gráfica de la web.

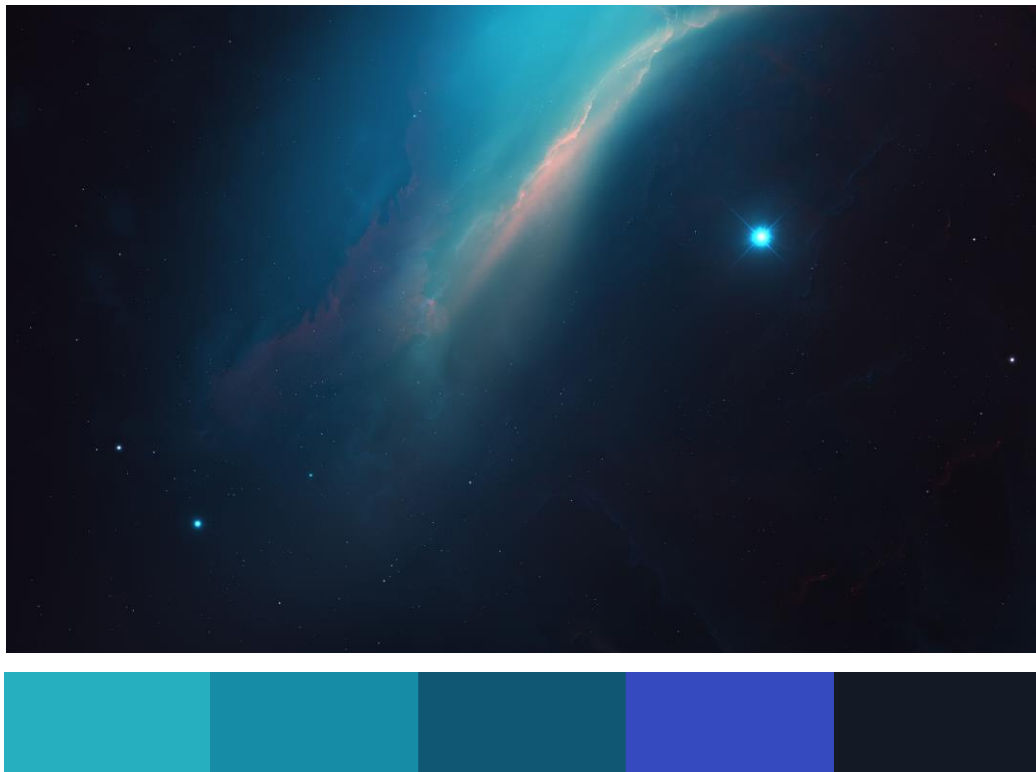
## 4.3 Técnicas de composición

### 4.3.1 Colores

El color es el medio esencial para tocar las emociones de los visitantes de un sitio web pues crea una reacción física y emocional en los espectadores. Los colores son capaces de establecer el tono adecuado para llevar el mensaje a los visitantes: pueden calmar, excitar, estimular a las acciones...

El color es extremadamente poderoso. Cuando se establece un esquema de color para el diseño web es importante hacerlo correctamente, guiándose por los principios de la teoría del color.

El fondo escogido de la web es el siguiente:



*Figura 7. Fondo y paleta de colores*

Donde el color predominante es el azul. El azul significa paz, tranquilidad, integridad, estabilidad. Se asocia con profesionalidad, profundidad, confianza y honor.

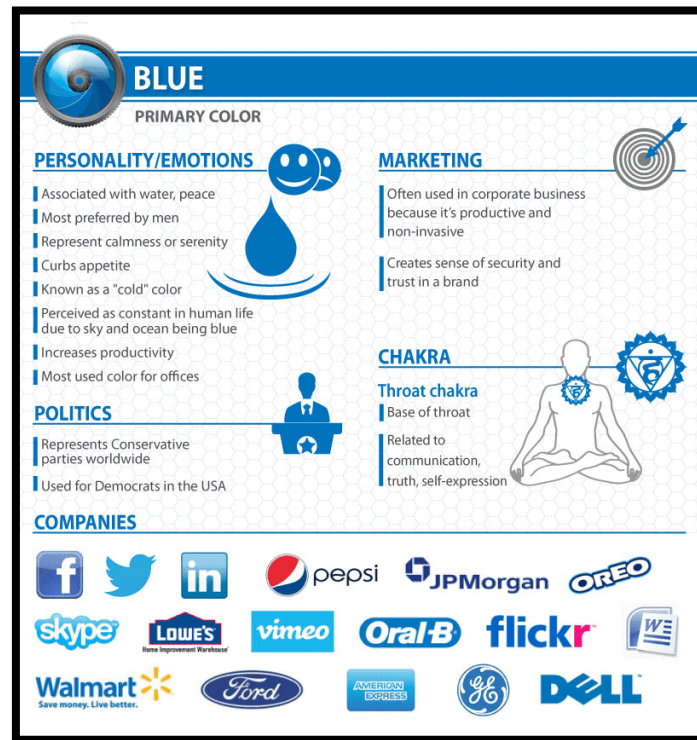


Figura 8. Significado color azul

Se ha usado una paleta de colores análoga, una paleta análoga posee colores adyacentes al color base en el espectro de tono. Utiliza 3 colores que se encuentran juntos en el círculo cromático. Suelen funcionar bien ya que tienen colores en común y no domina ninguno sobre otro. Son la combinación perfecta para resaltar y contrastar un elemento específico sin demasiada interrupción.

Se usa uno de los colores para que sea el principal y los otros dos como secundarios (uno para sustentar y un tercer color para acentuar).

#### 4.3.2 Fuentes

La tipografía en un sitio web busca coherencia de estilo entre la fuente, el tamaño, el color, el diseño, la alineación, y otros múltiples factores.

La fuente utilizada es Roboto Sans-Serif, una fuente geométrica y elegante ideal para la web, cuya temática es ciencia.

Roboto, Sans Serif

Figura 9. Fuente Roboto, Sans Serif

### 4.3.3 Accesibilidad y usabilidad

La **accesibilidad** hace referencia al **acceso universal** a nuestro sitio web independientemente del hardware, el software, la localización geográfica o las infraestructuras de red de las que dispone el visitante en cuestión.

Las circunstancias de cada usuario son distintas: el país en el que reside y el idioma que habla, sus capacidades visuales, motrices, auditivas y cognitivas... Una web accesible tiene en cuenta estas circunstancias para poder brindar a la mayoría de usuarios un fácil acceso a las tecnologías.

Por tanto, la accesibilidad es algo esencial en todo tipo de páginas y es un ejercicio de solidaridad con todo tipo de perfiles de usuario que pueden llegar a consultarla, con la intención de no dejar a nadie fuera por ser incapaz de usar un sitio web. Frecuentemente se asocia a las páginas gubernamentales, ya que las instituciones deben poder dar soporte a todos los ciudadanos, pero, en realidad, es un ámbito en el que todos debemos participar.

El tipo de contenido debe ser aquel que se pueda leer correctamente, facilitando cosas como las descripciones textuales del contenido de las imágenes. También se debe procurar que el código del HTML se pueda entender semánticamente, usando las etiquetas correctas que aporten significado a su contenido.

Ahora que se ha definido que es la accesibilidad vamos a hacer una prueba de daltonismo para un usuario con **deuteranopía** (disfunción visual correspondiente a la percepción del color verde), otro con **protanopía** (disfunción visual correspondiente a la percepción del color rojo), y otro con **tritanopía** (disfunción visual correspondiente a la percepción del color azul),

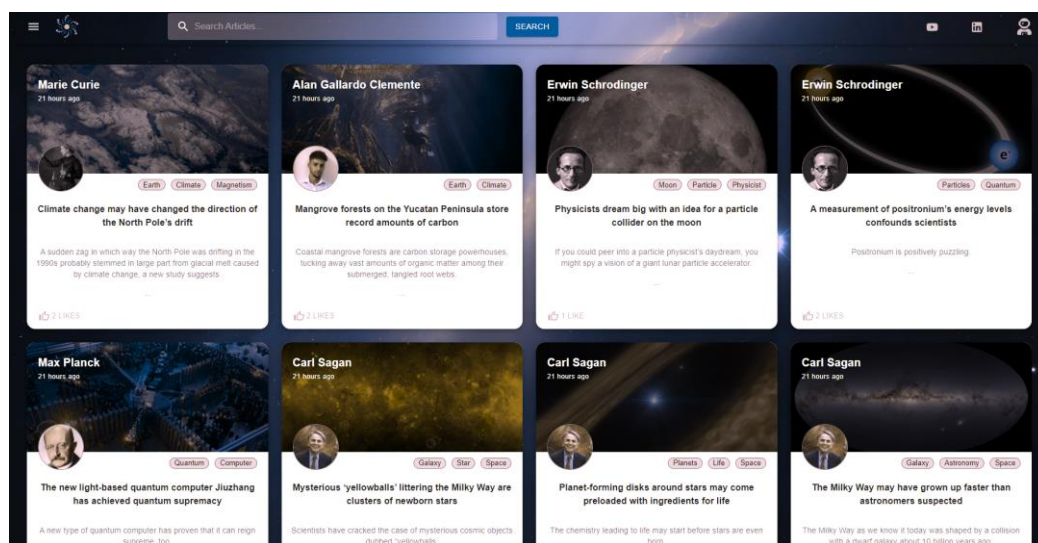


Figura 10. Deuteranopía



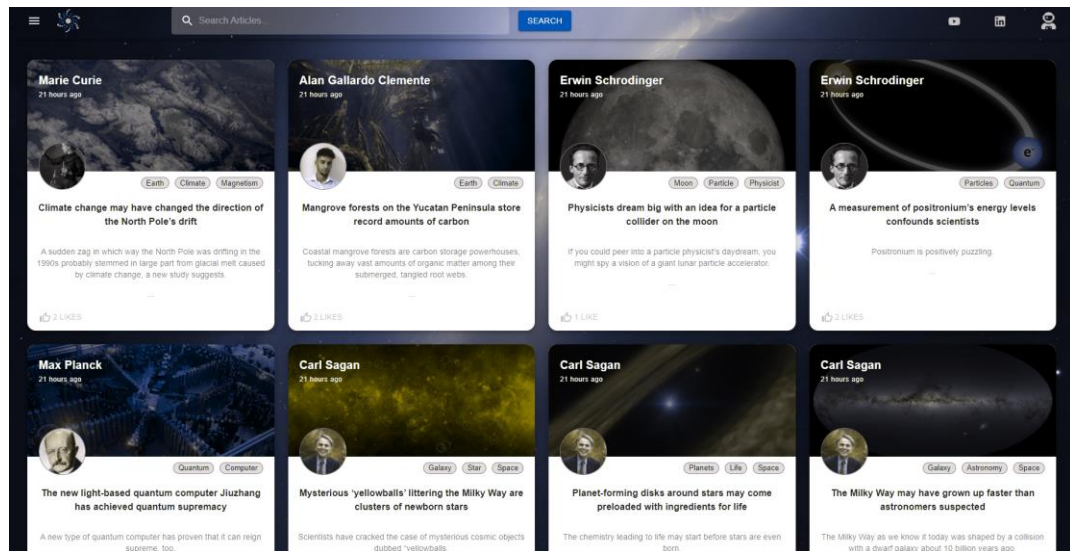


Figura 11. Protanopía

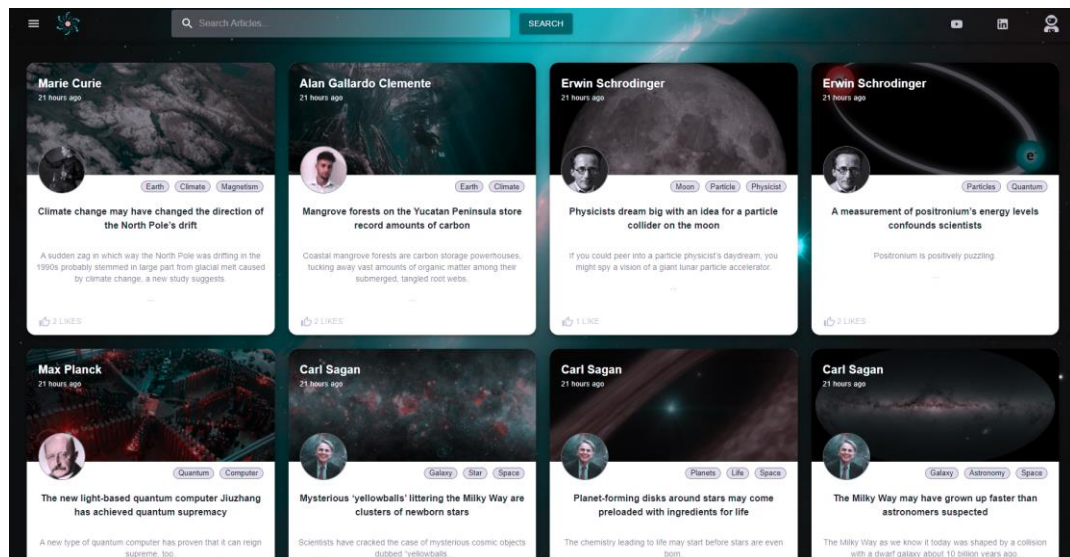


Figura 12. Tritanopía

A pesar de que los colores de los encabezados de los artículos se vean distintos, se pueden apreciar perfectamente los textos, los cuales son el principal objetivo de la web.

Un usuario con dificultades auditivas no debería tener problema al manejarse por la web, ya que carece de sonidos.

La web está totalmente en inglés, lo cual puede ser un problema para un usuario que desconozca el idioma, en un futuro, sería productivo añadir varios idiomas a la plataforma.

La **usabilidad** permite que los usuarios puedan alcanzar sus objetivos con un mínimo esfuerzo y unos resultados máximos. Es decir, que puedan navegar por el entorno gráfico de nuestra web de manera sencilla y eficaz. Los usuarios que accedan a nuestra página deben de disponer de autonomía a la hora de navegar por la web: deben de tener el control del sitio.

La **rapidez** es imprescindible. Los sitios cuya velocidad de carga es demasiado lenta hacen que los visitantes desistan de su intento por visitar la página. Según estadísticas, cuando un sitio web tarda más de 20 segundos en cargarse el 80% de sus visitantes acaba abandonando la página antes de verla.

Un sitio de apariencia **limpia** y **sencilla** permitirá a los usuarios navegar con facilidad y encontrar siempre aquello que busquen en nuestra web. Cuanto más cómodo sea el navegar por la página más efectiva será la búsqueda de información de nuestro visitante, lo que se traduce en una mejor experiencia de usuario.

La interfaz de usuario no debe de suponer un problema, tiene que requerir un mínimo proceso de aprendizaje y resultar amigable, es decir, **intuitiva**.

La información es esencial. Los títulos y descripciones son una gran ayuda para encontrar lo que se está buscando en menor tiempo. También debemos tener en cuenta que leer en pantalla no es tan cómodo como puede resultar en papel. Por eso, el contenido tiene que resumirse para que el usuario no tenga una sobrecarga de información. Debemos proporcionar una lectura **amena** y **concisa**.

La percepción de los **colores** no es la misma para todos (ni para todos los dispositivos). Hay que usar los colores con precaución para que todos usuarios puedan navegar fácilmente por nuestro sitio.

Lo mismo ocurre con la **legibilidad**. Para que todo aquel que visite nuestra web pueda disfrutar de su contenido tenemos que cuidar que los textos contrasten con el fondo y que el tamaño de la fuente sea correcto para permitir su lectura con comodidad.

Ya visto lo que es la usabilidad, vamos a realizar un test, para ello he dejado que una persona recorra la web y responda a unas preguntas del 1 al 5 siendo 1 en completo desacuerdo y 5 completamente de acuerdo.

Después de responder obtendrán los resultados del test restando 1 a las preguntas impares y las preguntas pares serán de 5 menos el valor asignado por la persona. Una vez obtenido el valor final se multiplicará por 2,5.

1. Creo que me gustará visitar con frecuencia esta web. **4**
2. Encontré la web innecesariamente compleja. **1**
3. Es fácil de usar la web. **5**
4. Creo que necesitaría del apoyo de un experto para recorrer la web. **1**
5. Encontré las diversas posibilidades de la web bastante bien integradas. **4**
6. Pensé que había demasiada inconsistencia en la web. **2**
7. Imagino que la mayoría de las personas aprenderían muy rápidamente a utilizar la web. **4**
8. Encontré la web muy grande al recorrerla. **1**
9. Me sentí muy confiado en el manejo de la web. **4**
10. Necesito aprender muchas cosas antes de manejarla en la web. **2**

$$((4 - 1) + (5 - 1) + (5 - 1) + (5 - 1) + (4 - 1) + (5 - 2) + (4 - 1) + (5 - 1) + (4 - 1) + (5 - 2)) * 2.5 = 85$$

El resultado del test de usabilidad es de **85 de 100**, una puntuación notable.



## 5. Implementación

### 5.1 Introducción

Para construir la web se ha usado la pila **MERN**. La pila MERN es un conjunto de marcos/tecnologías utilizados para el desarrollo web de aplicaciones.

Consta principalmente de **MongoDB** (base de datos), **ReactJS** (frontend), **ExpressJS** y **NodeJS** (backend).

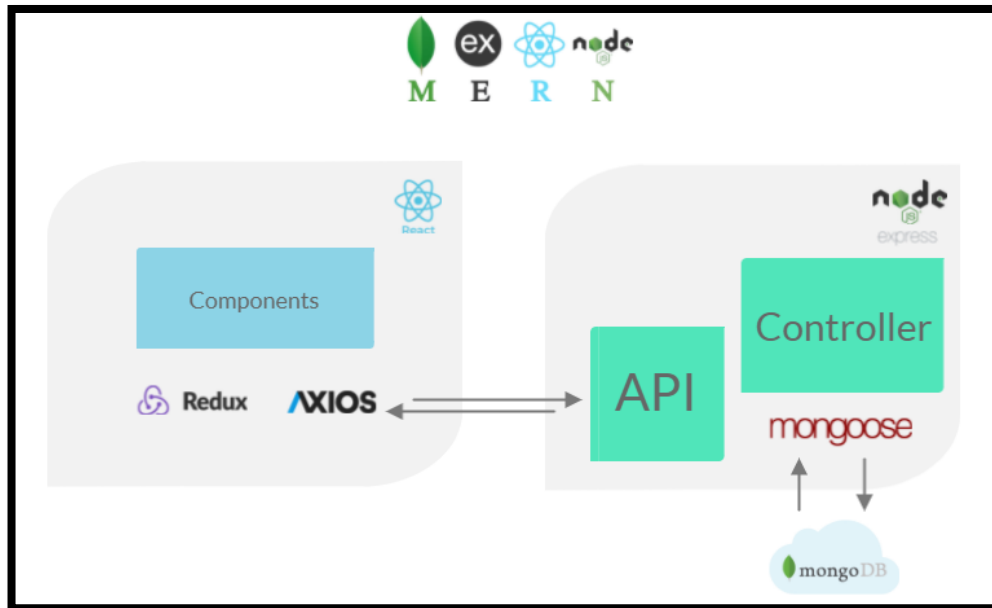


Figura 13. Estructura MERN

#### - MongoDB

Es una de las mejores bases de datos NoSQL que se conoce y está orientada a documentos. Una base de datos MongoDB se puede utilizar para almacenar los datos de la aplicación, cada registro es un documento que consta de pares clave-valor que son similares a los objetos JSON (JavaScript Object Notation).

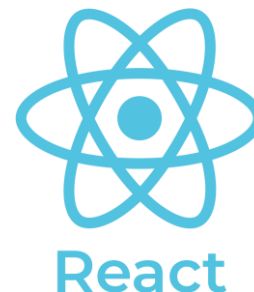
MongoDB es flexible y permite a sus usuarios crear esquemas, bases de datos, tablas, etc, sin los requerimientos de una pesada base de datos SQL.

En este proyecto se alojan los documentos en la nube que nos ofrece MongoDB.



## - **ReactJS**

Es una biblioteca creada por Facebook, utilizada muy ampliamente en la actualidad, está basado en un paradigma llamado programación orientada a componentes en el que cada componente es una pieza con la que el usuario puede interactuar. Estas piezas se crean usando una sintaxis llamada JSX permitiendo escribir HTML y CSS dentro de objetos JavaScript.



Estos componentes son reutilizables y se combinan para crear componentes mayores hasta configurar una web completa. Esta es la forma de tener HTML con toda la funcionalidad de JavaScript y el estilo gráfico de CSS centralizado y listo para ser abstraído y usado en cualquier otro proyecto.

## - **ExpressJS**

Es un marco que se ha superpuesto en la parte superior de NodeJS y se puede utilizar para crear el backend del sitio web con la ayuda de las estructuras y funciones de NodeJS.



Sin embargo, como NodeJS está destinado a ejecutar JavaScript del lado servidor, pero no para desarrollar sitios web, ExpressJS está destinado justo a esto, a crear sitios web.

## - **NodeJS**

Es un entorno de ejecución para JavaScript que puede permitirle ejecutar JavaScript del lado servidor y no en un navegador. Un interesante concepto a tener en cuenta en Node.js es el concepto de módulo, recursos que pueden ser más o menos simples o complejos en funcionalidad y que contiene un código JavaScript que podemos reutilizar en toda nuestra aplicación.



Estos módulos tienen su propio contexto y no interfieren entre sí. Esto es una notable ventaja pues podemos crear nuestro proyecto a medida sin complicaciones, sorpresas ni comportamientos inesperados.

### Beneficios de usar la pila **MERN**:

- Abarca todo el ciclo de desarrollo, desde frontend (lado del cliente) hasta el backend (lado del servidor).
- Facilita el proceso de trabajar con una arquitectura modelo vista controlador (MVC) haciendo que el desarrollo fluya sin problemas.
- Ayuda a evitar el trabajo pesado innecesario, por lo que mantiene el desarrollo de la aplicación web muy organizado.
- Se basa en 4 tecnologías probadas y ampliamente respaldadas: Mongo DB, ReactJS, Express, NodeJS.
- Añaden un conjunto de herramientas de prueba prediseñadas.
- Frameworks basados en código abierto y con el respaldoado por los apoyos de su comunidad.

## 5.2 Lado Cliente

### 5.2.1 Introducción

La parte cliente o frontend, es la parte de un sitio web que interactúa con los usuarios, por eso se denomina parte cliente. Son todas las tecnologías de diseño y desarrollo web que se ejecutan en el navegador y que se encargan de la interactividad con el usuario.

El frontend se desarrolla, principalmente, a través de tres lenguajes: HTML (HyperText Markup Language), CSS (Cascading Style Sheets) y JS (Javascript). Cada uno de estos lenguajes se usa para desarrollar diferentes partes del front.

En este caso, al usar ReactJS, HTML y CSS se desarrollan a través de Javascript, facilitando el proceso.

Aparte de ReactJS, en el frontend se usan otras tecnologías como Redux o Axios, que serán definidas más adelante.

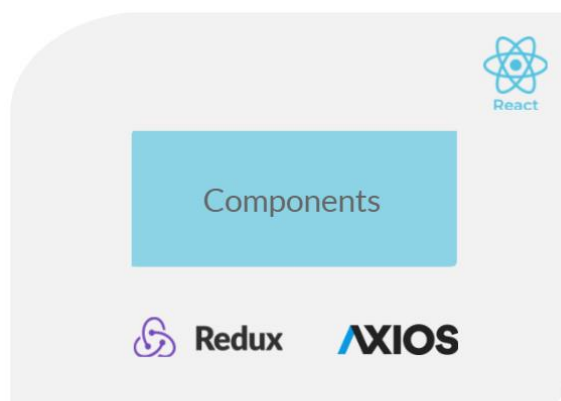


Figura 14. Estructura Cliente

## 5.2.2 Tecnologías y librerías

### 5.2.2.1 React

Como ya se ha definido en apartados anteriores React está basado en un paradigma llamado programación orientada a componentes en el que cada componente es una pieza con la que el usuario puede interactuar. Estas piezas se crean usando una sintaxis llamada JSX permitiendo escribir HTML y CSS dentro de objetos JavaScript.

Estos componentes son reutilizables y se combinan para crear componentes mayores hasta configurar una web completa.

### 5.2.2.2 Redux

Redux es una herramienta que suele asociarse con React, sirve para la gestión de estados en webs JavaScript.

Un estado, en el frontend, incluye las respuestas del servidor, la información cacheada y los datos generados directamente en local que no se han guardado en el servidor, así como el estado de la interfaz: rutas activas, spinners, controles de paginación...

El propósito de Redux es hacer predecibles los cambios de estado, imponiendo ciertas restricciones sobre cómo y cuándo pueden producirse las actualizaciones. Redux consigue que la gestión del estado sea transparente y determinista, lo que entre otras cosas aporta:

- Mejor comprensión de la evolución del estado en un momento dado
- Facilidad para incorporar nuevas características a la app
- Un nuevo abanico de herramientas de debugging (como el time travelling)
- Capacidad de reproducir un bug
- Mejoras en el proceso de desarrollo, pudiendo reiniciar la ejecución a partir de un estado concreto.

El diagrama de flujo que sigue Redux, desde una interacción, hasta que la aplicación actualiza la UI sería el siguiente:

- 1.El componente recibe un evento (click, por ejemplo) y emite una acción.
- 2.Esta acción, se pasa al store, que es donde se guarda el estado único.
- 3.El store comunica la acción junto con el estado actual a los reducers.
- 4.Los reducers, devuelven un nuevo estado, probablemente modificado en base a la acción.
- 5.Los componentes reciben el nuevo estado del store.

### 5.2.2.3 Axios

Axios es una librería JavaScript que puede ejecutarse en el navegador y que nos permite hacer sencillas las operaciones como cliente HTTP, por lo que podremos configurar y realizar solicitudes a un servidor y recibiremos respuestas fáciles de procesar.

Así que Axios es una alternativa que nos brinda multitud de ventajas:

- Está optimizado para facilitar el consumo de servicios web, API REST y que devuelvan datos JSON.
- De fácil utilización y como complemento perfecto para las páginas convencionales.
- Pesa poco, apenas 13KB minimizado. Menos aún si se envía comprimido al servidor.
- Compatibilidad con todos los navegadores en sus versiones actuales.

### 5.2.2.4 Material-UI

Material-UI es una eficiente herramienta para construir aplicaciones a nivel de diseño y animación.

Posee una amplia variedad de componentes para construir la interfaz de usuario de manera eficiente y sencilla.

### 5.2.2.5 Moment

Moment es una librería de manejo de fechas que te libra de usar el objeto Date de JavaScript.

Aparte de eso, Moment también mejora las capacidades de Date, permitiendo el uso de tiempos relativos, duraciones y horarios.

### 5.2.2.6 Google-Login

Google Login básicamente permite iniciar sesión con tu cuenta de Google en cualquier web que tenga implementada un sistema de autenticación.

Usa un protocolo de autorización llamado OAuth2 el cual permite que los usuarios autoricen a terceros a acceder a su información sin que estos tengan que conocer las credenciales del usuario.

### 5.2.3 Estructura

La estructura resultante del frontend sería la siguiente:

- **Actions.** Este directorio contiene todas las acciones que pueden hacer las distintas entidades de la web (usuarios, artículos...), dichas acciones realizan llamadas al controlador del backend, que será el encargado de realizar las distintas operaciones de un CRUD.
- **Api.** Contiene las distintas rutas de las acciones.

```
export const fetchArticles = (page) => API.get(`/articles?page=${page}`);
export const fetchArticlesBySearch = (searchQuery) => API.get(`/articles/search?searchQuery=${searchQuery}`);
export const fetchArticle = (id) => API.get(`/articles/${id}`);
export const createArticle = (newArticle) => API.post(`/articles`, newArticle);
export const updateArticle = (id, updatedArticle) => API.patch(`/articles/${id}`, updatedArticle);
export const likeArticle = (id) => API.patch(`/articles/${id}/likeArticle`);
export const deleteArticle = (id) => API.delete(`/articles/${id}`);

export const signIn = (formData) => API.post(`/user/signin`, formData);
export const signUp = (formData) => API.post(`/user/signup`, formData);
export const updateUser = (id, updatedUser) => API.patch(`/user/${id}`, updatedUser);

export const fetchCreditCard = (user) => API.get(`/creditCard/${user}`);
export const createCreditCard = (newCreditCard) => API.post(`/creditCard`, newCreditCard);
export const deleteCreditCard = (id) => API.delete(`/creditCard/${id}`);

export const fetchQuestions = () => API.get(`/forum`);
export const fetchQuestion = (id) => API.get(`/forum/${id}`);
export const createQuestion = (question) => API.post(`/forum`, question);
export const updateQuestion = (id, updatedQuestion) => API.patch(`/forum/${id}`, updatedQuestion);
export const deleteQuestion = (id) => API.delete(`/forum/${id}`);
export const answerQuestion = (id, answer) => API.patch(`/forum/${id}/answerQuestion`, answer);

export const fetchAnswer = (id) => API.get(`/answers/${id}`);
export const createAnswer = (answer) => API.post(`/answers`, answer);
export const updateAnswer = (id, updatedAnswer) => API.patch(`/answers/${id}`, updatedAnswer);
export const deleteAnswer = (id) => API.delete(`/answers/${id}`);
```

Figura 15. Rutas

- **Components.** Son los distintos componentes que conforman la interfaz de usuario, dentro de ellos se usan otros componentes proporcionados por Material-UI para construir la interfaz de forma sencilla.
- **Constants.** El fichero de constantes de cada acción.

```
export const CREATE = 'CREATE';
export const UPDATE = 'UPDATE';
export const DELETE = 'DELETE';
export const FETCH_ALL = 'FETCH_ALL';
export const FETCH_ARTICLE = 'FETCH_ARTICLE';
export const FETCH_BY_SEARCH = 'FETCH_BY_SEARCH';

export const START_LOADING = 'START_LOADING';
export const END_LOADING = 'END_LOADING';

export const AUTH = 'AUTH';
export const LOGOUT = 'LOGOUT';
export const UPDATE_USER = 'UPDATE_USER';

export const FETCH_CREDITCARD = 'FETCH_CREDITCARD';
export const CREATE_CREDITCARD = 'CREATE_CREDITCARD';
export const DELETE_CREDITCARD = 'DELETE_CREDITCARD';

export const FETCH_ALL_QUESTIONS = 'FETCH_ALL_QUESTIONS';
export const FETCH_QUESTION = 'FETCH_QUESTION';
export const CREATE_QUESTION = 'CREATE_QUESTION';
export const UPDATE_QUESTION = 'UPDATE_QUESTION';
export const DELETE_QUESTION = 'DELETE_QUESTION';

export const FETCH_ANSWER = 'FETCH_ANSWER';
export const CREATE_ANSWER = 'CREATE_ANSWER';
export const UPDATE_ANSWER = 'UPDATE_ANSWER';
export const DELETE_ANSWER = 'DELETE_ANSWER';
```

Figura 16. Constantes

- **Reducers.** Aquí se manejan los distintos estados de cada entidad.
- **Images.** Directorio para almacenar las imágenes de la interfaz.

## 5.3 Lado Servidor

### 5.3.1 Introducción

La parte servidor o backend es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata del conjunto de acciones que pasan en una web pero que no vemos como, por ejemplo, la comunicación con el servidor.

Los lenguajes más utilizados en el backend suelen ser PHP, .Net o NodeJS, este último ha sido la elección escogida para el desarrollo del proyecto.

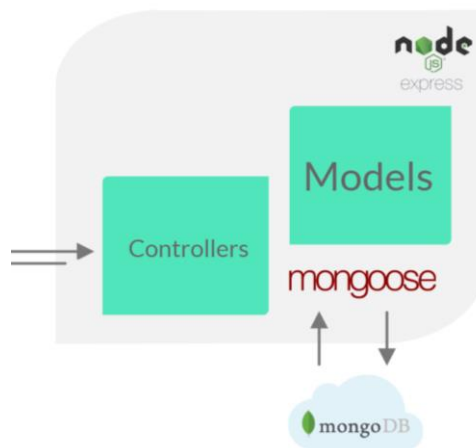


Figura 17. Estructura Servidor

### 5.3.2 Tecnologías y librerías

#### 5.3.2.1 NodeJS

Resumiendo de manera fácil qué es node.js podríamos decir que es un entorno de código abierto, multiplataforma y que ejecuta el código Javascript fuera de un navegador. Y es precisamente la necesidad de ejecutar este lenguaje del lado del servidor el que hace surgir Node.js.

En términos más técnicos podemos definirlo como un entorno de ejecución de Javascript que se orienta a eventos asíncronos (los eventos no dependen de que otros se hayan ejecutado previamente) y que puede construir aplicaciones en red escalables.

Node.js solo opera con un subproceso, lo que le permite soportar decenas de miles de conexiones al mismo tiempo mantenidas en un bucle de eventos ya que no depende de procesos previos.

Las ventajas de usar Node son:

- Tiene Javascript incorporado, un lenguaje sencillo de aprender y estructurado.

- Es Open Source por lo que, de manera libre, se pueden escoger módulos de sus librerías que nos faciliten el trabajo. Además, la comunidad de programadores en Node.js crece cada día siendo cada vez más grande.
- Actualmente es una de las plataformas de software más utilizadas superando a entornos y lenguajes conocidos como PHP o C.
- Tiene un menor tiempo de ejecución frente a otros sistemas. Esto lo consigue gracias a los eventos asíncronos ya que no ejecuta el código línea a línea, sino que procesa lo que puede en cualquier momento. Esto también le permite tener múltiples procesos ejecutándose y otros en espera, razón por la cual resulta especialmente efectiva en programaciones de aplicaciones en tiempo real o que sean amigables con el uso en dispositivos móviles.
- Escalabilidad: como hemos mencionado antes este es el punto fuerte de node.js. Esto facilita la creación de proyectos de gran envergadura donde se necesita agilidad para ejecutar multitud de procesos, recordemos que solo ejecuta uno sin crear un nuevo hilo para cada solicitud.
- La gran comunidad y el código abierto permiten además que las actualizaciones sean frecuentes aumentando su rendimiento y seguridad. Sin embargo, debido a estas frecuentes actualizaciones, la API donde se sustenta puede sufrir cambios que la hacen incompatible con versiones anteriores.

### 5.3.2.2 Express

Express es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles.

Posee miles de métodos de programa de utilidad HTTP y middleware a su disposición, la creación de una API sólida es rápida y sencilla.

Express proporciona una delgada capa de características de aplicación web básicas, que no ocultan las características de Node.js que tanto ama y conoce.

### 5.3.2.3 Mongoose

Mongoose es una librería para Node.js que nos permite escribir consultas para una base de datos de MongoDB, con características como validaciones, construcción de queries, middlewares, conversión de tipos y algunas otras, que enriquecen la funcionalidad de la base de datos.

La parte central del uso de Mongoose está en la definición de un esquema donde se indica la configuración de los documentos para una colección de MongoDB. Y aunque MongoDB es una base de datos NoSQL, donde los documentos se almacenan sin un esquema predefinido, el uso de un esquema te permite normalizar tu información, sin sacrificar la flexibilidad. Además, hace que la transición de SQL a NoSQL, sea más sencilla.

En palabras prácticas, Mongoose funciona como una capa adicional sobre MongoDB a través de la cual se implementan y automatizan muchas de las tareas habituales de trabajar con una base de datos.



### 5.3.3 Estructura

La estructura resultante del frontend sería la siguiente:

- **Controllers.** Como su nombre indica el controlador es la parte web dónde debe ir el código que recibe y trata las peticiones del cliente. Según las acciones del usuario en la web, el controlador solicitará al modelo una u otra información y mostrará una vista u otra.
- **Models.** El modelo es todo aquel código de nuestra aplicación que interactúa con la base de datos o que trata la persistencia de la información. La lectura, inserción, actualización y borrado de datos son tareas comunes de los modelos.
- **Routes.** Aquí se definirán las distintas rutas para cada operación.
- **Middleware.** Aquí ira un bloque de código que se ejecuta entre la petición que hace el usuario hasta la petición que llega al servidor. Sirve para dar acceso a una determinada URL a un usuario logueado.

## 6. Despliegue

### 6.1 Hosting

#### 6.1.1 Introducción

La parte del despliegue se ha dividido en dos, por una parte, se ha decidido desplegar la parte cliente en Netlify y la parte servidor en Heroku.

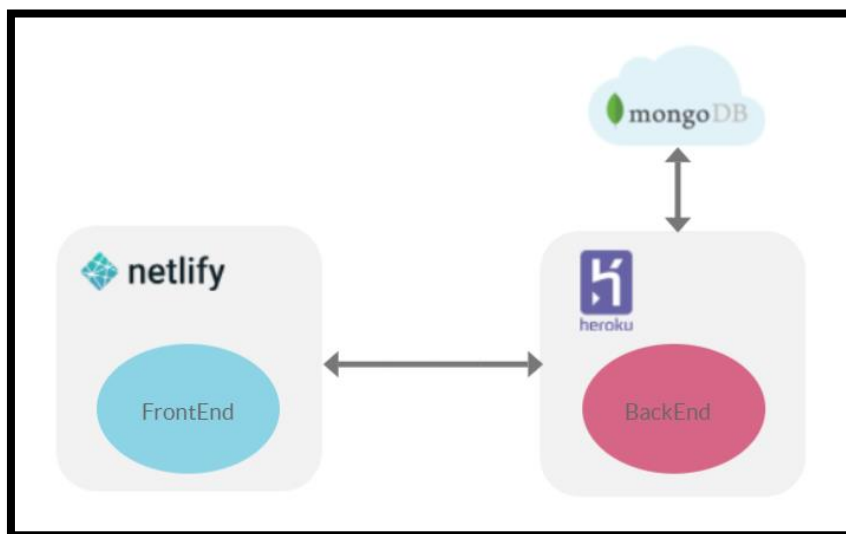


Figura 18. Estructura despliegue

#### 6.1.2 Despliegue de la parte cliente

Para el despliegue de la parte cliente se ha usado Netlify.

Netlify es una plataforma que nace para automatizar proyectos webs estáticos. Aúna las tareas de integración continua y despliegue de infraestructura web en un solo flujo de ejecución.

El desarrollo web se caracteriza por, entre otras cosas, el cambio continuo en los diseños. Constantemente se añaden nuevos elementos o se modifican los ya añadidos. Es muy importante poder ver en todo momento qué aspecto tiene o ha tenido nuestra web.

Con Netlify el proceso de despliegue se convierte en algo muy sencillo: únicamente hay que enlazar la herramienta a un repositorio Git donde se encuentren los archivos que componen la

página web y crear un deploy que provocará que la aplicación se compile y se despliegue automáticamente en una determinada URL.

Su potencia viene dada por su capacidad de despliegue continuo. Todos los cambios que se hayan realizado en la aplicación web constituyen versiones desplegadas de la misma, a las que se puede tener acceso en cualquier momento. Si la versión actual de la web no nos convence, podemos dejarla en un estado en el que se encontraba anteriormente.

El procedimiento de despliegue es muy sencillo, simplemente tendremos que crear la build de la parte cliente, eso generará una carpeta a la que simplemente debemos arrastrar dentro de la interfaz de Netlify.

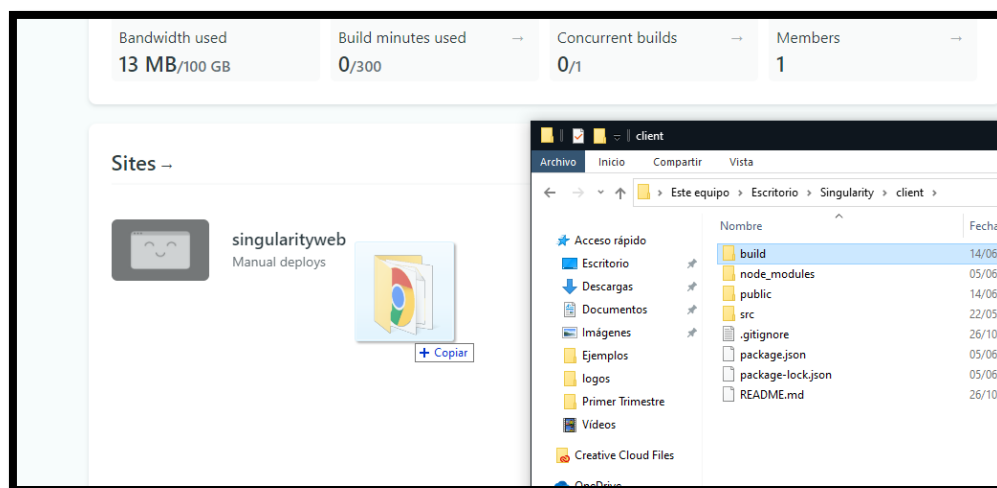


Figura 19. Proceso de despliegue en Netlify

Hecho esto la web procesará la build y generará un enlace con nuestra web desplegada.

### 6.1.3 Despliegue de la parte servidor

Para el despliegue de la parte servidor se ha usado Heroku.

Heroku es una plataforma en la nube que permite alojar tu web sin coste ninguno y absenta a los desarrolladores de preocuparse por la infraestructura.

A diferencia de otras plataformas permite desarrollar prácticamente con cualquier lenguaje de programación: Ruby, Java, PHP, NodeJS...

También permite desplegar versiones, hacer rollback, gestionar dependencias...

Dispone de los denominados add ons, gracias a los que podemos añadir funcionalidad extra a nuestras aplicaciones de forma realmente sencilla, por ejemplo, memcached, redis, postgres, mongolab, etc.

Principales ventajas de Heroku:

- Es gratuito para aplicaciones de poco consumo.
- Permite el uso de diferentes lenguajes de programación.
- Es una plataforma fácil de usar.
- Integra varios servicios dentro de su estructura.
- Las actualizaciones en Heroku no afectan a nuestra plataforma informática.

- Se puede tener acceso desde cualquier lugar y dispositivo compatible con la computación en la nube.

El despliegue de la parte servidor en Heroku también ha sido muy sencillo. Simplemente he tenido que conectar mi repositorio de GitHub.

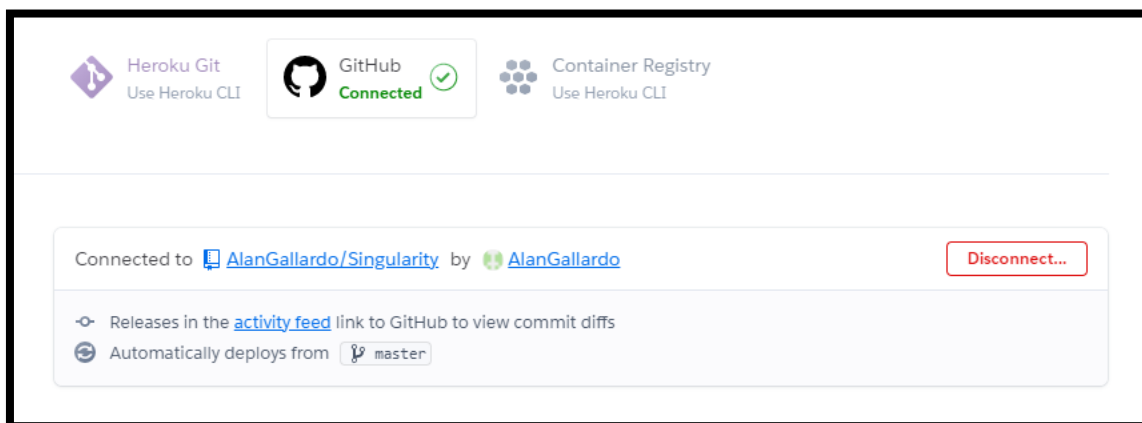


Figura 20. Proceso de despliegue en Heroku

## 6.2 Documentación

Para generar la documentación de mi código de manera sencilla, he usado **JSDoc**.

JSDoc es un generador de documentación para el lenguaje de Javascript, muy similar a phpDocumentor o Javadoc que funciona para Java. JSDoc analiza el código Javascript y automáticamente genera una página HTML con la documentación en ella.

Para generar la documentación en el código he usado **Document This**, el cual añade encima de una función su correspondiente bloque de documentación con tan solo un click.

El resultado de la documentación creada por JSDoc es el siguiente:

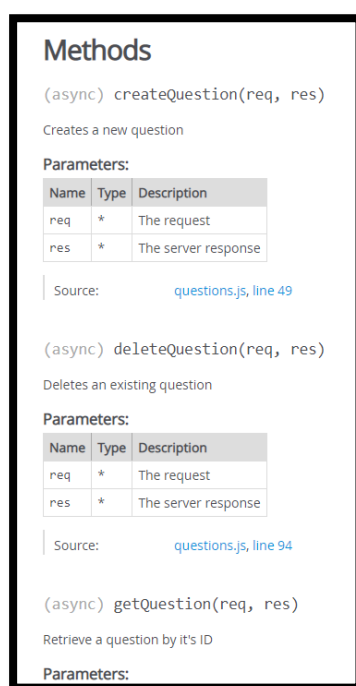


Figura 21. Documentación

## 6.3 Control de versiones

Para el control de versiones se ha usado GitHub.

GitHub es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador, y que fue comprada por Microsoft en junio del 2018. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas, y que como usuario no sólo puedas descargar la aplicación, sino también entrar a su perfil para leer sobre ella o colaborar con su desarrollo.

Como su nombre indica, la web utiliza el sistema de control de versiones Git diseñado por Linus Torvalds. Un sistema de gestión de versiones es ese con el que los desarrolladores pueden administrar su proyecto, ordenando el código de cada una de las nuevas versiones que sacan de sus aplicaciones para evitar confusiones. Así, al tener copias de cada una de las versiones de su aplicación, no se perderán los estados anteriores cuando se va a actualizar.

Así pues, Git es uno de estos sistemas de control, que permite comparar el código de un archivo para ver las diferencias entre las versiones, restaurar versiones antiguas si algo sale mal, y fusionar los cambios de distintas versiones. También permite trabajar con distintas ramas de un proyecto, como la de desarrollo para meter nuevas funciones al programa o la de producción para depurar los bugs.

Las principales características de la plataforma es que ofrece las mejores características de este tipo de servicios sin perder la simplicidad, y es una de las más utilizadas del mundo por los desarrolladores. Es multiplataforma, y tiene multitud de interfaces de usuario.

Así pues, GitHub es un portal para gestionar las aplicaciones que utilizan el sistema Git. Además de permitirte mirar el código y descargar las diferentes versiones de una aplicación, la plataforma también hace las veces de red social conectando desarrolladores con usuarios para que estos puedan colaborar mejorando la aplicación.

El flujo de trabajo con Git ha sido el siguiente:

1. Crear el repositorio remoto en Github.
2. Inicializar el repositorio local con Git.
  - *Git init*
3. Crear una nueva rama para añadir cambios nuevos específicos.
  - *Git branch [rama]*
  - *Git checkout [rama]*
4. Desarrollar el código específico.
5. Una vez terminada la implementación nueva, fusionar los cambios a la rama principal
  - *Git checkout master*
  - *Git merge [rama]*
6. Cada día guardaba los progresos en el repositorio local.
  - *Git add .*
  - *Git commit -m [mensaje]*
7. Cada semana subía los ficheros al repositorio remoto.
  - *Git push -u origin master*

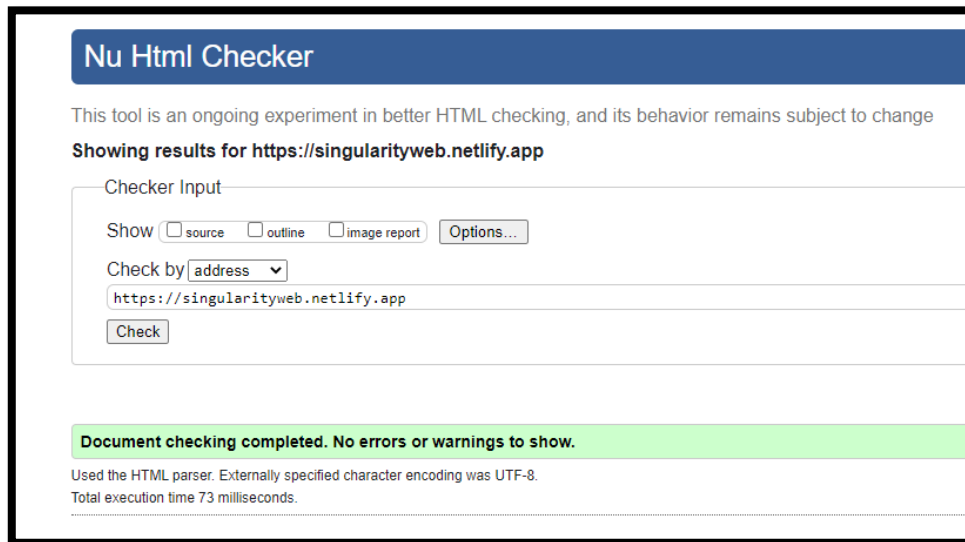
## 7. Validaciones

### 7.1 Introducción

La fase de evaluación y pruebas concluye el ciclo de vida del proyecto y lo prepara para subirlo al servidor y que se ejecute. Esta fase mide el nivel de calidad que ofrece al usuario la aplicación creada. Las herramientas que vamos a usar para realizar estas pruebas son gratuitas y las podemos encontrar online.

### 7.2 Validación de enlaces

En la web <http://jigsaw.w3.org/> podemos validar los enlaces de nuestra web, en mi caso no he tenido problemas:

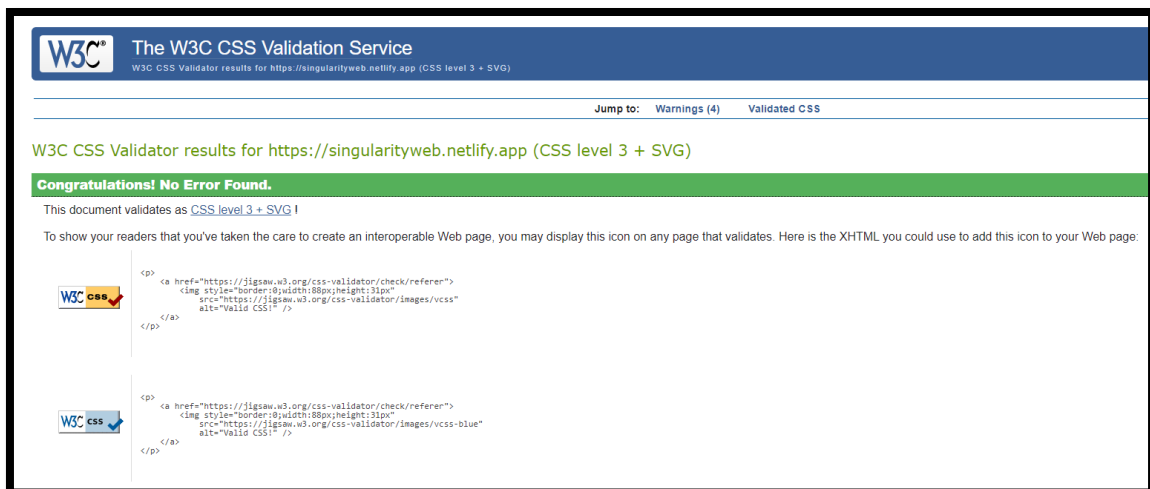


The screenshot shows the Nu Html Checker interface. At the top, it says "Nu Html Checker". Below that, a note states: "This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change". The main heading is "Showing results for https://singularityweb.netlify.app". Under "Checker Input", there are checkboxes for "source", "outline", and "image report", along with an "Options..." button. The "Check by" dropdown is set to "address". The URL "https://singularityweb.netlify.app" is entered in the input field. A "Check" button is visible. Below the input section, a green banner reads "Document checking completed. No errors or warnings to show." At the bottom, it says "Used the HTML parser. Externally specified character encoding was UTF-8. Total execution time 73 milliseconds."

Figura 22. Validación de enlaces

### 7.3 Validación de estilos

En la web <http://jigsaw.w3.org/css-validator/> podemos validar nuestros distintos estilos si introducimos la URL de nuestra web desplegada. En mi caso no he tenido ningún problema:



The screenshot shows the W3C CSS Validation Service interface. At the top, it says "The W3C CSS Validation Service". Below that, it says "W3C CSS Validator results for https://singularityweb.netlify.app (CSS level 3 + SVG)". There are links for "Jump to: Warnings (4) Validated CSS". The main heading is "W3C CSS Validator results for https://singularityweb.netlify.app (CSS level 3 + SVG)". Below that, a green banner reads "Congratulations! No Error Found." At the bottom, it says "This document validates as CSS level 3 + SVG". Below that, it says "To show your readers that you've taken the care to create an interoperable Web page, you may display this icon on any page that validates. Here is the XHTML you could use to add this icon to your Web page." There are two examples of XHTML code snippets, one for the "Valid CSS" icon and one for the "Valid CSS" icon with a blue border.

Figura 23. Validación de estilos

## 7.4. Validación de la resolución

La resolución recomendada para visualizar la web correctamente es 1920x1080 píxeles, ya que el desarrollo de la web ha partido de esa resolución. No obstante, si reducimos la resolución los elementos de la web deberían ajustarse al contenido, ya que el diseño es en parte responsive.



Figura 24. Singularity Móvil

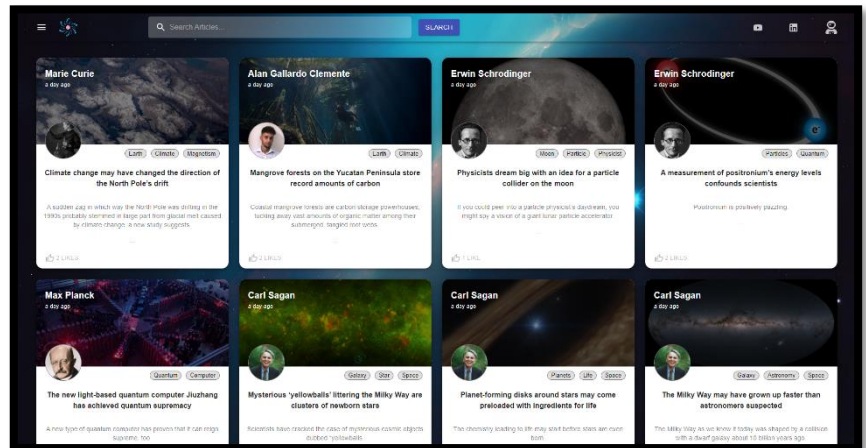


Figura 25. Singularity Desktop

## 8. Conclusiones

### 8.1 Retrospectiva del trabajo realizado

En este apartado se realiza un resumen del trabajo realizado desde la idea del proyecto hasta su finalización. El primer paso fue dar con una idea, me decanté por una web de artículos, pero quería darle un toque de ciencia y que no fueran artículos de cualquier cosa. En cuanto al nombre de la web, Singularity, viene dado por la propia definición de la palabra en términos astrofísicos, toda la masa de un agujero negro se concentra en un punto matemático llamado Singularidad. Con esto se puede construir la analogía, un punto (la web) donde se concentran los artículos.

Ya con la idea y nombre en mente había que construir los modelos UML para que sirviesen de guía en la futura implementación. Luego pasé a la interfaz gráfica, usé AdobeXd para realizar un prototipo de web. Con los diagramas y el prototipo construido podía pasar a programar. Hice el scaffolding oportuno para organizar y definir bien el frontend del backend. Este sería el flujo que seguí cuando implementaba, por ejemplo, los artículos:

1. Crear su modelo correspondiente, con todas sus propiedades.
2. Crear su controlador correspondiente, con todas sus operaciones (obtener, crear, actualizar, borrar).
3. Añadir las correspondientes rutas que apuntan a las distintas operaciones.
4. Implementar las rutas en el frontend, que se conectan con las acciones.
5. Implementar las acciones, las cuales se conectarán con el controlador.

6. Añadir las distintas acciones en el reducer, el cual te devuelve el estado de un componente cuando se solicite.
7. Crear e implementar el componente en la interfaz gráfica.
8. Aplicarle estilos.

Este sería, de forma resumida, el flujo de trabajo al implementar los cinco modelos existentes en la web. Después de eso también se han tenido que implementar otros componentes para hacer la web más vistosa y sólida.

## 8.2 Posibles ampliaciones

A continuación, se listan una serie de ampliaciones que podrían implementarse en un futuro:

- Crear otro tipo de usuario llamado usuario administrador, el cual revisará los artículos que otros usuarios creen antes de publicarlos.
- Comentarios de otros usuarios en los artículos.
- Algún tipo de sistema de puntuación cada vez que un usuario premium suba un artículo.

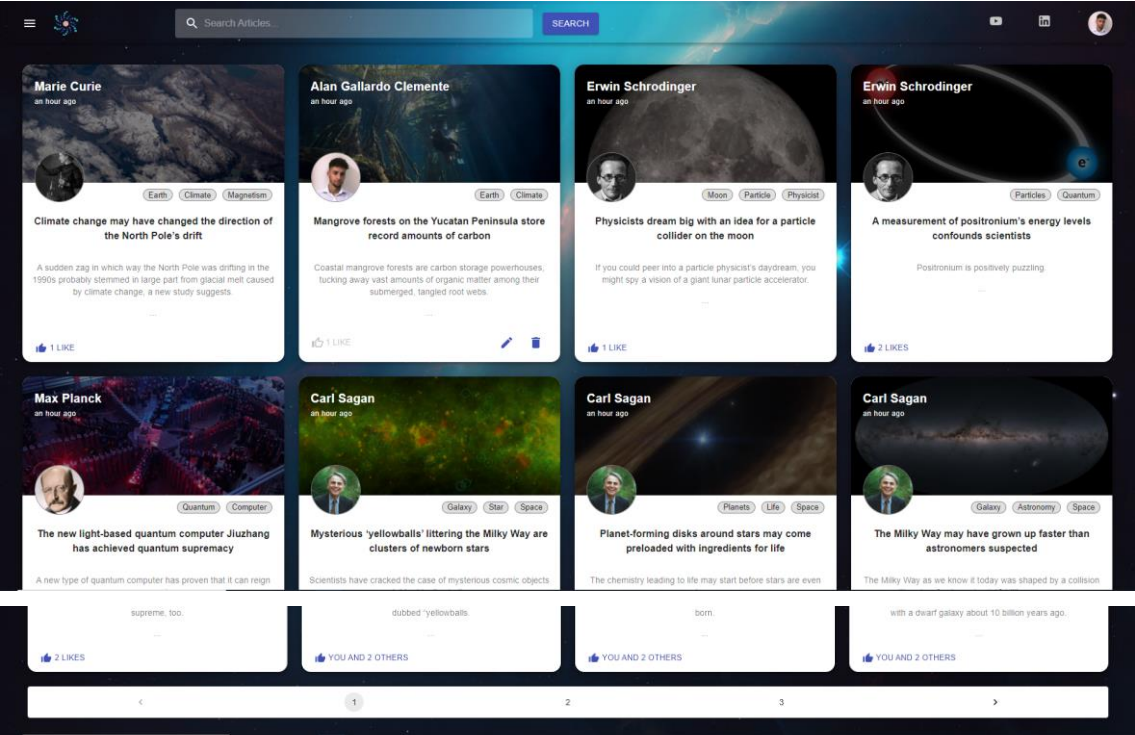
## 9. Bibliografía

- ReactJS  
<https://es.reactjs.org/>
- Redux  
<https://es.redux.js.org/>
- Axios  
<https://github.com/axios/axios>
- Material-UI  
<https://material-ui.com/>
- NodeJS  
<https://nodejs.org/es/>
- Express  
<https://expressjs.com/es/>
- Mongoose  
<https://mongoosejs.com/>
- GitHub  
<https://github.com/axios/axios>
- MongoDB  
<https://www.mongodb.com/>
- Trello  
<https://trello.com/>
- Wikipedia, la enciclopedia libre  
<https://es.wikipedia.org/>
- Youtube  
<https://es.wikipedia.org/>



- Xataka  
<https://www.xataka.com/>
- Stack Overflow  
<https://stackoverflow.com/>
- Zeet  
<https://zeet.co/>
- Tutorial de React I  
<https://es.reactjs.org/tutorial/tutorial.html>
- Tutorial de React II  
<https://www.youtube.com/watch?v=w7ejDZ8SWv8>
- Tutorial de React III  
<https://www.youtube.com/watch?v=O6P86uwfdR0>
- Tutorial de React IV  
<https://www.youtube.com/watch?v=0ZJgIjIuY7U>
- Tutorial de React V  
<https://www.youtube.com/watch?v=vAvCcjsAGDY>
- Tutorial de MERN Stack  
<https://www.youtube.com/watch?v=7CqJlxBYj-M>

10. Anexo



Search Articles

SEARCH

Search Articles

SEARCH

Profile ^

Name

Alan Gallardo Clemente

Email

alan.gallardo.ahu@iescampanillas.com

SUBSCRIBE

Being a singularity subscriber allows you to post articles (You need to add a payment method).

Posted Articles ^

Alan Gallardo Clemente

an hour ago

Earth

Climate

Mangrove forests on the Yucatan Peninsula store record amounts of carbon

Coastal mangrove forests are carbon storage powerhouses, sucking away vast amounts of organic matter among their submerged, tangled root wells.

1 LIKE

Payment Method ^

Number\*\*

0000111122223333

Full name\*

Alan Gallardo Clemente

Expiry Date\*

05/25

CVV\*

100

Page 44 | 47

