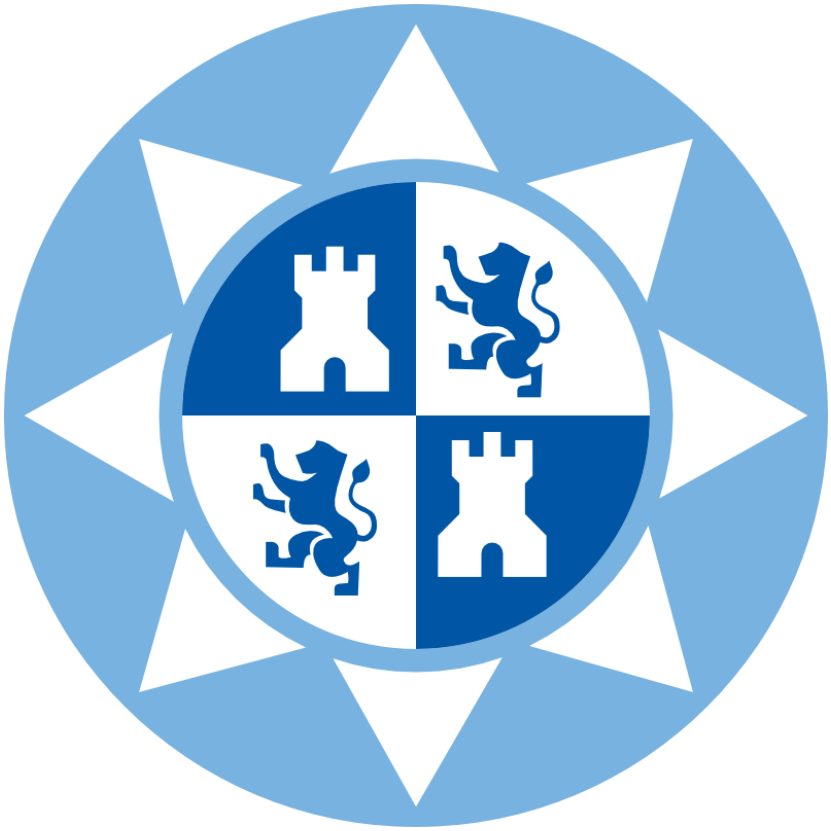
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA



Trabajo Fin de Grado

Tempo. Desarrollo de una herramienta de búsqueda de negocios basada en Android.

Autor: *Josep Nicolás González*

Director: *Esteban Egea López*

UPCT

Julio de 2015

Autor: Josep Nicolás González	**E-mail:** josep.bldz@gmail.com
Director: Esteban Egea López	**E-mail:** esteban.egea@upct.es
Título TFG:	Tempo. Desarrollo de una herramienta de búsqueda de negocios basada en Android.

Resumen:

El objetivo de este proyecto es proporcionar una alternativa eficaz de acceso al mundo de la publicidad y las nuevas tecnologías a todo tipo de negocios. Además, proporcionar un potente buscador de estos negocios para cualquier tipo de usuario. Por ello se desarrolla Tempo.

TEMPO consiste en una herramienta de búsqueda y demostración para los negocios, donde pequeñas y grandes empresas, tiendas de barrio, bares y restaurantes, y cualquier negocio o establecimiento, pueda tener su propia página o "Sitio" con múltiples funcionalidades adaptadas a su sector. Similar a una red social multiplataforma, orientada a Smartphones, debido a su fácil manejo y accesibilidad.

Titulación: Grado en Ingeniería Telemática
Departamento: Departamento TIC
Fecha de presentación: Julio de 2015

Para mis padres, agradecerles tanto con tan poco,
para los que se fueron y para los que confiaron en mí.

Porque los puntos se conectan de adelante hacia atrás.

Índice

Capítulo 1. Introducción

1.1 – Motivación

En estos últimos años, coincidiendo con mi etapa universitaria, ha crecido en mí una pasión por la tecnología y por el diseño de esta, la cual me ha llevado a este punto en el que me encuentro y de la que nace este proyecto.

En los días que vivimos, es fácil sentir interés por la tecnología. Nos encontramos rodeados de tecnología, está presente en todos los campos y avanza a pasos agigantados. No es posible imaginar un mundo sin ella ni tampoco un futuro. Se ha convertido en una fuente inagotable de recursos y herramientas que nos hace la vida más sencilla.

De la búsqueda por unir tecnología, herramientas y diseño, nace Tempo.

1.2 – Objetivos

Tempo es una herramienta ideada con el fin de auto-promocionar cualquier tipo de negocio, empresa o lugar, de ahora en adelante *Sitios*, facilitando la interacción entre estos y sus clientes potenciales, logrando establecer una vía de comunicación entre ellos.

A la hora de enfocar este proyecto, se decide que los objetivos principales debían ser dos.

El primero es el de dar la posibilidad a cualquier negocio, independientemente de su capacidad económica, de poder publicitarse.

A fin de conseguir esto, con Tempo se le brinda la oportunidad de crear un perfil de su Sitio y gestionarlo de la manera que más conveniente le parezca. Es decir, podrán añadir información sobre su negocio, horarios y cualquier otro dato que consideren pertinente. Además, y lo que se cree más interesante, el Sitio quedará geolocalizado en el mapa, de manera que los usuarios podrán hacerse una idea exacta de su ubicación. Para ello, en este proyecto se implementa una funcionalidad para la aplicación de Tempo, en la cual, todo aquel usuario que desee registrar su Sitio, dispondrá de un sistema de registro formado por 3 formularios, con el cual, de una manera sencilla, rápida e intuitiva, el Sitio quedará registrado, y tal como se desarrolla más adelante en la **sección 4.4 – Registro de Sitio** dentro del **Capítulo 4**.

El segundo objetivo es el de ofrecer un potente buscador callejero a toda persona que necesite un servicio en cualquier momento y en cualquier lugar y de manera rápida y eficiente. Los usuarios de Tempo podrán hacer uso del buscador y ver en el mapa los establecimientos más cercanos a su ubicación, acceder a los perfiles e informarse de dichos Sitios, para así poder seleccionar aquellos más afines a sus necesidades y asegurar el éxito de su elección. Para ello, en este proyecto se implementa un sistema de búsqueda filtrada mediante un *Buscador,* además de un sistema de sugerencias para facilitar la búsqueda al usuario, y tal como se desarrolla más adelante en la ***sección 4.5 – Usando el Buscador*** dentro del **Capítulo 4**.

Con estos objetivos se busca conseguir una alternativa eficaz de acceso de un *Sitio* al mundo de la publicidad a través de las posibilidades que ofrece el nuevo mundo de las comunicaciones, obteniendo dos claros servicios diferenciados, por un lado el marketing y la promoción que se consigue de nuestro Sitio, pudiendo atraer a nuevos clientes, y por otro lado, ofrecer un potente buscador a los usuarios de Tempo.

Se decide que Tempo sea una herramienta orientada para su uso en Smartphones, debido a su fácil manejo y accesibilidad, y desarrollado bajo la plataforma de Android.

1.3 – Estructura del documento

Este documento ha sido estructurado en varios capítulos con el fin de agilizar y facilitar al lector el seguimiento y desarrollo del proyecto.

La estructura empleada se detalla a continuación.

En el **Capítulo 2**, se explican las tecnologías principales empleadas para el desarrollo de la aplicación, exponiendo las características principales de estas.

En el **Capítulo 3**, se desarrollan las implementaciones más interesantes llevadas a cabo en el desarrollo de la aplicación, ofreciendo una breve explicación al lector del proceso.

En el **Capítulo 4**, se expone de forma detalla el uso y funcionamiento de la aplicación desarrollada, con el fin de iniciar al lector en el uso de Tempo.

En el **Capítulo 5**, se presentan las conclusiones finales del desarrollo de la aplicación, así como una serie de propuestas para la mejora de Tempo.

Capítulo 2. Tecnologías empleadas

2.1 – Android

Android es una plataforma de código abierto para dispositivos móviles que está basada en Linux y desarrollada por Open Handset Alliance. Fue diseñada principalmente para dispositivos móviles con pantalla táctil, como Smartphones, Tablets… hoy en día podemos encontrar incluso TV's y SmartWatches corriendo bajo este sistema operativo.

Fue desarrollado inicialmente por Android Inc., una empresa que fue comprada por Google en 2005. Más tarde se presentó como el principal producto de un consorcio de fabricantes y desarrolladores de hardware, software y operadores de servicio, denominado como Open Handset Alliance.

Android fue creado para ser independiente de cualquier tipo de arquitectura hardware, además, su portabilidad, flexibilidad y seguridad hacen que sea tan atractivo antes los fabricantes y desarrolladores.

La arquitectura de Android podemos dividirla en los siguientes niveles.

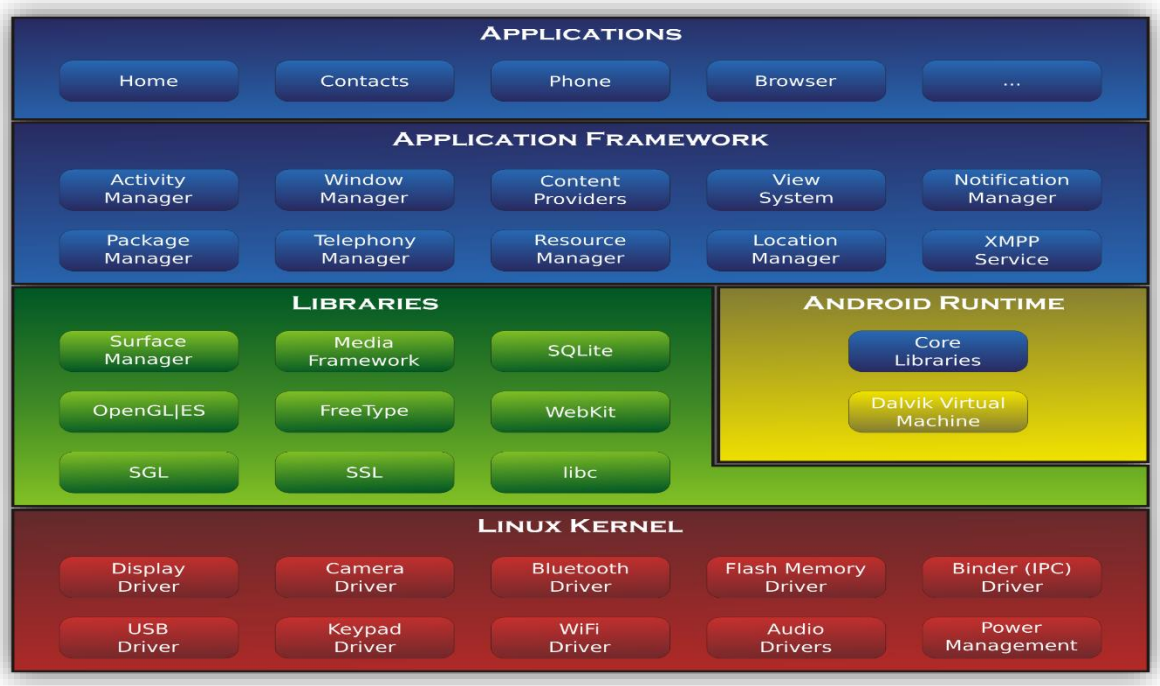


Figura 1 · Arquitectura Android

En el nivel más bajo nos encontramos el **Linux Kernel**, es el núcleo, el motor, el cerebro. Se encarga del control de los drivers, periféricos, Servicios: seguridad, gestión de memoria, procesos, etc. Entre las diferentes versiones de android, el número de Kernel varía. Prácticamente todos están basados en un Kernel Linux 2.6.x (siendo x variable entre actualizaciones), pero a partir de Android 4.0 se usa un Kernel Linux 3.0.x.

Linux es el núcleo o *Kernel* del Sistema Operativo libre denominado GNU/Linux. Lanzado bajo la licencia pública general (GPL – *General Public License*) de GNU y desarrollado gracias a contribuciones provenientes de todo el mundo.

Es la capa encargada de interconectar el hardware del dispositivo con las aplicaciones y todo el software disponible utilizando el hardware del dispositivo.

Figura 2 · Kernel de Linux

El nivel superior se corresponde con las **Librerías**, las bibliotecas nativas de Android, también llamadas librerías. Están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Estas normalmente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma "más eficiente". Entre las librerías incluidas habitualmente encontramos OpenGL (motor gráfico), Bibliotecas multimedia (formatos de audio, imagen y video), Webkit (navegador), SSL (cifrado de comunicaciones), FreeType (fuentes de texto), SQLite (base de datos), entre otras.

Al mismo nivel tenemos **Android Runtime**, este nivel incluye la librería del kernel y la máquina virtual Dalvik, la cual se usa para ejecutar los programas de Java, la VM Dalvik está especialmente mejorada para sistemas embebidos, y sus programas ocupan menor espacio de memoria; al mismo tiempo que es capaz de ejecutar varios programas simultáneamente.

En el siguiente nivel encontramos el **Framework** de aplicaciones, el cual representa el conjunto de herramientas, clases y servicios, que utilizan directamente las aplicaciones para realizar sus funciones.

Todas las aplicaciones utilizan el mismo conjunto de API y el mismo framework representado en el este nivel. La mayoría de los componentes de este nivel son librerías Java que acceden a recurso de capas inferiores para poder re alizar su función.

Por último, la capa superior de **Aplicaciones**, en la cual se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, las nativas (programadas en C o C++) y las administradas (programadas en Java), las que vienen preinstaladas en el dispositivo y aquellas que el usuario ha instalado.

En esta capa encontramos también la aplicación principal del sistema: Inicio (Home) o lanzador (launcher), porque es la que permite ejecutar otras aplicaciones mediante una lista y mostrando diferentes escritorios donde se pueden colocar accesos directos a aplicaciones o incluso widgets, que son también aplicaciones de esta capa.

2.2 – Entorno de desarrollo. Android Studio

Tempo está basado en el sistema operativo y lenguaje de programación de Android. Para desarrollar bajo este lenguaje de programación, existen múltiples entornos de desarrollo integrados o IDEs (Integrated Development Enviroment), entre los más comunes **Eclipse y Android Studio**.

Eclipse es un IDE, de código abierto y multiplataforma, diseñado para ser extendido de forma indefinida a través de plugins. Fue concebido desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. No se basa en ningún lenguaje específico, sino que es un IDE genérico. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones en Java, en el que podemos encontrar todas las herramientas y funciones necesarias para programar, con una interfaz interactiva y agradable al uso. Como se ha indicado, Eclipse permite el desarrollo en cualquier lenguaje de programación a través de plugins. **ADT** es un plugin para el IDE Eclipse cuyas siglas són Android Development Tools, diseñado para dar un potente entorno integrado para la construcción de aplicaciones de Android.

Figura 3 · Logo Eclipse IDE

Figura 4 · Logo ADT

Eclipse era uno de los entornos más utilizados, casi exclusivo, para el desarrollo de aplicaciones Android. Por ello nació **Android Studio**, un IDE exclusivo para el desarrollo Android.

Por las múltiples ventajas que puede ofrecer un entorno de desarrollo exclusivo para una plataforma, y a pesar de estar en continuo desarrollo debido a su reciente creación, se opta en este proyecto por usar este el IDE Android Studio para llevar a cabo Tempo. Esta decisión es tomada con la mente puesta en líneas futuras, ya que se cree que ADT quedará obsoleto para que Android Studio monopolice el desarrollo en Android.

Android Studio es un IDE para el desarrollo de la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O de Google. Basado en el software IDEA JetBrains 'IntelliJ, Android Studio está diseñado para el desarrollo exclusivo de Android. Se encuentra bajo licencia Apache, de forma gratuita. Nació como sustituto de Eclipse con el plugin ADT.

Según datos de la International Data Corporation (IDC), Android consiguió consolidar durante el 2014 su posición dominante en el mercado de sistemas operativos para Smartphones, consiguiendo una espectacular cuota de mercado del 81,5% y superando la barrera de los mil millones de dispositivos vendidos. Con estos datos, Google no ha tenido otra opción que desarrollar un IDE exclusivo para Android, Android Studio.

Figura 5 · Logo AndroidStudio IDE

Android Studio ofrece mejoras sobre el ADT de Eclipse, entre las cuales destacamos las siguientes.

Ofrece un sistema de construcción basado en Gradle, cuyas características principales es la utilización de un DSL(Lenguaje Específico de Dominio) basado en Groovy, sin XMLs complejos como ADT.

Ayudas para la codificación, cuyo editor de código ofrece unas características únicas que facilitan considerablemente el desarrollo. La edición es más fluida, la refactorización es más potente y existe un análisis de código que ayuda a la mejora de este.

Otra de las mejoras que ofrece Android Studio sobre Eclipse con el plugin ADT, es la posibilidad de previsualización de los recursos utilizados en el proyecto o la posibilidad de elección de colores sobre una paleta de colores.

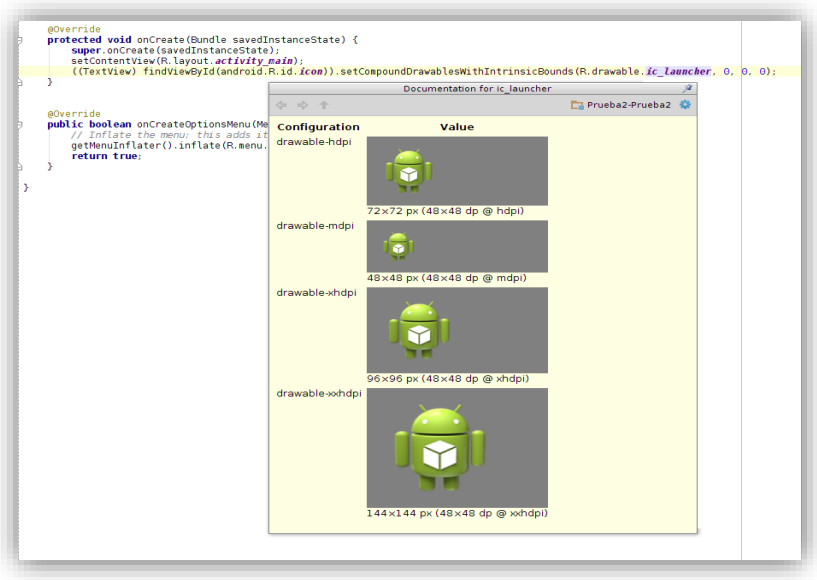


Figura 6 · AndroidStudio – Previsualicación de recursos

Android Studio incluye diálogos para la generación de recursos para distintas configuraciones. Por ejemplo, a la hora de crear un String, Android Studio ofrece la posibilidad de seleccionar el calificador de región y se mostrará una lista de países que podemos ir seleccionando.

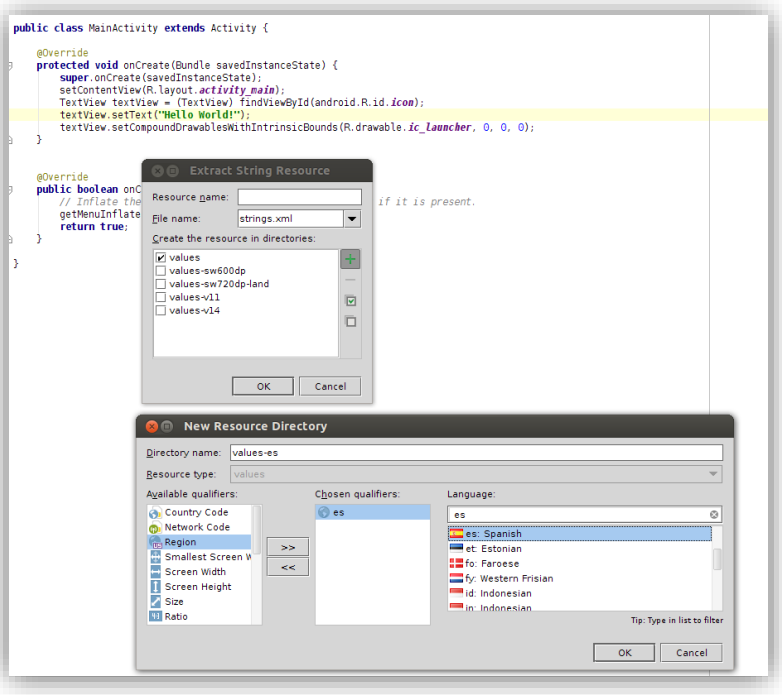


Figura 7 · AndroidStudio – Generación de recursos

Se ofrece un sistema de detección de errores de forma transparente al usuario, que ayuda a la mejora de calidad del código. De forma automática, mientras se escribe, Android Studio va indicando sobre posibles fallos o mejoras de código.

Figura 8 · AndroidStudio – Detección de erores

La refactorización en Android Studio es mucho más potente que en Eclipse, es posible cambiar el nombre de un recurso y automáticamente se cambia el nombre del identificador, o a la inversa funcionaria exactamente igual.

Otra de las ventajas y más interesantes es la herramienta que proporciona para el desarrollo de diseños de Android. La previsualización de los layouts está bastante cuidada, marcando los elementos sobre los que estamos trabajando y permitiendo elegir entre distintos dispositivos reales existentes en el mercado, configuraciones, temas....

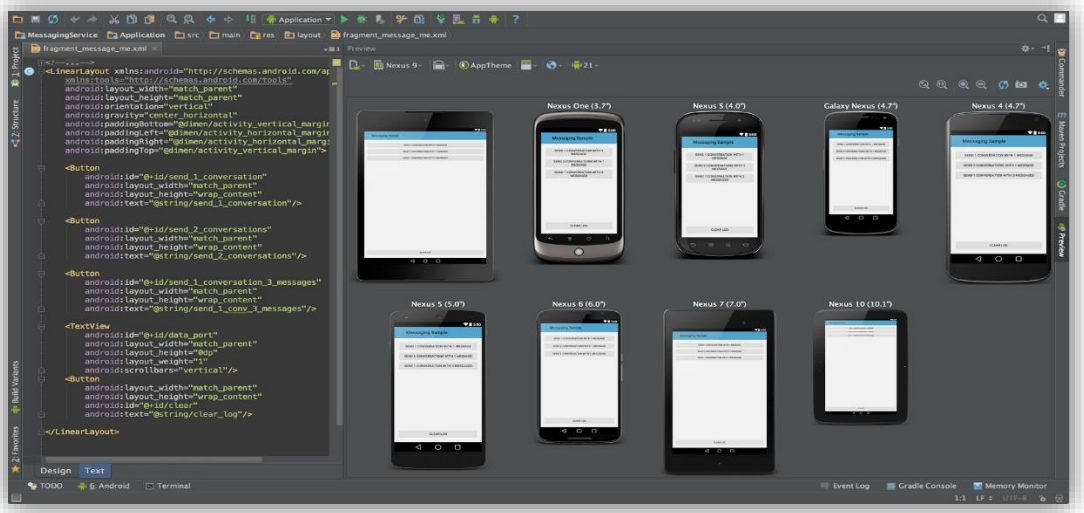


Figura 9 · AndroidStudio – Previsualización de layouts

Además se ha hecho mucho hincapié en la facilitación del uso de los servicios de Google en las aplicaciones. Con Android Studio se quiere facilitar el acceso a todas las herramientas de Google en el IDE.

La ventaja que se considera más destacable es la facilidad para crear distintas versiones de una misma aplicación, por ejemplo, para hacer una distribución multi-apk, para distintos dispositivos, o una versión gratis y otra de pago, o una versión de prueba que carga distintos recursos, apunta a unos webservices ditintos, con estadísticas distintas, etc.

Y entre otras muchas, están la facilitación de reusar código y recursos, la facilitación para la configuración, extensión y personalización del proceso, la búsqueda en la distribución de código para trabajar en equipos, la gestión de dependencias de forma cómoda y potente (basada den Maven) y la posibilidad de compliar desde línea de comando.

2.3 – WEB SERVICES

Los **servicios web** o **web services** son la revolución informática de la nueva generación de aplicaciones que trabajan colaborativamente en las cuales el software está distribuido en diferentes servidores.

La informática se inició con programas monousuarios implantados en grandes ordenadores. Posteriormente estas primeras aplicaciones alcanzaron la capacidad de atender a diferentes usuarios. Pasaron los años y llegó la arquitectura cliente-servidor, que gracias a este modelo de desarrollo, la aplicación se dividía en una parte que interaccionaba con el usuario y otra parte destinada al procesamiento de información. En este acercamiento se consiguió que cada una de las partes que constituían la aplicación pudiera residir en máquinas distintas. Con el paso del tiempo, la computación aumento y llegó la era de las aplicaciones distribuidas en las cuales los procesos se realizaban en diferentes unidades. De este paso surgió la tecnología Internet para solventar las problemáticas asociadas a fallo de aplicación centralizado.

Como punto final a esta cronología, los web services o servicios web, son un paso adelante en la computación ya que de esta forma un ordenador ya no se considerara como un núcleo de cómputo sino como un repositorio de servicios de n aplicaciones distribuidas por internet.

Figura 10 · Arquitectura Cliente-Servidor

El mejor ejemplo para entender el funcionamiento de un **Web Service** es explicar su funcionamiento en Tempo. La aplicación Tempo en Android, para llevar a cabo diferentes fines, necesita en ocasiones, ejecutar u obtener servicios que por ella sola no puede realizar. Por ello, Tempo interactúa con un servidor, el cual le proporciona los servicios deseados atendiendo a las peticiones que la aplicación le realiza. Por ejemplo, cuando un usuario se registra en Tempo, los datos del usuario se almacenan en una base de datos. La aplicación por sí sola no es capaz de llevar a cabo

la inserción de los datos en la base de datos, por ello, manda la información que desea guardar al servidor, el cuál ejecuta unos scripts que procesan la información recibida y se encargan de almacenar dicha información en el lugar deseado.

2.4 – Google Maps

Google Maps es un servidor de aplicaciones de mapas de *Google* que ofrece imágenes vía satélite de todo el planeta, combinadas, en el caso de algunos países, con mapas de sus ciudades, lo que unido a sus posibilidades de programación abierta ha dado lugar a diversas utilidades.

Figura 11 · Logo GoogleMaps

Desde su lanzamiento en febrero de 2005, la aplicación cartográfica de *Google* ha conmocionado a la comunidad de desarrolladores. Si bien sus principios técnicos de base eran ya conocidos, incluso utilizados desde hacía tiempo, la aplicación de *Google* los combina de manera inteligente, y sobre todo ofrece una accesibilidad sin igual.

Para poder hacer uso de esta herramienta en el sistema operativo Android, Google proporciona una API específica para Android. Una API (Application Programming Interface), es una especificación formal sobre cómo un módulo de un software se comunica o interactúa con otro. En otras palabras, las API son un conjunto de comandos, unciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos para ciertos sistemas operativos. Las API simplifican en gran medida el trabajo de un programador, ya que no tiene que programar códigos desde cero.

Google proporciona la Google Maps Android API v2, para poder hacer uso de Google Maps en dispositivos con Android. Esta versión de API fue presentada por Google en diciembre de 2012, y proporciona una fantástica herramienta cuyo uso da

mucho juego en cuanto a aplicaciones.

Con la API de Google Maps para Android, se permite la inserción de mapas basados en Google Maps dentro de una aplicación Android. La API se encarga de automatizar de manera abstracta el acceso a los servidores de Google Maps, la descarga de datos, la visualización del mapa y la respuesta a los eventos sobre el mapa. También permite la posibilidad de agregar marcadores, polígonos y superposiciones de un mapa básico, así como cambiar la vista del usuario de un mapa a diferentes zonas del mapa. Estos objetos proporcionan información adicional para ubicaciones de mapa y permiten la interacción completa entre usuario y mapa. La API permite entre otros, agregar estos gráficos a un mapa:

- Iconos o Markers anclados a posiciones especifícas en el mapa, basados en la longitud y latitud en el mapa.
- Juego de segmentos de líneas.
- Segmentos cerrados (polígonos).
- Gráficos de mapa de bits anclados a posiciones específicas en el mapa.

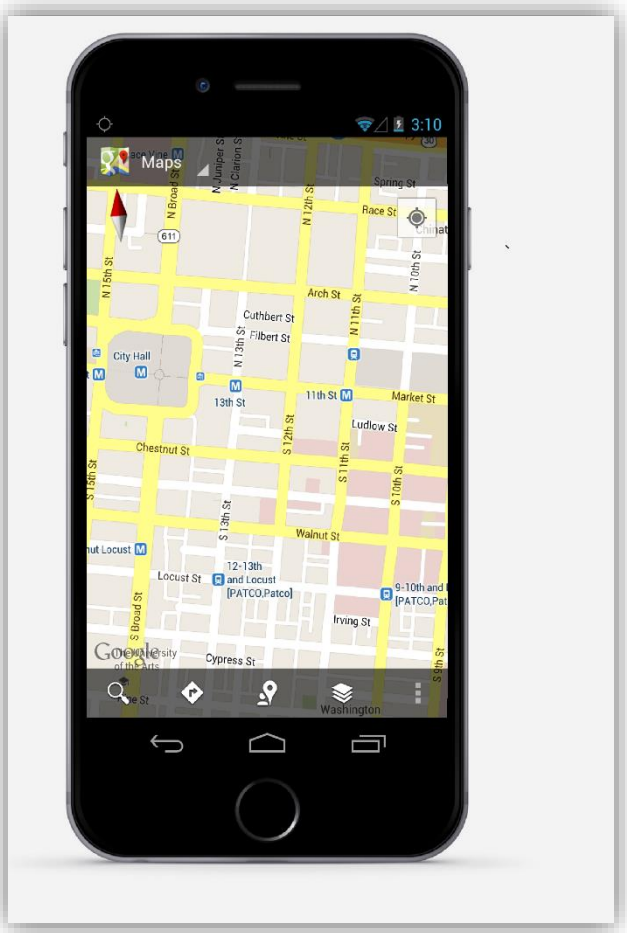


Figura 12 · Visualicación GoogleMaps

Gracias a la API de Google Maps, nuestra aplicación en Android puede disponer de mapas e interactuar con ellos. Pudiendo establecer marcadores, mostrar la ubicación actual del Smartphone o iniciar la navegación desde un punto del mapa a otro.

Google Maps en Android permite tener hasta 4 tipos distintos de mapas:

- Mapa Normal. Es un mapa político. Cuanto mayor es el nivel del zoom en el mapa nos proporciona mayor cantidad de información sobre el sitio.
- Mapa Terreno. Nos ofrece información sobre los desniveles del terreno e información básica.
- Mapa Satélite. Nos permite ver el mundo con imágenes tomadas por satélite.
- Mapa Híbrido. Añade la información del mapa normal al mapa satélite.

En la siguiente imagen podemos observar el mapa de Google Maps del tipo Híbrido en Tempo.

Figura 13 · Vista mapa híbrido

2.5 – Base de datos

Una **Base de Datos** es el conjunto de datos informativos organizados en un mismo contexto para su uso y vinculación.

Se le llama base de datos a los bancos de información que contienen datos relativos a diversas temáticas y categorizados de distinta manera, pero que comparten entre sí algún tipo de vínculo o relación que busca ordenarlos y clasificarlos en conjunto.

Una base de datos puede ser de diverso tipo, desde un pequeño fichero casero para ordenar libros y revistas por clasificación alfabética hasta una compleja base que contenga datos de índole gubernamental en un Estado u organismo internacional. Recientemente, el término base de datos comenzó a utilizarse casi exclusivamente en referencia a bases construidas a partir de software informático, que permiten una más fácil y rápida organización de los datos. Las bases de datos informáticas pueden crearse a partir de software o incluso de forma online usando Internet. En cualquier caso, las funcionalidades disponibles son prácticamente ilimitadas.

Una base de datos o BD (Base de datos), desde el punto de vista informático, es un soporte digital que tiene como fin el almacenamiento masivo de información en formato texto plano. No es capaz de almacenar imágenes, ni almacena otro tipo de datos.

Figura 14 · Arquitectura Base de datos

Podemos clasificar las bases de datos en dos según su variabilidad.

Podemos clasificarla las bases de datos en estáticas, las cuáles son bases de datos únicamente de lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de

un conjunto de datos a través del tiempo, realizar proyecciones, tomar decisiones y realizar análisis de datos para inteligencia empresarial.

Otra clasificación de las bases de datos es clasificarla en **dinámicas**, que son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, borrado y edición de datos, además de las operaciones fundamentales de consulta. Un ejemplo, puede ser la base de datos utilizada en un sistema de información de un supermercado.

Existen programas denominados sistemas gestores de bases de datos, cuyas siglas son DBMS, que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos DBMS, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Para este proyecto se ha decidido trabajar en MySQL como sistema gestor de la base de datos de Tempo.

MySQL es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos.

Figura 15 · Logo MySQL

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interactuación con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

Capítulo 3. Implementación de Tempo

En este capítulo, se explicará de forma detallada los aspectos a destacar sobre la implementación de Tempo, los elementos que lo componen y sus funcionalidades.

La arquitectura o jerarquía de este proyecto se puede clasificar según la siguiente imagen.

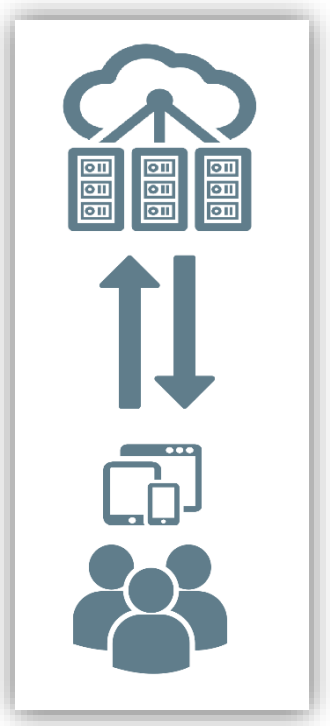


Figura 16 · Arquitectura de Tempo

Como observamos en la figura 16, la jerarquía es muy sencilla, siguiendo una arquitectura *Cliente-Servidor*, los usuarios de Tempo pueden hacer uso de las funciones y herramientas que Tempo les proporciona. Tempo a su vez, requiere en ocasiones la interactuación con una máquina servidora para obtener distintos recursos o realizar ciertas operaciones. A continuación, vamos a explicar con más detalle la arquitectura implementada.

La arquitectura está formada en su escalafón más alto por un **Servidor**, en el cual hay alojado un servidor HTTP Apache y una Base de Datos. Este Servidor se encarga de atender las peticiones que la aplicación de Tempo le realiza, con el fin de realizar alguna operación que la aplicación no es capaz de llevar a cabo. En él, hay almacenados los *Scripts* que llevan a cabo las distintas operaciones que Tempo necesita realizar, así como el acceso y consultas a la Base de Datos. En la figura 17 observamos lo que acabamos de explicar.

Por el escalafón inferior, tenemos los **dispositivos** que tienen instalada la aplicación de Tempo. La aplicación consiste en un conjunto de actividades que realizan diversas funciones, las cuales se desarrollan en la **sección 3.1 – Estructura del proyecto** de este mismo Capítulo 3. Estas actividades son las que, en ocasiones, necesitan de servicios complementarios los cuales son los realizados por el Servidor, por lo que envían peticiones de servicio al Servidor y este las lleva a cabo.

En la figura 17 se observan el proceso de solicitud y respuesta entre los dispositivos y el Servidor.

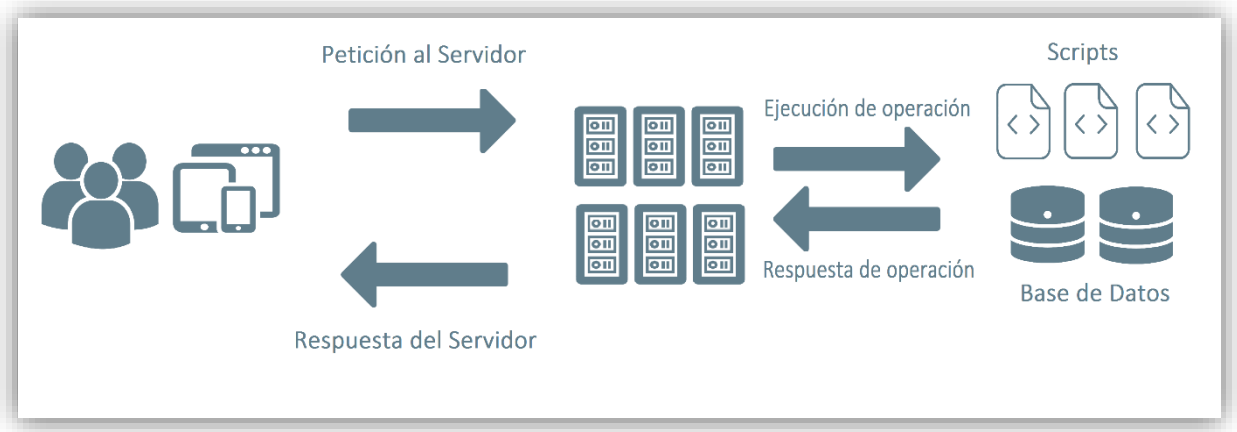


Figura 17 · Arquitectura de Tempo

A continuación procedemos a explicar más detalladamente la implementación y los recursos utilizados para el desarrollo de Tempo.

3.1 – Estructura del proyecto

Como se ha explicado en puntos anteriores, para llevar a cabo Tempo, se ha usado el lenguaje de programación Android y el entorno de desarrollo de Android Studio para trabajar con la plataforma Android. Por ello, se han programado una serie de clases para hacer posible todas las funcionalidades de Tempo. Al llevar a cabo la programación, se ha buscado emplear la mayor efectividad posible con el reúso de métodos y recursos con el fin de tener una estructura de Tempo en cuanto a código lo más simple y nítida.

A continuación observaremos la estructura del proyecto.

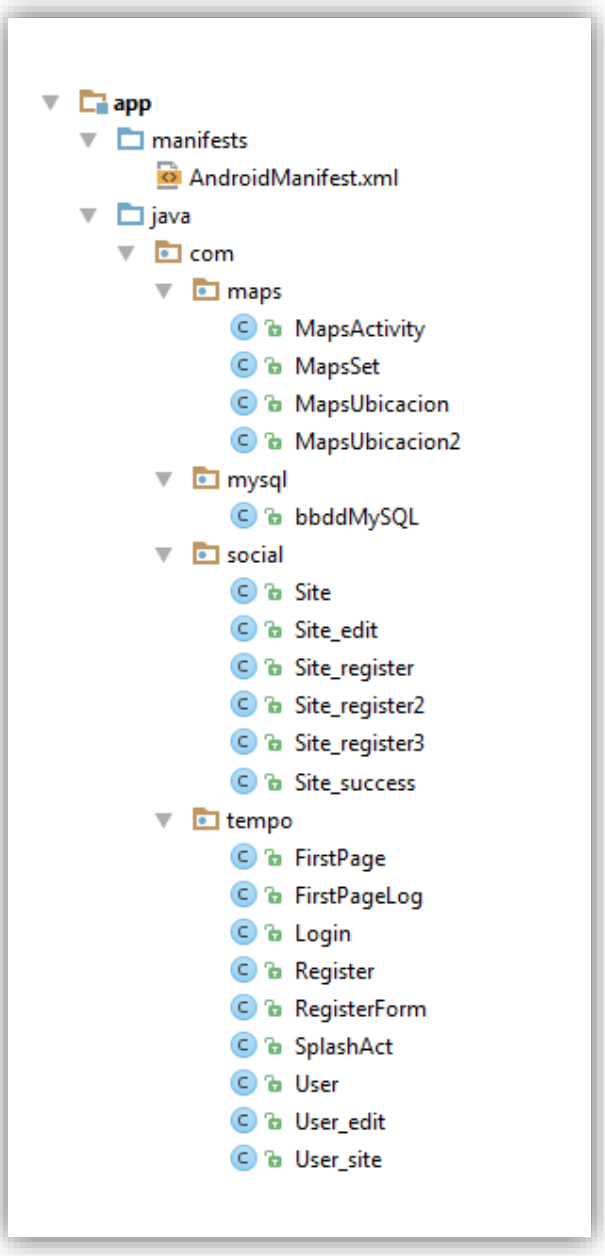


Figura 17 · Estructura de clases

Como podemos observar en la figura 17, se ha tomado la decisión de que la mejor manera de estructurar las clases de código del proyecto sería agrupándolas en 4 carpetas distintas.

Una de las carpetas o agrupación de clases es la de **tempo**, en ella se ha decidido clasificar las clases que guardaran relación con las funcionalidades básicas de la aplicación como por ejemplo, la pantalla principal, registros, pantalla de usuario… Esta agrupación está formada por las siguientes clases:

- ***SplashAct***: Esta clase hace las función de Screen Splash de la aplicación, una pantalla donde se muestra el logo de Tempo y hace las funciones de telón a la aplicación.
- ***FirstPage***: Esta clase implementa las funciones de la página principal de Tempo siempre y cuando el usuario no se haya logueado en la aplicación. Funciones de buscador de Sitios, acceso a mapa con nuestra ubicación exacta y registro/login de usuario, así como acceso a la configuración de la aplicación.
- ***FirstPageLog:*** Al igual que la clase FirstPage, esta clase hace las funciones de página principal de Tempo en el caso de que el usuario este logueado en Tempo. La única diferencia con la clase Firstpage es que proporciona un botón de acceso para la gestión de la cuenta de usuario, en lugar del acceso al registro o logueo de usuario.
- ***Register:*** Esta clase guarda las funciones de puente entre FirstPage y los formularios de registros o logueo.
- ***Login:*** Ofrece las funciones de formulario para introducir los credenciales de usuario y proceder al inicio de sesión en Tempo.
- ***RegisterForm:*** Esta clase ofrece un formulario de registro para crear una cuenta en Tempo.
- ***User:*** Ofrece las funciones de gestión de la cuenta de usuario, pudiendo acceder a las clases User_edit para funciones de modificación de cuento, o User_site para funciones relacionadas con el Sitio del usuario.
- ***User_edit:*** En esta clase podemos modificar los datos personales de la cuenta de usuario.
- ***User_site:*** En ella se implementa las funciones relacionadas de poder acceder a ver el Sitio registrado del usuario, de poder modificar el Sitio o de poder borrarlo.

Siguiendo con la estructuración del proyecto, se procede a justificar la carpeta **social**, en la cual se clasifican las clases que realizan funciones sobre las funcionalidades básicas del Sitio, como por ejemplo, registro del Sitio, modificación de este, acceso a un Sitio… Las clases pertenecientes a esta carpeta son las siguientes:

- ***Site:*** En esta clase se implementan las funciones que podemos encontrar cuando se visualiza un Sitio en Tempo, visualización de toda la información e imágenes del Sitio, posibilidad de valoración del Sitio y acceso a las redes sociales.
- ***Site_edit:*** Esta clase realiza las funciones para poder editar la información del Sitio.
- ***Site_resiter:*** Esta clase ofrece las funciones de primer formulario para llevar a cabo el registro de un Sitio, junto con Site_register2 y Site_register3, forma el formulario completo para registrar un Sitio propio en Tempo. Realiza el almacenado de la información principal del Sitio.
- ***Site_register2:*** Como se ha explicado en el punto anterior, esta clase ofrece el segundo formulario de registro para llevar a cabo el registro de un Sitio. Realiza la subida de imágenes del Sitio al servidor.
- ***Site_register3:*** Al igual que las dos clases anteriores, ofrece el tercer y último formulario de registro para llevar a cabo el registro de un Sitio en Tempo. Realiza el almacenado de información complementaria del Sitio.
- ***Site_succes:*** En esta clase se implementa la función de notificar al usuario que el registro de su Sitio en Tempo se ha llevado a cabo con éxito, y procede a enlazar con la clase Site para previsualizar el Sitio recientemente registrado.

En la carpeta **mysql**, la cual forma parte de la estructura del proyecto, se han clasificado las clases que realizan funciones internas para llevar a cabo la conexión con la base de datos que almacena diferente información tanto de los Sitios en Tempo como de sus usuarios. Sólo contiene una clase, la cual es la única clase funcional del proyecto, todas las demás clases se entienden como *Activities* de Android. Formada por la clase siguiente:

- ***bbddMySQL:*** Esta clase implementa las funciones para llevar a cabo una correcta conexión con la base de datos de Tempo, tanto como para almacenar datos como para obtener datos.

Por último, la carpeta **maps** contiene las clases relacionadas con las funciones de GoogleMaps, como localización del usuario de Tempo o la ubicación de los Sitios. Las clases pertenecientes a esta carpeta son las siguientes:

- ***MapsActivity:*** Esta clase realiza las funciones de proceder a la búsqueda de la localización exacta del usuario en el mapa, así como de ubicar los Sitios registrados en Tempo más cercanos a la ubicación del usuario.
- ***MapsUbicacion:*** En ella se implementa la ubicación de un solo Sitio en el Mapa.
- ***MapsUbicacion2:*** Realiza la misma función que la clase *MapsUbicacion*, salvo que esta se usa para ubicar más de un Sitio.
- ***MapsSet:*** En esta clase se guaran las funciones que permiten al usuario establecer la ubicación de su Sitio cuando procede al registro de este en el formulario de registro implementado en la clase *Site_register*.

3.2 – Interacción con Base de Datos

Tempo necesita almacenar diferentes datos para poder ofrecer al usuario diferentes funciones y servicios. Para ello, la aplicación hace uso de una *Base de Datos*, en la que se almacena, entre otros, los datos relacionados con la información de los Sitios y la información de la cuenta de usuario.

Para el almacenamiento de estos datos, se ha llegado a la conclusión que, con el fin de alcanzar la mayor optimización y simplicidad posible, la clasificación y almacenamiento de los datos se debía llevar a cabo en las siguientes tablas:

- users_tempo
- sitios_tempo
- horarios_tempo
- valoraciones_tempo

En la tabla **users_tempo**, se almacenan la información relacionada con las cuentas de usuario de los usuarios de Tempo. A continuación observamos en la figura 18 la estructura de la tabla.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	**id_user**	int(4)			No	*Ninguna*	AUTO_INCREMENT
2	**username**	varchar(10)	latin1_swedish_ci		No	*Ninguna*	
3	**password**	varchar(40)	latin1_swedish_ci		No	*Ninguna*	
4	**email**	varchar(25)	latin1_swedish_ci		No	*Ninguna*	
5	**sitio**	varchar(1)	latin1_swedish_ci		No	*Ninguna*	

Figura 18 · Tabla users_tempo

Esta tabla contiene el identificador de usuario, así como el nombre de usuario y su contraseña, el correo electrónico y un campo sitio que identifica si el usuario tiene un Sitio registrado en Tempo.

La siguiente tabla es **sitios_tempo**, en ella se lleva a cabo el almacenamiento de todos los campos de información de un Sitio registrado en Tempo. En la figura 19 observamos la estructura de la tabla.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
1	id_sitio	int(4)			No	*Ninguna*	AUTO_INCREMENT
2	id_user	int(4)			No	*Ninguna*	
3	tipo	varchar(1)	utf8_spanish_ci		No	*Ninguna*	
4	categoria	varchar(30)	utf8_spanish_ci		No	*Ninguna*	
5	categoria_tag	varchar(30)	utf8_spanish_ci		No	*Ninguna*	
6	ciudad	varchar(30)	utf8_spanish_ci		No	*Ninguna*	
7	nombre	varchar(30)	utf8_spanish_ci		No	*Ninguna*	
8	latitud	varchar(30)	latin1_swedish_ci		No	*Ninguna*	
9	longitud	varchar(30)	latin1_swedish_ci		No	*Ninguna*	
10	sitio_web	varchar(30)	utf8_spanish_ci		No	*Ninguna*	
11	telefono	varchar(12)	utf8_spanish_ci		No	*Ninguna*	
12	imagenes	varchar(1)	utf8_spanish_ci		No	*Ninguna*	
13	facebook	varchar(20)	utf8_spanish_ci		Sí	*NULL*	
14	twitter	varchar(20)	utf8_spanish_ci		Sí	*NULL*	
15	instagram	varchar(20)	utf8_spanish_ci		Sí	*NULL*	
16	info	varchar(400)	utf8_spanish_ci		No	*Ninguna*	

Figura 19 · Tabla sitios_tempo

Esta tabla contiene el identificador del Sitio así como el usuario que ha registrado el Sitio, información del Sitio como su nombre, categoría, tipo... información sobre la dirección, redes sociales, contacto o información adicional del Sitio.

La siguiente tabla que forma parte de la base de datos de Tempo, es **horarios_tempo**. Esta tabla es una tabla adicional a la tabla sitios_tempo, ya que almacena los horarios de un Sitio. Observamos en la figura 20 la estructura de esta tabla.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	**id_sitio**	int(4)			No	*Ninguna*
2	**lunes**	varchar(23)	latin1_swedish_ci		No	*Ninguna*
3	**martes**	varchar(23)	latin1_swedish_ci		No	*Ninguna*
4	**miercoles**	varchar(23)	latin1_swedish_ci		No	*Ninguna*
5	**jueves**	varchar(23)	latin1_swedish_ci		No	*Ninguna*
6	**viernes**	varchar(23)	latin1_swedish_ci		No	*Ninguna*
7	**sabado**	varchar(23)	latin1_swedish_ci		No	*Ninguna*
8	**domingo**	varchar(23)	latin1_swedish_ci		No	*Ninguna*

Figura 20 · Tabla horarios_tempo

Como se observa en la figura anterior, la principal función de esta tabla es el almacenamiento del horario por cada día de la semana junto al identificador del Sitio al que pertenece el horario.

La tabla **valoraciones_tempo**, almacena las puntuaciones y comentarios llevados a cabo por los usuarios sobre un Sitio. Observamos en la figura 21 la estructura de la tabla.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	**id_sitio**	int(4)			No	*Ninguna*
2	**id_user**	int(4)			No	*Ninguna*
3	**valoracion**	varchar(5)	utf8_spanish_ci		No	*Ninguna*
4	**palabra**	varchar(20)	utf8_spanish_ci		No	*Ninguna*
5	**comentario**	varchar(120)	utf8_spanish_ci		No	*Ninguna*

Figura 21 · Tabla valoraciones_tempo

Esta tabla relaciona el identificador del Sitio con el usuario que llevó a cabo la valoración, y almacena la puntuación, la palabra que describe mejor al sitio y el comentario sobre el sitio.

Para llevar a cabo el almacenamiento de la información en la base de datos, la aplicación hace uso de scripts, los cuales ejecuta en la máquina servidora y estos se encargan de recibir los datos y almacenarlos. Las funciones principales de estos

Scripts es el almacenamiento y obtención de datos de la base de datos.

Un **Script** es un programa usualmente simple, que almacena un conjunto de instrucciones generalmente almacenadas en un archivo de texto y que deben ser interpretadas línea a línea en tiempo real para su ejecución, esto último es lo que los distingue de los programas compilados. En nuestra aplicación, los Scripts realizan funciones que la aplicación de Tempo solicita y para lo cual no está configurada, como por ejemplo el almacenamiento de datos o la obtención de datos de la base de datos. Los Scripts están almacenados en la máquina servidora la cual se encarga de ejecutarlos cuando la aplicación de Tempo manda una petición al servidor para la ejecución de un Script.

Estos Scripts están programados en lenguaje PHP debido a su flexibilidad, potencia y alto rendimiento que ofrece. PHP es el acrónimo recursivo de *Hypertext Preprocessor*, es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. Puede interactuar con muchos motores de bases de datos tales como *MySQL, MS SQL, Oracle, Informix, PostgreSQL*, y otros muchos.

Figura 22 · Logo PHP

Un ejemplo de implementación de un Script programado en lenguaje PHP y del que hace uso Tempo es cuando un usuario se da de alta en Tempo. Para ello, Tempo necesita almacenar en la base de datos en la tabla *users_tempo*, diferentes campos de información los cuales hemos explicado anteriormente. Estos campos de información son los siguientes:

- Identificador de usuario.
- Nombre de usuario.
- Contraseña.
- Correo electrónico.
- Identificador de Sitio.

Para llevar a cabo el almacenamiento de esta información en la tabla users_tempo, la aplicación Tempo se encarga de mandar la información al Servidor. Como hemos explicado en el punto anterior, la clase que se encarga de llevar a cabo el registro de un usuario en tempo es la clase *RegisterForm.java*, la cual ofrece un formulario de registro al usuario, el cual ha de introducir un nombre de usuario, una contraseña y el correo electrónico.

Veamos un fragmento de la implementación de dicha clase para enviar la información a almacenar al servidor.

```
public boolean loginstatus() {
     /*Inicialización de variable logstatus*/
    logstatus=-1;

    /*Codificación de la password de usuario antes de enviarla al servidor*/
    computeSHAHash(EditPass.getText().toString());

    /*Creación de ArrayList para el almacenamiento de los campos de información a enviar.*/
    ArrayList<NameValuePair> postparameters2send= new ArrayList<NameValuePair>();

    /*Almacenamiento de la información a enviar al servidor en el ArrayList que hemos creado*/
    postparameters2send.add(new BasicNameValuePair("username",EditUser.getText().toString()));
    postparameters2send.add(new BasicNameValuePair("password",pw));
    postparameters2send.add(new BasicNameValuePair("email",EditMail.getText().toString()));

    /*Invocación del método getserverdata(parametros a enviar, url de conexión), del objeto post de la clase bbddMySQL.*/
    /*Creación de un JSONArray para la obtención del resultado*/
    JSONArray jdata=post.getserverdata(postparameters2send, URL_connect);

    /*Comprobacion y procesamiento del resultado obtenido*/
    if (jdata!=null && jdata.length() > 0){
        JSONObject json_data;
        try {
            json_data = jdata.getJSONObject(0);
            logstatus=json_data.getInt("logstatus");
            Log.e("loginstatus", "logstatus= " + logstatus);
        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        if (logstatus==0){
            Log.e("loginstatus ", "invalido");
            return false;
        }
        else{
            Log.e("loginstatus ", "valido");
            return true;
        }
    }else{
        Log.e("JSON  ", "ERROR");
        return false;
    }
}
```

Figura 23 · Implementación RegisterForm.java

Observamos cómo esta clase sólo envía el nombre de usuario, la contraseña y el correo electrónico. Los campos de información de identificador de usuario e identificador del Sitio, no es información que deba enviar está clase, ya que por un lado, el identificador de usuario es un valor auto incremental de la base de datos, es decir, conforme se van añadiendo usuarios se incrementa el valor, y por otro lado, el

identificador de Sitio se inicializa siempre con valor o. Los campos de información son enviados para la ejecución y procesamiento a la siguiente URL.

```
private String URL_connect="http://"+IP_Server+"/BBDDTempo/adduser.php";
```

Figura 24 · URL script adduser.php

Con lo cual, se envía la información a un Script *adduser.php*, alojado en el servidor cuya ruta es la que se observa en la figura 24. Este Script recibe los parámetros y procesa el almacenamiento en la base de datos de la forma que observamos en la siguiente figura.

```php
<?php

    /*Recogida de parámetros*/
    $usuario = $_POST['username'];
    $passw = $_POST['password'];
    $email = $_POST['email'];

    /*
    *Llamada a script con funciones, llamado funciones_bbdd.php
    */
    require_once 'funciones_bd.php';
    $db = new funciones_bd();

    /*
    *Comprobamos si existe el usuario mediante la función isuserexist().
    */
    /*Si el usuario existe, se envía un logstatus = 0*/
    if($db->isuserexist($usuario)){
        $resultado[]=array("logstatus"=>"0");
    }else{ /*Si el usuario no existe, se llama a la función adduser para almacenar en la BD.*/
        if($db->adduser($usuario,$passw,$email)){
                if($user_id = $db->checkID($usuario)){
                    $resultado[]=array("logstatus"=> $user_id);
                }else{
                    $resultado[]=array("logstatus"=>"0");
                }
        }
    }
    echo json_encode($resultado);

?>
```

Figura 25 · Script adduser.php

Este Script *adduser.php*, en primer lugar, recibe los parámetros mandados por la clase RegisterForm.java mediante el método POST. Una vez recibidos y almacenados los campos de información a almacenar en la base de datos, comprueba mediante la función isuserexist(), si el nombre de usuario ya está registrado en la base de datos de Tempo. Esta función es la que observamos en la siguiente imagen.

```php
    /**
     * Verificar si el usuario ya existe por el username
     */
    public function isuserexist($username) {

        /*Creación de la consulta a la BD.*/
        $result = mysql_query("SELECT username from users_tempo WHERE username = '$username'");

        /*Comprobación de obtención de resultado.*/
        $num_rows = mysql_num_rows($result);

        /*Si se obtiene algún resultado, es que el usuario existe.*/
        if ($num_rows > 0) {
            return true;
        } else { /*Si no se obtiene resultado alguno, es que el usuario no existe*/
            return false;
        }
    }
```

Figura 26 · Función isuserexist

Tras llevar a cabo la comprobación, si el nombre de usuario ya existe en la base de datos de *users_tempo*, entonces el Script devolverá un *logstatus* = *o* a la clase *RegisterForm.java*, de lo contrario, procederá a almacenar el nuevo usuario invocando a la función *adduser()*, la cual recibe los parámetros a almacenar. Observamos esta función en la siguiente figura.

```php
    /**
     * Añadir un nuevo usuario a la BD.
     */
    public function adduser($username, $password, $email) {

        /*Creación de la consulta para llevar a cabo el almacenamiento de un nuevo usuario en la BD.*/
        $result = mysql_query("INSERT INTO users_tempo(username, password, email,sitio) VALUES('$username', '$password', '$email', '0')");

        /*Si el almacenamiento se ha llevado a cabo de forma correcta devuelve true.*/
        if ($result) {
            return true;
        } else {/*Si el almacenamiento no se ha llevado a cabo devuelve false.*/
            return false;
        }

    }
```

Figura 27 · Función adduser

La función *adduser()* lleva a cabo el almacenamiento mediante una consulta SQL de la información del usuario en la base de datos de Tempo. Observamos como para el campo *sitio* de la tabla *users_tempo*, se asigna un valor o, debido a que se inicializa siempre con ese valor. Si el almacenamiento se realiza con éxito, el Script adduser.php incovado por la clase *RegisterForm.java*, habrá realizado con éxito el registro de un nuevo usuario en Tempo, y devolverá a la clase java el identificador de usuario. Si por el contrario no se lleva a cabo el almacenamiento con éxito, entonces

el Script *adduser.php* devolverá a RegisterForm.java un indicador de fallo.

Hasta ahora se ha hablado sobre la interacción de los Scripts con la aplicación para llevar a cabo el almacenamiento de información en la base de datos. Además, se ha explicado que en caso de fallo en el registro de un usuario, el Script **adduser.php** devuelve un campo logstatus con valor 0, o en caso de llevar a cabo el registro de un nuevo usuario con éxito, devuelve un campo logstatus con valor igual al identificador del nuevo usuario registrado. Es decir, existe un intercambio de información entre el servidor y la aplicación.

Tempo, además de almacenar datos en la base de datos, necesita la obtención de información almacenada en la base de datos con el fin de poder ofrecer sus servicios a los usuarios. Esta obtención de información, o intercambio de información, es llevado a cabo entre el servidor y la aplicación mediante los Scripts en lenguaje PHP, al igual que el almacenamiento de información en la base de datos.

Siguiendo con el ejemplo mostrado en la figura 25, la última línea de código del script corresponde al envío de un resultado o información necesaria desde el servidor mediante el Script hasta la aplicación. Observamos el código de envío del resultado del Script adduser.php en la siguiente figura.

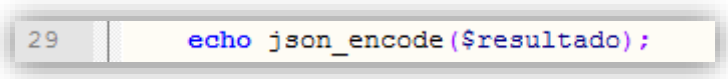


Figura 28 · JSON en Script

La información es enviada desde el servidor a la aplicación en formato JSON.

JSON, del acrónimo JavaScript Object Notation, es un formato para el intercambios de datos, básicamente JSON describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. JSON nació como una alternativa a XML. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.

Figura 29 · Logo JSON

El principio básico de una cadena JSON, es con pares atributo-valor, éstos deben estar encerrados entre llaves { , } que es lo que definen el inicio y el fin del objeto. Un ejemplo de cadena JSON, sería el mostrado en la figura siguiente.

```
{"Fruteria":
 [
  {"Fruta":
   [
    {"Nombre":"Manzana","Cantidad":10},
    {"Nombre":"Pera","Cantidad":20},
    {"Nombre":"Naranja","Cantidad":30}
   ]
  },
  {"Verdura":
   [
    {"Nombre":"Lechuga","Cantidad":80},
    {"Nombre":"Tomate","Cantidad":15},
    {"Nombre":"Pepino","Cantidad":50}
   ]
  }
 ]
}
```

Figura 30 · Estructura JSON

La cadena Frutería tiene a su vez dos objetos, Fruta y Verdura, que a su vez tienen información sobre cada objeto, como el nombre y la cantidad, formando una cadena JSON.

Tempo utiliza el formato de codificación JSON para la obtención y posterior procesamiento de información almacenada en su servidor.

Como ejemplo, una de las implementaciones que se realizan con JSON es en la funcionalidad de Buscador que ofrece Tempo. Tal y como se detalla en la **sección 3.3 – Implementación del Buscador** de este mismo capítulo, el usuario puede realizar búsqueda de Sitios mediante el buscador. Cuando el usuario realiza una búsqueda, el texto clave introducido por el usuario en el buscador es enviado por un método a un Script alojado en el servidor. Este Script se encarga de recoger toda la información almacenada de un Sitio y devolverla al método. Para ello, la información es codificada en formato JSON. Observamos la siguiente figura.

```
while($query_row = mysql_fetch_array($result)){
    $id_sitio = utf8_encode($query_row['id_sitio']);
    $id_user = utf8_encode($query_row['id_user']);
    $categoria = utf8_encode($query_row['categoria']);
    $ciudad = utf8_encode($query_row['ciudad']);
    $nombre = utf8_encode($query_row['nombre']);
    $latitud = utf8_encode($query_row['latitud']);
    $longitud = utf8_encode($query_row['longitud']);
    $facebook = utf8_encode($query_row['facebook']);
    $twitter = utf8_encode($query_row['twitter']);
    $instagram = utf8_encode($query_row['instagram']);
    $lunes = utf8_encode($query_row['lunes']);
    $martes = utf8_encode($query_row['martes']);
    $miercoles = utf8_encode($query_row['miercoles']);
    $jueves = utf8_encode($query_row['jueves']);
    $viernes = utf8_encode($query_row['viernes']);
    $sabado = utf8_encode($query_row['sabado']);
    $domingo = utf8_encode($query_row['domingo']);
    $telefono = utf8_encode($query_row['telefono']);
    $web = utf8_encode($query_row['sitio_web']);
    $imagenes = utf8_encode($query_row['imagenes']);
    $info = utf8_encode($query_row['info']);
    $data[$i]= array('id_sitio' => $id_sitio,'id_user' => $id_user, 'categoria' => $categoria, 'ciudad' =>$ciudad,
            'nombre' => $nombre, 'latitud' => $latitud, 'longitud' => $longitud, 'facebook' => $facebook, 'twitter' => $twitter,
            'instagram' => $instagram, 'lunes' => $lunes, 'martes' => $martes, 'miercoles' => $miercoles, 'jueves' => $jueves, 'viernes' => $viernes,
            'sabado' => $sabado, 'domingo' => $domingo, 'web' => $web, 'telefono'=> $telefono, 'imagenes' => $imagenes, 'info' => $info  );
    $i++;
}
return $data;
```

Figura 31 · Script de obtención de Sitio

La figura 31 es un fragmento del código del Script que lleva a cabo la recogida de información de un Sitio. Este Script almacena toda la información en una variable del tipo array denominada *$data*. Esta variable se envía como respuesta a la petición codificada en formato JSON de la manera que se observa en la siguiente figura.

```
json_encode($data)
```

Figura 32 · Codificación JSON

Una vez la respuesta ha sido codificada y enviada, esta será recibida por el método que envió la petición y llevará a cabo el proceso necesario. En el caso que exponíamos anteriormente, el método envía la petición de obtener la información de un Sitio, si la petición se ha llevado a cabo con éxito, entonces este recibirá la respuesta codificada en formato JSON. Una vez obtenida la respuesta, el proceso para la obtención de los campos de información contenidos en el array en formato JSON que ha enviado el servidor, es el que observamos en la siguiente figura.

```
public class FiltroSugerencias extends AsyncTask<String, String, String> {

    protected String doInBackground(String... params) {
        sugg.clear();
        post = new bbddMySQL();
        filtro = Busqueda.getQuery().toString();
        ArrayList<NameValuePair> postparameters2send = new ArrayList<~>();
        postparameters2send.add(new BasicNameValuePair("filtro", filtro));
        jdata = post.getserverdata(postparameters2send, URL_connect);

        return "ok";
    }

    protected void onPostExecute(String result) {

        if (jdata != null && jdata.length() > 0) {

            try {
                /*Creamos un objeto JSON*/
                JSONObject row;
                /*Obtenemos el elemento de la posición 0 del array recibido y almacenamos en la variable creada.*/
                row = (JSONObject) jdata.get(0);
                /*Obtenemos del elemento 0 del JSONArray obtenido, el campo de información que contiene como
                * identificador la palabra nombre. Tras obtenerlo lo almacenamos en la lista sugg.*/
                sugg.add(row.get("nombre").toString());

            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }

    }
}
```

Figura 33 · Método FiltroSugerencias

Como observamos en la figura 33, se recibe un objeto en formato JSON que se almacena en la variable *jdata* del tipo JSONArray. Tras comprobar que se ha obtenido una respuesta y que la variable *jdata* no está vacía, extraemos el primer elemento del array, en el cual está la cadena de datos con toda la información referente al sitio. Un JSONArray, como hemos explicado anteriormente, es una array en formato JSON, y como array, puede almacenar en distintas posiciones elementos del tipo cadenas de texto. En este caso, sólo nos interesa el elemento de la posición 0, que es dónde está almacenada la información que nos interesa. Una vez extraído el elemento 0 del JSONArray y después de haberlo almacenado en la variable *row* del tipo JSONObject, podremos extraer los campos de información que nos interesen mediante su identificador.

Tal y como vemos en la figura 33, se extrae el campo de información cuyo identificador es la palabra nombre, el cual coincide con el nombre del Sitio, es decir, estamos extrayendo el nombre del Sitio guardado bajo el identificador *nombre*. En la siguiente figura vemos de forma detallada como se lleva a cabo la extracción de un campo de información concreto.

```
row.get("nombre").toString();
```

Figura 34 · Extracción en JSON

3.3 – Implementación del Buscador

Como más adelante se desarrolla en la **sección 4.5** en el **Capítulo 4**, Tempo ofrece un sistema de búsquedas de Sitios mediante palabras clave, además de un servicio de sugerencias en tiempo real.

Figura 35 · Buscador

La funcionalidad e implementación de este sistema la vamos a detallar a continuación.

En primer lugar, y como observamos en la figura 35, el usuario dispone de un buscador en el que introducir el Sitio o Sitios a buscar. Este buscador esta implementado por el componente *SearchView* de Android, el cual dispone de un capturador de eventos, que por un lado recoge un evento de pulsación cuando el usuario pulsa el botón en forma de lupa ubicado a la derecha del *SearchView* que hace las funciones de *Submit*, y por otro lado recoge un evento cuando se modifica el texto existente en el SearchView. Es decir, siempre que el usuario pulsa el botón de lupa o modifica el texto existente, el SearchView capturará un evento y realizará la acción que se le indique. En la siguiente figura observamos la implementación del SearchView en la clase FirstPage.

```
/*Declaración del componente SearchView*/
Busqueda = (SearchView) findViewById(R.id.fpl_searchView);
/*Manejador de eventos del componente SearchView*/
Busqueda.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    //Evento pulsación de botón Submit.
    @Override
    public boolean onQueryTextSubmit(String query) {
        if (Busqueda.getQuery().toString().equals("")) {
            sugg.clear();
            adapter.notifyDataSetChanged();
        }else{
            new FiltroSugerencias().execute();
            submit=true;
        }
        return false;
    }
    //Evento modificación de texto.
    @Override
    public boolean onQueryTextChange(String newText) {
        if (Busqueda.getQuery().toString().equals("")) {
            sugg.clear();
            adapter.notifyDataSetChanged();

        }else{
            new FiltroSugerencias().execute();
        }
        return false;
    }
});
```

Figura 36 · Implementación SearchView

Como se observa en la figura 36, cuando existe un evento de pulsación de botón *Submit* o un evento de modificación del texto del SearchView, se invoca a una función **FiltroSugerencias()**, la cual recoge el texto existente en el buscador y lo envía a la siguiente dirección del servidor que observamos en la siguiente figura.

```
private String URL_connect = "http://" + IP_Server + "/BBDDTempo/sugerencias.php";
```

Figura 37 · URL script sugerencias.php

Esta dirección, es la que aloja un Script que realiza las funciones de búsquedas de Sitios filtrando los nombres de estos con el texto introducido en el buscador.

La función *FiltroSugerencias()* la observamos en la siguiente figura.

```
public class FiltroSugerencias extends AsyncTask<String, String, String> {

    protected String doInBackground(String... params) {
        sugg.clear();
        post = new bbddMySQL();
        filtro = Busqueda.getQuery().toString();
        ArrayList<NameValuePair> postparameters2send = new ArrayList<~>();
        postparameters2send.add(new BasicNameValuePair("filtro", filtro));
        jdata = post.getserverdata(postparameters2send, URL_connect);
        return "ok";
    }
    protected void onPostExecute(String result) {
        if (jdata != null && jdata.length() > 0) {
            try {
                JSONObject row;
                row = (JSONObject) jdata.get(0);
                sugg.add(row.get("nombre").toString());
                row = (JSONObject) jdata.get(1);
                sugg.add(row.get("nombre").toString());
                row = (JSONObject) jdata.get(2);
                sugg.add(row.get("nombre").toString());
                row = (JSONObject) jdata.get(3);
                sugg.add(row.get("nombre").toString());
                row = (JSONObject) jdata.get(4);
                sugg.add(row.get("nombre").toString());

            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

Figura 38 · Método FiltroSugerencias

Esta función, como se ha indicado anteriormente, recoge el texto que el usuario ha introducido en el Buscador, y lo envía para el procesamiento en el Servidor por el Script *sugerencias.php*, el cual se especifica en la dirección que observábamos en la figura 37. Este Script se encarga de consultar a la Base de Datos de Tempo, los Sitios que coincidan en el nombre con el texto introducido por el usuario en el Buscador. A continuación observamos en la siguiente figura el script *sugerencias.php*.

```php
<?php

$palabra = $_POST['filtro'];


require_once 'funciones_bd.php';
$db = new funciones_bd();

$array = $db->suggestion($palabra);

    if(empty($array)){
        echo json_encode($array = array("nombre"=>"0"));
    }else{
        echo json_encode($array);
    }

return $array;

?>
```

Figura 39 · Script sugerencias.php

Una vez llevado a cabo la consulta por parte del Script, este devolverá a la aplicación los resultados de la búsqueda en la Base de Datos en formato JSON, desarrollado en la **sección 3.2 – Interacción con Base de Datos** en este mismo capítulo. Estos resultados consistirán en la información completa de aquellos Sitios que coincidan con el texto introducido por el usuario en el buscador. Los resultados son devueltos a la función FiltroSugerencias(), la cual se encarga de la recepción de la información así como su posterior procesado para mostrar las sugerencias o mostrar un Sitio.

3.4 – Archivo AndroidManifest

Situado en la raíz de las aplicaciones Android como AndroidManifest.xml, es un archivo de configuración en lenguaje XML donde podemos aplicar las configuraciones básicas de nuestra aplicación. Su configuración puede realizarse a través de una interfaz gráfica, pero es recomendable conocer la sintaxis ya que en muchas ocasiones será más fácil y rápido poder hacerlo manualmente desde el propio archivo. El AndroidManifest está situado en la raíz de cada aplicación y contiene todos los componentes que aparecen en la aplicación Android, actividades, servicios, permisos... En él se puede encontrar la versión mínima requerida de Android para mover la aplicación, permisos necesarios de la aplicación, estilos, icono de la aplicación, actividad principal, entre mucho más.

En Tempo, se han especificado, entre otros menos importantes, las siguientes configuraciones de la aplicación.

En primer lugar, se ha establecido las versiones máxima y mínimas sobre la API de Android que puede ejecutar la aplicación.

```
<uses-sdk
    android:minSdkVersion="15"
    android:targetSdkVersion="21" />
```

Figura 40 · AndroidManifest - versiones

En segundo lugar, se han especificado los permisos requeridos para el correcto funcionamiento de la aplicación.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="cookingbread.googlemapstest.permission.MAPS_RECEIVE" />
```

Figura 41 · AndroidManifest - Permisos

Observamos en la figura 41, que se le ha dado a la aplicación diferentes permisos de acceso a internet, de acceso a la cámara y permisos para GoogleMaps entre otros.

Además se han establecido las actividades pertenecientes a la aplicación, como observamos en la siguiente figura.

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name=".SplashAct"
        android:label="@string/app_name"
        android:screenOrientation="portrait" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".FirstPage"
        android:screenOrientation="portrait" />
    <activity
        android:name=".Register"
        android:screenOrientation="portrait" />
    <activity
        android:name=".Login"
        android:screenOrientation="portrait" />
    <activity
        android:name=".RegisterForm"
        android:screenOrientation="portrait" />
```

Figura 42 · AndroidManifest - Actividades

Como se observa en la figura 42, se ha establecido la actividad inicial de la aplicación a SplashAct.java, y se ha añadido las demás actividades pertenecientes a la aplicación.

3.5 – Implementación GoogleMaps

En el **Capítulo 2**, en la ***sección 2.4- Google Maps***, se ha hablado sobre GoogleMaps.

Para poder implementar **GoogleMaps** en una aplicación Android, en primer lugar hay que tener instalados los *Google Play Services*, los cuales nos proporcionan lo necesarios para poder trabajar con la API v2 de GoogleMaps.

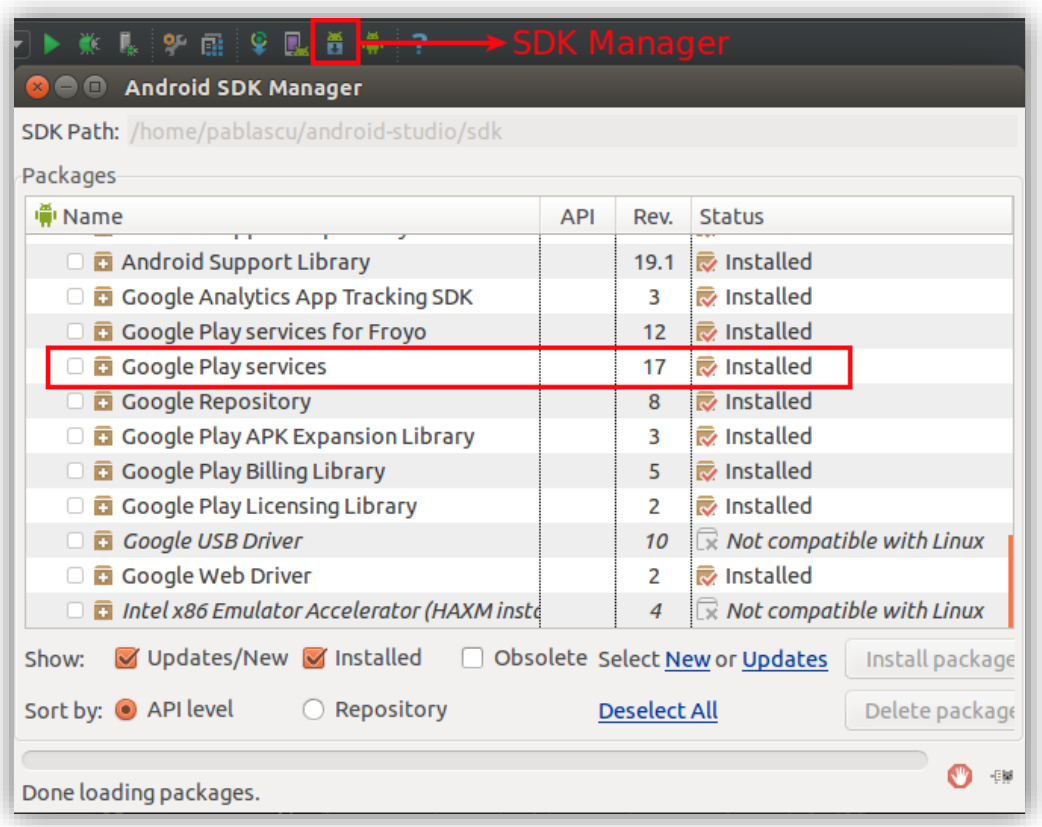


Figura 43 · GoogleMaps – Google Play Services

A continuación, debemos crear un proyecto en la Consola de desarrolladores de Google. Introducimos la información que se solicita.

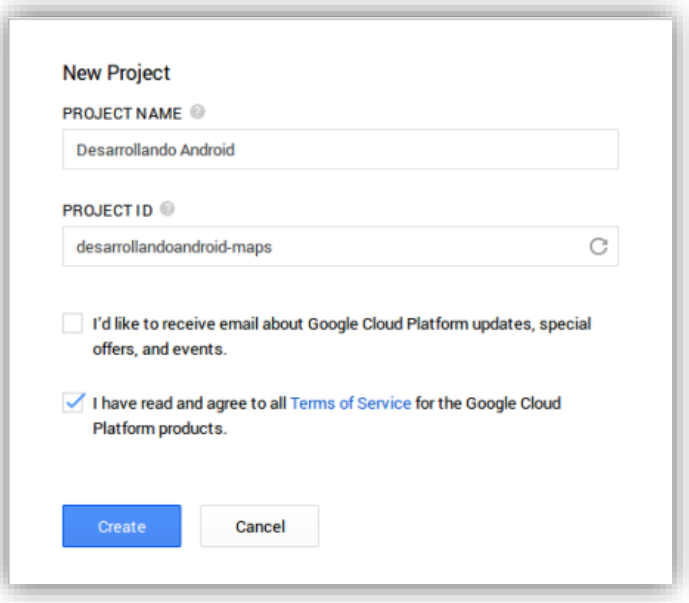


Figura 44 · GoogleMaps – Creación proyecto

Una vez creado el proyecto, debemos habilitar las siguientes API's:

- Google Maps API v2.
- Google Maps Coordinate API.
- Google Maps Engine API.
- Google Maps Geolocation API.
- Translate API.

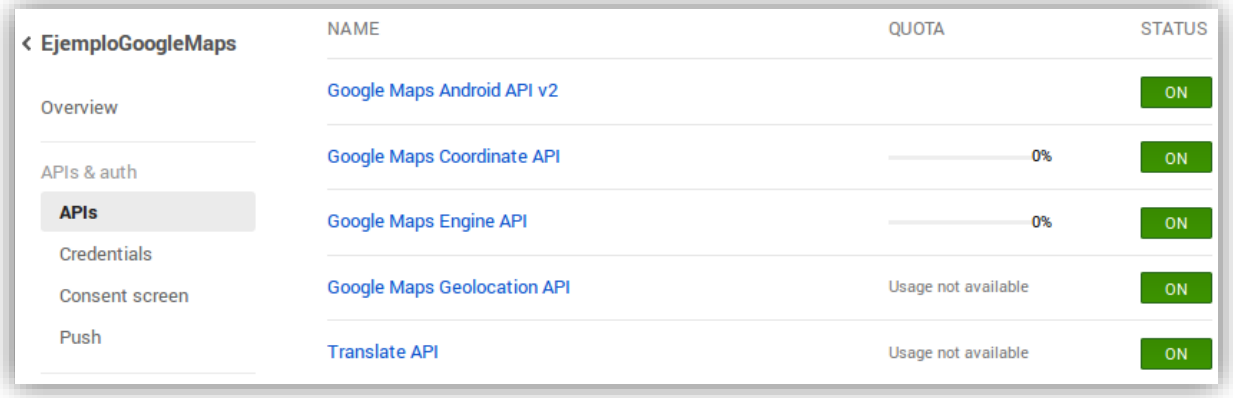


Figura 45 · GoogleMaps – API's

Una vez habilitadas las API's, debemos crear una nueva Key de Google. Para ello, debemos introducir la clave SHA1 de nuestra clave keystore con la que firmamos nuestra aplicación e introducir el nombre del paquete de la aplicación.

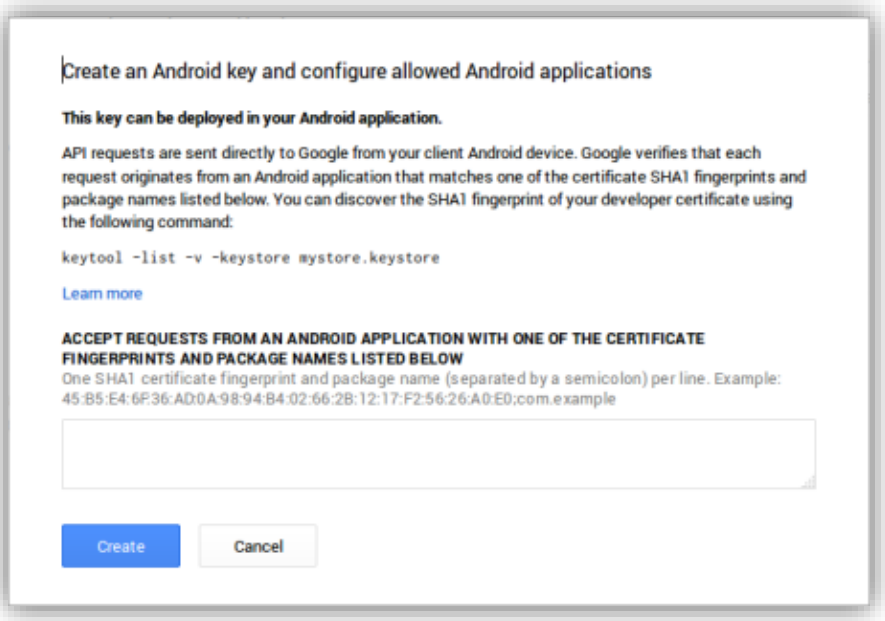


Figura 46 · GoogleMaps – Creación Key

Una vez creada la Key de google con nuestra clave SHA1 y el nombre de nuestro paquete, se generará la Key de Google. Esta key será la que necesitemos para habilitar los servicios de GoogleMaps en nuestra aplicación. Por ello debemos copiarla en nuestro fichero AndroidManifest.xml.

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="YOUR_KEY_HERE" />
```

Figura 47 · GoogleMaps – Key en AndroidManifest

Además debemos de darle los siguientes permisos a la aplicación.

```
<uses-feature
    android:glEsVersion="0x00020000"
    android:required="true"/>
```

Figura 48 · GoogleMaps – Permisos AndroidManifest

Una vez realizados los pasos anteriores, el servicio de **GoogleMaps** estará disponible para su implementación.

Como ejemplo, una de las implementaciones del servicio de GoogleMaps llevadas a cabo en este proyecto, se ha realizado en la clase MapsUbicación.java, la cual hemos desarrollado en la **sección 3.1 – Estructura del proyecto** en este mismo capítulo. Esta clase se encargaba de situar la ubicación exacta de un Sitio en el Mapa, en el caso de que un usuario llevara a cabo una búsqueda de un Sitio en el buscador, el cual tras filtrar y seleccionar el Sitio que el usuario buscaba, nos llevaba a un Mapa con la dirección exacta del Sitio.

Esto es posible, en primer lugar, a que la persona que registró el Sitio, tuvo que realizar de manera exacta la ubicación de este durante el proceso de registro, obteniéndose así las coordenadas basadas en la longitud y latitud de la ubicación del Sitio. Gracias a conocer estas coordenadas, es muy sencillo mostrar al usuario la ubicación de un Sitio en la clase **MapsUbicacion.java**.

Para ello, desde la clase **Firstpage.java** o **FirstPageLog.java**, las cuales ambas llevan implementado el servicio buscador explicado en el punto anterior, se iniciaba la transición a la nueva actividad de la clase *MapsUbicación.java*, cuando el usuario buscaba un Sitio, con el fin de mostrar el Mapa y el punto exacto donde se encuentra el Sitio. Tanto la clase *FirstPage* como la clase *FirstPageLog*, llevaban a cabo el paso de datos en formato JSON, con toda la información del Sitio, entre las que se encuentran la longitud y latitud del Sitio, tal y como observamos en la figura siguiente.

```
/*Iniciamos actividad MapsUbicación*/
Intent intent = new Intent(FirstPage.this, MapsUbicacion.class);
/*Pasamos información entre actividades. En este caso un JSONArray con la información del Sitio*/
intent.putExtra("json", jdata.toString());
intent.putExtra("i",-1);
/*Inicio de la actividad MapsUbicación*/
startActivity(intent);
```

Figura 49 · GoogleMaps – Inicio de actividades

Tras esto, la actividad *MapsUbicación* se inicia e invocando a una función llamada **getJSONArray()** la cual se encarga de recoger los datos enviados por la actividad anterior, con el fin de obtener el JSONArray con la información del Sitio. Observamos en la dos siguientes figuras la invocación del método y el método.

```
/*Recogida de datos de la actividad anterior*/
getJSONArray();
```

Figura 50 · GoogleMaps – Paso de información

```java
/*Método getJSONArray()*/
private void getJSONArray() {
    try {
        /*Recogida de parámetros de la actividad anterior*/
        int i = getIntent().getExtras().getInt("i");
        sitioJArray = new JSONArray(getIntent().getStringExtra("json"));
        if (i == -1) {

            /*Obtencion y almacenado de la información deseada almacenada en el Array JSON.*/
            sitioObj = sitioJArray.getJSONObject(0);

            latitud = sitioObj.getString("latitud");
            longitud = sitioObj.getString("longitud");
            nombre = sitioObj.getString("nombre");
            categoria = sitioObj.getString("categoria");
            markers = true;
        }
    } catch (JSONException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Figura 51 · GoogleMaps – Método getJSONArray

Como observamos en la figura 51, esta clase obtiene del array en formato JSON enviado por la actividad anterior, las coordenadas de latitud y longitud del Sitio. Con esto, se invoca a un método propio con el fin de establecer un Marker en el Mapa con la ubicación exacta del Sitio. El método setMarker, se encargar de establecer la posición de un marcador, su título, su información, el diseño del marcador, su tamaño y su opacidad. En la figura siguiente observamos el método setMarker.

```java
/*Método setMarker*/
private void setMarker(LatLng position, String title, String info, float opacity, float dimension1, float dimension2, int icon) {
    mMap.addMarker(new MarkerOptions()
            .position(position)
            .title(title)
            .snippet(info)
            .alpha(opacity)
            .anchor(dimension1, dimension2)
            .draggable(true)
            .icon(BitmapDescriptorFactory.fromResource(icon)));
}
```

Figura 52 · GoogleMaps – Método setMarker

Tras llevar a cabo estos pasos, el usuario podrá visualizar en el mapa un marcador con la ubicación exacta que deseemos, en este caso, la ubicación del Sitio.

Capítulo 4. Paseando por Tempo

Cómo se ha descrito previamente, **Tempo** es una herramienta que ofrece dos servicios claramente diferenciados. Por un lado, se ofrece la posibilidad de poder publicitar el Sitio de un usuario, y por otro lado, se puede utilizar como un potente buscador callejero de Sitios.

En este apartado, vamos a llevar a cabo una introducción al funcionamiento de Tempo desde el punto de vista de un usuario.

4.1 – Instalación

Para proceder a la instalación y hacer uso de Tempo, es indispensable poseer de un Smartphone que corra bajo el sistema operativo Android. Podemos llevar a cabo la instalación de la aplicación optando por dos alternativas distintas.

La primera alternativa es instalando el *APK* de la aplicación. La palabra APK son las siglas de Android Aplication Package, y hacen referencia a un archivo para el sistema operativo Android, con extensión .apk, una variante del formato JAR de Java, y se usa para distribuir e instalar componentes empaquetados para la plataforma Android. Es decir, es un archivo que hace referencia a un tipo de formato para archivos Android, que permite la instalación de aplicaciones en un dispositivo con el sistema operativo de Android. El archivo APK contiene una serie de archivos de la aplicación Android, entre otros, el AndroidManifest.xml, Clases.dex, Resources.arsc...

La instalación de un fichero APK es un proceso sencillo y rápido, ya que Android se encarga de instalar automáticamente los archivos con esta extensión, con una mínima interacción del usuario.

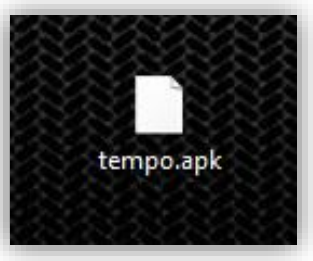


Figura 53 · APK de Tempo

La segunda opción, es usando el servicio *Play Store* que proporciona Android, el cual consiste en una plataforma de distribución digital de aplicaciones móviles para el propio Sistema Operativo. Aquí los usuarios pueden buscar e instalar de manera

sencilla las aplicaciones que deseen sin necesidad de obtener previamente APK de la aplicación.

El usuario, usando el buscador que proporciona esta plataforma, puede encontrar la aplicación de Tempo, pulsar sobre el botón instalar, y como anteriormente hemos explicado, y al igual que en el caso del archivo de instalación APK, la instalación se realizará de forma automática, gracias a la funcionalidad que proporciona Android, y con una mínima interacción por parte del usuario.

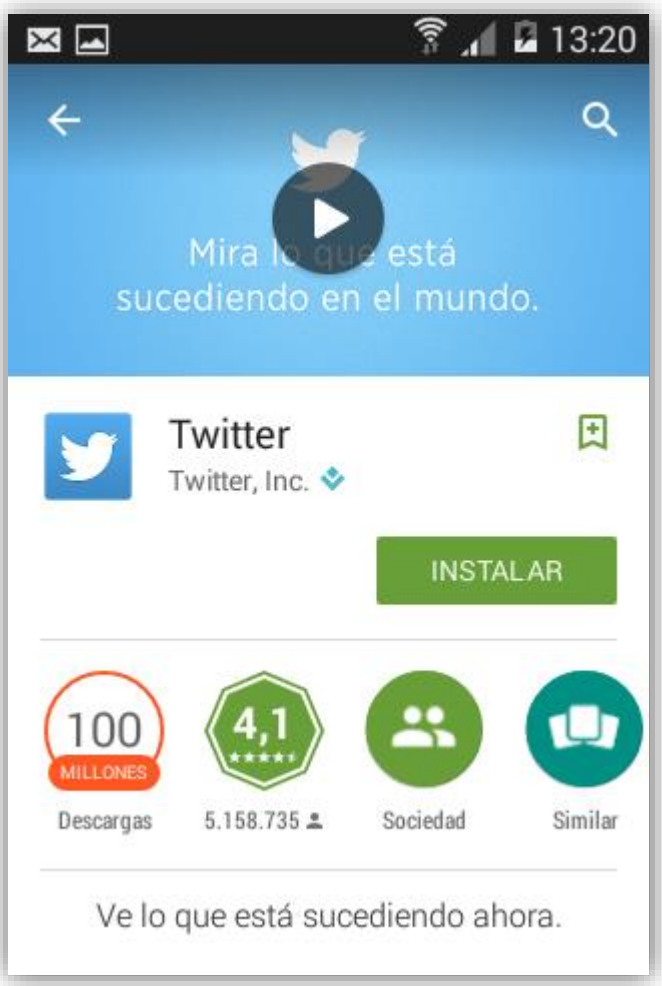


Figura 54 · Play Store de Android

Una vez hemos finalizado la instalación, podremos visualizar el icono de Tempo en nuestro dispositivo, el cual hace las funciones de acceso directo a nuestra aplicación. Pulsando sobre el icono, accederemos a la aplicación y podremos hacer uso y disfrute de Tempo.

Figura 55 · Icono de Tempo

4.2 – Primeros pasos

Lo primero que nos encontramos al iniciar la aplicación de Tempo es una *Splash Screen* de la aplicación, con una duración de 3 segundos, y que hace las funciones de telón de nuestra aplicación.

Figura 56 · Tempo - Splash Screen

Tras haber finalizado el Splash, accederemos a la pantalla principal de nuestra aplicación. La pantalla principal es la fuente de todos los recursos que nos proporciona Tempo. En ella podemos, entre otras funcionalidades menos importantes, hacer uso del Buscador de Sitios y acceder a registrar/administrar nuestro Sitio, en el caso de tenerlo.

Figura 57 · Tempo - Pantalla Principal

Como apreciamos en la figura 57, en esta pantalla principal, disponemos de un buscador, con el que podemos buscar el servicio que necesitemos en cualquier momento dado y en cualquier lugar, y el cuál será descrito más adelante en un apartado independiente. Además, disponemos de una serie de botones, situados en la parte inferior de la pantalla.

Figura 58 · Tempo – Botones Pantalla principal

El primero de los botones que nos encontramos, es el botón de **Inicia Sesión**. Este botón nos llevará a una pantalla en la cual podremos iniciar sesión en Tempo, mediante nuestra cuenta de usuario previamente registrada o, en el caso de no tener todavía una cuenta en Tempo, nos proporcionará la posibilidad de crearnos una cuenta. Más adelante, se explicará en un apartado independiente, el proceso recientemente descrito.

El segundo botón que se ofrece, es el botón de **Tu ubicación**, pulsando sobre él, accederemos a un mapa donde se nos mostrará nuestra ubicación exacta, y además, los Sitios que tengamos en un radio cercano a nuestra ubicación. Pudiendo visualizar los Sitios que tengamos justo a nuestro alrededor, hacernos una idea exacta de su ubicación y poder visitar su perfil, con el fin de satisfacer cualquier interés o necesidad.

Figura 59 · Tempo – Ubicación

Por último, el botón **Configuración,** con el cual podremos acceder a configurar la aplicación.

4.3 – Registro en Tempo

Tempo ofrece la posibilidad al usuario de disponer de una cuenta en Tempo y disfrutar de las funcionalidades que poseen los usuarios registrados, destacando entre muchas, la posibilidad de poder registrar y administrar tu Sitio en Tempo, con el fin de que cualquier usuario de Tempo disponga de la posibilidad de encontrar tu Sitio, así como de acceder al perfil de este, potenciando la publicidad de tu Sitio, y haciéndolo llegar al alcance de personas que, de otras maneras, no sería posible. Además, sólo los usuarios registrados pueden puntuar, valorar y comentar los Sitios.

El registro en Tempo, se realiza de una manera intuitiva y eficaz, sin informaciones irrelevantes ni formularios de registros pesados. Para proceder a iniciar sesión en Tempo, debemos acceder mediante el botón ***Inicia Sesión*** que se dispone en la pantalla principal de la aplicación, previamente explicada en el punto anterior, y que observábamos en la figura 45. Tras pulsar sobre él, accederemos a la pantalla de Inicia Sesión.

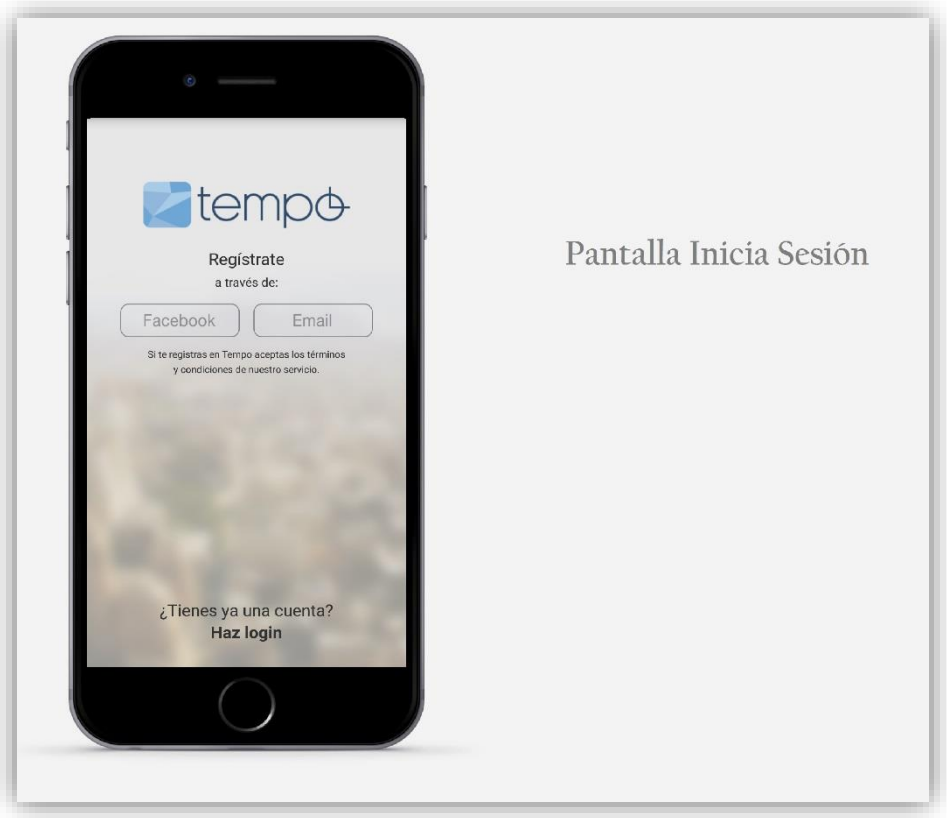


Figura 60 · Tempo – Inicia Sesión

En esta pantalla, se da la posibilidad al usuario de registrarse en la comunidad de Tempo, a través de dos alternativas, mediante la cuenta de Facebook del usuario o

mediante su email. Cabe decir que se ofrece dos alternativas adaptadas a los tiempos y efectivas para el registro en Tempo, buscando siempre la comodidad para el usuario.

En el caso de poseer ya una cuenta de Tempo, se da la opción al usuario de llevar a cabo un login mediante sus credenciales de usuario, pulsando sobre las palabras ***Haz login***, que se aprecian en la parte inferior de la pantalla de Inicio de Sesión. Esta opción nos lleva a una Pantalla de Login, en la cual podemos de manera sencilla, introducir nuestro nombre de usuario en Tempo y nuestra contraseña.

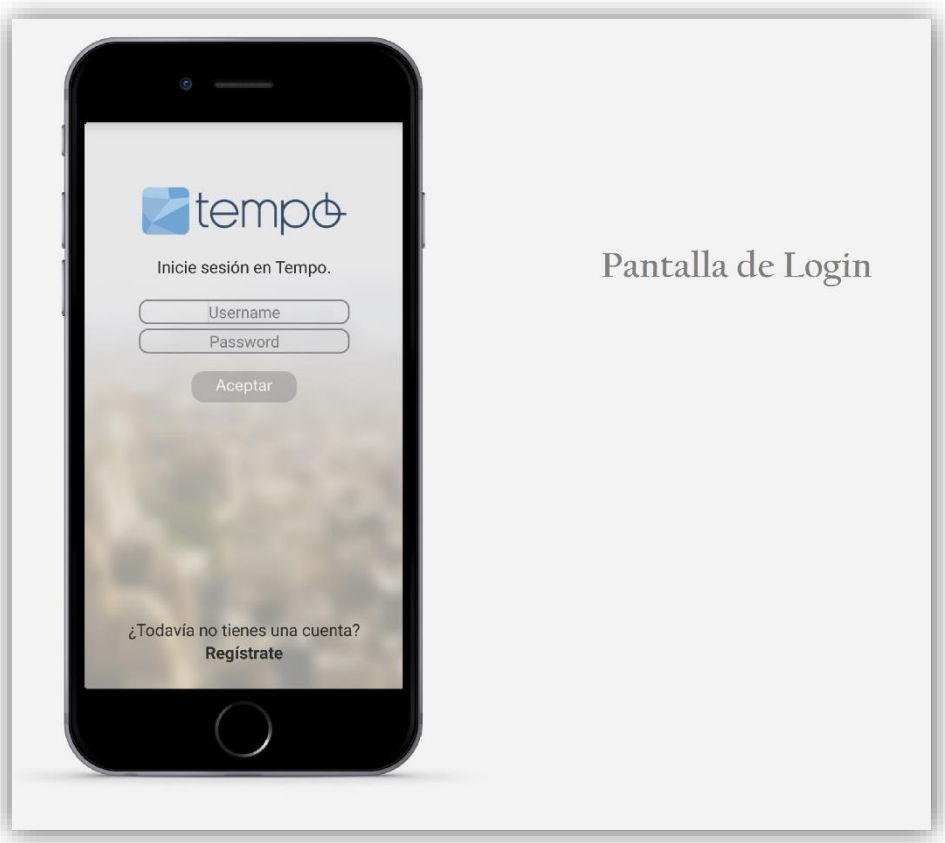


Figura 61 · Tempo - Login

Volviendo a la figura 61, y teniendo en cuenta que el usuario no dispone de una cuenta en Tempo, el usuario elegirá una de las dos opciones que se ofrecen para llevar a cabo el registro en Tempo, mediante email o mediante Facebook, como ya hemos explicado anteriormente. Si el usuario elige y pulsa sobre la opción de registro mediante email, accederemos a una Pantalla de Registro en la que se ofrece un formulario de registro. En esta pantalla el usuario podrá introducir una serie de datos que se le solicitan, entre otros, nombre de usuario, contraseña, email...

Figura 62 · Tempo – Formulario de Registro

Una vez rellenados todos los campos y pulsando sobre el botón aceptar, el proceso de registro finalizará con éxito y el usuario quedará registrado en Tempo. Cabe decir que Tempo guardará los credenciales de usuario, tanto después de llevar a cabo un registro, como después de haberse logueado manualmente en la aplicación, es decir, Tempo recordará el usuario que está usando la aplicación, sin necesidad de tener que introducir los credenciales de usuario cada vez que se quiera hacer uso de Tempo.

Como antes indicábamos, el usuario registrado tiene unas ventajas sobre el usuario no registrado, pudiendo acceder a más funcionalidades que ofrece Tempo. Lo primero que apreciamos tras haber iniciado sesión en Tempo es un cambio sobre los botones de la pantalla principal.

Figura 63 · Tempo – Botones usuario

Ahora el usuario registrado de Tempo dispone de un botón **Mi perfil,** el cual le da acceso a su cuenta de usuario.

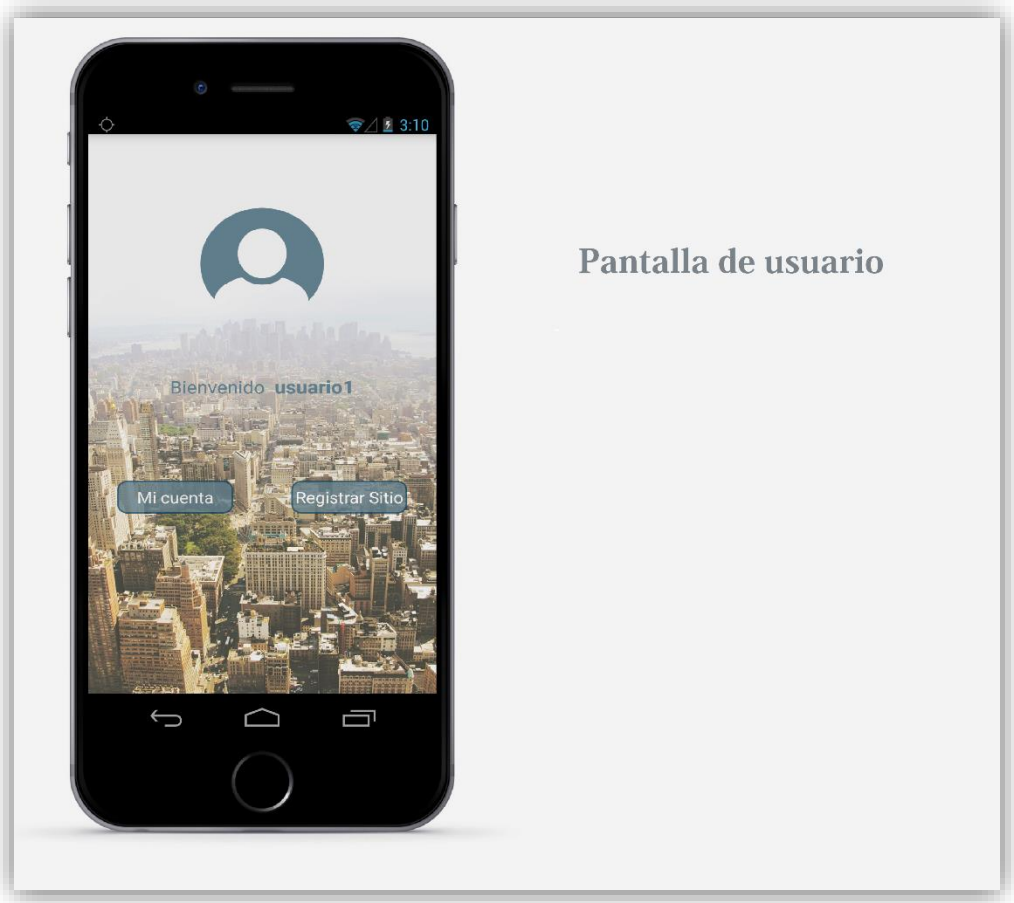


Figura 64 · Tempo – Pantalla de usuario

Al acceder a su cuenta de usuario, el usuario dispone de múltiples funciones a realizar. Como observamos en la figura 64, y que a continuación explicamos.

Una de las funcionalidades que se ofrece al usuario, es la posibilidad de modificar sus datos personales, es decir, la posibilidad de modificar, si así lo precisa, su nombre de usuario, su contraseña o su email, así como la posibilidad de eliminar su cuenta de Tempo. Para ello es necesario pulsar sobre el botón **Mi cuenta**, lo que nos llevará a la pantalla que observamos en la siguiente figura.

Figura 65 · Tempo – Pantalla de usuario

Otra de las funcionalidades de las que el usuario registrado en Tempo puede beneficiarse, es la posibilidad de registrar su sitio en caso de no haberlo hecho aún, y en caso de que así lo desee, y la otra posibilidad es la de modificar los datos de su sitio, en el caso de que haya registrado previamente su Sitio. Esta funcionalidad será explicada más adelante, en la sección siguiente, la sección **4.4 – Registro de Sitio**.

4.4 – Registro de Sitio

Un usuario puede añadir su Sitio de forma sencilla, rápida e intuitiva, sin necesidad de tener conocimientos técnicos avanzados, sólo el propio manejo de su Smartphone. Registrar un sitio en Tempo ofrece innumerables ventajas, poder registrar tu Sitio en una red de Sitios los cuales son accedidos por clientes buscando un servicio en concreto, no es más que una manera fantástica de potenciar tu restaurante, tu tienda, tu hotel, tu negocio, tu pequeña o gran empresa, o en definitiva, el servicio que ofreces mediante tu Sitio.

Una alternativa eficaz, gratuita, con potencial, al alcance de cualquiera, de autogestión, y que sólo ofrece ventajas, en comparación con las herramientas de las que hoy en día se dispone.

Con Tempo, no es el cliente el que acude al Sitio, sino que es el Sitio el que acude al cliente, estableciendo una vía directa de comunicación entre el servicio que ofrece nuestro Sitio y el cliente que anda en la necesidad de cubrir ese servicio.

El registro de un Sitio, se realiza de una manera muy simple e intuitiva, buscando la eficacia y comodidad para el usuario. Sólo con un Smartphone y Tempo, seremos capaces de situar nuestro negocio en el mapa y al alcance de todos. Para llevar a cabo el registro de nuestro Sitio en Tempo, como se ha indicado anteriormente, el usuario debe haberse creado una cuenta en Tempo. De ser así, tras pulsar en el botón **Mi Perfil** de la pantalla principal de la aplicación y acceder a la pantalla de Usuario, tal y como observábamos en la figura 51, tan sólo con pulsar el botón de **Registrar Sitio** accedemos al proceso para llevar a cabo el registro.

El proceso de registro se realiza en tres sencillos pasos. Cabe decir que en busca de la mayor facilidad y comodidad para el usuario mencionada anteriormente, el proceso de registro se lleva a cabo rellenando la información solicitada sobre la plantilla que más tarde será el perfil público de nuestro Sitio, el cual será al que los clientes accedan a visualizar para saber acerca de nuestro Sitio.

Primer paso

Figura 66 · Tempo – Registro 1

En primer lugar tenemos que establecer la imagen principal de nuestro Sitio. Pulsando sobre el cuadro que lo indica, se nos dará la opción de realizar una captura con nuestra cámara o buscar una imagen en nuestra galería. Esta será la imagen principal de nuestro Sitio.

En segundo lugar, tenemos que pulsar sobre el botón **Sitio**, tras ello, se nos abrirá un dialogo en el cual tendremos que introducir información relacionada con nuestro sitio, en particular, tendremos que rellenar 3 campos, el nombre, la dirección web y el teléfono de nuestro Sitio.

En tercer lugar, debemos establecer la dirección exacta del sitio, tras pulsar sobre el botón **Dirección**, se nos abrirá un mapa con nuestra ubicación, en el cual

deberemos buscar la dirección de nuestro Sitio y dejar pulsado sobre la pantalla en el lugar exacto donde se encuentre nuestro Sitio. Tras ello, aparecerá marcado el lugar seleccionado mediante un marcador de Tempo. En el caso de que no estemos satisfechos con la ubicación que hemos seleccionado tenemos dos opciones para modificarla. La primera opción es volver a dejar pulsado sobre la nueva ubicación exacta que deseemos, estableciéndose un nuevo marcador y eliminándose el anterior. La segunda opción es pulsar sobre el marcador y arrastrarlo al nuevo lugar que deseemos. Tras ello, deberemos pulsar sobre la flecha que se encuentra en la leyenda situada en la parte inferior del mapa, y tras esto nuestra ubicación quedará establecida.

El siguiente paso es establecer la categoría de nuestro Sitio, para ello pulsamos sobre el botón **Categoría**, se nos abrirá un dialogo en el cual podremos asignar la categoría de nuestro sitio entre un listado de categorías. Además, para poder potenciar la búsqueda de nuestro Sitio cuando un usuario use el buscador de Tempo, tenemos la opción si así lo deseamos, de añadir un *tag* a nuestro Sitio, es decir, podremos introducir manualmente una especificación sobre la categoría de nuestro Sitio, para especificar aún más las actividades que lleva a cabo nuestro Sitio. Por ejemplo, si disponemos de un restaurante, tras seleccionar la categoría de *Restaurante*, podremos especificar de qué tipo de restaurante se trata, Restaurante Chino o Hamburguesería, Pizzería, Asador Argentino, Buffet libre…

Por último, la última información a añadir en este primer paso del registro es el horario del sitio. Tempo muestra los horarios del Sitio cuando el usuario visita el perfil de este, indicando si se encuentra *Abierto* o *Cerrado* según el día y la hora en el que nos encontremos. Para ello, tras pulsar en el botón **Horario**, se nos abrirá un dialogo en el que podremos establecer distintos horarios en función de los días de la semana. Es sabido que un Sitio puede tener horarios distintos en función del día de la semana. Por ejemplo, una tienda de ropa puede tener un horario de Lunes a Viernes de apertura al público de 10:00 a 14:00 horas y de 17:00 a 20:30 horas, mientras que el Sábado puede tener un horario de 10:00 a 22:00 horas y el día Domingo encontrarse cerrado. Tempo proporciona en este apartado las herramientas oportunas para establecer varios horarios en función del día de la semana.

Tras haber completado todos los campos propuestos en la pantalla, accederemos al segundo paso pulsando sobre **Siguiente**. En el caso de que uno de los campos no haya sido establecido correctamente, se nos indicará mediante un mensaje. De tener todo correcto, accederemos al Segundo paso.

Segundo paso

Figura 67 · Tempo – Registro 2

Tras completar con éxito la información principal de nuestro sitio, accedemos a un segundo paso, en el cual podemos, de manera opcional, añadir hasta 8 imágenes más, a parte de la principal, para que los usuarios que visiten el perfil de nuestro Sitio, se hagan una idea más aproximada de nuestro Sitio, así como de poder atraerlos.

Como apreciamos en la imagen de la figura 67, para añadir una imagen, debemos pulsar sobre el icono + del primer recuadro. De nuevo se nos dará la opción de poder capturar una fotografía con la cámara o seleccionarla directamente desde la galería. Tras haber seleccionado una imagen, esta se mostrará en la parte superior, con el fin de poder visualizarla y garantizar la correcta elección. En el caso de que deseemos, modificar o eliminar una de las imágenes seleccionadas, debemos pulsar sobre ella, dónde se nos dará la opción de llevar a cabo una acción sobre la imagen pulsada.

Como hemos comentado anteriormente, la opción de añadir más imágenes, además de la principal, es opcional. Hay que tener en cuenta que un Sitio con una

galería de imágenes puede atraer a más clientes cuando estos visiten el perfil.

Tras haber añadido el número de imágenes deseado, pulsamos sobre el botón **Siguiente,** el cual nos llevará a un tercer paso.

Tercer paso

Figura 68 · Tempo – Registro 3

En este tercer paso, debemos redactar una breve descripción sobre nuestro sitio. Aquí, podemos introducir la información que consideremos más relevante, con el fin de vender nuestro sitio de la mejor manera posible y destacando los aspectos más significativos de nuestro Sitio para atraer al mayor número de clientes posible.

Tras llevar a cabo este paso, pulsamos **Siguiente**, y con esto se habrá completado el registro de nuestro Sitio. Nuestro Sitio quedará almacenado en Tempo, y será accesible por todos.

4.5 – Usando el buscador

Tempo es una red de Sitios que ofrece un potente buscador callejero a toda persona que necesite un servicio en cualquier momento y en cualquier lugar y de manera rápida y eficiente. Los usuarios de Tempo podrán hacer uso del buscador y ver en el mapa los establecimientos más cercanos a su ubicación, acceder a los perfiles e informarse de dichos Sitios, para así poder seleccionar aquellos más afines a sus necesidades y asegurar el éxito de su elección.

Como observábamos anteriormente en la figura 57, en la pantalla principal de la aplicación se dispone de un buscador de sitios. Para hacer uso de esta fantástica herramienta, el usuario tan solo tiene que introducir lo que desea encontrar. Además, Tempo ofrece un servicio de sugerencias mientras el usuario introduce la palabra clave a buscar.

Figura 69 · Tempo - Buscador

El usuario tiene la opción de pulsar sobre una de las sugerencias que le puedan aparecer, en el caso de buscar un Sitio exacto y este se encuentre entre las sugerencias, al hacerlo Tempo nos llevara al mapa con la ubicación exacta del Sitio.

También el usuario puede optar por buscar mediante el texto introducido en el buscador, y Tempo se encargará de buscar los Sitios relacionados con el texto introducido y mostrar los Sitios relacionados en el mapa.

Por ejemplo, si un usuario de Tempo se encuentra en Barcelona, más exactamente en Plaça Catalunya, e introduce la palabra Museo, le aparecerá en las sugerencias los museos más cercanos registrados en Tempo, entre ellos el Museo de Arte de Barcelona, próximo a la ubicación del usuario, y pulsando sobre dicha sugerencia nos llevará al mapa con la ubicación exacta del Sitio. De no ser así y buscar la palabra clave *Museo* mediante el buscador, nos llevará a un mapa con las ubicaciones exactas de los museos registrados en Tempo próximos a nuestra ubicación, entre otros, el Museo de Arte de Barcelona.

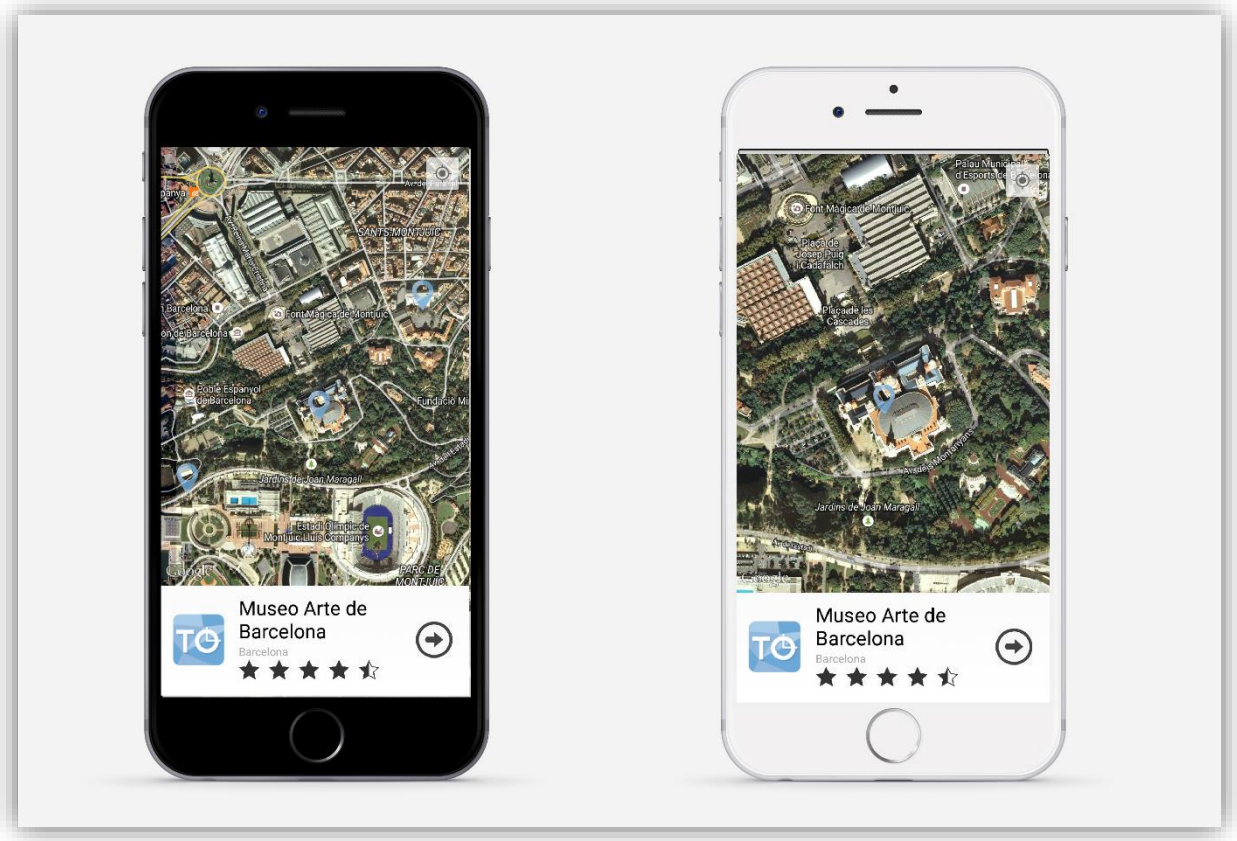


Figura 70 · Tempo – Ubicación Sitio

Cuando nos encontramos en el mapa visualizando la ubicación exacta de los Sitios, podemos pulsar sobre el marcador de cada Sitio, consiguiendo que una leyenda en la parte inferior del mapa nos muestre información relacionada al Sitio, y pudiendo acceder al perfil de este pulsando sobre la flecha.

Como apreciamos en la imagen, al acceder al perfil de un Sitio, tenemos información de primera mano sobre el Sitio, horarios, redes sociales, galería, descripción del lugar...

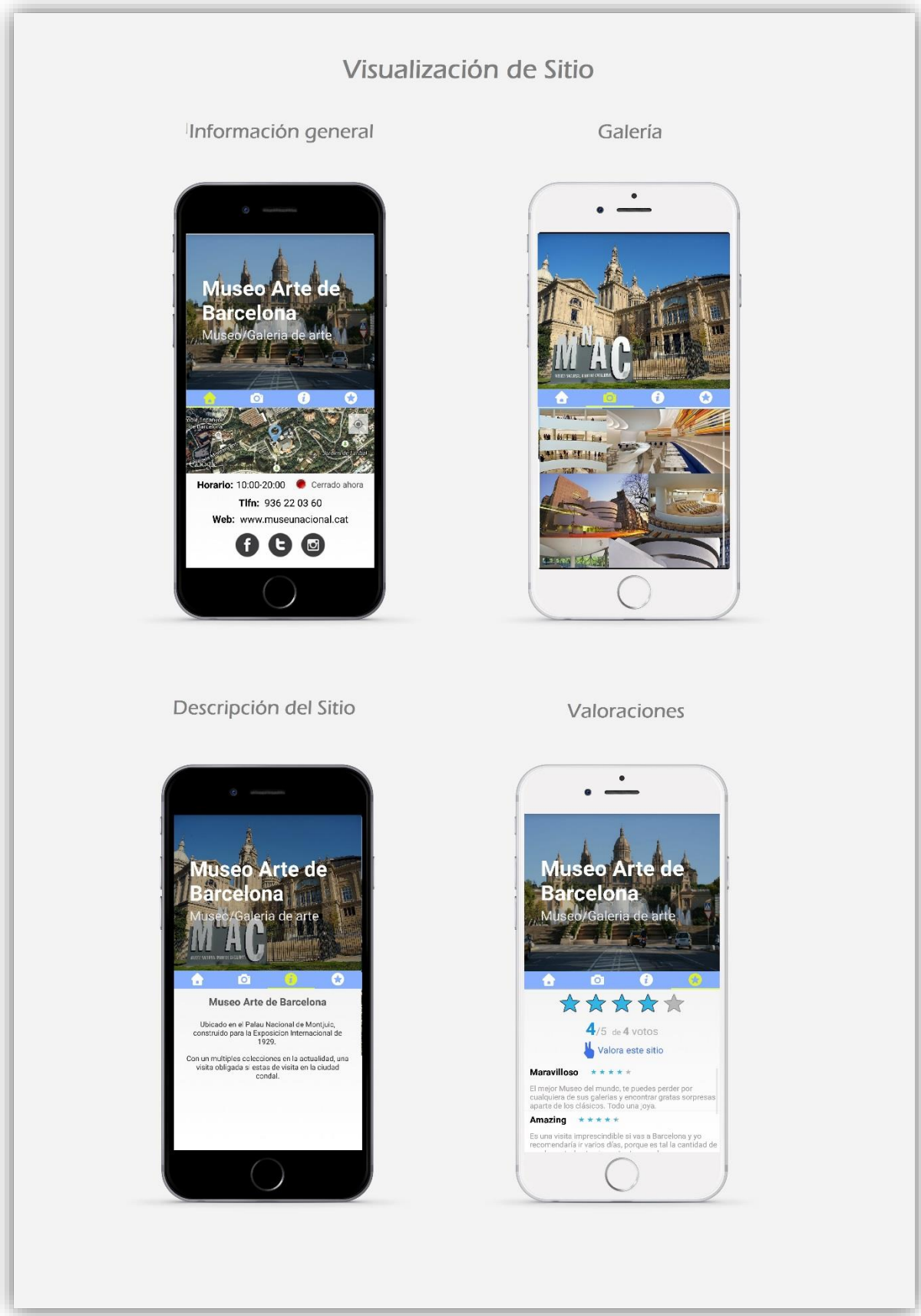


Figura 71 · Tempo - Sitio

Capítulo 5 – Conclusiones y líneas futuras

Tras la finalización de este proyecto, se puede afirmar que el objetivo ha sido cumplido. Se ha desarrollado una herramienta fiel a los principios y funcionamientos con los que un día fue imaginada.

Partiendo de unas capacidades y habilidades limitadas para llevar a cabo el proyecto, se han ido adquiriendo, en el transcurso de la realización, una serie de conocimientos que han hecho posible el desarrollo de la aplicación, logrando además, un importante avance tanto personal como profesional. Gracias a estos avances, se han llevado a cabo las siguientes implementaciones:

- Sistema de usuarios, logrando poder ofrecer funcionalidades extras a usuarios registrados en la aplicación.
- Sistema de registro de Sitios, ofreciendo el registro de un Sitio a través de formularios, para su posterior almacenado en la Base de Datos de la aplicación.
- Sistema de visualización de Sitios, dando la posibilidad al usuario de acceder al perfil de un Sitio para su visionado.
- Sistema de auto-gestión de datos. El usuario puede modificar sus datos personales, así como los datos de su Sitio en el caso de haber registrado uno.
- Sistema de búsqueda. Implementación de un buscador de Sitios, mediante la introducción de palabras claves, con servicios de filtrado y sugerencias.

Tempo es fruto de la innovación y el emprendimiento.

Se ha podido comprobar que el potencial de la tecnología no tiene límites, y que desde hace unos años hacia adelante, juega un papel vital en nuestra sociedad, ofreciendo posibilidades ilimitadas en todos los campos y en beneficio de la sociedad.

La viabilidad del proyecto es una realidad. Tan sólo con un Smartphone se puede hacer uso de Tempo. Además, el desarrollo de la aplicación ha sido realizado siempre teniendo en cuenta la búsqueda de la mayor simplicidad en el manejo de la aplicación por parte del usuario, sin necesidad de tener conocimientos avanzados de ningún tipo más que el manejo de un Smartphone.

5.1 – Líneas futuras

Cabe decir que esto es sólo el comienzo de Tempo, la adición de funcionalidades y mejoras no son una lista acotada y crece en función de las necesidades sociales del momento. Por esto, la posibilidad de ampliar Tempo con mayor número de funcionalidades o mejoras es muy grande, aunque esto no quiere decir que la aplicación está incompleta, al contrario, sería un gran error no aceptar que siempre se puede ir a mejor.

Unos ejemplos de próximas mejoras previstas para añadir a las ya funcionalidades que ofrece Tempo son los siguientes:

- Recomendaciones personalizadas para cada usuario, en función de diversos criterios como búsquedas o valoraciones de Sitios.
- Posibilidad de ofrecer ofertas por parte de los Sitios mediante la aplicación de Tempo.
- Inclusión de perfil de usuario.
- Mejorar el sistema de valoraciones.
- Mejorar y optimizar el código.
- Sistema contador de visitas a un Sitio.


tempo