

AMPLIACIÓN DEL RETO FERIA VALENCIA.



DISEÑO Y ENVÍO DE UNA CAMPAÑA DE MARKETING DIGITAL USANDO MJML RETOS.



ANÁLISIS Y PLANIFICACIÓN

En este caso practico lo que se pide es crear un correo para un **boletín informativo** para **Salón del Cómic de València**, destacando noticias y promociones especiales.

Para la **paleta de colores** como debíamos de usar una imagen que nos ha proporcionado la **feria de muestras** he usado los mismos colores tanto como de la imagen para darle el **mismo estilo**.

En este caso he **resaltado** en amarillo **palabras clave** para que destaquen sobre las otras ademas les he **subido el tamaño** de la fuente para que se vean mas aun, resaltando que con la compra de **5 comics** te damos **uno gratis** o que **yendo** con **5 amigos** o mas puede entrar **otro mas de manera gratuita** asi incentivamos que venga mas gente y acaben **comprando mas cosas**.

Debo de analizar los **objetivos** de este **correo**, el cual ira **desde gente joven** a la que les puede gustar tanto como los **comics, anime, series** animadas, **películas** de superhéroes hasta un publico mas **adulto** la cual suelen ser a los que les gustan mas los **comics** en especifico , **edades** desde unos **15 hasta 40 años aproximadamente**.

Este **MJML** gasta unos colores mas vistosos que los otros para llegar a **captar la atención** del publico **mas joven** resaltando en colores vistosos y con **letras mas grandes palabras clave** ya que el publico mas joven **no suele leer mucho**.

DAFO

Debilidad: Un diseño muy vistoso y con letras más grandes **pensado para jóvenes** podría ser percibido como **poco serio o incluso infantil** por el segmento más adulto (30-40 años), que también es un objetivo clave.

Amenaza: Los usuarios, especialmente los más **jóvenes**, están expuestos a una gran **cantidad de estímulos** y comunicaciones, lo que **dificulta destacar en su bandeja** de entrada.

Fortaleza: Las **ofertas** específicas (5+1 en cómics, 5+1 en entradas) **son incentivos claros** y potentes para **fomentar** tanto la **asistencia** como el **consumo** dentro del evento.

Oportunidad: El interés por los **cómics**, el **anime**, los **videojuegos** y las series **está en auge**, lo que asegura una base de público potencial muy amplia y receptiva.

El objetivo de esta campaña es informar sobre las principales **noticias** y las principales **promociones** del salo del comic de valencia para que la gente siga con las ganas de ir y no

se pierda el hype, además las promociones que he puesto **incentivan** a que mas **gente venga** y le digan a sus amigos que **compren entradas**.

DISEÑO

La **planificación** del **diseño** de mi correo ha sido **principalmente** haciendo una **plantilla** en **figma** en la cual puse la estructura que iba a tener, añadiendo a el principio para captar la atención una imagen del salon del comic con la fecha cuando se realizara y debajo los principales invitados y mas abajo las principales noticias y promociones.

He usado **colores** mas **llamativos** y **textos** mas **grandes** en las **palabras clave** para llamar mas la atención de le **gente joven**.

Al principio del correo tenemos los principales invitados, a continuacion las principales noticias y por ultimo tenemos la seccion de promociones especiales

Aqui podrás ver el **modelo** en **figma** que hize para poder ver como cree la **plantilla** para poder crear el correo.

<https://www.figma.com/design/s79SJ6M9o9Q49anigrFYAy/TRABAJO-MJML-COMIC?t=ALRogvphHY8RwG25-0>

Para que el **MJML** sea **responsive** se piden **3 media queries** aqui pondre unas imágenes usando 3 distintas.



Aqui se veria con las **dimensiones mas grandes posibles**.



CLÀSSICS DEL CÒMIC

Esta exposición única reúne una colección de originales de los grandes autores que sentaron las bases del cómic moderno.

Desde las aventuras de Foster y Raymond hasta el humor de Schulz y Herriman, los visitantes podrán observar el trazo real sobre el papel, acercándose al trabajo artesanal que marcó generaciones.

Además de celebrar la importancia cultural de estas obras, la muestra destaca cómo estos clásicos siguen inspirando a creadores y emocionando al público contemporáneo.

Una oportunidad irrepetible para disfrutar del patrimonio histórico y artístico de este medio.

Exposición producida por el Área de Cultura de la Diputación de Valencia

CÓMIC PALESTINO

La imagen, el concepto de Palestina está cincelada en los imaginarios del mundo a base de bombas, sangre y destrucción.

Hay que tener mucha vida para haber visto y escuchado otra cosa diferente sobre Palestina.

En ese discurso dominante que se prolonga ya más de 75 años las voces palestinas han sido durante mucho tiempo marginadas, fragmentadas y descartadas.

A través de estas obras gráficas, los y las artistas palestinos rechazan el silencio impuesto, utilizando sus trazos para alterar las representaciones coloniales.

El intelectual palestino Edward Said sostenía que negarle la voz a alguien es negarle la existencia y desde esa perspectiva estas narraciones otigen ser leídas no como recuerdos pasivos, sino como intervenciones activas y dinámicas contra la amnesia histórica.

PROMOCIONES ESPECIALES




Presenta tu carnet de **socio club Fnac** socio club Fnac en las taquillas del salón **para acreditar tu descuento.**



Este sería un **tamaño mediano** y se vería de esta manera.


He colaborado como humorista gráfico en varias revistas y periódicos y he ganado varios premios en distintos concursos de cómic.



Miguel Gómez

Miguel Gómez Andrea (Madrid, 22 de febrero de 1960), más conocido como Gol, es un historietista, dramaturgo y actor español.

PRINCIPALES NOTICIAS



CLÀSSICS DEL CÒMIC

Esta exposición única reúne una colección de originales de los grandes autores que sentaron las bases del cómic moderno.

Desde las aventuras de Foster y Raymond hasta el humor de Schulz y Herriman, los visitantes podrán observar el trazo real sobre el papel, acercándose al

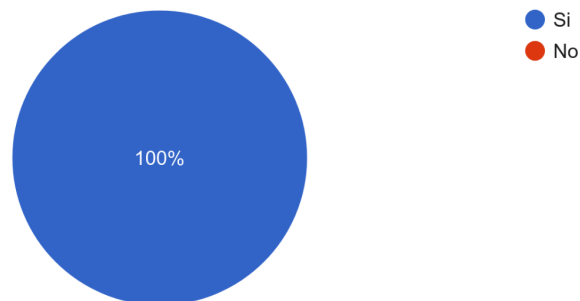
Por ultimo como se vería en un **formato** mas pequeño como en un **teléfono**.

TEST CON USUARIOS

<https://docs.google.com/forms/d/e/1FAIpQLSekEbgP4yfK4jTBPQyn8DfgggnpknCkClcSXJIRCHcmIaDcug/viewform?usp=dialog>

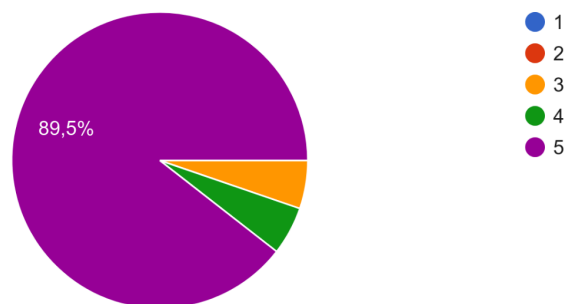
1 - ¿Considera que el trabajo cumplió con los objetivos planteados?

19 respuestas



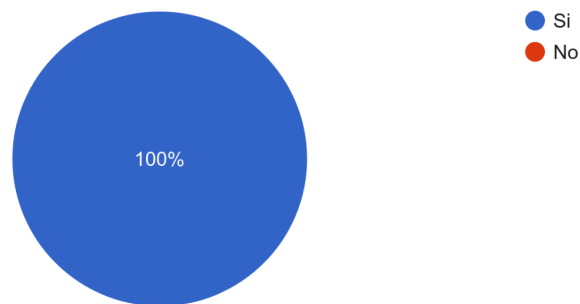
2 - En una escala de 1 a 5, ¿cómo calificaría el trabajo?

19 respuestas



3 - ¿Crees que la información es relevante y útil para los usuarios o interesados?

19 respuestas



5 - ¿Qué sugerencias o recomendaciones tendrías para mejorar el proyecto?

7 respuestas

Creo que es claro y conciso

Nada, esta muy bien

Ninguna esta genial

Creo que el boletín debería haber tenido una introducción o una portada

Esta perfecto

No tengo ninguna

Nada está muy bonito

6 - ¿Crees que falta alguna información relevante en el proyecto?

9 respuestas

No

No, esta muy completo

Me encanta así

No estoy bien enterado del tema así que entiendo que está toda la información que se pueda recibir de ese evento

No para mi esta toda la información necesaria

DESARROLLO Y ENVÍO DE CORREOS

Para empezar lo que hare es crear una maquina virtual con AWS, la crearemos buscando EC2 y creando una instancia. Seleccionamos ubuntu y le damos a los ajustes gratuitos para estudiantes, en el momento de seleccionar par de claves le damos a crear claves y le ponemos un nombre y en formato .pem una vez lo hagamos se descargaran y deberemos de guardarla ya que es muy importante, una vez echo crearemos la maquina virtual.

Para conectar mi ordenador a mi ec2 con ssh, para lo cual necesito el archivo (**jaade.pem**) y la ip publica de mi EC2 (**100.24.61.249**) y el nombre de usuario (**ubuntu**).

Un archivo **.pem** es un formato de archivo muy común que se usa para almacenar claves criptográficas y certificados digitales.

Voy donde he creado mi .pem este caso en una carpeta llamada pem :

```
cd /home/dam/Escritorio/pem
```

Y ejecutas el siguiente comando para darle los permisos necesarios :

```
chmod 400 jaade.pem
```

Ahora lo que debes hacer es conectarte a la instancia EC2 :

```
ssh -i /home/dam/Escritorio/pem/jaade.pem ubuntu@100.24.61.249
```

De esta manera ya esta conectado nuestro ordenador con la instancia EC2.

A continuación voy a crear un **Node.js** como un entorno donde puedes ejecutar código JavaScript fuera de un navegador web.

Creo una carpeta yo la llamo **enviador-emails-mjml** y dentro de ella dos archivos llamados:

package.json : Guarda información sobre tu aplicación y las librerías de Node.js que necesito.

index.js : Este será el archivo principal donde estará todo el código JavaScript para leer datos, procesar MJML y enviar correos.

En el archivo **package.json** estara este codigo:

```
{
  "name": "enviador-emails-mjml",
  "version": "1.0.0",
  "description": "Script para enviar correos masivos con MJML en Node.js",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "keywords": [],
  "author": "Tu Nombre",
  "license": "ISC",
  "dependencies": {
    "mjml": "^4.0.0",
    "nodemailer": "^6.0.0",
    "googleapis": "^128.0.0"
  }
}
```

Para instalar las librerías en la carpeta **enviador-emails-mjml** ponemos el siguiente comando :

npm install

Este comando lo que hace es leer el archivo package.json y descargara las librerías y las guardara en una carpeta llamada node_modules

En el archivo **index.js** estara este codigo:

```
// --- Importar librerías necesarias ---
const mjmltohtml = require('mjml'); // Para convertir MJML a HTML
const nodemailer = require('nodemailer'); // Para enviar correos
const fs = require('fs'); // Para leer archivos del sistema (plantillas, datos)
const path = require('path'); // Para manejar rutas de archivos de forma segura

// --- 1. Configuración de Variables de Entorno (MUY IMPORTANTE!) ---
// Estas variables NO deben ir directamente aquí en el código por seguridad.
// Debes configurarlas en el entorno de tu servidor EC2 antes de ejecutar el script.
// Ejemplo: export EMAIL_APP_PASSWORD=... contraseña de 16 dígitos
const EMAIL_APP_PASSWORD = process.env.EMAIL_APP_PASSWORD; // La contraseña de 16 dígitos que generaste en Google
const SENDER_EMAIL_GMAIL = process.env.SENDER_EMAIL_GMAIL; // El correo de Gmail que usas como remitente

// Asegúrate de que las variables de entorno están cargadas
if (EMAIL_APP_PASSWORD || SENDER_EMAIL_GMAIL) {
  console.error('¡Error! Faltan variables de entorno para la configuración de Gmail con contraseña de aplicación.');
  console.error('Asegúrate de que EMAIL_APP_PASSWORD y SENDER_EMAIL_GMAIL estén configuradas antes de ejecutar el script.');
  console.error('Ejemplo: export EMAIL_APP_PASSWORD="tu_contraseña_de_16_dígitos" (sin espacios al pasado en la terminal, pero copiando todo); export SENDER_EMAIL_GMAIL="tu_email@gmail.com"');
  process.exit(1); // Sale del programa si no hay credenciales
}

// --- 2. Función para leer la plantilla MJML desde un archivo ---
// Asume que la plantilla MJML está en la misma carpeta que index.js
function leerPlantilla(pathArchivoPlantilla) {
  const rutaPlantilla = path.join(__dirname, nombreArchivo(path));
  try {
    console.log('Intentando leer plantilla MJML desde: ' + rutaPlantilla);
    return fs.readFileSync(rutaPlantilla, 'utf8');
  } catch (error) {
    console.error('¡Error! No se pudo leer la plantilla MJML. ' + nombreArchivo(path));
    console.error('Detalles: ' + error.message);
    process.exit(1); // Sale del programa si no puede leer la plantilla
  }
}

// --- 3. Función para renderizar MJML a HTML con datos dinámicos ---
function renderizarPlantilla(datos) {
  const plantillaPersonalizada = mjmltohtml(
    // Ejemplifica los placeholders como {nombreCliente} con los datos reales
    // (controla si datos) {
    // Usa una expresión regular para reemplazar todas las ocurrencias de {clave}
    plantillaPersonalizada = plantillaPersonalizada.replace(new RegExp('{' + clave + '}', 'g'), datos[clave]);
    // }

    const { text, errors } = mjmltohtml(plantillaPersonalizada);
    if (errors && errors.length > 0) {
      console.error('¡OCCURRENCIA! Errores al renderizar MJML:', errors);
    }
    return text;
  }
}

// --- 4. Función para leer datos de clientes desde un archivo JSON ---
// Asume que el archivo clientes.json está en la misma carpeta que index.js
function leerDatosClientes(nombreArchivoDatos) {
  const rutaDatos = path.join(__dirname, nombreArchivo(rutaDatos));
  try {
    console.log('Intentando leer datos de clientes desde: ' + rutaDatos);
  }
}
```



```
const dataCruada = fs.readFileSync(datos, 'utf8')
return JSON.parse(dataCruada); // Intenta parsear el contenido como JSON
} catch (error) {
  console.error(ERROR: No se pudieron leer o parsear los datos de clientes desde "${nombreArchivoDatos}");
  console.error(Asegura de que el archivo exista, se llame "${nombreArchivoDatos}" y sea un JSON válido.");
  console.error(Detalles: ${error.message});
  process.exit(1); // Sale del programa si los datos no son válidos
}

// --- 5. Función para enviar un correo con Gmail usando Contraseña de Aplicación ---
async function enviarCorreoConGmail(datosCliente, asunto, htmlContent) {
  try {
    const transporter = nodemailer.createTransport({
      host: 'smtp.gmail.com',
      port: 587, // Puerto estándar para SMTP con STARTTLS
      secure: false, // Usar STARTLS, no SSL/TLS directo
      rejectUnauthorized: true, // Forzar STARTLS
      auth: {
        user: SENDER_EMAIL_GMAIL,
        pass: GMAIL_APP_PASSWORD,
      },
    });
    // Verificar la configuración. Mejor puede ser útil para probar si tienes problemas de certificados.
    // El "true" se recomienda para producción por motivos de seguridad.
    // En producción, es mejor que el servidor SMTP tenga un certificado válido y dejar esto en true o no definido.
    transporter.verify().then(() => {
      // Verificar la configuración de la contraseña de aplicación.
    }, () => {
      console.error('Error al verificar la configuración de la contraseña de aplicación. Verifica que sea correcta y que el servidor SMTP tenga un certificado válido.');
    });
  } catch (error) {
    console.error('Error al configurar el transporte de correo: ', error);
    return null;
  }

  const mailOptions = {
    from: `Tu Empresa <${SENDER_EMAIL_GMAIL}>`, // Nombre del remitente y tu correo
    to: datosCliente.email,
    subject: asunto,
    html: htmlContent,
  };

  const info = await transporter.sendMail(mailOptions);
  console.log('Correo enviado exitosamente a ${datosCliente.email} (ID del mensaje: ${info.messageId}).');
  return info;
} catch (error) {
  console.error(ERROR: No se pudo enviar el correo a ${datosCliente.email}. Asegura de que la Contraseña de Aplicación y el email del remitente sean correctos.);
  console.error(Detalles del error: ${error});
  // Aquí puedes añadir lógica para registrar o registrar el fallo
}

// --- 6. Lógica Principal: Orquestación del Proceso de Envío ---
async function main() {
  console.log('--- Iniciando proceso de envío de correo ---');

  // 1. Cargar los datos de clientes desde el archivo JSON
  const nombreArchivoDatos = 'clientes.json'; // Nombre de los datos de clientes
  const nombrePlantillaMJML = 'plantilla-mjml'; // Nombre de la plantilla MJML
  const datosClientes = await leerDatosClientes(nombreArchivoDatos, nombrePlantillaMJML);

  // 2. Leer los datos de los clientes
  const clientes = datosClientes;

  // 3. Procesar los datos de los clientes
  if (clientes.length === 0) {
    console.warn(ADVERTENCIA: No se encontraron clientes en el archivo de datos. El proceso de envío ha finalizado sin enviar correos.);
    return;
  }

  // 4. Enviar los correos
  console.log('Se encontraron ${clientes.length} clientes. Procesando envíos...');

  // 5. Enviar cada correo
  for (const cliente of clientes) {
    // 5.1. Validación básica del email del cliente
    if (!validarEmail(cliente.email)) {
      console.warn(ADVERTENCIA: Cliente con dirección de correo inválida o faltante. Saltando envío para: ${JSON.stringify(cliente)});
      continue; // Pasa al siguiente cliente
    }

    // 5.2. Datos para personalizar la plantilla (adapta esto a los campos de tu JSON)
    // Por ejemplo, si tu JSON tiene 'nombre' y 'telefono', puedes usarlos así:
    const datosParaPlantilla = {
      nombreCliente: cliente.nombre, // Estimado Cliente, // Usa el nombre si existe, si no, un valor por defecto
      // Añade más placeholders que uses en tu MJML, por ejemplo:
      // numeroCliente: cliente.telefono, // nuestro código,
      // ofertaExclusiva: cliente.oferta, // nuestra oferta,
    };

    // 5.3. Asunto del correo (puedes personalizarlo también)
    const asuntoCorreo = `Hola, ${datosParaPlantilla.nombreCliente}! Tenemos noticias para ti.`;

    // 5.4. Renderizar la plantilla MJML a HTML, con los datos del cliente
    const htmlFinal = renderizarPlantillaMJML(datosParaPlantilla, asuntoCorreo);

    // 5.5. Enviar el correo
    await enviarCorreoConGmail(cliente.email, asuntoCorreo, htmlFinal);

    // 5.6. Opcional: Pequeña pausa entre envíos para evitar saturar al servidor de correo
    // Usa si tienes límites de tasa o quieres enviar un envío más lento.
    // await new Promise(resolve => setTimeout(resolve, 1000)); // Espera 1 segundo
  }

  console.log('--- Proceso de envío de correo finalizado ---');
}

// Ejecutar la función principal para iniciar el proceso
main();
```

lo que hace este código es:

- **Configuración Segura:** Primero, el script se asegura de tener tu contraseña de aplicación de Gmail y mi correo de remitente configurados como variables de entorno. Si no están, detiene el proceso.
- **Plantilla MJML:** Lee una plantilla de correo electrónico en formato MJML (un lenguaje para crear emails responsivos) desde un archivo.
- **Datos de Clientes:** Carga una lista de clientes y sus datos (como nombres y correos) desde un archivo JSON.
- **Personalización y Conversión:** Por cada cliente en la lista, toma la plantilla MJML, reemplaza los datos genéricos (como {{nombreCliente}}) con la información específica de ese cliente, y luego convierte el MJML a HTML (el formato que entienden los clientes de correo).

- **Envío por Gmail:** Finalmente, utiliza Nodemailer (una librería para enviar correos) para enviar el correo personalizado a cada cliente usando

En la carpeta **enviador-emails-mjml**, creare estos dos archivos:

plantilla-email.mjml (con el mjml)

clientes.json

```
[
  {
    "nombre": "irene",
    "email": "irenemaza@gmail.com"
  },
  {
    "nombre": "salva",
    "email": "leo30salva@gmail.com"
  },
  {
    "nombre": "javi",
    "email": "javierdescalsfernandez@gmail.com"
  }
]
```

A continuación lo que hay que hacer es poner el correo electrónico y el código de aplicación de la misma para que el programa pueda enviar correos desde este.

```
export GMAIL_APP_PASSWORD='CONTRASEÑA_DE_APLICACION'
```

```
export SENDER_EMAIL_GMAIL='CORREO_QUE_ENVIARA_MJ@gmail.com'
```

Por último usamos este comando y se enviarán los correos:

```
node index.js
```

INTEGRACIÓN CON REDES SOCIALES



CONTROL DE VERSIONES CON GITHUB

El control de versiones de este MJML se encuentra en este link:

<https://github.com/Liljaade/mjmlComic>

DATOS

Como se pedia la creacion de una base de datos instale mySQL

```
sudo apt update
```

sudo apt install mysql-server -y

Ejecuta el script de seguridad de MySQL

sudo mysql_secure_installation

Me conecto a mysql y creo la base de datos

sudo mysql -u root -p

CREATE DATABASE emails_db;

Creamos un usuario y le damos permisos

Creamos una tabla clientes

USE emails_db;
CREATE TABLE clientes (
 id INT AUTO_INCREMENT PRIMARY KEY,
 nombre VARCHAR(255) NOT NULL,
 email VARCHAR(255) UNIQUE NOT NULL
);

He inserto datos de prueba:

```
INSERT INTO clientes (nombre, email) VALUES ('Ana', 'ana.ejemplo@gmail.com');
INSERT INTO clientes (nombre, email) VALUES ('Carlos', 'carlos.ejemplo@gmail.com');
INSERT INTO clientes (nombre, email) VALUES ('Maria', 'maria.ejemplo@gmail.com');
INSERT INTO clientes (nombre, email) VALUES ('Javier',
'javierdescalsfernandez@gmail.com');
```

Salimos de la consola exit;

Modificamos el [index.js](#) para que funcione con la base de datos

Y configuramos las variables de entorno

```
export GMAIL_APP_PASSWORD='CONTRASEÑA_DE_16_DIGITOS_SIN_ESPACIOS'
export SENDER_EMAIL_GMAIL='tu_correo_de_envio@gmail.com'
export DB_HOST='localhost'
export DB_USER='app_user'
export DB_PASSWORD='tu_contraseña_segura'
export DB_NAME='emails_db'
```

Y por ultimo ejecutamos el script

node index.js

PERSONALIZACIÓN DEL CORREO MJML



- Por cada cliente en la lista, toma la plantilla MJML, reemplaza los datos genéricos (como {{nombreCliente}}) con la información específica de ese cliente, y luego convierte el MJML a HTML (el formato que entienden los clientes de correo).

DOCUMENTACIÓN FINAL

Gracias a este trabajo he aprendido a como realizar un **proyecto desde 0** empezando desde **análisis y bocetos** hasta hacer el **envío de correos** y ver las **opiniones** de varios **usuarios** y así poder **mejorar** en varios aspectos

De primeras tuve que ver **cual era el objetivo** de este correo para así poder ver como plantearlo.

He aprendido a analizar el público objetivo y adaptar la estrategia de diseño del email.

Planificar el diseño visual del email desde el boceto inicial hasta la creación de un modelo en Figma, prestando atención a la paleta de colores y la jerarquía visual.

Adaptabilidad del diseño (responsive) para diferentes dispositivos (grandes, medianos y móviles).

Realizando **test de usuarios** para ver en que puedo mejorar y realizar cambios para tener un mayor acercamiento al publico.

Desarrollar una infraestructura de envío de correos desde cero:

- La configuración de una **máquina virtual en AWS (EC2)** y la conexión segura a través de SSH.
- La implementación de un **entorno Node.js** con librerías clave como MJML (para convertir el diseño a HTML).
- La **personalización de correos** con datos dinámicos de los destinatarios.
- Uso de **variables de entorno** para manejar credenciales de forma segura.

Integrar una base de datos MySQL para gestionar la lista de clientes, demostrando habilidades en la creación de bases de datos, tablas, usuarios y la inserción de datos.

Implementar un **control de versiones** eficaz usando GitHub para gestionar el código del proyecto.