Proyecto Final

“Generación de códigos QR desde 0”

Datos y algoritmos II

Jeremías Figueroa García

2025

EAFIT

Introducción:

El objetivo de este proyecto fue desarrollar un sistema capaz de generar códigos QR desde cero, sin utilizar librerías externas para la construcción de la matriz ni el manejo del formato. El enfoque principal fue comprender en profundidad el funcionamiento interno del estándar QR, desde la estructura de la matriz hasta los mecanismos de codificación y corrección de errores.

Para ello, se utilizó el lenguaje de programación **Python**, dividiendo el código en módulos para una mayor organización. Se trabajó con estructuras de datos personalizadas y lógica matemática adaptada al estándar QR versión 2 (33x33 módulos).

1. Comprensión del funcionamiento

El funcionamiento base de los códigos qr esta dado por 3 características claves como lo son:

- **Los finder patterns** que son los 3 cuadrados idénticos que están en todos los qr, estos se usan para poder detectar la orientación de un código aun así este volteado, ayudan a dimensionar el tamaño de esté en una imagen y al ser una característica invariante los scanners los reconocen fácilmente.

- **Los Alignment patterns** es un cuadrado más pequeño que se encuentra algo más adentro de la esquina inferior derecha este se usa para corregir distorsiones y para mapear correctamente el qr en superficies irregulares, su posición puede variar según la versión del qr .

- **Los timing patterns** estas son las dos lineas que acompañan a los finder patterns y sirven como una regla para definir el tamaño de los pixeles, permitiendo ajustar la escala para su lectura y evitar errores por desplazamientos

Estructura de llenado:

Para comenzar con el llenado de información en el código QR, primero es necesario convertir la URL en una secuencia binaria.

Una vez obtenida esta secuencia, se inicia el proceso de llenado en la parte inferior derecha de la matriz del QR. El algoritmo sigue un patrón específico: avanza primero hacia arriba en una columna doble (de derecha a izquierda), luego cambia de dirección y baja en la siguiente columna doble, repitiendo esta secuencia en forma de zigzag vertical. Durante este recorrido, se evitan las posiciones ya reservadas para patrones de detección, alineación y sincronización.

El llenado de datos comienza con los primeros 4 bits, que indican el modo de codificación utilizado (por ejemplo, numérico, alfanumérico o binario). A continuación, se insertan 8 bits que representan la longitud de la URL (es decir, el número de caracteres que contiene). Finalmente, se procede a insertar los bloques de datos, donde cada carácter de la URL es codificado en un bloque de 8 bits.

Control de errores:

Una vez completado el llenado de los datos binarios correspondientes a la URL, se procede a incorporar la información de control de errores. Esta etapa es fundamental para garantizar que el código QR pueda ser leído correctamente incluso si ha sufrido daños parciales o está parcialmente obstruido.

El control de errores en los códigos QR se implementa utilizando el algoritmo de corrección Reed-Solomon. Este algoritmo añade una serie de bloques adicionales que no codifican información directa del mensaje, pero permiten recuperar los datos originales en caso de errores durante la lectura.

La cantidad de datos de corrección generados depende del nivel de corrección elegido (L, M, Q o H), siendo H el más resistente a errores, pero también el que deja menos espacio disponible para datos. En una implementación estándar, estos códigos de corrección se calculan a partir de la secuencia binaria completa de datos y se insertan en la matriz de forma similar, siguiendo el mismo patrón de llenado en zigzag y respetando las posiciones reservadas.

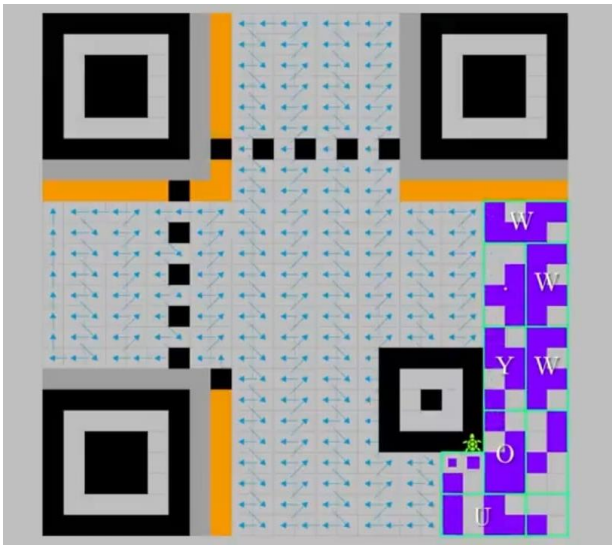
Este proceso asegura que, aunque una parte del código QR se deteriore, el lector pueda reconstruir la información original utilizando los datos de corrección insertados.

Implementacion de algoritmo de lectura de datos:

El algoritmo de lectura comienza desde la parte inferior derecha de la matriz del QR y sigue el mismo recorrido en zigzag vertical que se utilizó para el llenado. Se leen dos columnas a la vez, subiendo y bajando alternadamente, mientras se saltan las posiciones reservadas (como los patrones de detección, alineación y sincronización).

Durante la lectura, se van extrayendo los bits en el orden en que fueron insertados. Los primeros 4 bits leídos indican el modo de codificación (por ejemplo, binario), seguidos por los 8 bits que representan la longitud del mensaje. A partir de ahí, se reconstruyen los bloques de datos, agrupando los bits en unidades de 8 para convertirlos de nuevo en caracteres ASCII.

(esta imagen ejemplifica bien la estructura que se plantea)



Construcción de la matriz QR

Para construir la matriz del código QR, se implementó primero una clase llamada matriz_Base encargada de definir todos los patrones fijos que siempre están presentes en un QR, como los patrones de detección (ubicados en tres esquinas), los patrones de alineación, el patrón de sincronización **(**timing pattern) y las áreas reservadas para la información de formato y versión.

Una vez establecida la estructura base de la matriz, se desarrolló en otra clase una función encargada del **llenado de los datos**. Esta función toma como entrada la secuencia binaria generada a partir de la URL y comienza a llenar la matriz desde la parte inferior derecha, siguiendo el patrón en zigzag previamente descrito. Durante este proceso, se asegura que:

- El llenado comience alineado correctamente con los patrones de alineación.
- Se respeten las posiciones ya ocupadas por los patrones fijos.
- Se mantengan las dimensiones adecuadas según la versión del código QR (en este caso, una matriz de 33x33 para la versión 2).
- Se aplique correctamente el algoritmo de llenado de datos, incluyendo los bits de modo, longitud, datos, relleno y control de errores.

Implementacion correción de errores:

Para garantizar la recuperación de datos en caso de que el código QR esté parcialmente dañado o sucio, se implementó un mecanismo de corrección de errores basado en el algoritmo **Reed-Solomon**, utilizado en el estándar QR.

Este algoritmo trabaja sobre bloques de datos y genera una cantidad adicional de símbolos (bits de corrección) que permiten reconstruir la información original si algunos bits se leen incorrectamente. En esta implementación, se utilizó una versión simplificada adaptada a las necesidades del proyecto: dado un bloque de datos, se genera un bloque de redundancia aplicando operaciones aritméticas en un campo finito (GF(256)), como suma y multiplicación polinomial.

Los pasos principales fueron:

- Se generó una tabla de coeficientes de Reed-Solomon adecuados al nivel de corrección elegido (por defecto, nivel M).
    - La secuencia binaria de datos se dividió en bloques según lo requerido por la versión del QR.
- A cada bloque se le aplicó el algoritmo para generar los **símbolos de corrección**, que fueron añadidos al final de la secuencia de datos original.
- Esta secuencia extendida (datos + corrección) se insertó en la matriz siguiendo el mismo patrón de llenado en zigzag, respetando las posiciones reservadas.

Conclusión:

Con este proyecto se logró generar códigos QR desde cero, entendiendo paso a paso cómo funcionan internamente. Se construyó una matriz válida con todos sus patrones fijos, se codificó una URL en formato binario, se llenó la matriz siguiendo el recorrido correcto y se implementó un sistema de corrección de errores.

- Esto permitió no solo obtener un código QR funcional, sino también aprender en detalle cómo está estructurado y cómo se asegura que pueda ser leído correctamente, incluso si tiene algunos errores.