

---

# CAPÍTULO 1

---

## Implementación

### 1.1. Introducción

En este capítulo se muestra la integración de los conocimientos adquiridos para poder llevar a cabo la realidad aumentada en una aplicación real. Si bien era de gran interés del proyecto la exploración de distintos métodos y algoritmos parecía importante poder poner en práctica todo lo desarrollado en un producto final que pudiera parecerse a un prototipo de aplicación comercial. La aplicación consta de distintas funcionalidades que se describen en este capítulo, tales como:

- (1) QR
- (2) Navegación por listas de cuadros
- (3) Servidor
- (4) Detección SIFT
- (5) Diferentes realidades aumentadas según la obra.

En las próximas secciones se describe más en detalle cada uno de estos puntos y su integración a la aplicación final.

### 1.2. Diagrama global de la aplicación

Para que sea más sencilla la comprensión de los bloques que componen la aplicación a continuación se muestra el *Storyboard* de la aplicación que es la interfaz de usuario y que sirve para visualizar bastante cada una de las clases que intervienen y cómo es el flujo de la aplicación.

#### IMAGEN DEL STORYBOARD COMPLETO

La aplicación cuenta con una pantalla de inicio con instrucciones en formato de audio al presionar el botón respectivo y una presentación sobre cómo es el recorrido. Se da la opción de elegir entonces una forma de recorrer el museo de manera automática, esto es, utilizando QRs para detectar la zona del museo en la que se está, procesamiento de imágenes (SIFT) para identificar el cuadro que se tiene enfrente y comunicación con un servidor que hace de procesador y comunicador de

resultados o una opción manual en la que el usuario es quien elige la zona del museo (autor) y el cuadro en el que está interesado.

Una vez que se está frente al cuadro existe una serie de interacciones habilitadas para el usuario, como ser: audioguía sobre el cuadro en particular, dibujo con lienzo libre para realizar tweets con la imagen dibujada, o la realidad aumentada vinculada a la obra que seleccionó (recorrido manual) o la que fue derivada luego del reconocimiento de imágenes (recorrido automático)

SEGUIR REDACTANDO ESTA SECCION

## 1.3. TableViewController

jjjjj

## 1.4. QR

### 1.4.1. QR. Una realidad

El uso de los identificadores QR (Quick Response), es cada vez más generalizado. Últimamente debido al incremento significativo del uso de *smart devices* el hecho de poder contar con cámara y poder de procesamiento hace que sea frecuente encontrar aplicaciones con el poder de reconocimiento de QRs. Comenzaron a utilizarse en la industria automovilística japonesa como una solución para el trazado en la línea de producción pero su campo de aplicación se ha diversificado y hoy en día se pueden encontrar también como identificatorios de entradas deportivas, tickets de avión, localización geográfica, vínculos a páginas web o en algunos casos también como tarjetas personales.

### 1.4.2. Qué son realmente los QRs?

Se puede decir que los QRs tienen muchos puntos en común con los códigos de barras pero con la ventaja de poder almacenar mucho más información debido a su bidimensionalidad. Existen distintos tipos de QRs, con distintas capacidades de almacenamiento que dependen de la versión, el tipo de datos almacenados y del tipo de corrección de errores. En su versión 40 con detección de errores de nivel L, se pueden almacenar alrededor de 4300 caracteres alfanuméricos o 7000 dígitos (frente a los 20-30 dígitos del código de barras) lo cual lo hace muy flexible para cualquier tipo de aplicación de identificación.

En la figura 1.1 se pueden ver las distintas partes que componen un QR como ser el bloque de control compuesto por las tres esquinas que dan información de la posición, alineamiento y sincronismo, así como también información de versión, formato, corrección de errores y datos. Fuera de toda esa información que podríamos denominar encabezado haciendo analogía con los paquetes de las redes de datos se encuentra la información a almacenar propiamente dicha que conforma el cuerpo del QR.

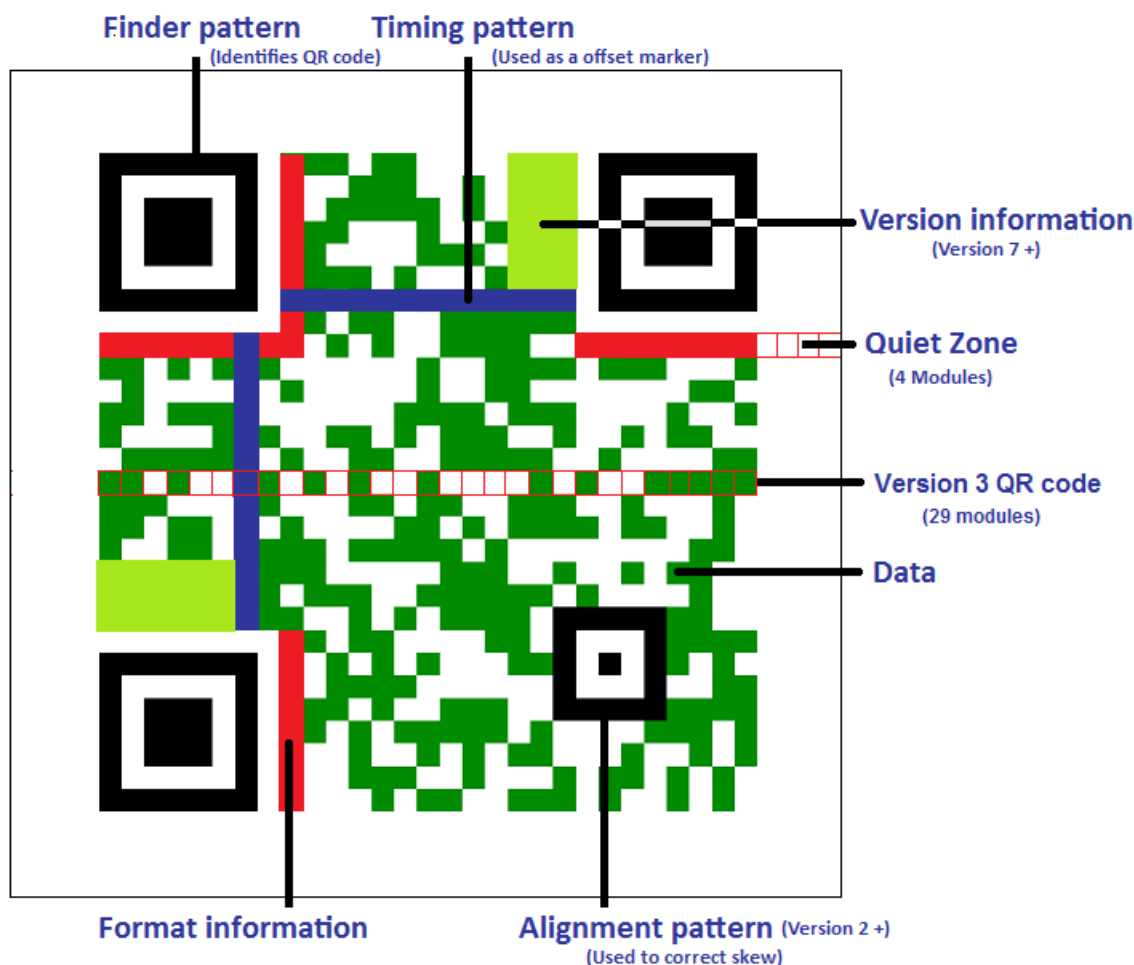


Figura 1.1: Las distintas componentes de un QR. Fuente [?].

### 1.4.3. Codificación y decodificación de QRs

Es fácil darse cuenta que la codificación resulta mucho más sencilla que la decodificación. Para la codificación es necesario comprender el protocolo, las distintas variantes y el tipo de información que se pretende almacenar. Sin embargo para la decodificación, además de tener que cumplir con lo anterior, es necesario contar con buenos sensores y ciertas condiciones de luminosidad y distancia que favorezcan a la cámara y se traduzcan en buenos resultados luego de la detección de errores. Si bien la plataforma es importante para lograr buenos resultados, dada una plataforma, existen variadas aplicaciones tanto para iOS como para Android que cuentan con performances bastante diferentes en función del algoritmo de procesamiento utilizado.

Debido a que el centro del proyecto de fin de carrera no fue la codificación y decodificación de QRs y que además ya existen distintas librerías que resuelven este problema se optó por investigar las distintas variantes e incorporar la más adecuada para la aplicación.

Dentro de todas las librerías que resuelven la decodificación se encuentran ZXing y ZBar como las más destacadas por su popularidad, simplicidad y buena documentación para la fácil implementación. ZXing, denominada así por "Zebra Crossing", es una librería open-source desarrollada en java y que tiene implementaciones que están adaptadas para otros lenguajes como C++, Objective C o JRuby entre otros.

Por su parte ZBar también tiene soporte sobre varios lenguajes y cuenta con un SDK interesante para desarrollar fácilmente aplicaciones que integren el lector de QR. Se trabajó sobre el código de

ejemplo que contiene la implementación de las clases principales para obtener un lector de QRs. Básicamente consta de una clase *ReaderSampleViewController* que hereda de *UIViewController* y que implementa un protocolo llamado *ZBarReaderDelegate*. Al presionarse el botón de detección se crea una instancia de la clase *ReaderSampleViewController* y se presenta la vista de cámara. Luego el protocolo se encarga de la captura y procesamiento del QR teniendo como resultado la información que tiene el QR en la variable denominada *ZBarReaderControllerResults*. Esta variable luego se mapea en una hash table con el contenido en formato *NSDictionary*. De esta manera se accede fácilmente al contenido en formato legible y es fácil de hacer una lógica de comparación y búsqueda en una base de datos.

#### 1.4.4. El QR en la aplicación

Para el caso particular de la aplicación se optó por tener un identificador QR para tres artistas elegidos del Museo Nacional de Artes Visuales (MNAV). Los mismos fueron Pedro Figari, Joaquín Torres García, Juan Manuel Blanes. De esta manera para el caso del recorrido del museo a través de la utilización con QRs es posible determinar la posición del usuario debido a imágenes QR debidamente ubicadas en cada zona. Esto sirve como localización y también sirve para lograr que el paso siguiente, que es la identificación de la obra que el usuario tiene enfrente, sea mediante una búsqueda en una base de datos discriminada por autor. Es decir, si el usuario no escanea el QR la búsqueda de la obra a identificar se hará en una base de datos global del museo, pero para el caso que el usuario sí decida escanear el QR entonces se cuenta con la posibilidad de realizar la búsqueda en una base de datos más reducida.

#### 1.4.5. Arte con QRs

La opción de usar los QRs de una manera distinta ha comenzado a ser notoria en los últimos tiempos. Hay quienes desafían a la información *cruda de 1s y 0s* incorporando imágenes y modificando colores y contornos en los QRs tradicionales para lograr un valor estético además del funcional. A continuación se muestran algunos ejemplos de tales casos en los que claramente se ve cómo puede lograrse el mismo resultado de información con el valor agregado de originalidad.

### 1.5. Servidor

Si bien el desarrollo de la aplicación es un prototipo de una aplicación comercial y para tal caso no se manejan muchas imágenes y otros datos y registros, para lograr escalabilidad se hace imprescindible contar con un servidor. Se pensó con el fin de almacenar toda aquella información relevante en cuanto a registro de obras (imagen, título y autor), descripciones de obras, audioguías, videos, modelos y animaciones para las realidades aumentadas asociadas y cualquier tipo de información que el museo quiera agregar y que por un tema de practicidad no se quiera almacenar dentro de la aplicación. En definitiva, almacenar toda esa información dentro de la aplicación quizá sea rentable para pocas obras, pero lo puede hacer inmanejable para un buen número de obras. Se pensó entonces en la instalación de un servidor que esté ubicado dentro del museo con el cual se tenga una conexión a través de una LAN de (54Mbps). Se aclara este punto pues, en caso de querer hacer un servidor remoto que tenga que ser accedido a través de internet, entonces todo es más lento, aunque funciona perfectamente.

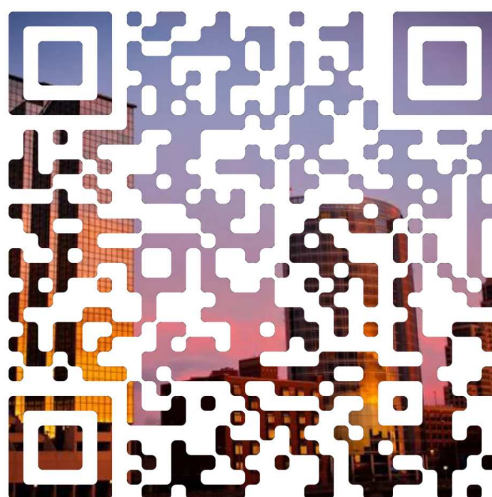


Figura 1.2: Ejemplo de un QR artista. Fuente [?].

### 1.5.1. Creando el servidor

Para la creación del servidor se buscó primeramente la alternativa de hacerlo sobre una máquina con sistema operativo con núcleo Linux, distribución Ubuntu. Luego también se buscó la posibilidad de tener el servidor corriendo sobre una plataforma Unix iOS. Para el segundo caso resultó incluso más sencilla que la primera dado que ya viene pensado por el sistema operativo el hecho de que funcione como servidor. A continuación se explica los pasos que se siguieron para implementar servidores en uno y otro sistema operativo.

#### 1.5.1.1. Servidor iOS

redactar pasos...

#### 1.5.1.2. Servidor LAMP

Se denota servidor LAMP por las siglas de Linux (Sistema Operativo), Apache (Servidor Web), MySQL (Gestor de base de datos), PHP/Perl/Python (lenguaje de programación).

Se instaló entonces el servidor Web Apache, que tiene dentro de sus principales ventajas el hecho de ser multiplataforma, gratis y de código abierto. Para eso desde terminal se debe hacer lo siguiente:

```
sudo apt-get install apache2
```

Con este comando se descarga el paquete *apache2* y se instala. Una vez finalizada la instalación de este paquete ya se cuenta con un servidor y se puede verificar ingresando desde la máquina donde se instaló el servidor abriendo el navegador e ingresando a *http://localhost* o equivalentemente a *http://127.0.0.1* y de esta manera aparece la página por defecto cuyo contenido está dado por el archivo */var/www/index.html*.

Luego de tener instalado el Apache se procede a instalar el php de la siguiente manera

```
sudo apt-get install php5
```

Para el caso particular de los intereses del servidor creado no fue necesario instalar MySQL. Entonces con esto, luego de reiniciar el servidor apache ya se tiene un servidor con intérprete php instalado. A partir de este momento todo se reduce a comprender bien el lenguaje php y poder



Figura 1.3: Impresión de pantalla al ingresar a <http://localhost>.

realizar pequeños módulos de programación que puedan tomar entradas, procesarlas y arrojar una salida.

### 1.5.2. Lenguaje php y principales scripts

Como fue dicho en la sección anterior para los intereses del servidor creado, lo fundamental es el hecho de poder recibir archivos o identificadores de los mismos, poder realizar un procesamiento en el servidor y devolver a la máquina cliente un archivo, mensaje o similar. Para esto se pasa a explicar algunos conceptos básicos de php.

El análogo a un Hello World en php es así:

```
<?php
echo"Hola Mundo";
?>
```

Esto se guarda en un archivo que con extensión .php en la ruta por defecto donde se alojan los php `/var/www/holamundo.php`. De esta manera al ingresar a la dirección `http://localhost/holamundo.php` se ve el print del texto.

como leer un input y hacer algo (ejemplo suma.php)

como hacer un accion.php ....

MOU SIGUE ESTO...

## 1.6. SIFT

## 1.7. Incorporación de la realidad aumentada a la aplicación

asdfasdfsdf

[?].