

Informe sobre el app escaner

Jhoan Bernal

21 de mayo de 2025

Método 1: Comandos individuales en PowerShell

```
# Crear estructura de directorios
mkdir qr-scan-app
cd qr-scan-app
mkdir assets, templates, internal, cmd
```

```
cd assets
mkdir css, js, images
cd ..
cd internal
mkdir handlers, models, qr, storage
cd ..
cd cmd
mkdir webserver
cd ..

# Crear archivos principales
New-Item -ItemType File templates/base.html
New-Item -ItemType File templates/login.html
New-Item -ItemType File templates/scanner.html
New-Item -ItemType File templates/files.html

New-Item -ItemType File internal/handlers/auth.go
New-Item -ItemType File internal/handlers/files.go
New-Item -ItemType File internal/handlers/scanner.go

New-Item -ItemType File internal/models/user.go
New-Item -ItemType File internal/models/file.fabrigo

New-Item -ItemType File internal/qr/generator.go
New-Item -ItemType File internal/qr/scanner.go

New-Item -ItemType File internal/storage/local.go
New-Item -ItemType File internal/storage/database.go

New-Item -ItemType File cmd/webserver/main.go
New-Item -ItemType File README.md

# Inicializar módulo Go
go mod init github.com/tuusuario/qr-scan-app
```

Método 2: Script PowerShell más eficiente

Guarda esto como create-project.ps1 y ejecútalo:

```
$projectRoot = "qr-scan-app"

# Crear estructura de directorios
$directories = @(
    "$projectRoot/assets/css",
    "$projectRoot/assets/js",
    "$projectRoot/assets/images",
    "$projectRoot/templates",
    "$projectRoot/internal/handlers",
    "$projectRoot/internal/models",
    "$projectRoot/internal/qr",
    "$projectRoot/internal/storage",
    "$projectRoot/cmd/webserver"
)

foreach ($dir in $directories) {
    New-Item -ItemType Directory -Force -Path $dir | Out-Null
}

# Crear archivos
$files = @(
    "$projectRoot/templates/base.html",
    "$projectRoot/templates/login.html",
    "$projectRoot/templates/scanner.html",
    "$projectRoot/templates/files.html",
    "$projectRoot/internal/handlers/auth.go",
    "$projectRoot/internal/handlers/files.go",
    "$projectRoot/internal/handlers/scanner.go",
    "$projectRoot/internal/models/user.go",
    "$projectRoot/internal/models/file.go",
    "$projectRoot/internal/qr/generator.go",
    "$projectRoot/internal/qr/scanner.go",

```

```

"$projectRoot/internal/storage/local.go",
"$projectRoot/internal/storage/database.go",
"$projectRoot/cmd/webserver/main.go",
"$projectRoot/README.md"
)

foreach ($file in $files) {
    New-Item -ItemType File -Force -Path $file | Out-Null
}

# Inicializar módulo Go
Set-Location $projectRoot
go mod init github.com/tuusuario/qr-scan-app
Set-Location ..

Write-Host "Estructura del proyecto creada exitosamente en
$projectRoot" -ForegroundColor Green

```

De acuerdo, vamos a empezar creando la interfaz de la página principal (index) usando el framework Gin. Esto implica dos cosas principales:

creando el index.html con su navbar

Crear el archivo HTML para la interfaz. Este archivo contendrá la estructura básica de tu página, incluyendo cualquier elemento visual y enlaces a archivos CSS y JavaScript. Lo guardaremos en el directorio templates/.

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Responsive Navbar</title>
    <link rel="stylesheet" href="/assets/css/style.css" />
  </head>
  <body>

```

```
<nav>
  <div class="logo">
    
    <h1>LOGO</h1>
  </div>
  <ul>
    <li>
      <a href="#">Home</a>
    </li>
    <li>
      <a href="#">Services</a>
    </li>
    <li>
      <a href="#">Blog</a>
    </li>
    <li>
      <a href="#">Contact Us</a>
    </li>
  </ul>
  <div class="hamburger">
    <span class="line"></span>
    <span class="line"></span>
    <span class="line"></span>
  </div>
</nav>
<div class="menubar">
  <ul>
    <li>
      <a href="#">Home</a>
    </li>
    <li>
      <a href="#">Services</a>
    </li>
    <li>
      <a href="#">Blog</a>
    </li>
    <li>
```

```
        <a href="#">Contact Us</a>
    </li>
</ul>
</div>

<script src="/assets/js/script.js"></script>
</body>
</html>
```

Modificar el archivo `main.go` para servir este archivo HTML cuando se acceda a la ruta principal ("/") de tu aplicación. Esto implica usar Gin para definir una ruta que renderice la plantilla HTML.

```
package main

import (
    "net/http"

    "github.com/gin-gonic/gin"
)

func main() {
    router := gin.Default()

    router.LoadHTMLGlob("templates/*")

    router.GET("/", func(c *gin.Context) {
        c.HTML(http.StatusOK, "index.html", gin.H{
            "title": "QR Scan App",
        })
    })
}
```

```
router.Run(":8080")
}
```

descargamos el framework de gin para poder renderizar la web

```
go get github.com/gin-gonic/gin
```

estilos css

vamos a asant y creamos un archivo llamado style.css agregamos los estilos

```
@import
url("https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,300;0,400;0,500;0,600;0,700;1,300;1,400;1,500;1,600;1,700&
display=swap");

* {
  box-sizing: border-box;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

body {
  padding: 0;
  margin: 0;
  font-family: "Poppins", sans-serif;
}

nav {
  padding: 5px 5%;
  display: flex;
  justify-content: space-between;
  align-items: center;
  box-shadow: rgba(50, 50, 93, 0.25) 0px 2px 5px -1px,
    rgba(0, 0, 0, 0.3) 0px 1px 3px -1px;
  z-index: 1;
```

```
}  
nav .logo {  
  display: flex;  
  align-items: center;  
}  
nav .logo img {  
  height: 25px;  
  width: auto;  
  margin-right: 10px;  
}  
nav .logo h1 {  
  font-size: 1.1rem;  
  background: linear-gradient(to right, #b927fc 0%, #2c64fc  
100%);  
  -webkit-background-clip: text;  
  background-clip: text;  
  -webkit-text-fill-color: transparent;  
}  
  
nav ul {  
  list-style: none;  
  display: flex;  
}  
nav ul li {  
  margin-left: 1.5rem;  
}  
nav ul li a {  
  text-decoration: none;  
  color: #000;  
  font-size: 95%;  
  font-weight: 400;  
  padding: 4px 8px;  
  border-radius: 5px;  
}  
  
nav ul li a:hover {  
  background-color: #f5f5f5;
```



```
}

.hamburger {
  display: none;
  cursor: pointer;
}

.hamburger .line {
  width: 25px;
  height: 1px;
  background-color: #1f1f1f;
  display: block;
  margin: 7px auto;
  transition: all 0.3s ease-in-out;
}

.hamburger-active {
  transition: all 0.3s ease-in-out;
  transition-delay: 0.6s;
  transform: rotate(45deg);
}

.hamburger-active .line:nth-child(2) {
  width: 0px;
}

.hamburger-active .line:nth-child(1),
.hamburger-active .line:nth-child(3) {
  transition-delay: 0.3s;
}

.hamburger-active .line:nth-child(1) {
  transform: translateY(12px);
}

.hamburger-active .line:nth-child(3) {
  transform: translateY(-5px) rotate(90deg);
}
```

```
.menubar {
  position: absolute;
  top: 0;
  left: -60%;
  display: flex;
  justify-content: center;
  align-items: flex-start;
  width: 60%;
  height: 100vh;
  padding: 20% 0;
  background: rgba(255, 255, 255);
  transition: all 0.5s ease-in;
  z-index: 2;
}

.active {
  left: 0;
  box-shadow: rgba(149, 157, 165, 0.2) 0px 8px 24px;
}

.menubar ul {
  padding: 0;
  list-style: none;
}

.menubar ul li {
  margin-bottom: 32px;
}

.menubar ul li a {
  text-decoration: none;
  color: #000;
  font-size: 95%;
  font-weight: 400;
  padding: 5px 10px;
  border-radius: 5px;
}
```

```
.menubar ul li a:hover {
  background-color: #f5f5f5;
}
@media screen and (max-width: 790px) {
  .hamburger {
    display: block;
  }
  nav ul {
    display: none;
  }
}
```

también agregamos el [scrip.js](#)

```
const mobileNav = document.querySelector(".hamburger");
const navbar = document.querySelector(".menubar");

const toggleNav = () => {
  navbar.classList.toggle("active");
  mobileNav.classList.toggle("hamburger-active");
};
mobileNav.addEventListener("click", () => toggleNav());
```

configurando Gin para servir los archivos estáticos del directorio assets.

Tienes razón. En el código actual de main.go, no estamos configurando Gin para servir los archivos estáticos del directorio assets. Por eso los estilos CSS y el JavaScript no se cargan en el navegador.

Para solucionar esto, necesitamos agregar un manejador en main.go que sirva el directorio assets bajo la ruta assets.

```
// Serve static files from the "assets" directory
router.Static("/assets", "./assets")
```

corremos el servidor local

Ahora, cuando reinicies tu aplicación Go y accedas a <http://localhost:8080>, los archivos CSS y JavaScript dentro de la carpeta assets deberían ser cargados correctamente por el navegador.

creando index.html con redireccionamiento al app generador.html

Para hacer que index.html sea la interfaz principal de tu aplicación y que contenga un botón en la barra de navegación que redirija a generador.html, puedes seguir estos pasos:

Paso 1: Modificar index.html

Agrega un enlace en la barra de navegación que redirija a generador.html. Aquí tienes un ejemplo de cómo podría verse tu archivo index.html con el enlace agregado:

```
<li>
  <a href="/generador">Generador de QR</a> <!-- Enlace al
generador -->
</li>
```

y en el menu responsi de movil

```
<li>
  <a href="/generador">Generador de QR</a> <!-- Enlace al generador -->
</li>
```

Paso 2: Asegurarte de que generador.html esté bien configurado

Tu archivo generador.html debe estar configurado para recibir la entrada del usuario y generar el código QR. Asegúrate de que el contenido de generador.html esté correctamente implementado, como se mostró anteriormente.

Paso 3: Verificar las Rutas en main.go

Asegúrate de que tu archivo main.go tenga las rutas configuradas correctamente. Aquí está el código que deberías tener:

```
// Ruta para servir index.html como la interfaz principal
router.LoadHTMLGlob("templates/*")
router.GET("/", func(c *gin.Context) {
```

```
c.HTML(http.StatusOK, "index.html", nil)

})
```

```
// Ruta para servir el formulario de generación de QR
router.GET("/generador", func(c *gin.Context) {
    c.HTML(http.StatusOK, "generador.html", nil)
})
```

```
// Ruta para generar el código QR
router.POST("/generate", func(c *gin.Context) {
    data := c.PostForm("data")
    qrCode, err := qrcode.Encode(data, qrcode.Medium, 256)
    if err != nil {
        c.String(http.StatusInternalServerError, "Error
generando el código QR")
        return
    }
    c.Header("Content-Type", "image/png")
    c.Writer.Write(qrCode)
})
```

Resumen

Interfaz Principal: index.html ahora actúa como la interfaz principal de tu aplicación.

Navegación: Se ha agregado un enlace en la barra de navegación que redirige a generador.html.

Rutas: Asegúrate de que las rutas en main.go estén configuradas correctamente para servir ambas páginas.

Diseño responsive

Para hacer que el diseño sea responsive y se adapte a diferentes dispositivos, desde resoluciones de escritorio hasta móviles y tabletas, puedes utilizar CSS Flexbox y Media Queries.

vamos a crear un archivo [stylegenerador.css](#)

```
* {
    box-sizing: border-box;
    -webkit-font-smoothing: antialiased;
```

```
    -moz-osx-font-smoothing: grayscale;
  }

body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  margin: 0;
  padding: 0;
}

h1 {
  text-align: center;
}

.qr-type {
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
  margin: 20px 0;
}

.qr-card {
  background: white;
  border: 1px solid #ccc;
  border-radius: 8px;
  padding: 20px;
  width: 200px;
  text-align: center;
  cursor: pointer;
  transition: transform 0.2s;
  margin: 10px; /* Espaciado entre tarjetas */
}

.qr-card:hover {
  transform: scale(1.05);
}

.input-container {
  display: none; /* Ocultar inicialmente */
  text-align: center;
  margin-top: 20px;
}
```

```
.input-container input {
    width: 80%;
    padding: 10px;
    margin-top: 10px;
}

.input-container button {
    padding: 10px 20px;
    margin-top: 10px;
}

/* Estilos responsivos */
@media (max-width: 1280px) {
    .qr-card {
        width: 150px; /* Ajustar el tamaño de las tarjetas en
pantallas más pequeñas */
    }
}

@media (max-width: 768px) {
    .qr-card {
        width: 100%; /* Tarjetas ocupan el 100% del ancho en
móviles */
    }
    .input-container input {
        width: 90%; /* Ajustar el ancho del input en móviles
*/
    }
}

@media (max-width: 480px) {
    h1 {
        font-size: 1.5em; /* Ajustar el tamaño del título en
móviles */
    }
    .input-container button {
        width: 100%; /* Botón ocupa el 100% del ancho en
móviles */
    }
}
```

```
}  
}
```

agregamos al script lo siguiente funciones para el input

```
function showInput(type) {  
    $('#inputContainer').show();  
    $('#data').attr('placeholder', 'Ingresa ' + type);  
    $('#data').data('type', type); // Guardar el tipo de QR  
}
```

Paso 3: Mostrar el Botón de Descarga

Asegúrate de que el botón de descarga se muestre solo cuando se haya generado un código QR. modificaremos la función generateQR() para mostrar el botón después de generar el QR:

```
function generateQR() {  
    const data = $('#data').val();  
    const type = $('#data').data('type');  
    $('#qrcode').empty(); // Limpiar el QR anterior  
    $('#qrcode').qrcode(data); // Generar nuevo QR  
    $('#downloadButton').show(); // Mostrar el botón de  
descarga  
}
```

vamos a agregar el botón de descarga vamos a generador.html

Agregaremos un botón de descarga justo después del div donde se muestra el código QR.

```
<button id="downloadButton" style="display: none;  
margin-top: 10px; padding: 10px 20px; background-color:  
#4CAF50; color: white; border: none; border-radius: 5px;  
cursor: pointer;" onclick="downloadQR()">Descargar QR</button>
```

Paso 2: Agregar la Función de Descarga

Ahora, necesitas agregar la función `downloadQR()` en tu archivo JavaScript (`script.js`) para manejar la descarga del código QR. agregaremos la función de descargar al boton de descarga vamos a [script.js](#)

```
function downloadQR() {
    const canvas = document.querySelector('#qrcode canvas');
    if (canvas) {
        // Convertir el canvas a blob
        canvas.toBlob((blob) => {
            // Crear un objeto URL para el blob
            const url = URL.createObjectURL(blob);

            // Crear un elemento <a> temporal
            const link = document.createElement('a');
            link.href = url;
            link.download = 'qrcode.png';

            // Agregar el link al documento
            document.body.appendChild(link);

            // Simular click
            try {
                link.click();
                // Mostrar mensaje de éxito
                const downloadPath = window.electron ?
'Descargas' : 'la carpeta de descargas predeterminada';
                alert(`El código QR se ha descargado
correctamente en tu carpeta de ${downloadPath}`);
            } catch (error) {
                console.error('Error al descargar:', error);
                // Fallback: Abrir en nueva ventana
                window.open(url, '_blank');
                alert('El código QR se ha abierto en una nueva
ventana. Puedes guardarlo haciendo clic derecho y
seleccionando "Guardar imagen como..."');
            }

            // Limpiar
```

```

        setTimeout(() => {
            document.body.removeChild(link);
            URL.revokeObjectURL(url);
        }, 100);
    }, 'image/png');
} else {
    alert('No hay un código QR generado para descargar.');
```

vamos a [main.go](#) y voy a crear un código que consiste en enviar el QR al servidor y generar un enlace de descarga directo. Primero, necesitamos agregar un endpoint en el servidor para manejar la descarga:

```

    "os"
    "strings"
    "time"

    "encoding/base64"
    "fmt"
// Endpoint para descargar QR
router.POST("/download-qr", func(c *gin.Context) {
    // Obtener los datos de la imagen del cuerpo de la
solicitud
    var data struct {
        ImageData string `json:"imageData"`
    }
    if err := c.BindJSON(&data); err != nil {
        c.JSON(400, gin.H{"error": "Datos inválidos"})
        return
    }

    // Decodificar la imagen base64
    imageData := strings.Split(data.ImageData, ",")[1]
    decodedData, err :=
base64.StdEncoding.DecodeString(imageData)
```

```

        if err != nil {
            c.JSON(400, gin.H{"error": "Error al decodificar
la imagen"})
            return
        }

        // Crear un nombre de archivo único
        filename := fmt.Sprintf("qrcode_%d.png",
time.Now().Unix())

        // Guardar la imagen en el servidor
        err = os.WriteFile("temp/"+filename, decodedData,
0644)

        if err != nil {
            c.JSON(500, gin.H{"error": "Error al guardar la
imagen"})
            return
        }

        // Devolver la URL para descargar
        c.JSON(200, gin.H{
            "downloadUrl": "/download/" + filename,
        })
    })

    // Endpoint para servir los archivos descargados
    router.GET("/download/:filename", func(c *gin.Context) {
        filename := c.Param("filename")
        c.File("temp/" + filename)
    })

```

Resumen

Botón de Descarga: Se ha agregado un botón que permite a los usuarios descargar el código QR generado.

Función de Descarga: La función `downloadQR()` maneja la lógica para descargar la imagen del QR.

Mostrar Botón: El botón de descarga se muestra solo después de que se haya generado un código QR.

Refresh input y botón de eliminar

Para implementar la funcionalidad donde al seleccionar un tipo de QR se refresque el input y se agregue un botón para eliminar lo que se haya escrito, puedes seguir estos pasos:

Paso 1: Modificar el HTML

Agrega un botón de eliminación junto al input y asegúramos de que el input se limpie cuando se seleccione un nuevo tipo de QR. vamos a el archivo `generador.html`:

```
<button onclick="clearInput()" style="margin-left: 10px;
padding: 10px 20px; background-color: #f44336; color: white;
border: none; border-radius: 5px; cursor:
pointer;">Eliminar</button>
```

Paso 2: Agregar la Función para Limpiar el Input

Ahora, necesitamos agregar la función `clearInput()` en tu archivo JavaScript (`script.js`) para manejar la limpieza del input:

```
function clearInput() {
  $('#data').val(''); // Limpiar el input
}
```

Paso 3: Modificar la Función `showInput`

Asegúrate de que la función `showInput()` limpie el input cada vez que se seleccione un nuevo tipo de QR:

```
function showInput(type) {
  $('#inputContainer').show();
  $('#data').attr('placeholder', 'Ingresa ' + type);
  $('#data').data('type', type); // Guardar el tipo de QR
  clearInput(); // Limpiar el input al seleccionar un nuevo
  tipo
}
```

Para agregar un botón que permita a los usuarios crear un nuevo código QR y borrar el que se está mostrando actualmente, seguire estos pasos:

Paso 1: Modificar el HTML

Agrega un botón "Crear Nuevo QR" junto al botón de eliminación. vamos al archivo generador.html:

```
<button onclick="createNewQR()" style="margin-left: 10px; padding: 10px 20px; background-color: #2196F3; color: white; border: none; border-radius: 5px; cursor: pointer;">Crear Nuevo QR</button>
```

Paso 2: Agregar la Función para Crear un Nuevo QR

Ahora, necesito agregar la función createNewQR() en el archivo JavaScript (script.js) para manejar la creación de un nuevo QR y borrar el que se está mostrando:

```
function createNewQR() {  
  $('#qrcode').empty(); // Limpiar el QR actual  
  $('#downloadButton').hide(); // Ocultar el botón de descarga  
  clearInput(); // Limpiar el input  
}
```

Paso 3: Modificar la Función generateQR

Asegúre de que la función generateQR() muestre el botón de descarga después de generar un nuevo QR:

```
function generateQR() {  
  const data = $('#data').val();  
  const type = $('#data').data('type');  
  $('#qrcode').empty(); // Limpiar el QR anterior  
  $('#qrcode').qrcode(data); // Generar nuevo QR  
  $('#downloadButton').show(); // Mostrar el botón de  
  descarga  
}
```

Resumen

Botón de Eliminación: Se ha agregado un botón que permite a los usuarios eliminar lo que han escrito en el input.

Limpieza del Input: La función `clearInput()` se encarga de limpiar el input.

Refresco del Input: Cada vez que se selecciona un nuevo tipo de QR, el input se limpia automáticamente.

Botón de Crear Nuevo QR: Se ha agregado un botón que permite a los usuarios crear un nuevo código QR y borrar el que se está mostrando actualmente.

Limpieza del QR: La función `createNewQR()` se encarga de limpiar el QR actual y ocultar el botón de descarga.

Refresco del Input: Cada vez que se crea un nuevo QR, el input se limpia automáticamente.

v5

creando escaneador qr

creamos una ruta para el módulo de app escaneo qr en el servido [main.go](#)

```
router.GET("/escaner", func(c *gin.Context) {
    c.HTML(http.StatusOK, "escaner.html", nil)
})
```

creando interfaz y funcionalidad nos vamos a `escaner.html` agregamos el código html con js vamos a crear una nueva página para el escáner:

```
<div class="scanner-container">
  <h1>Escáner de Códigos QR</h1>
  <div class="controls">
    <button id="startButton">Iniciar Cámara</button>
    <button id="stopButton" disabled>Detener Cámara</button>
    <input type="file" id="fileInput" accept="image/*"
style="display: none;">
    <button id="selectFileButton">Seleccionar QR
Archivos</button>
  </div>
  <div id="video-container">
    <video id="video" playsinline></video>
    <canvas id="canvas"></canvas>
  </div>
  <div class="result-container">
    <h2>Resultado del Escaneo</h2>
    <div id="result">Esperando escaneo...</div>
```

```
</div>
</div>
```

ahora vamos a agregar el [style.css](#)

```
/*-----escaner
estilo-----*/
.scanner-container {
  max-width: 800px;
  margin: 0 auto;
  text-align: center;
}
#video-container {
  position: relative;
  width: 100%;
  max-width: 640px;
  margin: 20px auto;
}
#video {
  width: 100%;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
#canvas {
  display: none;
}
.result-container {
  margin-top: 20px;
  padding: 15px;
  background: white;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
.controls {
  margin: 20px 0;
}
```

```

button {
  padding: 10px 20px;
  margin: 0 10px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  background-color: #4CAF50;
  color: white;
  font-size: 16px;
}
button:hover {
  opacity: 0.9;
}
#result {
  margin-top: 10px;
  padding: 10px;
  word-break: break-all;
}

```

agregaremos el código js para que funcione el app de escaneo vamos a SCRIPT.JS

```

let video = document.getElementById('video');
let canvas = document.getElementById('canvas');
let startButton = document.getElementById('startButton');
let stopButton = document.getElementById('stopButton');
let resultDiv = document.getElementById('result');
let stream = null;
let scanning = false;

// Función para iniciar la cámara
async function startCamera() {
  try {
    // Asegurarse de que cualquier stream anterior esté
detenido
    if (stream) {
      stopCamera();
    }
  }
}

```



```

    // Reiniciar el estado
    scanning = false;
    resultDiv.textContent = "Esperando escaneo...";

    // Obtener nuevo stream de la cámara
    stream = await navigator.mediaDevices.getUserMedia({
        video: {
            facingMode: "environment",
            width: { ideal: 1280 },
            height: { ideal: 720 }
        }
    });

    // Configurar el video
    video.srcObject = stream;
    await video.play();

    // Actualizar estado de los botones
    startButton.disabled = true;
    stopButton.disabled = false;

    // Iniciar el escaneo
    scanning = true;
    scanQR();
} catch (err) {
    console.error("Error al acceder a la cámara:", err);
    resultDiv.textContent = "Error al acceder a la cámara.
Por favor, asegúrate de dar permisos de cámara.";
    startButton.disabled = false;
    stopButton.disabled = true;
}
}

// Función para detener la cámara
function stopCamera() {
    if (stream) {

```

```

        // Detener todas las pistas del stream
        stream.getTracks().forEach(track => {
            track.stop();
        });
        video.srcObject = null;
        stream = null;
    }

    // Reiniciar el estado
    scanning = false;
    startButton.disabled = false;
    stopButton.disabled = true;
    resultDiv.textContent = "Cámara detenida";
}

// Función para escanear el código QR
function scanQR() {
    if (!scanning) return;

    if (video.readyState === video.HAVE_ENOUGH_DATA) {
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;
        let context = canvas.getContext('2d');
        context.drawImage(video, 0, 0, canvas.width,
canvas.height);

        let imageData = context.getImageData(0, 0,
canvas.width, canvas.height);
        let code = jsQR(imageData.data, imageData.width,
imageData.height);

        if (code) {
            resultDiv.textContent = `Código QR detectado:
${code.data}`;
            handleQRCode(code.data);
        }
    }
}

```

```

        if (scanning) {
            requestAnimationFrame(scanQR);
        }
    }

// Función para manejar diferentes tipos de códigos QR
function handleQRCode(data) {
    try {
        if (data.startsWith('http://') ||
data.startsWith('https://')) {
            resultDiv.innerHTML += '<br>Tipo: URL';
        } else if (data.startsWith('BEGIN:VCARD')) {
            resultDiv.innerHTML += '<br>Tipo: Tarjeta de
Contacto';
        } else if (data.startsWith('mailto:')) {
            resultDiv.innerHTML += '<br>Tipo: Correo
Electrónico';
        } else if (data.startsWith('tel:')) {
            resultDiv.innerHTML += '<br>Tipo: Número de
Teléfono';
        } else if (data.startsWith('WIFI:')) {
            resultDiv.innerHTML += '<br>Tipo: Credenciales
WiFi';
        } else if (data.startsWith('geo:')) {
            resultDiv.innerHTML += '<br>Tipo: Ubicación
Geográfica';
        } else {
            resultDiv.innerHTML += '<br>Tipo: Texto Plano';
        }
    } catch (error) {
        console.error('Error al procesar el código QR:',
error);
    }
}

// Event listeners

```

```
startButton.addEventListener('click', startCamera);
stopButton.addEventListener('click', stopCamera);

// Funcionalidad para escanear archivos QR
const fileInput = document.getElementById('fileInput');
const selectFileButton =
document.getElementById('selectFileButton');

// Función para manejar la selección de archivo
selectFileButton.addEventListener('click', () => {
    fileInput.click();
});

fileInput.addEventListener('change', (event) => {
    const file = event.target.files[0];
    if (file) {
        const reader = new FileReader();
        reader.onload = function(e) {
            const img = new Image();
            img.onload = function() {
                // Crear un canvas temporal para procesar la
imagen
                const tempCanvas =
document.createElement('canvas');
                const tempContext =
tempCanvas.getContext('2d');
                tempCanvas.width = img.width;
                tempCanvas.height = img.height;
                tempContext.drawImage(img, 0, 0);

                // Obtener los datos de la imagen
                const imageData = tempContext.getImageData(0,
0, tempCanvas.width, tempCanvas.height);

                // Intentar decodificar el código QR
                const code = jsQR(imageData.data,
imageData.width, imageData.height);
```

```

        if (code) {
            resultDiv.textContent = `Código QR
detectado: ${code.data}`;
            handleQRCode(code.data);
        } else {
            resultDiv.textContent = "No se pudo
detectar un código QR en la imagen.";
        }
    };
    img.src = e.target.result;
};
reader.readAsDataURL(file);
}
});

```

Características implementadas:

1. Compatibilidad con PC y Móvil:

- Usa la API `getUserMedia` para acceder a la cámara
- Prefiere la cámara trasera en dispositivos móviles
- Interfaz responsiva

2. Funcionalidades:

- Iniciar/Detener la cámara
- Escaneo en tiempo real
- Detección automática del tipo de QR
- buscar y escanear QR que está en el explorador de archivos de la pc
- Muestra el resultado del escaneo

3. Manejo de diferentes tipos de QR:

- URLs
- Tarjetas de contacto (vCard)
- Correos electrónicos
- Números de teléfono
- Credenciales WiFi
- Ubicaciones geográficas
- Texto plano

4. Interfaz de usuario:

- Diseño limpio y moderno
- Controles intuitivos
- Feedback visual del escaneo
- Mensajes de error claros

Uso:

1. Accede a la página del escáner
2. Haz clic en "Iniciar Cámara"
3. Permite el acceso a la cámara cuando el navegador lo solicite
4. Apunta la cámara hacia un código QR
5. El resultado se mostrará automáticamente
6. Usa "Detener Cámara" cuando hayas terminado

vamos a crear la interfaz de index.html


vamos a agregar la estructura del index.html

```
<link rel="stylesheet"
href="/assets/css/style_index.css" />

<!-- Sección Hero con Video -->
<section class="hero-section">
  <video class="video-background" autoplay muted loop
playsinline>
    <source src="/assets/videos/background.mp4"
type="video/mp4">
    Tu navegador no soporta el elemento de video.
  </video>
  <div class="hero-content">
    <h1 class="hero-title">QR Scanner App</h1>
    <p class="hero-description">
      Tu solución completa para generar y escanear
códigos QR.
      Fácil de usar, rápido y compatible con todos
los dispositivos.
    </p>
  </div>
</section>

<!-- Sección de Características -->
<section class="features-section">
```

```

<div class="features-grid">
  <div class="feature-card">
    <div class="feature-icon">

```

```

        </p>
    </div>

    <div class="feature-card">
        <div class="feature-icon">📞</div>
        <h3 class="feature-title">Números de
Teléfono</h3>
        <p class="feature-description">
            Escanea códigos QR con números de teléfono
para iniciar llamadas rápidamente.
        </p>
    </div>

    <div class="feature-card">
        <div class="feature-icon">📶</div>
        <h3 class="feature-title">Credenciales
WiFi</h3>
        <p class="feature-description">
            Conéctate a redes WiFi automáticamente
escaneando códigos QR.
        </p>
    </div>
</div>

</section>

<script src="/assets/js/script_index.js"></script>

```

ahora vamos a agregar el css en el [style_index.css](#)

```

.hero-section {
    position: relative;
    height: 100vh;
    width: 100%;
    overflow: hidden;
}

.video-background {

```



```
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    object-fit: cover;
    z-index: -1;
}

.hero-content {
    position: relative;
    z-index: 1;
    color: white;
    text-align: center;
    padding: 2rem;
    background: rgba(0, 0, 0, 0.5);
    height: 100%;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

.hero-title {
    font-size: 3rem;
    margin-bottom: 1rem;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
}

.hero-description {
    font-size: 1.2rem;
    max-width: 800px;
    margin: 0 auto;
    text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.5);
}

.features-section {
```

```
padding: 4rem 2rem;
background: #f5f5f5;
}

.features-grid {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(300px,
1fr));
gap: 2rem;
max-width: 1200px;
margin: 0 auto;
padding: 2rem;
}

.feature-card {
background: white;
border-radius: 10px;
padding: 2rem;
box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
transition: transform 0.3s ease;
opacity: 0;
transform: translateY(20px);
}

.feature-card.visible {
opacity: 1;
transform: translateY(0);
}

.feature-card:hover {
transform: translateY(-5px);
}

.feature-icon {
font-size: 2.5rem;
margin-bottom: 1rem;
color: #4CAF50;
```

```

}

.feature-title {
  font-size: 1.5rem;
  margin-bottom: 1rem;
  color: #333;
}

.feature-description {
  color: #666;
  line-height: 1.6;
}

@media (max-width: 768px) {
  .hero-title {
    font-size: 2rem;
  }
  .hero-description {
    font-size: 1rem;
  }
  .features-grid {
    grid-template-columns: 1fr;
  }
}

```

ahora vamos a agregar el código js para la animaciones de las card

```

// Función para animar las cards al hacer scroll
function animateCards() {
  const cards = document.querySelectorAll('.feature-card');
  const observer = new IntersectionObserver((entries) => {
    entries.forEach(entry => {
      if (entry.isIntersecting) {
        entry.target.classList.add('visible');
      }
    });
  });
}

```

```

    }, {
      threshold: 0.1
    });

    cards.forEach(card => {
      observer.observe(card);
    });
  }

// Iniciar la animación cuando el DOM esté cargado
document.addEventListener('DOMContentLoaded', animateCards);

```

resumen

se creó la estructura del index.html
 se creó los estilos para él la estructura
 se creó las animaciones con el script

anexos

