Proyecto de Desarrollo

de Aplicaciones Multiplataforma

Sistema FlipBook

Tutor: José Luis González Sánchez

Alumno: Jorge Segade Martín de la Leona

Curso académico 2020-2021



C.F.G.S. Desarrollo de Aplicaciones Multiplataforma

**Índice**

[1. PRESENTACIÓN DEL PROYECTO. 3](#__RefHeading___Toc531_228099368)

[1.1 Explicación resumida. 3](#__RefHeading___Toc533_228099368)

[1.2 Estudio de mercado. 4](#__RefHeading___Toc535_228099368)

[1.3 Valor del producto. 6](#__RefHeading___Toc537_228099368)

[2. PLANIFICACIÓN DE TAREAS Y ESTIMACIÓN DE COSTES. 6](#__RefHeading___Toc539_228099368)

[2.1 Planificación y organización de tareas. 6](#__RefHeading___Toc541_228099368)

[2.2 Estimación de costes y recursos: hardware, software y humanos. 7](#__RefHeading___Toc543_228099368)

[2.3 Herramientas usadas. 9](#__RefHeading___Toc545_228099368)

[2.4 Gestión de riesgos. 9](#__RefHeading___Toc547_228099368)

[3. ANÁLISIS DE LA SOLUCIÓN. 10](#__RefHeading___Toc549_228099368)

[3.1 Análisis de la solución (requisitos, mapa de historia de usuario…). 10](#__RefHeading___Toc551_228099368)

[3.2 Análisis de escenarios (casos de uso, mapa de interacción …). 11](#__RefHeading___Toc553_228099368)

[4. DISEÑO DE LA SOLUCIÓN. 16](#__RefHeading___Toc555_228099368)

[4.1 Diseño de la interfaz de usuario y prototipos. 16](#__RefHeading___Toc557_228099368)

[4.2 Diseño de la persistencia de la información. 29](#__RefHeading___Toc559_228099368)

[4.3 Diseño de la arquitectura del sistema. 33](#__RefHeading___Toc561_228099368)

[5. IMPLEMENTACIÓN DE LA SOLUCIÓN. 33](#__RefHeading___Toc563_228099368)

[5.1 Justificación tecnológica. 33](#__RefHeading___Toc565_228099368)

[5.2 Aspectos esenciales de la implementación. 35](#__RefHeading___Toc567_228099368)

[5.3 Desarrollo de la funcionalidad indicada por el tutor/a. 35](#__RefHeading___Toc569_228099368)

[6. TESTEO Y PRUEBAS DE LA SOLUCIÓN. 35](#__RefHeading___Toc571_228099368)

[6.1 Plan de pruebas (unitarias, integración, sistema y usuarios). 35](#__RefHeading___Toc573_228099368)

[6.2 Solución a problemas encontrados. 35](#__RefHeading___Toc575_228099368)

[7. LANZAMIENTO Y PUESTA EN MARCHA. 35](#__RefHeading___Toc577_228099368)

[7.1 Aspectos relevantes del despliegue y puesta en marcha del sistema. 35](#__RefHeading___Toc579_228099368)

[7.2 Manual de uso. 35](#__RefHeading___Toc581_228099368)

[8. VALORACIÓN Y CONCLUSIONES. 36](#__RefHeading___Toc583_228099368)

[9. BIBLIOGRAFÍA Y RECURSOS UTILIZADOS. 36](#__RefHeading___Toc585_228099368)

# 1. PRESENTACIÓN DEL PROYECTO.

### 1.1 Explicación resumida.

Se trata de un sistema de Bookcrossing donde los usuarios dejarán libros en lugares públicos para que los recojan otros lectores, que después harán lo mismo.

En el sistema **móvil** (APP) encontramos las siguientes características/funciones:

-Los libros disponibles podrán localizarse por GPS.

-Línea de tiempo con los libros más populares indicando su estado, puntuación, comentarios.

-Un listado de contactos donde podemos seleccionar uno de ellos para quedar e intercambiar libros.

-Una línea de tiempo donde veremos el historial de nuestras lecturas.

-Una zona de perfil público con el nombre, descripción del usuario, número de seguidores, fotografía de perfil, lecturas recientes y un código QR único para que otros usuarios puedan añadirnos como contacto.

-Un buscador de libros en todo el sistema.

-La reserva de un libro se realiza a través de su código de barras, escaneándolo desde la aplicación e indicando la fecha de devolución.

-A la hora de mostrar los datos de un libro nos debe indicar en qué puntos de recogida podemos obtenerlo y la ruta para llegar.

-Podremos tener un listado de usuarios de nuestra plataforma e intercambiar información con ellos.

-Para agregar a un usuario o seguirlo, podemos buscar su nombre de usuario o simplemente escanear el QR que nos puede mandar o mostrar por su pantalla.

-El sistema de creación de cuentas e identificación, puede ser propio y además implementará los sistemas de autorización de Google y Twitter.

-Podremos recibir notificaciones en todo momento.

En el sistema de **administración** (Programa para PC) encontramos las siguientes características/funciones:

-Gestión de libros.

-Gestión de reservas.

-Análisis de reservas.

-Toma de medidas para usuarios que hacen mal uso del sistema o no devuelven libros en la fecha indicada.

-Notificaciones y correos.

-Informes detallados.

En el **servidor** encontramos las siguientes características/funciones:

-Todos los datos se almacenan remotamente en un servidor (imágenes, usuarios, historial, etc.)

-La autorización se hará utilizando el sistema de Firebase o en su defecto JWT.

-Las aplicaciones móviles serán multiplataforma y funcionar en Android o iOS.

-La aplicación debe cumplirrá con los patrones de diseño Material o iOS, debe ser usable y asegurar una correcta experiencia de uso.

-Los datos se almacenarán remotamente en una base de datos tipo relacional PostgreSQL o No-SQL MongoDB/Firebase.

-Se usarán tecnologías de cámara, geolocalización y geoposiconamiento, así como navegación.

-El acceso a almacenamiento se hará́ a través de una API REST propia en SpringBoot u otro mecanismo, estará securizada (JWT) y será probable y testeable completamente sin aplicación cliente con herramientas como POSTMAN.

### 1.2 Estudio de mercado.

**Estudio de mercado.**

Queremos estudiar los gustos y preferencias de nuestros posibles clientes, así como su ubicación, educación y ocupación, entre otros aspectos. Para diseñar y ofrecer los productos que ellos nos soliciten.

El propósito de este estudio es conocer a nuestros clientes, competidores y saber que es lo que se demanda en lo que se refiere a aplicaciones y así dar solución a los problemas encontrados y lograr que nuestros clientes se sientan satisfechos.

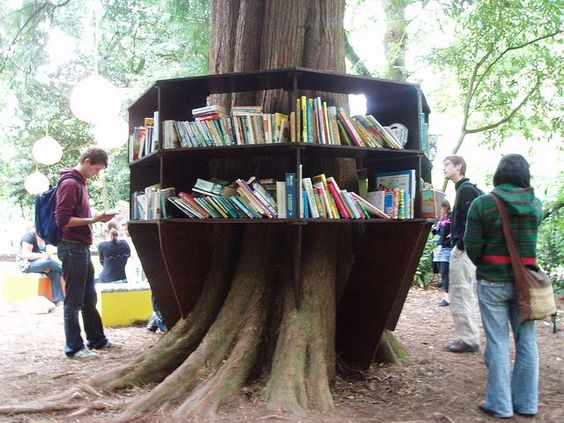
Como competidores directos se encontrarían instagram, wallapop, whatsapp, telodoygratis y milanuncios entre otros.





Como clientes potenciales podemos prever que serán hombres y mujeres ( El 64,9 % de las mujeres lee libros por placer, una actividad practicada por el 54,4 % de los hombres según el **Barómetro de Hábitos de Lectura y Compra de Libros**).

Como necesidad que cubre nuestro sistema tenemos la de fomentar la lectura gratuita y favorecer las relaciones humanas entre lectores.



### 1.3 Valor del producto.

Sobre el valor del producto podemos decir que un sistema así no tiene valor físico sino que tiene un valor intangible. Aunque si hubiese muy buenas noticias respecto a la cantidad de usuarios que usan la aplicación no se descarta en un futuro que tuviese valor tangible.

# 2. PLANIFICACIÓN DE TAREAS Y ESTIMACIÓN DE COSTES.

### 2.1 Planificación y organización de tareas.

### 2.2 Estimación de costes y recursos: hardware, software y humanos.

Proyecto FlipBook

Desarrollo de contenidos

Diseño gráfico

Maquetación

Proyecto iOS/Android

Desarrollo técnico

Software

Aplicación IOS / Android

Complementos

Notificaciones Push

Acceso a la App

mediante Login

Publicar App / Cuenta de desarrollador

Google Play

App Store

Desplegar App

AWS

Mantenimiento y

actualizaciones

Desarrollo y publicación de la aplicación.

|  |  |  |
| --- | --- | --- |
| **Descripción** | **Horas** | **Coste (17€ / h)** |
| Diseño gráfico. | 10h | 170€ |
| Maquetación. | 10h | 170€ |
| Desarrollo del software. | 240h | 4.080€ |
| Cuenta desarrollador. | 1h | 17€  + 25€(Android) +85€/año(IOS) |
| Despliegue en AWS. (EC2 + IP Elástica + 30GB/mes) | 10h | 170€ + 240€/año |
| Mantenimiento y actualizaciones anuales. | 10h | 170€ |
| **TOTAL** | 281h | A continuación. |

\* El primer año la cuota de AWS es gratuita.

\* Los precios no incluyen IVA.

El tiempo estimado es de **14 semanas.**

Precio total sin IVA el primer año: 4.887€

Precio total sin IVA el segundo año: 495€

### 2.3 Herramientas usadas.

Las herramientas a usar serán en su totalidad software libre:

* Justinmind para la maquetación.
* Vscode para el desarrollo del front-end.
* Eclipse para el desarrollo del back-end.
* Xampp para el despliegue en local.
* Github como control de versiones del código.
* Github desktop para la gestión del repositorio.
* Trello para la gestión de tareas.

### 2.4 Gestión de riesgos.

|  |  |  |
| --- | --- | --- |
| **Riesgo** | **Probabilidad** | **Magnitud** |
| Planificación demasiado optimista. | 30% | 1 semana |
| Cambios no esperados. | 60% | 2 semanas |
| Añadir una característica. | 30% | 1 semana |
| Diseño inadecuado. | 20% | 4 semanas |

# 3. ANÁLISIS DE LA SOLUCIÓN.

### 3.1 Análisis de la solución (requisitos, mapa de historia de usuario…).

Requisitos funcionales:

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas.

-RF1: El sistema debe permitir geolocalizar libros.

-RF2: El sistema debe mostrar una línea de tiempo con los libros más populares indicando su estado, puntuación, comentarios.

-RF3: El sistema debe mostrar un listado de contactos donde podemos seleccionar uno de ellos para quedar e intercambiar libros.

-RF4: El sistema debe mostrar una línea de tiempo donde veremos el historial de nuestras lecturas.

-RF5: El sistema debe mostrar una zona de perfil público con el nombre, descripción del usuario, número de seguidores, fotografía de perfil, lecturas recientes y un código QR.

-RF6: El sistema debe disponer de un buscador de libros.

-RF7: El sistema debe permitir reservar un libro.

-RF8: El sistema debe mostrar los datos de recogida de un libro

-RF9: El sistema debe mostrar un listado de usuarios de nuestra plataforma.

-RF10: El sistema debe permitir intercambiar información con los otros usuarios.

-RF11: El sistema debe permitir agregar usuarios.

-RF12: El sistema debe disponer de un sistema de creación de cuentas.

-RF13: El sistema debe permitir la identificación de un usuario.

-RF14: El sistema debe disponer de notificaciones en todo momento.

-RF15: El sistema debe permitir gestionar los libros subidos.

-RF16: El sistema debe permitir al usuario editar su perfil.

-RF17: El sistema debe estar securizado.

-RF18: El sistema debe ser testeable.

Requisitos no funcionales:

Un requisito no funcional o atributo de calidad es, en la ingeniería de sistemas y la ingeniería de software, un requisito que sabe bien y especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que estos corresponden a los requisitos funcionales.

-RNF1: El sistema debe cumplir los patrones de diseño Material.

-RNF2: El sistema debe cumplir los patrones de diseño iOS.

-RNF3: El sistema debe ser usable.

-RNF4: El sistema debe asegurar una correcta experiencia de uso.

-RNF5: Los datos del sistema se almacenarán remotamente en una base de datos tipo relacional PostgreSQL o No-SQL MongoDB/Firebase.

-RNF6: El sistema debe permitir leer el código de barras o QR con la cámara del móvil.

-RNF7: El acceso a almacenamiento se hará́ a través de una API REST.

-RNF8: El sistema se desarrollará con un framework para apps híbridas.

-RNF9: El sistema debe ser probable y testeable completamente con herramientas como POSTMAN.

-RNF10: El sistema debe estar desplegado y accesible desde cualquier momento o lugar utilizando Docker, Heroku, AWS, Servidor y Tienda de Apps.

-RNF11: El método de autenticación del sistema debe implementar los sistemas de autorización de Google y Twitter.

Requisitos de información:

Son aquellos requisitos que se refieren a los datos tratados en el programa.

-RI1: Usuario. Representa el modelo con el que se tratará para la gestión de los usuarios, sus campos son: id, nombre, apellidos, email, contraseña, foto de perfil, fecha de registro.

-RI2: Libro. Representa el modelo con el que se tratará para la gestión de los libros, sus campos son: id, titulo, autor, genero.

-RI3: localización. Representa el modelo con el que se tratará para la gestión de las localizaciones de los libros, sus campos son: id, idLibro ,latitud, longitud.

-RI4: Reserva. Representa el modelo con el que se tratará para la gestión de las reservas de los libros, sus campos son: id, idUsuario, idLibro, fechaReserva, fechaDevolucion.

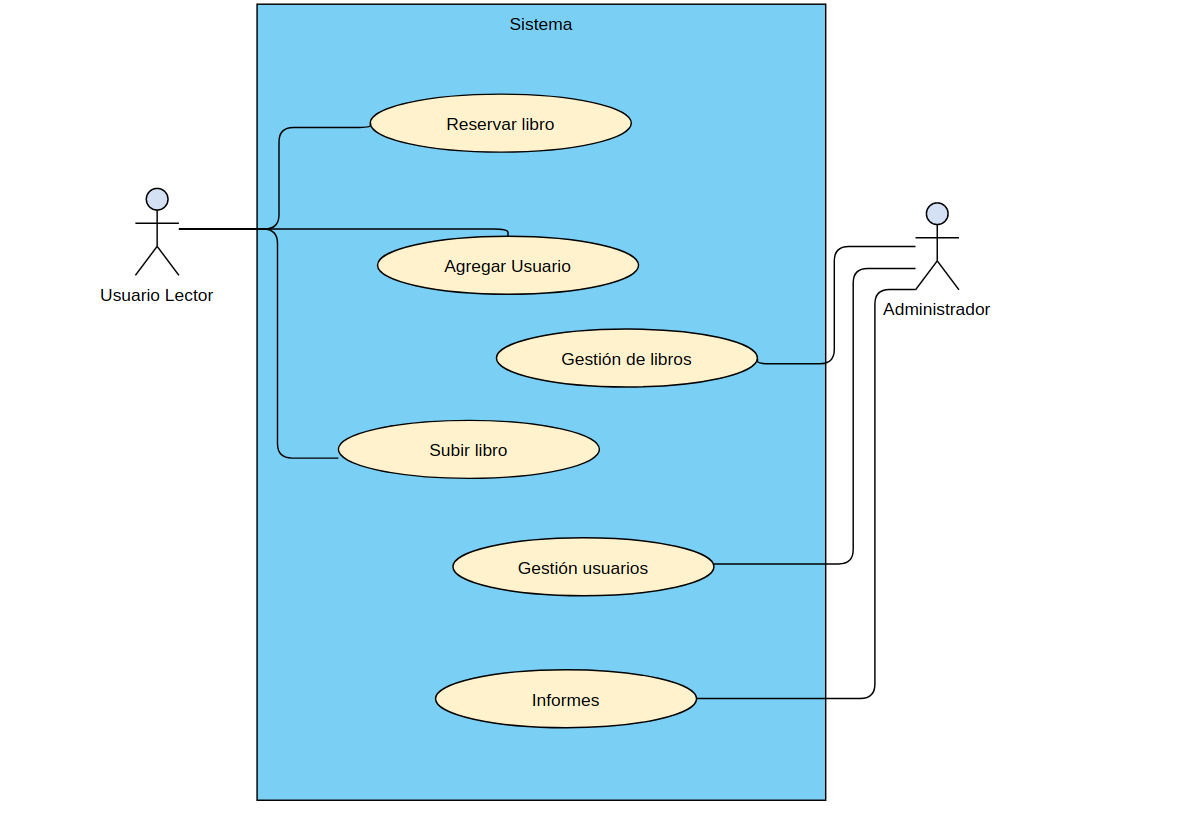
### 3.2 Análisis de escenarios (casos de uso, mapa de interacción **…**).

**Escenarios.**

Un escenario es una secuencia de pasos que describen una interacción entre un usuario y un sistema.

**Actores.**

Los actores y el sistema:

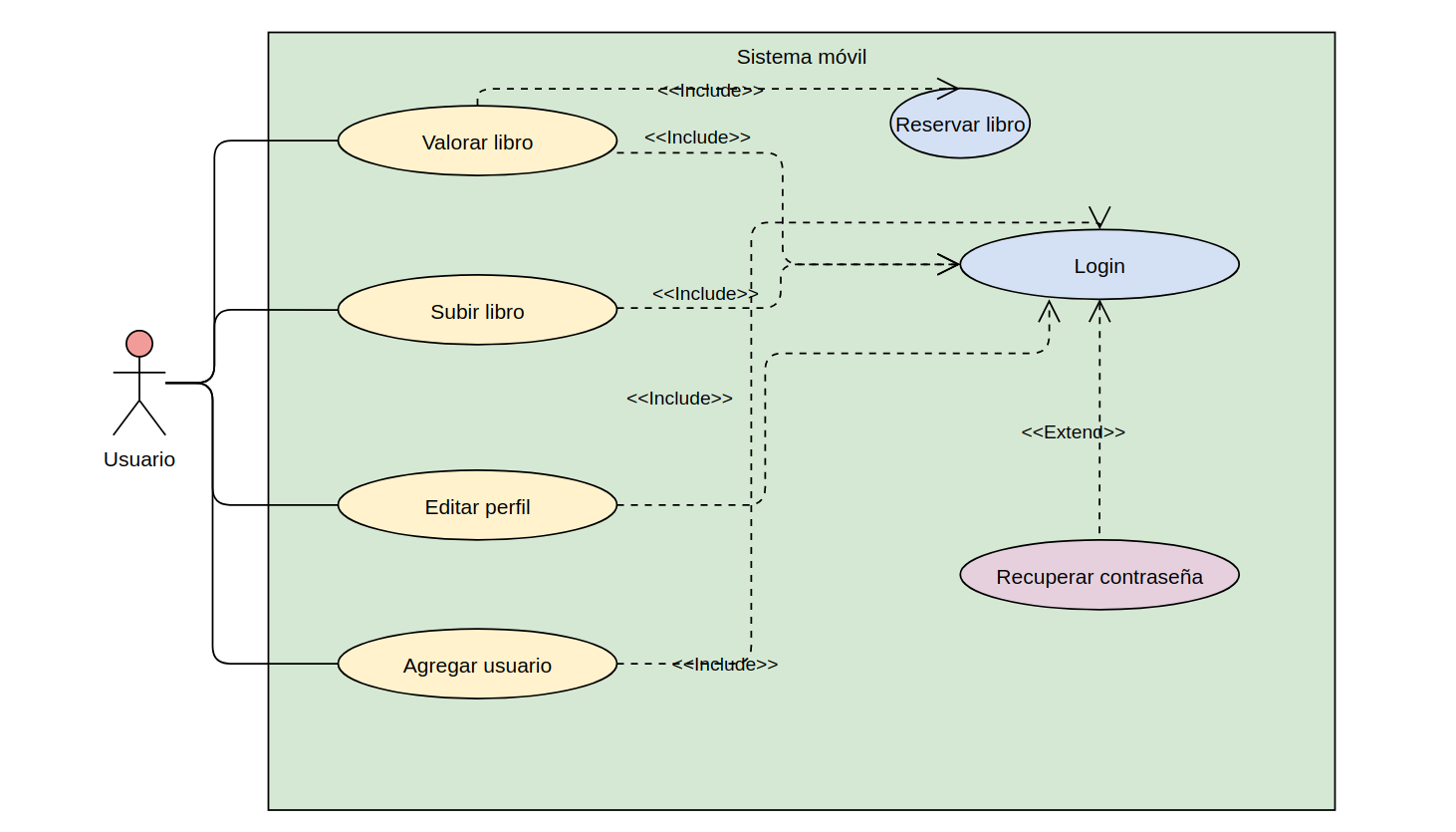
****

**Diagrama de casos de uso**

Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas.

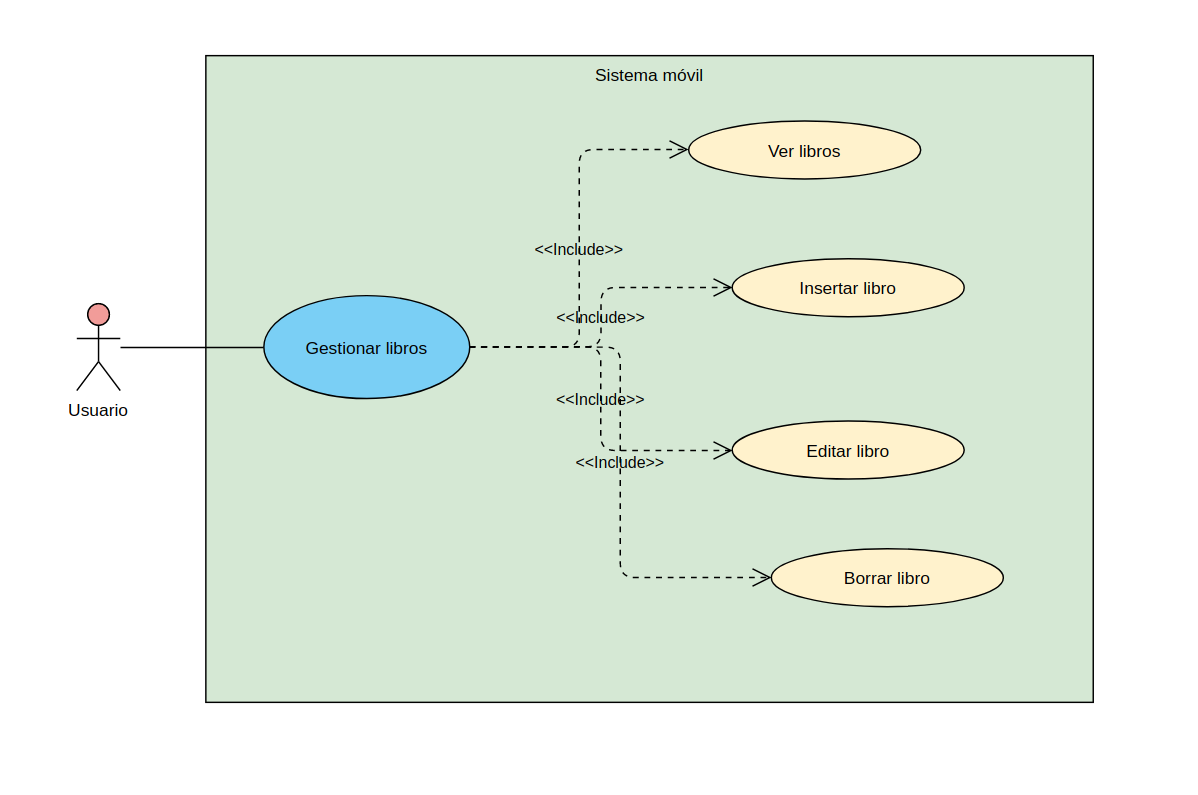
DCU-01

Diagrama general.

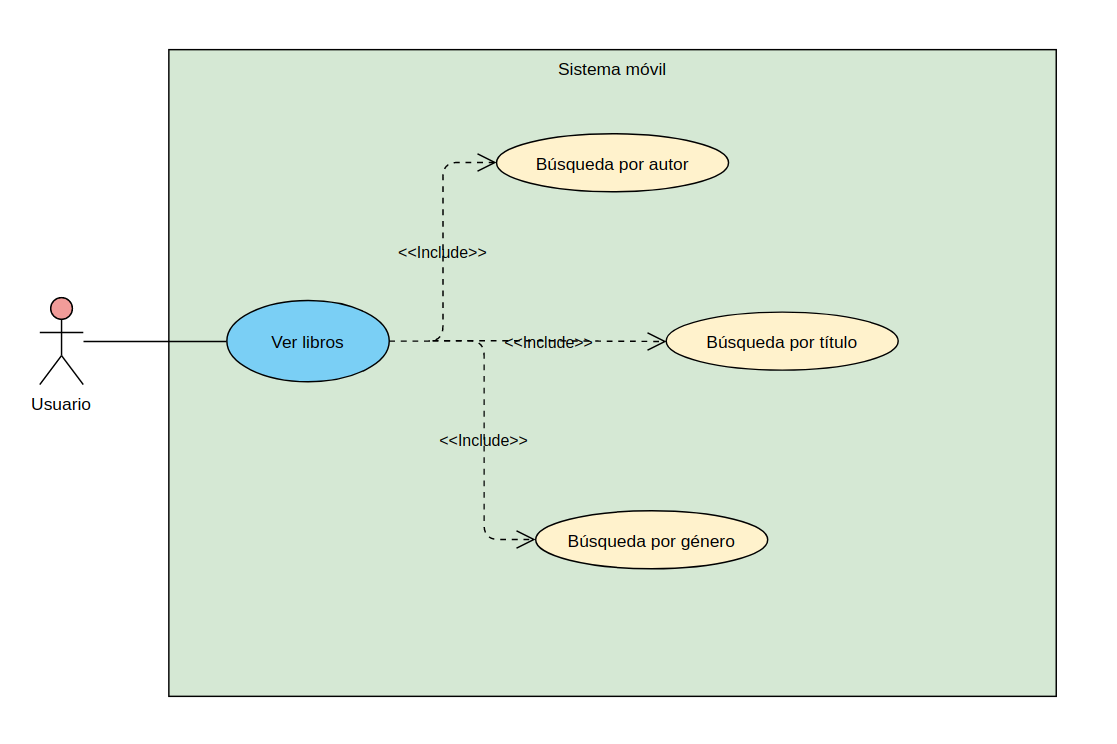


DCU-02

Diagrama del crud libros.



DCU-03

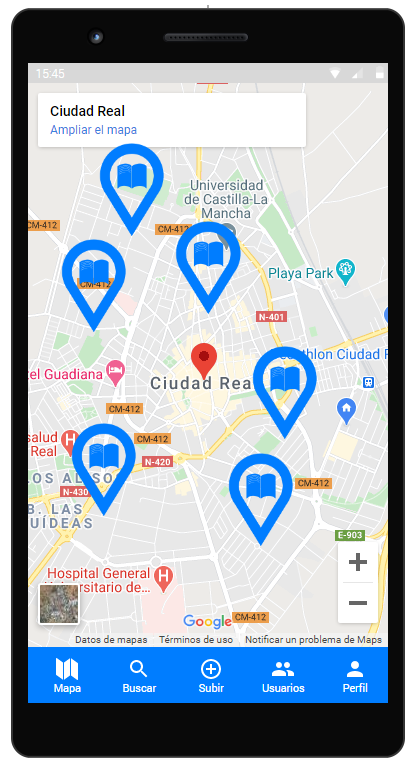
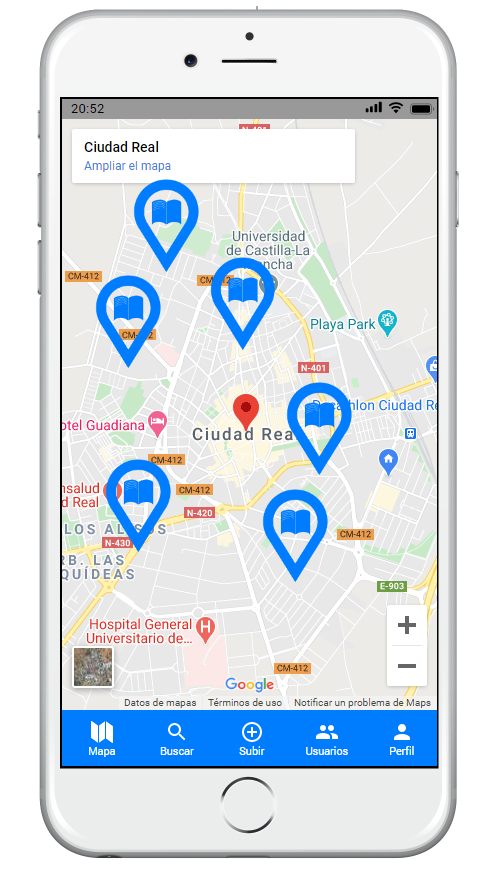
Diagrama de búsqueda de libro.

# 4. DISEÑO DE LA SOLUCIÓN.

### 4.1 Diseño de la interfaz de usuario y prototipos.

La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo.

Normalmente suelen ser fáciles de entender y fáciles de accionar, aunque en el ámbito de la informática es preferible referirse a que suelen ser "usables", "amigables e intuitivos" porque es complejo y subjetivo decir "fácil".



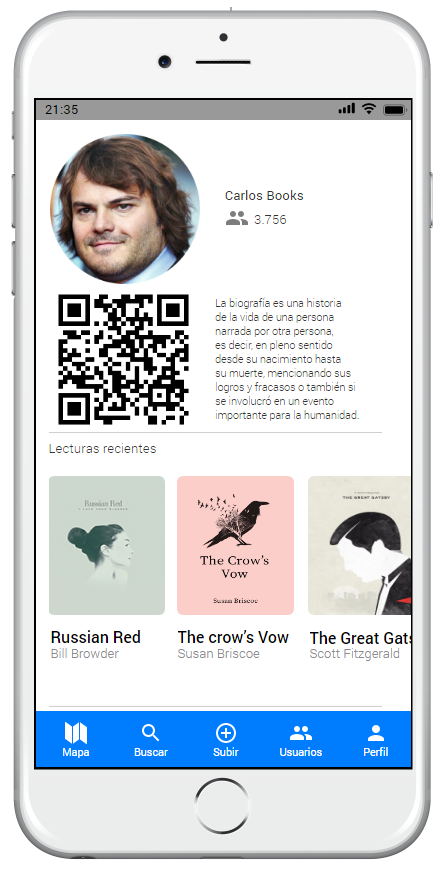
Fuente:[https://es.wikipedia.org/wiki/Interfaz\_de\_usuario#:~:text=La%20interfaz%20de%20usuario%20es,el%20usuario%20y%20el%20equipo.](https://ikastaroak.birt.eus/edu/argitalpen/backupa/20200331/1920k/es/DAM/DI/DI02/es_DAM_DI02_Contenidos/website_63_elementos_interactivos_de_la_interfaz_de_usuario.html)

Elementos interactivos de la interfaz de usuario.

En el diseño interactivo de interfaces, todos los elementos que fomenten la realimentación son fundamentales, ya que el usuario debe estar informado en cada momento de las acciones que realiza.

Para el diseño de elementos interactivos en interfaces , hay que tener en cuenta el tipo de acciones que se van a llevar a cabo y cuál es la interacción más apropiada de cada una con el usuario.

En este caso, observamos una interfaz dotada por elementos interactivos, siendo su finalidad el testeo de ésta y la decisión final de la interfaz de la aplicación.



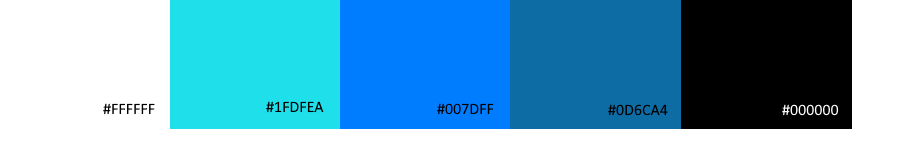
Fuente:[https://es.wikipedia.org/wiki/Interfaz\_de\_usuario#:~:text=La%20interfaz%20de%20usuario%20es,el%20usuario%20y%20el%20equipo.](https://ikastaroak.birt.eus/edu/argitalpen/backupa/20200331/1920k/es/DAM/DI/DI02/es_DAM_DI02_Contenidos/website_63_elementos_interactivos_de_la_interfaz_de_usuario.html)

Paleta de colores

La paleta de colores que se ha elegido para la aplicación consta de 4 colores sencillos para no bombardear al usuario con un abanico de colores haciendo así que la aplicación adquiera personalidad propia.

El logo está compuesto por alguno de ellos.





Tipografía

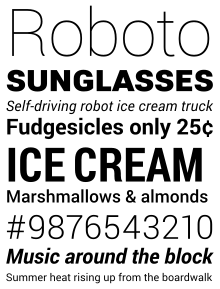
La importancia de la tipografía.

Cuando hablamos de diseño también hablamos de la estructura, el esqueleto, de la app en sí. No es simplemente algo que se deje influenciar por las tendencias. Aunque estas hacen que varíe el diseño de apps con nuevas tendencias, lo cierto es que la lógica de interacción que definimos en el diseño permanece prácticamente intacta e invariable.

La tipografía es uno de los elementos que más ha variado con el paso del tiempo. Ha pasado de ser un elemento olvidado a ser un paso más en el proceso de diseño de una aplicación móvil.

En este caso se ha optado por la tipografía Roboto, ​es una familia de fuentes tipográficas del tipo sans-serif desarrollada por Google como fuente del sistema operativo móvil Android 4.0 “Ice Cream Sandwich”.

​Google describe la fuente como “moderna, pero a la vez accesible”.

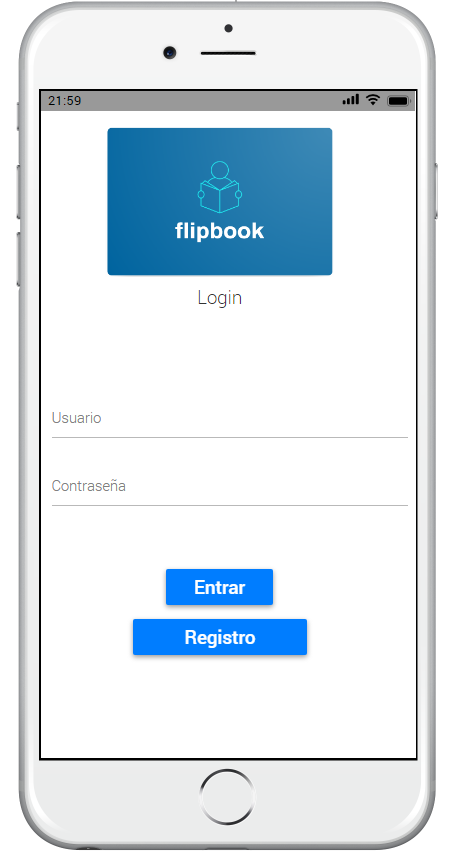


Fuente: <https://www.yeeply.com/blog/diseno-de-apps-la-importancia-de-la-tipografia/>

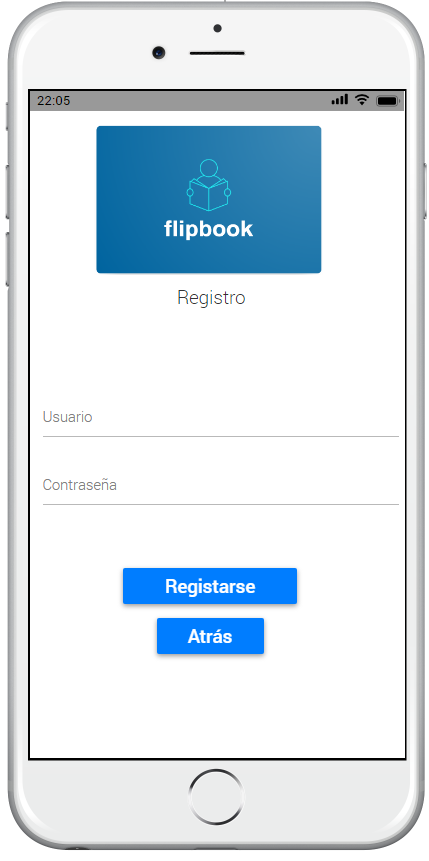
Navegabilidad

Aquí se va a representar la navegabilidad de la aplicación, se ha usado como modelo la aplicación en IOS.

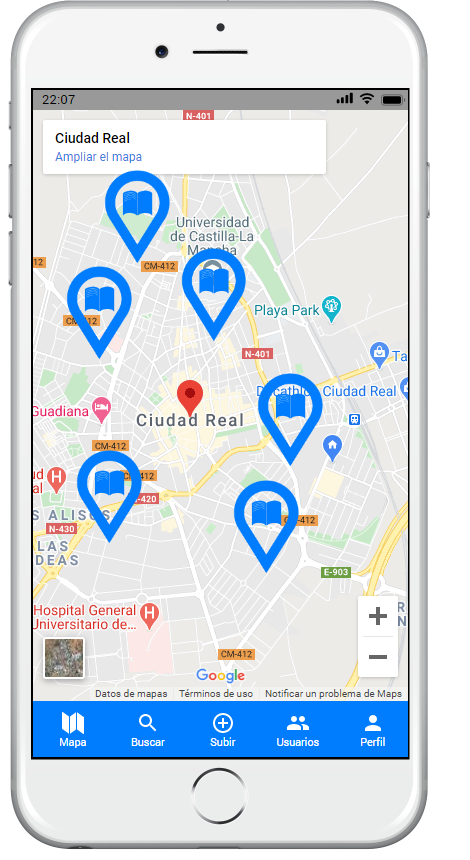
**Login**: Introduciremos nuestros datos de inicio de sesión para poder entrar al sistema, también podremos registrarnos pulsando el botón “Registro”.



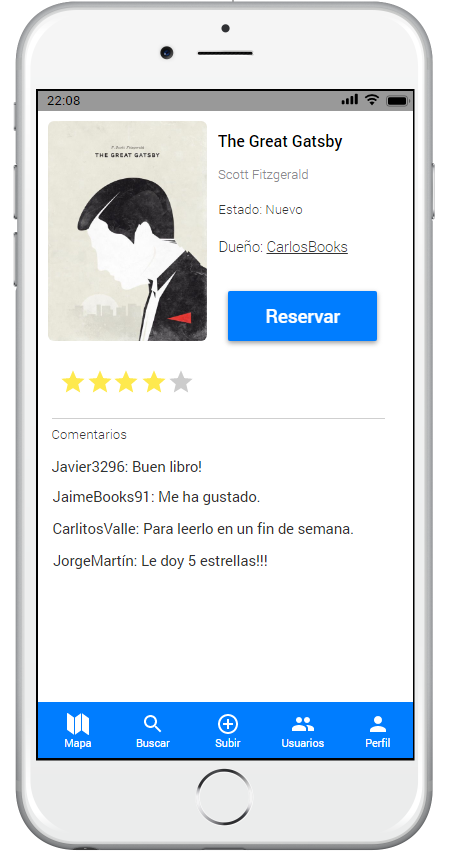
**Registro**: Nos permitirá crear una cuenta para tener acceso a la aplicación o podemos pulsar “Atrás” para volver a la pantalla de login.



**Mapa**: Una vez iniciada sesión lo primero que veremos será el mapa con los libros de los usuarios cercanos:



Si pulsamos en cualquiera de los libros nos redirigirá a la pantalla de visualización de éste:



Aquí podremos:

-Ver la información del libro.

-Ver su dueño.

-Ver la valoración del libro.

-Ver los comentarios del libro.

-Reservar el libro.

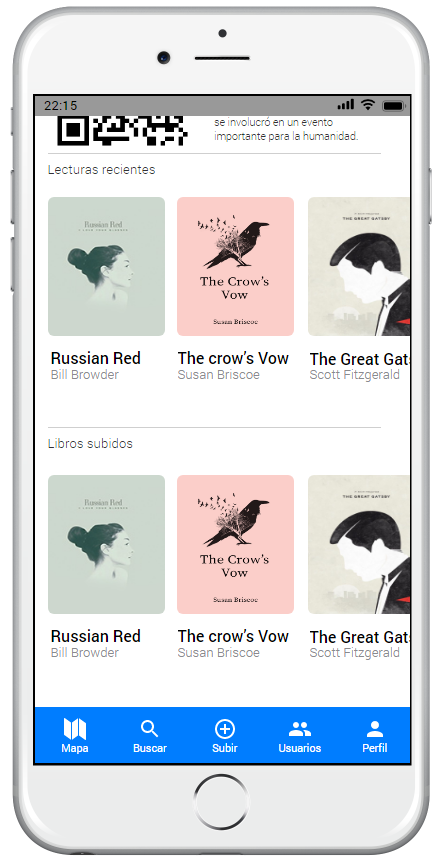
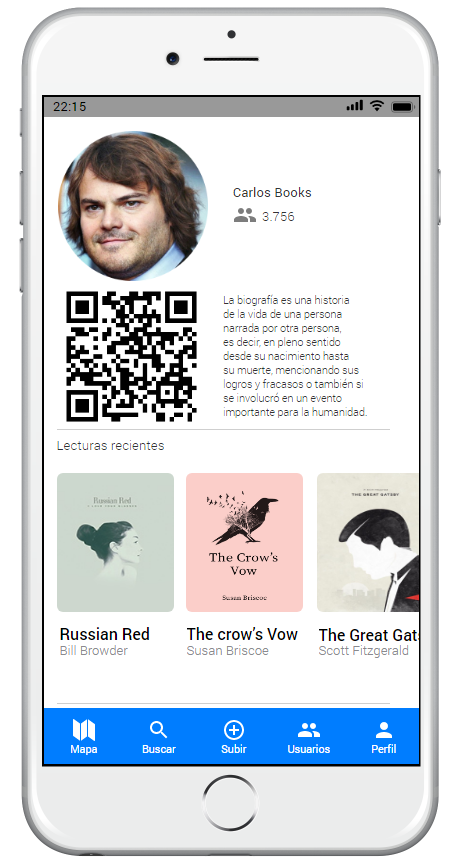
-Ir al perfil del dueño del libro.

Una vez reservado el libro, nos devolverá al mapa por si se quisiera reservar otro libro.

**Perfil del dueño del libro**:

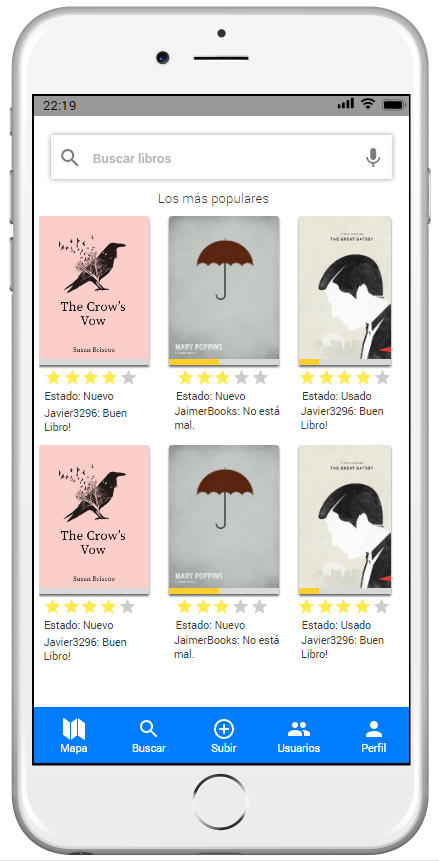
Aquí aparecen los datos de usuario junto con sus lecturas más recientes y los libros que tiene subidos a la plataforma.

Aquí también podremos reservar un libro.



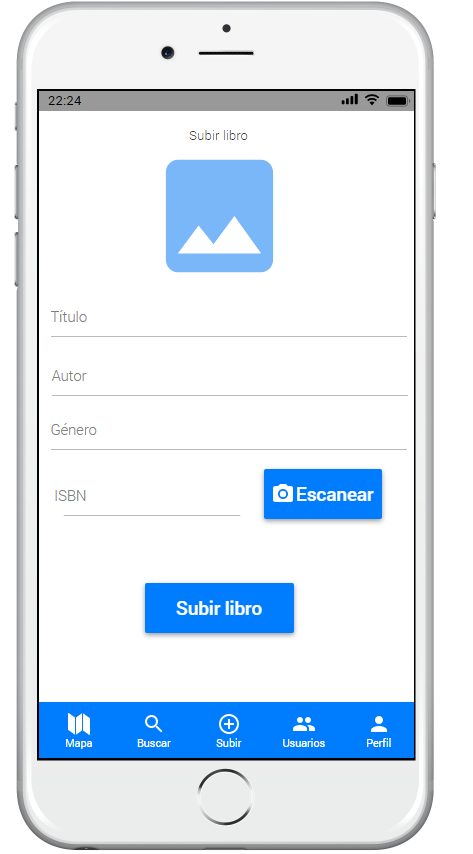
**Buscar**: Esta ventana nos permitirá buscar libros según su título, autor y género, introduciendo los datos de forma escrita o por reconocimiento de voz.

Antes de buscar ningún libro aparecerá una línea de tiempo con los libros más populares, indicando su valoración, estado y su comentario mas reciente como previsualización.



**Subir**: Aquí podremos subir un libro a la plataforma para que puedan disfrutar de él otros usuarios.

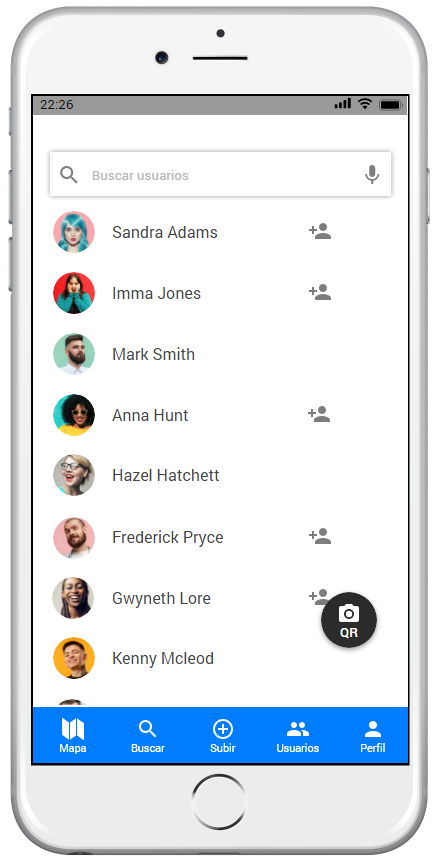
Se deberá agregar una imagen, ya sea de la galería o tomada por la cámara en ese momento, su título, autor, género e ISBN, el cual se podrá escanear con la cámara para mayor fluidez.



**Usuarios**: Ventana donde se pueden buscar usuarios en el sistema para agregarlos, las búsquedas se pueden realizar por voz o por introducción de texto.

Los usuarios no agregados aparecerán con el icono de agregar a su derecha.

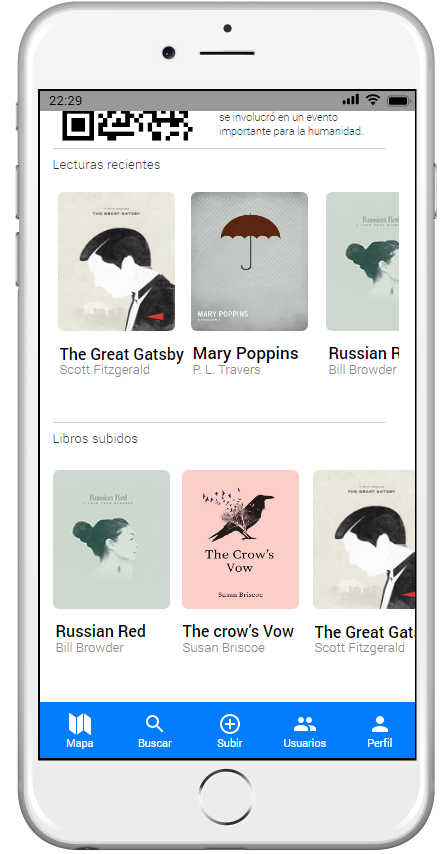
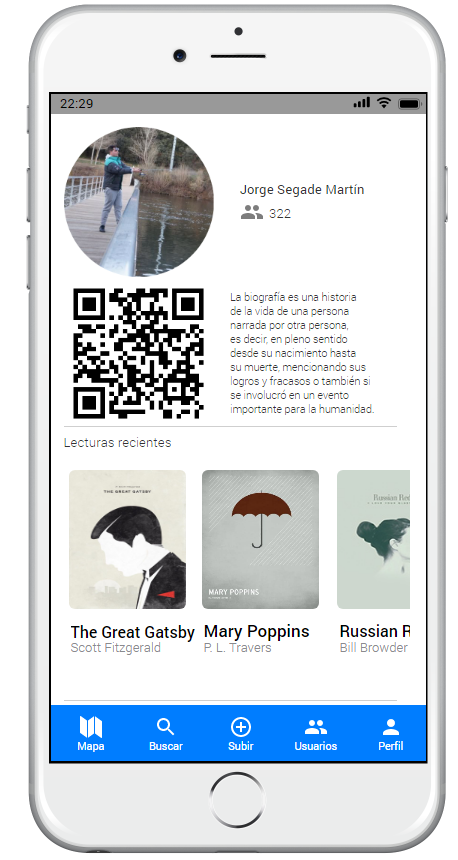
Dispone de un botón de QR para leer el código QR del perfil de un usuario con el que nos encontremos.



**Perfil**: Aquí se puede ver y editar el perfil propio, lo verán los demás usuarios y así podrán saber un poco más unos de otros y fomentar las relaciones interpersonales.

Elementos del perfil:

* Imagen del usuario.
* Nombre del usuario.
* Número de seguidores.
* Código QR del usuario.
* Biografía.
* Carrousel con lecturas recientes del usuario.
* Carrousel con los libros subidos a la plataforma por el usuario.



Vídeo demostración de la interfaz en ios: <https://youtu.be/IyQQmh3IJYA>

Vídeo demostración de la interfaz en Android: <https://youtu.be/ml7F5kM1rQ0>

### 4.2 Diseño de la persistencia de la información.



SQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones; usado por muchos sitios web grandes y populares, como Wikipedia, Google (no para búsquedas), Facebook, Twitter, Flickr, y YouTube.

La diferencia conceptual entre SQL y NoSQL, es que resuelven escenarios completamente diferentes y excluyentes el uno del otro; ya que para lo que resulta ideal SQL, no lo es NoSQL y al revés .

Algunas diferencias:

SQL permite combinar de forma eficiente diferentes tablas para extraer información relacionada, mientras que NoSQL no lo permite o muy limitadamente.

NoSQL permite distribuir grandes cantidades de información; mientras que SQL facilita distribuir bases de datos relacionales.

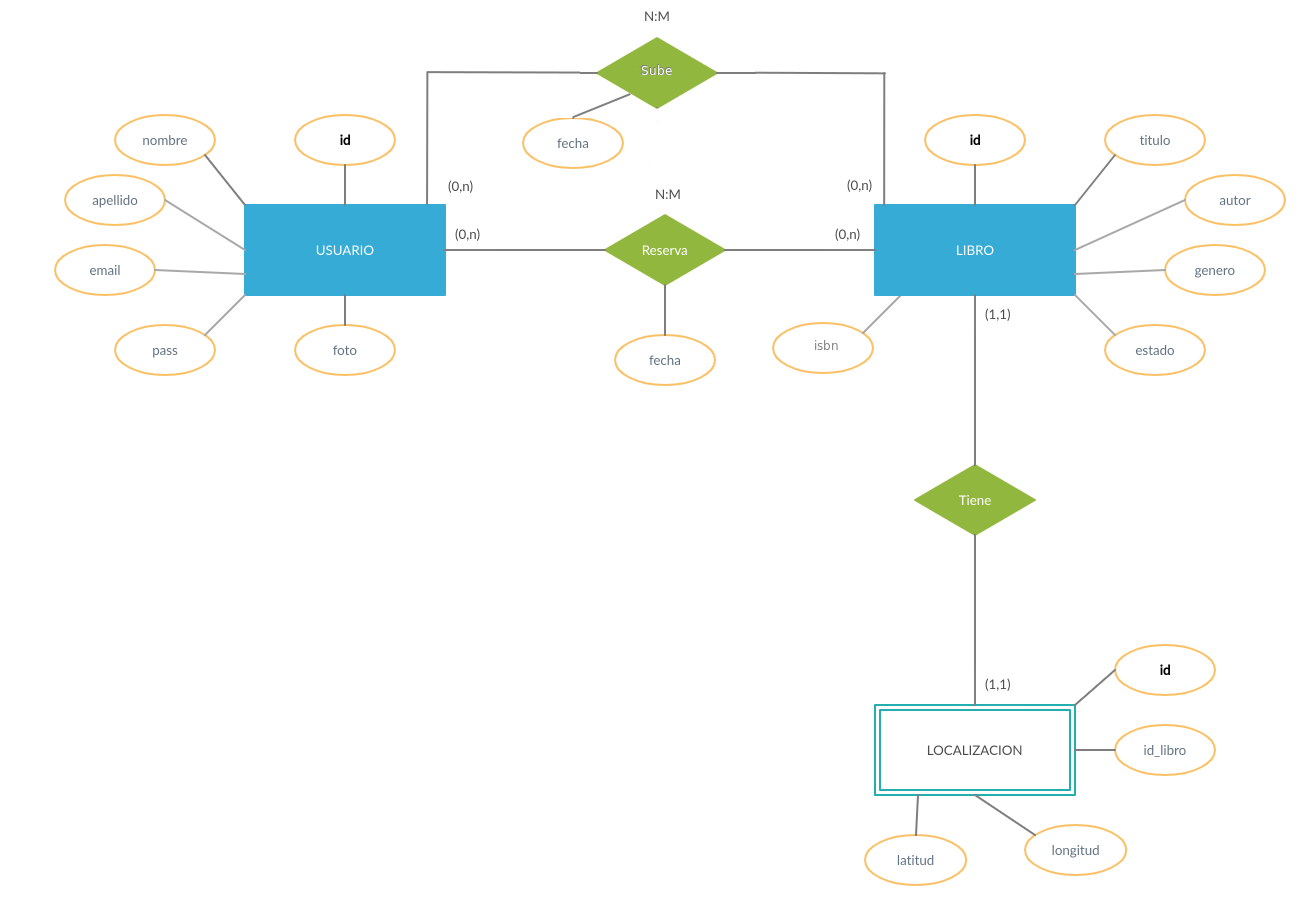
SQL permite gestionar los datos junto con las relaciones existentes entre ellos; en NoSQL no existe este tipo de utilidades .

Es por ello que se ha decidido implantar un modelo de base de datos relacional

y con ello SQL.

Fuente: <https://www.facilcloud.com/noticias/sql-vs-nosql-which-one-should-i-use/>

Diagrama Entidad-Relación



Estructura de la base de datos

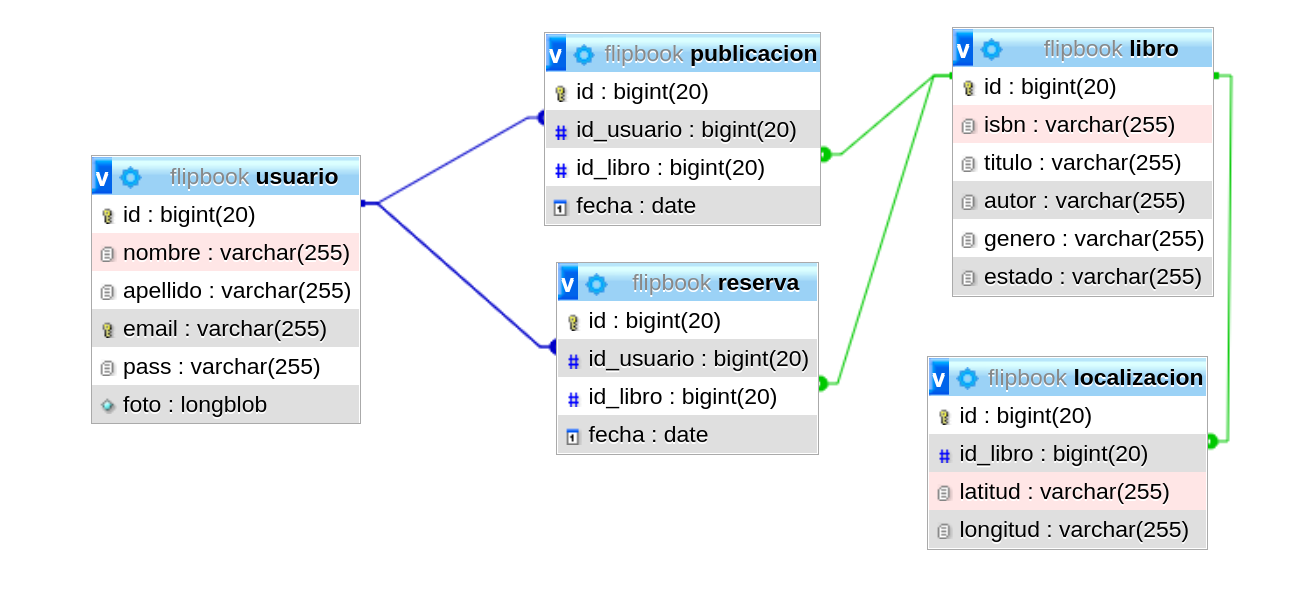
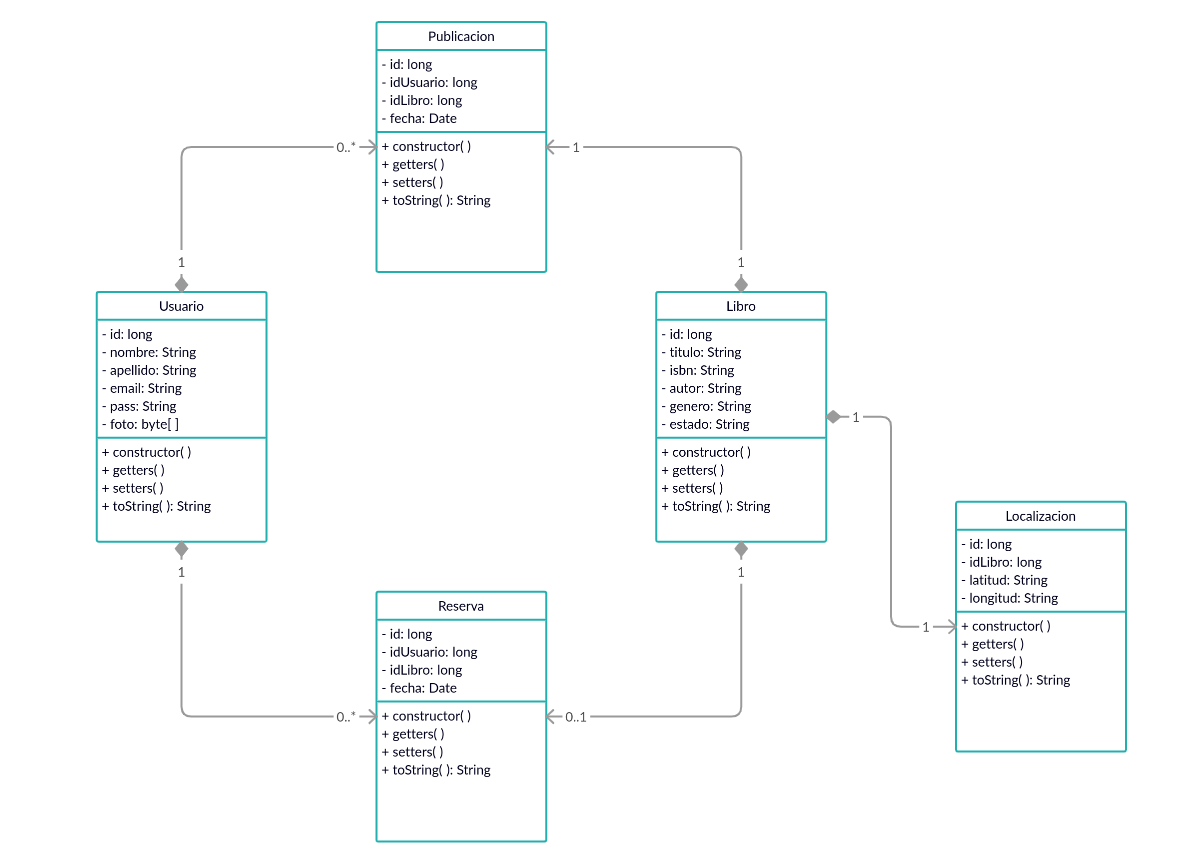


Diagrama de Clases

En ingeniería de software, un diagrama de clases en Lenguaje Unificado de Modelado (UML) es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos) y las relaciones entre los objetos.



Un usuario puede reservar uno o varios libros.

Un libro puede ser reservado por un usuario.

Un usuario puede subir uno o varios libros.

Un libro puede ser subido por un usuario.

Una reserva se puede hacer de un libro.

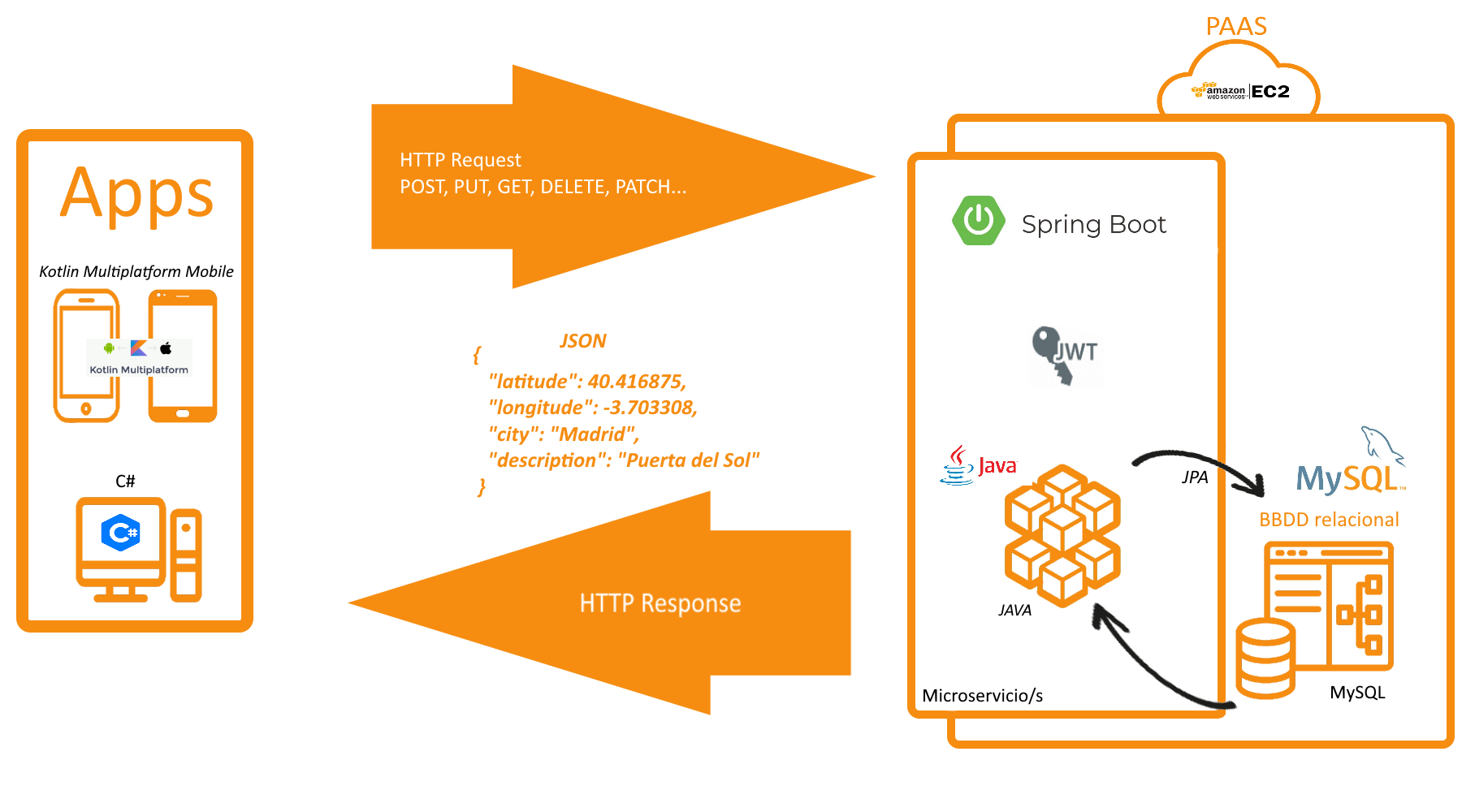
Un libro puede tener una o ninguna reserva .

Un libro tiene una localización.

Una localización pertenece a un libro.

Fuente: <https://es.wikipedia.org/wiki/Diagrama_de_clases>

### 4.3 Diseño de la arquitectura del sistema.



# 5. IMPLEMENTACIÓN DE LA SOLUCIÓN.

### 5.1 Justificación tecnológica.

Análisis tecnológico

Dispositivos móviles.

Cuando se lanzó Google Play en 2008 creando un competidor para la App Store, los desarrolladores se vieron preocupados por un desafío simple: cómo crear aplicaciones que puedan ejecutarse en iOS y Android sin duplicar todo el trabajo.

Existe una solución para esto, Kotlin Multiplatform Mobile, es una tecnología bastante nueva, pero está respaldada por una de las empresas de desarrollo más importantes de la actualidad, JetBrains.



Con Kotlin Multiplatform, podemos generar una biblioteca que se ejecuta de forma nativa en plataformas iOS y Android.

Esto es un paso importante en la batalla entre iOS y Android. Hasta ahora, ejecutar código de forma nativa en estas dos plataformas requiere bibliotecas C ++ complejas y una alta complejidad de codificación, lo que no es una buena idea para invertir en aplicaciones simples, esto ha cambiado con Kotlin Multiplatform. Ahora, cualquier persona con un conocimiento básico de Kotlin (utilizado para crear aplicaciones Android) puede escribir fragmentos de código que se pueden ejecutar en Android e iOS y es por ello por lo que lo he elegido.

Como otras opciones posibles estaban, Ionic, React Native o Flutter pero me he decidido por Kotlin por que creo que puede ser una opción algo menos costosa.

Sistema del administrador(Escritorio).

He elegido C# como lenguaje para el sistema del administrador ya que permite crear interfaces vistosas y programas robustos. Además las peticiones HTTP son muy sencillas.

Microsoft .NET Framework, que proporciona acceso a los elementos de la interfaz de Windows nativas y como interfaz de programación de aplicación gráfica Windows Forms ya que es una tecnología madura y estable.

Como otras posibles opciones estaban Windows Presentation Foundation(WPF) y Java Swing, la primera más vistosa pero la he descartado ya que me llevaría mas tiempo y esta última tiene su parecido visual con la elegida pero he preferido hacerla en C# y así utilizar las diferentes tecnologías aprendidas.

Servidor.

SpringBoot permite configurar contenedores o servidores web Apache Tomcat, estructurar la aplicación, tener todas las dependencias en un formato legible en un archivo POM para la gestión Maven o conectar con la base de datos, todo en un momento. SpringBoot permite comenzar a desarrollar microservicios casi de inmediato e implementar sin complicaciones, incorpora JPA(Java Persistence Api) como ORM(Object Relational Mapping) así que, ahorraría la mayoría de DML(Data Manipulation Language) y DDL(Data Description Language) en base de datos y podría usar JPQL(Java Persistence Query Language) cuando necesitase una consulta más específica.

Como base de datos he dudado entre PostgreSQL y MySQL pero me he decidido por MySQL ya que es mas sencilla de configurar, aunque, daría igual utilizar una u otra, ya que, gracias a JPA para el programador es transparente la base de datos que haya detrás. Es por esto que lo he elegido como framework para el backend.

Otra opción posible sería hacer un web service con PHP y Mysql pero lo he descartado ya que no tendría la misma seguridad a parte de JWT y me quedaría un código menos limpio.

### 5.2 Aspectos esenciales de la implementación.

### 5.3 Desarrollo de la funcionalidad indicada por el tutor/a.

# 6. TESTEO Y PRUEBAS DE LA SOLUCIÓN.

### 6.1 Plan de pruebas (unitarias, integración, sistema y usuarios).

### 6.2 Solución a problemas encontrados.

# 7. LANZAMIENTO Y PUESTA EN MARCHA.

### 7.1 Aspectos relevantes del despliegue y puesta en marcha del sistema.

### 7.2 Manual de uso.

# 8. VALORACIÓN Y CONCLUSIONES.

# 9. BIBLIOGRAFÍA Y RECURSOS UTILIZADOS.