Mushroom Webapp

Memoria de Proyecto Final de Máster

Máster universitario de Desarrollo de Sitios y Aplicaciones Web

Área de Informática, multimedia y telecomunicación

Autor: Dionisio García García
Consultor: Miguel Calvo Matalobos
Profesor: César Pablo Córcoles Briongos

Fecha de entrega: 06/06/2022

Dedicatoria

Siempre en el camino encuentras gente de la que inspirarte para seguir el tuyo propio. Simplemente hay que estar bien atento porque en aquellas situaciones donde aparentemente no existe un aprendizaje, quizás sea el contexto o lugar donde surja la oportunidad de mejorar en lo personal o lo profesional.

En el apartado técnico cuento con grandes amigos de carrera a los que quiero agradecer las mil y una veces que me han ofrecido ayuda para poder resolver retos tecnológicos. Salvador Caño, Daniel Dinu y Sergi Cuenca son pilares en mi formación y desde lo más profundo les deseo lo mejor en sus trayectorias académicas o profesionales.

En el plano docente y más concretamente en la enseñanza de la educación secundaria, donde ejerzo a diario mi labor, mando un fuerte abrazo a mis compañeros de departamento del CEEDCV (curso 2020/2021). Sin duda guardaré con cariño los recuerdos del que fue mi primer curso como docente. Especial agradecimiento a Sergi, Alfredo y Vanessa por su paciencia infinita conmigo y por la confianza depositada.

Quiero mostrar mi agradecimiento a los centros donde el presente curso 2021/2022 imparto docencia: Antiga Panderola e Historiador Viciana. En especial por su comprensión y apoyo en la decisión de compaginar estudios superiores y ejercer como docente en búsqueda de mejorar mi perfil técnico y poder ofrecer el mejor de los servicios posible al alumnado.

En el plano personal quiero agradecer a mi pareja y a mis padres todo el apoyo logístico que conlleva que un miembro de la familia se embarque en cursar estudios superiores y trabajar al mismo tiempo. Sin lugar a dudas, gran parte del éxito es suyo. A mi solo me quedó recorrer el camino que me propuso la UOC con este más que interesante Máster universitario de Desarrollo de Sitios y Aplicaciones Web.

En el plano académico, quiero agradecer la confianza y las incontables muestras de apoyo recibidas del consultor asignado para la supervisión del avance de la asignatura de Trabajo de Fin de Máster: Miguel Calvo Matalobos. De igual modo, me gustaría trasladar mi agradecimiento al cuerpo de profesores que me ha dado acompañamiento en las diferentes asignaturas de la titulación.

Abstract

El presente documento justifica y describe los detalles del proceso de desarrollo de Mushroom Webapp, una aplicación web capaz de proporcionar información sobre la posible salida de setas en un conjunto de municipios de la provincia de Castellón y Aragón.

Las fuentes de datos sobre las que se cimenta la aplicación provienen principalmente de tres organismos especializados en información meteorológica: AEMET, AVAMET o Meteoclimatic.

Un proceso servidor será el encargado de hacer la recogida de datos para las distintas estaciones meteorológicas a diario. Las bases de datos mantendrán un conjunto de datos relativos a precipitación, temperaturas, humedad relativa o viento de hasta 30 días por estación. Sobre el conjunto de datos, un algoritmo evaluará la posibilidad de hallar setas en un determinado punto de interés, en función de los valores meteorológicos de los que dispone.

Cualquier dispositivo dotado de conectividad y un navegador podrá acceder previo proceso de autenticación a un mapa de Google Maps donde la iconografía superpuesta ofrecerá indicaciones sobre la salida de setas. Complementariamente, se contará con un catálogo de setas capaz de proporcionar información detallada de cada especie. Adicionalmente podrán consultarse registros diarios, mensuales o anuales relativos a cada estación objeto de estudio.

El stack tecnológico se compone de un conjunto de *scripts* de Node.js para la recogida de datos programada mediante el demonio cron, la base de datos Nosql Firebase, la cual almacenará los registros meteorológicos diarios y la información relativa a las setas y Angular de la parte frontend.

Palabras clave: Angular, Node.js, Estaciones meteorológicas, Google Maps API, Firebase, Restful API, cron, AVAMET, AEMET, Meteoclimatic.

Abstract (english version)

The present document justifies and describes the details of the development's process of Mushroom Webapp, a web application capable of providing information about the possibility of mushroom grown in a group of towns of the provinces Castellón and Aragón.

The data sources that form the basis of the application are coming mainly from three organizations specialized in meteorological information: AEMET, AVAMET and Meteoclimatic.

A server process will be in charge of gathering all the data from the meteorological stations daily. The databases will store a set of data relative to raining, temperature, humidity or wind till 30 days per station. Over the data set an algorithm will evaluate the possibility of fetching mushrooms in a specific location, according to the meteorological registers that are available.

Any device provided with connectivity and a browser will be able to access before an authentication process to a customized Google Maps in which the iconography layered will offer indications about mushrooms grown. Additionally, the web app is having a mushroom's catalog ready to provide detailed information of a set of species enabled to the system. Furthermore daily, monthly and annual registers attached to the stations will be able to be consulted.

The technological stack is composed of a set of Node.js scripts for the data gathering programmed through a cron daemon, the Nosql Firebase database, which will store the meteorological registers and mushroom info and Angular at the front-end side.

Keywords: Angular, Node.js, Meteorological stations, Google Maps API, Firebase, Restful API, cron, AVAMET, AEMET, Meteoclimatic.

Notaciones y Convenciones

Para una mejor lectura y comprensión se han decidido aplicar unos estilos de formato y estructuración de la memoria que se describirán a continuación.

La memoria viene estructurada en **28 apartados**, los cuales tienen si así lo requieren sus respectivos subapartados. Por tanto, la presente obra no promueve un tercer nivel de encabezados o al menos no lo refleja en el índice. Adicionalmente acompañan a estos apartados, **9 anexos** y los apartados previos al índice: Dedicatoria, Abstract, Abstract (inglés) y Notaciones y convenciones.

Se ha intentado que la terminología empleada haga un **uso menor de anglicismos**, dado que si buscamos con cariño encontraremos sus homólogos en nuestra lengua. Aún con todo, son introducidos términos propios de entornos de trabajo los cuales han sido entrecomillados o escritos en cursiva para diferenciarlos del castellano.

Los **acrónimos son desglosados en sus componentes atómicos** sobretodo cuando estos son presentados por primera vez en la obra, sobretodo si estos aportan un valor significativo para el lector en el ámbito tecnológico o requieren de una explicación. No así para algunos de los nombres comerciales que se emplean que ocupan un lugar secundario en la obra.

Las **explicaciones adicionales textuales** serán presentadas con una tonalidad gris clara cuando requieran un mínimo de extensión textual. Las pequeñas aclaraciones son resultas utilizando el paréntesis.

Se intenta no abusar de la negrita en el texto. El subrayado no ocupa lugar alguno para indicar al lector un grado de atención extra, de modo que su cometido es el de indicar la existencia de un hipervínculo en el texto.

Las **citas a la bibliografía o webgrafía** en el texto emplean el **sistema IEEE**, caracterizado por el uso de los corchetes y un índice numérico ascendente y una hoja posterior de referencias.

Las **figuras y tablas** son introducidas en el texto y cuentan con un pie de imagen/tabla que en Arial 9 puntos.

El presente documento reune un **índice de tablas y figuras** para su rápida localización.

La fuente utilizada es Arial a 10 puntos para el texto normal, Arial 20 para encabezados de primer nivel y Arial 13 puntos para encabezados de segundo nivel. Los identificadores o fragmentos de código presentado utilizarán la fuente Consolas a 10 puntos.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1. Prefacio

El artículo: “Las setas a través de la historia” de Guadalajara Diario [1] versa sobre los diferentes usos que han sucedido a lo largo de la historia respecto a la recolección de setas. Algunas de las actividades vinculadas al mundo fungi son: la gastronomía, la medicina, la actividad comercial o un tanto más exóticas quizás; los rituales espirituales basados en las propiedades toxicológicas de algunas especies.

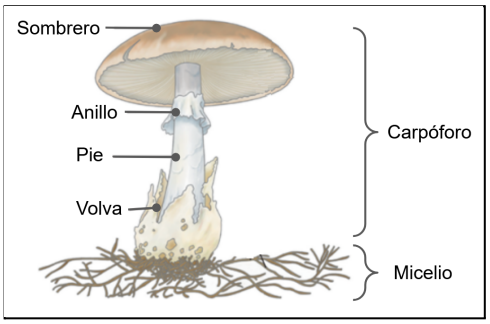


Figura 1: Partes de una seta.

En la Figura 1 y a modo de contextualización para el lector iniciado se muestran las partes más comunes de una seta. Se hará mención a las mismas en puntos más avanzados de la presente documentación.

Para los urbanitas que no hemos tenido la suerte de contar con la experiencia de un lugareño que nos inicie en el maravilloso mundo de la micología in situ, solo nos queda su estudio formal mediante manuales, charlas en asociaciones especializadas o las múltiples salidas al campo por nuestra cuenta.

Según “Setas para Todos” [2] la aparición de los carpóforos, aquellos cuerpos visibles a los que llamamos setas que finalmente son recolectados como el fruto, únicamente hacen acto de presencia si se dan unas condiciones meteorológicas determinadas. La vegetación del lugar y el tipo de sustrato también juegan un papel decisivo. Es importante incidir en la singularidad de algunas especies, más fugaces que otras en temporada por exigir unas condiciones muy concretas para su aparición.

En concreto, el micelio productor de cada especie de seta se activará con una cantidad de agua recibida en un determinado momento. El intercambio de nutrientes que produce la micorriza con la vegetación y una climatología suave permitirá el crecimiento y desarrollo de los frutos recolectables pasados unos días. Las interrupciones más comunes para la salida de carpóforos son:

- La sequía del suelo producida por días continuados de viento.
- La aparición de temperaturas extremas. Un exceso de calor arruinará la cosecha (favorecerá la aparición de gusanos) y la llegada de las heladas “quemará” las especies que hayan asomado su sombrero e interrumpirá cualquier otra salida que se fuese a producir.

Los meses de transición contenidos en otoño o primavera (también en veranos suaves o inviernos contenidos), son el objetivo clave para los recolectores de setas.

Es difícil determinar si un sitio que ha dado frutos en años anteriores, lo vaya a ser sin equivocación las temporadas siguientes. El cúmulo de factores decisivos que hacen que se produzca una salida de setas, si tenemos en cuenta la caprichosidad de la meteorología del lugar, hacen que la recogida de setas se torne una actividad de expertos que requiere de varios años de veteranía para el aficionado.

La tecnología está del lado del aficionado recolector de setas. Con la ayuda de un dispositivo inteligente, son varias las aplicaciones que proveen de utilidades al recolector como son: información de referencia, la posibilidad de registrar los puntos más productivos y en algunos casos, realizar un reconocimiento de la especie fotografiada mediante técnicas de inteligencia artificial.

El segundo gran dispositivo electrónico por excelencia para el recolector es el GPS. Se trata de un seguro de vida para aquellos que se aventuran a recorrer largas distancias en terreno desconocido y luego quieren encontrar el camino de vuelta hacia el coche. Algunos permiten la carga de mapas topográficos sobre los cuales posicionar puntos basados en latitud / longitud. Cuentan con la posibilidad de definir rutas (conjuntos de puntos) y además proporcionan servicio de brújula electrónica, altímetro y opciones de estación meteorológica para los equipos más completos.

La figura 2 muestra la a edición Oregon 300 Setas de Garmin [3] incluía un catálogo de especies embebido de serie en el propio dispositivo.

Figura 2: Edición Oregon 300 Setas de Garmin.

Aún con todo, el recolector sigue sin disponer de un sistema informatizado que vaticine la salida de setas como tal. Los motivos son bastante obvios, este sistema debería de nutrirse de una información localizada meteorológica, la cual no suele ser fácil de obtener. Tradicionalmente el aficionado debe consultar los medios de información donde se publican los partes meteorológicos de sus zonas de interés y extraer sus propias conclusiones en función de la experiencia de haber visitado esos lugares en temporadas anteriores con una climatología similar.

2. Definición

El presente trabajo fin de máster justifica el paso por el *Master Universitario de Desarrollo de Sitios y Aplicaciones Web* documenta la elaboración de Mushroom WebApp, una aplicación fundamentada en las tecnologías web que entre otras gestiones es capaz de proporcionar información objetiva de consulta sobre la salida de setas en lugares concretos de la provincia de Castellón y Aragón.

La entrada de datos para la confección del mapa de salida de carpóforos de esta aplicación se basa en el estudio de los recuentos meteorológicos obtenidos directamente de las redes de estaciones meteorológicas: AEMET [4], AVAMET [5] o Meteoclimatic [6]. Estos organismos de ámbito estatal, autonómico o regional, ponen a disposición de la ciudadanía el total de registros diarios, mensuales o anuales para cada una de sus estaciones vinculadas. El único requisito de uso para la información que proporcionan es citar las fuentes.

Concretamente, la aplicación hace uso de la API OpenData [7] en el caso de AEMET y de la técnica web scraping para obtener los valores diarios de las estaciones de AVAMET y Meteoclimatic. Estas dos últimas organizaciones se caracterizan por proporcionar servicio en lugares con gran singularidad meteorológica.

La recogida de datos se produce gracias a un *script* programado en NodeJS en el lado servidor. Justo antes de reiniciar los valores diarios de las estaciones, dicho *script* capta esta información y la almacena en Firestore, una base de datos descentralizada en la nube, sobre la cual poder conectar desarrollos de tecnologías web o multiplataforma si fuese necesario.

La recogida de datos practicada durante al menos 15 días y máximo 30, permite obtener un estimativo de si uno o varios ciclos de salida de setas han sido viables en función del recuento meteorológico. La interrupción de estos ciclos se programa en función de existir varias jornadas con temperaturas extremas o de viento. El estímulo que permite estudiar el inicio del ciclo de salida es una cantidad de lluvia destacada.

Mushroom Webapp adicionalmente ofrecerá la posibilidad elaborar fichas técnicas con las características más representativas de cada especie para su identificación y estudio.

Las estaciones que sirven como fuentes de datos también tienen su apartado en la aplicación, pudiendo consultar en “tiempo real” la climatología del momento y características propias del lugar.

La trastienda o backend de la aplicación finalmente permite para los usuarios administradores gestionar la adición o supresión de estudio de estaciones meteorológicas, así como verificar el estado de salud de las fuentes de datos o incluso generar datos ficticios que simulan una temporada activa de salida de setas.

El listado de secciones y funcionalidades para Mushroom WebApp queda de la siguiente forma:

Usuario no registrado:

El usuario no registrado podrá formar parte de la comunidad de usuarios mediante el registro por formulario, siempre y cuando el sistema esté habilitado para tal fin. Un usuario con rol administrador deberá previamente conectarse a la aplicación y habilitar la opción de entrada de usuarios. En todo momento el formulario de registro informará a los usuarios sobre la disponibilidad o no para la creación de nuevos perfiles, así como el estado de la acción, si esta es completada satisfactoriamente o no.

La información de registro se materializa en un nuevo usuario en Firestore [8] / “Authentication” y una pequeña colección exclusiva para los usuarios habilitados con información de perfil de usuario.

El usuario correctamente registrado navegará a la página de *login* para proceder a su autenticación.

Usuario no autenticado:

El usuario no autenticado podrá hacer el proceso de autenticación mediante formulario adjuntando el correo electrónico facilitado en el proceso de registro y la contraseña proporcionada. Tras la operación, será informado sobre el estado del proceso de autenticación.

Para un usuario autenticado con éxito, sea rol usuario o administrador, este será enrutado al mapa de salida de setas como punto de inicio en la parte “privada” de la aplicación.

Respecto a “Auth” de “Firestore”, con el proceso de autenticación exitoso se obtiene el identificador de usuario y a continuación, la información de usuario complementaria que será persistida en el almacenamiento local del navegador.

Acceso usuario registrado y administrador:

Tanto el usuario común como el que posee privilegios de administrador pueden acceder en igualdad de condiciones a las siguientes áreas de la aplicación:

- **Mapa de consulta de salida de especies**: es el punto de partida de todo usuario autenticado. En este mapa se muestra una iconografía personalizada que sustituye los markers o pines por defecto de Google Maps sobre un mapa topográfico de la región del interior de Castellón y parte de Teruel. Una leyenda acompaña la iconografía empleada, además de unos controles básicos habilitados para la gestión en el estudio del mapa como son: el zoom o el centrado.

- **Listado de estaciones meteorológicas en "tiempo real"**: este área proporciona un listado de "cards" con diseño totalmente personalizado en SASS **[9]** donde se provee información periódicamente actualizada (cada 5 minutos) sobre la climatología para cada una de las estaciones objeto de estudio.

  La misma "card" es a su vez la vía de acceso a una página de detalle para la estación en concreto. La página de detalle amplía la información aportando acumulados de lluvia mensuales/anuales o detalles concretos como coordenadas geográficas o altitud de la estación física que proporciona las lecturas de datos sobre el medio natural.

- **Catálogo de fichas técnicas de especies de setas**: este área supone un apartado para la formación de los usuarios. A su vez, los usuarios podrán dar de alta nuevas fichas de especies de setas mediante un diálogo implementado utilizando Angular Material **[10]**. Cada especie está representada por un *card* el cual muestra un titular, subtítulos, fotografía y descripción abreviada. La misma *card* conduce hacia una página de detalle donde poder ampliar la información de cada especie: descripción extensa y características distintivas de cada ejemplar. Cada ficha permite acceder a una página de detalle de la seta.

Acceso exclusivo para el usuario administrador (Backoffice):

El usuario administrador podrá gestionar las siguientes operaciones:

- Consulta mediante información tabular de las fichas dadas de alta en el catálogo de especies, pudiendo acceder a la página extendida y habilitar o descartar esta información. *A modo de ejemplo, en Wikipedia/Wikimedia se construye el conocimiento gracias a la colaboración de los usuarios pero esta información se habilita en la plataforma tras la aprobación de un supervisor.*
- Consulta sobre la información tabular sobre las estaciones objeto de estudio, pudiendo inhabilitar temporalmente la visibilidad de estas en el front-end (que no su recogida de datos en el back).

Ambas tablas de administración son componentes Angular Material que cuentan con filtrado y paginación de elementos.

- Dos controles a modo de “switch” para habilitar:
    - La creación de nuevos usuarios en la comunidad.
    - Que los nuevos usuarios que se creados tengan el rol administrador.

Procesos cron activos en el lado servidor:

Mediante un demonio cron programado en el lado servidor se lanza periódicamente un *script* Node.js [11] de recogida y actualización de datos. Concretamente:

A las 23:55 se dispara **“*WriteRegisters*”** (sin flags) para obtener el registro diario que será “encolado” en la matriz de estudio de los últimos 30 días. Inmediatamente después, dicho *script* efectúa la llamada a dos funciones más: una para la actualización de acumulados de lluvias y otra para puntuar la matriz de 30 días. Concretamente esta última, escribe un número entero en el intervalo [0-4] que servirá para poder asignar un icono en el mapa desde el front-end que representará la salida o no de setas en un determinado lugar.
Adicionalmente, cada cinco minutos y reutilizando ***“WriteRegisters”*** el demonio cron programado invoca de nuevo a este *script* añadiendo el flag **‘-hourly’**. De este modo se practica una recogida de datos para traer la información más reciente sobre la climatología de las estaciones dadas de alta en el sistema para poder ser presentada esta en el front-end.

La recogida de datos la practican internamente a “WriteRegisters” tres funciones especializadas para cada uno de los tres tipos de estaciones con las que trabaja el sistema. De este modo sería factible en un futuro añadir más estaciones de control a la aplicación que pertenecieran a las organizaciones AEMET, AVAMET y Meteoclimatic.

Para la recogida de datos de las estaciones Meteoclimatic y Avamet se utilizan las librerías Axios **[12]** y Cheerio **[13]**. En particular, Cheerio es la librería que efectúa web scraping y que por tanto extrae los datos posicionándose sobre los elementos del *Document Object Model* (DOM) utilizando para ello los selectores pertinentes y accediendo a los nodos de texto con el dato en cuestión. Los datos son normalizados mediante una pequeña función de autoría que asegura una homogeneidad y consistencia a la hora de presentar la información o tratar esta.

Para la recogida de datos de las estaciones Aemet se utiliza la API OpenData que permite traer los conjuntos de datos requeridos con notablemente menos esfuerzo programático que el proceso “manual” de efectuar web scraping. Concretamente OpenData facilita conjuntos de datos registrados para cada hora del día, siendo los más recientes el objetivo principal de consulta.

La ventaja principal de utilizar una API es el “compromiso” de quien ofrece la interfaz de acceso a datos con los usuarios que la consumen de mantener la compatibilidad con sus desarrollos. El web scraping practicado sobre portales web es más propenso a errores, dado que las variaciones sobre el DOM de las páginas objetivo harán que no sirva la programación efectuada y que por tanto obtengamos errores en nuestra aplicación. En este sentido sería interesante preparar una estrategia preventiva de errores y disponer de un apartado en el backoffice que permita consultar el “estado de salud” de los datos traídos por web scraping en cualquier momento.

Podría darse el caso puntual de que en el front de las páginas web de las estaciones de AVAMET o Meteoclimatic, por problemas técnicos no se obtuviesen los valores objetivo y que se mostrasen “nulls” / “undefineds” o códigos de error que no permitiesen efectuar el adecuado tratamiento de la información.

3. Objetivos

En este punto se verán los objetivos que se desean alcanzar con la realización del presente proyecto. En primer lugar serán descritos los objetivos principales, inamovibles e impuestos por las directrices de aprovechamiento de la propia asignatura y en segundo lugar los objetivos secundarios, los cuales complementan y refuerzan la motivación de realización del desarrollo web y toda su documentación anexa.

3.1 Principales

- Poner en práctica los conocimientos adquiridos sobre las tecnologías web presentadas a lo largo de la titulación.

- Realizar un proyecto en un contexto real en todas sus fases, seleccionando los procedimientos más adecuados para llevarlo a cabo.

- Adquirir experiencia en afrontar los retos que supone sacar adelante un proyecto completo.

3.2 Secundarios

- Redacción de una documentación técnica de calidad que estructura sus contenidos en función de las necesidades del proyecto tecnológico al que se refiere.

- Elaboración de una aplicación basada en tecnologías web con vistas a ser mantenida y ampliada en funcionalidad para uso personal o colectivo reducido.

- Profundizar en los principales frameworks basados en tecnologías web más actuales.

4. Marco teórico

La recolección de setas silvestres de temporada es una labor que lugareños o empresas especializadas llevan a cabo explotando una zona acotada sobre la cual existen restricciones en su aprovechamiento como: el número de kilogramos recogidos por persona y día o la intencionalidad de venta a posteriori.

No se conoce para la recolección profesional de setas de una aplicación informática o “radar” que permita orientar a las empresas del sector o recolectores particulares en su labor. Generalmente es la experiencia acumulada del sector y la gente del lugar la que permite que se inicie una campaña de recolección.

Mushroom WebApp se basa en un apartado concreto del programa de TV3 “Caçadors de Bolets” [14] presentado por Enric Gràcia. El mismo muestra a la audiencia un mapa de Cataluña que expone zonas propensas a ser fructíferas en un determinado momento. En la Figura 3 se muestra cómo “Caçadors de Bolets” habla de las especies que pueden aparecer además de su estado de conservación.

Figura 3: El mapa de Caçadors de Bolets.

Caçadors de Bolets generalmente ha sido emitido en temporada de otoño y por tanto no se contemplan otras épocas del año que también pudieran ser fructíferas. Para otras zonas geográficas diferentes a Cataluña esta información puede ser interesante como referencia. Sin embargo la orografía de otras zonas de interés, su climatología y su vegetación harán que el panorama de recolección de especies sea muy distinto de esta predicción realizada sobre el terreno Catalán.

El incesante crecimiento de la comunidad de programadores de dispositivos móviles hace que existan aplicativos en Google Play Store [15] que ofrecen soluciones interesantes a la hora de practicar la búsqueda de setas. Como se aprecia en la Figura 4, el **estado del arte** para estas aplicaciones va en la dirección de:

- Proveer información al usuario mediante catálogos de especies ya insertadas en el sistema.
- La posibilidad de marcar puntos en el mapa que simbolizan que allí existe un setal.
- Reconocimiento de imagen para poder identificar especies.
- Juegos con escenarios 2D/3D que emulan para el usuario un entrenamiento para la búsqueda de seta.

En la dirección de orientar a un buscador de setas o hacer más provechosa su salida al campo existen unas pocas aplicaciones como “Cotos de setas” de “Ictiotech” [16] que muestra los dominios de varios cotos localizados de recolección en la geografía española. Como curiosidad, esta aplicación también permite obtener permisos de recolección para algunas de las zonas de recolección de las que provee información.

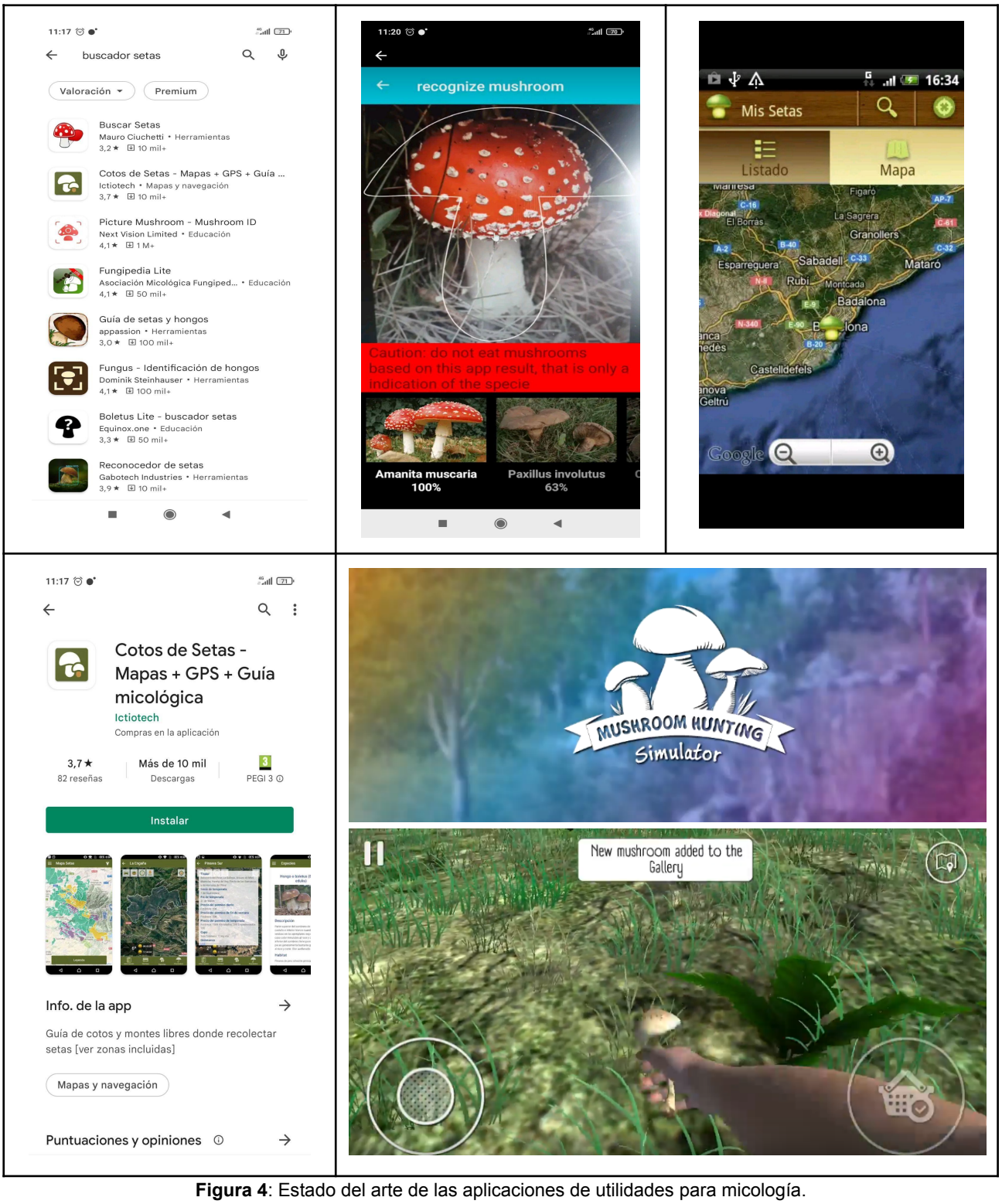


Figura 4: Estado del arte de las aplicaciones de utilidades para micología.

5. Contenidos

La aplicación Mushroom WebApp consistirá en un portal web accesible sobre gran variedad de dispositivos móviles y de escritorio para lo cual se disponen de las siguientes entidades software programadas:

5.1 Contenidos del lado servidor

Demonio Cron

Permite disparar la función "`writeRegisters`" con el flag '-hourly' activado y una periodicidad de cinco minutos con objetivo de actualizar el estado climatológico actual de las estaciones. Como puede apreciarse en la Figura 5, también Cron invoca "`writeRegisters`" a las 23:55 de cada día (sin flag) para traer la tupla de datos diaria para cada estación.

```
# m h  dom mon dow   command
*/5 * * * * node ~/MushroomBE/writeRegisters.js -hourly
55 23 * * * node ~/MushroomBE/writeRegisters.js
```

Figura 5: Programación de cron.

Package.json

Fichero de configuración de dependencia del desarrollo back-end.

El mismo muestra la instalación de:

- Axios
- Cheerio
- Firebase
- Node-fetch
- Request
- Typescript

package.json > ...

```
{
  "name": "MushroomBE",
  "version": "1.0.0",
  "description": "",
  "type": "module",
  "main": "index.js",
  Depuración de ▷
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "axios": "^0.26.1",
    "cheerio": "^1.0.0-rc.10",
    "firebase": "^9.6.8",
    "node-fetch": "^3.2.3",
    "request": "^2.88.2",
    "typescript": "^4.6.2"
  }
}
```

Figura 6: Package.json del back-end.

Fichero de código `fb_config.js`

Importa las librerías de 'Firestore', incluye la API KEY de usuario vinculada a la cuenta personal de Google y exporta una instancia hacia la base de datos para la posterior utilización de la misma en las distintas funciones de lectura y escritura de documentos y colecciones.

Fichero de código `WriteRegisters.js`

Este fichero dispone de la lógica programática necesaria para la recogida de datos de las estaciones y escritura de los mismos en 'Firestore', de un modo mantenible y apoyándose en funciones auxiliares que serán presentadas en este apartado. En primer lugar, '`WriteRegisters`' procesa el *flag* (si lo hay) y trae el total de estaciones dadas de alta en el sistema. Para cada estación efectúa una llamada a la correspondiente función especializada para la recogida de la tupla de datos dependiendo del tipo de estación, en función de si esta pertenece a la red AVAMET, AEMET o Meteoclimatic. Finalmente con la tupla de datos y el *flag* o ausencia del mismo, escribe la tupla en el conjunto de registros diarios o utiliza esta para actualizar la información del momento de cada estación.
Si se trata del caso de añadir la tupla en el conjunto de registros diarios, adicionalmente una función auxiliar evaluará el conjunto de registros diarios para poder pintar el correspondiente icono en el mapa y actualizará los registros de los acumulados de lluvia mensual y anual.

A continuación se enumeran las funciones auxiliares que incluye el fichero de *script* principal.

Función **writeNewRegister**

Recibe como argumentos una tupla de constantes meteorológicas y la estructura de datos estación. Adiciona dicha tupla al conjunto de registros mensuales o bien reemplaza los datos del momento de una estación. Finalmente graba los cambios en Firestore.

Función **evaluateStation**

Acepta como argumento una estructura de datos de tipo estación metereológica y evalúa el estado de hasta 30 registros diarios. A rasgos generales recorre la matriz de registros y verifica los estímulos de lluvia y posibles inclemencias meteorológicas que puedan cortar un ciclo de crecimiento de setas. Si las condiciones meteorológicas lo permiten otorga un flag en el intervalo [0-4], donde '0' significa ausencia de setas, '1' salida activa, '2' en crecimiento, '3' heladas y '4' clima seco.

Función **updateStatRains**

Función que actualiza los acumulados de lluvias mensuales sobre un array cuyas componentes refieren a meses y los acumulados de lluvia anual. Adicionalmente también contea la lluvia caída los últimos cinco días.

Función `composeDate`

Permite mediante el objeto Date y sus miembros de acceso crear marcas horarias para el control de adición de registros en Firestore.

Fichero de código `stationsBaseInfo.js`

Exporta un Javascript Object Notation (JSON) con la información mínima e imprescindible para el conjunto de estaciones a la hora de inicializar las estructuras pertinentes para la recogida de datos en Firestore.

La información base para una estación es: un identificador en el ámbito de su organización, la altitud sobre el nivel del mar, el tipo de organización a la que pertenecen y la latitud y longitud que corresponde a su posición geográfica. Ver Figura 7.

```
3  export const statsInfo = {
4
5    "Castellote": {
6      id: "9561X",
7      altitud: 755,
8      tipo: "aemet",
9      lat: "40.798428",
10     long: "-0.315396",
11   }, // y resto de estaciones..
```

Figura 7: Fragmento de 'stationsBaseInfo'.

Fichero de código `scrapFuncions.js`

Contiene tres funciones especializadas para la obtención de datos de las estaciones AVAMET, Meteoclimatic y AEMET.

Las tres funciones reciben un identificador en el contexto de la organización a la que pertenecen y retornan una promesa que de cumplirse devuelve un array de 6 constantes meteorológicas: temperatura media, máxima, mínima, precipitación, humedad y humedad.

La Figura 8 muestra las cabeceras de las funciones que practican la recogida de datos para las estaciones.

A continuación se enumeran las funciones contenidas en `scrapFunctions.js`:

```
JS scrapFunctions.js > normalizeToDecimals
1   import axios from 'axios';
2   import cheerio from 'cheerio';
3   import fetch from "node-fetch";
4
    Complexity is 8 It's time to do something...
5   export function scrapAEMET(id) { ···
45  }
46
    Complexity is 5 Everything is cool!
47  export function scrapAVAMET(id) { ···
92  }
93
    Complexity is 5 Everything is cool!
94  export function scrapMeteoclimatic(id) {
135 }
136
137 // Returns a float value
138 function normalizeToDecimals(str) {···
144 }
```

Figura 8: Fragmento de 'scrapFunctions'.

Función `scrapAEMET`

Utiliza la API OpenData y el *token* de usuario para realizar una operación 'HTTP GET' mediante la función 'fetch' y arma el array de 6 componentes que ha de retornar.

Función `scrapAVAMET`

Utiliza una URL base de las estaciones AVAMET y la completa con el identificador pasado como argumento. Mediante 'Cheerio' se carga el DOM de la página objetivo y se apunta a los nodos concretos donde reside la información metereológica de valor y se arma el array de 6 componentes.

Función `scrapMeteoclimatic`

Lógica interna análoga a `scrapAvamet` en el contexto de las estaciones de Meteoclimatic.

Función `normalizeToDecimals`

Función auxiliar que aplana los datos extraídos con el proceso *web scraping*, eliminando símbolos contiguos que impidan tratar la información como numérica y ser a posteriori procesada.

Fichero de código `initDB.js`

Importa el JSON base de todas las estaciones, prepara las estructuras de datos adicionales para la recogida de datos e inserta cada estructura Estación en Firestore. En términos generales, inicializa Firestore respecto a la colección de Estaciones.

5.1 Contenidos del lado cliente

Aplicación del lado cliente basada en el framework Javascript Angular CLI [17] que consta de las siguientes partes:

Conjunto de módulos Angular y componentes protegidos algunos en su enrutado mediante un Guard y la estrategia 'canActivate' que verifica la existencia de un usuario autenticado en el sistema:

- Módulo login / registro. És de ámbito público y con los componentes que permiten el registro y autenticación en el sistema.

- Módulo compartido Angular Material. De ámbito público permite definir las importaciones y exportaciones de los componentes gráficos de Angular Material utilizados en el proyecto. A posteriori deberá importarse este módulo a aquellos que precisen de este framework gráfico.

- Módulo de gestiones de mapa. De ámbito privado, permite mostrar una página con un mapa y la iconografía y leyenda que recuerdan en parte a la predicción meteorológica.

- **Módulo de gestión del catálogo de setas**. De ámbito privado, contiene los componentes que permiten:
    - Mostrar un listado de fichas de setas dadas de alta.
    - Mostrar una previsualización en forma de tarjeta de Angular Material en el listado.
    - Mostrar una página de detalle de la ficha.
    - Habilitar un modal para dar de alta una nueva seta.

- **Módulo de gestión del conjunto de estaciones**. De ámbito privado, contiene los componentes que permiten :
    - Mostrar un listado de estaciones dadas de alta.
    - Mostrar una previsualización en forma de tarjeta elaborada con SASS en el listado.
    - Mostrar una página de detalle de la estación.

- **Módulo de gestión del backoffice.** De ámbito doblemente privado, es decir, se requiere ser usuario y con poderes de administrador para poder acceder a este apartado. Muestra dos tablas Angular Material con el conjunto de fichas de setas y las estaciones dadas de alta. Adicionalmente cuenta con dos switches para permitir la entrada de nuevos usuarios y que los nuevos usuarios tengan poderes de administrador.

- **Módulo compartido *footer* / *header* / *error*.** De acceso público y presentes el *footer* y *header* en todo el funcionamiento de la aplicación. Recibe especial mención el componente *header*. Este tiene estado y es el encargado de mostrar uno u otro conjunto de enlaces que permite enrutar al usuario en función de si está logueado, es usuario genérico o un usuario administrador. El estado se mantiene en todo momento gracias a una suscripción a los cambios que puedan producir el proceso de logueo / deslogueo sobre las variables `isLogged`, `isAdmin`, y `name`.

La aplicación obtiene su estado gracias a los cuatro servicios programados en el ámbito de localstorage y las diferentes entidades persistentes en Firestore: Usuario, Estaciones y Setas.

La aplicación cuenta con una arquitectura de hojas de estilo SCSS. En primer lugar, se cuenta las hojas de de ámbito global y una en exclusiva para las variables SCSS, las cuales aplican para valores candidatos aser reutilizados o para valores complejos como por ejemplo la definición de un gradiente lineal. En segundo lugar, se cuenta con una hoja de estilos exclusiva para cada componente definido en los módulos.

Los *assets* de aplicación consisten en un primer conjunto de iconos que participan en los componentes tarjeta para la entidad Estación en el front, un segundo conjunto de iconos para indicar la salida de setas en el mapa y las imágenes de fondo para ambientar la aplicación en el medio natural.

6. Metodología

El proceso metodológico llevado a cabo para el desarrollo de Mushroom WebApp es eminentemente ágil. Si bien el presente caso no se ajusta al pie de la letra a las características propias de Scrum [18] por falta de roles participantes como puedan ser *stakeholders* o *product owners*, el enfoque sí requiere niveles altos de flexibilidad al cambio.

Las variables que hacen que el producto no pueda ser desarrollado de forma secuencial como lo propone el ciclo de vida clásico en cascada son obvias. El arranque y las contiguas fases de desarrollo del producto suceden en un contexto donde las posibilidades que ofrece el *stack* tecnológico todavía no han sido estudiadas en su totalidad acorde a las asignaturas técnicas planificadas en el segundo semestre del máster.

En parte la metodología vendrá condicionada por el alcance del proyecto, la planificación para su realización y ésta a su vez fijada por calendario de la asignatura de Trabajo Final de Máster.

Según R.S. Pressman, B.R. Maxim [19], los dos principios metodológicos de naturaleza iterativa que fundamentan cualquier metodología ágil emergente son: el modelo en espiral y el desarrollo basado en prototipo. Es por tanto que escogeremos una metodología que contemple las bondades de los dos enfoques.

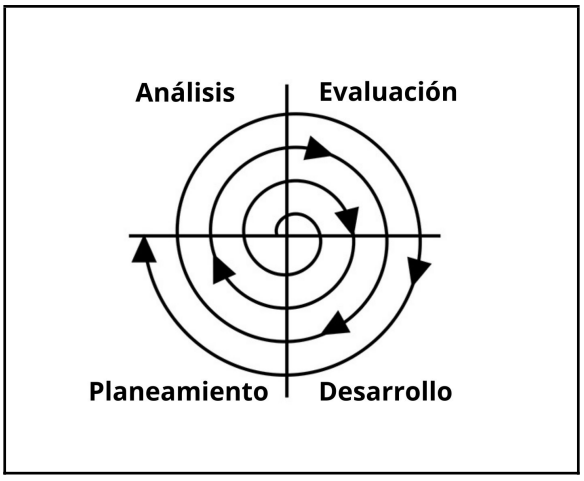


Figura 9: Modelo en espiral.

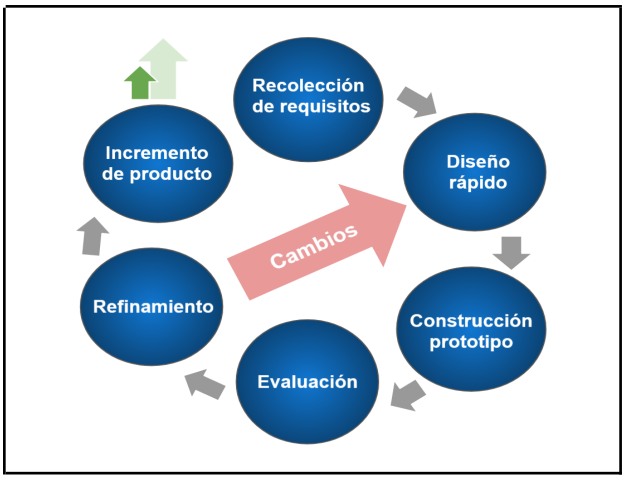


Figura 10: Desarrollo de prototipos.

Dado que fijamos conjuntos de acciones programadas en cada una de las PEC a una fecha concreta, tiene sentido hablar de “sprint” para la realización exclusiva del *backlog* programado. En esta ocasión el “sprint” comprende una cuantía de semanas variable y otorga como producto final un desarrollo incrementado en funcionalidad sujeto a valoración y una documentación técnica en forma de memoria de Trabajo Final de Máster, junto a materiales audiovisuales complementarios.

7. Arquitectura de la aplicación

La arquitectura de Mushroom WebApp sigue el modelo SPA (Single Page Application). En IONOS [20] matizan que las SPA se caracterizan por embeber en el navegador una interacción de elementos con el usuario a modo de aplicación de escritorio. La no recarga de páginas enteras, si no de únicamente aquellos elementos que presentan cambios, hace que las SPA sean una alternativa más eficiente a modelos que solicitan páginas enteras al servidor. En términos generales, una SPA permite emular las características de una aplicación de escritorio siendo gestionada por el motor de renderizado del navegador.

A continuación se describirá a Mushroom WebApp desde la perspectiva de un conjunto de interfaces que establecen comunicación y que por tanto permiten definir los aspectos concretos de su arquitectura.

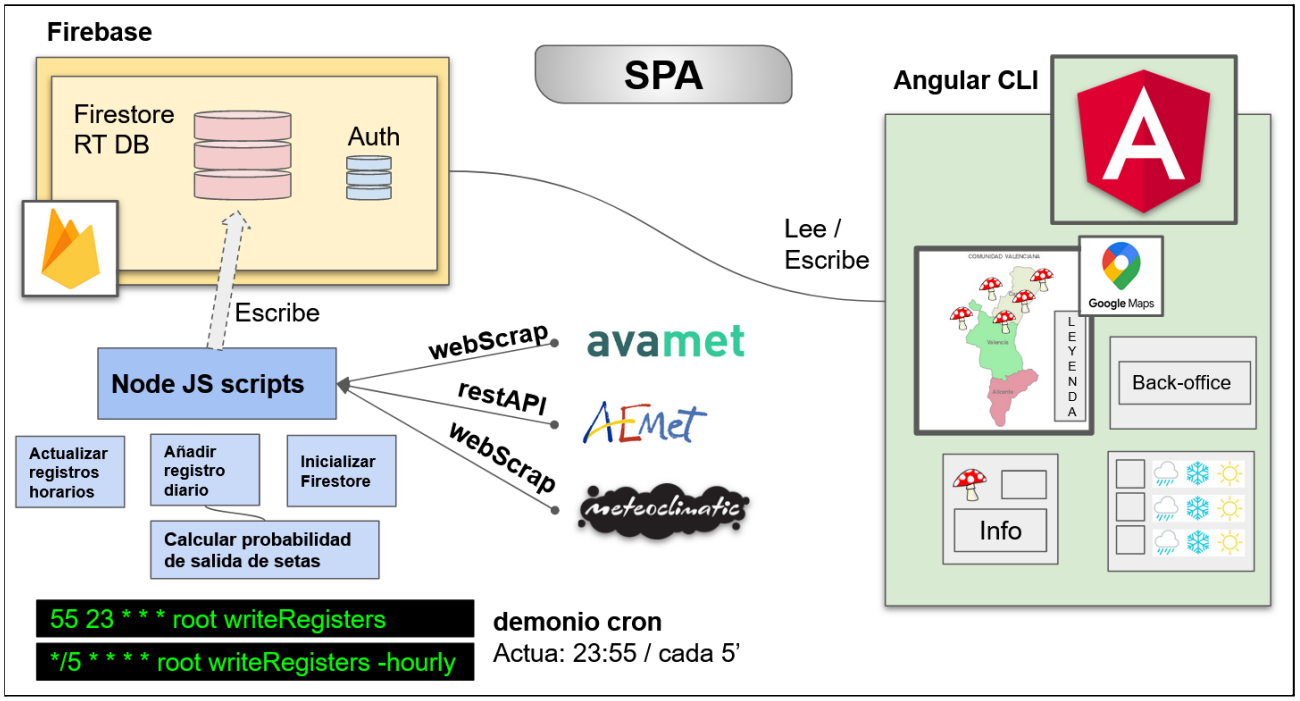


Figura 11: Descripción de la arquitectura de Mushroom WebApp.

La figura 11 muestra que en la parte front-end, un conjunto de servicios de Angular basados en patrón *Observer*, obtendrán autorización para la lectura y escritura de los datos en Firestore. Para poder establecer comunicación con Firestore, se utiliza la librería '`@angular/fire`' [21]. La aplicación Angular consta de un conjunto de módulos de ámbito público / privado, los cuales son cargados bajo demanda. La relación de módulos y componentes permite ofrecer al usuario: la consulta a un mapa de salida de setas, un catálogo de especies, un conjunto de estaciones meteorológicas y una zona *backoffice* reservada a los administradores.

Del lado del backend se ejecutarán periódicamente los *scripts* de adición / actualización de constantes metereológicas, orquestados por un demonio cron programado. La persistencia de los registros meteorológicos se efectuará en el sistema NoSQL Firestore, donde previa autenticación en el subsistema "Auth" [22] se permitirá la escritura en la base de datos FireStore a petición del servidor.

Auth lo conforman un conjunto de usuarios dados de alta en el sistema Firestore y un set de reglas de acceso que otorgan permisos de escritura / lectura sobre las colecciones y documentos de Firestore. En particular, Auth supone la entidad que regula el acceso legítimo a la información contenida en una base de datos de Firebase [23]. El método elegido de entre los que propone Auth para poder registrar y gestionar nuevos usuarios en la plataforma es proporcionando una cuenta de correo y una contraseña.

Para el módulo-componente del mapa de previsión de salida de setas se precisa de un cálculo de probabilidad de salida que permita emplear una iconografía adecuada. El cálculo se efectuará justo al finalizar la adición de una nueva tupla de datos diaria a cada una de las estaciones. En concreto, el algoritmo evaluador recorrerá un conjunto de hasta 30 registros diarios y otorgará un entero en el rango de [0-4] condicionado por la aparición de un estímulo inicial de lluvia para una estación sucedido y cuan suaves o abruptos sean los registros adyacentes de temperatura, viento y humedad. Para una mejor comprensión se proporciona la Figura 12.

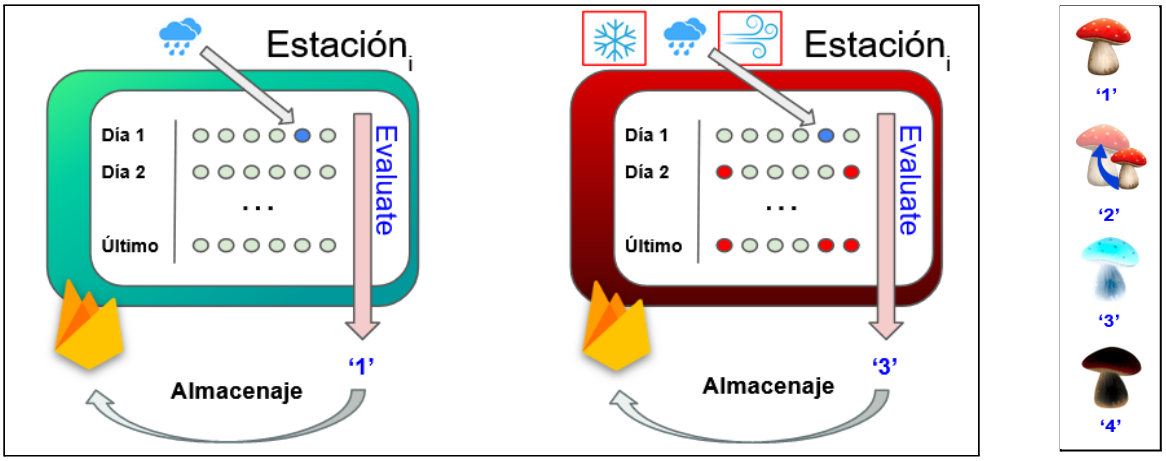


Figura 12: Principio de funcionamiento del algoritmo evaluador de registros metereológicos.

Las comunicaciones con las APIs de Firestore y Google Maps suceden en Angular tal como se describe en la documentación de las mismas. Los servicios programados, por tanto, no hacen uso del objeto `HttpClient` y las llamadas HTTP POST/PUT/PATCH/DELETE a bajo nivel, si no que se cargan las API KEYS en los environments de desarrollo / producción y se importan las librerías previamente instaladas, las cuales harán de interfaz de comunicación mediante un conjunto de métodos asociados a determinadas interacciones con bases de datos, autenticación (sistema Auth) o gestión de mapas de Google.

La Figura 13 brinda una representación del esquema de base de datos basado en las estructuras colección-documento en Firestore quedando de la siguiente forma:

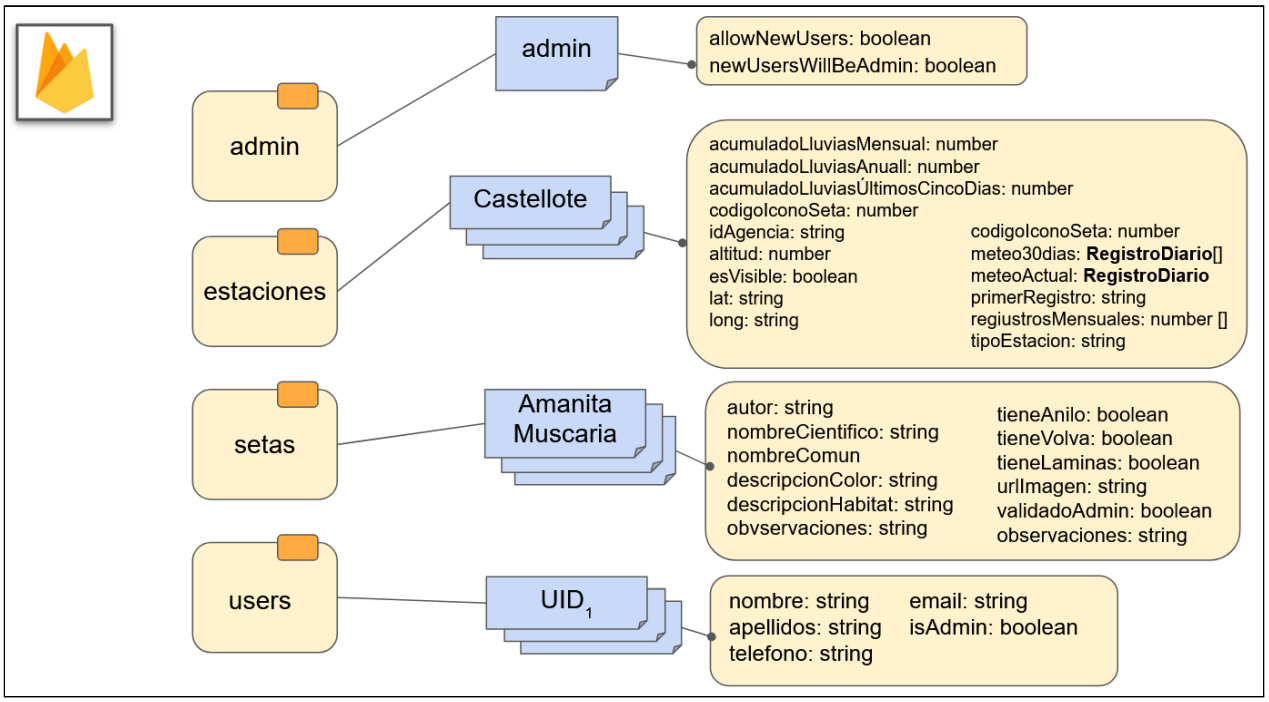


Fig 13: Esquema de base de datos Firestore.

8. Plataforma de desarrollo

Para el desarrollo de la aplicación Mushroom WebApp se han precisado una serie de requisitos previos de equipamiento respecto al hardware y una configuración software de entorno de desarrollo.

Requerimientos software:

Sistema operativo Windows / Linux.
Instalación de Node.js en su versión LTS.
Instalación de Visual Studio Code en su versión community [24].
Extensiones de VS Code: Code Metrics [24], Prettier [25].
Instalación de Git [26] para disponer de repositorios de código en local.
Instalación de Angular CLI (versión 13).
Instalación de @angular/fire como interfaz de comunicación con Firestore.
Una cuenta de Google para acceder a los servicios de Cloud Firestore / Auth.
Una cuenta de Github [27] para almacenar los repositorios de código en la nube.
Alta en Google Maps Platform [28] para establecer un plan de acceso a los servicios de Google Maps.
Habilitación de los ficheros de configuración de entorno con las API KEYS para los servicios utilizados.
Software de edición de imágenes: Inkscape [29] / GIMP [30].

Requerimientos hardware:

Equipo de trabajo con un procesador quad-core y 8 GB de ram.
Interfaz de red Gigabit Ethernet.
Conexión a Internet de banda ancha.
Una cuota de disco de 20 GB libres para los fuentes, *assets*, documentación, etc.

Requerimientos servidor:

Servidor de pruebas / dedicado o virtual host.
Sistema operativo Linux Ubuntu.
Instalación de Node.js.
Instalación y programación del demonio cron y su fichero `crontab`.
Instalación de Cheerio y Axios.
Instalación de librería de Firebase.
Instalación de un servidor web: Apache [31] / Nginx [32].
Habilitación de los ficheros de configuración de entorno con las API KEYS de las APIS consumidas.

9. Planificación

La planificación de elaboración del Mushroom Webapp viene dada por las entregas PEC fijadas por calendario en la asignatura *M4.259 - Trabajo final de máster* en el curso 2021/2022. A continuación se describe el desglose de pormenores de las PEC entregas.

PEC 1: Presentación, alcance y análisis de requisitos funcionales del proyecto (Inicio: 16/2 - Fin: 1/3):
Elaboración y validación de propuesta de proyecto final de máster.
Análisis funcional de la aplicación.
Elaboración de memoria: Contexto y alcance del proyecto. Planificación de tareas.

PEC 2: Definición formal del proyecto e inicio del desarrollo (Inicio: 2/3 - Fin: 30/3):
Descripción formal del proyecto: arquitectura, diagramas UML, etc.
Creación del entorno de desarrollo y prueba de conectividad de APIs
Creación de diagramas de clases / casos de uso.
Creación de wireframes a bajo nivel.
Elaboración de memoria: Descripciones de arquitectura, interacciones y de modelo de datos.

PEC 3: Fase de desarrollo del producto (Inicio: 31/3 - Fin: 8/5):
Llevar el producto a una fase beta respecto a la arquitectura descrita.
Documentación de código
Generación de audiovisuales de demostración / manuales.
Elaboración de memoria: Redacción de apartados complementarios y desviaciones producidas.

Entrega final: FInalización (Inicio: 9/5 - Fin: 6/6):
Finalización del desarrollo web.
Finalización de la memoria del proyecto.
Realización de soporte audiovisual para la defensa del proyecto y para el público general.
Redacción de autoinforme de evaluación sobre el uso de las competencias transversales.

Defensa virtual: (Inicio: 13/6 - Fin: 24/6):
Estudio personal de cara a la defensa.

La Figura 14 representa el Work Bench Structure (WBS) o descomposición de trabajo para Mushroom WebApp. La Figura 15 muestra un diagrama de Gantt que deja ver la cronología de los paquetes de trabajo definidos en la planificación. El diagrama ha sido elaborado utilizando la herramienta de software libre Ganttproject [33].

Nombre	Fecha de inicio	Fecha de fin
PEC1 - Propuesta Trabajo Final de Máster	16/2/22	19/2/22
PEC1 - Análisis funcional	20/2/22	25/2/22
PEC 1 - Memoria: Contexto, alcance y planificación	26/2/22	1/3/22
PEC2 - Descripción formal: arquitectura y diagramas	2/3/22	5/3/22
PEC2 -Entorno de desarrollo y conectividad APIs	6/3/22	11/3/22
PEC2 - Creación DC y DCU	12/3/22	18/3/22
PEC 2 - Creación Wireframes	19/3/22	25/3/22
PEC2 - Memoria: arquitectura, interacciones y modelo de datos	26/3/22	30/3/22
PEC3 - Llevar el producto a fase beta	31/3/22	23/4/22
PEC3 - Documentación de código	24/4/22	27/4/22
PEC3 - Generación de audiovisuales	28/4/22	3/5/22
PEC3 - Memoria: apartados complementarios.	4/5/22	8/5/22
PEC4 - Finalización desarrollo web	9/5/22	19/5/22
PEC4 - Finalización memoria	20/5/22	26/5/22
PEC4 - Realización audiovisual defensa	27/5/22	3/6/22
PEC4 - Realización autoinforme	4/6/22	6/6/22
PEC5 - Estudio defensa virtual	7/6/22	24/6/22

Figura 14: WBS para Mushroom WebApp.

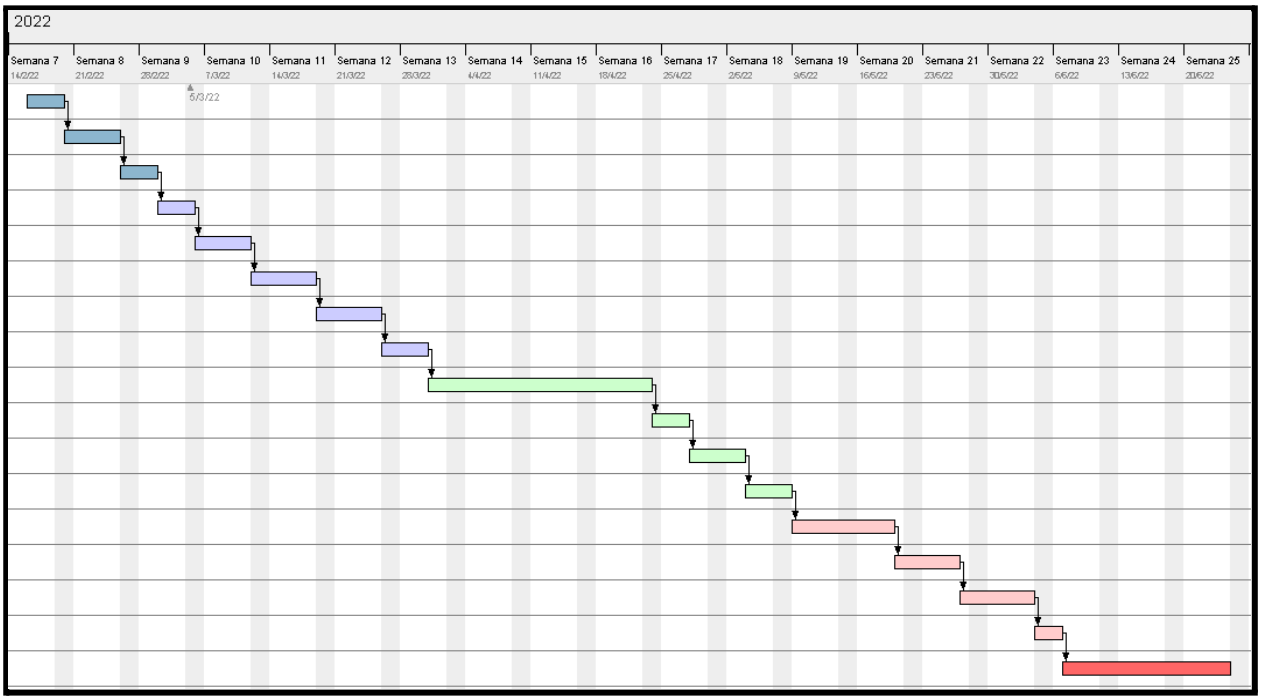


Figura 15: Diagrama de Gantt de Mushroom WebApp.

10. Proceso de trabajo

El proceso de trabajo involucra el crecimiento de la aplicación a la par que la documentación técnica asociada: memoria de Trabajo Fin de Máster y los audiovisuales de presentación y soporte al usuario final.

La planificación de la asignatura propone la realización de 4 entregas PEC que fijan en la planificación del presente proyecto los hitos clave para la liberación del trabajo en forma de incremento parcial acumulado.

La pauta de trabajo es por tanto la superación del “sprint” comprendido en cada una de las PEC realizadas, desde la fecha de inicio hasta la fecha de finalización del mismo.

En las fases primeras del proyecto, correspondiendo con las PECs 1 y 2, pondera con mayor peso sin duda la el análisis funcional y diseño de componentes para la aplicación a desarrollar. Sin embargo, este no será un periodo libre de codificación, todo lo contrario. En esta fase se espera la habilitación del entorno de desarrollo y un el testeo de API 's de terceros de las que la aplicación tomará datos o precisará servicio.

Tanto en la tercera como en la cuarta PEC (en la tercera sobre todo) se dará respuesta al trabajo de codificación en base al planteamiento arquitectónico, de diseño y funcional inicialmente planteado. El proceso de trabajo en este periodo viene dado por sesiones de codificación combinadas con pruebas sucesivas que verifican la consistencia e integridad de datos de la aplicación en primera instancia y que proveen de una interfaz sofisticada y usable para el potencial cliente. En esta fase el sistema de control de versiones de código en la nube registrará el mayor número de *commits* / *pushes*. La PEC 4 se focalizará principalmente en pulir detalles tanto en el ámbito de la depuración de errores o mejora de aspectos de usabilidad tanto como en la finalización de documentación técnica asociada: código y escrita.

La cuarta PEC es la última entrega que libera el alumnado y esta comprende:

- Una video-defensa grabada y subida al repositorio Present@ de la UOC.
- Subir al REC de la UOC en un fichero comprimido:
    - La versión definitiva de la memoria del Trabajo Final de Máster al REC.
    - Un fichero comprimido con todos los fuentes que conforman la aplicación Mushroom WebApp.
    - Una presentación dirigida al público general o inversores donde se prime la promoción.

Una vez entregada la PEC4 es momento de preparar la defensa virtual y familiarizarse con la herramienta *Blackboard Collaborate*. La defensa síncrona tendrá lugar el día jueves 16/6/2022 de 18:15 a 19:00 horas.

11. APIs utilizadas

Google Maps Platform API

A fin de poder tener un mapa interactivo con relieves que permita centrarnos en una zona geográfica concreta, como es el caso de Mushroom WebApp, se ha decidido utilizar la API de Google Maps.

Las alternativas a Google Maps son variadas en el presente momento. “MapBox” [34], “Bing” [35] o “Here” [36] son algunos competidores que ofrecen entre sus planes de captación un uso limitado pero gratuito de sus servicios. No obstante, se ha optado por utilizar Google Maps dado que supone un añadido extra respecto a la familiaridad de uso para los futuros usuarios.

Respecto a la obtención de una API KEY para poder utilizar en los desarrollos web, es requisito por parte del desarrollador dar de alta una tarjeta de crédito para poder cargar los costes que generaría una utilización masiva de los mapas según las necesidades crecientes de una comunidad de usuarios o un abuso del servicio del tipo que sea. La API KEY queda embebida en el HTML raíz (modo inline) y debe volver a ser cargada de nuevo si se requieren mapas en módulos cargados de forma perezosa.

API OpenData de AEMET

La API de OpenData de AEMET es una herramienta de difusión y reutilización de la información meteorológica para la red de estaciones de la propia Agencia. La API es proporcionada mediante un conjunto de endpoints RESTful a los que consultar de forma segura mediante una API KEY. El registro del desarrollador que requiera de los servicios web de OpenData es sencillo y finaliza con un correo electrónico donde se envía la API KEY. Los endpoints están documentados en Swagger [37], pudiendo practicarse pruebas de obtención de datos.

Figura 16: Página web de AEMET OpenData.

Aemet cita textualmente que se puede hacer uso de la información que proporcionan siempre y cuando se mencione al propio organismo. Esta condición de utilización se mantiene para las redes de estaciones: AVAMET o Meteoclimatic.

12. Diagramas UML

A continuación se presentan los diagramas UML de clases y de casos de uso. La realización de los mismos ha sido posible gracias a la herramienta Diagrams.net [38], la cual cuenta con una integración perfecta con Google Drive para su almacenamiento.

12.1 Diagrama de clases

El diagrama de clases deja entrever los modelos de datos participales para Mushroom WebApp y sus interacciones si las hubiera. Como puede observarse se cuenta con tres entidades principales: los usuarios de la aplicación, las estaciones meteorológicas y las fichas técnicas de las setas. La Figura 17 ofrece el diagrama de clases empleado en Mushroom WebApp.

El diagrama de clases se materializa en la aplicación de Angular en forma de interfaces o clases Data Transfer Object (DTO), permitiendo crear nuevas instancias de los objetos a posteriori o tipar las diferentes entidades software con Typescript.

Para la obtención de las interfaces de los modelos de datos he usado la plataforma Quicktype [39], la cual recibiendo un JSON con una estructura de datos en crudo es capaz de proporcionar una interfaz que la representa para múltiples lenguajes de programación, entre ellos TS.

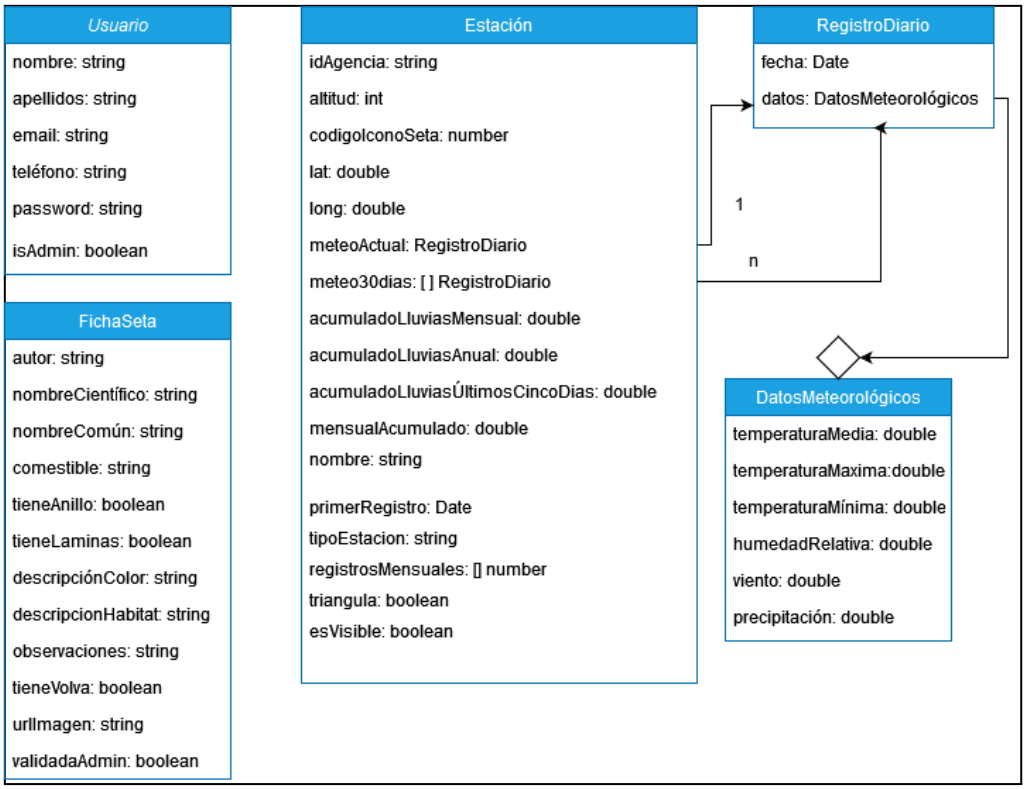


Figura 17: Diagrama de clases de Mushroom WebApp.

La **entidad `FichaSeta`** tiene lugar en la aplicación para la creación y gestión de las fichas técnicas de setas incluidas en el área del catálogo de setas.

La **entidad `Usuario`** participa cuando un nuevo usuario se registra en la aplicación y cuando este se autentica en el sistema. Se discrimina si un usuario es administrador o no con el atributo booleano '`isAdmin`'.

La **entidad `Estación`** es el pilar fundamental de la aplicación. Posee información sobre la estación y conjuntos de datos y registros meteorológicos que permiten estudiar la evolución climatológica de un determinado lugar. Su información participa parcialmente en la confección del mapa de salida de setas y en el catálogo de estaciones meteorológicas. Es en la página de detalle de una estación concreta donde se puede acceder a la información extendida de cada estación dada de alta.

La **entidad `RegistroDiario`** es un tipo de datos utilizado internamente en las estaciones para poder gestionar una marca de fecha de la obtención de datos junto a un array de valores numéricos que representan en este orden: temperatura media, temperatura máxima, temperatura mínima, precipitación, humedad y viento. La **entidad DatosMeteorológicos** no se da en la práctica en la aplicación, es puramente ilustrativa para poder entender el array de datos que contiene un `RegistroDiario`.

12.1 Diagrama de casos de uso

En este caso, el diagrama de casos de uso clarifica el nivel de interacción que el usuario registrado y el administrador tienen sobre el conjunto de funcionalidades habilitadas en la interfaz cliente. Las distintas acciones realizables por cada rol de usuario han sido agrupadas para dotar al diagrama de un mayor nivel de comprensión sobre su ubicación en la propia aplicación. La Figura 18 muestra el diagrama de Casos de Uso elaborado para Mushroom WebApp. El mismo ha sido elaborado con Diagrams.net.

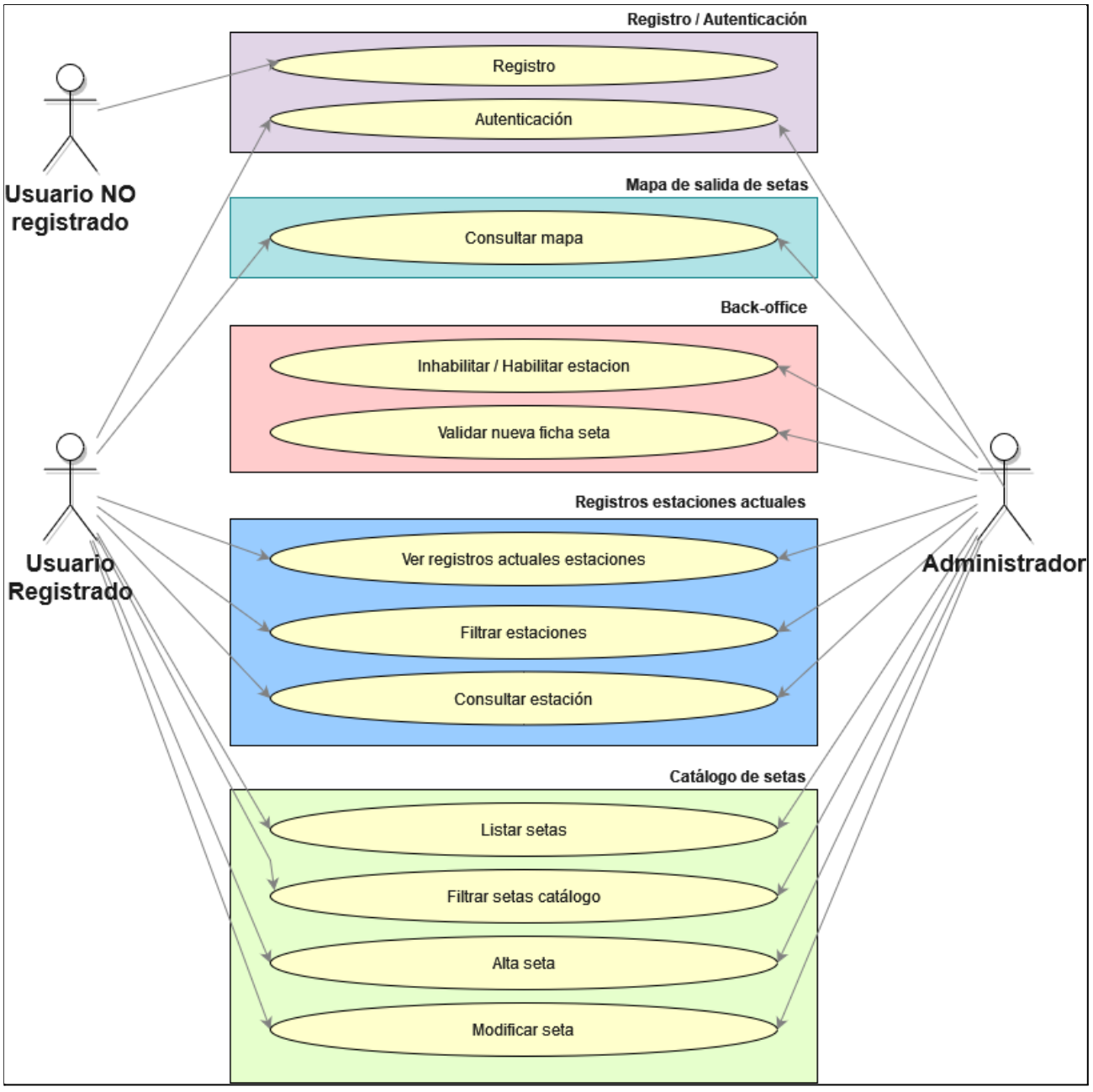


Figura 18: Diagrama de casos de uso de Mushroom WebApp.

13. Prototipos

En el presente apartado se muestran los diferentes conjuntos de prototipos realizados para describir el nivel de detalle y funcional de las interfaz de Mushroom WebApp. De menor a mayor nivel de detalle se presentan: sketches, wireframes a bajo nivel y prototipos Hi-Fi.

13.1 Sketches

Los bocetos o “sketches” han sido realizados a mano alzada y escaneados a posteriori con la aplicación móvil VFlat [40]. El resultado final se puede apreciar en la Figura 19. Esta primera aproximación en el diseño de interfaces promueven la creación de ideas y a su vez permite definir aspectos de navegación entre las diferentes pantallas que se suceden en el flujo de interacción. Los diseños son creados, aprobados o refinados de forma fácil e intuitiva. Las anotaciones complementarias a las interfaces representadas proporcionan un nivel de entendimiento adicional.

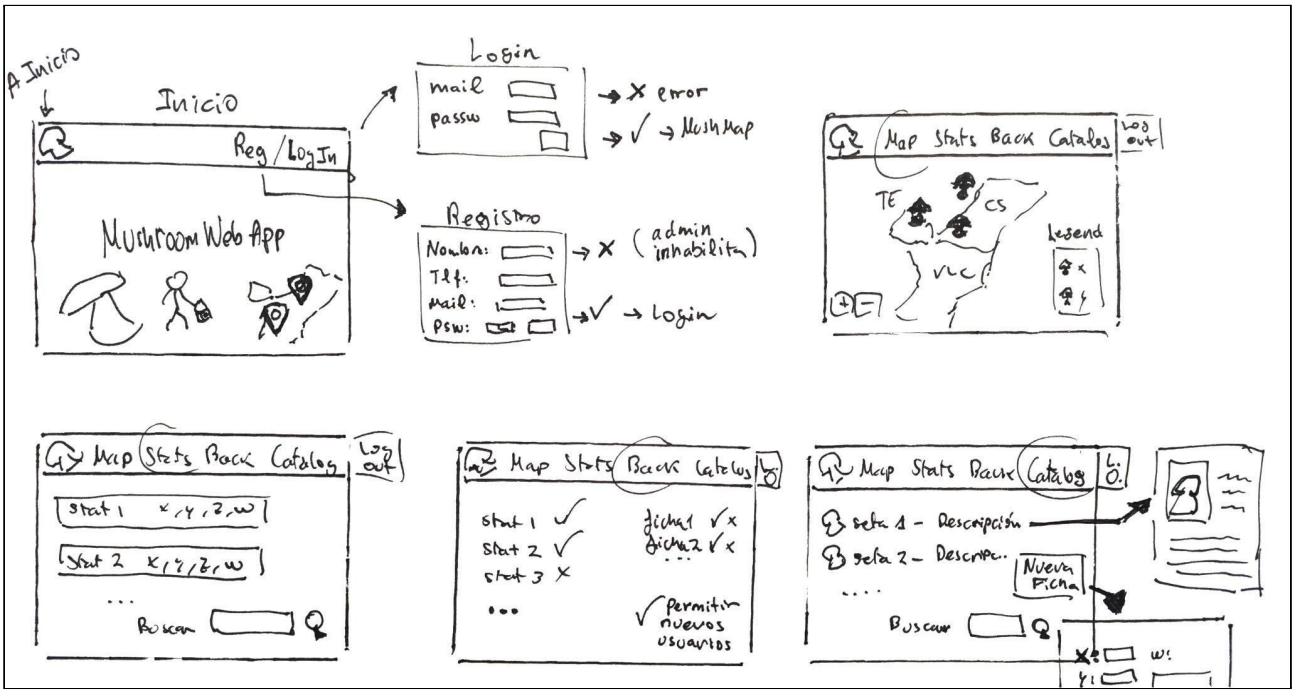


Figura 19: Sketches a mano alzada para Mushroom WebApp.

El objetivo de este conjunto de *sketches* es precisamente fijar un punto inicial en el número de páginas hábiles en el sitio, definir aspectos de navegación y establecer las bases para la arquitectura de la información que opera sobre la aplicación.

13.2 Lo-Fi

A continuación se presenta un conjunto de *wireframes* Lo-Fi (bajo nivel) para representar con más detalle el conjunto de interfaces que simbolizan el núcleo de Mushroom WebApp. Los prototipos a bajo nivel han sido realizados con la aplicación Figma [41]. Los *wireframes* Lo-Fi, cimentados en los "sketches" presentados en el apartado anterior, constituyen una segunda iteración en la búsqueda de una interfaz apropiada para el tipo de encargo y que a la vez la interfaz resulte usable. La Figura 20 muestra el conjunto de *wireframes* Lo-Fi realizados.

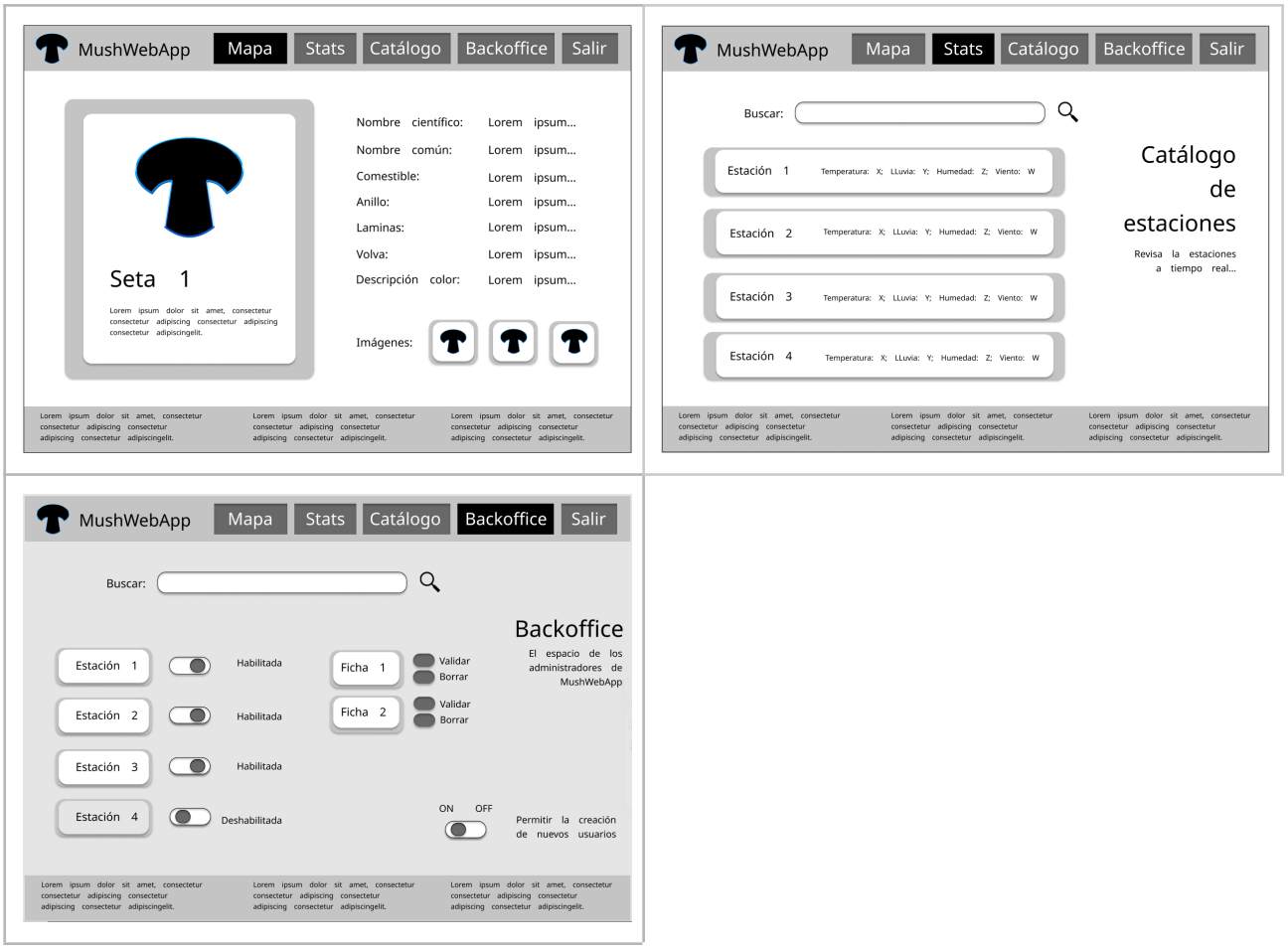


Figura 20: Diferentes Wireframes Lo-fi de Mushroom WebApp.

Los wireframes Lo-Fi consolidan el número de páginas y definen con mayor detalle los componentes que conformarán cada una de las páginas. El contenido respecto a fondos o texto incluido es poco relevante pero sirve para confeccionar un layout de aproximación. El layout todavía sufrirá cambios en la siguiente iteración con la confección de prototipos Hi-Fi.

En el momento de realización de los *wireframes* se cuenta con nociones de diseño de componentes sí, pero no con una idea clara de las tecnologías o posibilidades de los *frameworks* que serán finalmente utilizados puedan ofrecer.

La filosofía de ofrecer un enfoque minimalista y con las menores ambigüedades para el usuario respecto a la arquitectura de la información permanecerá fiel respecto al refinamiento hacia unos *wireframes* de alto nivel o incluso en el diseño finalmente elaborado mediante tecnologías de codificación.

13.3 Hi-Fi

El desarrollo del conjunto de prototipos Hi-Fi ha sido enteramente realizado mediante la herramienta Figma. En este caso, llevamos a Figma a sus límites con la dotación de efectos de gradiente, transparencias y superposición de capas, como si de un programa de diseño gráfico profesional se tratase.

Los prototipos Hi-Fi liberan una tercera iteración que busca un compromiso sólido con el diseño gráfico que finalmente implementará la aplicación. Tomando como referencia los wireframes Lo-Fi del apartado anterior, los Hi-Fi propondrán ligeros cambios en estructura, manteniendo eso sí, la arquitectura de la información que viene heredada. No obstante, los prototipos aquí presentados podrán presentar cambios menores con la aplicación finalmente presentada. Figma no deja de ser una herramienta de prototipado y el producto que genera una aproximación de la realidad. A continuación se muestran algunas de las propuestas de interfaces más características para la aplicación Mushroom WebApp.

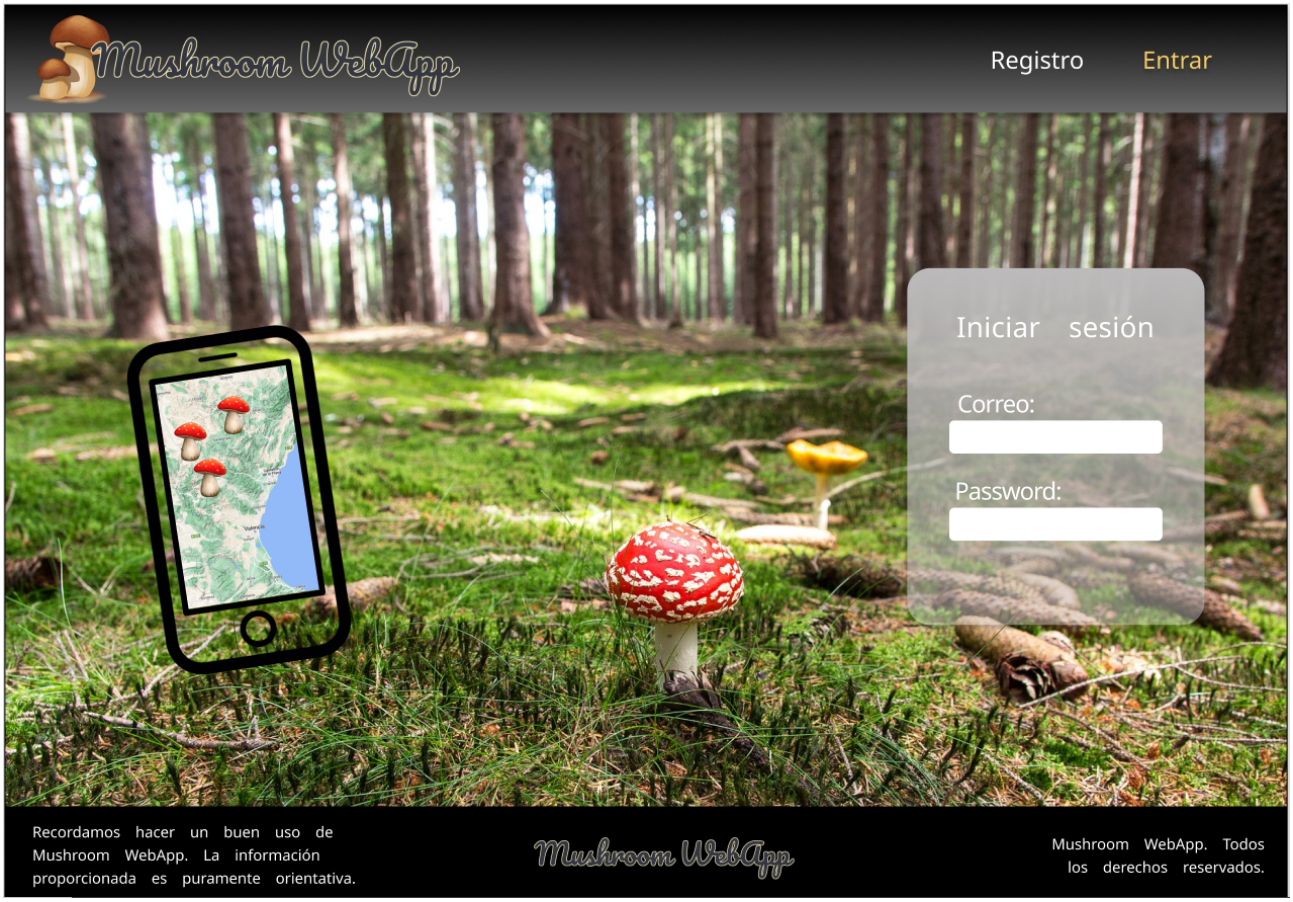


Figura 21: Hi-Fi de Página de inicio de Mushroom WebApp.

Figura 22: Hi-Fi de Mapa de consulta de salida de setas de Mushroom WebApp.

Figura 23: Hi-Fi de Conjunto de estaciones a “tiempo real” de Mushroom WebApp.

15. Perfiles de usuario

A continuación se describe un arquetipo de potencial usuario para la aplicación Mushroom WebApp.

Juan López: El incansable buscador de setas	
Nombre: Juan López Hernández **Edad**: 33 años **Sexo**: Varón **Aficiones**: montañismo, deporte, micología, tecnologías, gps, cartografía, posicionamiento. **Trabajo:** profesor asociado de universidad **Ubicación:** Castellón (España)	

Objetivos
Su objetivo principal es mantenerse informado y poder predecir las futuras salidas de setas para un conjunto de zonas productoras que estudia temporada tras temporada. Gran aficionado a la recogida de setas, pertenece a una asociación micológica de ámbito comarcal donde se forma constantemente y comparte hallazgos sobre las especies de setas que aparecen en la zona limítrofe de la comunidad valenciana y Teruel. Necesita centralizar los datos meteorológicos registrados en una temporada de setas para poder dar explicación al ciclo de crecimiento de algunas especies, así como para hallar las mismas a pie de campo. En la actualidad registra los datos de los pluviómetros de algunas estaciones meteorológicas en papel. Depende de ciertas indicaciones de otros aficionados a las setas para poder hacer una salida de campo mínimamente fructífera, dada la falta de información precisa de las zonas productoras.

Expectativas	**Frustraciones**
Contar con un sistema informatizado que le permita rápidamente obtener datos fiables sobre la posible salida de setas en las zonas que le son afines. Disponer de un catálogo de fichas técnicas de las especies más frecuentes en sus zonas.	Tener que llevar el recuento de las precipitaciones caídas en varias zonas productoras de setas. Ver como otros buscadores de setas se le adelantan a la recogida de algunas especies más preciadas y no es capaz de cazar ejemplares para su estudio.

16. Usabilidad

Mushroom WebApp tiene presentes tres segmentos de dispositivos:

- Los de tamaño de pantalla grande: televisores y monitores de equipos de escritorio.
- Los de tamaño mediano: principalmente tabletas.
- Los de tamaño pequeño: teléfonos inteligentes.

Los *layouts* que conforman las páginas del sitio contemplan dos *breakpoints* principalmente para poder dar soporte a los tres segmentos de dispositivos descritos anteriormente:

- Por encima de 1000px se encuentran los diseños panorámicos que disponen el contenido sin perder el centrado de los elementos y por ende la atención del usuario.
- Por debajo de 1000px se encuentran los diseños destinados a satisfacer las necesidades de las tabletas. Este diseño prescinde grandes márgenes en los elementos y espaciados internos en los contenedores. Además prepara el sentido de verticalidad del dispositivo, con áreas donde *flexbox* y la configuración *flex-wrap* permiten recolocar los elementos en nuevas filas. También resulta interesante en este punto revisar tamaños de fuente de texto. Esta aproximación se conoce como el patrón de retícula ***Mostly fluid*** [42] y consiste básicamente en mantener contenedores cuyo crecimiento será dinámico en función de los puntos de ruptura.
- Por debajo de 570px se presta soporte a los phablets y a los teléfonos móviles. Dentro de la condición de este *breakpoint* se eliminan casi por completo márgenes y *paddings* y se presentan los elementos ocupando prácticamente el ancho total del dispositivo. Se aplica un factor correctivo para los tamaños de fuente. Se recolocan elementos gráficos como el logotipo mediante reglas de estilos basadas en *position*.

Los patrones de interacción utilizados en la zona pública son el formulario de inscripción y el formulario de registro. Ambos formularios están implementados con Angular Material y cuentan con un sistema de validación de campos basado en formularios reactivos de Angular. El formulario efectúa la operación submit únicamente si el estado global del mismo es válido.

Los patrones de interacción de la zona privada acorde con la librería de patrones de Welie [43] son: caja de búsqueda, paginador, thumbnail, homelink.

La Figura 24 describe un mapa de navegación de Mushroom WebApp teniendo en cuenta el proceso de registro/autenticación y la diferencia de roles: usuario común / administrador.

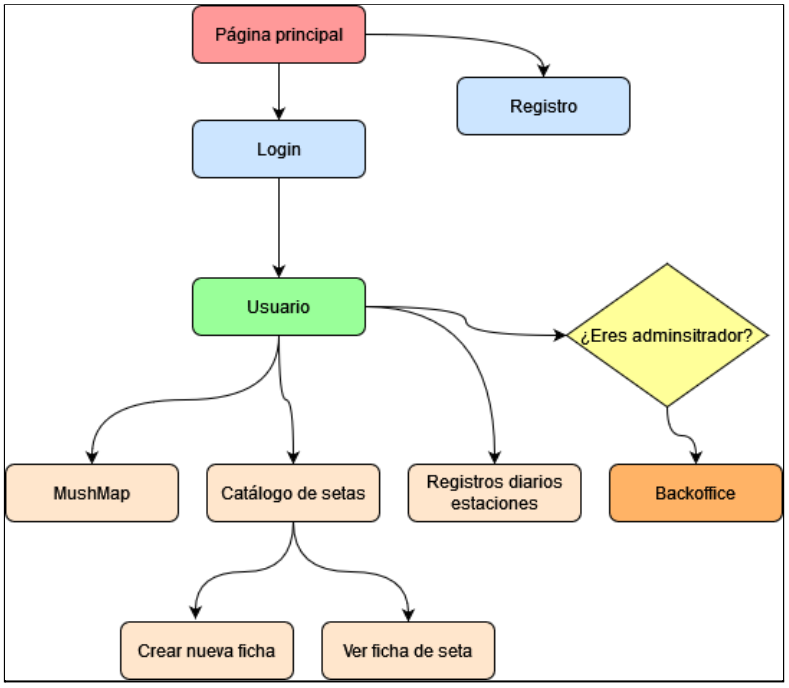


Figura 24: Mapa de navegación de Mushroom WebApp.

17. Seguridad

Subsistema Auth

Tal como cita la documentación de Firestore y los mensajes que se reciben cuando configuras los servicios, no disponer de un conjunto de reglas de uso de la base de datos puede conllevar riesgos.

Firestore deja abierta la posibilidad de utilizar la totalidad de documentos y colecciones, simplemente con estar en posesión de la API KEY vinculada a una cuenta de google y proyecto y configurado debidamente el entorno e importadas las librerías que permiten la comunicación. Este es el modo más desprotegido de todos.

El siguiente paso es habilitar la creación de usuarios en el sistema 'Authentication' y a posteriori programar las reglas de acceso y modificación de datos que corresponda a los usuarios. La regla aplicada para Mushroom WebApp es que al menos, el acceso de a la base de datos se produzca a través de un proceso de autenticación, respecto a un usuario dado de alta en el sistema 'Authentication' tal como muestra la Figura 25.

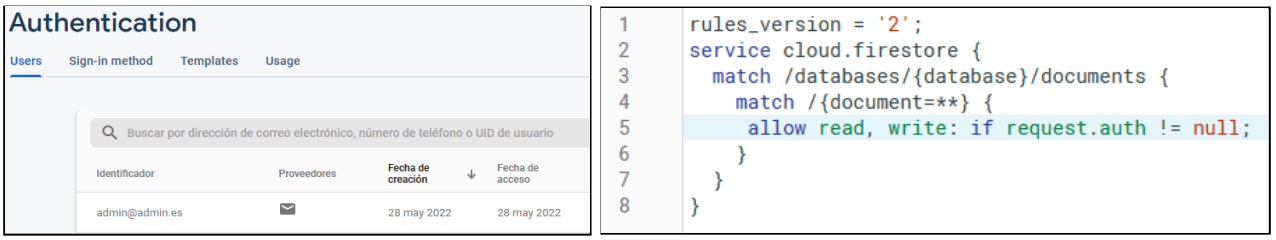


Figura 25: Authentication en Firestore y reglas de uso.

Las posibilidades van más allá. Se pueden vincular contenidos y permisos sobre los mismos directamente a sus respectivos creadores, la separación de roles, etc. Se debe ajustar un juego de reglas preciso si Mushroom WebApp acabase en producción, dado que el número de accesos y modificaciones sobre las colecciones y documentos podría generar costes económicos si se rebasa el plan de utilización elegido.

La propuesta de mejora del consultor sobre la habilitación de un sistema de confirmación a través del envío de un correo electrónico al usuario que se registra en la aplicación no ha sido posible materializarla. El enlace de confirmación que se envía al correo electrónico del usuario para confirmar su registro debe incluir la URL de dominio para redireccionar al usuario a la aplicación. En este caso no se cuenta con esta posibilidad dado que se ha desplegado la aplicación en un servidor Cloudtime AWS [44] de la UOC que genera una IP pública distinta en cada sesión de trabajo.

Certificado autofirmado con OpenSSL

De tal forma que el tráfico de red entre clientes y servidor viaje cifrado se activa para el servidor web Apache un certificado SSL utilizando la herramienta OpenSSL [45]. Los pasos para su generación y habilitación son:

La instalación de OpenSSL:

```
sudo apt-get install openssl
```

Habilitar los modos 'ssl' y de redirección de apache:

```
a2enmod ssl
a2enmod rewrite
```

Inhabilitar el servicio apache2 temporalmente (si estaba activo).

```
sudo systemctl stop apache2
```

La creación y situación sobre el directorio para el certificado:

```
mkdir /etc/apache2/certificate && cd /etc/apache2/certificate
```

La generación del certificado en dicho directorio e introducción de información solicitada:

```
sudo openssl req -new -newkey rsa:4096 -x509 -sha256 -days 365 -nodes -out
apache-certificate.crt -keyout apache.key
```

Se debe prestar especial atención a 'Common name', pues debe apuntar a la IP Pública del servidor.

Editar el archivo de configuración de Apache para el sitio web servido por defecto, con opción de redirección de tráfico HTTP a HTTPS (ver Figura 26):

```
sudo nano /etc/apache2/sites-enabled/000-default.conf
```

```
<VirtualHost *:80>
       RewriteEngine On
       RewriteCond %{HTTPS} !=on
       RewriteRule ^/?(.*) https://%{SERVER_NAME}/$1 [R=301,L]
</virtualhost>
<VirtualHost *:443>
       ServerAdmin webmaster@localhost
       DocumentRoot /var/www/html
       ErrorLog ${APACHE_LOG_DIR}/error.log
       CustomLog ${APACHE_LOG_DIR}/access.log combined
       SSLEngine on
       SSLCertificateFile /etc/apache2/certificate/apache-certificate.crt
       SSLCertificateKeyFile /etc/apache2/certificate/apache.key
</VirtualHost>
```

Figura 26: Configuración de virtualhost en Apache para Mushroom WebApp.

Restablecer el servicio apache:

```
sudo systemctl start apache2
```

Figura 27. Protocolo HTTPS en funcionamiento para Mushroom WebApp.

La Figura 27 muestra como el navegador es capaz de establecer comunicación con el servidor Cloudtime utilizando el protocolo HTTPS.

Nota: dado que el servidor Cloudtime, al iniciarse es publicado en una IP distinta se requiere de la generación del certificado apuntando a esta nueva dirección en cada sesión de trabajo para que funcione correctamente el protocolo HTTPS. El candado verde no aparece debido a que no tenemos un dominio asociado al sitio web, sin embargo esto no es problema para que el protocolo sea reconocido y funcione como se espera.

18. Tests

En el desarrollo de software moderno prima la confiabilidad y la robustez que ofrecen baterías de pruebas efectuadas sobre los proyectos desarrollados. Las tendencias actuales elevan la perspectiva del Test Driven Development [46] a ser una metodología de trabajo que promueva la transformación de requisitos de usuario en pruebas unitarias y que el software que crece de manera incremental lo hace acorde a las directrices que establecen las pruebas.

Las buenas prácticas apuntan a una cobertura de código testado de un 80% de líneas (que no caminos lógicos testados).

Las pruebas unitarias en el ámbito de Angular son desarrolladas con el entorno Javascript de testing Jasmine [47], junto a Karma [48], el entorno web que nos permite la ejecución de los test desarrollados con Jasmine. Karma y Jasmine vienen por defecto con la creación de un nuevo proyecto Angular.

Tras habilitar la cobertura de código en el fichero 'angular.json' se nos creará un directorio con una estructura web informativa sobre el nivel de cobertura que ofrece nuestro conjunto de pruebas.

Los test unitarios se basan en la comprobación de:

- La creación exitosa de componentes.
- La realización exitosa de un trabajo por parte de un método (por ejemplo de un servicio o un pipe).
- La comprobación de que se enruta correctamente según la interacción del usuario y un objeto de tipo espía que escucha el enrutado que se produce en todo momento..

Cada fichero '`.spec`' deberá importar y preparar aquellas dependencias externas requeridas, de tal forma que se asegure la atomicidad máxima de la prueba, haciendo que los resultados sean los más objetivos posibles y correspondan únicamente a la entidad software a testar.

Las dependencias deben ser provistas tal como son inyectadas estas en los constructores de los servicios / componentes. En el caso de Mushroom WebApp, para poder testar el componente '`listarSetas`', se debe proveer el servicio que recoge el catálogo entero y a su vez, debemos proveer las librerías de Firestore que permiten su funcionamiento.

La sucesión de sentencias '`it`' conformará cada uno de los test embebidos en en un fichero '`.spec`', siendo karma quien evalúe el total de ficheros de test y muestre el balance de cobertura de código.

19. Versiones de la aplicación

La versión entregada de la aplicación ‘Mushroom Web App’ en la PEC4 es por su grado de madurez una versión 1.0 Alpha con vistas a ser continuada.

Las sucesivas versiones de la aplicación deberán:

- Revisar el tema de CORS expuesto apartado ‘Bugs’. Las líneas de investigación deberán asegurar que:
    - No es una configuración faltante de Apache o Netlify [49].
    - Que la librería utilizada para gestionar los mapas tenga otros métodos de carga de API o plantearse la utilización de una librería equivalente como es AGM [50]

  En definitiva, la aplicación debería funcionar bien aún con el bloqueador de anuncios habilitado.

- Migrar el backend ahora presente en UOC Cloudtime AWS a un alojamiento que permita la recogida y actualización de datos de la aplicación cliente. Es requisito pues, disponer de un funcionamiento 24/7 del demonio cron que dispara el *script* que efectúa la recolección y manda persistir la información.

- Trabajar con iconografía SVG. Uso de imágenes en formato ‘webp’ que sustituyan aquellas con formato ‘png’ y ‘jpeg’. Se debe aligerar el tiempo de carga de las imágenes.

- Experiencia de usuario más amigable e información más detallada. Las líneas de trabajo en este aspecto son la posibilidad de personalización de contenidos para el usuario o la recepción de un feedback inequívoco tras cada acción ejecutada (Barra de estado / hints).

20. Requisitos de instalación y uso

Requisitos de instalación

Para poder disfrutar de una experiencia de usuario correcta en el uso de Mushroom Web app, preferentemente se debe disponer de:

- Un ordenador / portátil o dispositivo inteligente, bien sea móvil o tablet que disponga de conexión a internet.

- Servicio de internet de banda ancha. Sería deseable fibra óptica e internet por cable en las estaciones de trabajo fijas. Conexión Wi-Fi con suficiente potencia para equipos adscritos a una red local. Tarifa de datos móviles con al menos 4G y buena señal de cobertura para los dispositivos móviles que trabajen de forma independiente. La aplicación depende de un cacheo de elementos multimedia en forma de fotografías y datos del sistema Firestore para habilitar su estado.

- El equipo debe contar con un procesador multi-core y una dotación mínima de procesamiento gráfico para poder renderizar la aplicación ofreciendo una buena experiencia de usuario que generalmente encontramos en la mayoría de las gamas de equipos actuales.

Requisitos de uso

Conocida la URL de publicación de la parte front-end se accede a la aplicación solicitando la página a través del navegador obteniendo una carga instantánea si se dispone de los requisitos anteriormente mencionados.

En la versión actual de Mushroom WebApp el proceso de login está siempre abierto.

El proceso de registro es dependiente de si un administrador habilita o no un botón switch que permita la entrada de nuevos usuarios en la comunidad (se preserva un crecimiento descontrolado de usuarios).

Con el proceso de registro completado el usuario es redirigido de nuevo al login.

El usuario autenticado podrá acceder a las zonas: mapa de setas, catálogo de fichas de especies y a la gestión de estaciones meteorológicas. Si el usuario resulta tener poderes de administrador podrá acceder a un backoffice de gestión de fichas, estaciones y usuarios.

21. Instrucciones para el tribunal evaluador

La aplicación Mushroom WebApp ha sido desplegada de forma oficial, de acuerdo con las directrices de la PEC4 en un servidor de pruebas público y accesible. En concreto se halla alojada en el servidor AWS Cloudtime, el cual ha sido proporcionado por la UOC, a petición del alumno y consultor.

Para acceder al servidor Cloudtime debo ingresar mis credenciales del campus UOC, dirigirme a la asignatura M4.259 Trabajo Final de Máster y pulsar sobre el apartado 'Cloudtime'. En este apartado se me proporciona información temporal y exclusiva para mi de la sesión que puedo iniciar si arranco la máquina tal como muestra la Figura 28.

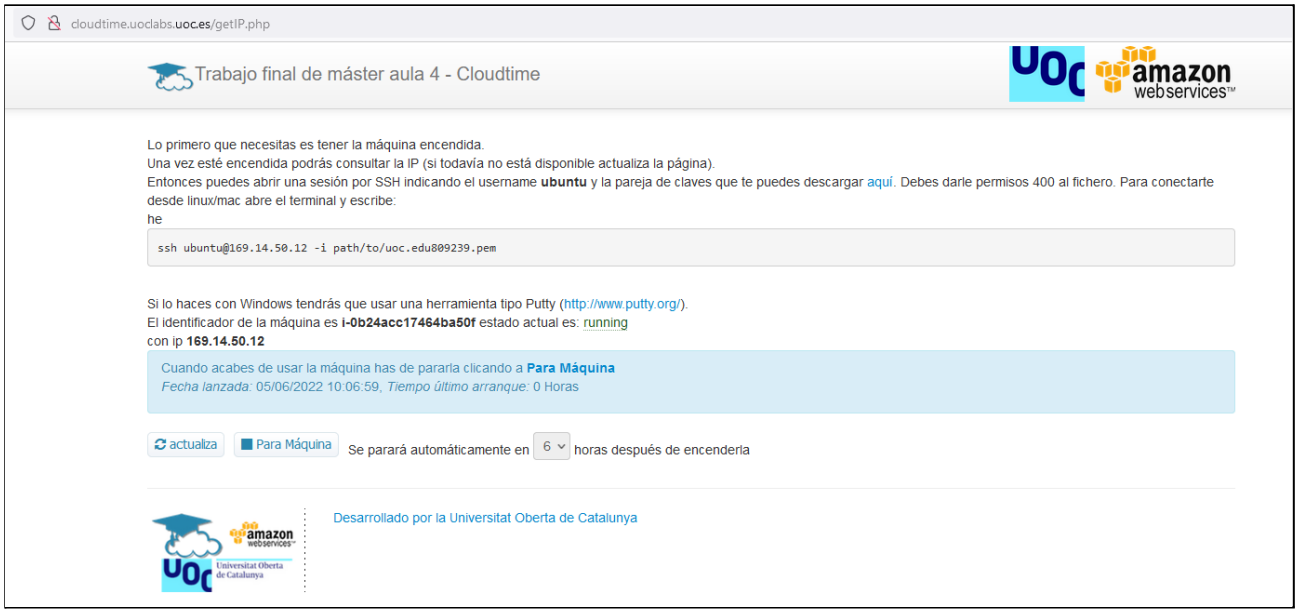


Figura 28: Interfaz web de UOC AWS Cloudtime (Han sido variadas las IPs públicas).

El identificador de la máquina es estático dado que es la entidad que permite apuntar a la parcela asignada de trabajo. Sin embargo la IP pública a través de la cual acceder al sitio web una vez desplegado es cambiante y depende de la sesión en activo. Las sesiones se pueden prolongar de 1 a 6 horas máximo.

Esta web cuenta también con la proporción de una clave privada RSA que deberá ser procesada en PuttyGen y cargada en Putty [50] para la conexión por terminal con el servidor mediante SSH. Para la subida de archivos se ha utilizado Filezilla. De nuevo filezilla requiere de la configuración de puerto, usuario y clave RSA, brindando finalmente acceso al servidor.

El desarrollo efectuado se sitúa en las siguientes ubicaciones principales:

`/home/ubuntu`
En esta ubicación encontramos los ficheros de código Javascript que serán ejecutados por el motor V8 [X] de Node dentro del directorio '`MushroomBE`'.

`/etc/apache2/sites-enabled/000-default.conf`
Fichero de configuración del virtualhost

`crontab -e`
Fichero de configuración del demonio cron

`/var/www/html`
Despliegue de la aplicación angular con la totalidad de archivos minificados HTML/JS/CSS y *assets* complementarios que proporciona el build de Angular para el desarrollo front-end Mushroom WebApp.

Nota: Quedo a disposición del tribunal evaluador si requiere del acceso al servidor a través de las credenciales que yo pueda generar respecto al acceso al servidor de pruebas UOC AWS Cloudtime:

- Durante el tiempo de evaluación del trabajo realizado.
- Durante la sesión que se celebrará de la defensa virtual.

Mi correo es: dgarciagarcia6@uoc.edu

De igual modo y dado que los repositorios de código en la nube tienen ficheros con configuraciones de entorno, API KEYS e información encastada en index.html sensible de producir costes económicos si cae en manos indebidas, la visibilidad de los mismos es privada. Por tanto, será el código liberado en el fichero comprimido de la 4a PEC donde se incluya el total de trabajo elaborado (pudiendose reproducir el entorno de desarrollo y despliegue en otras máquinas).

Aún con todo, quedo a disposición del tribunal evaluador para admitir como colaboradores a sus usuarios Github respecto a los repositorios privados utilizados para el desarrollo de Mushroom WebApp:

- https://github.com/DioniGarciaUOC/Mushroom-WebApp.git
- https://github.com/DioniGarcia/MushWebAppBack.git

22. Instrucciones de uso

Registro de usuario. Ubicado en la página de inicio puede encontrarse abierto o cerrado si el administrador considera que se permite la entrada de nuevos usuarios. Se trata de un formulario donde se precisa la introducción de nombre, apellidos, correo electrónico y contraseña. Si la validación de campos es correcta y la entrada de nuevos usuarios está habilitada se procede a registrar al usuario con el botón de envio. El usuario es reconducido a la página de *login*.

Autenticación de usuario (*login*). Mediante la cuenta de correo y contraseña proporcionadas en el proceso de registro el usuario se autentica en el sistema. Si la información proporcionada corresponde con un usuario válido del sistema este será enrutado al área del mapa de salida de setas.

Mapa de salida de setas. Primer área privada que un usuario autenticado puede visitar. Se ofrece un mapa de Google Maps centrado en el Sistema Ibérico este, donde un conjunto de iconos contextualizados en el mundo fungi otorgan una predicción de salida de setas en unas determinadas localizaciones. El mapa dispone de tres botones funcionales de zoom y centrado, así como la posibilidad de ampliar el mismo a pantalla completa. El mapa es acompañado por una leyenda que aclara el significado de la iconografía empleada.

Gestión de fichas técnicas de setas. Segundo área privado. Permite ver y filtrar un listado de tarjetas sobre conjunto de setas dadas de alta como ficha técnica. En la ventana del listado es posible la adición de nuevas fichas mediante un botón con un símbolo más que habilita un modal para la complimentación de los datos: nombre científico, nombre común, descripción de color, hábitat, comestibilidad, características morfológicas, una URL de imagen y un apartado de observaciones. Al clicar en la tarjeta se conduce al usuario a una página de detalle ampliado de la ficha técnica en cuestión.

Gestión de estaciones. Tercer área privada. Análoga a la gestión de fichas técnicas, se presentan la totalidad de las estaciones en formato tarjeta con la metereología del momento como: temperaturas, viento, humedad y precipitación. Al clicar en cada una de estas tarjetas se conduce al usuario a una página de detalle ampliado que ofrece recuentos de lluvia mensuales, anuales o de los últimos 5 días, así como información complementaria geográfica.

Gestiones backoffice. Área únicamente visible y accesible para a los usuarios administradores. Muestra unas tablas de gestión para fichas técnicas y estaciones dadas de alta que permiten la paginación y el filtrado. Este área dispone de botones switch para habilitar la creación de nuevos usuarios y regular si estos tienen el rol de administrador o no.

23. Bugs

Mapa de salida de setas

Los botones de control de zoom in / out / centrado parar el mapa de salida de setas no obedecen al primer clic. No se generan errores por consola. No es el comportamiento esperado pero este está acotado a la carga del módulo y componente del mapa y es anómalo durante las dos primeras pulsaciones, dado que a partir de la tercera ya funciona sin problemas. No se trata de un error atribuible a un navegador, Firefox [51] y MS Edge [52] lo reproducen exactamente igual.

Cors

La aplicación ha sido desplegada en dos servidores:

- UOC AWS Cloudtime: front-end y back-end.
- Netlify: front-end (URL: https://capable-flan-a8d54e.netlify.app)

Al utilizar en los navegadores de desarrollo Ublock Origin, al producir la carga de la API de Google Maps en el index.html tal como sugiere la documentación oficial de Google en modo 'inline', en la consola del navegador aparece un mensaje de CORS sin que la carga de la página o funcionamiento de la aplicación se vea afectada.

La página del mapa de salida de setas se encuentra en un módulo privado, cuyo acceso está custodiado por un guard y para el que el fichero de Typescript de su lógica efectúa una recarga de la API de Maps mediante JSONP. Esta solución ha sido encontrada en la documentación de la librería que se utiliza para poder gestionar los mapas de Google Maps (URL: https://github.com/angular/components/tree/main/src/google-maps#readme).

Tras varios despliegues en Netlify y Apache (Cloudtime AWS) y ajustar los parámetros de ficheros de redirección en Netlify o la habilitación del módulo 'headers' y la adición de permisividad de CORS en el virtualhost que gestiona Apache para el sitio web no consigo eliminar el error.

Si elimino la carga de la API en el index.html veo que tengo una carga inicial del home sin mensajes de CORS.Sin embargo, **no es hasta cuándo desactivo el bloqueador de anuncios Ublock Origin cuando me doy cuenta de que ya no aparece este mensaje en la consola del navegador**.

24. Proyección a futuro

Las posibilidades de ampliación del proyecto a corto o medio plazo son muchas, posibles y muy variadas. En ningún caso se contempla la subida a un repositorio público o un aprovechamiento comercial. Algunas de las mejoras a proponer son:

Añadir nuevas estaciones meteorológicas:

Podrían añadirse estaciones meteorológicas no contempladas en la aplicación siempre y cuando estas pertenezcan a las redes de estaciones: AEMET / AVAMET / Meteoclimatic. Cabe decir que el elemento clave que permitiría dar de alta e informatizar una nueva estación sería el ID de gestión dentro del organismo a la que pertenece. Este IDs es el elemento que permite que las funciones de obtención de datos puedan practicar la recogida de datos para una determinada estación. La dificultad de implementación sería baja y en términos generales conlleva la creación de un formulario de alta de estación en el *backoffice*.

De surgir una nueva red de estaciones de interés, debería habilitarse una función de obtención de datos basada en la utilización de una API de consulta de datos o la técnica de web scraping si esta no existiera.

Añadir nuevas estaciones meteorológicas trianguladas

Podrían añadirse estaciones meteorológicas cuya situación geográfica esté comprendida en distancia mínima entre tres estaciones ya dadas de alta y nutrirse de una “información en conjunto”. Esta información es todavía más imprecisa que la de las propias estaciones, pero interesante igualmente, más aún, si los registros meteorológicos de las estaciones que proveen la triangulación de datos registran valores similares. La dificultad de esta implementación es media-alta: debería calcularse la idoneidad por proximidad de las tres estaciones que triangularán a la nueva estación. Deberá establecerse un criterio para “poblar” los datos de la nueva estación, donde quizás las estaciones más próximas sean más influentes que el resto.

Determinar qué especies de setas aparecen en un determinado lugar en un instante concreto

Se podría concretar sobre qué especies de setas podrían aparecer para una estación determinada. Una primera aproximación comprendería el habilitar categorías para referirse al tipo de vegetación en las fichas técnicas y en las estaciones, de tal modo que puedan hacerse asociaciones de lugares y tipos de setas. Para afinar más, podríamos definir para las setas unas cotas de altitud mínima y máxima asociadas a su aparición en el monte. La complejidad de esta implementación es baja-media por el cambio a realizar en los modelos de datos y el cruce de datos de los mismos.

25. Presupuesto

25.1 Mano de obra

Los costes de mano de obra se estiman en 30€ / hora con independencia del rol técnico involucrado en una determinada fase de desarrollo del proyecto. Las jornadas de trabajo serán de 8 horas diarias.

El presupuesto incluye desde las fases de análisis y viabilidad del proyecto hasta el despliegue de la aplicación y formación de usuarios potenciales. El presupuesto contempla y no más, el desarrollo de funcionalidades descrito en la fase de análisis. La adición de nuevas características físicas o lógicas conllevará una variación en el presupuesto y recursos inicialmente planteados y por tanto, reflejará un coste económico distinto del mismo.

Llegados al punto de haber desplegado la aplicación correctamente y verificado su funcionamiento íntegro, se proporcionará la posibilidad de realizar durante los 15 días posteriores un mantenimiento correctivo en la dirección de subsanar errores de desarrollo o funcionalidades más relevantes. La Tabla 1 muestra un desglose de los conceptos atribuibles a la mano de obra.

Concepto	Tiempo	Horas	Importe
Definición del proyecto: viabilidad	3 días	24h	720€
Planificación: temporal y de recursos	2 días	16h	480€
Análisis y refinación de requisitos	4 días	32h	960€
Diseño de prototipos	6 días	48h	1440€
Desarrollo y test de la aplicación	12 días	96h	2880€
Despliegue, retirada de recursos y test de integración	4 días	32h	960€
Formación de usuarios	2 días	16h	480€
Total estimado:			**7920€**

Tabla 1: Importe estimado en concepto de mano de obra.

25.2 Equipamiento técnico

Se contará con un perfil técnico analista/desarrollador con perfil full-stack en tecnologías web.

El equipo de trabajo será un sobremesa dotado de:

- **Torre**: Intel Core I5 de 4a generación. 4 núcleos a 3.7Ghz. 8GB de RAM. 256 SSD.

- **Pantalla**: 24 pulgadas de cristal curvo FullHD conectada por HDMI.

Las consultas a OPEN DATA de AEMET no generan coste económico alguno.

Google Maps Platform promueve que su plataforma puede ofrecer hasta 28.500 cargas de mapas al mes sin coste económico. No hay previsión de rebasar esta limitación y por tanto no se prevé coste alguno en su integración para Mushroom WebApp.

El plan Spark de Firebase promueve:

- Hasta 1GiB de datos almacenados en el cloud de Firestore sin coste.
- Las “cloud functions” están inhabilitadas en este plan.
- 20K operaciones de escritura por mes.
- 50K operaciones de lectura por mes.
- 20K operaciones de borrado por mes.

La contratación de un servidor VPS es una parte vital para esta aplicación. Necesitamos que un demonio cron active la recogida de datos.

Podría estudiarse la opción de realizar las labores de servidor con un microcontrolador Raspberry PI o similares, el requisito en estos casos es que cuente con conectividad a internet, suministro eléctrico continuo y disponibilidad de recursos para ejecutar los *scripts* de actualización de datos. En cualquier caso, es conveniente que el equipo se dedique principalmente a esta labor, dado que de su desempeño horario/diario depende la consistencia de datos que sustenta la aplicación.

Podemos contratar un servidor VPS en Axarnet por 29 euros mensuales o en un pago anual de 347€.

La Tabla 2 reune el total de conceptos asociados a la infraestructura requerida.

Recursos	Periodicidad pago	Importe estimado	Importe anual estimado
Dominio (.es)	anual	14.95€	14.95€
Certificado SSL	anual	20€	20€
Servidor VPS Axarnet	mensual (precio pack anual)	29€	347€
Importe total estimado anual			**381€**

Tabla 2: Importe estimado en concepto de infraestructura.

26. Análisis de mercado

Audiencia potencial

Las opciones de explotación o comercialización de Mushroom Web App como un servicio o herramienta de apoyo son variadas. Una primera aproximación sobre la segmentación de potenciales usuarios de Mushroom WebApp que pueden beneficiarse es:

- La gestión inteligente de acotados de setas privados o destinados a la explotación micológica.
- Reclamo para pueblos y enclaves paisajísticos que quieran dar a conocer sus tierras y bosques.
- Sistema de gestión / formación para asociaciones micológicas.
- Lanzamiento de la aplicación como un servicio exclusivo de pago por uso.

El mayor problema que presenta la aplicación a priori es la contextualización de los lugares y zonas de los que se pretende dar soporte informativo acerca de la probabilidad de aparición de especies. El crecimiento de número de usuarios sería seguro un handicap añadido, sujeto según la implementación actual a condiciones de pago por uso de Google Maps y Firestore respecto al número de lecturas/escrituras de sus servicios web.

Competencia

A continuación se presenta un posible competidor a nivel estatal que opera bajo financiamiento público y que se consolida tras más de una década de actividad en el sector de desarrollos tecnológicos orientados al mundo fungi.

FungiGo [55] es una iniciativa basada en un modelo piloto de “Parques micológicos inteligentes” de Castilla León y Teruel donde se promueve la sostenibilidad de los lugares y la comercialización del recurso con las máximas garantías sociales. FungiGO trabaja conjuntamente con ECM Ingeniería Medioambiental [56], una entidad con base tecnológica donde en su web se publicitan como expertos con más de diez años desarrollando soluciones específicas en el campo de la micología.

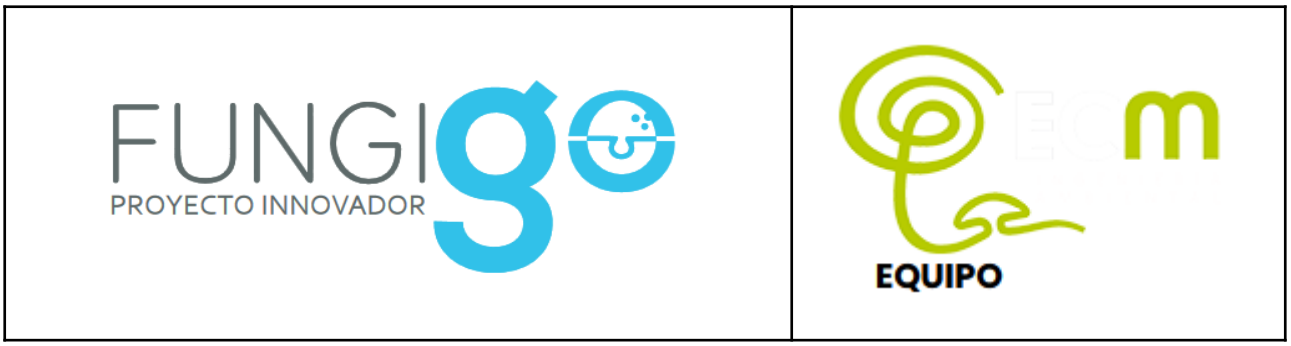


Figura 29: Logotipos de FungiGo y ECM Ingeniería.

Los productos de ECM Ingeniería Medioambiental en algún punto pueden ser competidores directos si Mushroom WebApp es presentada como candidata de reemplazo para algún proyecto de los desarrollados por ECM.

Márgenes de precios

Según la web de FungiGo han recibido más de medio millón de euros de financiación desde el Fondo Europeo Agrario de Desarrollo Rural y de la Administración General del Estado. En este sentido, una partida presupuestaria como la descrita por FungiGo permitiría sufragar los gastos de un equipo desarrollador dedicado a generar Mushroom WebApp como mínimo cubriendo los gastos descritos en el apartado anterior “Presupuestos”.

La liberación del producto para la ciudadanía en un formato de aplicación de uso libre sería entonces una opción al haber recibido fondos públicos. Otra opción es preservar Mushroom WebApp de forma total o parcial como herramienta de gestión interna de las administraciones para la supervisión de campañas futuras de recolección de setas, donde se promuevan buenas prácticas de preservación del ecosistema.

El ciclo de vida de Mushroom WebApp en el nicho de la empresa privada consistiría básicamente en la confección del sistema y la concesión de explotación de la misma en las condiciones que se acuerden entre la entidad desarrolladora y la empresa que adquiere los derechos de explotación y/o exclusividad. La empresa que explota el sistema decidirá (si los derechos adquiridos se lo permiten) aplicar un pago por uso basado en suscripción anual o de temporada para el usuario final. Dada la singularidad del servicio, podrían aplicarse precios moderados de suscripción si es que finalmente se puede ofrecer un grado de certeza en la información proporcionada para los buscadores de setas estimados en:

- Cuota anual de acceso a los servicios Mushroom WebApp: 60€
- Cuota de temporada (estación del año o 3 meses) a los servicios Mushroom WebApp: 30€

27. Marketing y Ventas

En el momento que Mushroom WebApp pasa a ser un servicio consumible por las administraciones o comunidad de usuarios finales hemos de pensar en una estrategia de márketing y ventas.

Si Mushroom WebApp es **explotado en el nicho de las administraciones públicas** dos opciones son posibles:

Caso nº 1: La herramienta sirve de apoyo interno para la gestión inteligente de los recursos y los ciudadanos no tienen acceso a la misma.

En este caso no hay mayor necesidad de branding que la de consolidar en la propia herramienta un distintivo basado en el nombre del producto y entorno operativo con tal que el personal técnico que la utilice reconozca el sistema debidamente y pueda acceder a él con facilidad.

Caso nº2: La herramienta se financia con fondos públicos y está enfocada a la promoción, formación y culturización general del ciudadano.

Este caso precisa un presupuesto extra destinado principalmente a dos objetivos bien diferenciados:

- Proveer a la aplicación de una interfaz, logotipos y detalles de accesibilidad sumamente cuidados de tal modo que la aplicación pueda ser utilizada por el mayor número de usuarios posible, teniendo especialmente presente las gentes de otros países que podrían contribuir a agrandar el turismo de paso por el lugar.

- Publicitar debidamente la herramienta en redes sociales, portales web de entidades colaboradoras con la administración local o emplear técnicas de cartelería o señalización en acotados de setas. La confección de rutas micológicas provistas de señalización dotada de QR 's que enlacen a determinadas áreas de la aplicación podría ser una forma de enlazar el medio natural con la solución tecnológica Mushroom WebApp. Una estratégia Search Engine Optimization (SEO) sería necesaria para hacer más visible el producto si finalmente este se asienta como un producto de naturaleza web.

Si Mushroom WebApp es **explotado en el nicho de la empresa privada** para su comercialización en un modelo de pago por uso.

No es que el mero hecho de que un producto pertenezca al sector privado este deba someterse a exigencias o controles de mayor calidad. Lo que sí es cierto es que, tanto si Mushroom WebApp irrumpe en

un timing donde es presentado al mercado como un producto innovador o este ya cuenta con un conjunto de competidores, es este hecho, la alta probabilidad de que el futuro deba de competir con otros productos que ofrezcan servicios similares el estímulo que empujará a crear una marca e imagen diferenciadoras.

En este sentido se recomienda el encargo del diseño gráfico profesional de logotipos, layouts y componentes, así como el ajuste de colorimetría acordes a la naturaleza del producto. Deberán registrarse estas decisiones y trabajos implementados de tal forma que un competidor no pueda apropiarse de forma indebida de la imágen del producto y por tanto hacer competencia desleal.

Las técnicas de SEO cobran igual o mayor importancia que cuando Mushroom WebApp se enfoca al sector de las Administraciones Públicas. Un potencial cliente que no es capaz de encontrar nuestro producto y si lo hace con los de la competencia es un miembro menos de nuestra comunidad de usuarios si es que los servicios ofrecidos por la competencia llegan primero y son atractivos a ojo de nuestro posible candidato.

Las campañas en redes sociales para **publicitar la herramienta**, así como la adición de publicidad en los espacios que nos puedan ceder socios colaboradores son vitales también. Las estrategias en este sentido son variables y van desde el intercambio de favores pactado de publicitar herramientas de asociaciones o entidades “afines” o el pago explícito por ceder espacios publicitarios.

Con una estrategia de monetización sobre Mushroom WebApp, la construcción de una marca y la publicitación de la misma son decisiones que tienen un impacto directo sobre el usuario final. Por tanto, es conveniente que como mínimo esta parte sea un encargo externalizado a personal cualificado en estas áreas, sobre todo en lo que respecta al establecimiento inicial de la marca del producto y estrategias asociadas de márketing.

28. Conclusiones

Mushroom WebApp supone para mi un acercamiento a un conjunto de tecnologías web, procesos y formas de trabajo que actualizan y que estructuran mejor mi entendimiento sobre cómo se constituye una aplicación moderna basada en la web 3.0.

Mi background de Grado en Informática junto con el estudio autodidacta y la preparación como opositor para el cuerpo de profesores de secundaria con especialidad informática ha permitido que pueda seguir (con algunas dificultades asociadas a la escasez de tiempo) la consecución del presente proyecto, junto con las PEC de las asignaturas de segundo semestre y además compaginar un trabajo como docente a tiempo completo.

Bajo mi punto de vista **ha sido crucial el poder elegir la realización de un proyecto afín** al estudiante. En momentos donde la confluencia de fechas de entrega para las PEC junto al trabajo semanal como docente de secundaria es realmente difícil sobrellevar la carga de trabajo y aparecen sentimientos de desánimo o falsas sensaciones de no poder compaginar el máster con la actividad laboral. En este sentido quedo muy agradecido con las incontables muestras de apoyo del tutor asignado, así como de los profesores al cargo de las diferentes asignaturas, pues se nota que tienen presente el conjunto de alumnos que tratan de compatibilizar estudios y trabajo.

Aún con todo, soy totalmente consciente que este máster no supone más que un inicio en una de las múltiples especializaciones en las que las tecnologías de la información y la comunicación contribuyen, la creación de aplicaciones y sitios web. La complejidad inherente que tiene la construcción de un sistema de estas características no pasa desapercibido. Los actuales frameworks JS como el estudiado en el máster, Angular, dan soporte para poder comunicar prácticamente cualquier dispositivo o servicio de terceros, dando tratamiento a los distintos flujos de datos entrantes/salientes, cumpliendo los mínimos de seguridad y consistencia, a la vez que promueven la construcción modular y mantenible de las aplicaciones.

Pese a establecer directrices de trabajo estándares dentro de los equipos de desarrollo el handicap que comporta la construcción de pro ejemplo un dashboard mínimamente contextualizado y elaborado hace que la personalización en el desarrollo web, por fortuna, siga siendo un campo reservado al personal técnico formado y especializado.

Anexo 1. Entregables del proyecto

La Tabla 3 muestra el detall de los archivos liberados en las diferentes entregas PEC es:

PEC1	Documento de viabilidad del proyecto y análisis de funcionalidad preliminar. Primera versión de la memoria de trabajo final de máster con el índice de trabajo y partes cumplimentadas respecto a la viabilidad y descripción funcional de Mushroom WebApp.
PEC2	Entrega de un archivo comprimido en el REC con: Ampliación de los apartados de la memoria que se indican en el enunciado de la PEC y que establecen las fases de análisis y diseño principalmente. Entrega de los primeros ficheros de código donde se inicia una aplicación angular y existe conectividad con Firestore y la API de Google maps con unos sencillos ejemplos.
PEC3	Directorio de drive con 4 videos de defensa de la PEC3: https://drive.google.com/drive/folders/1IsOIX6dRptejd9H9NWE1I1aOPyrPT3jG?usp=sharing Adición del tutor como miembro colaborador del repositorio privado de Github para la parte de *scripts* Node.js: https://github.com/DioniGarcia/MushWebAppBack.git Adición del tutor como miembro colaborador del repositorio privado de Github para la parte front-end Angular: https://github.com/DioniGarciaUOC/Mushroom-WebApp.git El servidor AWS con el entorno instalado y publicando Mushroom WebApp en abierto. Incremento parcial de la memoria de trabajo fin de máster.
PEC4	Entrega de un archivo comprimido en el REC con: ● Un directorio para la totalidad del código desarrollado: parte front y back ● Una presentación en formato PDF dirigida al público general e inversores. ● Un autoinforme en formato PDF. Subida al repositorio Prensent@ de la UOC de una video-defensa de 20 minutos de duración: 15/16 para presentar el proyecto/producto y 5 de demostración del producto software elaborado.

Tabla 3: Archivos liberados en las diferentes entregas PEC.

Anexo 2. Extractos de código fuente

El presente anexo mostrará algunos de los extractos de código más relevantes con respecto a la codificación de Mushroom WebApp. No es de extrañar que estos correspondan a miembros internos de los servicios programados, dado que es en estos ficheros donde se traslada la lógica de programación más compleja o candidata a ser reutilizada.

El primer fragmento muestra el método '**`crearUsuario`**' contenido dentro del servicio '`authService`'. Como se puede apreciar realiza la creación de un usuario en el sistema "Authorization" de "Firebase" y añade a una pequeña colección de usuarios, infomación de registro que será tratada a posteriori como información de perfil.
La Figura 30 muestra la lógica exacta programada para la creación de un nuevo usuario.

```
crearUsuario(
  email: string,
  password: string,
  nombre: string,
  apellidos: string,
  telefono: string
) {
  return (
    this.auth
      //Creamos usuario en Auth
      .createUserWithEmailAndPassword(email, password)
      Complexity is 3 Everything is cool!
      .then((cred) => {
        //Creamos usuario con información extendida en una BD de usuarios

        this.firestore
          .collection('admin')
          .doc('admin')
          .valueChanges()
          .subscribe((adminInfo: any) => {
            this.firestore.collection('users').doc(cred.user?.uid).set({
              isAdmin: adminInfo.newUsersWillBeAdmin,
              nombre: nombre,
              apellidos: apellidos,
              email: email,
              telefono: telefono,
            });
          });
      })
  );
}
```

Figura 30: Método '`crearUsuario`' del servicio '`authService`'.

Los siguientes fragmentos presentados por la Figura X corresponden a los métodos ‘verificarSiEsAdmin’ y ‘desloguearUsuario’ también pertenecientes a ‘authService’. ‘verificarSiEsAdmin’ participa leyendo la información de un determinado usuario en la colección ‘Users’ y varia el estado del componente Header con una operación propia de observables ‘.next’ que efectúa una actualización. ‘desloguearUsuario’ comprende una lógica simple pero necesaria. Se obtienen comportamientos anómalos si no se desloguean correctamente los usuarios activos en Firestore. La Figura 31 muestra ambos métodos con su implementación.

```
verificarSiEsAdmin(uid: string): void {
  this.firestore
    .collection('users')
    .doc(uid)
    .valueChanges()
    .pipe(take(1))
    .subscribe((userData: any) => {
      this.headerManagement.next({
        isAdmin: userData.isAdmin,
        logged: true,
        nombre: userData.nombre,
        apellidos: userData.apellidos,
      });
    });
}
```

```
desloguearUsuario(): void {
  this.auth.signOut().then(() => {
    this.headerManagement.next({
      logged: false,
      isAdmin: false,
      nombre: 'ERASED_USER',
      apellidos: 'ERASED_USER',
    });
  });
}
```

Figura 31: Métodos ‘verificarSiEsAdmin’ y ‘desloguearUsuario’.

La Figura 32 muestra la lógica de ‘getIconFromProbability’, un método del servicio estacionesService.ts que recibiendo un número entero en el rango de [0-4] elige un asset basado en un icono para el mapa de salida de setas, respondiendo a la necesidad de graficar la probabilidad de que exista una florada de setas en un lugar concreto.

```
private getIconFromProbability(probability: number) {
  let iconLocation = '';

  switch (probability) {
    case 1:
      iconLocation = '../../../assets/icons/legend-map/mush-grown-sm.png';
      break;
    case 2:
      iconLocation = '../../../assets/icons/legend-map/mush-growing-sm.png';
      break;
    case 3:
      iconLocation = '../../../assets/icons/legend-map/mush-frozen-sm.png';
      break;
    default:
      iconLocation = '../../../assets/icons/legend-map/mush-drown-sm.png';
  }

  return iconLocation;
}
```

Figura 32: Lógica de selector para 'getIconFromProbability'.

No menos relevante es '`getMarkersAndProbability`' un método que según muestra la Figura X permite la carga del conjunto de iconos/markers para las estaciones meteorológicas dadas de alta en el sistema. Tal y como se puede apreciar, es el objeto '`markerOptions`' el que define aspectos de personalización avanzados para un determinado icono / marker. La Figura 33 muestra la lógica del método en cuestión.

```
getMarkersAndProbabilityFS(): Observable<any> {
  let markers: any = [];

  this.getEstacionesFS().subscribe((estaciones) => {
    estaciones.forEach((estacion: Estacion) => {
      markers.push({
        position: {
          lat: parseFloat(estacion.lat),
          lng: parseFloat(estacion.long),
        },
        markerOptions: {
          draggable: false,
          animation: google.maps.Animation.BOUNCE,
          icon: this.getIconFromProbability(estacion.codigoIconoSeta),
          visible: this.isVisible(estacion.codigoIconoSeta),
          title: estacion.nombre,
        },
      });
    });
  });
  return of(markers);
}
```

Figura 33: Obtención de la infomación personalizada de iconos con 'getMarkersAndProbability'.

Un peso pesado en el desarrollo de Mushroom WebApp es sin duda el método NodeJS ‘`writeRegisters`’. Tal como muestra la Figura 34, ‘`writeRegisters`’ lee en primera instancia el posible *flag* adscrito a su invocación. Ello determinará la bifurcación de su lógica. En función del tipo de estación ‘`writeRegisters`’ llama a función especializada en la obtención de datos. Finalmente según la adición del *flag* o no, escribe la tupla en la matriz de los 30 días o actualiza la información actual de la climatología de las estaciones dadas de alta en el sistema.

```
let isHourlyUpdate = false;

if(process.argv[2] && process.argv[2] === '-hourly') isHourlyUpdate = true;

const querySnapshot = await getDocs(collection(db, "estaciones"));
Complexity is 9 It's time to do something...
querySnapshot.forEach((statDoc) => {

    const stat = statDoc.data();
    const statType = stat.tipoEstacion;

    if (statType === 'avamet') {

        scrapAVAMET(stat.idAgencia).then((newRegister) => {
            writeNewRegister(newRegister, stat, stat.nombre, isHourlyUpdate);
        })

    } else if (statType === 'aemet') {

        scrapAEMET(stat.idAgencia).then((newRegister) => {
            writeNewRegister(newRegister, stat, stat.nombre, isHourlyUpdate);
        })

    } else {

        scrapMeteoclimatic(stat.idAgencia).then((newRegister) => {
            writeNewRegister(newRegister, stat, stat.nombre, isHourlyUpdate);
        })
    }

    if(!isHourlyUpdate){

        //Obtenemos el codigo de icono
        const codigoIcono = evaluateStation(stat);
        console.log("El código de icono para "+stat.nombre+" es: "+codigoIcono) // REMOVEME
        stat.codigoIconoSeta = codigoIcono;

        //Actualizamos los diferentes acumulados de lluvia en la estación (mensual/anual/ultimos 5 dias..)
        updateStatRains(stat);

        // Updateamos la estacion en Firestore
        const estacionesRef = doc(db, 'estaciones', stat.nombre);
        setDoc(estacionesRef, stat);
    }
});
```

Figura 34: Lógica principal de ‘writeRegisters’.

A modo de ejemplo la Figura 35 muestra '`scrapMeteoClimatic`' un método que recibe el identificador de una estación dentro del ámbito de Meteoclimatic y que se encarga de apuntar al DOM de la página de dicha estación y obtener los datos objetivo mediante la técnica de web scraping y la ayuda de "Axios" y "Cheerio". Finalmente el método devuelve un array de 6 componentes que conforma un registro meteorológico diario o actual.

```
export function scrapMeteoclimatic(id) {

    Complexity is 3 Everything is cool!
    return new Promise((resolve, reject) => {

        const url = `https://www.meteoclimatic.net/perfil/${id}`;

        /*tme,tmx,tmn,prc,hum,vie*/
        const statValues = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

        axios(url).then((response) => {

            // Cargar el HTML en cheerio
            const html = response.data;
            const $ = cheerio.load(html);

            let tMed = $(".capsaactuals tr:nth-child(2) td:nth-child(1)").text();
            statValues[0] = normalizeToDecimals(tMed);

            let tMax = $(".capsaactuals span.vermell").text();
            statValues[1] = normalizeToDecimals(tMax);

            let tMin = $(".capsaactuals span.blau").text();
            statValues[2] = normalizeToDecimals(tMin);

            let prec = $(".capsaactuals tr:nth-child(2) td:nth-child(9)").text();
            statValues[3] = normalizeToDecimals(prec);

            let hum = $(".capsaactuals tr:nth-child(2) td:nth-child(3)").text();
            statValues[4] = normalizeToDecimals(hum);

            let viento = $(".capsaactuals tr:nth-child(2) td:nth-child(5)").text();
            statValues[5] = normalizeToDecimals(viento);

        }).then(() => {
            console.log("METEOCLIMATICscrap: " + statValues)
            resolve(statValues);
        })
    })
}
```

Figura 35: Lógica para el método '`scrapMeteoclimatic`'.

Anexo 3. Librerías utilizadas

Librería `@angular/fire v7.1.0-rc.5`

Instalación: **`ng add @angular/fire`**

Interfaz de comunicación con “Firestore” y el subsistema “Auth”. Es importada en el servicio ‘`authService`’, permitiendo inyectar como dependencias a “AngularFireAuth” y “AngularFirestore”. “AuthFirestore” es importado también en `estacionesService` y `setasService` para la obtención de documentos y colecciones

Los métodos de “AngularFireAuth” utilizados son:

- `createUserWithEmailAndPassword`
- `signInWithEmailAndPassword`
- `signOut`

Los métodos de “AngularFirestore” utilizados son:

- Acceso a las colecciones y documentos a través de la referencia con `valueChanges()` para la lectura de datos.
- Acceso a las colecciones y documentos a través de la referencia con `set()` para producir cambios en la base de datos o la creación de nuevas ocurrencias para una determinada entidad.

Para trabajar con Firebase se precisa del alta de un proyecto web en Firebase/Firestore en una cuenta de usuario de Google e importar dicha información como variable de entorno en `/environments/environments.ts` y `/environments/environments.prod.ts`

Librería `@angular/google-maps v13.3.3`

Instalación: **`npm i @angular/google-maps`**

Interfaz de comunicación con Google Maps Platform. Permite definir un mapa de Google Maps como un elemento HTML interno a un componente al cual le son configurados una serie de atributos. Desde el fichero Typescript de la lógica del componente donde participa el mapa se controlan opciones como el *zoom*, tipo de mapa, la habilitación de doble clic, el centrado, entre otros. En dicho fichero y a través de una llamada “*HttpClient*” se recarga la API. Adicionalmente, en el fichero de lógica se describen los métodos que controlarán los aspectos de *zoom* y centrado con los controles de botón sobre el mapa.

Para poder ofrecer un mapa de Google Maps Platform se precisa de una API KEY que debe ser obtenida tras un proceso de selección de plan de pago por el usuario. Esta API KEY deberá ser cargada en el index raíz del proyecto y cargada de nuevo mediante JSONP para aquellos módulos internos cargados de forma perezosa.

Anexo 4. Capturas de pantalla

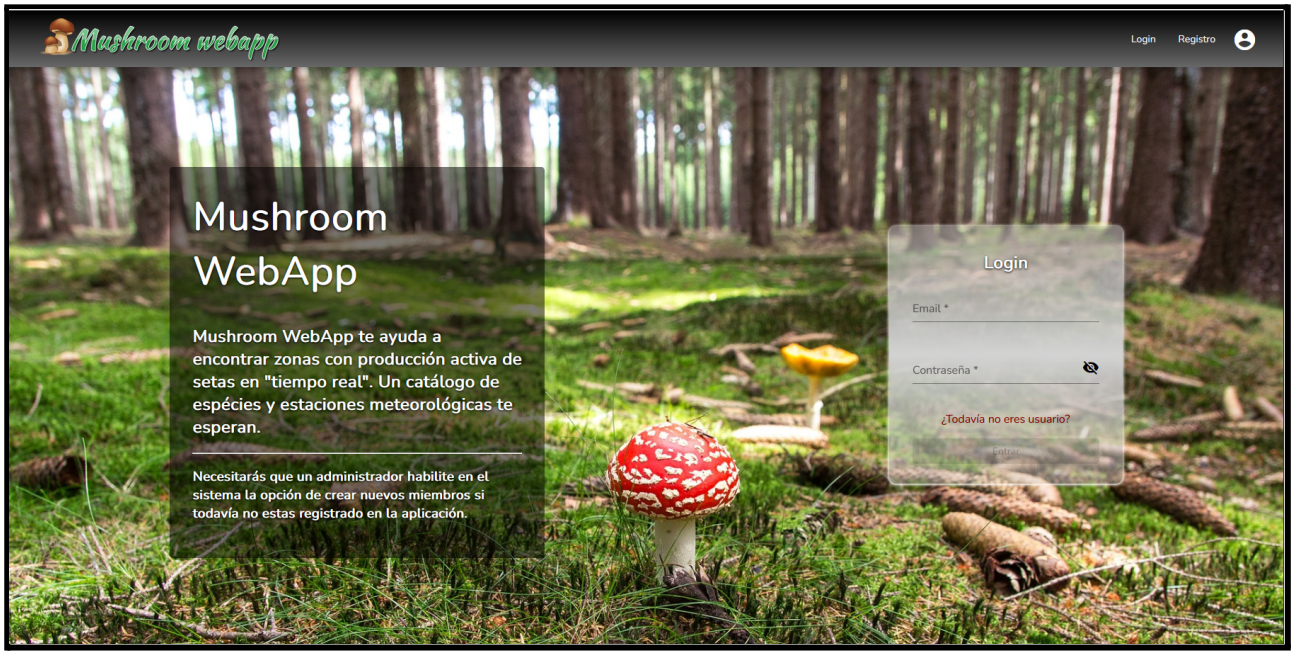


Figura 36: Página de inicio de Mushroom WebAPP.

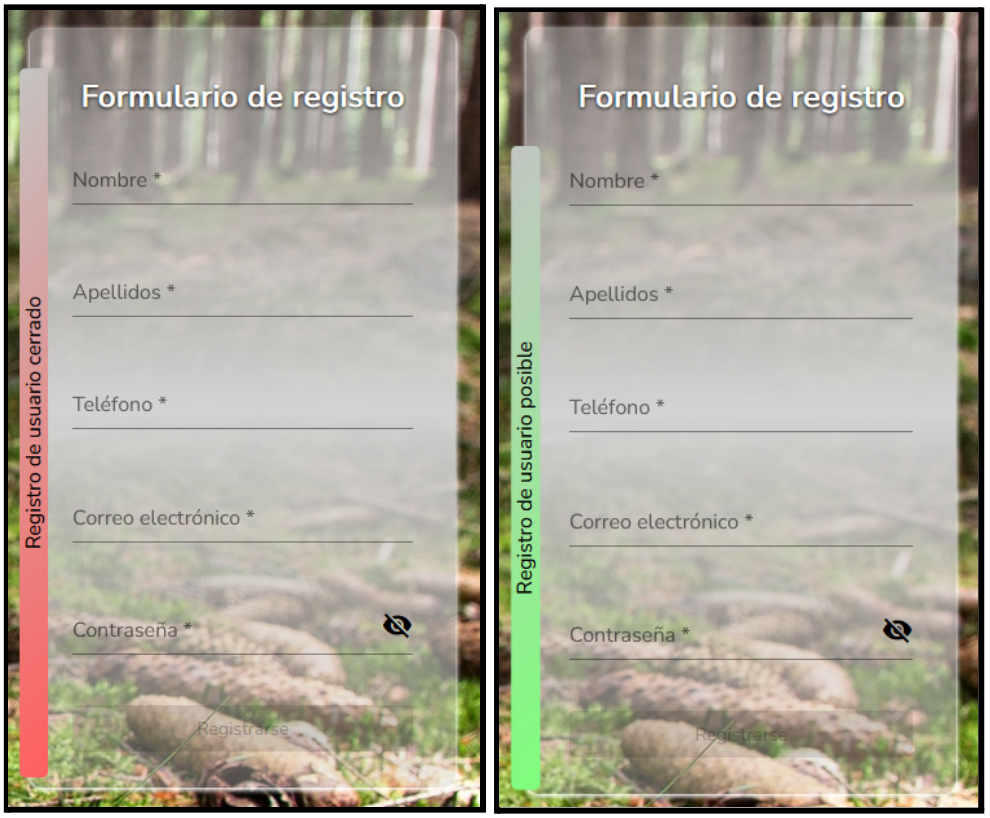


Figura 37: Formularios de registro.

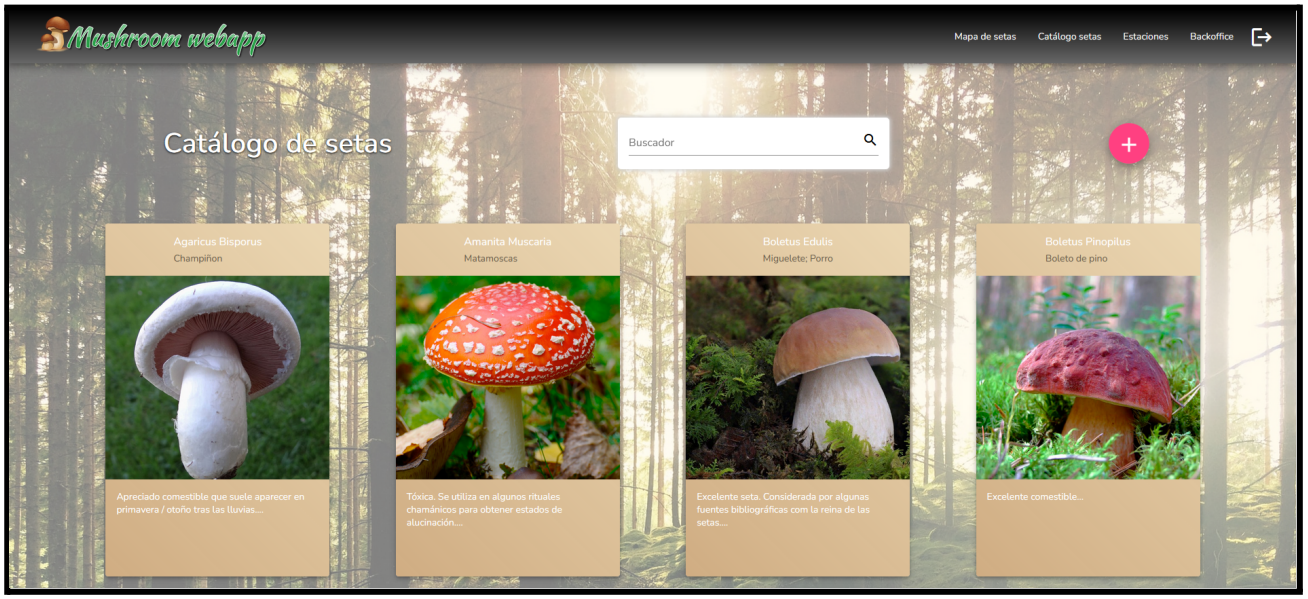


Figura 38: Catálogo de especies.

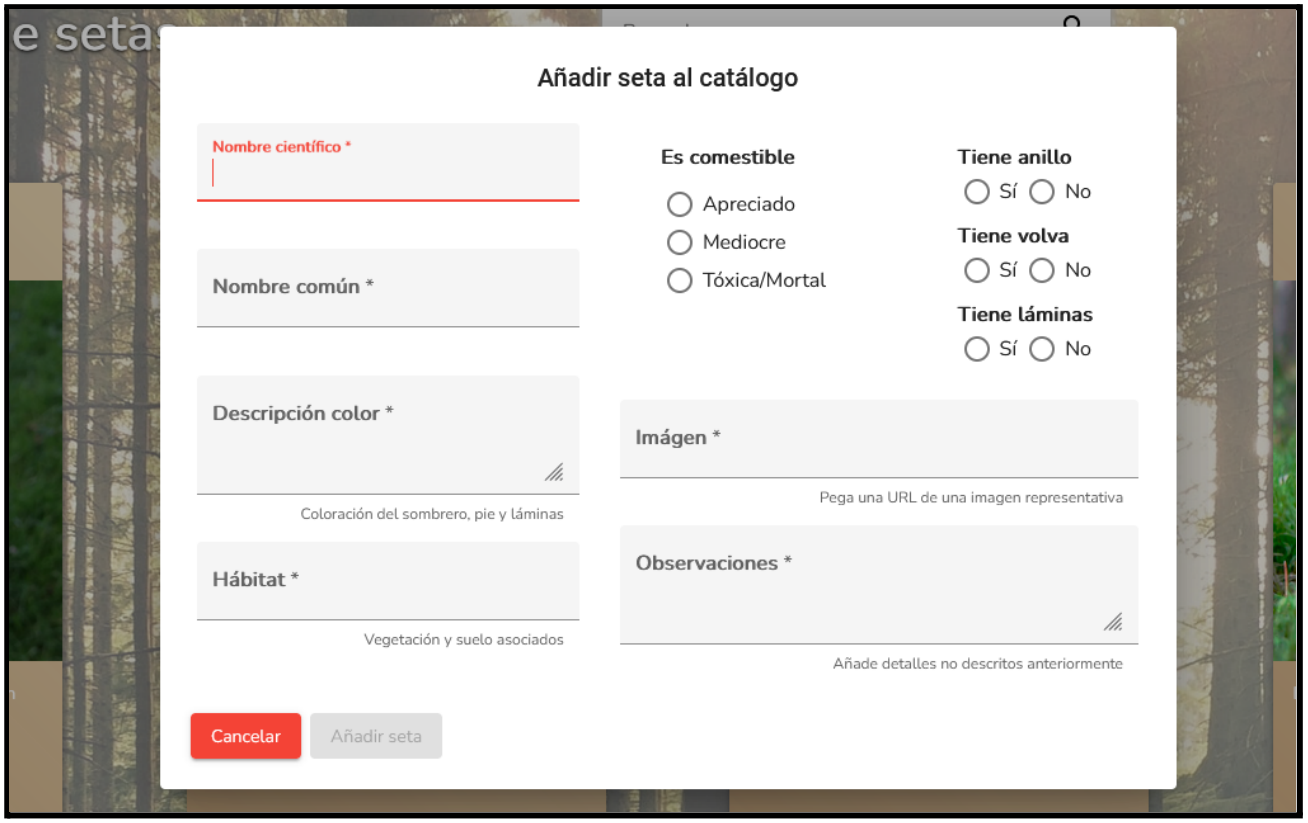


Figura 39: Modal de alta de ficha técnica de seta.

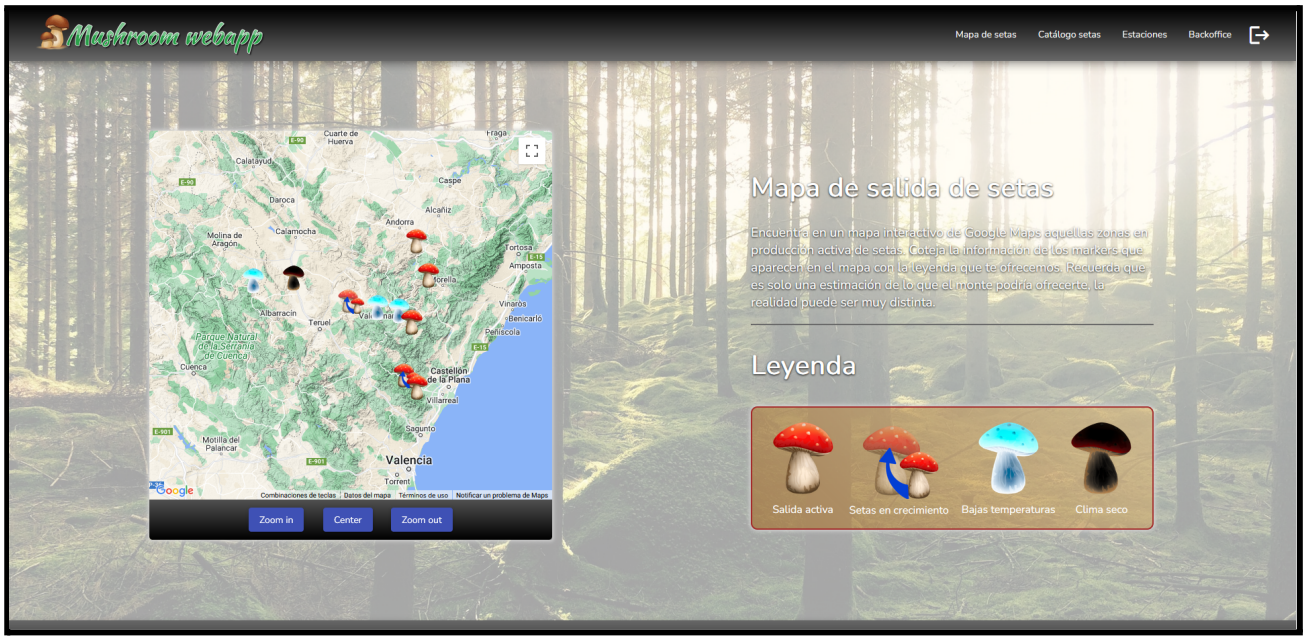


Figura 40: Mapa de predicción de salida de setas.

Figura 41: Página de detalle ampliado para una ficha técnica.

Figura 42: Listado de estaciones de control meteorológico.

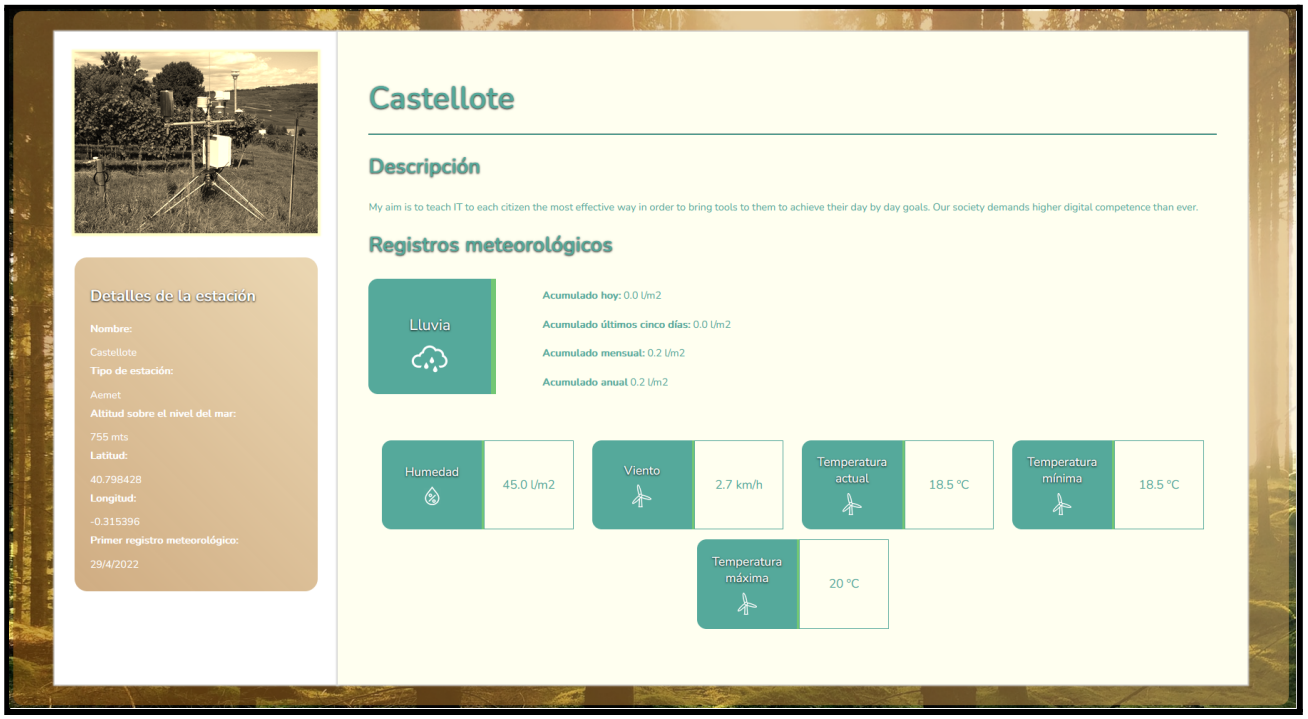


Figura 43: Página de detalle ampliado para una estación de control meteorológico.

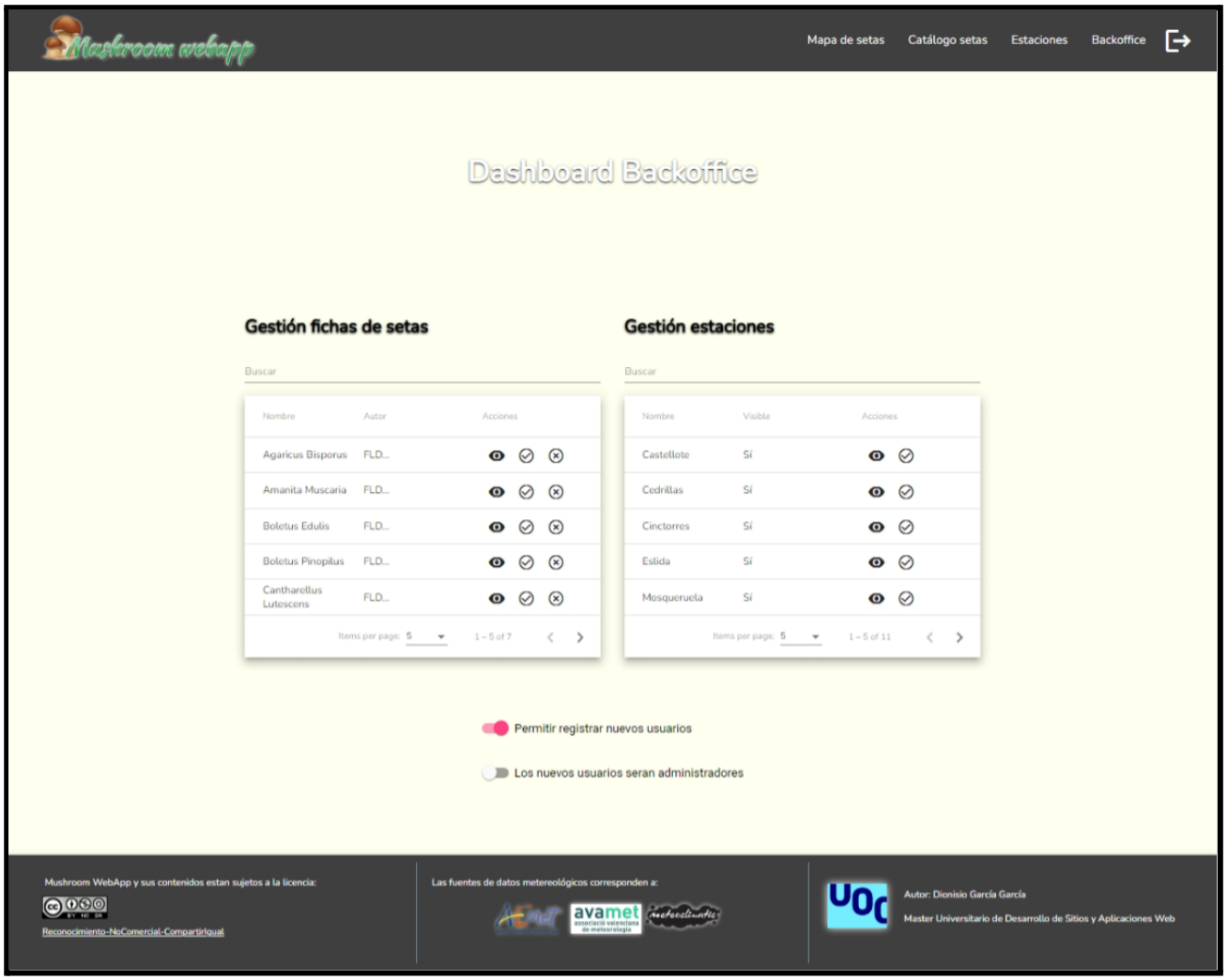


Figura 44: *Backoffice* de gestiones de administrador.

Anexo 5. Guía de usuario

La Figura 45 muestra una infografía como Guía de usuario. La misma ha sido elaborada con Canva.

Figura 45: Guia de usuario en formato cartel-infografía.

Anexo 6. Libro de estilo

Logotipo Mushroom WebAPP

Se trata de una composición por capas elaborada con GIMP mediante un ícono gráfico de una pareja de Boletus Edulis y un texto generado con la plataforma web australiana '*Flaming text*' [57]. Las condiciones de utilización de los logos generados con la plataforma están libres de costes para fines personales o académicos. Fuente: "*Kaushan Script*"; Color base: `#319e34`. La Figura 46 muestra el aspecto del logotipo.

Figura 46: Logotipo Mushroom WebApp.

Iconografía de setas en el mapa y favicon

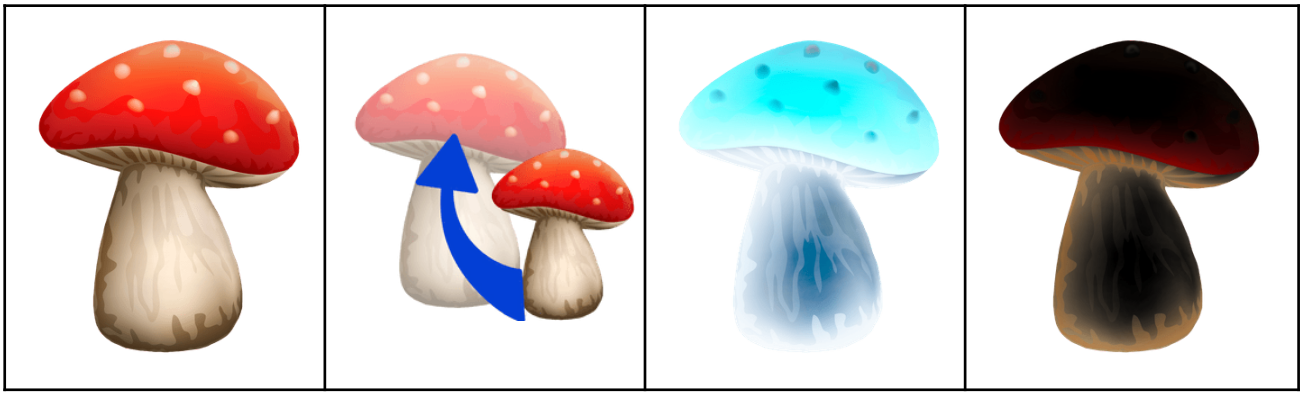


Figura 47: Iconografía de mapa.

Tal como muestra la Figura 47, se trata de tres variaciones personalizadas de la primera imágen para indicar los estados de salida de setas, crecimiento, congelación y cosecha arruinada por excesiva sequedad del terreno. Como se puede apreciar los cuatro iconos parten del primero. Las variaciones han sido obtenidas con GIMP.

El favicon que utiliza el sitio web ha sido generado a partir de la primera imagen en tamaño 32 x 32 con GIMP.

Iconografía para las estaciones meteorológicas.

La Figura 48 muestra la realización de la iconografía metereológica elaborada con Inkscape:

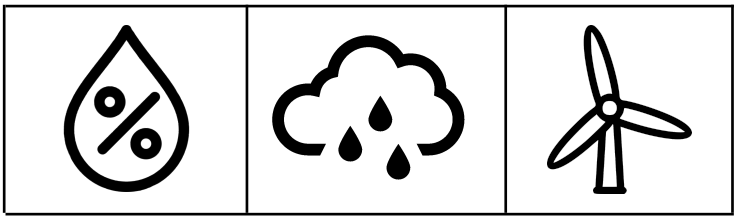


Figura 48: Iconografía para las estaciones de control meteorológico.

Imágenes de fondo utilizadas.

Fuente: “Pixabay” [58] / “Unsplash” [59]. Ver Figura 49. Son imágenes libres de derechos de autor.

Figura 49: Imágenes de fondo

Tabla 4: Paleta de colores

`$green-dark-bg: #55a99b;`	
`$green-light-bg: #72c774;`	
`$header-gradient: linear-gradient(rgb(70, 70, 70), black);`	
`$footer-gradient: linear-gradient(rgb(80, 80, 80), black);`	
`$bx-sh-lt: 0 0 5px 2px lightgrey;`	
`$terrain-color: linear-gradient(45deg, #a4611085, #dcb97991);`	
`$new-user-available: linear-gradient(45deg, #7eff7c, #c3c5c3);`	
`$no-user-available: linear-gradient(45deg, rgb(255, 94, 94), #c3c5c3);`	
`$metal-skin: linear-gradient(45deg, grey, white, grey);`	

Tabla 4: Paleta de colores para Mushroom WebApp.

Frameworks de estilos: Angular Material. Aplicado a botones, formularios de entrada de datos, diálogos, tablas de gestión en la parte *backoffice*, iconografía, etc. La Figura 50 realiza una muestra de Material.

Se requiere de la instalación: `ng add @angular/material`.

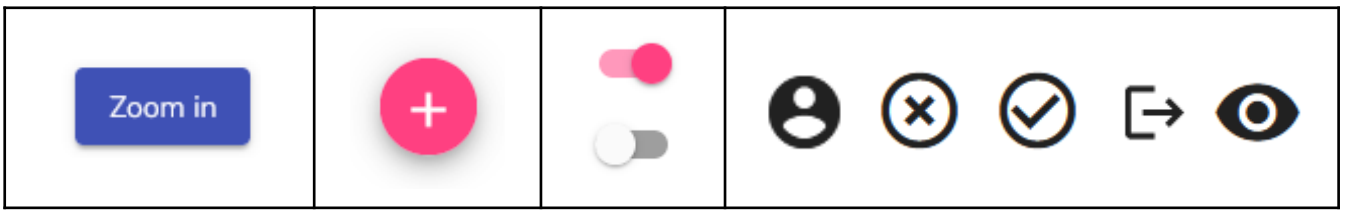


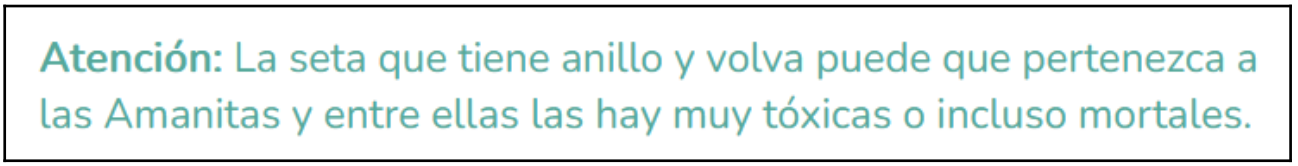
Figura 50: Botonería de Angular Material.

Fuente principal para texto y titulares: “Nunito” (Importación desde Google Fonts)

Todas las medidas de tamaño de fuente están expresadas en múltiplos/divisores de unidades REM (relativas al tamaño de fuente de root o elemento raíz).

Los titulares aplican tamaños de fuente superiores y sombreado de texto.

Extracto de texto embebido en un elemento párrafo de la página de detalle de una ficha técnica.



Arquitectura SCSS

La arquitectura de hojas de estilos que siga la aplicación se basa en:

- Un fichero general de estilos que afecta a toda la aplicación: `styles.scss`
- Un fichero de variables globales SASS: `_variables.scss`
- Una hoja de estilo por cada componente de la aplicación con los estilos únicos como si de una entidad software aislada se tratase.

Algunas imágenes han sido reducidas en tamaño para aligerar los tiempos de carga de la aplicación en el navegador utilizando la plataforma ‘I love img’ [60].

Anexo 7. Resumen ejecutivo

A continuación se presenta en la Tabla 5 un resumen ejecutivo para el producto Mushroom WebApp.

Nombre comercial		Mushroom WebApp.
Resumen comercial		Aplicación web que proporciona información sobre la posibilidad de aparición de setas en un conjunto de lugares del interior de Castellón y provincia de Teruel. Del mismo modo proporciona información meteorológica de los lugares y de formación mediante un catálogo de especies de setas.
Modelo de negocio		Prestar servicio a una comunidad foral que desea dar a conocer sus parajes a turistas de paso tomando como reclamo las temporadas de búsqueda de setas y el grueso de visitantes aficionados a la micología.
Productos y servicios		Habilitación de un portal web de infomación micológica.
Mercado		Administraciones públicas y comunidades forales o asociaciones privadas que quieran realizar gestiones de aprovechamiento micológico o usar Mushroom WebApp como medio formativo.
Competencia		Empresas tecnológicas especializadas en desarrollos de sistemas de gestión inteligente forestal o de optimización micológica. Ej: ECM Ingeniería Medio Ambiental.
Plan de marketing		Publicidad en los medios de difusión oficiales como páginas web y redes sociales de ayuntamientos y ministerios de medioambiente. Campañas publicitarias basadas en cartelería y promoción en emisoras de radio local.
Inversión inicial y costes a corto y medio plazos		Desarrollo de Mushroom WebApp: **7920€.** Alojamiento en un servidor (primer año): **381€.** Campaña de promoción (primer año): **3000€.**
Proyección económica corto y medio plazos y ROI		Se prevé un crecimiento económico de las zonas participantes de un 10-15% en el sector servicios (hostelería y comercio local) durante la temporada activa de recolección de setas.
DAFO	**Debilidades**	La climatología poco propicia para la salida de setas (ausencia de lluvia). La aplicación marcará que no se esperan salidas y por tanto no habrá reclamo para los turistas micólogos.
	Amenazas	Competidores como ECM Ingeniería Medio Ambiental pueden poner en funcionamiento una aplicación de similares características.
	Fortalezas	No existen hasta la fecha aplicaciones ofertadas al público con similares características.
	Oportunidades	Enfocada desde la perspectiva de la concienciación de ecosistemas puede ser un contribuyente a la conservación de la zona.

Tabla 5: Resumen ejecutivo para Mushroom WebAPP.

Anexo 8. Glosario

Micología: ciencia biológica que se dedica al estudio de los hongos.

Carpóforo: cuerpo fructífero de los hongos superiores, ascomicetos, basidiomicetos y gasteromicetos.

Renderizado: término usado en computación para referirse al proceso de generar gráficos desde un modelo.

Framework: conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Assets: todos los recursos referenciados desde el código HTML de las páginas como son las hojas de estilo, *scripts*, imágenes, etc.

Script: secuencia de comandos o guion es un término informal que se usa para designar a un programa relativamente simple.

SVG: formato de gráficos vectoriales bidimensionales, tanto estáticos como animados, en formato de lenguaje de marcado.

Webp: formato gráfico en forma de contenedor, que surge de la mano de Google y que permite tanto la compresión con pérdida como sin ella.

SASS (Syntactically Awesome Style Sheets): lenguaje de hoja de estilos en cascada que otorga una capa superior y funcionalidades propias de los lenguajes de programación al lenguaje de estilos CSS. Requiere de la compilación de código en CSS para que este sea interpretado por el navegador.

SCSS: Extensión de ficheros SASS que se caracteriza por el uso de brackets en vez de la indentación de código.

Backoffice: parte privada que sirve para administrar la aplicación.

SEO (Search Engine Optimization): conjunto de estrategias y técnicas de optimización que se hacen en una página web para que aparezca orgánicamente en los motores de búsqueda de los navegadores web.

Anexo 9. Bibliografía

[1] Guadalajara diario. Las setas a través de la historia. https://www.guadalajaradiario.es/ocio-y-cultura/19733-las-setas-a-traves-de-la-historia.html [Consulta: 29 de marzo de 2022]

[2] Palazón Lozano, F. (2006). Setas para todos. Editorial Pirineo.

[3] Garmin. https://www.garmin.com/es-MX/p/14903 [Consulta: 5 de junio de 2022]

[4] Aemet. http://www.aemet.es/es/portada [Consulta: 29 de marzo de 2022]

[5] Avamet. https://www.avamet.org/ [Consulta: 29 de marzo de 2022]

[6] Meteoclimatic https://www.meteoclimatic.net/ [Consulta: 29 de marzo de 2022]

[7] Open Data AEMET. https://opendata.aemet.es/ [Consulta: 29 de marzo de 2022]

[8] Cloud Firestore. https://firebase.google.com/docs/firestore [Consulta: 5 de junio de 2022]

[9] SASS https://sass-lang.com/ [Consulta: 5 de junio de 2022]

[10] Angular Material https://material.angular.io/ [Consulta: 5 de junio de 2022]

[11] Node.js https://nodejs.org/es/ [Consulta: 5 de junio de 2022]

[12] Axios https://axios-http.com/docs/intro [Consulta: 5 de junio de 2022]

[13] Cheerio https://cheerio.js.org/ [Consulta: 5 de junio de 2022]

[14] Caçadors de bolets. TV3. https://www.ccma.cat/tv3/cacadors-de-bolets/ [Consulta: 29 de marzo de 2022]

[15] Google Playstore https://play.google.com/store/apps?hl=es [Consulta: 5 de junio de 2022]

[16] Cotos de setas. Ictiotech. https://cotosdesetas.es/ [Consulta: 29 de marzo de 2022]

[17] Angular CLI. https://angular.io/cli [Consulta: 5 de junio de 2022]

[18] Scrum. Atlassian. https://www.atlassian.com/es/agile/scrum [Consulta: 5 de junio de 2022]

[19] R.S. Pressman & B.R. Maxim. Software Engineering: A practitioner's approach. Irwin Computer Science. 8th Edition.

[20] Single page application: definición, funcionamiento y utilidad. IONOS. https://www.ionos.es/digitalguide/paginas-web/creacion-de-paginas-web/single-page-application/ [Consulta: 29 de Marzo de 2022]

[21] @angular-fire. Angular. https://www.npmjs.com/package/@angular/fire [Consulta: 29 de Marzo de 2022]

[22] Firebase Auth. Google. https://firebase.google.com/docs/auth [Consulta: 29 de Marzo de 2022]

[23] Firebase. Google. https://firebase.google.com/ [Consulta: 29 de Marzo de 2022]

[24] Visual Studio Code. https://code.visualstudio.com/ [Consulta: 5 de junio de 2022]

[25] Codemetrics. https://marketplace.visualstudio.com/items?itemName=kisstkondoros.vscode-codemetrics [Consulta: 5 de junio de 2022]

[25] Prettier. https://prettier.io/ [Consulta: 5 de junio de 2022]

[26] Git https://git-scm.com/ [Consulta: 5 de junio de 2022]

[27] Github https://github.com/ [Consulta: 5 de junio de 2022]

[28] Google Maps Platform. Google. https://developers.google.com/maps [Consulta: 29 de Marzo de 2022]

[29] Inkscape. https://inkscape.org/es/ [Consulta: 5 de junio de 2022]

[30] GIMP. http://www.gimp.org.es/descargar-gimp.html [Consulta: 5 de junio de 2022]

[31] Apache. https://httpd.apache.org/ [Consulta: 5 de junio de 2022]

[32] Nginx https://www.nginx.com/ [Consulta: 5 de junio de 2022]

[33] Ganttproject. https://www.ganttproject.biz/ [Consulta: 29 de Marzo de 2022]

[34] Mapbox. https://www.mapbox.com/ [Consulta: 5 de junio de 2022]
[35] Bing. https://www.bing.com/maps/ [Consulta: 5 de junio de 2022]
[36] Here. https://www.here.com/ [Consulta: 5 de junio de 2022]
[37] Swagger. https://swagger.io/ [Consulta: 5 de junio de 2022]
[38] Diagrams.net. https://www.diagrams.net/ [Consulta: 29 de Marzo de 2022]
[39] Quicktype. https://quicktype.io/ [Consulta: 5 de junio de 2022]
[40] Vflat. VoyagerX. https://play.google.com/store/apps/details?id=com.voyagerx.scanner&hl=en&gl=US
[41] Figma. https://www.figma.com/ [Consulta: 29 de Marzo de 2022]
[42] Mostly Fluid. AnalyticaWeb
https://www.analyticaweb.com/desarrollo-web/implementando-tecnicas-responsive-mostly-fluid [Consulta: 5 de junio de 2022]
[43] Welie. http://www.welie.com/ [Consulta: 5 de junio de 2022]
[44] Cloudtime https://www.eduappcenter.com/apps/125 [Consulta: 5 de junio de 2022]
[45] OpenSSL https://www.openssl.org/ [Consulta: 5 de junio de 2022]
[46] TDD. Wikipedia https://es.wikipedia.org/wiki/Desarrollo_guiado_por_pruebas [Consulta: 5 de junio de 2022]
[47] Jasmine https://jasmine.github.io/ [Consulta: 5 de junio de 2022]
[48] Karma https://angular.io/guide/testing [Consulta: 5 de junio de 2022]
[49] Netlify https://www.netlify.com/ [Consulta: 5 de junio de 2022]
[50] Librería Angular Google Maps (AGM) https://angular-maps.com/ [Consulta: 4 de Juinio de 2022]
[51] Putty https://www.putty.org/ [Consulta: 5 de junio de 2022]
[52] Mozilla Firefox. https://www.mozilla.org/es-ES/firefox/new/ [Consulta: 5 de junio de 2022]
[53] Microsoft Edge. https://www.microsoft.com/es-es/edge [Consulta: 5 de junio de 2022]
[54] Raspberry PI. https://www.raspberrypi.org/ [Consulta: 5 de junio de 2022]
[55] Axarnet https://axarnet.es/ [Consulta: 5 de junio de 2022]
[56] FungiGO. https://www.fungigo.es/ [Consulta: 8 de Mayo de 2022]
[57] ECM Ingeniería Medioambiental. https://www.ecmingenieriaambiental.com/ [Consulta: 8 de Mayo de 2022]
[58] Flaming text. https://flamingtext.es/ [Consulta: 8 de Mayo de 2022]
[59] Pixabay. https://pixabay.com/es/ [Consulta: 8 de Mayo de 2022]
[60] Unsplash. https://unsplash.com/es [Consulta: 8 de Mayo de 2022]
[61] I love img. https://www.iloveimg.com/es [Consulta: 8 de Mayo de 2022]