

AMPLIACIÓN DEL RETO FERIA VALENCIA.



REGISTRATE ANTES DEL
1 DE SEPTIEMBRE Y VEN
GRATIS CON UN ACOMPAÑANTE

GANÁ 2 ENTRADAS PARA
EL CIRCUITO DE CHESTE



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



2Ruedas
Salón de la Moto y la Bici de Valencia

15-17
Nov.'24
Colaborando con
GRAN PREMIO
MOTUL
DE LA COMUNITAT
VALENCIANA



ANÁLISIS Y PLANIFICACIÓN

En este caso practico lo que se pide es crear un correo para ganar **2 entradas de motoGP** en Cheste rellenando un **formulario** para poder participar antes del 1 de septiembre.

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 como de la misma pagina de para darle el mismo estilo.

He **resaltado en rojo** palabras clave para que destaque sobre las otras, resaltando que se pueden ganar 2 entradas solo rellenando un simple **formulario** y vivir la experiencia de manera **totalmente gratuita**.

Tambien añadí una sección de **galería** en la cual puedes ver varias fotos de la competición para que te den ganas de estar allí.

Tengo que analizar a que objetivos va a ir este correo, el cual ira desde **gente joven** a la que les puede gustar las motos hasta un **público mas adulto** la cual suele ser la que mas le gusta este tipo de deportes, edades desde unos **20 hasta 50 años aproximadamente**.

He creado el MJML yendo un poco mas hacia **algo sencillo** ya que a la gente algo mas mayor no suele estar tan familiarizada con este tipo de correos y pueden llegar abrumarles un poco si hay muchas cosas.

DAFO

Debilidad: El éxito inicial depende de que el correo supere los filtros de **spam** y sea abierto por los destinatarios.

Amenaza: Algunos **usuarios pueden ser reacios** a proporcionar sus datos personales en **formularios** online por motivos de **privacidad** o por temor a recibir **publicidad no deseada**.

Fortaleza: Ofrecer dos entradas para un **evento tan popular** como MotoGP en Cheste es un **gancho muy potente** y de alto valor percibido para el público objetivo.

Oportunidad: Existe una **base de aficionados** muy amplia y apasionada por MotoGP , lo que garantiza un interés **potencial elevado en la promoción**.

El **objetivo** de esta campaña es intentar conseguir la **mayor cantidad de registros** en el formulario para que vaya yendo de boca en boca de que se va a realizar el evento y asi mas gente se vaya enterando de esta y asi conseguir **mas ventas**.

DISEÑO

La **planificación** del diseño de mi correo ha sido principalmente a mano en una libreta en la cual puse la **estructura** que iba a tener, añadiendo a el principio para captar la atención una imagen de las motos de motoGP oscurecida con un texto de que se realiza un **sorteo de 2 entradas** y que pueden ser tuyas simplemente registrandote.

Continuando por un section con dos columnas en la cual en la mitad de la izquierda pone en colores chillones para llamar mas la atencion lo de las entradas gratuitas y que el formulario esta debajo. En la columna de la izquierda puedes ver el **circuito de Cheste** para que veas como es ya que se va a realizar ahí.

Debajo tenemos el **famoso formulario** el cual sale con un texto grande y debajo esta el **botón** que te lleva a la pagina en la cual se realizaría el formulario (en este caso lleva a la pagina de 2 ruedas de la feria de valencia ya que no hay formulario).

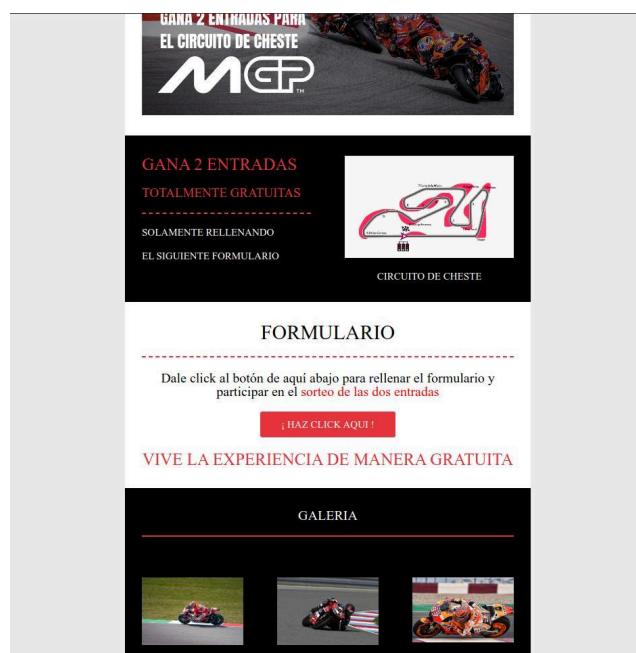
Después esta la **galería** en la cual puedes ver fotos de las imágenes de anteriores competiciones realizadas para que te den mas ganas de ir a el evento.

Tambien esta la **newsletter** en el cual tienes la opción de poner tu correo y recibir correos cada vez que tengamos novedades

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/l11G8aqzjHDuq9i7sr6ub5/TRABAJO-MJML-MOTOGP?node-id=0-1&p=f&t=WGbAhmBXvvQAIFaJ-0>

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



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



Este seria un tamaño **mediano** y se veria de esta manera.



FORMULARIO

Dale click al botón de aquí abajo para llenar el formulario y participar en el [sorteo de las dos entradas](#)

[¡HAZ CLICK AQUI!](#)

VIVE LA EXPERIENCIA DE MANERA GRATUITA

GALERIA



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

FORMULARIO

Dale click al botón de aquí abajo para llenar el formulario y participar en el [sorteo de las dos entradas](#)

[¡HAZ CLICK AQUI!](#)

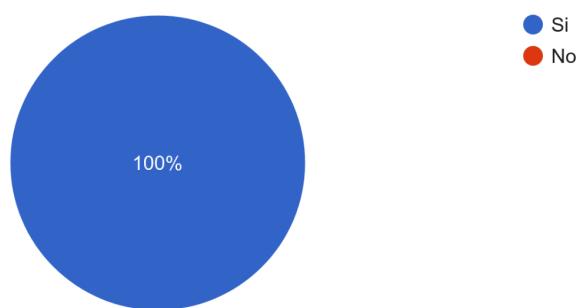
VIVE LA EXPERIENCIA DE MANERA GRATUITA

GALERIA

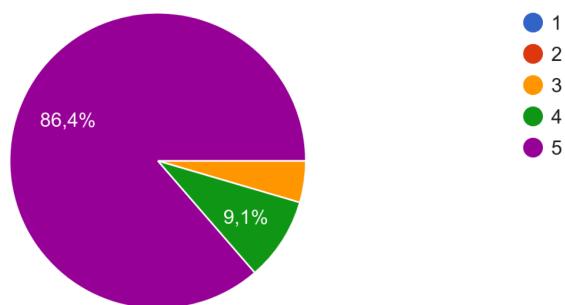
TEST CON USUARIOS

https://docs.google.com/forms/d/e/1FAIpQLSfF26ZHFT2i6ItqC_htWk3MgxZ4Hg64nIOZ6SLWT9UQTDMc5g/viewform?usp=dialog

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

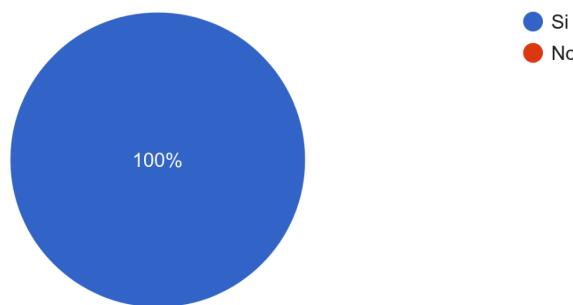


2 - En una escala de 1 a 5, ¿cómo calificaría el trabajo?
22 respuestas



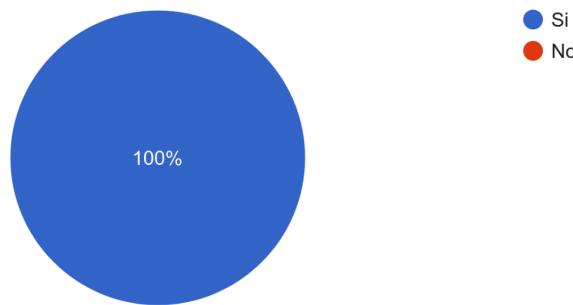
3 - ¿Crees que la información es relevante y útil para los usuarios o interesados?

22 respuestas



4 - ¿Crees que el proyecto ha sido diseñado de forma intuitiva y accesible?

22 respuestas



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

10 respuestas

Ninguna, ya que para mi es un trabajo muy bien hecho, se nota el esfuerzo

Más colores

Ninguna.No

Ninguna

No tengo ninguna

Esta completo

Ninguna, creo que cumple con las necesidades requeridas

Visualmente deja que desear, pero cumple con todo lo necesario para informar y enterarnos de la feria

Creo que esta bien así

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

12 respuestas

No

No, esta toda la informacion necesaria

Creo que se podría poner un poco más de información

No, ya que se entiende perfectamente

Que se me ocurra ninguna

No, está muy bien informado

DESARROLLO Y ENVÍO DE CORREOS

Para empezar lo que haremos es crear una máquina 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 máquina virtual.

Para conectar mi ordenador a mi ec2 con ssh, para lo cual necesito el archivo (**jaade.pem**) y la ip pública 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 está 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 llame **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** estará este código:

```
{
  "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 mjmlToString = require('mjml');
const fs = require('fs');
const path = require('path');
const os = require('os');
const readline = require('readline');
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});
const file = path.join(__dirname, 'index.html');
const file2 = path.join(__dirname, 'index2.html');

// Para convertir MJML a HTML
function convertMJMLtoHTML(file) {
  const MJML = fs.readFileSync(file);
  const MJMLString = MJML.toString();
  const MJMLObject = JSON.parse(MJMLString);
  const MJMLHTML = mjmlToString(MJMLObject);
  return MJMLHTML;
}

// Para leer archivos del sistema (plantilla, datos)
function readfile(file) {
  const data = fs.readFileSync(file);
  return data;
}

// Para manejar los archivos de forma segura
function safePath(path) {
  return path.replace(/[^a-zA-Z0-9_]/g, '');
}

// --- 1. Configuración de Variables de Entrada (Variables de Entorno) ---
// Asegurarse de que las variables de entorno estén configuradas
// en el archivo .env
// const GMAIL_APP_PASSWORD = process.env.GMAIL_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
// const RECIPIENT_EMAIL_GMAIL = process.env.RECIPIENT_EMAIL_GMAIL; // El correo al que deseas enviar el correo
// const FILENAME = process.env.FILENAME; // El nombre del archivo que deseas enviar

// Asegurarse de que las variables de entorno estén configuradas
// en el archivo .env
// const GMAIL_APP_PASSWORD = process.env.GMAIL_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
// const RECIPIENT_EMAIL_GMAIL = process.env.RECIPIENT_EMAIL_GMAIL; // El correo al que deseas enviar el correo
// const FILENAME = process.env.FILENAME; // El nombre del archivo que deseas enviar

// --- 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(file) {
  const data = fs.readFileSync(file);
  const MJML = data.toString();
  const MJMLObject = JSON.parse(MJML);
  const MJMLHTML = mjmlToString(MJMLObject);
  return MJMLHTML;
}

// --- 3. Función para renderizar MJML a HTML, con datos dinámicos ---
function renderizarPlantilla(mjmlString, datos) {
  let plantillaPersonalizada = mjmlString;
  if (datos) {
    plantillaPersonalizada = plantillaPersonalizada.replace(/\{cliente\}/g, datos);
  }
  for (const clave in datos) {
    if (datos[clave] === undefined) {
      continue;
    }
    plantillaPersonalizada = plantillaPersonalizada.replace(`{{${clave}}}`, datos[clave]);
  }
  const error = mjmlString !== plantillaPersonalizada;
  if (error) {
    console.warn(`ADVERTENCIA: Errores al renderizar MJML: ${error.message}`);
  }
  return plantillaPersonalizada;
}

// --- 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) {
  try {
    const data = fs.readFileSync(`./${nombreArchivoDatos}`);
    const MJML = data.toString();
    const datos = JSON.parse(MJML);
    return datos;
  } catch (error) {
    console.error(`ERROR: No se pudo leer o parsear los datos de clientes desde '${nombreArchivoDatos}'`);
    console.error(`Detalles: ${error.message}`);
  }
  return {};
}

// --- 5. Función para enviar un correo con Gmail usando Contraseña de Aplicación ---
async function enviarCorreoConGmail(datos, asunto, htmlContent) {
  try {
    const transporter = nodemailer.createTransport({
      host: 'smtp.gmail.com',
      port: 465,
      secure: true, // Use STARTTLS, no SSL/TLS directo
      requireTLS: true, // Forzar STARTTLS
      auth: {
        user: SENDER_EMAIL_GMAIL,
        pass: GMAIL_APP_PASSWORD
      }
    });
    if (requireTLS === 'false') {
      console.log(`NOTA: 'false' puede ser útil para probar si tienes problemas de certificados.`);
    }
    if (port === '465') {
      console.log(`NOTA: '465' se recomienda para producción por motivos de seguridad.`);
    }
    if (process.env.NODE_ENV === 'production') {
      console.log(`NOTA: Si usas el servidor SMTP tiene un certificado válido y dejar esto en true o no definido.`);
    }
    if (requireTLS === false) {
      console.log(`NOTA: 'false' es útil para probar si tu conexión es segura.`);
    }
    const mailOptions = {
      from: 'Tu Empresa <' + SENDER_EMAIL_GMAIL + '>', // Nombre del remitente y tu correo
      to: destino, // Correo electrónico del destinatario
      subject: asunto,
      html: htmlContent,
      attachments: []
    };
    const info = await transporter.sendMail(mailOptions);
    console.log(`Email enviado exitosamente a ${info.to[0]} (ID del mensaje: ${info.id}).`);
  } catch (error) {
    console.error(`Error: ${error.message}`);
    console.error(`Detalles del error: ${error}`);
  }
}

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

  // 1. Nombre de los archivos de entrada
  const NOMBRE_PLANTILLA_MJML = 'plantilla-email.mjml'; // Asegúrate de que este archivo exista
  const NOMBRE_CLIENTES = 'clientes.json'; // Asegúrate de que este archivo exista y esté bien formado

  // 2. Leer los datos de los clientes
  const clientes = leerDatosClientes(NOMBRE_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;
  }

  // 3. Leer la plantilla
  const plantilla = leerPlantilla(NOMBRE_PLANTILLA_MJML);

  // 4. Personalizar la plantilla
  const personalizada = plantilla.replace(/\{cliente\}/g, clientes);
  const JSONpersonalizada = JSON.stringify(personalizada);
  const MJMLpersonalizada = mjmlToString(JSONpersonalizada);

  // 5. Enviar el correo
  enviarCorreoConGmail(MJMLpersonalizada, 'Envío de correos', 'Tu correo ha sido enviado');
}

// Datos para personalizar la plantilla (adóptalo esto a los campos de tu JSON)
// Por ejemplo, si tu JSON tiene 'nombre' y 'apellido', puedes usarlos aquí.

```

```

const datosParaPlantilla = {
  nombreCliente: cliente.nombre || 'Estimado Cliente', // Usa el nombre si existe, si no, un valor por defecto
  // Aquí puedes añadir más variables que necesites para tu M&M, por ejemplo:
  // ciudadCliente: cliente.ciudad || 'nuestra ciudad',
  // ofertaExclusiva: cliente.oferta || 'nuestra oferta',
};

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

// Renderizar la plantilla MJML a HTML, con los datos del cliente
const renderizado = e rendervarPlantilla(myResource, datosParaPlantilla);

// Envío el correo
asuntoCorreoConGmail(cliente.email, asuntoCorreo, renderizado);

// Opcional: Puedes ponerle pausa entre envíos para evitar saturar el servidor de correo
// Util si tienes límites de tasa o quieres simular 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_APPLICACION'  
  
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/mjmlMotoGP>

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



¡Hola, Javi!

GANA 2 ENTRADAS

CIRCUITO DE CHESTE

- 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 asi poder **mejorar** en varios aspectos

De primeras tuve que ver **cual era el objetivo** de este correo para asi 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.