PDIH - TRABAJO

Programación de códigos QR con Python, Java, JavaScript, C++

Jorge Sánchez - Carlos Guasp

UNIVERSIDAD
DE GRANADA

ÍNDICE

Introducción..... 3
Estructura del proyecto..... 3
Implementación..... 4
Python..... 4
Java..... 6
JavaScript..... 9
C++..... 10

Introducción

Este trabajo consiste en el desarrollo de uno de los temas propuestos por el profesor para el trabajo, el desarrollo de programas en diferentes lenguajes de programación con la capacidad de generar códigos QR que direccionen al enlace enviado.

Para el desarrollo de realizó el código en los lenguajes: Python, Java, C++ y JavaScript. Para el desarrollo del programa en cada uno de ellos ha sido necesario investigar acerca de las librerías disponibles para cada uno de ellos y diseñar un código compacto, pero completamente funcional.

Estructura del proyecto

```
|- Trabajo/
|------ C++/
|         |------ src/
|                   |------ lodepng.cpp
|                   |------ lodepng.h
|                   |------ main.cpp
|
|------ Java/QrCode/
|         |------ .idea/
|         |------ src/main/java/
|         |         |------ QRGenerador.java
|         |------ pom.xml
|
|------ JavaScript/
|         |------ generar_qr.js
|
|------ python/
          |------ main.py
```

Implementación

Python

En python gracias a las comodidades que ofrece el lenguaje el desarrollo del código fue realmente sencillo ya que al importar la librería **qrcode** teníamos todo lo necesario para el desarrollo del generador del código qr.

Tanto a este lenguaje como a los demás nos aseguramos de que las url empezaran por 'http://' o 'https://' para garantizar una cierta seguridad al dirigirse a sus url correspondientes.

Al crear un objeto de la clase QRCode se podía añadir la url usando el método *'add_data'* y generar después la imagen con el método *'make_image'*.

Código:

```
# Generador de código QR en Node.js en Python
# Este script genera un código QR a partir de una URL proporcionada como
argumento de línea de comandos.
# Requiere la librería qrcode, que se puede instalar con pip:
# pip install qrcode[pil]
#
# Uso: python3 main.py <URL>
# Ejemplo: python3 main.py https://www.ejemplo.com
#
# El código QR se guardará en un archivo llamado 'codigo_qr.png' en el
directorio actual.
# Al ser escaneado, con un teléfono móvil o un lector de QR, abrirá la
URL proporcionada.
#
# Trabajo realizado por: Jorge Sánchez y Carlos Guasp

import sys
import qrcode


# Función para validar la URL
# Parámetros:
# - url: La URL que se desea validar
def validar_url(url):
    # Comprobar si la URL comienza con http:// o https://
    if not url.startswith(('http://', 'https://')):
        raise ValueError("La URL debe comenzar con 'http://' o
'https://'")

# Función para generar un código QR a partir de una URL
```

```
# Parámetros:
# - url: La URL que se desea codificar en el QR
# - ficheroSalida: Nombre del archivo donde se guardará la imagen del QR
def genera_Qr(url, ficheroSalida='codigo_qr.png'):
    # Crear un objeto de tipo QRCode
    qr = qrcode.QRCode(
        version=1,
        error_correction=qrcode.constants.ERROR_CORRECT_L,
        box_size=10,
        border=4,
    )
    # Añadir la URL al objeto QR
    qr.add_data(url)
    qr.make(fit=True)

    # Validar la URL
    try:
        validar_url(url)
    except ValueError as e:
        print(f"Error: {e}")
        return

    # Crear una imagen del código QR
    img = qr.make_image(fill_color="black", back_color="white")

    # Guardar la imagen en un archivo
    img.save(ficheroSalida)
    print(f"Código QR guardado en {ficheroSalida}")


if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Uso: python generar_qr.py <URL>")
    else:
        url = sys.argv[1]
        genera_Qr(url)
```

Resultado

Java

Para el desarrollo del código en Java optamos por el desarrollo de un proyecto de Maven en le IDE Intellij Idea ya que permite desarrollar proyectos de Maven de una forma muy sencilla y rápida.

Investigando acerca de cómo realizar el código en Java nos encontramos con **ZXing** que nos permitía crear y escanear tanto qr como códigos de barras, por lo que optamos por esta librería para el desarrollo del código.

Al igual que en python decidimos comprobar que la url era “segura”, es decir que empezaba por 'http://' o 'https://', para generar el QR usamos la clase *QRCodeWriter* y usando una *BufferedImage* creamos el QR.

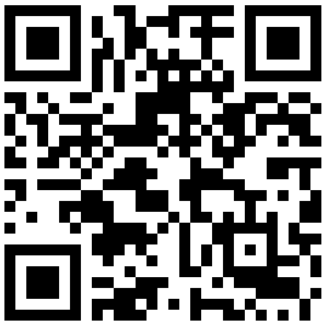
Código:

```
// Generador de código QR en Java
// Este programa genera un código QR a partir de una URL proporcionada
como argumento de línea de comandos.
// Requiere la biblioteca ZXing para la generación de códigos QR.
//
// Para compilar y ejecutar este programa es recomendable usar un IDE
como IntelliJ IDEA ya que facilita la
// compilación y ejecución del proyecto de Maven.
//
// Uso (IteliJ IDEA):
// 1. Selecciona el archivo QRGenerator.java en el panel de la
izquierda.
// 2. Haz clic derecho y selecciona "Run 'QRGenerator.main()'".
// 3. En la ventana de ejecución, proporciona la URL que deseas
codificar en el código QR como parametro.
// 4. El código QR se generará y se guardará como "codigo_qr.png" en el
directorio de trabajo actual.
//
// Trabajo realizado por: Jorge Sánchez y Carlos Guasp


import com.google.zxing.BarcodeFormat;
import com.google.zxing.WriterException;
import com.google.zxing.qrcode.QRCodeWriter;
import com.google.zxing.common.BitMatrix;
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

/**
```

```java
 * Clase QRGenerator
 * Esta clase genera un código QR a partir de una URL proporcionada como argumento de línea de comandos.
 * El código QR se guarda como una imagen PNG en el directorio de trabajo actual.
 */
public class QRGenerator {
    // Este método recibe una URL como argumento y genera un código QR a partir de ella.
    public static void main(String[] args) {
        // Verifica que se haya proporcionado un argumento
        if (args.length != 1) {
            System.out.println("Uso: java QRGenerator <URL>");
            return;
        }

        // Obtiene la URL del argumento y la verifica
        String url = args[0];

        if (!url.startsWith("http://") && !url.startsWith("https://")) {
            System.out.println("La URL debe comenzar con 'http://' o 'https://'");
            return;
        }

        // Fichero de salida
        String ficheroSalida = "codigo_qr.png";

        // Genera el código QR y lo guarda en el fichero de salida
        try {
            genera_Qr(url, 350, ficheroSalida);
            System.out.println("Código QR guardado en " + ficheroSalida);
        } catch (WriterException | IOException e) {
            System.out.println("Error al generar el código QR: " + e.getMessage());
        }
    }

    /**
     * Genera un código QR a partir de una cadena de texto y lo guarda como una imagen PNG.
     *
     * @param Data La cadena de texto que se codificará en el código QR.
     * @param tam El tamaño del código QR.
     * @param ficheroSalida La ruta del archivo donde se guardará la
```

```
imagen del código QR.
     * @throws WriterException Si ocurre un error al generar el código
QR.
     * @throws IOException Si ocurre un error al guardar la imagen.
     */
    private static void genera_Qr(String Data, int tam, String
ficheroSalida)
            throws WriterException, IOException {
        QRCodeWriter qrCodeWriter = new QRCodeWriter();

        BitMatrix bitMatrix = qrCodeWriter.encode(Data,
BarcodeFormat.QR_CODE, tam, tam);

        BufferedImage bufferedImage = new BufferedImage(tam, tam,
BufferedImage.TYPE_INT_RGB);
        for (int i = 0; i < tam; i++) {
            for (int j = 0; j < tam; j++) {
                // Establece el color del píxel en blanco o negro
                int color = (bitMatrix.get(i, j)) ? 0xFF000000 :
0xFFFFFFFF;
                bufferedImage.setRGB(i, j, color);
            }
        }

        // Guarda la imagen como un archivo PNG
        ImageIO.write(bufferedImage, "png", new File(ficheroSalida));
    }
}
```

Resultado



JavaScript

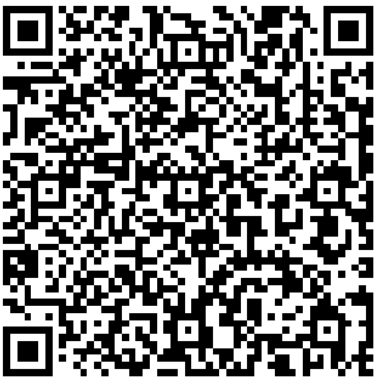
Para el caso de JavaScript el desarrollo es muy similar al de python ya que se usa también una librería llamada *'qrcode'* que desarrolla todo el proceso de una manera muy simple. Y usando *'QRCode.toFile*' podemos crear la imagen correspondiente al qr.

Código:

```
// Generador de código QR en Node.js en JavaScript
// Este script genera un código QR a partir de una URL proporcionada
como argumento de línea de comandos.
// Requiere la biblioteca 'qrcode' que se puede instalar con 'npm
install qrcode'
//
// Uso: node generar_qr.js <URL>
// Ejemplo: node generar_qr.js https://www.ejemplo.com
//
// Al ejecutar el script, se generará un archivo PNG con el código QR en
el directorio actual.
// Escaneando el código QR, se abrirá la URL proporcionada en un
navegador web.
//
// Trabajo realizado por: Jorge Sánchez y Carlos Guasp


// Importa la biblioteca 'qrcode' para generar códigos QR
const QRCode = require('qrcode');

const url = process.argv[2];

// Verifica si hay un argumento de tipo URL
if (!url) {
  console.log('Uso: node generar_qr.js <URL>');
  process.exit(1);
}

// Archivo de salida donde se guardará el código QR generado
const ficheroSalida = 'codigo_qr.png';

// Genera el código QR y lo guarda en un archivo PNG
QRCode.toFile(ficheroSalida, url, {
  width: 350,
  errorCorrectionLevel: 'H',
}, function (err) {
  if (err) throw err;
```

```
  console.log(`Código QR guardado en ${ficheroSalida}`);
});
```

Resultado



C++

Para el desarrollo del código en c++ nos encontramos con bastantes más problemas que en el resto de lenguajes ya que la descarga de una librería para implementar el generador de códigos QR no era tan sencilla como en el resto de lenguajes, por lo que investigando en internet topamos con **lodepng** capaz de pasar el qr a PNG.

Código:

```
// QR Code Generator in C++
// Este programa genera un código QR a partir de una URL proporcionada
como argumento de línea de comandos.
// Requiere la biblioteca QRencode y LodePNG para la generación y
guardado de imágenes PNG respectivamente.
//
// Uso:
// g++ -o qr_generator main.cpp lodepng.cpp -lqrencode
// ./qr_generator "https://example.com"
//
// Al ejecutar el programa, se generará un archivo PNG llamado
"codigo_qr.png" en el directorio actual.
//
// Trabajo realizado por: Jorge Sánchez y Carlos Guasp

#include <iostream>
#include <vector>
#include <qrencode.h>
#include "lodepng.h"
```

```
using namespace std;

// Función para guardar la matriz QR como PNG
void genera_Qr(const vector<unsigned char>& qr, int tam, const string&
ficheroSalida) {
   vector<unsigned char> imagen(tam * tam * 4);
    for (int i = 0; i < tam; i++) {
        for (int j = 0; j < tam; j++) {
            int indice = i * tam + j;
            unsigned char color = qr[indice] & 0x01 ? 0 : 255; // negro
o blanco
            int indicePixel = 4 * indice;
            imagen[indicePixel] = color;
            imagen[indicePixel + 1] = color;
            imagen[indicePixel + 2] = color;
            imagen[indicePixel + 3] = 255;
        }
    }
    unsigned int error = lodepng::encode(ficheroSalida, imagen, tam,
tam);
    if (error) {
        cerr << "Error guardando PNG: " << lodepng_error_text(error) <<
endl;
    } else {
        cout << "QR guardado en " << ficheroSalida << endl;
    }
}

int main(int argc, char* argv[]) {
    // Compobación del número de argumentos
    if (argc != 2) {
        cout << "Uso: " << argv[0] << " <URL>" << endl;
        return 1;
    }
    string url = argv[1];

    //Valida la url de que empiece por https o http
    if (url.substr(0, 7) != "http://" && url.substr(0, 8) != "https://")
{
        cerr << "Error: La URL debe comenzar con 'http://' o 'https://'"
<< endl;
        return 1;
    }

    // Generar QR
    QRcode* qrcode = QRcode_encodeString(url.c_str(), 0, QR_ECLEVEL_L,
```

```
QR_MODE_8, 1);
    if (!qrcode) {
        cerr << "Error al generar QR" << endl;
        return 1;
    }

    int tam = qrcode->width;
    // Guardar imagen en PNG
    genera_Qr(vector<unsigned char>(qrcode->data, qrcode->data + tam *
tam), tam, "codigo_qr.png");

    QRcode_free(qrcode);
    return 0;
}
```

Resultado