

SpeedPost

Memoria Técnica del Trabajo Fin de Grado



Julio García Serrano

ÍNDICE

- ¿Que es SpeedPost?
- Tecnologías empleadas
- Esquema de la base de datos
- Selección de colores
- Métodos, funciones y hooks
- Hosting
- Guia de Uso

¿Que es SpeedPost?

SpeedPost es una red social en tiempo real la cual permite a sus usuarios interactuar mediante publicaciones de texto o imagen las cuales pueden ser comentadas entre ellos en el foro principal. Los usuarios también pueden crear salas de conversación donde debatir de un tema en concreto. Además en la última versión se incluyó un “paint” rudimentario el cual permite a los usuarios dibujar y compartir estos dibujos en la plataforma.

Tecnologías empleadas

Para el desarrollo de la aplicación he empleado diferentes tecnologías, las principales y las cuales conforman el esqueleto de la aplicación son React.js y Firebase.

React encargándose de la parte front de la aplicación y Firebase de ofrecer el backend y hosting para la misma.

Para la parte de los estilos y la apariencia de la aplicación se ha usado íntegramente css.

Ahora voy a explicar más en profundidad todas las tecnologías que encontramos presentes en la aplicación.

- React.js

React es uno de los frameworks más usados y conocidos para javascript siendo propiedad de Facebook el cual fue lanzado en el 23 de mayo de 2013 y hasta día de hoy a logrado posicionarse como uno de los frameworks más importantes estando presente en aplicaciones como Instagram o Facebook entre las más relevantes.

Lo que hace tan útil a React es la facilidad que da a los desarrolladores para crear vistas declarativas de manera sencilla lo cual hace un código más intuitivo y más fácil de depurar también permite la creación de componentes encapsulados los cuales son capaces de manejar su estado de manera independiente lo que te da una interfaz de usuario más compleja los cuales serán actualizadas y renderizados por react en tiempo real inclusive es capaz de renderizar el servidor usando node.

Además de que otorga mucha facilidad a la hora de pasar información entre componentes y de tratar esta misma utilizando los **Hooks** de React los cuales son una nueva API de la librería de React.

- [HTML](#)

HTML, siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). El lenguaje HTML es un estándar reconocido en todo el mundo y cuyas normas define un organismo sin ánimo de lucro llamado World Wide Web Consortium, más conocido como W3C. Como se trata de un estándar reconocido por todas las empresas relacionadas con el mundo de Internet, una misma página HTML se visualiza de forma muy similar en cualquier navegador de cualquier sistema operativo. El propio W3C define el lenguaje HTML como "un lenguaje reconocido universalmente y que permite publicar información de forma global". Por convención, los archivos de formato HTML usan la extensión .htm o .html.

- [CSS](#)

Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C es el encargado de formular la especificación de las hojas de estilo que

servirá de estándar para los agentes de usuario o navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML. En este último podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "style". Las ventajas de utilizar CSS (u otro lenguaje de estilo) son:

- Control centralizado de la presentación de un sitio web completo, con lo que se agiliza de forma considerable la actualización del mismo.
- Los navegadores permiten a los usuarios especificar su propia hoja de estilo local que será aplicada a un sitio web remoto, con lo que aumenta considerablemente la accesibilidad.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre.
- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.

- [JavaScript](#)

JavaScript es un lenguaje interpretado utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java. Sin embargo, al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia. Es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM (Modelo de Objetos del Documento). JavaScript se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

- Firebase

Firebase se trata de una plataforma móvil creada por Google, cuya principal función es desarrollar y facilitar la creación de apps de elevada calidad de una forma rápida. Fue creada en el año 2010 por James Tamplin, Andrew Lee y adquirida por google en el año 2014.

Firebase lo que nos permite es utilizar backed as a services para nuestras aplicaciones facilitandonos el desarrollo de las mismas además de proporcionarnos una base de datos la cual se actualiza en tiempo real.

Pero esas son solo unas de las muchas cualidades que nos ofrece Firebase, otra de sus más famosas es el hosting que ofrece para las aplicaciones basadas en su base de datos.

Estas son algunas de las funcionalidades extra que nos ofrece firebase

- Autenticación de usuarios.
- Almacenamiento en la nube.
- Crash Reporting.
- Test Lab.
- Remote Config.
- Cloud Messaging.
- Hosting.

Para la aplicación también emplearemos una funcionalidad de Firebase que es la autenticación de usuarios la cual nos deja elegir entre múltiples opciones para el registro y logeo de usuarios, de entre ellas me he decantado por el provider de Google el cual me ofrece un registro y login muy rápido y sencillo usando únicamente un correo. Lo bueno de esto es que gracias al provider obtengo información del usuario como su foto de perfil o correo la cual puedo reutilizar en mi aplicación.

- **Firestore**

Cloud Firestore es una base de datos de documentos NoSQL que permite almacenar, sincronizar y consultar fácilmente datos

Usa colecciones y documentos para estructurar los datos con facilidad. Crea jerarquías para almacenar datos relacionados y recuperar los datos que necesitas mediante consultas expresivas de manera sencilla. Todas las consultas se escalan con el tamaño del conjunto de resultados (y no con el del conjunto de datos), por lo que tu aplicación está lista para escalar desde el primer día.

Cloud Firestore se distribuye con SDK web y para dispositivos móviles, y un conjunto completo de reglas de seguridad para que puedas acceder a tu base de datos sin necesidad de crear tu propio servidor. Con Cloud Functions, nuestro producto de procesamiento sin servidores, puedes ejecutar un código de backend alojado que responda a los cambios de datos en tu base de datos. Por supuesto, también puedes acceder a Cloud Firestore con las bibliotecas cliente tradicionales (es decir, Node, Python, Go y Java)

Con Cloud Firestore, puedes sincronizar automáticamente los datos de tu app en distintos dispositivos. Te notificaremos los cambios en los datos a medida que ocurran para que puedas crear experiencias colaborativas y apps en tiempo real con facilidad. Los usuarios pueden acceder y realizar cambios en sus datos en cualquier momento, incluso sin conexión. El modo sin conexión está disponible para iOS, Android y la Web.

- **Cloud Storage para Firebase**

Cloud Storage para Firebase se creó para los desarrolladores de apps que necesitan almacenar y entregar contenido generado por usuarios, como fotos o videos.

Cloud Storage para Firebase es un servicio de almacenamiento de objetos potente, simple y rentable construido para el escalamiento de Google. Los SDK de Firebase para Cloud Storage agregan la seguridad de Google a las operaciones de carga y descarga de archivos de tus apps de Firebase, sin importar la calidad de la red.

- Funciones clave

Operaciones robustas	Los SDK de Firebase para Cloud Storage realizan las operaciones de carga y descarga sin importar la calidad de la red. Las cargas y descargas son robustas, lo que significa que se reinician en el punto en el que se interrumpieron para así ahorrar tiempo y ancho de banda a los usuarios.
Seguridad sólida	Los SDK de Firebase para Cloud Storage se integran con Firebase Authentication a fin de brindar autenticación intuitiva y sencilla para los programadores. Puedes usar nuestro modelo de seguridad declarativa para permitir el acceso según el nombre de archivo, el tamaño, el tipo de contenido y otros metadatos.
Gran escalabilidad	Cloud Storage se diseñó con el fin de escalar a exabytes si tu app se vuelve viral. Puedes pasar de la fase de prototipo a la de producción con facilidad mediante la

misma infraestructura que respalda a Spotify y Google Fotos.

- [Firebase Hosting](#)

Con Firebase Hosting, puedes implementar una página de destino de una app para dispositivos móviles, una aplicación web de una sola página o una app web progresiva sin complicaciones.

El contenido se publica rápidamente, sin importar la ubicación del usuario. Los archivos implementados en Firebase Hosting se almacenan en caché en SSD ubicados en servidores perimetrales de una CDN en todo el mundo.

Firebase Hosting aprovisiona y configura automáticamente un certificado SSL para cada sitio que implementes. Conecta un dominio personalizado con una verificación simplificada.

Solo necesitas un comando para implementar tu app en la Web desde un directorio local. Consulta el historial de implementaciones y revierte la versión a una anterior desde Firebase console.

- [Visual Studio Code](#)

Para desarrollar esta aplicación me he decantado por utilizar Visual Studio Code ya que me facilita mucho la organización y navegación de archivos , además de poder emplear distintos plugins que me ayudan en el desarrollo de la app, incluye también una terminal propia la cual de manera sencilla y cómoda me permite introducir los comandos necesarios para la seguir con el desarrollo de la aplicación como comando de inicio de la app o de instalación de paquetes npm.

- **Adobe XD**

Adobe XD es una herramienta de la suite Adobe la cual he empleado para hacer un prototipo de la maquetación y navegación de la aplicación y así poder guiarme en el desarrollo de la misma .

- **InkScape**

InkScape es un software de dibujo vectorial el cual voy a emplear para diseñar el logo de la aplicación en formato SVG.

- **Paquetes NPM**

En este proyecto he instalado algunos paquetes npm que me han ayudado con ciertos aspectos de la aplicación.

todos los comandos de instalacion han sido obtenidos de <https://www.npmjs.com>

- **Material-ui**

```
$ npm install @material-ui/core
```

Material-ui es un framework de React que permite utilizar componentes ya creados para facilitar el diseño de la aplicación.

Yo he utilizado este framework para distintos iconos que empleo en la aplicación como el botón de subir fotos o el de pintar.

- **Firebase**

```
npm install --save firebase
```

Instalo el paquete de firebase para poder tener acceso a todas las herramientas y métodos que me ofrece firebase.

- **React Canvas Draw**

```
npm install react-canvas-draw --save
```

Con este paquete npm consigo poder incorporar un canvas y además que pueda ser dibujado en React ya que de forma nativa en JS no logre hacer que funcione.

- Use Sound

```
npm install use-sound
```

Use Sound es un paquete de npm que al igual que React Canvas Draw tuve que instalar ya que no logre que funcionara de forma nativa con JS.

Base de Datos

La base de datos de la aplicación está dividida en 5 tablas aunque Firebase al ser NoSQL no se divide en tablas y campos como tal si no en colecciones y documentos en formato JSON.

- **Users** : Esta tabla se rellena con la información proporcionada por el provider de Google y forma parte de la herramienta de Auth que me da Firebase

Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario ↑
garciaserranojulio@gmail....		25 abr. 2021	1 jun. 2021	8T8c54JSxShHTm22mrKsyoMGf4...
julio.garcia.alu@iescamp... ...		25 abr. 2021	1 jun. 2021	xJXwDV9mORxYnKfUijhWlIEzkl1
juliusgarciaserrano@gmail....		27 abr. 2021	18 may. 2021	xUlgKMWRP5dkmR1BAhPyKPQ50...

- **Post** : En esta tabla se guardan los post de los distintos usuarios , ya sean posts únicamente de texto o posts que cuenten con su propia imagen

```
content: "hola"

date: "Tue Jun 01 2021"

profileUrl: "https://lh3.googleusercontent.com/a/AOh14GhvxFH-
PiuiYhHK_X4Z5HK7NN9ie-VOJrfi0ge4EoA=s96-c"

timestamp: 1 de junio de 2021, 18:52:49 UTC+2

userId: "8T8c54JSxShHTm22mrKsyoMGf4h2"

username: "garciaserranojulio"
```

- **Comments** : Es una “subtabla” de Post donde como su nombre indica se guardan en un campo Array los comentarios de los usuarios en los distintos Posts.
- **Rooms** : En esta tabla se almacena información referente al usuario que creó la sala , la temática de la sala y su fecha de creación, además de contener en ella la tabla de messages.

```
date: "Thu May 13 2021"

profileUrl: "https://lh3.googleusercontent.com/a/AOh14GhvxFH-
PiuiYhHK_X4Z5HK7NN9ie-VOJrfi0ge4EoA=s96-c"

timestamp: 13 de mayo de 2021, 18:37:36 UTC+2

title: "Programming"

userId: "8T8c54JSxShHTm22mrKsyoMGf4h2"

username: "garciaserranojulio"
```

- **Messages** : Aquí se almacenan los mensajes de cada sala en concreto ya sean mensajes de texto con fotos .

```

content: "buenas"

profileUrl: "https://lh3.googleusercontent.com/a/AOh14GhClHbq8rA17v-
qi25tEEudHuRADLWdQF6U8qP0=s96-c"

timestamp: 13 de mayo de 2021, 23:52:13 UTC+2

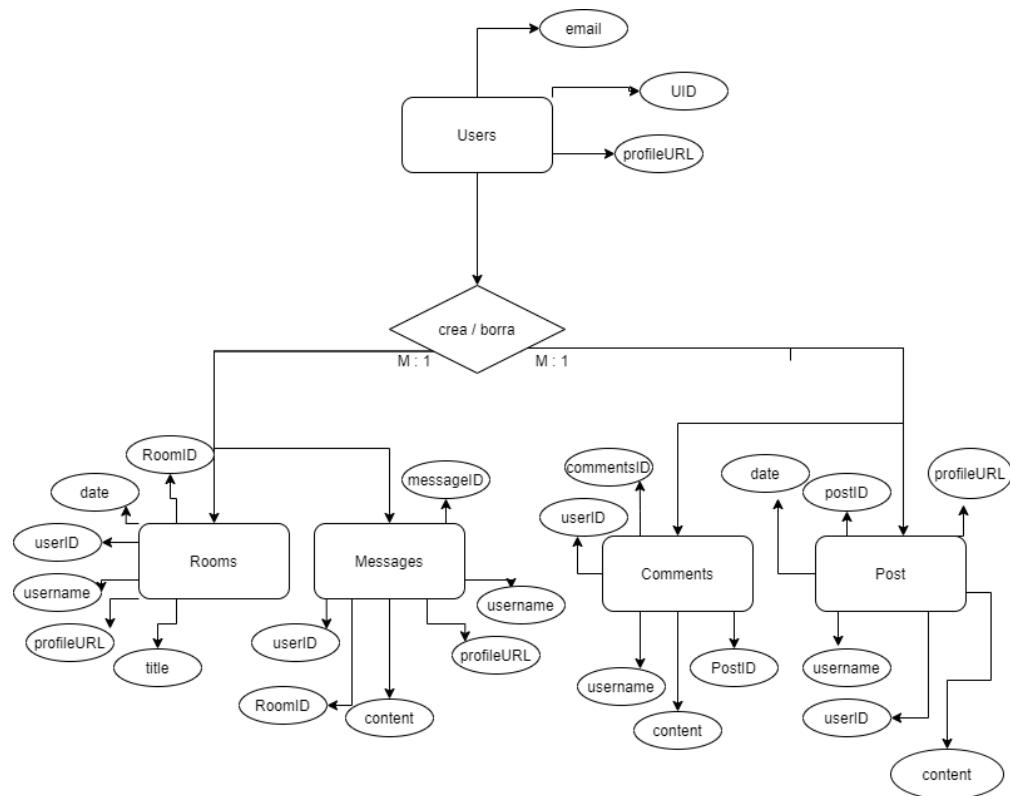
userId: "xUigKMWRP5dkmR1BAhPyKPQ50SI1"

username: "juliusgarciaserrano"

```

Estas serían las 5 tablas que conforman la base de datos de la aplicación.

Esquema de la base de datos



Diseño de la Aplicación

Para el diseño de la aplicación me he inspirado en las redes sociales de Instagram y Twitter dando como resultado una estructura similar a la de Instagram pero con una gama de colores similar a la de Twitter, intentando a su vez crear una interfaz sencilla de entender para el usuarios utilizando elementos visuales para algunos de los botones y que así se intuya que hace cada botón o apartado de la aplicación

Enlazo [aquí](#) un video donde se muestra el diseño y funcionalidad de la aplicación.

Selección de colores

Como ya he mencionado anteriormente para el diseño me he basado tanto en instagram como en twitter y de ahí es de donde he sacado los colores para mi aplicación.

Para el background elegí un whiteSmoke que lograse contraste con el resto de componentes que tienen un fondo blanco puro.

Para el navbar he usado un tono de azul apagado que de ese guiño al azul característico de twitter.

Métodos , Funciones y Hooks

- User Context

```
import { createContext, useState } from 'react';

export const UserContext = createContext();

export const UserContextProvider = (props) => {
  const [user, setUser] = useState(null);

  return (
    <UserContext.Provider value={{ user, setUser }}>
      {props.children}
    </UserContext.Provider>
  );
};
```

La función del UserContext es la de almacenar y proveer a toda la aplicación de la información del usuario que proviene del provider de Google

```
function App() {
  return (
    <UserContextProvider>
      <div className="App">
        <Home/>
      </div>
    </UserContextProvider>
  );
}
```

Añadimos el <UserContextProvider> al archivo App de nuestra aplicación para proveer de la información del usuario a todos los componentes.

- Google Provider

El proveedor de Google nos ayuda ofreciéndonos un login de manera sencilla para nuestro usuarios, esto lo obtenemos gracias al SDK de firebase pero siempre y cuando lo hayamos activado previamente en nuestra en nuestra consola de firebase.

Proveedores de acceso		
Proveedor		Estado
✉ Correo electrónico/contraseña		Inhabilitado
📞 Teléfono		Inhabilitado
🇬 Google		Habilitada

Una vez habilitado lo podemos llamar desde el archivo de configuración de firebase.js

```
const firebaseApp = firebase.initializeApp(firebaseConfig);

const db = firebaseApp.firestore();

const auth = firebase.auth();

const storage = firebase.storage();

const provider = new firebase.auth.GoogleAuthProvider();

export {db, auth, provider, storage};
```

En el archivo auth.js importamos auth y provider para nuestro login con Google

```

import {auth , provider} from "../firebase";

export const signInWithGoogle = async () => {
let user;
| await auth.signInWithPopup(provider)
.then((res) =>{
    console.log(res.user);
    user = res.user;
})
.catch((error) => {
    console.log(error.message);
});

return user;
};

```

Esta función será la que devuelva el usuario que luego se almacenará en el context

```

export default function SiginButton() {
    const [user, setUser] = useContext(UserContext).user;

    const signInClick = async () => {
        //set the user when the user is loged
        let userBySingIn = await signInWithGoogle()

        if(userBySingIn) setUser(userBySingIn);
        console.log(userBySingIn);
    };

    return (
        <div className="signInButton" onClick={signInClick}>
            <p>Sign In With Google</p>
        </div>
    )
}

```

Utilizando también el auth.js implemento la función de logout

```
export const logout = async () => {
  let logout_sucess;

  await auth.signOut()
    .then(() =>{
      logout_sucess = true;
    })
    .catch((error) => {
      console.log(error.message);
    })

  return logout_sucess;
}
```

Esta función de logout irá localizada en el navbar y solo será visible cuando el usuario esté logeado

```
export default function Navbar() {
  const [user, setUser] = useContext(UserContext).user

  return (
    <div className="navbar">
      <img src={logo} className="App-logo" alt='logo' />
      <p style={{paddingLeft: '4vh', fontSize: '3vh'}}>SpeedPost</p>
      <div className="navbarOptions">
        <p></p>
        {user ? <LogoutButton style={{cursor:'pointer', fontSize:'2.5vh'}}/> : <p></p>}
        {user ? <img className='imgProfile' src={user.photoURL}/> : <p></p>}
      </div>
    </div>
  );
}
```

Funcionalidad del LogoutButton

Consiste en una arrow function asíncrona que espera la respuesta del método logout

una vez que la obtiene setea el usuario a lo obtenido por el logout y una vez hecho esto recarga la pagina lo cual hace que se resetee la sesion.

```
export default function LogoutButton() {
  const [user, setUser] = useContext(UserContext).user;
  const logoutClick = async () => {
    let userLogout = await logout();
    setUser(userLogout);

    document.location.reload();

    // Let userBySingIn = await signInWithGoogle()

    // if(userBySingIn) setUser(userBySingIn);
  }
}
```

- Create Post

Image Preview: una de las funciones principales que tiene la creación de post es la preview que ofrece de la imagen seleccionada , esto sirve para confirmarle al usuario que la imagen que selecciono para subir esta la correcta.

```
const handleChange = (e) => {
  //generate an image preview
  if(e.target.files[0]){
    setImage(e.target.files[0]);

    var selectedImageSrc = URL.createObjectURL(e.target.files[0]);

    var imagePreview = document.getElementById("image-preview");
    imagePreview.src = selectedImageSrc;
    imagePreview.style.display = "block";
  }
}
```

Create Post

what's happening?



Create Post : A la hora de crear el post se tienen en cuenta algunos factores como por ejemplo si el usuario ha seleccionado una foto previamente o si el paint está activado.

```
const handleUpload = () => {
    //create a post
    var usernameFilter = user.email.split('@', 1);
    //check if user select an image
    if(image) {
        var imageName = makeid(10);
        const uploadTask = storage.ref(`images/${imageName}.jpg`)
            .put(image);

        uploadTask.on("state_changed", (snapshot) => {
            const progress = Math.round((snapshot.bytesTransferred/snapshot.totalBytes)*100);

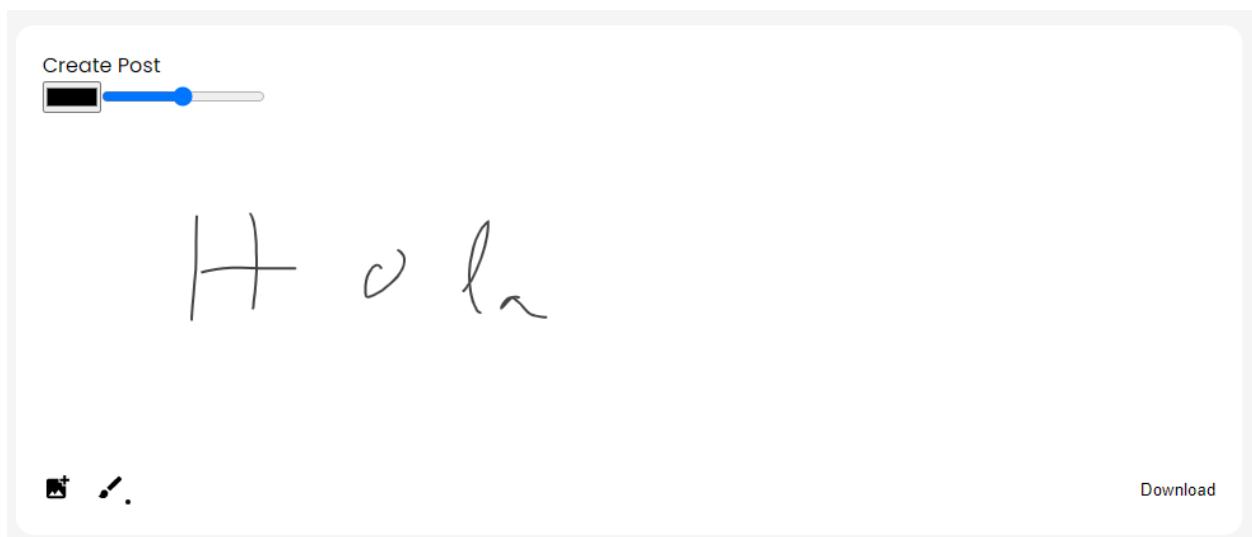
            setProgress(progress);
        }, (error) => {
            console.log(error);
        }, () => {
            storage.ref("images").child(`.${imageName}.jpg`)
                .getDownloadURL()
                .then((imageUrl) => {
                    db.collection("posts").add({
                        timestamp: firebase.firestore.FieldValue.serverTimestamp(),
                        date: ts.toDateString(),
                        content : caption,
                        photoUrl: imageUrl,
                        username: usernameFilter[0],
                        profileUrl: user.photoURL,
                        userId : user.uid,
                    })
                })
            playUpload();
            setImage();
            setCaption("");
            setProgress(0);
            document.getElementById("image-preview").style.display = "none"
        });
    }
};
```

Cuando se crea un post se genera el username del usuario mediante su gmail, si por otro lado también se va a subir una imagen se le genera una ID automática para poder almacenarla en el storage de firebase.

En el momento en el que el post va a almacenarse en la base de datos se guardan:

- La fecha en la cual se creo el post
- El timestamp
- El texto
- La URL de la imagen almacenada en el storage
- El username
- La foto del usuario
- La userID

Por otro lado si el usuario activa la funcionalidad de paint entonces el componente se crear un post cambia para permitiendo al usuario dibujar en un canvas y luego dandole la opcion de descargar lo que ha dibujado en formato png para subirlo posteriormente si desea a la plataforma

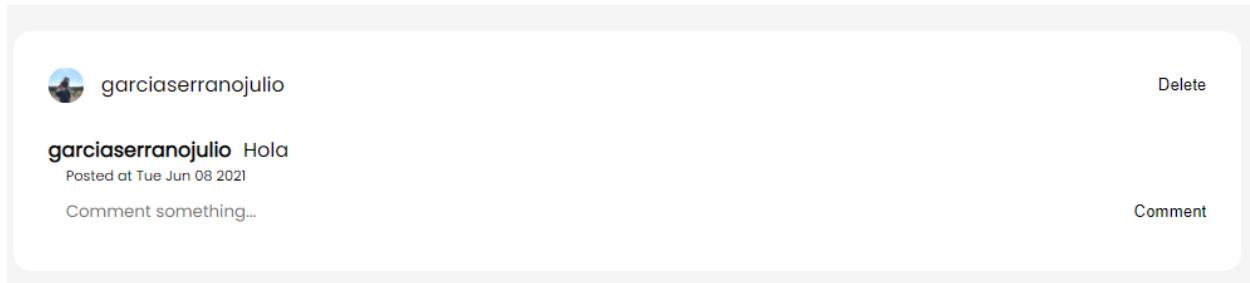


```
//check if user are using the paint function
}else if(paint){
  const data = myCanvas.current.getSaveData();
  console.log(data);
  const imageN = myCanvas.current.canvasContainer.childNodes[1].toDataURL("image/jpeg" , 1.0);

  var imageName = makeid(10);
  const link = document.createElement("a");
  link.href = imageN;
  link.download = `${imageName}.png`;
  link.click();
}
```

```
const paintChange = () => {
  if(paint){
    setPaint(false)
    setButtonCaption('Upload')
  }else{
    setPaint(true)
    setButtonCaption('Download')
  }
}
```

El resultado después de crear un post debería de ser algo como esto:



O en el caso de que sea un post con imagen algo asi:

 mohamed.chahdi



mohamed.chahdi Quiero un perro como este :(

Posted at Tue Jun 08 2021

javier.lr96 Yo te quiero a tí

Comment something...

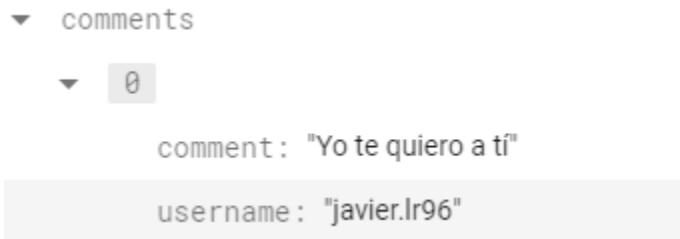
Comment

- **Create Comment** : Otro aspecto de los post es que pueden ser comentados por otros usuarios como se muestra en la foto anterior, para ello se emplea la siguiente función:

```
const addComment = () =>{
    //add comment to the post document
    if(comment != ""){
        commentArray.push({
            comment : comment,
            username: user.email.replace("@gmail.com", "").replace("@iescampanillas.com" , ""),
        });

        db.collection("posts")
        .doc(id)
        .update([
            comments: commentArray,
        ]).then(function(){
            setComment("");
            console.log('comment added');
        }).catch(function (error) {
            console.log(`Error ${error}`);
        });
    }
}
```

Los comentarios son almacenados en un campo del documento el cual es tipo array map , esto gracias a que la base de datos es NoSQL, este campo array map contiene en su interior dos atributos tipo string en los que se almacena el texto del comentario y el username de quien lo escribio



Esto lo hice así para no depender de las subcolecciones las cuales no borran bien sus datos cuando la colección principal es eliminada con esto me ahorraba ese problema.

- Create Room :

Al igual que se pueden crear post e interactuar mediante ellos SpeedPost también permite que sus usuarios creen rooms en las cuales puedan debatir y conversar sobre temas en concreto.

```
const handleUpload = () => [
    //create a new room
    var usernameFilter = user.email.split('@', 1);
    if(caption != ""){
        db.collection("rooms").add({
            timestamp: firebase.firestore.FieldValue.serverTimestamp(),
            date: ts.toDateString(),
            title : caption,
            username: usernameFilter[0],
            profileUrl: user.photoURL,
            userId : user.uid,
        })
        playUpload();
        setCaption("");
        setProgress(0);
    }
]
```

El funcionamiento es simple en la room se almacena la información referente al usuario que la creó, la fecha en la que fue creada y el tema sobre el que trata.

La función real de la room es almacenar en ella los mensajes de los usuarios.

```
<div className="room_body">
{
  messages.map(({id , message})=>{
    return(
      <Message
        id={id}
        roomId={roomId}
        content={message.content}
        username={message.username}
        photoUrl={message.photoUrl}
        userId={message.userId}
      />
    )
  })
}

</div>
```

También en el interior de la room encontramos el componente de create message.

- **Create Message:** La funcionalidad de create message es simple únicamente almacena en una subcolección de messages de la room correspondiente el contenido , al igual que con los post estos pueden contener imágenes

```

if(image) {
  var imageName = makeid(10);
  const uploadTask = storage.ref(`images/${imageName}.jpg`)
    .put(image);

  uploadTask.on("state_changed" , (snapshot) => {

    const progress = Math.round((snapshot.bytesTransferred/snapshot.totalBytes)*100);

    setProgress(progress);
  }, (error) => {
    console.log(error);
  }, () => {
    storage.ref("images").child(` ${imageName}.jpg`)
      .getDownloadURL()
      .then((imageUrl) => {
        db.collection('rooms').doc(id).collection('messages').add({
          timestamp: firebase.firestore.FieldValue.serverTimestamp(),
          content : caption,
          photoUrl: imageUrl,
          username: usernameFilter[0],
          profileUrl: user.photoURL,
          userId : user.uid,
        })
      })
    playUpload();
    setImage();
    setCaption("");
    setProgress(0);
    document.getElementById("image-preview").style.display = "none"
  });
}

```

- Feed : Esta es la parte encargada de mostrar los post en la feed de la aplicación.

```
const [posts, setPosts] = useState([]);  
//get all data from the database  
useEffect(() => {  
    db.collection('posts').orderBy('timestamp', 'asc').onSnapshot((snapshot) =>  
    {  
        setPosts(snapshot.docs.map((doc) =>({id: doc.id, post: doc.data()})));  
    })  
, [])  
const postR = posts.slice(0).reverse();  
return (  
    <div className="feed">  
        {postR.map(({id, post})=> {  
            return (  
                <Post  
                    key={id}  
                    id={id}  
                    profileUrl={post.profileUrl}  
                    username={post.username}  
                    photoUrl={post.photoUrl}  
                    content={post.content}  
                    userId={post.userId}  
                    likes={post.likes}  
                    comments={post.comments}  
                    date={post.date}  
                />  
            )  
        })  
    </div>  
)
```

Los parámetros que son enviados como Props a través del componente de Post son recogidos en este componente.

```
export default function Post({profileUrl , username , id , photoUrl ,
content , comments, userId, likes , Nlike , date})
```

Y mostrados de la siguiente forma

```
return (
  <div className="post">
    <div className="post_header">
      <div className="post_headerLeft">
        <img className="post_userImg" src={profileUrl} />
        <p style={{marginLeft: '1.5vh'}}>{username}</p>
      </div>
      {userId === user.uid && <button className="post_deleteBtn" onClick={handleDelete}>Delete</button>}
    </div>
    <div className="post_center">
      <img className="post_postPhoto" src={photoUrl}/>
    </div>
    <div className="post_footer">
      <p>
        <span style={{marginRight: '1vh'}}><b>{username}</b></span>
        {content}
      </p>
    </div>
    <div className="post_date">
      <p>Posted at {date}</p>
    </div>
    {comments ? comments.map((comments) => <Comment username={comments.username} content={comments.comment}/>) : <></>}
    { user ? <CommentInput comments={comments} id={id} /> : <p></p> }
  </div>
)
```

Pasa lo mismo para la feed de las rooms

```
useEffect(() => {
  db.collection('rooms').orderBy('timestamp', 'asc').onSnapshot((snapshot) =>
  {
    setRooms(snapshot.docs.map((doc) =>({id: doc.id, room: doc.data()})));
  })
}, [])

const roomR = rooms.slice(0).reverse();

return (
  <div className="feed">
    {roomR.map(({id, room})=> {
      return (
        <Room
          key={id}
          id={id}
          profileUrl={room.profileUrl}
          username={room.username}
          title={room.title}
          userId={room.userId}
          Roomtimestamp={room.date}
        />
      )
    })}
  </div>
)
```

También en las rooms está la peculiaridad de que actual a su vez como feed de las subcolección de messages.

```
useEffect(() => {
  db.collection('rooms').doc(id).collection('messages').orderBy('timestamp', 'asc').onSnapshot((snapshot) =>
  {
    setMessages(snapshot.docs.map((doc) =>({id: doc.id, message: doc.data()})));
  })
}, [])
```

```
<div className="room_body">
{
  messages.map(({id, message})=>{
    return(
      <Message
        id={id}
        roomId={roomId}
        content={message.content}
        username={message.username}
        photoUrl={message.photoUrl}
        userId={message.userId}
      />
    )
  })
}

</div>
```

- **Función Delete:** La función delete se encarga de eliminar el contenido relacionado a un usuario en concreto en función de su id.

```
const handleDelete = () =>{
  if(userId == user.uid){
    if(photoUrl){
      let pictureRef = storage.refFromURL(photoUrl);
      pictureRef.delete()
        .then(() => {
          console.log('the image was remove from storage')
        });
    }

    db.collection("rooms").doc(id).delete().then(() => {
      console.log("Document successfully deleted!");
    }).catch((error) => {
      console.error("Error removing document: ", error);
    });
    playDelete();
  }
}
```

La función verifica que la id del usuario y el campo userId de la publicación son los mismos una vez verificado pasan a ver si tiene alguna foto ligada al post para eliminarla del storage y luego elimina el resto del contenido.

- Hosting

Abrimos la consola como administrador y nos dirigimos a la carpeta del proyecto.

Una vez allí introducimos el comando

```
firebase init
```

Una vez introducido nos preguntará si queremos inicializar un proyecto de firebase y le decimos que sí.

Despues nos aparecerá una lista de opciones como esta

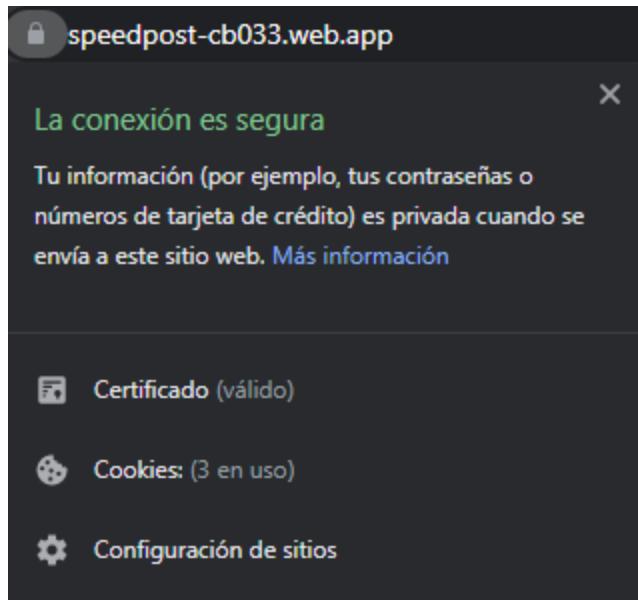
```
Narayans-MacBook-Pro:reactfirebase kaloraat$ firebase init
FIREBASE
You're about to initialize a Firebase project in this directory:
/Users/kaloraat.firebaseio/reactfirebase

? Which Firebase CLI features do you want to setup for this folder? Press Space to select features, then Enter to confirm your choices. (Press <space> to select)
❯○ Database: Deploy Firebase Realtime Database Rules
○ Firestore: Deploy rules and create indexes for Firestore
○ Functions: Configure and deploy Cloud Functions
○ Hosting: Configure and deploy Firebase Hosting sites
○ Storage: Deploy Cloud Storage security rules
```

Seleccionamos la opción de hosting y damos enter.

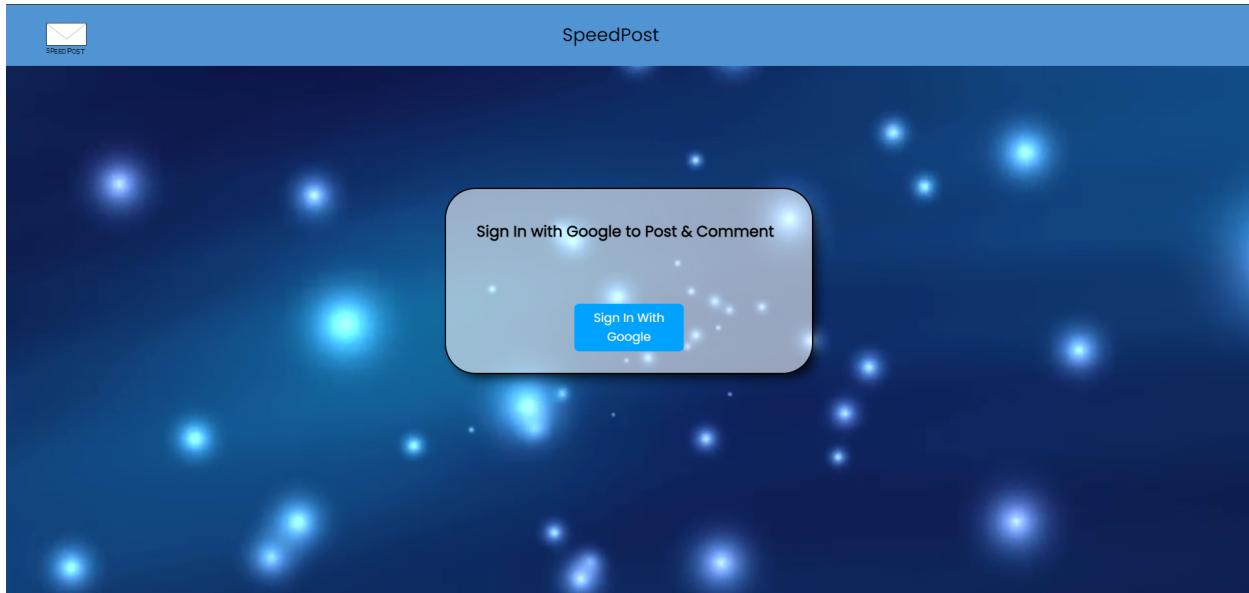
Nos preguntara si queremos emplear nuestro directorio public , en mi caso yo puse que utilizare el folder de build, despues de esto ejecuto el comando npm run build y firebase deploy, esto me da como resultado el enlace a mi aplicación.

Una vez en la web podemos ver que se ha desplegado sin problemas y que contamos con la certificación SSL que nos proporciona firebase hosting

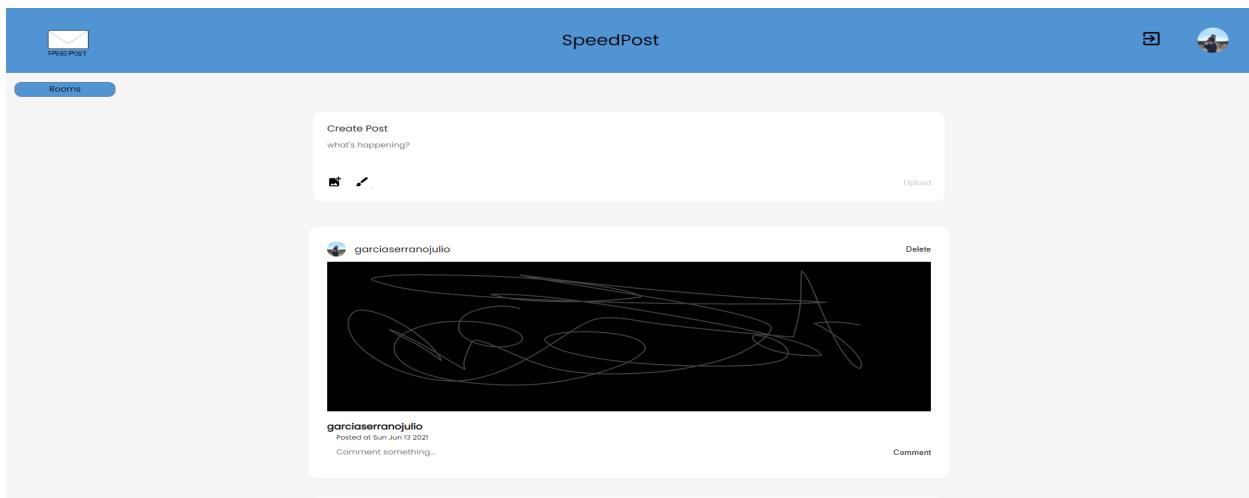


Guia de Uso

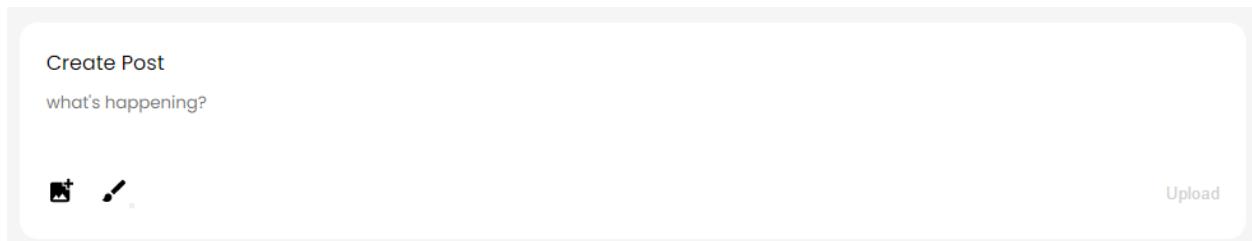
Para el logear solo tienes que pulsar en el botón de Sign in with Google y seleccionar la cuenta de gmail con la cual quieras acceder a la aplicación.



Una vez dentro vemos distintas opciones como el icono de logout , el switch para alternar entre las rooms y los post y el input de crear post.



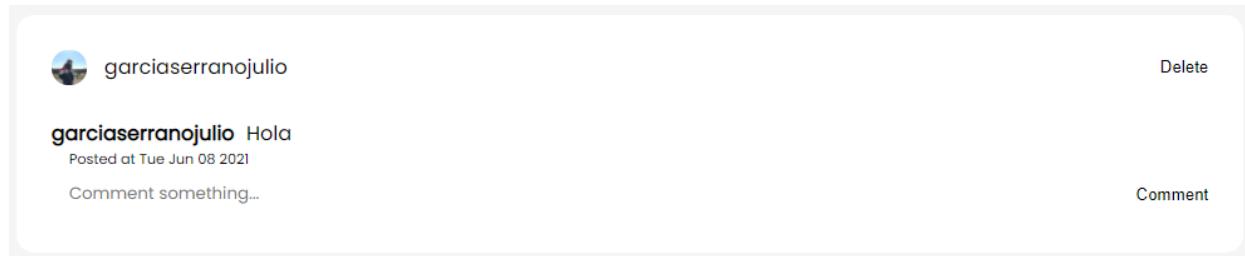
Centrándonos en el input de crear post podemos ver dos iconos principales uno de una imagen , otro de un pincel y un texto apagado que dice “Upload”.



La función de estos es la siguiente.

- La imagen : Te permite añadir imágenes al post.
- El pincel : Te permite dibujar en un canvas y descargarte lo que dibujes
- Upload : Este permite subir el post a la plataforma , solo se enciende si hay algo escrito o un foto seleccionada.

Una vez se ha creado el post vemos lo siguiente:





mohamed.chahdi

**mohamed.chahdi** Quiero un perro como este :(

Posted at Tue Jun 08 2021

javier.lr96 Yo te quiero a tí

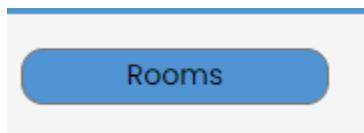
Comment something...

Comment

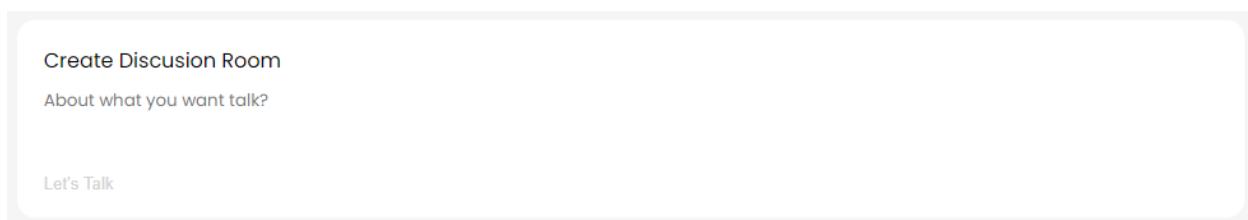
Como vemos en la parte superior del post se encuentra la foto del usuario y su username , la foto en el centro y en la parte inferior el username y el texto que el usuario puso en el post, debajo de esto esta la fecha en la cual fue publicada la foto.

En la parte inferior se ve el comentario de otro usuario , para comentar solo se tiene que escribir algo donde pone “comment something” y pulsar el texto de Comment.

Si pulsas en el botón azul de Rooms este cambiara el feed de Post por el de Rooms y asu vez el valor de este botón pasará a ser Post



En el apartado de Rooms al igual que en del Post podemos crear Rooms mediante un input de create Room



Simpsons

Created at Tue Jun 08 2021

juliusgarciaserrano

Hola que opinan sobre los simpsons?

garciaseerranojulio Delete



what's happening?

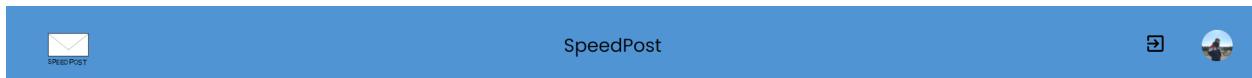


Upload

Una vez creamos una room podremos ver que en la parte superior tendremos el título de la room y su fecha de creación y dentro de ella se mostrarán los mensajes de los usuarios.

La creación de mensajes para las rooms funciona de manera similar a la de los post.

En el navbar podremos apreciar un icono al lado izquierdo de la foto del usuario



Este es el botón de logout el cual permite cerrar la sesión y redirigir al usuario al login