PROYECTO

FOCUSON

AUTOR: Jorge Alcaraz Bravo

Málaga, 11 de junio de 2018

ÍNDICE

Capítulo 1
INTRODUCCIÓN

1.1. FINALIDAD DE LA APLICACIÓN

Esta aplicación está enfocada al aumento de la productividad y concentración en el entorno de trabajo o estudio, sobre todo para programadores. Se ha reunido en una sola aplicación web la posibilidad de controlar la distracción acústica del ambiente, una herramienta de trabajo atractiva, métodos para disminuir el estrés y una herramienta de organización.

El control del ambiente acústico se ha conseguido gracias al uso de sonidos ambiente de distintos tipos, que están dirigidos a disminuir, de forma neutra, el efecto de los sonidos externos.

Por otro lado, para favorecer el correcto entorno de trabajo, se dota a la aplicación de un editor de texto propio, convirtiendo la aplicación en una herramienta de uso laboral autosuficiente que evita toda distracción. El editor está enfocado al desarrollo de aplicaciones, ofreciendo soporte de un gran número de lenguajes.

Los métodos de disminución de estrés se han proporcionado incorporando instrucciones sobre técnicas de relajación. Se han escogido técnicas de distinta dificultad y formato, dando a elegir al usuario la realización de estas dependiendo del tiempo y espacio que posea.

La posibilidad de organización se ofrece mediante una agenda en la que el usuario podrá anotar las tareas pendientes y consultarlo sin necesidad de utilizar terceras aplicaciones, evitando así de nuevo las distracciones.

A esta aplicación web, se añade una aplicación Android que está conectada con la herramienta de audio. Permitirá modificar fácil y rápidamente los volúmenes o sonidos activos de la lista de reproducción actualmente seleccionada en Angular.

1.2. TECNOLOGÍAS USADAS

Con respecto a las tecnologías usadas, en lo referente al Front-end, se ha utilizado Angular para el formato web, con lo que se ha podido desarrollar una aplicación web de una sola página (SPA; Single Page Application), cuyas ventajas serán analizadas más adelante. Junto a esto, se ha empleado Angular Material para el diseño.

La versión móvil del Front-end se ha desarrollado en Android, siguiendo también el patrón de diseño marcado por Material Design.

Por otro lado, para el back-end se ha empleado Spring, en concreto Spring Boot, desarrollando así un servicio REST (Representational State Transfer).

Por último, para la capa de persistencia, se ha utilizado Hibernate, en el servicio REST, para gestionar la comunicación con la base de datos en MySQL y, por otro lado, Firebase para aquella información compartida entre Angular y Android, ya que se necesitaba la comunicación a tiempo real.

En referencia a los sistemas operativos que se han utilizado, el proyecto ha sido desarrollado tanto en el sistema operativo Linux, utilizando Visual Studio Code, Eclipse y PhpMyAdmin, como en Windows, con herramientas como Android Studio y Firebase, este último desde su gestor propio.

Para el despliegue de la aplicación, será necesario un servidor Tomcat (por el uso de Java en el Back-end) en el que serán desplegados el Front-end y el Back-end.

Capítulo 2
CONCEPTOS BÁSICOS

Para comenzar, se va a distinguir entre una aplicación web, como lo es el presente proyecto, y una página web.

Una aplicación web es aquella que permite a un usuario realizar acciones a través de un navegador. Por el contrario, una página web, es un conjunto de información estática, por lo su uso está limitado a la presentación de información.

Para el desarrollo de una aplicación web profesional, será beneficioso utilizar frameworks. Un framework es una herramienta que permite desarrollar aplicaciones escalables y, en el caso de Angular, adaptables a distintos formatos. Además, el framework permitirá al programador desarrollar funcionalidades más complejas o avanzadas de las que podría realizar sin esta herramienta. La dificultad radica en el aprendizaje que requiere un framework, ya que dota al programador de un gran número de posibilidades según sus propios conceptos y técnicas, aunque de la misma forma ofrece una forma de trabajo sobre la que construir un proyecto. A su vez, evita el proceso de selección de distintas bibliotecas incluyendo “opiniones” por defecto, que ahorrarán tomas de decisiones.

Se tratarán a continuación los distintos frameworks utilizados en FocusOn, comenzando por la aplicación web, desde el Front-end hasta la capa de persistencia, y continuando con la aplicación de Android.

2.1. ANGULAR

Angular es un framework basado en TypeScript/JavaScript que aglutina todas las tecnologías básicas necesarias para el desarrollo del Front-end (HTML, JS y CSS), abstrayendo algunas de las necesidades y permitiendo ampliar los límites de lo que supone la programación en el navegador.

Está especialmente indicado para realizar las conocidas como SPA (Single Page Application) que no requieren de recargas del navegador para la navegación por la web. Además una gran parte de la capa de negocio de la aplicación es procesada directamente en el cliente (en este caso un navegador), lo cual supone liberar de una importante carga de procesamiento al servidor.

La arquitectura a nivel de servidor más común de las SPA es la de un servicio RESTful, que por su propia naturaleza “Stateless” combina muy bien con las SPA, delegando todo el manejo del “Estado” al cliente, mientras que el servidor únicamente se encarga de procesar peticiones de listado/guardado/eliminación de recursos.

Esta delegación de carga de trabajo en el cliente es insignificante a nivel individual, pero supone una enorme ventaja para las empresas que deban implementar sus servidores, abaratando costes a nivel de hardware necesario.

Angular, como la mayoría de frameworks refuerza el uso de patrones de diseño. Concretamente, ya que se lleva gran parte de la lógica del servidor, el cual a menudo implementaba una arquitectura Modelo, Vista, Controlador (MVC), Angular hace lo propio en el cliente, además de automatizar la Inyección de Dependencias (DI).

La principal ventaja del uso de patrones de diseño, es el incremento en escalabilidad que recibe el proyecto, permitiendo que una aplicación pase por más iteraciones de desarrollo antes de llegar a ser demasiado grande y sea necesario una refactorización.

Además, Angular utiliza Typescript, lo que permite crear un código más consistente gracias a un tipado fuerte, trayendo consigo un código más limpio y escalable que con Javascript, tanto con respecto a la reutilización del código como de inteligibilidad.

En Angular se trabaja con Componentes. Un componente es un fragmento de HTML, JS y CSS encapsulados. Este fragmento puede ser más o menos complejo, ya que depende del programador el decidir qué va a contener un componente. Los componentes permiten el ser reutilizados, tanto en un mismo proyecto como de un proyecto a otro, por lo que se pueden encontrar componentes ya creados que son útiles para el proyecto a desarrollar y que ya existan, ahorrando así tiempo. Los componentes vendrán identificados mediante etiquetas HTML. Los componentes a menudo actúan como C de MVC (controladores), y los “Servicios” (componentes sin vista, inyectados al entorno mediante inyección de dependencias) manejan la conexión con la base de datos y mantienen el Estado.

Por otro lado, Angular es una herramienta en la que se puede confiar a largo plazo gracias al respaldo de una gran empresa como Google y que, si incluye alguna actualización que conlleve incompatibilidad, permitirá una fácil migración, aunque hasta ahora siempre ha habido retrocompatibilidad.

Por último, gracias a la popularidad de este framework, existen IDEs que añaden herramientas para trabajar cómodamente, como es VS Code, con plugins como TSLint para la corrección de sintaxis de TypeScript. Además, WebPack (transpilador de TypeScript a JavaScript) corrige a nivel de pre-ejecución, pese a ser TypeScript un lenguaje interpretado.

2.2. SPRING

Spring es un framework destinado al desarrollo de programas con el lenguaje Java. Es un framework no intrusivo, ya que no afecta a la interpretación del código, sino que se incluye mediante anotaciones. Las anotaciones son indicaciones que poseen la estructura “@nombredeanotacion”. Spring contiene un gran número de anotaciones para indicar al framework que la clase y objeto asociado a la anotación es, por ejemplo, un “Bean”, una “Entity”, etc, conceptos que se explicarán a continuación.

Gracias al uso de las anotaciones, si en el futuro se decidiera no utilizar Spring en la aplicación, no sería necesaria modificar el código para la adaptación de este a otra tecnología.

2.2.1. Anotaciones a destacar

Las anotaciones utilizadas en la aplicación son:

2.2.1.1. CrossOrigin

Indica que se permite el CrossOrigin (CORS) en la clase que la incluya. El CORS consiste en, mediante el uso de cabeceras en las peticiones, permitir el acceso desde un servidor a otro.

2.2.1.2. RestController

Indica que el componente que lo incluye es un controlador y le permite responder a las peticiones.

2.2.1.3. RequestMapping

Permite mapear una clase para que el cliente pueda realizar peticiones. Se debe indicar a qué ruta va a corresponder. Un ejemplo sería: @RequestMapping("/files").

2.2.1.4. Autowired

Esta anotación permite hacer referencia a un objeto ya creado por Spring, favoreciendo el ahorro de memoria y evitando la generación de basura que frecuentemente ocurre en las aplicaciones desarrolladas en Java.

2.2.1.5. Scope

Gestiona el alcance de una instancia dentro de la aplicación.

2.2.1.6. Transactional

Define el alcance de las gestiones de transmisión de datos con la base de datos.

2.2.1.7. Repository

Indica que una clase es un DAO (Data Access Object) y va a contener toda la lógica relacionada con el acceso a la información de la base de datos. Un ejemplo sería: @Repository(value="filesDao").

2.2.1.8. Entity

Esta anotación indica que una clase es una entidad, es decir, es un objeto que representa una tabla en la base de datos.

2.2.1.9. Table

Acompaña a la anotación “Entity” e indica a qué tabla de la base de datos hace referencia la clase que la posee. Un ejemplo sería: @Table(name="files"), siendo “files” el nombre de la tabla en la base de datos.

2.2.1.10. Id

Esta anotación pertenece a uno de las propiedades de una clase entidad. Indica cuál de sus propiedades es la clave primaria de la tabla. Esta id es normalmente un campo numérico y autoincrementable, ya que esto permite a Hibernate gestionar automáticamente la creación de nuevos registros.

2.2.1.11. GeneratedValue

Acompaña a la anotación id para indicar el método de generación de claves.

2.2.1.12. Column

Esta anotación indica que una propiedad es un campo de la tabla a la que representa la entidad. Un ejemplo sería: @Column(name="ID"), donde ID es el nombre del campo de la tabla. Existe una anotación similar, @JoinColumn, que se utiliza con la misma finalidad pero en las propiedades que pertenezcan a una relación @ManyToOne.

2.2.1.13. ManyToOne, OneToMany, OneToOne y ManyToMany

Estas anotaciones representan las relaciones entre campos de distintas tablas.

2.2.1.14. Service

Indica que una clase es un servicio, es decir, pertenece a la capa que comunica el controlador con la capa de persistencia. Un ejemplo sería: @Service(value = "filesService").

2.2.1.15. SpringBootApplication

Indica cuál es la clase de arranque de la aplicación, además de incluir propiedades que permiten indicar qué paquetes son los que revisar al mapear las clases.

2.2.1.16. Bean

Permite que Spring gestione un objeto generado en la configuración.

2.2.2. Módulos

Spring posee un gran número de distintos módulos, por lo que requiere de un estudio previo de dichos módulos para saber cuáles son acordes al proyecto a desarrollar. Estos módulos se dividen por capas, siendo estas Web, Común, de Servicio y de Datos. Cada una de ellas cubren las funcionalidades necesarias en las distintas capas de una aplicación web.

2.2.3 Project Foundation

En el presente proyecto, se ha utilizado lo que en Spring se conoce como un “Project Foundation”, lo cual significa que es un proyecto ya generado que incluye una configuración adaptada a las dependencias elegidas por el programador al generar el proyecto. El más destacado se conoce como Spring Boot, el cual es el empleado en FocusOn. Para generarlo, basta con indicar en el siguiente enlace https://start.spring.io/ las dependencias y descargará la plantilla del proyecto sobre el que se desarrollará la aplicación. El hecho de no tener que invertir tiempo en la configuración de cada proyecto, significa un ahorro de recursos beneficioso para la empresa o persona, y facilita la construcción del proyecto a las personas que se están iniciando en esta herramienta.

2.3. HIBERNATE ORM

Hibernate es un framework enfocado en el mapeo objeto-relacional para Java. Tiene como principal ventaja la facilidad de migrar de una base de datos a otra sin necesidad de modificar el código (o con mínimas alteraciones), reduciendo significativamente el coste que supondría este hecho.

Además de esto, hibernate trae consigo unas herramientas que facilitan la creación de POJOS y la conexión con la base de datos, pero esto no es relevante si se está utilizando Spring, ya que este lleva consigo la configuración de la conexión utilizando un simple archivo “properties”.

2.4. ANDROID

Android es un sistema operativo para móviles desarrollado por Google, basado en una versión de Linux. Normalmente, sus aplicaciones se desarrollan en Java, aunque puede combinarse con C/C++, o incluso utilizarse Go.

La estructura de una aplicación de Android se puede reducir en layouts, archivos xml que contienen la estructura de la vista de la actividad, y en clases Java, que contienen la capa

lógica. Además, en este caso concreto se han añadido unas clases Java conocidas como “Entidades”, que representan los objetos que se encuentran en la base de datos y permite trabajar con estos más fácilmente.

2.5. FIREBASE

Firebase es un Backend como un Servicio (BaaS - Backend as a Service). En referente a su funcionalidad como base de datos, se define como una base de datos no relacional (noSQL). Permite el manejo de datos a tiempo real sin necesidad de peticiones. La conexión se establece mediante WebSocket en lugar de una conexión HTTP, siendo más rápido. No hay que realizar distintas conexiones para cada petición, ya que es suficiente con una única conexión. Además, Firebase actualiza los datos de la aplicación en cuanto se produce alguna modificación en los datos almacenados.

Otra ventaja de Firebase es que permite trabajar con datos binarios, por lo que puede almacenar archivos en su nube.

Además, en cuanto a mantenimiento y coste, no es necesario tener que preparar un servidor, ya que Firebase hostea sus propios servicios y te da acceso a la nube. En caso de aumentar el tamaño del proyecto habría que estudiar los distintos planes de pago. Pero para el caso de esta aplicación sería gratuito.

Capítulo 3
ESTRUCTURA DE LA APLICACIÓN

3.1. BACK-END

Se divide en 4 capas:

3.1.1. Configuración (Config)

Incluye aquellos archivos necesarios para la configuración del proyecto, en concreto, para la gestión del token (JWTFilter) y para la autorización y gestión de CORS (Cross Origin Resourcing Sharing).

Con respecto al token, la clase JwtFilter contiene un método denominado doFilter. Este método proviene de la clase GenericFilterBean. Consiste en un método a través del cual pasará las peticiones y respuestas entre cliente y back-end, para las distintas autorizaciones o comprobaciones a realizar. Este método formará parte de una cadena (FilterChain) que indicará el número y orden de filtros a recorrer. En el caso de este proyecto, se incluyen un filtro que comprobará la veracidad del token enviado por el cliente, el de la clase presente, y un método para la gestión de CORS, que se desarrollará a continuación. Al final de este método se incluye otra llamada al siguiente método doFilter de la cadena que invoca al método doFilter de la clase SimpleCORSFilter.

Por otro lado, la clase SimpleCORSFilter incluye el método doFilter que configura los headers a enviar en la respuesta al cliente y que permitirán la conexión de este con el servidor, evitando los problemas provocados por el intercambio de información entre distintos servidores.

3.1.2. Persistencia (Persistence)

Es la capa más próxima a la base de datos.

Por un lado, contendrá lo conocido como entidades, que serán aquellas clases que contengan el esquema de los objetos que representan las tablas que se encuentran en la base de datos.

Por otro, se encontrará el DAO (data access object), que contendrá toda la parte de Java que tiene como objetivo la transmisión de datos desde y hacia la base de datos. En FocusOn se ha generado una clase abstracta, que incluirá los métodos básicos. Por cada entidad, se generará una clase DAO, la cual extenderá de la clase abstracta. Cada una de estas clases DAO contendrán métodos con los que gestionar las consultas hechas por el

usuario, los cuales pueden ser tratados por los métodos ya contenidos en la clase abstracta u otros implementados en la misma clase.

3.1.3. Servicio (Service)

Provee la parte lógica que engloba el tratamiento de datos que se envían o reciben de la base de datos. Además, proporciona más seguridad al proporcionar una capa intermedia entre el cliente y el DAO.

Cada entidad tendrá asociada una clase servicio. Cada clase servicio estará relacionado con el DAO que también se asocia a dicha entidad.

3.1.4. Controlador (Controller)

Contiene la capa dedicada a la recepción, enrutamiento y respuesta a las peticiones del cliente.

Figura 3.1.- *Flujo de información en el REST*

3.2. FRONT-END

A nivel de arquitectura, sigue la estructura dictada por las guías de diseño de Angular/Angular CLI, con ciertas mejoradas que se han añadido para facilitar la escalabilidad.

La aplicación se compone de un Feature Module por cada sección en la barra de menú, además de un Shared Module que importa todos los módulos, y el cual contiene todos los componentes y servicios que son necesarios de forma global.

Cada Feature Module divide la aplicación en un conjunto semántico de funcionalidades a gran escala, y cada componente lo divide a pequeña escala dentro de cada módulo. Cada uno de éstos componentes son en realidad un trozo de HTML, CSS y JS que componen la web. Angular sigue el patrón de diseño MVC, y los componentes serían los controladores.

A su vez existen en la aplicación una serie de Services encargados de la mayor parte de la lógica de negocio, proporcionando al controlador la información que requieren.

Concretamente destacar el servicio SoundPlayerService el cual maneja en segundo plano todo lo relacionado con la reproducción de sonidos y los perfiles de los mismos. Esto permite que aun moviéndose por la web a otras secciones, el usuario pueda seguir escuchando los sonidos que ha elegido desde la sección correspondiente.

En el tema de dependencias, se hace uso de NPM (Node Package Manager) para la inclusión y manejo de dependencias externas al servidor. El archivo de configuración de NPM se encuentra en la raíz del proyecto con el nombre de package.json.

El proyecto también cuenta con Angular CLI, un gestor de comandos de Angular que automatiza la mayoría de procesos repetitivos que se realizan durante el desarrollo de un proyecto, como la creación de componentes y servicios, su declaración en el módulo correspondiente, etc. Además éste instala WebPack, un Transpilador/Module Bundler, el cual se hace necesario a la hora de pasar el proyecto a producción, ya que transpila de TypeScript a JavaScript todos los archivos, y los agrupa en unos pocos archivos llamados Bundles, los cuales serán los que sirva el servidor web. De lo contrario, las peticiones de carga de la web serían muy largas (un proyecto simple de Angular puede tener miles de archivos debido a las dependencias).

3.3. APLICACIÓN ANDROID

Consta de dos actividades, con sus respectivas vistas, y varios objetos, los cuales permitirán tratar los datos recibidos y enviados de la base de datos.

La actividad principal presenta un lector de código QR, que recogerá la información del código presentado en la aplicación web. Si el código QR es correcto (es decir, identifica a un usuario), se ejecutará la siguiente actividad, la cual incluye un desplegable que permite seleccionar la lista que se desea modificar y un listado de audios con sus volúmenes asociados.

Capítulo 4
ANÁLISIS DE LOS REQUISITOS DEL SISTEMA

No existen unos requisitos mínimos del sistema con respecto a la aplicación web ya que, por parte del usuario, solo será necesario tener acceso a internet y un navegador web, por lo que tampoco requiere de una instalación previa. Esto presenta una ventaja gran frente a las aplicaciones de escritorio, ya que es fácilmente accesible.

Por otro lado, para la aplicación en Android, el usuario sí deberá instalarse la APK en un dispositivo móvil con el sistema operativo Android (o su derivado MIUI) con una versión 27 o superior.

Para el despliegue de la aplicación por parte del propietario, será necesario tener activo un servidor Tomcat que contendrá tanto el Front-end como el Back-end. Si en un futuro fuera necesario el uso de servidores separados, el REST se encuentra preparado para ello.

La aplicación en Android y Firebase no necesitan servidores.

Capítulo 5
MODELO ENTIDAD RELACIÓN

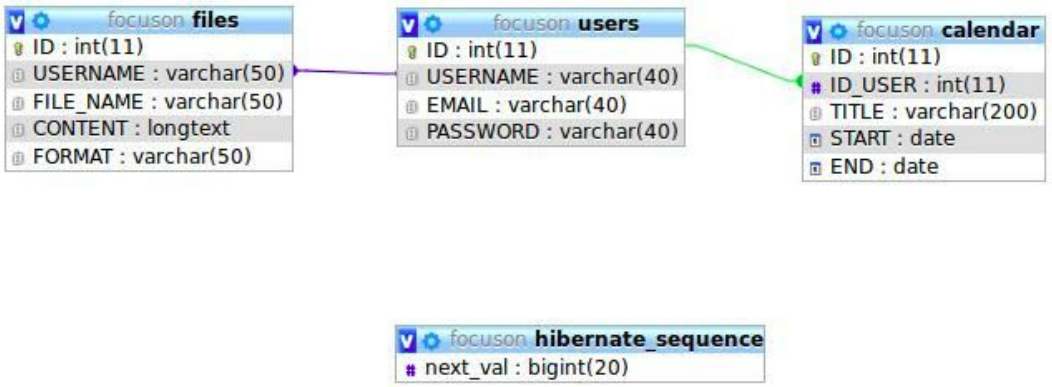

Figura 6.1.- *Modelo Entidad Relación*

Aunque no se incluya en el modelo Entidad Relación, también cabe destacar la parte de persistencia perteneciente a la base de datos de Firebase al ser una base de datos no relacional (noSQL), en la cual se recoge la estructura de datos de la funcionalidad de Audio de la aplicación.

Capítulo 6
DIAGRAMA DE FLUJO DE DATOS

6.0. DIAGRAMA DE CONTEXTO

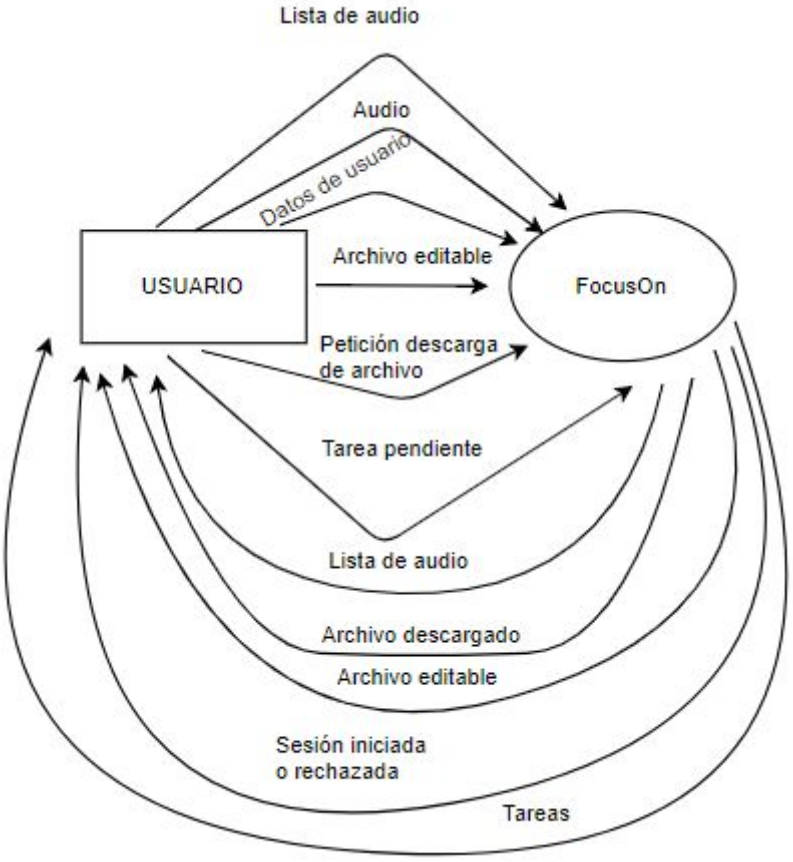

Figura 7.1.- *Diagrama de flujo de datos general*

6.1. DIAGRAMA DE FOCUSON

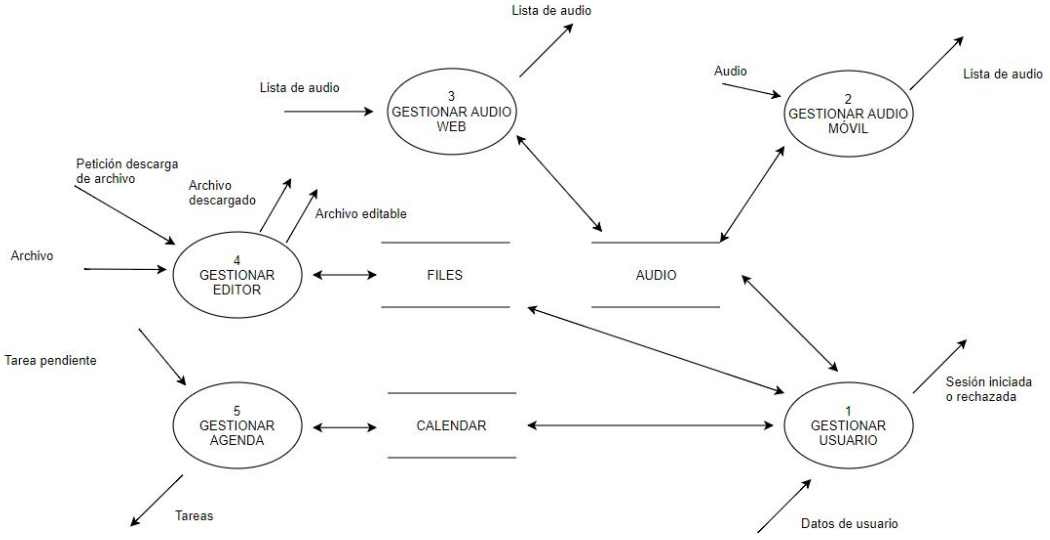

Figura 7.2.- *Diagrama de flujo de datos general*

6.1. GESTIONAR USUARIO

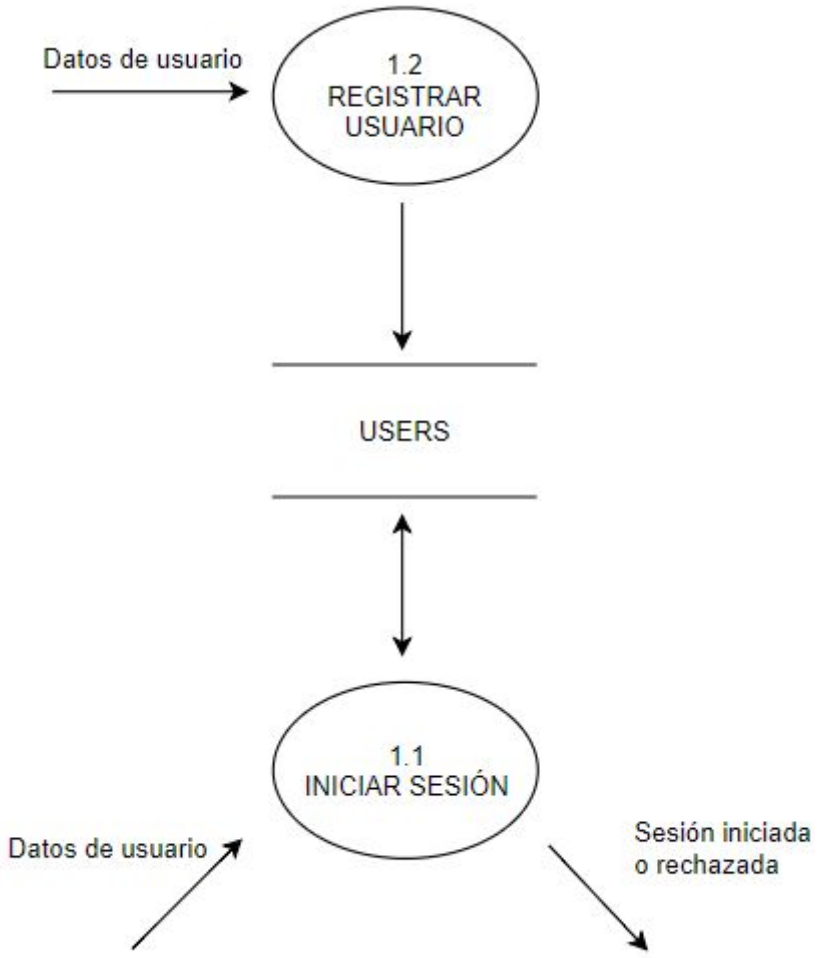

Figura 7.3.- *Diagrama de flujo de datos de Gestionar usuario*

6.2. GESTIONAR AUDIO MÓVIL

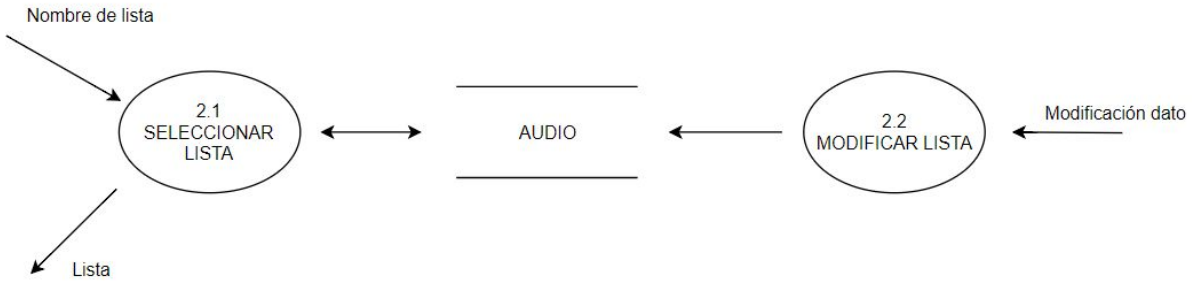

Figura 7.4.- *Diagrama de flujo de datos de Gestionar Audio Móvil*

6.3. GESTIONAR AUDIO WEB

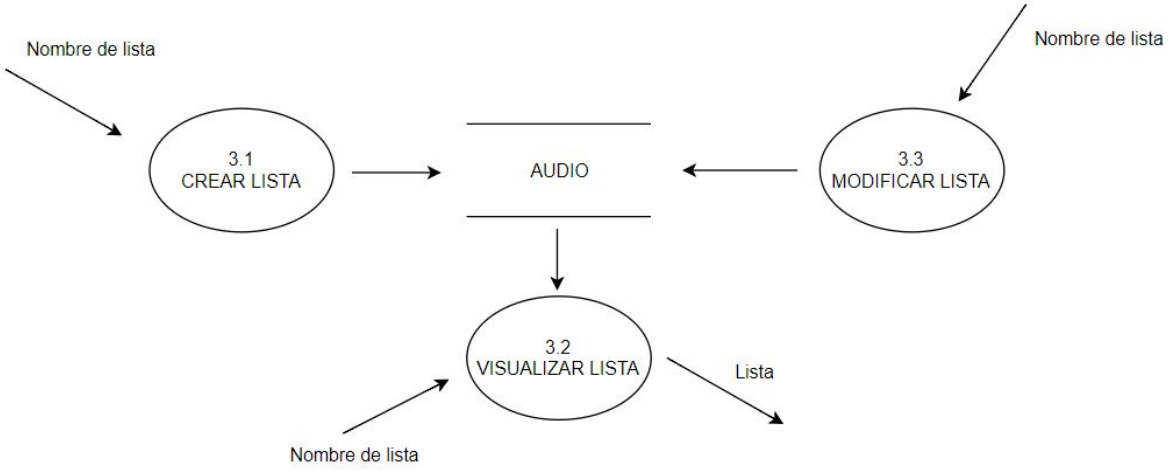

Figura 7.5.- *Diagrama de flujo de datos de Gestionar Audio Web*

6.4. GESTIONAR EDITOR

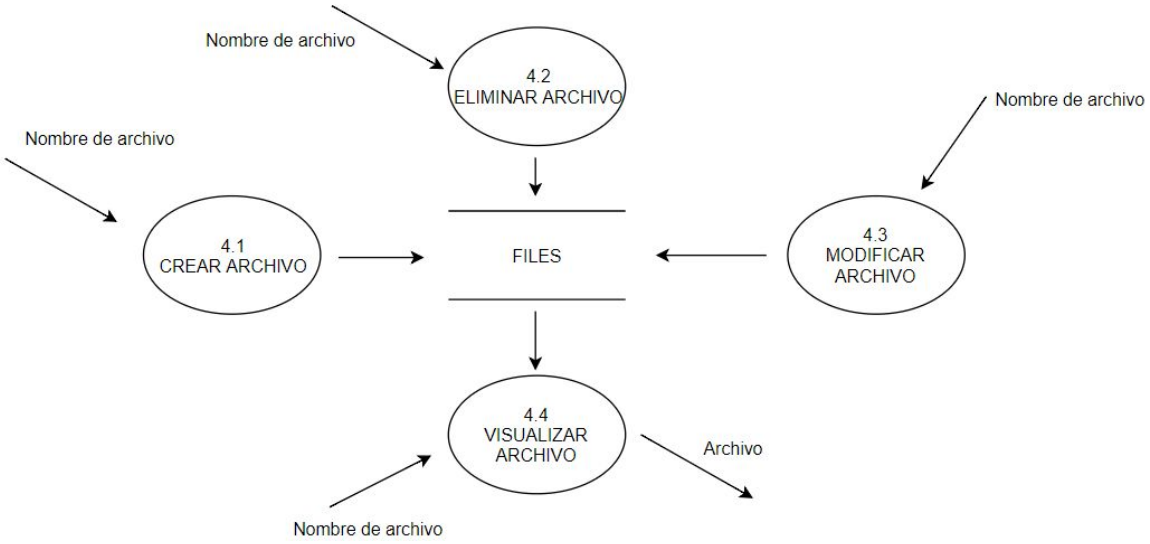

Figura 7.6.- *Diagrama de flujo de datos de Gestionar Editor*

6.5. GESTIONAR AGENDA

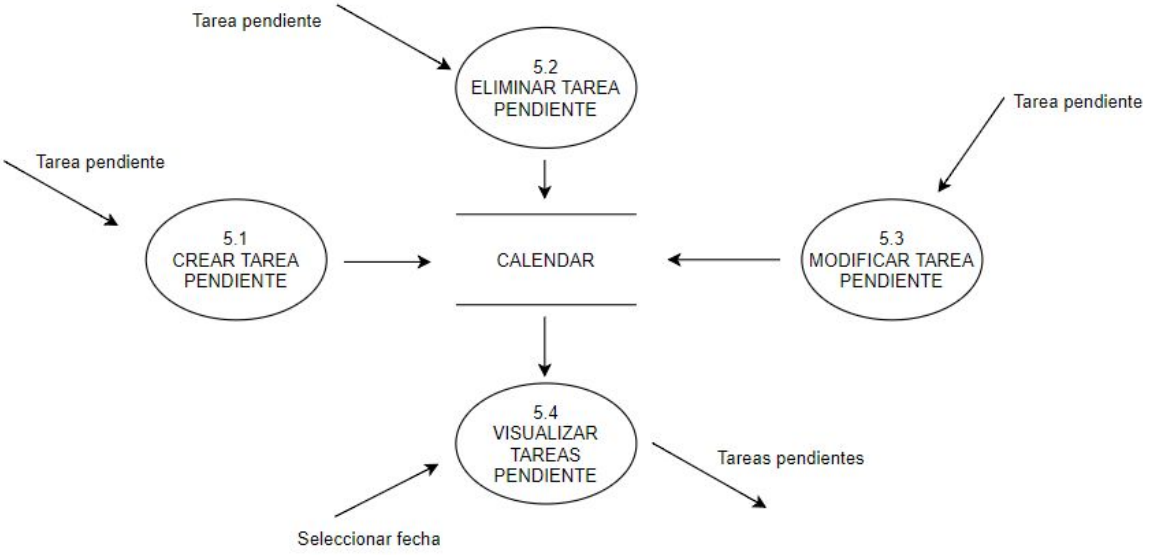

Figura 7.7.- *Diagrama de flujo de datos de Gestionar Editor*

6.6. DICCIONARIO DE DATOS

Lista de audio = {Profile}

{Profile} = ProfileName + {Audio} + {Audio} + {Audio} + {Audio} + {Audio} + {Audio} + {Audio} + {Audio}

{Audio} = AudioVolume + FileName + IsPlaying

Archivo = {Files}

{Files} = @Id + Username + FileName + Content + Format

Datos de usuario = {User}

{User} = @Id + Username + Correo + Password

Tareas = {Calendar}

{Calendar} = @Id + User + Title + Start + End

Capítulo 7
CASOS DE USO

7.1. LISTAS DE AUDIO

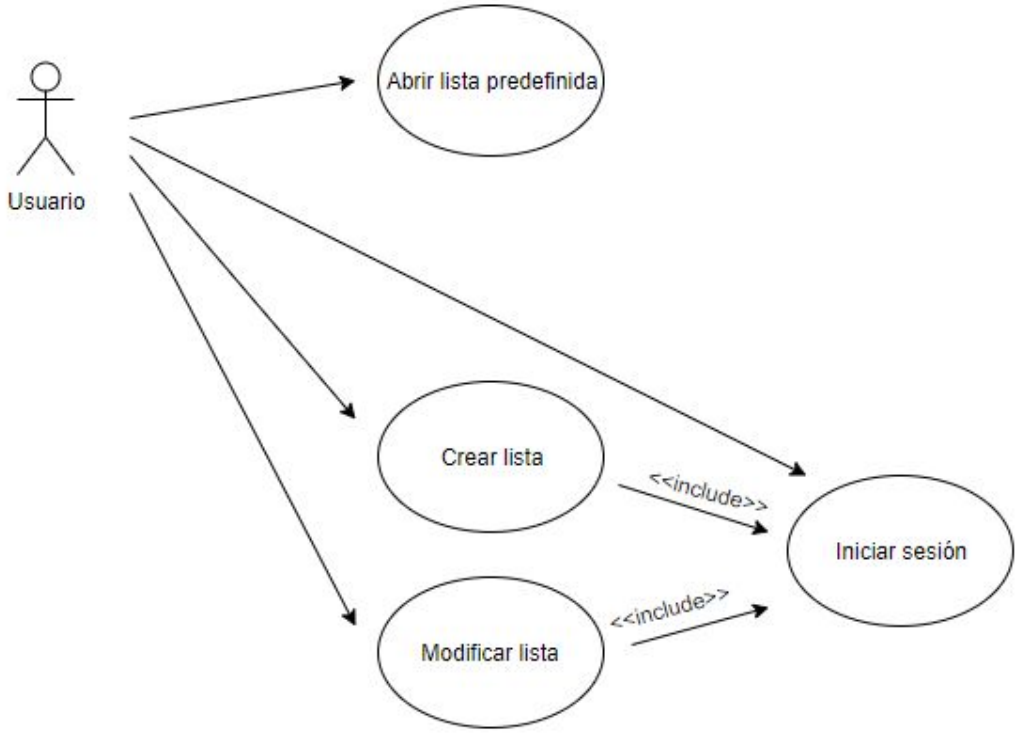

Figura 8.1.- *Caso de uso de Listas de audio*

7.2. EDITOR

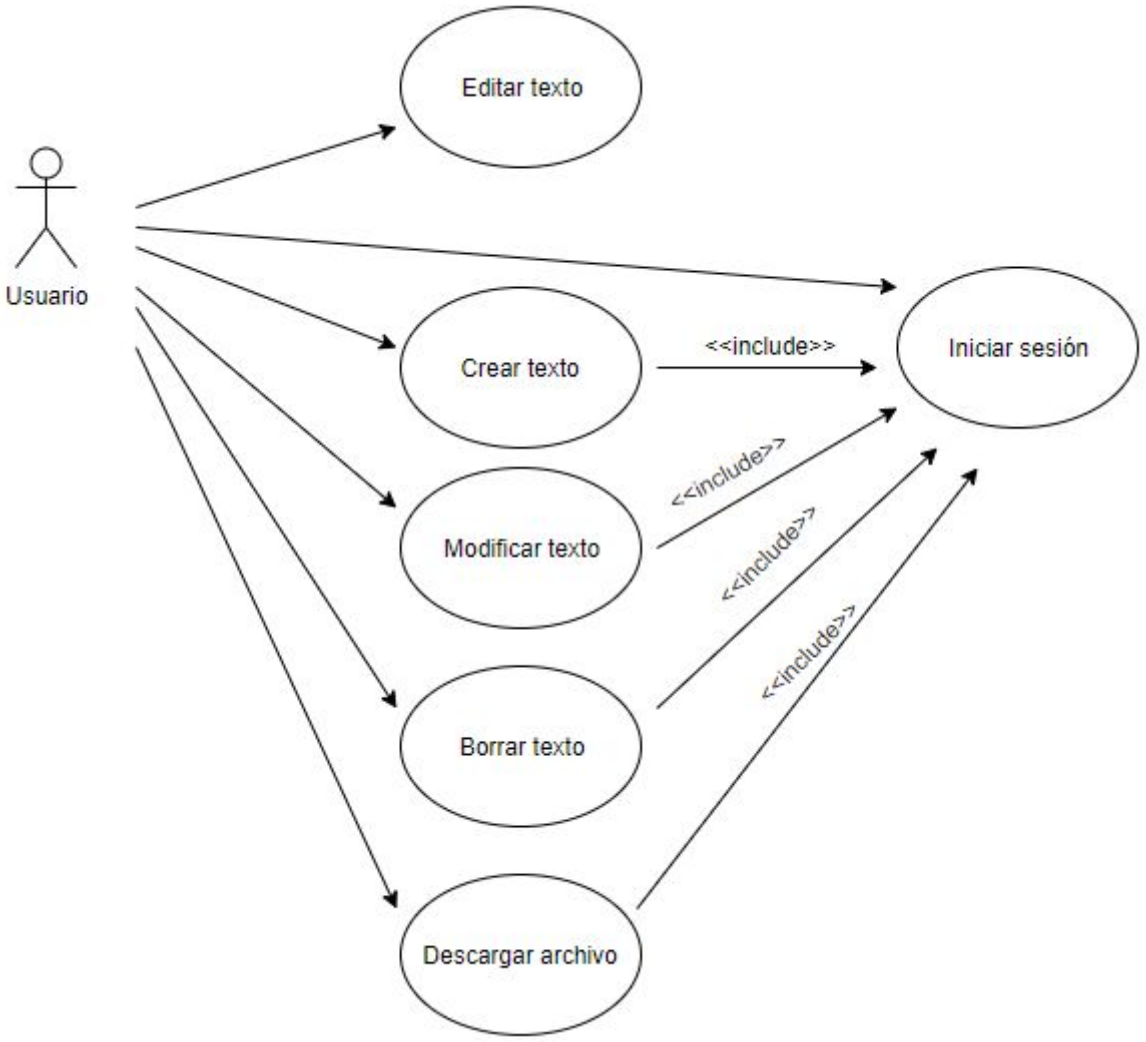

Figura 8.2.- *Caso de uso de Editor*

7.3. AGENDA

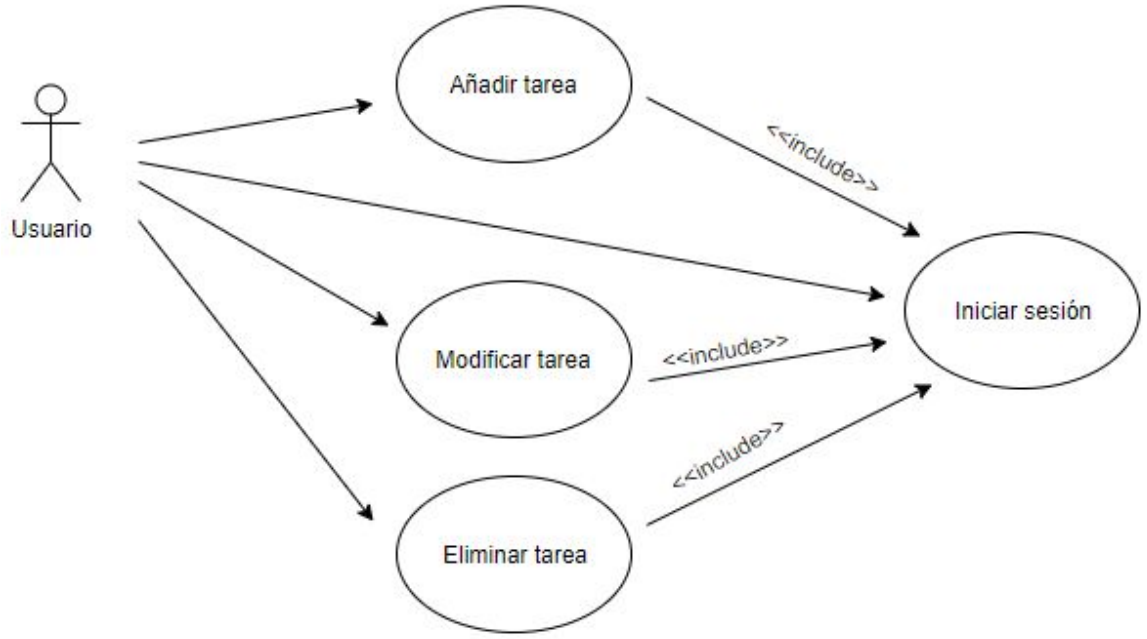

Figura 8.3.- *Caso de uso de Agenda*

7.4. EJERCICIOS

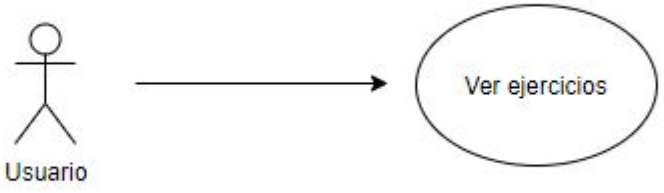

Figura 8.4.- *Caso de uso de Ejercicios*

Capítulo 8
LA APLICACIÓN

FocusOn es una herramienta enfocada en la concentración del usuario que la utilice, creando un ambiente de trabajo que aumenta la productividad y aísla de las posibles distracciones del medio. Para conseguirlo, se basa en la proporción al usuario de todas las utilidades que necesite para crear dicho ambiente, estando enfocada en los trabajos en los que sea necesario el uso de ordenador.

Posee un editor agradable y preparado para escribir código en un gran número de lenguajes, que permite almacenar los archivos en la aplicación y acceder a ellos en cualquier lugar o descargarlos para poder incluirlos en el proyecto local y compilarlo. Por lo tanto, extrae al programador de la globalidad del proyecto, permitiendo su concentración en la tarea actual.

A su vez, el usuario podrá enfocar su concentración en el desarrollo de código en dicho editor gracias a los sonidos ambiente que presenta la aplicación, los cuales pueden ser combinados a gusto de la persona. Si esta desea guardar la combinación, podrá guardarla y cargarla cuando lo desee o modificar una lista que ya posea.

Además, si la persona desea mantener este ambiente pero levantarse un momento para descansar o realizar algún ejercicio de relajación, los cuales también son proporcionados en la propia web, podrá conectarse a través de la aplicación de Android de FocusOn. No será necesario detenerse en el inicio de sesión en la aplicación, evitando de nuevo la distracción, pudiendo acceder a las listas desde el móvil mediante el escaneo de un código QR. La aplicación incorpora un propio lector de QR para evitar tener que utilizar terceras aplicaciones para esta funcionalidad.

Por último, si el usuario necesita de un método de organización, también se proporciona una agenda en la misma aplicación, por lo que el usuario poseerá una herramienta en la que organizar y consultar fácilmente sus tareas pendientes.

Todas estas funcionalidades se recogen y se ofrecen al usuario en una aplicación web desarrollada en Angular. Ésto quiere decir que una vez cargada inicialmente la web, no ocurren más recargas de páginas. Todos los recursos que requiera el usuario están disponibles sin necesidad de “recargar” el navegador (es decir, la aplicación funcionaría incluso sin conexión, exceptuando los accesos a base de datos, como el guardado de un archivo).

Además, la interfaz sigue la guía de diseño Material Design (Google) mediante el uso de la librería Angular Material, que apuesta por un estilo minimalista y enfocado al uso en dispositivos móviles y tablets.

El Back-end ha sido diseñado con herramientas que permiten la adaptabilidad de la aplicación a su crecimiento o a las peticiones del cliente. Spring ofrece infinidad de herramientas que cubren las posibles necesidades que pueda presentar la creación de un REST. Además, si el cliente decidiera utilizar otra tecnología para el servicio REST, es posible sustituir Spring por otro framework fácilmente. De la misma forma, si el cliente decidiera utilizar otra base de datos, sería posible su rápida adaptabilidad gracias al uso de Hibernate.

Capítulo 9
MANUAL DE LA APLICACIÓN

9.1. Aplicación web

Al abrir la aplicación por primera vez, se accederá al menú de elección de una lista de audio, teniendo que elegir pulsar tres botones: para aumentar la productividad, para favorecer la relajación o aleatorio. Se podrá pulsar unas de esas opciones, accediendo a una lista predefinida de audio, o seleccionar una de las opciones del menú.

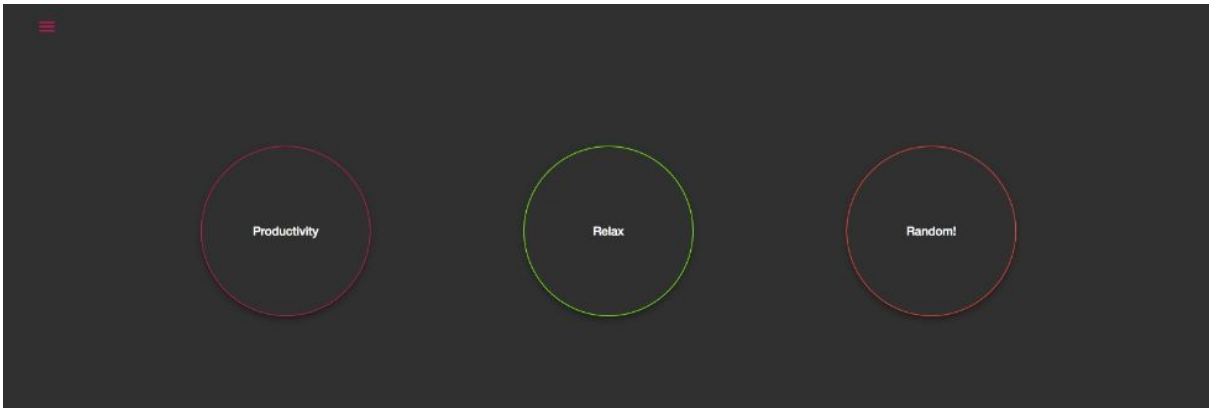

Figura 10.1.- *Ventana inicio de la aplicación web*

Para acceder al menú de la aplicación, se debe pulsar el icono de la esquina superior izquierda. Al desplegarse se encuentran “List”, “Editor”, “Exercises” y “Log in”. Si la persona ha iniciado sesión, también le aparecerá “Calendar”.

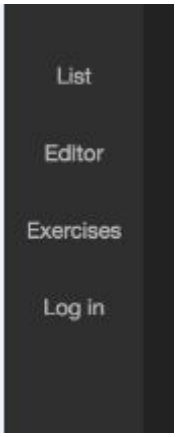

Figura 10.2.- *Menú usuario invitado*

Un usuario puede registrarse en la web introduciendo un correo, nombre de usuario y una contraseña. Si no posee una cuenta de usuario, tendrá limitaciones en las funcionalidades de la web. No podrá guardar listas, archivos o tareas en la agenda. Además, tampoco podrá utilizar la aplicación Android asociada a la web.

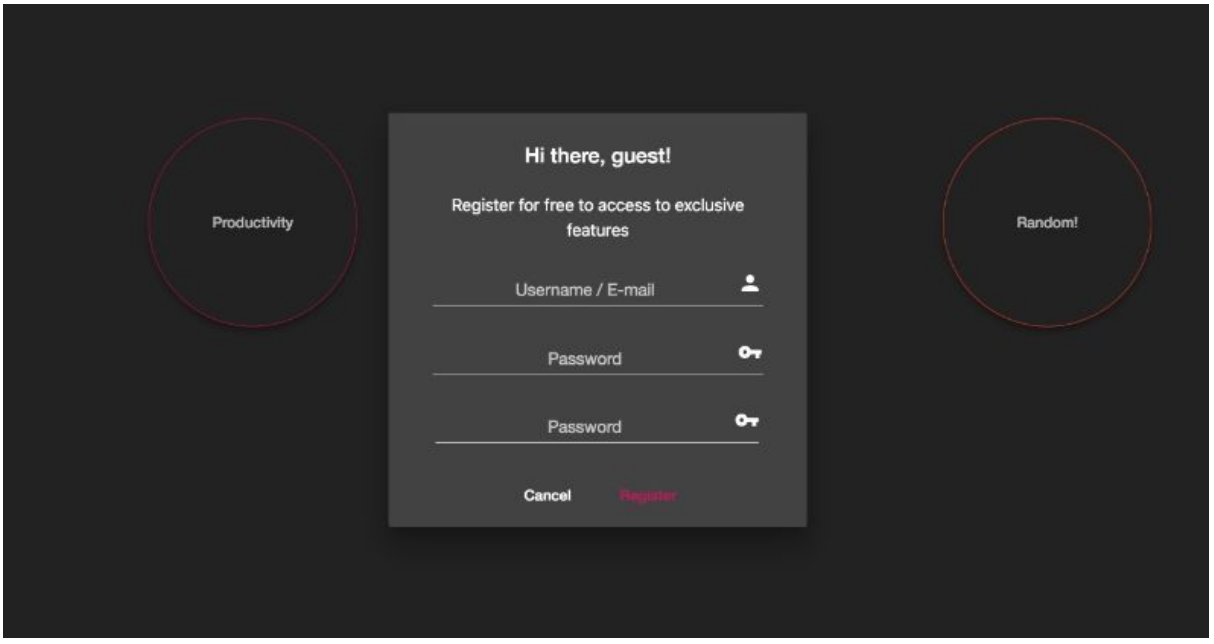

Figura 10.3.- *Registro de usuario*

Al pulsar en Log in, la persona podrá iniciar sesión y acceder a las funcionalidades exclusivas de usuarios.

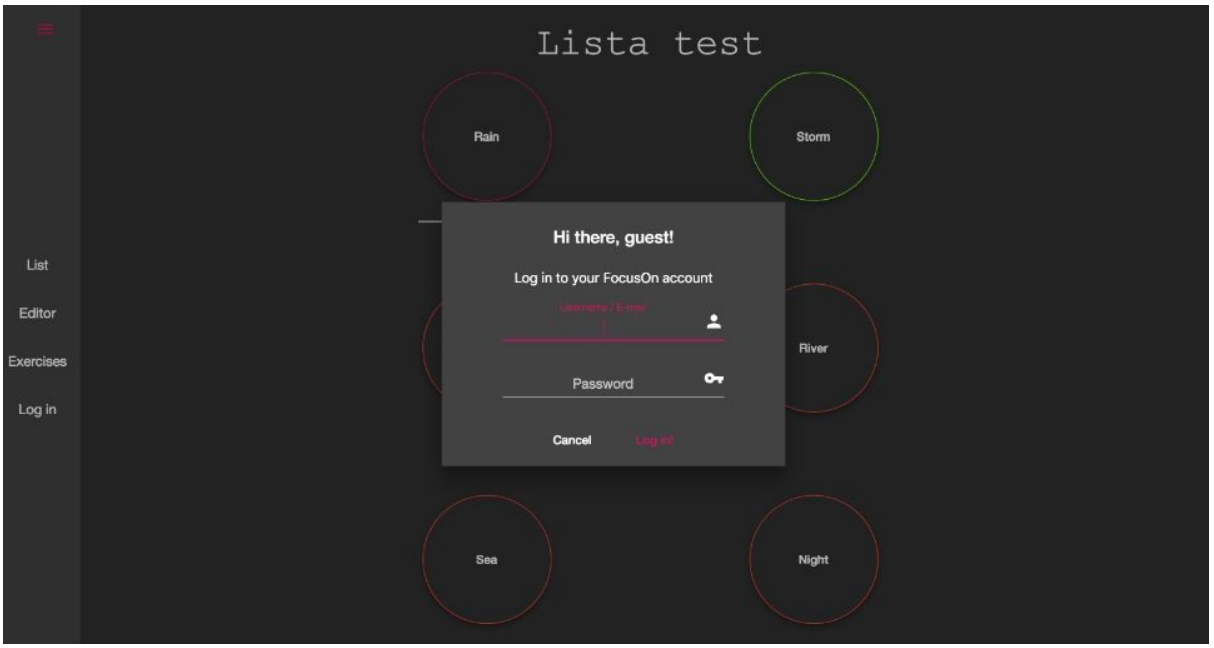

Figura 10.4.- *Inicio de sesión*

Si se pulsa en la primera opción del menú, List, presenta la misma vista a la que se accede pulsando una de las opciones de la pantalla de inicio. En esta ventana se podrá guardar una lista de reproducción con la combinación actualmente seleccionada o cargar una lista ya creada. Aunque para poder acceder a esas funcionalidades es necesario haber iniciado sesión. Sin iniciar sesión solo se podrá modificar los valores actuales.

A continuación, se muestra la vista de un usuario que no ha iniciado sesión.

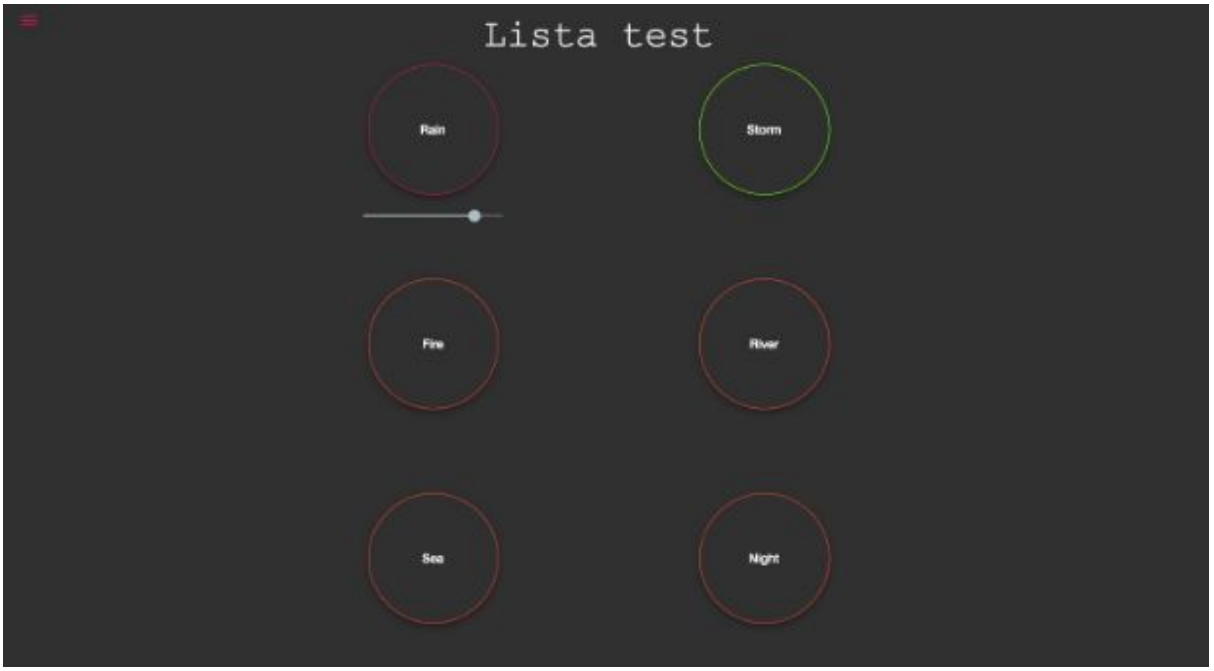

Figura 10.5.- *Lista de sonidos sin iniciar sesión*

Al pulsar en el menú Editor, se presenta la vista que contiene el editor de código. En la esquina inferior derecha se encuentra un desplegable que permite seleccionar el lenguaje en el que se va a programar y descargar el archivo. En la parte superior izquierda del editor, se encuentran los botones que permiten crear, guardar o eliminar un archivo. En la parte central superior se puede modificar el nombre del archivo. En la esquina superior derecha se encuentra el botón que permite descargar el archivo, cuya extensión dependerá del lenguaje seleccionado en el desplegable mencionado anteriormente. Al igual que en la vista de listas de reproducción, para poder acceder a todas las funcionalidades se debe haber iniciado sesión previamente. Si no se inicia, se podrá únicamente escribir en el editor y descargar un archivo.

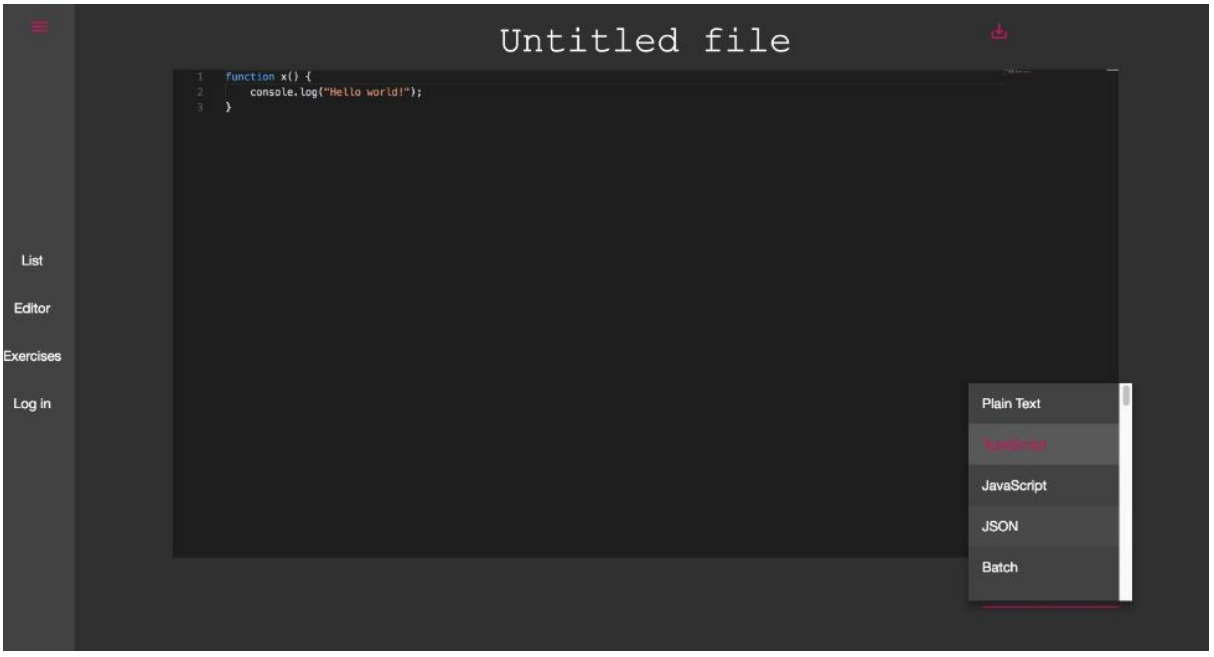

Figura 10.6.- *Editor sin iniciar sesión*

Para acceder a los ejercicios, se debe pulsar en el menú Exercises. A este contenido podrán acceder los usuarios sin necesidad de identificación.

En primer lugar, se mostrarán los ejercicios a elegir. Si se pulsa en uno de ellos, se presentarán las instrucciones del ejercicio.

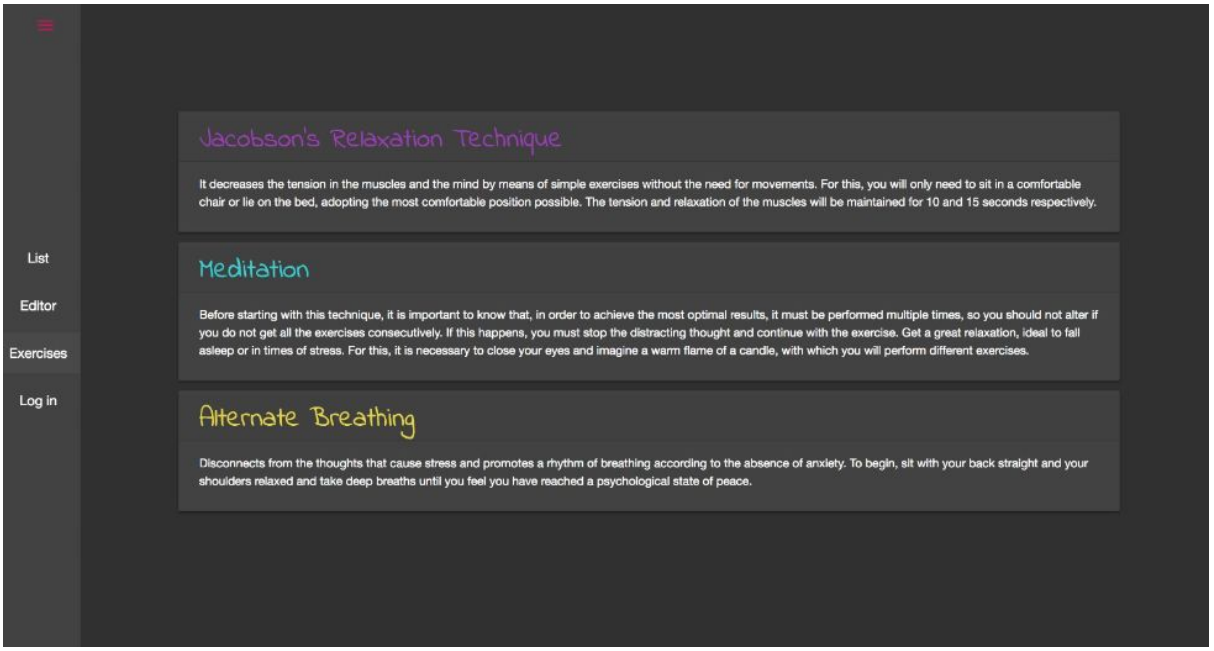

Figura 10.7.- *Listado de ejercicios*

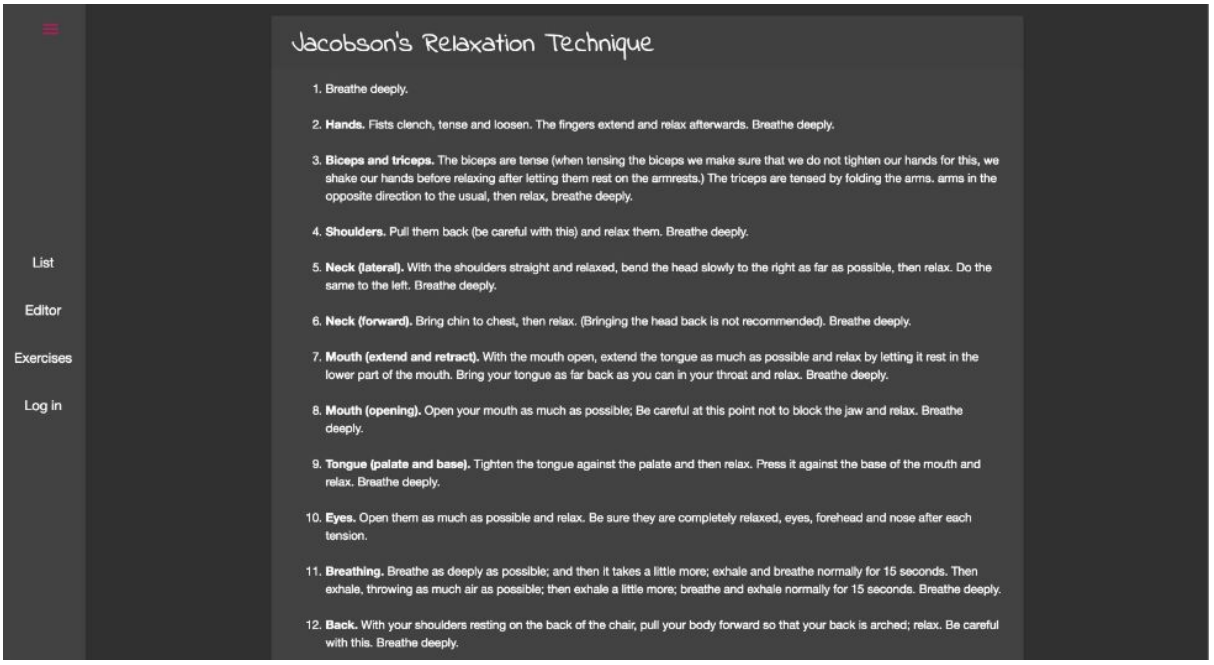

Figura 10.8.- *Instrucciones de ejercicio seleccionado*

Si accedemos a Calendar en el menú, se presentará la siguiente pantalla, donde el usuario podrá ver sus tareas guardadas y añadir otras nuevas.

Figura 10.9.- *Vista de Calendar*

9.2. Android

Para instalar esta aplicación, simplemente es necesario instalar el archivo APK en un móvil con sistema operativo Android o MIUI con una versión superior a 27. Para ello, se debe abrir dicho archivo y aceptar la instalación.

Una vez instalada, si se abre la aplicación, aparecerá la siguiente vista.

Figura 10.1.- *Inicio de aplicación en Android*

La pantalla inicial consta de un lector QR que está destinado al inicio de sesión sin necesidad de introducir los datos por parte del usuario, agilizando así su uso.

El texto de debajo del recuadro de cámara se modificará cuando el programa detecte un código QR.

El botón “exit” permite cerrar la aplicación.

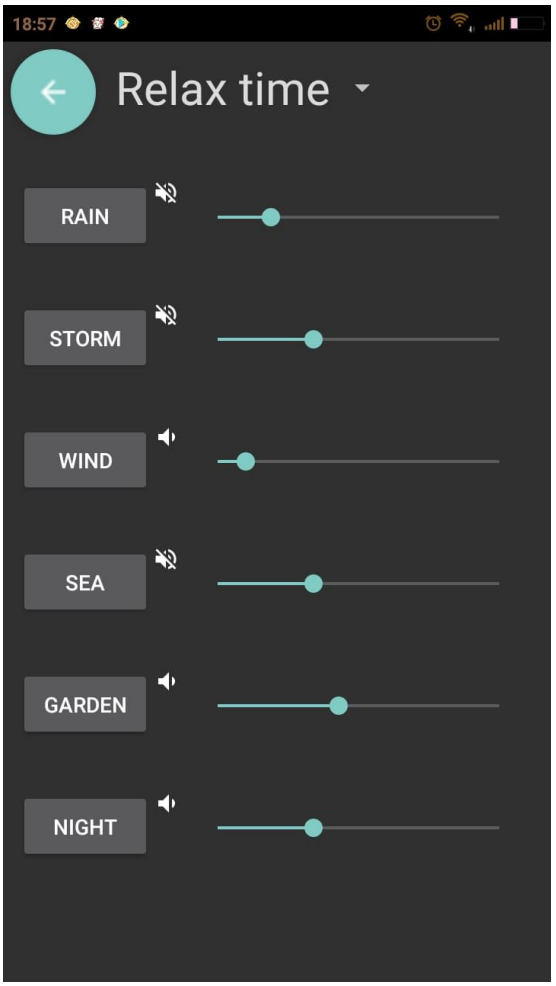

Figura 10.2.- *Vista de lista de reproducción de aplicación en Android*

La pantalla a la que se accede una vez se ha permitido el acceso es la que se muestra a la izquierda.

En la esquina superior izquierda aparece un botón que permite regresar a la ventana inicial.

En la parte superior central se encuentra un desplegable que recoge todas las listas del usuario que ha iniciado sesión.

En la parte central de la ventana aparece un listado de audios con scroll. Cada audio posee un botón, que contiene el nombre del audio, un icono, el cual indica si ese audio está silenciado o no, y una barra que indica el valor del volumen actual.

Para mutear el sonido, el usuario deberá pulsar el botón del audio correspondiente. Si se desea modificar el volumen, se debe arrastrar el indicador de la barra hasta el punto seleccionado y soltarlo. Si no se libera el indicador, el cambio no se llevará a cabo.

Capítulo 10

CONCLUSIONES

FocusOn es una herramienta que puede aportar una mejora al usuario en su entorno de trabajo. Cada detalle de la aplicación está pensado para que el usuario tenga que distraerse el menor tiempo posible de su tarea, accediendo a las herramientas básicas de trabajo de una forma sencilla e intuitiva.

El desarrollo de este trabajo ha conllevado una gran inversión temporal investigando acerca de tecnologías en las que no había trabajado como lo son Spring y Angular, además de un estudio más profundo de otras que, aunque ya había investigado de forma autónoma, desconocía ciertas funcionalidades.

Este trabajo ha aportado una gran cantidad de conocimientos útiles para el ámbito laboral y, en concreto, sobre el desarrollo de una aplicación web en todos sus niveles, creando una base sobre la que expandir mi futuro profesional.

Ha sido una gran satisfacción el finalizar esta aplicación, por el esfuerzo invertido, los conocimientos adquiridos y la superación de cada dificultad, resultando todo en una herramienta de trabajo con una gran utilidad para los programadores.

CAPÍTULO 11

BIBLIOGRAFÍA

Angular.io: Angular

Github: Primeng

Primefaces: Primeng

Github: Monaco editor

Github: Angular material

Reactivex.io: Rxjs

Toddmoto: Angular

Github Pages

Campusmvp: Angular, ventajas

Reddit: Spring, utilidad Spring Boot

Stackexchange: Spring, utilidad capa servicio

Campusmvp: Typescript

Therealdanvega: Spring, definición de componente y bean

Developer.mozilla: CORS

Howtofirebase: Firebase

Spring.io: Spring

Baeldung: Spring

StackOverflow: Spring

Developer.android: Material Design

TutorialsPoint: Hibernate