Paso 1: Verifica la URL del código QR

* Cada código QR debe apuntar a una URL única que incluya un identificador del QR. Por ejemplo

https://tudominio.com/qr?id=12345

* Aquí, 12345 es el ID único del código QR en tu base de datos.
* Si estás usando Replit, asegúrate de que la URL sea accesible públicamente. Puedes usar el dominio proporcionado por Replit o configurar un dominio personalizado.

Paso 2: Implementa el endpoint para rastrear escaneos

En el backend, necesitas un endpoint que maneje las solicitudes cuando se escanea un código QR. Aquí tienes un ejemplo básico en Node.js con Express :

const express = require("express");

const app = express();

const bodyParser = require("body-parser");

const mongoose = require("mongoose");

// Conexión a la base de datos (usando MongoDB como ejemplo)

mongoose.connect("mongodb+srv://usuario:contraseña@cluster.mongodb.net/qr\_app?retryWrites=true&w=majority", {

useNewUrlParser: true,

useUnifiedTopology: true,

});

// Modelo para almacenar estadísticas de escaneos

const ScanSchema = new mongoose.Schema({

qrId: String, // ID del código QR

timestamp: Date, // Fecha y hora del escaneo

location: String, // Ubicación aproximada (opcional)

device: String, // Dispositivo utilizado (opcional)

});

const Scan = mongoose.model("Scan", ScanSchema);

// Middleware para analizar JSON

app.use(bodyParser.json());

// Endpoint para rastrear escaneos

app.get("/qr", async (req, res) => {

const qrId = req.query.id; // Obtener el ID del QR desde la URL

if (!qrId) {

return res.status(400).send("ID del QR no proporcionado.");

}

// Registrar el escaneo en la base de datos

const scanData = {

qrId: qrId,

timestamp: new Date(),

location: req.headers["x-forwarded-for"] || req.connection.remoteAddress, // Ubicación aproximada

device: req.headers["user-agent"], // Información del dispositivo

};

try {

const newScan = new Scan(scanData);

await newScan.save();

console.log(`Escaneo registrado para QR ID: ${qrId}`);

} catch (error) {

console.error("Error al registrar el escaneo:", error);

}

// Redirigir al usuario a una página o mensaje

res.send(`<h1>¡Gracias por escanear el código QR!</h1><p>ID del QR: ${qrId}</p>`);

});

// Iniciar el servidor

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

console.log(`Servidor iniciado en el puerto ${PORT}`);

});

Paso 3: Configura la generación de códigos QR

Cuando generas un código QR, asegúrate de que el contenido del QR sea la URL correcta con el ID del QR. Por ejemplo:

const QRCode = require("qrcode");

// Generar un código QR con un ID específico

const generateQR = async (qrId) => {

const url = `https://tudominio.com/qr?id=${qrId}`;

const qrCode = await QRCode.toDataURL(url); // Genera el QR como una imagen base64

return qrCode;

};

Paso 4: Muestra las estadísticas en el frontend

Para mostrar las estadísticas de escaneos en el frontend, puedes crear un endpoint que recupere los datos de la base de datos. Por ejemplo :

// Endpoint para obtener estadísticas de un QR específico

app.get("/stats/:qrId", async (req, res) => {

const qrId = req.params.qrId;

try {

const scans = await Scan.find({ qrId }).sort({ timestamp: -1 }); // Ordenar por fecha descendente

res.json(scans);

} catch (error) {

console.error("Error al obtener estadísticas:", error);

res.status(500).send("Error al obtener estadísticas.");

}

});

Luego, en el frontend, puedes hacer una solicitud a este endpoint para mostrar las estadísticas:

// Ejemplo de solicitud al backend para obtener estadísticas

fetch(`/stats/${qrId}`)

.then((response) => response.json())

.then((data) => {

console.log("Estadísticas:", data);

// Mostrar los datos en la interfaz

})

.catch((error) => console.error("Error al cargar estadísticas:", error));

RECOMENDACION ADICIONAÑ

1. Pruebas locales :
   * Usa herramientas como Postman para probar el endpoint /qr y asegurarte de que registra los escaneos correctamente.
   * Escanea un código QR generado localmente y verifica que los datos se almacenen en la base de datos.
2. Depuración :
   * Asegúrate de que el backend esté recibiendo las solicitudes correctamente. Puedes agregar console.log en el endpoint para depurar.
3. Ubicación geográfica :
   * Para obtener la ubicación aproximada, puedes usar la IP del usuario (req.headers["x-forwarded-for"]) y pasarla a un servicio como [ipstack](https://ipstack.com/) o [ipinfo.io](https://ipinfo.io/) .
4. Dispositivos :
   * La cabecera User-Agent (req.headers["user-agent"]) te da información sobre el dispositivo utilizado para escanear el QR.
5. Seguridad :
   * Si planeas usar esta aplicación en producción, asegúrate de proteger tus endpoints con autenticación y validación de datos.