Diseño e implementación de una aplicación web para la gestión de anillamiento de aves

Memoria de Proyecto Final de Máster

Máster Universitario en Aplicaciones Multimedia

Itinerario Profesionalizador

Autor: Manuel Pérez Alfonso

Consultor: Sergio Schvarstein Liuboschetz

Profesores: David García Solórzano, Laura Porta Simó

12 de Junio de 2017

Créditos/Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Diseño e implementación de una aplicación web para la gestión de anillamiento de aves*
Nombre del autor:	*Manuel Pérez Alfonso*
Nombre del consultor/a:	*Sergio Schvarstein Liuboschetz*
Nombre del PRA:	*David García Solórzano, Laura Porta Simó*
Fecha de entrega (mm/aaaa):	06/2017
Titulación:	*Máster Universitario en Aplicaciones Multimedia*
Área del Trabajo Final:	*Trabajo Fin de Máster*
Idioma del trabajo:	*Español*
Palabras clave	*Aplicación web, gestión, anillamiento, aves, ornitología, móviles.*

Resumen del Trabajo:

Este TFM pretende ser una primera aproximación a una aplicación web dedicada al anillamiento de aves y dirigida a anilladores profesionales. Aunque incompleta, la aplicación web desarrollada contiene los principales elementos de que constaría su versión final, y hace especial incidencia en la integración del back-end y el front-end en determinados núcleos de funcionalidad, también reutilizables en futuros proyectos. Se trata, dicho de otro modo, de un ejercicio de integración de las tecnologías Django (back-end) y Angular 4 (front-end) de forma que los servicios web desarrollados puedan ser utilizados en un futuro por aplicaciones nativas de las distintas plataformas móviles.

También se ha realizado una importante labor de despliegue de las distintas partes de que consta el sistema, así como de las medidas de seguridad correspondientes para la correcta configuración de la restricción CORS, o la encriptación correcta de los datos intercambiados con el servidor, entre otras tareas de despliegue.

Todo el trabajo realizado ha puesto de manifiesto la gran complejidad que supone el desarrollo de un sistema de este calibre, en el cual el front-end es la parte visible desde el punto de vista del usuario. De todos modos, la línea metodológica propuesta en este TFM permite apreciar la complejidad de todos los pasos dados, desde el diseño a alto nivel del producto final mediante mock-ups, pasando después al diseño del back-end y los servicios web, para finalizar con el desarrollo del front-end.

Abstract:

This project aims to be a first approach to a bird ringing web application used by professional banders. Although unfinished, the developed web application contains all the major elements that a final version would have, and focuses especially on the the back-end and front-end integration of certain core areas, which could potentially be used in future projects. It is, in other words, an integration exercise of different technologies, Django (back-end) and Angular 4 (front-end), in such a way that the implemented web services could be used by potential mobile applications in different mobile platforms.

Also, this project implies a great effort in terms of the deployment of the different system elements, including all security measures to implement the CORS restriction, or even the correct encryption of the data exchanged between the different parties, among other tasks.

The development process highlights the complexity of the system behind the front-end element (which is the only visible element from a user's perspective), taking as a start point a hight-level design of the application using mock-ups, defining and implementing the back-end functionalities and web services as a second step, and finally implementing the front-end elements which will interact with the user and the back-end.

Dedicatoria

Dedico este TFM a mi madre Encarnación, a mi mujer Sònia y a mi hijo Manuel.

Agradecimientos

A Jesús Alfonso i Prieto, por proporcionarme la idea inicial de este TFM.

A Sergio Schvarstein Liuboschetz, por su guía en este TFM, paciencia y feedback sin falta de detalles.

Abstract

This project aims to be a first approach to a bird ringing web application used by professional banders. Although unfinished, the developed web application contains all the major elements that a final version would have, and focuses especially on the the back-end and front-end integration of certain functionality areas, which could potentially be used in future projects. It is, in other words, an integration exercise of different technologies, Django (back-end) and Angular 4 (front-end), in such a way that the implemented web services could be used by potential mobile applications in different mobile platforms.

Also, this project implies a great effort in terms of the deployment of the different system elements, including all security measures to implement the CORS restriction, or even the correct encryption of the data exchanged between the different parties, among other tasks.

The development process highlights the complexity of the system behind the front-end element (which is the only visible element from a user's perspective), taking as a start point a hight-level design of the application using mock-ups, defining and implementing the back-end functionalities and web services as a second step, and finally implementing the front-end elements which will interact with the user and the back-end.

Resumen

Este TFM pretende ser una primera aproximación a una aplicación web dedicada al anillamiento de aves y dirigida a anilladores profesionales. Aunque incompleta, la aplicación web desarrollada contiene los principales elementos de que constaría su versión final, y hace especial incidencia en la integración del back-end y el front-end en determinados núcleos de funcionalidad, también reutilizables en futuros proyectos. Se trata, dicho de otro modo, de un ejercicio de integración de las tecnologías Django (back-end) y Angular 4 (front-end) de forma que los servicios web desarrollados puedan ser utilizados en un futuro por aplicaciones nativas de las distintas plataformas móviles.

También se ha realizado una importante labor de despliegue de las distintas partes de que consta el sistema, así como de las medidas de seguridad correspondientes para la correcta configuración de la restricción CORS, o la encriptación correcta de los datos intercambiados con el servidor, entre otras tareas de despliegue.

Todo el trabajo realizado ha puesto de manifiesto la gran complejidad que supone el desarrollo de un sistema de este calibre, en el cual el front-end es la parte visible desde el punto de vista del usuario. De todos modos, la línea metodológica propuesta en este TFM permite apreciar la complejidad de todos los pasos dados, desde el diseño a alto nivel del producto final mediante mock-ups, pasando después al diseño del back-end y los servicios web, para finalizar con el desarrollo del front-end.

Palabras clave

Aplicación web, gestión, anillamiento, aves, ornitología, móviles.

Notaciones y Convenciones

Todas las referencias a comandos o fragmentos de código se han diferenciado del resto del texto utilizando el color azul.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

1. Introducción

El estudio del comportamiento (entre otras características) de las aves se sitúa en el foco de interés de multitud de personas con diferentes perfiles, agrupados por diferentes intereses y motivaciones, base de conocimientos y edades. Las personas interesadas en el mundo de la ornitología pueden transitar de un perfil a otro dependiendo del grado de implicación y conocimiento que adquieran, y los estadios de mayor conocimiento de este ámbito se fundamentan en gran medida en la acción del voluntariado. Este voluntariado hace posible la movilización de los recursos humanos y materiales necesarios para llevar a cabo el proceso de anillamiento de aves, documentación, integración de datos y su posterior análisis. Se trata, éste último, del perfil correspondiente a personas formadas y con las aptitudes necesarias para realizar una toma precisa de datos y, al mismo tiempo, no suponer una acción perjudicial para las aves capturadas.

Toda esta actividad de anillamiento de aves genera un gran volumen de datos que muchas veces se encuentra disperso en diferentes herramientas ofimáticas, sujeto a errores de introducción manual e inconsistencias. Y es por ello que las nuevas tecnologías de desarrollo de aplicaciones, ya sean aplicaciones web, nativas o de escritorio, pueden ayudar a una recopilación, consolidación y procesamiento más efectivo de todos estos datos.

Por tanto este TFM trata de conectar el mundo de la ornitología con el de las tecnologías móviles, y que los dispositivos móviles pasen a ser la nueva libreta de anotación de datos en las labores de campo de estos profesionales y aficionados.

2. Descripción

Este proyecto nace, en una primera fase, ante la necesidad de una mejor gestión de los datos de anillamiento, derivada de la experiencia de personas aficionadas al mundo de la ornitología. El presente proyecto pretende definir los requisitos tanto funcionales como técnicos, así como una primera implementación, de una aplicación web que facilitará la introducción y gestión de datos de diferentes sesiones de anillamiento de aves.

En el aspecto técnico de la propuesta, se pretende realizar el diseño gráfico de la aplicación, así como realizar una primera implementación centrada en dispositivos móviles, priorizando en un segundo lugar el diseño para la versión escritorio.

En posteriores fases, y fuera del alcance del presente TFM, se pretende dotar a la herramienta de un carácter más cooperativo, con una explotación más inteligente de los datos recopilados y, en una fase más alejada, la introducción de algoritmos de AI para la detección de determinadas características biológicas a partir de muestras visuales.

3. Objetivos generales

El objetivo general de este TFM es poder realizar una primera versión de aplicación completa, que integre la mayoría de los aspectos trabajados durante este Máster, y producir finalmente un producto que sea funcional y esté disponible en un servidor remoto.

Este objetivo se puede desglosar desde los siguientes puntos de vista:

Objetivos de la aplicación:

- Implementación modular extensible a futuras aplicaciones móviles
- Comunicación cifrada entre los diferentes componentes de la arquitectura
- Adaptación a las nuevas tendencias de aplicaciones web reactivas

Objetivos para el usuario:

- Automatización en las labores de introducción de datos de anillamiento
- Evitar incidencias de duplicidad y pérdida de datos

Objetivos personales del autor del TF:

- Implementación de un sistema base a partir del cual iniciar futuros proyectos
- Evaluación de las últimas tendencias en la programación de aplicaciones web

4. Metodología y proceso de trabajo

El proceso parte del análisis de una de las aplicaciones para ornitólogos existente, NouBioPro *[16]*, la cual ya implementa muchas de las funcionalidades requeridas en el proceso de anillamiento, pero que sólo está disponible en una versión instalable en un entorno local, no accesible de forma remota y centralizada.

A continuación se han analizado diferentes tecnologías que posibiliten una implementación modular y flexible, y que permitan el desarrollo de aplicaciones móviles en futuras fases del proyecto con el menor impacto posible, tras lo cual se ha decidido utilizar las tecnologías Django (para la implementación del back-end) y Angular 4 (para el front-end). Esta elección se justifica por la familiaridad previa del autor del presente TFM en estas tecnologías.

Con el fin de poder realizar una mejor planificación de las tareas a realizar durante las distintas fases, se ha partido de una primer labor de diseño de mock-ups para poder tener una mejor visibilidad sobre todos los objetos a desarrollar, tanto en el front-end como el back-end.

A partir de este punto, el proyecto se ha dividido fundamentalmente en tres fases:

- Desarrollo del back-end
- Desarrollo del front-end
- Despliegue

Estas tres frases se han desgranado en subfases o hitos, que se han planificado a lo largo del tiempo en forma de cascada, según se mostrará en el apartado de planificación del proyecto.

Debido a este tipo de planificación en cascada, cualquier imprevisto detectado ha requerido la replanificación de las demás tareas afectadas.

5. Planificación

En los siguientes subapartados se muestra la planificación del proyecto según la tabla de hitos y también según el diagrama de Gantt.
La realización de los mock-ups para el diseño de la interfaz de la aplicación web, en su versión móvil, se ha planificado en primer lugar con un doble propósito:

- Es más sencillo identificar los web services que va a ser necesario implementar en el back-end, para que estén disponibles en los distintos componentes del front-end.
- Los mock-ups se pueden enviar a potenciales usuarios para recoger feedback, mientras, en paralelo, se avanza en las siguientes tareas.

5.1. Tabla de hitos

BE: Back-end
FE: Front-end

Nombre	Duración	Inicio	Final
PEC 1 – Propuesta formal del proyecto	15d	27/02/2017	13/03/2017
PEC 2 – Mandato del proyecto y planificación	14d	14/03/2017	27/03/2017
PEC 3 - Entrega 1	28d	28/03/2017	24/04/2017
1. Diseño de mock-ups	12d	28/03/2017	08/04/2017
2. BE – Diseño y programación de modelos	6d	09/04/2017	14/04/2017
3. BE – Config. y programación de REST API	9d	15/04/2017	22/04/2017
4. BE – Configuración y pruebas Oauth2	2d	23/04/2017	24/04/2017
PEC 4 - Entrega 2	28d	25/04/2017	22/05/2017
5. FE – Alta de usuarios y autenticación	11d	25/04/2017	05/05/2017
6. FE – Módulo de administración	10d	06/05/2017	15/05/2017
7. FE – HTML de los componentes restantes	3d	16/05/2017	18/05/2017
8. Configuración y despliegue en el servidor VPS	4d	19/05/2017	22/05/2017
PEC 5 - Cierre	21d	23/05/2017	12/06/2017
9. Mejora de la usabilidad de la aplicación y resolución de incidencias detectadas	7d	23/05/2017	29/05/2017
10. Memoria del proyecto	6d	30/05/2017	04/06/2017
11. Presentación del producto	5d	05/06/2017	09/06/2017
12. Presentación pública	3d	10/06/2017	12/06/2017
Defensa del proyecto	18d	13/06/2017	30/06/2017

Tabla 1: Tabla de hitos

5.2. Diagrama de Gantt

El siguiente diagrama se ha realizado con el programa Planner disponible en Ubuntu:

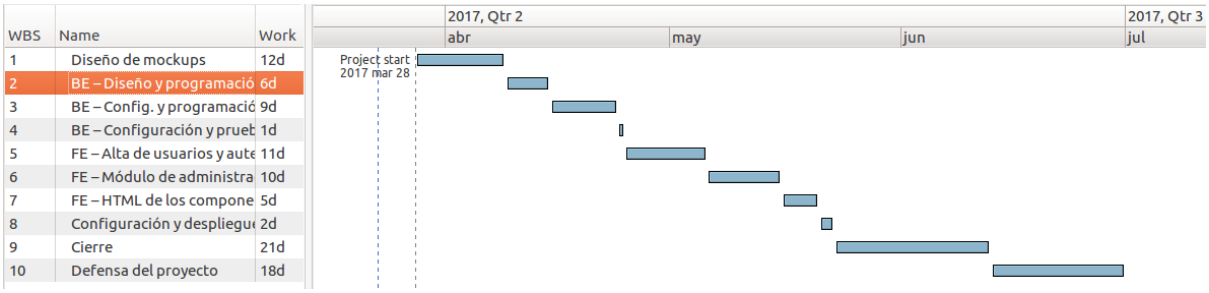


Figura 1: Diagrama de Gantt

6. Presupuesto

En la siguiente tabla se especifica el desglose por jornadas necesarias para realizar la parte de la aplicación correspondiente a este TFM, junto con el coste por hora de cada perfil y el presupuesto total. El coste por hora de cada perfil es el siguiente:

- Diseñador: 30€
- Analista: 30€
- Programador: 25€
- Administrador: 30€

Y se asume una jornada de 8 horas, y un número de 5 días laborables por semana.

Nombre	Duración	Perfil	Coste
1. Propuesta del proyecto, mandato y planificación	15d	Analista	3600
2. Diseño de mock-ups	9d	Diseñador	2160
3. BE – Diseño y programación de modelos	5d	Programador	1000
4. BE – Config. y programación de REST API	7d	Programador	1400
5. BE – Configuración y pruebas Oauth2	2d	Programador	400
6. FE – Alta de usuarios y autenticación	7d	Programador	1400
7. FE – Módulo de administración	6d	Programador	1200
8. FE – HTML de los componentes restantes	3d	Programador	600
9. Configuración y despliegue en el servidor VPS	3d	Administrador	600
		Total	**12.360,00 €**

Tabla 2: Presupuesto

7. Estructura del resto del documento

Capítulo 2: Contiene el estado del arte, donde se realiza una revisión de las aplicaciones actuales (que cubren en mayor o menor medida las funcionalidades que se pretende implementar con este TFM) y se llega a una serie de conclusiones que fundamentan el trabajo a realizar. También se definen los objetivos específicos que se pretende alcanzar.

Capítulo 3: En este apartado se resume la arquitectura del sistema que hospeda la aplicación, y se establecen las bases de diseño para la parte back-end y fron-end del sistema.

Capítulo 4: Detalla las líneas generales de desarrollo del back-end y fron-end, así como todo el proceso de despliegue y configuración en el servidor VPS para lograr una aplicación web operativa, con las medidas de seguridad correspondientes.

Capítulo 5: En este capítulo se muestran los diagramas de navegación con capturas de las pantallas programadas.

Capítulo 6: Conclusiones a las que se ha llegado tras la realización de este TFM y líneas de trabajo para futuras fases del proyecto.

Capítulo 2: Análisis

1. Estado del arte

El proceso de anillamiento en Europa se gestiona a través de las llamadas Oficinas de Anillamiento, que se ocupan de proporcionar sus propias anillas (las cuales llevan asociado un remite) y gestionar los datos que se obtienen a partir del anillamiento. En España existen dos oficinas de anillamiento: la de la Sociedad de Ciencias Aranzadi, y la del Ministerio de Medioambiente. A su vez, el organismo EURING se encarga de coordinar el anillamiento y sus oficinas a nivel europeo *[2]*.
Aun siendo un hecho ajeno al gran público, el proceso de anillamiento también retroalimenta en muchos sentidos la divulgación general sobre el conocimiento de las aves, lo cual se manifiesta también en diferentes tipologías de soluciones informáticas, que, como se detalla a continuación, abarcan desde las aplicaciones puramente divulgativas y educativas, hasta las orientadas al trabajo de campo del personal que participa en las sesiones de anillamiento.

1.1. Aplicaciones divulgativas

Por una parte, se manifiesta un perfil que consume contenido multimedia relacionado con aspectos divulgativos del mundo de las aves. Como casos ilustrativos de esta tipología, se han estudiado las siguientes aplicaciones:

Merlin Bird ID

Se trata de una app disponible en Google Play *[15]* y en Apple Store, con la mayor popularidad de las analizadas (entre 100.000 y 500.000 instalaciones en el momento de la realización de este TFM), y creado para observadores de aves principiantes e intermedios.
Principales características

- Permite una identificación de aves interactuando de forma sencilla con el usuario.
- Es posible descargar paquetes específicos de contenidos según regiones.
- Identificación aproximada a partir de imágenes.

Desventajas

- Se trata de una app que cubre 650 especies más comunes, pero sólo de Estados Unidos y Canadá.
- La identificación de especies es aproximada y automática, sin apoyo de expertos que ayuden con las identificaciones.
- No permite un enfoque de trabajo colaborativo.
- No cubre las necesidades de los anilladores.

A continuación se pueden ver algunas de las pantallas de que consta esta app:

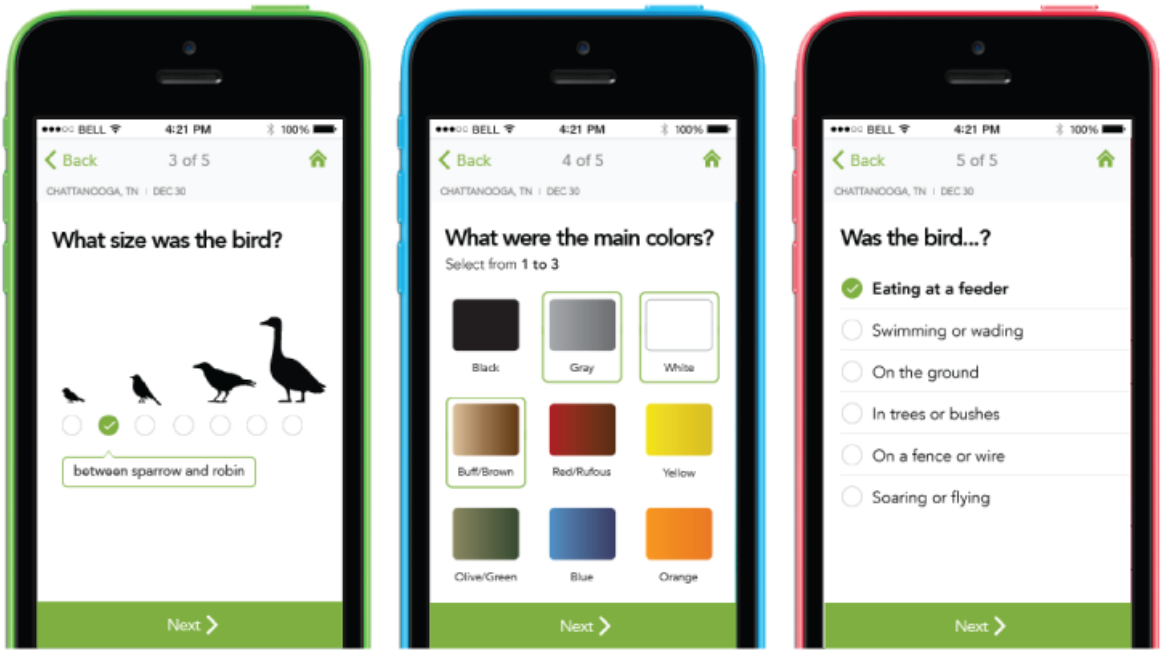


Figura 2: Merlin Bird ID

Aves de España

Esta app, disponible en Google Play *[10]* y Apple Store, ha sido desarrollada por la organización SEO BirdLife y por la Fundación BBVA, y proporciona una amplia cantidad de contenido multimedia entorno a las especies de aves en España.

Principales características

- Enfoque divulgativo tanto para el gran público y para especialistas
- Basado en la Enciclopedia de las aves de España
- Disponibilidad de contenido offline

Desventajas

- No existe mecanismo de identificación automática.
- No permite retroalimentación de las observaciones de los usuarios.
- No cubre las necesidades de los anilladores.
- La app no se ha actualizado desde el año 2015.

A continuación se muestran algunas de las pantallas de que consta esta app:

Figura 3: Aves de España

BirdPaper

Esta app, también de SEO BirdLife, está disponible sólo en Google Play *[11]*, e instala un salvapantallas en el teléfono móvil que reproduce las principales especies de la península ibérica. Se creó como un medio de recaudación de fondos (es la única app que no es gratuita), pero ha tenido una escasa repercusión (entre 100 y 500 descargas) y se actualizó por última vez en 2015.

1.2. Aplicaciones para anilladores

En este apartado se han analizado varias herramientas para los anilladores que, en mayor o menor grado, automatizan o ayudan en sus tareas.

NouBioPro09

Desarrollado por el ICO (Institut Català d'Ornitologia) *[16]*, se trata de una aplicación de escritorio que ha ido incorporando muchas de las necesidades de los anilladores desde el año 2002.

Principales características

- Cumple con los estándares de anillamiento del ICO y está adaptada al remitente Aranzadi.
- Sistematiza la introducción y extracción de datos sobre anillas, anilladores, sesiones de anillamiento y balance anual, con funcionalidades adicionales de análisis de datos.

Desventajas

- Los datos se almacenan en el equipo local, con lo que están sujetos a pérdida o corrupción, y tampoco permite el trabajo colaborativo.
- Está enfocada sólo para anilladores, sin fines divulgativos o de transmisión de conocimiento.

A continuación se muestra una captura de pantalla de esta aplicación:

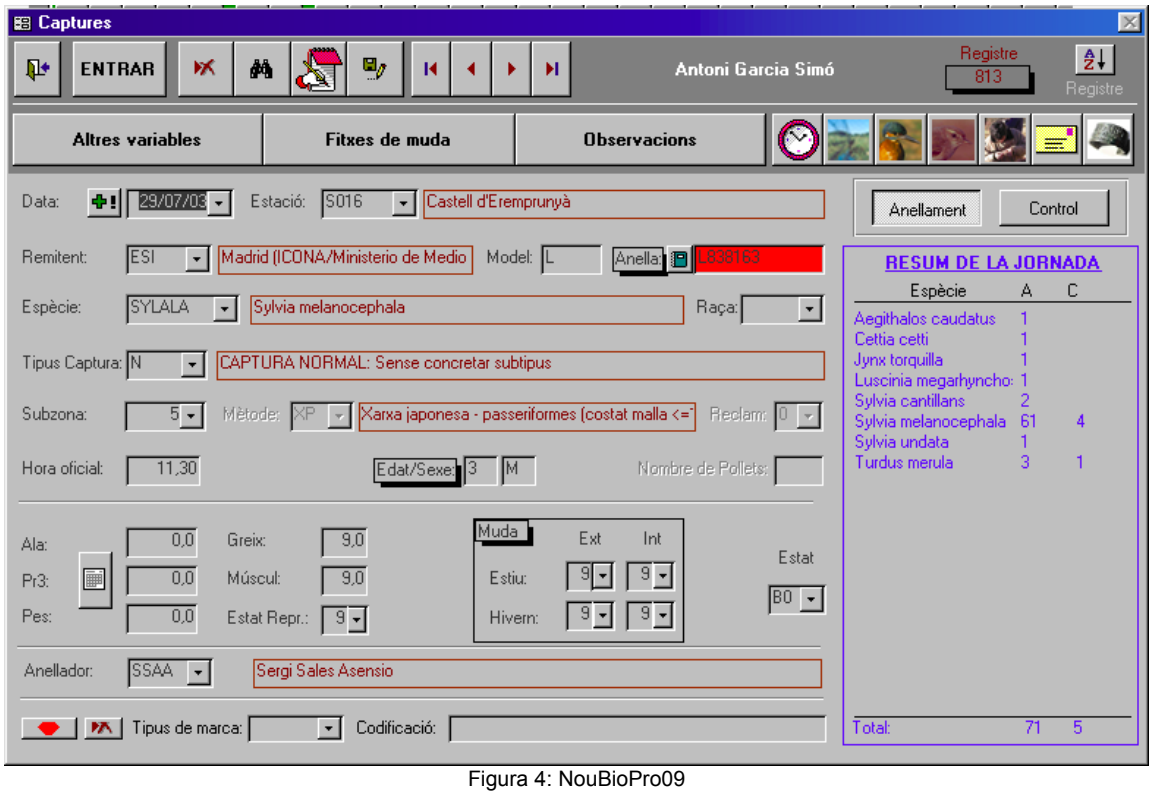


Figura 4: NouBioPro09

Bird Ringing Notes

Se trata de una app publicada en la plataforma fulcrum *[9]* y automatiza en gran medida la introducción de datos de anillamiento.

Principales características

- Se divide en dos partes: datos sobre sesiones de anillamiento, y los datos de cada captura específica.
- La app se puede adaptar para recoger una lista de especies locales.

Desventajas

- No permite el trabajo colaborativo ni divulgación al gran público.
- La app está publicada por un usuario de la plataforma pero no se da visibilidad sobre el grado de mantenimieno y soporte de la aplicación.

BirdRing

Esta app está disponible sólo en Google Play *[12]* y tiene un ámbito de uso en el norte de Europa, más concretamente en los paises bajos.

Principales características

- Se utiliza para leer anillas de colores con fines científicos y evitar errores comunes como la utilización de colores o letras incorrectas.

Desventajas

- No permite el trabajo colaborativo ni divulgación al gran público.
- Tiene un uso muy concreto y tiene un impacto limitado en el proceso de anillamiento

Otras aplicaciones

También se han analizado las siguientes aplicaciones, pero que por ser muy específicas o muy parecidas a las anteriores, se han agrupado en este apartado.

- BTO Ringers Info: Similar a BirdRing, utilizada en el Reino Unido.
- BirdTrack: Registro de observaciones, utilizadas por el British Trust for Ornithology para fines educativos y de investigación.
- Apps de SEO BirdLife: Esta organización dispone de otras aplicaciones diferentes a las mencionadas anteriormente, y específicas para diferentes programas de seguimiento propios, para aves acuáticas y también para la detección de aves raras o exóticas.

1.3. Conclusión

Analizadas las herramientas informáticas disponibles a día de hoy en el mercado, se puede concluir que no existe ninguna que cubra los tres puntos siguientes:

1) Herramienta de gestión de tareas y datos de anillamiento, tanto con o sin conexión a internet.
2) Punto de divulgación para observadores principiantes e intermedios.
3) Herramienta de caracterización automática de especies, con retroalimentación colaborativa

Las herramientas existentes cubren aspectos concretos, y en muchas ocasiones para territorios determinados, con lo cual se dispone de una buena oportunidad mediante este proyecto de iniciar una plataforma que consiga conciliar estas tres dimensiones del mundo de la ornitología.

2. Definición de objetivos/especificaciones del producto

Los objetivos del proyecto son los siguientes:

- Configuración de un servidor VPS que alojará todos los componentes de la arquitectura.
- Construcción de un back-end con Django 1.11, que implemente todos los modelos necesarios y exponga los servicios REST correspondientes.
- Diseño del front-end de la aplicación, acorde a los estándares actuales de diseño web responsivo y dando prioridad a la versión móvil de la aplicación web. Se tomará como base la aplicación NouBioPro09 *[16]* como base principal para el diseño de la aplicación, ya que es la aplicación que maś se ajusta a lo que se pretende conseguir en esta fase inicial correspondiente al TFM.
- Construcción de parte del front-ent con Angular 4. Se ha dado prioridad a la autenticación Django – Angular 4 y a la interacción de determinadas pantallas con el back-end. Se ha

intentado, aunque no haya interacción con el back-end, diseñar la mayoría de las pantallas con HTML y Bootstrap, esto es, que aparezcan en el front-end pero que no exista una lógica detrás que interactúe con Django.

- Despliegue de las dos partes desarrolladas en el servidor VPS para conseguir tener una aplicación web operativa.

En línea de los objetivos anteriores, se citan a continuación las partes del alcance del proyecto en forma de entregables o partes del producto final:

- Wireframes de la versión móvil de la aplicación web a desarrollar.
- Servidor VPS que alberga la aplicación web, con la configuración y medidas de seguridad correspondientes, y la documentación resultante.
- Back-end programado en Django que expone mediante un API REST los modelos correspondientes a la aplicación web y los endpoints necesarios para autenticación de los usuarios.
- Aplicación web programada en Angular 4 que implementa el ciclo completo de alta de usuarios y activación, así como la autenticación mediante los endpoints expuestos por el backend. Esta aplicación web implementa la mayoría de las pantallas diseñadas en los wireframes, pero, atendiendo a razones de restricciones temporales del TFM, interactúan o no con el back-end mediante web services.

Los siguientes objetivos o funcionalidades más concretas quedarán fuera del alcance de este proyecto:

- Diseño e implementación de un sistema de notificación automática de errores para la depuración de la plataforma. Sistema de recopilación de estadísticas para análisis de usabilidad. Una alternativa es implementar Piwik.
- Diseño y programación de los casos de prueba automáticos.
- Modificar el ciclo de cambio de e-mail de usuarios para que se vuelva a validar el nuevo e-mail.
- Traducción de la aplicación (back-end y front-end) a diferentes idiomas, aunque se configurará la plataforma para que pueda soportar diferentes idiomas.
- Sistema de reporting
- Migración de datos iniciales (países, provincias, localidades...)
- Gestión de archivos. En esta primera versión sólo se contempla el intercambio de datos, no archivos, entre front-end y servidor.
- Módulos adicionales como: blog, página de contacto, tienda online…
- Utilidades de importación/exportación masiva de datos.
- Se pospondrá la utilización del estándar Material Design y no se incluirá en este TFM.

Capítulo 3: Diseño

1. Arquitectura general de la aplicación

En el siguiente diagrama se muestra la arquitectura del sistema que soporta la aplicación:

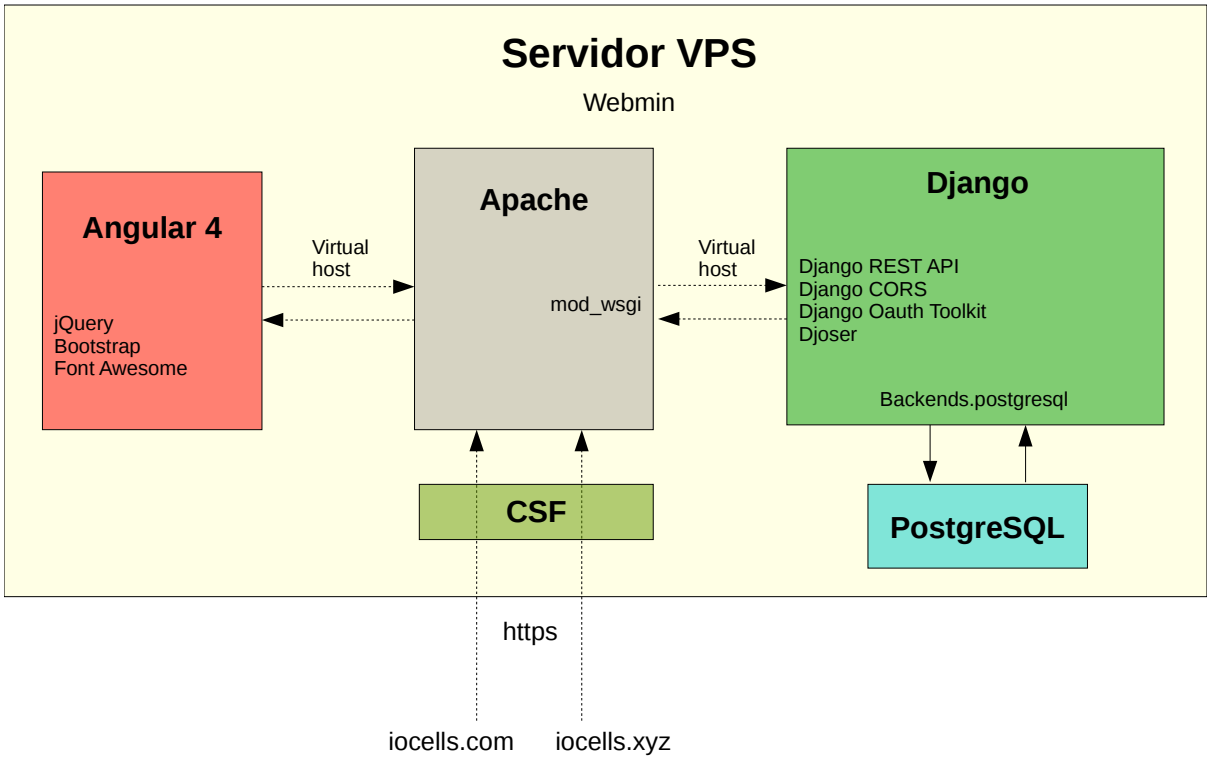


Figura 5: Arquitectura del sistema

Esta arquitectura se divide fundamentalmente en tres partes:

Servidor web (Apache): Se encarga de atender todas las peticiones http que llegan al servidor VPS. Dependiendo de la url (iocells.com o iocells.xyz), redireccionará la petición a uno de los dos extremos de los que se componen la aplicación: Django o Angular 4. Todas las peticiones http son redireccionadas al puerto 443 y encriptadas con su correspondiente certificado digital emitido por la compañía Let's encrypt.

Back-end (Django): Responde a todas las peticiones realizadas a la url iocells.xyz, a través del servidor web, y se encarga de implementar los servicios REST utilizando diferentes librerías de este framework, y también de gestionar el acceso a la base de datos PostgreSQL (en el entorno local se utiliza una base de datos SQLite).

Front-end (Angular 4): Responde a las peticiones realizadas a la url iocells.com, redirigidas también por Apache, aunque realmente el código del front-end se descarga y ejecuta en el navegador del usuario. Como librerías adicionales, se utiliza Bootstrap para dar carácter responsivo a la aplicación,

jQuery para implementar determinados efectos de navegación y mejorar la usabilidad, y Font Awesome para los iconos.

Además, el servidor VPS dispone del firewall CSF en el que se han habilitado sólo los puertos necesarios para el funcionamiento de la aplicación, y también de la utilidad Webmin para gestionar de forma más amigable el propio servidor VPS.

2. Diseño back-end

2.1. Diseño de modelos

La base de datos constará de las siguientes tablas, para poder implementar las funcionalidades perseguidas:

Mock-up	Tablas implicadas
Inicio Panel de inicio Panel de administración	No hay tablas implicadas
Login Registro de usuario Recuperación de contraseña Cambio de contraseña	Tabla de usuario. Tabla de tokens de acceso (para autenticación). Tabla de tokens de renovación (para autenticación).
Especies	Tabla de especies
Subespecies	Tabla de subespecies
Municipios	Tabla de municipios
Provincias	Tabla de provincias
Países	Tabla de países
Localidades	Tabla de localidades
Estaciones	Tabla de estaciones
Remitentes	Tabla de remitentes
Usuarios	Tabla de usuarios
Variables personalizadas, Variables de captura	Tabla de variables
Anillas	Tabla de anillas
Jornadas, detalle de jornadas	Tabla de jornadas
Subzonas de jornada	Tabla de subzonas
Metorología de jornada	Tabla de meteorología
Reclamos de jornada	Tabla de reclamos
Capturas	Tabla de capturas
Fichas de muda de captura	Tabla de fichas de muda de captura
Variables de captura	Tabla de variables de captura
Balance anual	Tabla de balance anual
Errores balance anual	Tabla de errores del balance anual

Tabla 3: Tablas de bases de datos

Para la creación de dichas tablas en forma de modelos se ha utilizado el framework Django en su versión 1.11. En el Anexo C se detalla la instalación en local de un entorno de desarrollo para Django, junto con el IDE utilizado PyCharm, en su versión Community, y también la creación inicial del proyecto Django.

La tabla de usuarios ya vendrá por defecto definida al instalar los módulos básicos de Django, aunque será necesario aplicar unas modificaciones para que el campo que identifique unívocamente al usuario sea el campo de e-mail, según se detalla en el Anexo D del presente documento. Además, se definirán nuevos campos para el modelo de usuario para que soporte los datos adicionales requeridos.

A partir de este inventario de tablas, se especifican los detalles de cada una de estas tablas en el Anexo H.

2.2. Servicios web

A partir de las tablas diseñadas, se extraen los siguientes servicios web necesarios:

Mock-up	Servicios web implicados
Inicio Panel de inicio Panel de administración	No hay servicios web implicados.
Login Registro de usuario Recuperación de contraseña Cambio de contraseña	Registro de usuario Activar usuario Cambiar e-mail Login Recuperar contraseña Confirmar nueva contraseña Refrescar token Logout
Especies	Administración de Especies Lista de Especies
Subespecies	Administración de Subespecies Lista de Subespecies
Municipios	Administración de Municipios Lista de Municipios
Provincias	Administración de Provincias Lista de Provincias
Países	Administración de Países Lista de Países
Localidades	Administración de Localidades Lista de Localidades
Estaciones	Administración de Estaciones Lista de Estaciones

Remitentes	Administración de Remitentes Lista de Remitentes
Usuarios	Administración de Usuarios Lista de Usuarios
Variables personalizadas	Administración de Variables Lista de Variables
Anillas	Administración de Anillas
Jornadas, detalle de jornadas	Gestión de Jornadas
Subzonas de jornada	Gestión de Subzonas
Metorología de jornada	Gestión de Metorología
Reclamos de jornada	Gestión de Reclamos
Capturas	Gestión de Capturas
Variables de captura	Gestión de Variables de captura
Fichas de muda de captura	Gestión de Fichas de muda
Balance anual	Gestión de Balance anual
Errores de balance anual	Errores de balance anual

Tabla 4: Servicios web

De forma análoga a la definición de tablas realizada en el apartado anterior, se exponen en el Anexo H las particularidades de cada uno de estos servicios web.

3. Diseño front-end

En los siguientes subapartados se especifican, en primer lugar, las decisiones de diseño del front-end según las funcionalidades que aporta Angular 4, y a continuación se muestran también los mock-ups realizados para poder iniciar el desarrollo de la aplicación (tanto del back-end como front-end),

3.1. Estructura del cliente Angular 4

Angular 4 es un framework de desarrollo frontend que utiliza el lenguaje de programación Typescript (aunque se pueden utilizar también otros lenguajes de programación, e incluso combinarlos, como es el caso de JavaScript) basado en el paradigma SPA. Según este paradigma, la aplicación web consta realmente de una sola página web (documento HTML), y la interacción con el usuario se construye dinámicamente conforme se producen los eventos en la capa de presentación.

Para poder implementar este paradigma, Angular 4 introduce una serie de elementos, cada uno de los cuales tiene un propósito determinado dentro del conjunto, lo cual hace que una aplicación desarrollada con este framework pueda llegar a ser altamente modular y reutilizable.

A continuación se describen las partes fundamentales de que se puede componer un proyecto realizado con Angular 4:

Concepto	Descripción
Proyecto	Se trata de la estructura de directorios y archivos que envuelve todo el proyecto, y que en

	su nivel más alto consta de una serie de archivos de configuración que se encargan de especificar todas las librerías implicadas con sus correspondientes versiones, y determinadas variables globales al proyecto, entre otros aspectos.
Módulo	Se trata de una agrupación lógica de subcomponentes, bajo la premisa de que ese mismo módulo sea independiente de los demás y se pueda reutilizar en otros proyectos diferentes. Se puede distinguir entre el módulo principal del proyecto, que importará todos los demás módulos necesarios, los módulos secundarios que contendrán al resto de subcomponentes, y también los módulos de enrutamiento, que servirán para determinar el enlace entre URLs y los componentes.
Componente	Constituye la unidad básica de interacción con el frontend y su finalidad es implementar una o varias partes de una página web, junto con toda la lógica asociada que gobernará esa parte de la interacción con el usuario así como la utilización de todos los recursos necesarios para llevar a cabo esa interacción. Los componentes están normalmente compuestos por cuatro ficheros con las siguientes extensiones: - ts: contiene toda la lógica del componente, en lenguaje Typescript. - html: contiene la plantilla en HTML del componente. - css: en este fichero se pueden definir estilos específicos para el componente, sin entrar en conflicto con los demás estilos de la aplicación. - spec.ts: se utiliza para programar los casos de prueba del componente. En el presente proyecto se han utilizado sólo los ficheros .ts y .html de los componentes.
Servicio	También llamado proveedor, se encarga de independizar a los componentes de la utilización de otros recursos externos, como bases de datos o servicios web REST.
Guardián	Se trata de un servicio dedicado exclusivamente a realizar determinadas comprobaciones cuando se producen eventos de navegación en la aplicación web. Pueden restringir el acceso a determinadas rutas (por ejemplo, si el usuario no está autenticado o no tiene suficientes privilegios), y también puede controlar el abandono de una determinada ruta bajo determinadas condiciones (abandono de un formulario con cambios pendientes, por ejemplo).
Directiva	Se trata de una abstracción de código que cambia el comportamiento o la apariencia de un elemento en la página web.
Pipe	Es otro tipo de abstacción, que, cambia el formato de un valor (texto, número, fecha…) mostrado en la página web.

Tabla 5: Conceptos Angular 4

Mejora del rendimiento con Angular 4

En este proyecto se ha utilizado la utilidad Lazy Loading de Angular 4, lo cual ha influenciado en la forma de estructurar la aplicación, como se verá en el siguiente subapartado.

Esta utilidad de Angular 4 hace que, tras el proceso de compilación de la aplicación, los ficheros resultantes puedan ser divididos según diferentes módulos de la aplicación, de forma que el navegador web sólo necesite descargar los ficheros correspondientes a la parte de la aplicación que se está utilizando.
Este aspecto es importante cuando existen diferentes perfiles de usuarios que utilizan la aplicación, de forma que cada uno accede a diferentes partes. Por ejemplo, un usuario que no tenga acceso a un módulo reservado para superusuarios no debería descargar en el navegador esa parte de la aplicación, lo cual ralentizaría el proceso de carga.
De esta forma, se consigue que cada parte de la aplicación se descargue al navegador web sólo cuando sea necesario.

Estructura del proyecto Angular 4

Partiendo de la introducción realizada en el apartado anterior, en este subapartado se detallan los módulos en que se ha dividido el proyecto de Angular 4, en base a los siguientes criterios:

- Lazy loading: se han separado las diferentes funcionalidades para que el tiempo de carga no se vea afectado por el tamaño total de la aplicación, sino por las funcionalidades que el usuario se dispone a utilizar en cada momento.
- Reutilización: existen determinadas partes del proyecto que pueden servir como base a futuros proyectos, por lo que se ha decidido crear un módulo base reutilizable para cualquier otro proyecto.
- Planificación: atendiendo a la planificación planteada para este TFM, se ha decidido separar en dos grupos la funcionalidad específica para la aplicación de anillamiento. Estos dos módulos específicos también responden a diferentes perfiles de usuario, por lo tanto no ha sido una decisión puramente de planificación sino que también se ha tenido en cuenta la carga por Lazy Loading.

Por tanto se distinguen tres bloques en la aplicación, que se detallan en los siguientes subapartados.

Módulo básico

La intención en el desarrollo de este módulo es hospedar todos aquellos componentes, servicios, pipes y directivas que se vayan a utilizar potencialmente en cualquiera de los demás módulos.
Este módulo implementa toda la funcionalidad que se encarga (en combinación con el backend) del alta, activación y mantenimiendo de la cuenta de usuario, cambio de e-mail y contraseña, inicio y cierre de sesión, además de la construcción del panel de navegación principal según el perfil de usuario, páginas de redirección por defecto (página no encontrada, página de error de servidor…), y definición de clases e interfaces generales que servirán de base para la construcción de componentes con características similares.

Módulo de administración

Este módulo agrupa todas las funcionalidades que permitirán a un usuario, con los privilegios adecuados, gestionar los datos maestros de la aplicación.

Exceptuando el mantenimiento de usuarios, una adecuada migración inicial de datos hará que el acceso a este módulo no sea tan frecuente como a los demás módulos.

Se podría haber utilizado el backoffice de Django para la administración de estos datos maestros pero se ha decidido implementar esta parte con Angular 4, ya que el backoffice de Django permite efectuar operaciones de mucho más riesgo. Para el frontend con Angular 4 se ha restringido, por ejemplo, el borrado de registros en la base de datos a través del REST API de Django, de forma que no se podrán eliminar datos maestros sino inactivarlos, para poder conservar datos históricos en el sistema.

Módulo de anillamiento

Por último, el módulo de anillamiento es el más complejo desde el punto de vista de interacción con el usuario, ya que es el que va a permitir a los usuarios normales efectuar todas las operaciones de registro, procesamiento y consulta de sus datos de anillamiento.

Se trata de la parte mollar de la aplicación que podrá determinar el éxito o no del proyecto, ya que los módulos anteriores son en cierta medida transparentes para el usuario normal de la aplicación.

Será en este módulo en el que se deberán tener en máxima consideración los criterios de usabilidad para que la interacción y navegación a través de la aplicación sea lo más intuitiva y satisfactoria posible.

3.2. Mock-ups

Para la realización de los siguientes mock-ups se ha utilizado la demo online de Balsamiq Mockups (consultar Anexo B para más detalles).

Pantallas sin autenticación

Éstas son las pantallas que cualquier usuario que no esté autenticado en el sistema podrá ver:

Mock-up	Observaciones

Figura 6: Mock-up inicio

Inicio

Esta pantalla será la pantalla de inicio del sitio web, y en su versión móvil será simplemente una pantalla de bienvenida sencilla con un botón que llevará a la pantalla de login.

Figura 7: Mock-up iniciar sesión

Login

En esta pantalla el usuario introducirá su usuario y contraseña para iniciar sesión, y también dispondrá dos enlaces a las siguientes pantallas:

- Recuperación de contraseña.
- Registro de usuario.

Figura 8: Mock-up registro usuario

Solicitud de registro de usuario

Mediante esta pantalla el usuario introducirá el e-mail y la contraseña con los que se podrá autenticar en el sistema.

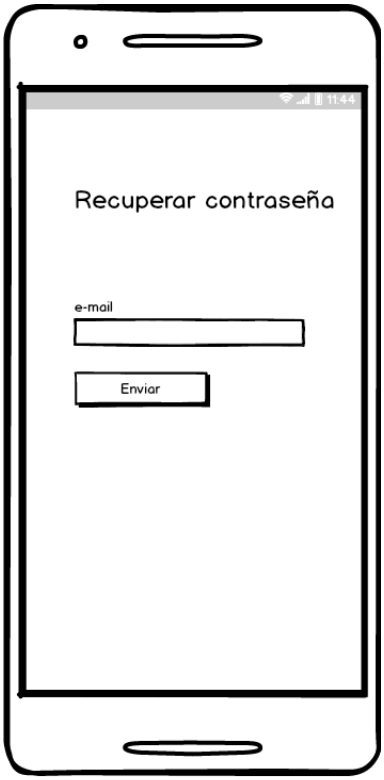


Figura 9: Mock-up recuperar contraseña

Solicitud de recuperación de contraseña

El usuario podrá en esta pantalla solicitar que se le envíe un correo para poder recuperar su contraseña, introduciendo el e-mail asociado a su cuenta.

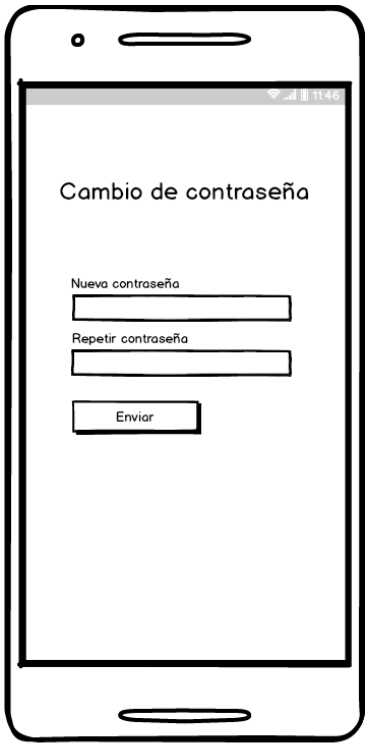


Figura 10: Mock-up cambio de contraseña

Cambio de contraseña

Tras solicitar la recuperación de la contraseña, el usuario recibirá un e-mail en su correo con un enlace que le llevará a esta pantalla, donde podrá elegir su nueva contraseña. Además tendrá que confirmarla de nuevo para evitar posibles errores.

Figura 11: Mock-up confirmación

Confirmación genérica

Al realizar muchas de las acciones, el sistema mostrará una pantalla de confirmación de la acción realizada. Este mock-up muestra de forma genérica las confirmaciones siguientes:

- Confirmación de logout
- Confirmación de cambio de contraseña
- Confirmación de la solicitud de recuperación de contraseña
- Confirmación de solicitud de registro de usuario
- Confirmación de activación de cuenta

Tabla 6: Pantallas sin autenticación

Panel de inicio

Una vez el usuario se haya autenticado, accederá al panel de inicio de la aplicación, que tendrá dos versiones, dependiendo de si se trata de un usuario normal o un superusuario:

Mock-up	Observaciones
 Figura 12: Mock-up panel inicio	**Panel de inicio de superusuario** Esta pantalla tendrá seis iconos, cada uno dirigirá a un módulo diferente de la aplicación. **Panel de inicio de usuario** Será igual que el de superusuario, pero no se mostrará el icono de Administración.

Tabla 7: Panel de inicio

Módulo de administración

En este apartado se esbozan las pantallas que formarán parte del mantenimiento de datos maestros de la aplicación, sólo disponible para superusuarios. Debido a que el diseño de las pantallas de mantenimiento es muy similar, se va a mostrar un ejemplo genérico de mock-up, y se indicarán las particularidades aplicables a cada uno de los modelos.

Mock-up	Observaciones

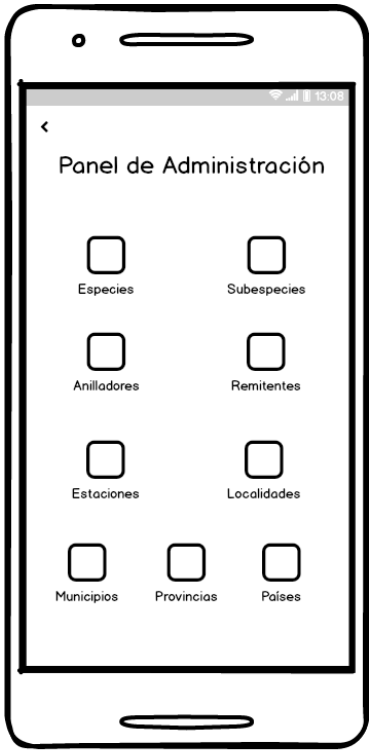


Figura 13: Mock-up panel administración

Panel de Administración

Mediante esta pantalla se puede acceder a todas las pantallas de mantenimiento de datos maestros, así como volver al panel inicial.

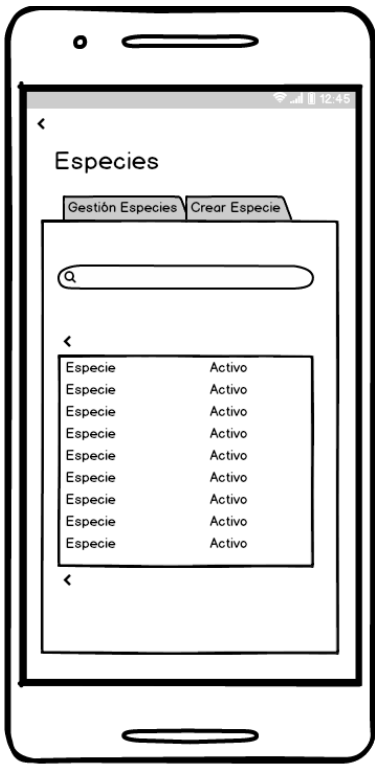


Figura 14: Mock-up búsqueda

Pantalla de búsqueda

Esta pantalla se encontrará en la pestaña “Gestión”, y es a la que se accederá por defecto al pulsar sobre alguna de las opciones del panel de administración.

Cada modelo tendrá unos campos de búsqueda y de resultados diferentes, que se detallan en la tabla siguiente.

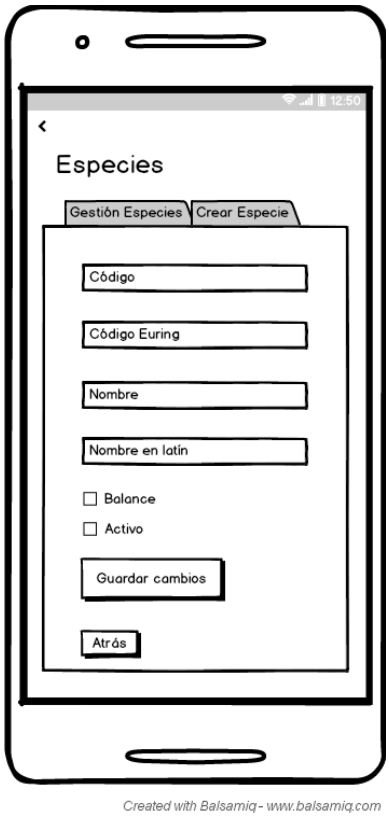


Figura 15: Detalle

Pantalla de detalle

Al pulsar sobre alguna de los resultados de la pantalla de búsqueda, aparecerá esta pantalla de detalle en la que se podrán modificar los datos del registro seleccionado.

Esta misma pantalla se utilizará también en la pestaña "Crear", con la diferencia que se tendrán que introducir todos los datos y el botón dirá "Crear" en lugar de "Guardar cambios".

Todas las pantallas de datos maestros tendrán un campo que determinará si está activo o no.

Tabla 8: Pantallas de administración

A continuación se resume en la siguiente tabla las particularidades de cada pantalla:

Pantalla	Campos búsqueda	Campos multiregistro	Campos detalle
Especies	Código Código Euring Nombre Nombre en latín Activo	Código Nombre Activo	Código Código Euring Nombre Nombre en latín Balance Activo
Subespecies	Código especie Código raza Nombre raza Activo	Código raza Nombre raza Activo	Código especie Código raza Nombre raza Activo
Municipios	Código Nombre Activo	Código Nombre Activo	Código Nombre Activo
Provincias	Código Nombre Activo	Código Nombre Activo	Código Nombre Activo
Países	Código Nombre Activo	Código Nombre Activo	Código Nombre Activo

Localidades	Código Nombre Activo	Código Nombre Activo	Código Nombre Municipio Provincia País UTM Activo
Estaciones	Nombre Nombre localidad Tipo de estación Estándar utilizado Activo	Nombre Activo	Nombre Localidad Extensión Tipo estación Estándar utilizado Activo
Remitentes	Código Nombre Nombre país Activo	Código Nombre Activo	Código Remitente País Activo
Usuarios	e-mail Nombre Apellidos Activo Superusuario Anillador	Nombre Apellidos e-mail Activo Superusuario	E-mail Nombre Apellidos Dirección Municipio Código postal Provincia Teléfono Superusuario Activo Anillador (sí/no)
Variables personalizadas	Código Variable Descripción	Código Variable	Código Variable Descripción

Tabla 9: Particularidades de las pantallas de administración

Módulo de anillamiento

Las pantallas que se muestran en este subapartado serán las que utilizarán los usuarios normales en sus actividades de registro y extracción de datos:

Mock-up	Observaciones

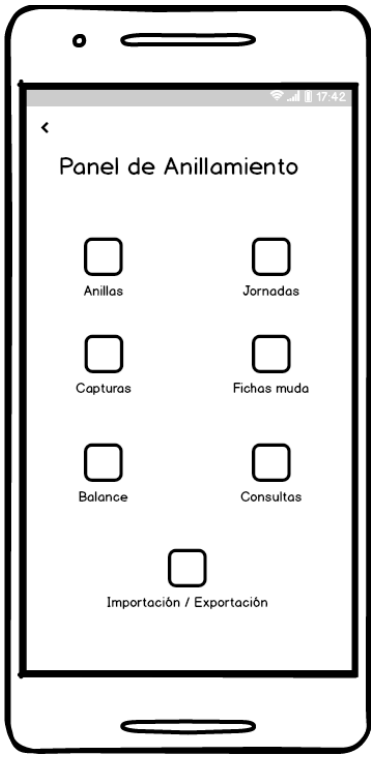


Figura 16: Panel de anillamiento

Panel de Anillamiento

Mediante este panel se accederán a las opciones de anillamiento de este módulo.

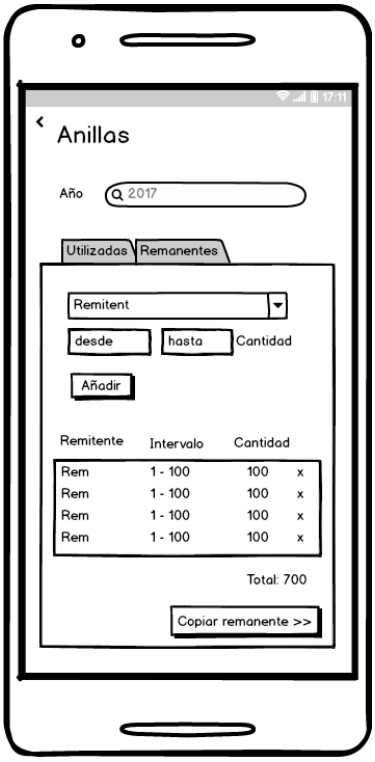


Figura 17: Mock-up anillas

Gestión de anillas

En esta pantalla existen dos pestañas: una para anillas del presente año, y la otra para las anillas remanentes para el año siguiente.

En la pestaña “Utilizadas” se gestionarán las anillas disponibles para un determinado año. Primero habrá un bloque para crear nuevos registros de anillas, que al crearlos pasarán a estar disponibles en la lista de la parte inferior. Los registros de la lista se podrán eliminar, y en el caso de que no se hayan utilizado todas las anillas de un año, se podrán asignar al año siguiente mediante el botón “Copiar remanente”.

En la pestaña “Remanentes” se mostrarán las anillas que han sobrado del año seleccionado y que se han asignado al año siguiente.

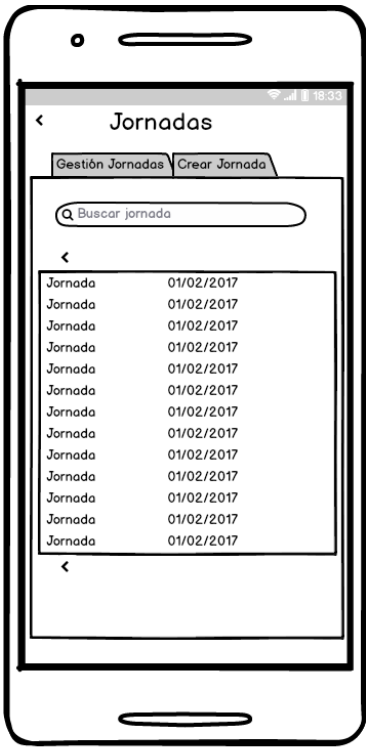


Figura 18: Mock-up jornadas

Jornadas de anillamiento (búsqueda)

Mediante esta primera pestaña “Gestión de jornadas” (a la que se accederá por defecto tras pulsar sobre la opción “Jornadas” del panel de administración), se podrán buscar las jornadas del usuario mediante diferentes criterios, y los resultados se mostrarán en forma de lista paginada. Se podrá pulsar en cada uno de los resultados para acceder al detalle, que se muestra a continuación.

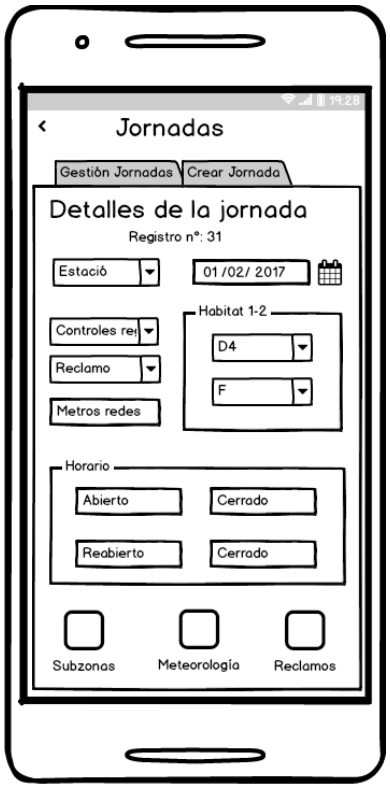


Figura 19: Mock-up detalles jornada

Jornadas de anillamiento (detalle)

En el detalle de la jornada se mostrará primero los campos pertenecientes a la cabecera de la jornada, que serán los referentes a:

- Estación
- Fecha de la jornada
- Controles registrados
- Uso de reclamos
- Metros de las redes
- Datos del hábitat
- Horario de la jornada

En el fondo de la pantalla se mostrarán tres iconos que enlazarán con el resto de datos asociados a la jornada: subzonas, meteorología y reclamos. Estos detalles de la jornada se muestran a continuación.

Figura 20: Mock-up subzonas

Jornadas de anillamiento (subzonas)

En el detalle de las subzonas asociadas a la jornada se dispondrá primeramente de una lista (o desplegable) con todas las subzonas creadas para la jornada. Al elegir una subzona, aparecerán los detalles a continuación, con los campos:

- Controles registrados
- Uso de reclamo
- Número de trampas
- Metros de las redes
- Datos del hábitat
- Horario

Además, se dispondrá de dos botones que realizarán lo siguiente:

- Reloj: copiará de la cabecera los datos del horario
- Pegar: copiará de la cabecera los datos cuyo nombre coincide

Por último, para crear una nueva subzona se dispondrá de un botón que llevará al formulario de creación de una nueva subzona.

Figura 21: Mock-up crear subzona

Jornadas de anillamiento (crear subzona)

Éste será un formulario idéntico al detalle de subzona de la pantalla anterior, pero servirá para introducir una nueva subzona.

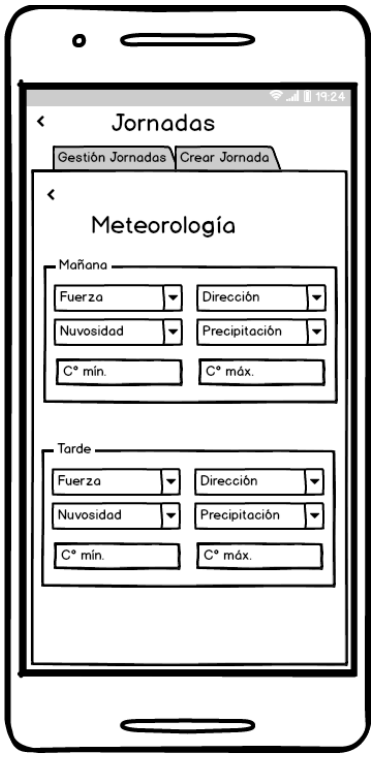


Figura 22: Mock-up meteorología

Jornadas de anillamiento (meteorología)

Desde la pantalla de cabecera de jornada se podrá acceder también a la pantalla de meteorología, donde se introducirán las variables significativas de este aspecto, para la franja horaria de mañana y la de tarde también.

Figura 23: Mock-up reclamos

Jornadas de anillamiento (Reclamos)

Similar al modo en el que funciona el mantenimiento de subzonas, para el mantenimineto de reclamos se dispondrá de una primera lista de selección de reclamo, y a continuación su detalle con los campos:

- Especie
- Subzona
- Método
- Horario

Y además dispondrá de un icono para agilizar la copia de la cabecera de los campos con el mismo nombre.

Por último, se dispone de un botón para crear un nuevo reclamo asociado a la jornada.

Figura 24: Mock-up crear reclamo

Jornadas de anillamiento (Crear reclamo)

En esta pantalla se dispone de los mismos campos que para el mantenimiento de reclamos, pero servirá sólo para crear un nuevo reclamo.

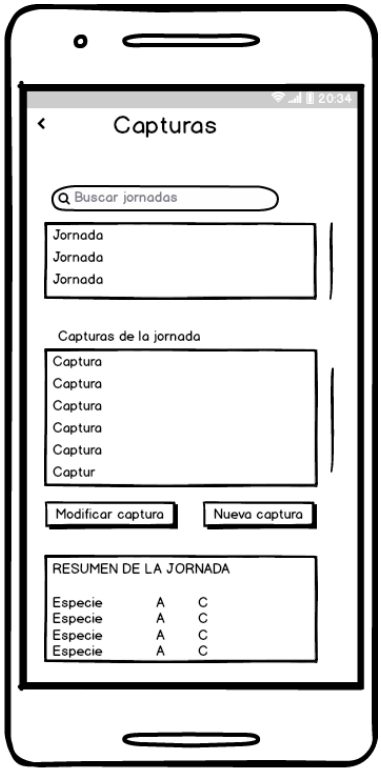


Figura 25: Mock-up capturas

Capturas

A esta pantalla se accederá desde el panel de anillamiento, y constará de los siguientes bloques:

- Búsqueda de jornada.
- Bloque de capturas asociadas a la jornada seleccionada. Se podrá modificar una captura seleccionada, o crear una nueva.
- Bloque textual donde se mostrará el resumen de la jornada.

Capturas (detalle)

Dada la complejidad del registro de capturas, se propone en este caso desglosar el contenido de la pantalla en subpestañas que separen los diferentes bloques de campos y prioricen los más importantes en las primeras posiciones. Así, se distinguen los siguientes bloques:

- Datos generales: contendrá información sobre la jornada, el remitente al cual irán los datos, el modelo de la anilla, la especie/subespecie, el tipo de captura, la subzona, el horario, la edad y el sexo del ave.
- Características: en este bloque se introducirán las características particulares observadas en cada captura.
- Variables personalizadas: en esta subpestaña se introducirán valores para las variables personalizadas definidas en la sección de administración.
- Por último, se introducirán en la subpestaña de ficha de muda las variables correspondientes que identifiquen la muda.

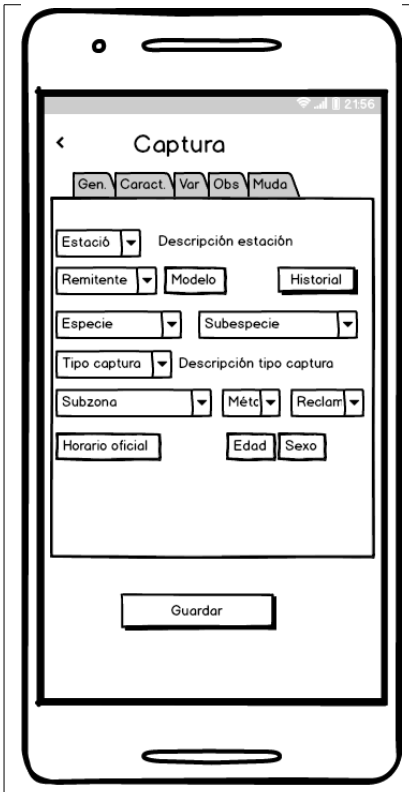


Figura 26: Mock-up capturas - datos generales

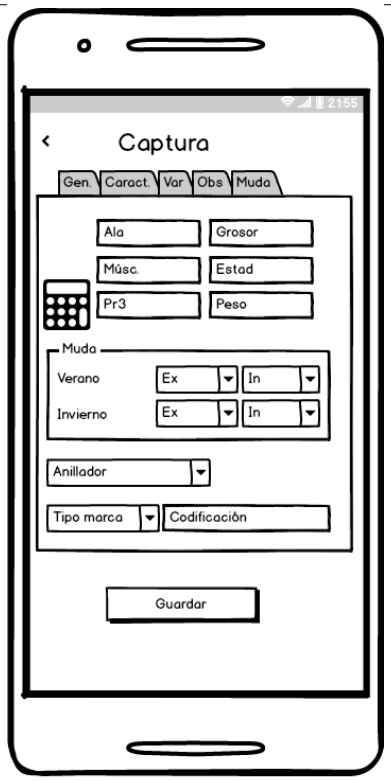


Figura 27: Mock-up capturas - características

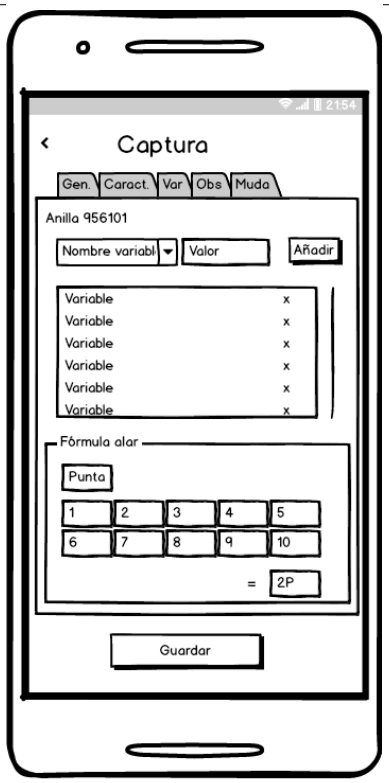
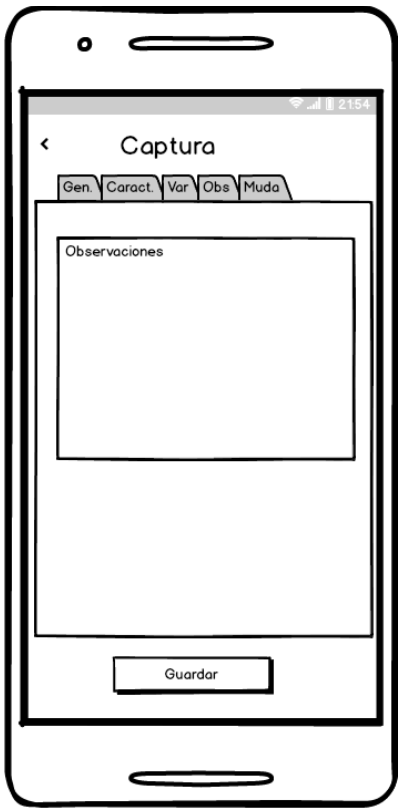
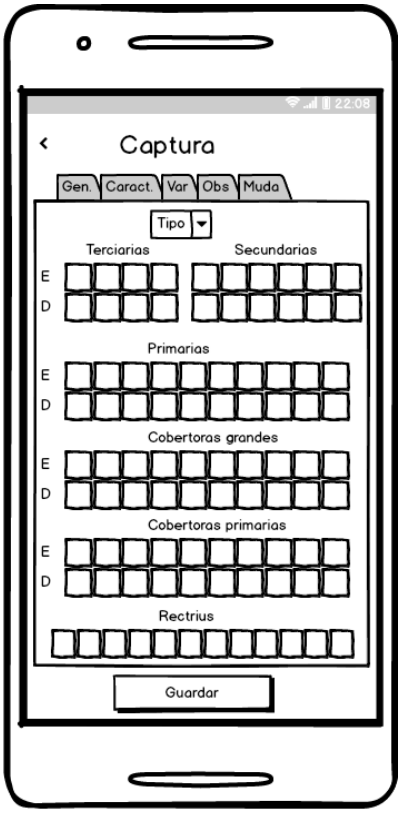


Figura 28: Mock-up capturas - variables



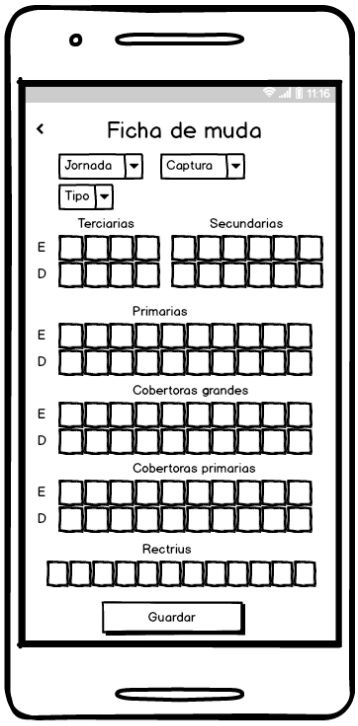
Created with Balsamiq - www.balsamiq.com

Figura 30: Mock-up capturas - observaciones



Created with Balsamiq - www.balsamiq.com

Figura 29: Mock-up capturas - ficha de muda



Created with Balsamiq - www.balsamiq.com

Figura 31: Mock-up ficha de muda

Fichas de muda

En esta pantalla se podrán introducir fichas de muda asociadas a una determinada captura, sin necesidad de pasar por la pantalla de capturas.

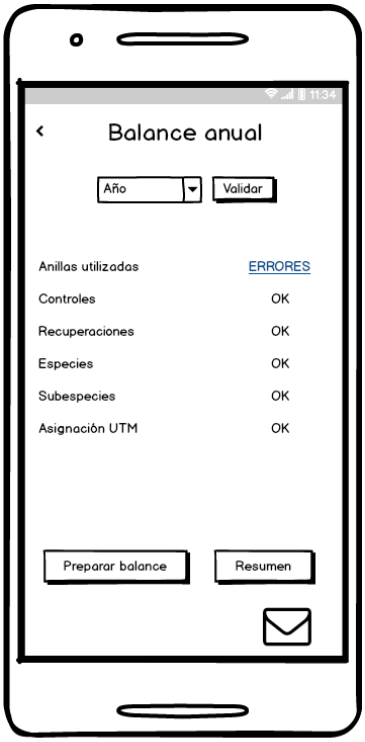


Figura 32: Mock-up balance anual

Balance anual

En la pantalla de balance anual finalizará el proceso de anillamiento para el año y se hará balance de las capturas realizadas.

Primero se introducirá el año, y a continuación se pulsará sobre el botón Validar. Los resultados de la validación se mostrarán a continuación. Si hubiese cualquier error se indicará con un enlace que llevará al detalle de los errores detectados.

Una vez se hayan subsanado los errores, se podrá generar el balance mediante el botón Preparar balance, ver un resumen de dicho balance, e incluso enviar por correo electrónico al usuario el contenido del resultado.

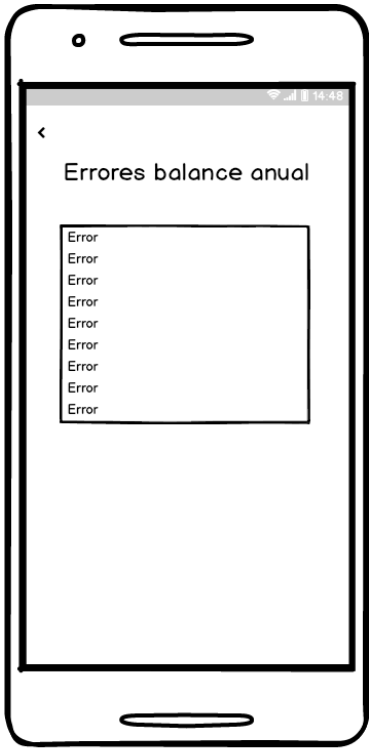


Figura 33: Mock-up errores del balance

Errores balance

En esta pantalla se mostrarán todas aquellas incidencias detectadas durante el proceso de validación del balance anual.

Tabla 10: Pantallas de anillamiento

El diseño de las pantallas de Consultas y Exportación/Importación se analizará en posteriores fases del proyecto, fuera del alcance de este TFM (aunque se deja como referencia en el panel de anillamiento), acorde con las especificaciones realizadas en el apartado correspondiente al alcance del TFM.

Capítulo 4: Implementación

1. Back-end

1.1. Programación de modelos

Todas las tablas diseñadas en el capítulo anterior se traducen en la creación de diferentes modelos que extenderán la clase “Model” de Django, y que se definirán en el fichero models.py de la aplicación “anilla”. Para crear dicha aplicación se ejecutará el siguiente comando, en el directorio raíz del proyecto (el que contiene el fichero manage.py):

```
>> sudo python3 manage.py startapp anilla
```

Como muestra, se expone la creación del modelo para las especies:

```
class Especie (models.Model):
    codigo = models.CharField(max_length=15, unique=True)
    codigo_euring = models.CharField(max_length=15, unique=True)
    nombre = models.CharField(max_length=300, verbose_name='Nombre de la especie')
    nombre_latin = models.CharField(max_length=300, verbose_name='Especie en latín')
    balance = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)

    def __str__(self):
        return self.codigo + ' - ' + self.nombre

    def __unicode__(self):
        return u'%s' % (self.codigo + ' - ' + self.nombre)

    def delete(self):
        self.is_active = False
        self.save()
```

Además de la definición de los campos de que consta el modelo, cabe notar los dos aspectos siguientes:

- Se definen las funciones __str__ y __unicode__ para configurar la gestión del modelo en el backoffice de Django.
- Con el fin de que no se eliminen datos maestros y se conserve un histórico de registros, se ha sobreescrito el método “delete” para que no se puedan borrar datos maestros de forma efectiva, sólo será posible inactivarlos. Más tarde, en el frontend (Angular), sólo se podrán elegir objetos activos cuando se creen otros objetos relacionados (por ejemplo: en el caso de definir una subespecie nueva, sólo se permitirá elegir especies activas). Este método se ha sobreescrito para todos los datos maestros, es decir: todas las tablas definidas excepto las de ficha de muda, anillo, jornadas, subzonas, reclamos y meteorología.

Tras definir todos los modelos en el fichero manage.py, es necesario ejecutar los siguientes comandos para que las modificaciones se trasladen a la base de datos:

```
>> sudo python3 manage.py makemigrations
```

```
>> sudo python3 manage.py migrate
```

También, ya que se ha modificado el modelo de usuarios, se creará un nuevo superusuario con el siguiente comando:

```
>> su python3 manage.py createsuperuser
```

Una vez creado el superusuario y dados de alta los modelos en el panel de administración propio de Django, según se detalla en el Anexo V, se podrá acceder a este backoffice mediante la URL http://localhost:8000/admin:

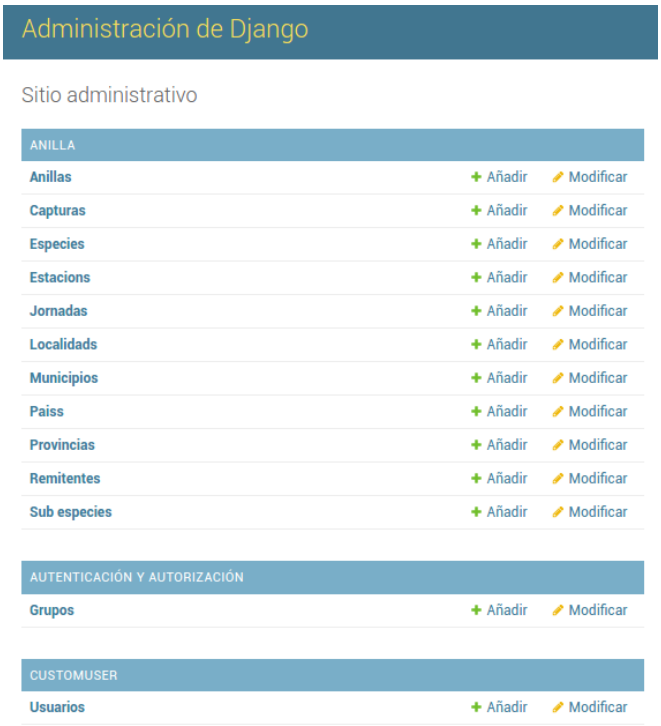


Figura 34: Back-office de Django

Este backoffice permite gestionar todos los modelos mediante la cuenta de superusuario. Es por tanto una herramienta restringida sólo a administradores del sistema, para poder realizar labores de mantenimiento y carga de datos, o correcciones puntuales.

Como última observación, cabe destacar que el motivo por el que se ha incluido en este apartado las tablas de tokens de autenticación y la de tokens de renovación es sólo por tenerlas inventariadas. Realmente estas tablas se definirán al instalar las librerías que manejarán la autenticación por Oauth2.

1.2. Programación de servicios web

Para la programación de los servicios web en Django se ha utilizado la librería externa Django Rest Framework, cuya instalación y configuración se detalla en el Anexo E.

Para definir un nuevo servicio web con Django Rest Framework, se han de seguir los siguientes pasos:

1. Definición del serializador: sirve para convertir en formato JSON los datos provenientes de los modelos, y consiste en definir nuevas clases que extienden a clases pre-existentes (HyperlinkedModelSerializer es la que se ha utilizado en este proyecto) donde se define el modelo al que va asociado el serializador y los campos de los modelos que se requiere convertir a formato JSON, o los datos que han de estar presentes al enviar un fichero JSON al servicio. Los serializadores se definen en el fichero serializers.py de la aplicación a la cual pertenecen los modelos.

Como ejemplo, se muestra a continuación la definición del serializador para especies:

```
class EspecieSerializer(serializers.HyperlinkedModelSerializer):
    url = HyperlinkedIdentityField(view_name='admin_especies-detail')
    class Meta:
        model = Especie
        fields = ('url','id','codigo','codigo_euring','nombre','nombre_latin','balance', 'is_active')
```

Utilizando la clase HyperlinkedModelSerializer podremos (como se verá posteriormente en este apartado) visualizar y probar los servicios web mediante el modo browsable de Django Rest Framework, sin necesidad de utilizar ninguna herramienta adicional (como Postman *[17]*). El valor del parámetro view_name se define en el tercer paso, cuando se definan las URLs en las que contestará cada servicio.

2. Definición de vistas: Representan la parte del servicio web que se va a encargar de realizar las validaciones previas antes de procesar los datos, y también será la capa que aplicará las reglas de seguridad definidas para cada servicio web, en base a las cuales definirá qué registros de la base de datos afectan a cada petición. Es posible también definir opciones particulares de paginación, y de búsqueda por campos. Las vistas se definen en el fichero views.py de cada aplicación Django que contiene los modelos asociados.

Opcionalmente, en caso de que las opciones de permisos no se ajusten a los requerimientos del servicio, es posible definir permisos adicionales en el fichero permissions.py, extendiendo la clase BasePermission. Más tarde, estas nuevas clases son importadas en views.py y utilizadas en las vistas que corresponda.

Siguiendo con el ejemplo del servicio web para especies, se muestra a continuación la definición de la vista para la administración de especies, y la definición de un permiso para que sólo los usuarios con privilegios de superusuario puedan acceder a este servicio web:

views.py:

```
class EspecieViewSet(viewsets.ModelViewSet):
    serializer_class = EspecieSerializer
    permission_classes = (permissions.IsAuthenticated,IsSuperuser)
    search_fields=('codigo', 'codigo_euring','nombre','nombre_latin')

    def get_queryset(self):
        user = self.request.user
        if user.is_superuser:
            return Especie.objects.all()

    def destroy(self, request, pk=None):
```

```
        object = self.get_object()
        object.is_active = False
        object.save()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

Esta vista pone de manifiesto los siguientes aspectos:

- Se hace referencia al serializador definido previamente
- Mediante el método get_queryset se restringen los datos a los que afecta la petición
- Mediante el método destroy se sobreescribe las acciones por defecto de borrado, y no se permite borrar sino desactivar el flag de activo.

permissions.py:

```
class IsSuperuser(permissions.BasePermission):
    def has_permission(self, request, view):
        return request.user.is_superuser
```

Sobreescribiendo el método has_permission, definimos si un usuario tiene acceso o no al servicio web en función de que tenga activado el flag is_superuser.

3. Definición de rutas: Finalmente, es necesario definir también las URLs en las que contestará cada uno de los servicios web, y esto se definirá en el fichero urls.py de cada aplicación. Para el caso de la administración de especies, se deberán incluir las siguientes líneas en dicho fichero:

```
from rest_framework import routers
router = routers.DefaultRouter()
router.register(r'admin_especies', EspecieViewSet, base_name='admin_especies')
urlpatterns = [
    url(r'^api/', include(router.urls)),
]
```

La clase routers nos permitirá definir de forma automática cada una de las URLs sin necesidad de definirlas una a una.

Tras definir cada uno de los servicios web, se consultará la dirección local http://localhost:8000/api y el navegador mostrará el mapa de todos los servicios web disponibles, y se podrá navegar de forma intuitiva a través de los enlaces incluidos en los resultados:

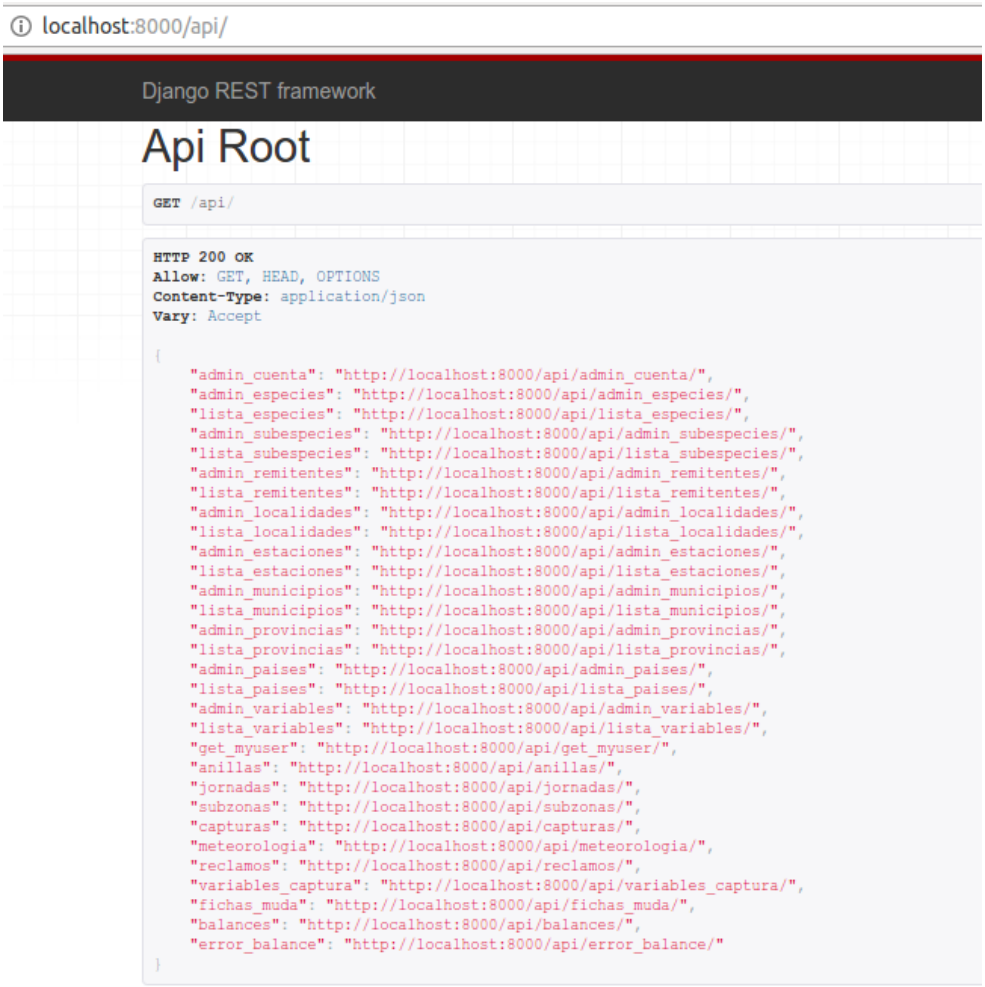


Figura 35: Mapa de endpoints

Para el ejemplo de las especies, se puede consultar la dirección http://localhost:8000/api/admin_especies/ y acceder a todas las opciones de que dispone este servicio web para la gestión de especies:

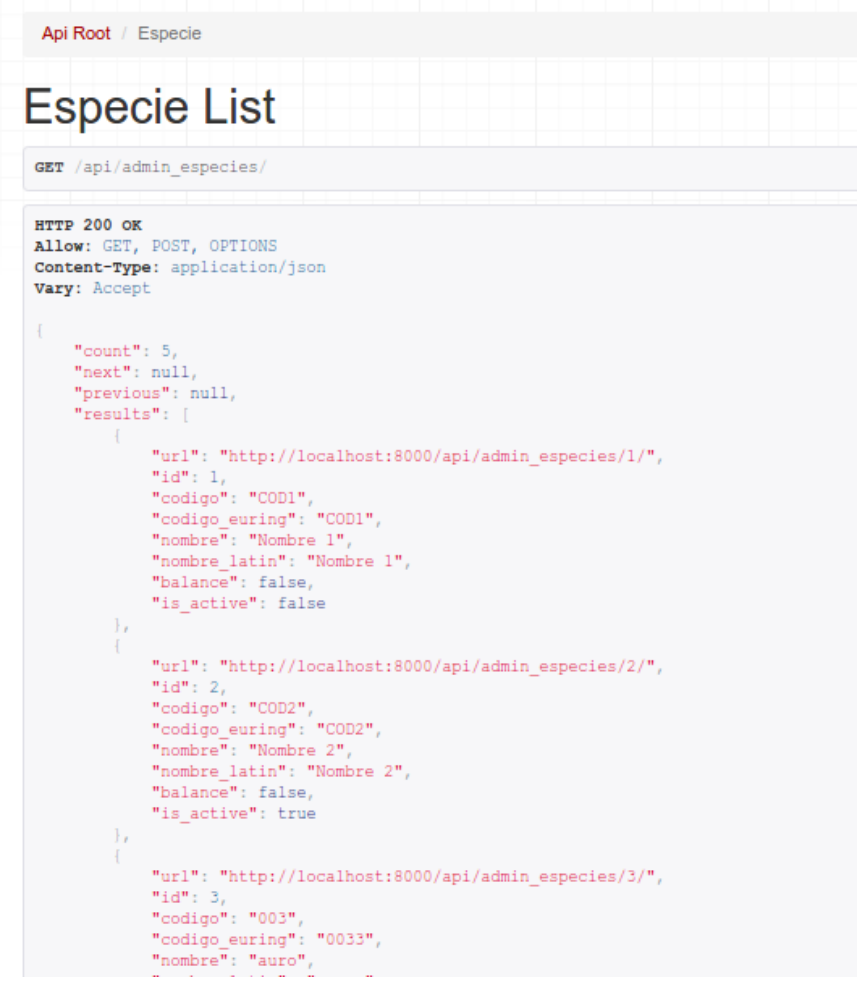


Figura 36: Webservice de administración de especies

Como se puede apreciar en la captura anterior, los resultados se devuelven en forma de lista y paginados:

- El campo “count” muestra el número de registros total recuperado.
- “next” es un enlace a la próxima página de resultados. En este caso, como el número de registros recuperados es menor que el número de resultados por página configurado (según se detallará en el Anexo E): Instalación y configuración de librerías externas), este campo está a nulo.
- “previous” es el enlace a la página de resultados anterior. En este caso, al haber sólo una página de resultados y ser la primera, este campo está a nulo.
- “results” muestra en forma de lista los resultados para la página actual.

Por cada uno de los resultados se muestra el campo “url”, que es un acceso directo a los detalles de la especie, en formato JSON. Si se pulsa sobre la URL correspondiente a la primera especie se puede navegar a los detalles:

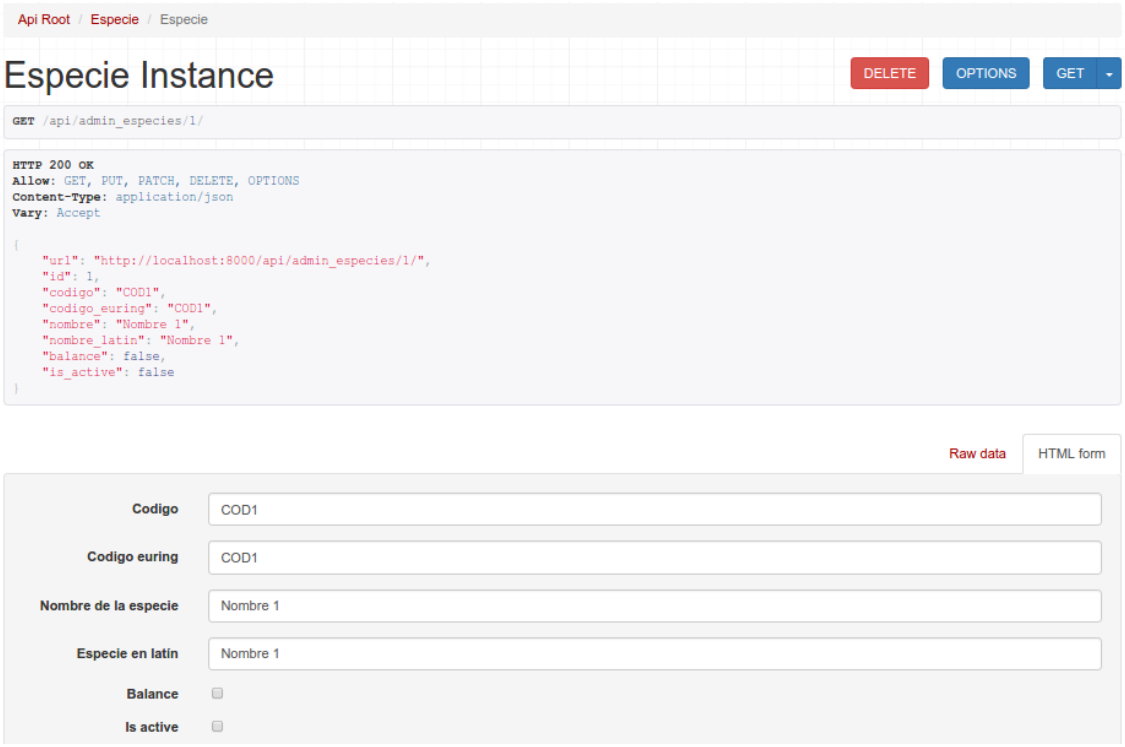


Figura 37: Detalle de especie con el browsable web API

Django Rest Framework, en su versión navegable, permite navegar a través de las distintas relaciones de los modelos y probar cada uno de los servicios programados, sin necesidad de utilizar ninguna herramienta adicional, como Postman *[4]*.

1.3. Librerías adicionales

Para poder implementar las funcionalidades referentes al ciclo de vida de cuentas de usuario, y a la autenticación según Oauth2, se han utilizado las librerías que se detallan a continuación.

Alta y gestión de usuarios

Para implementar el registro y activación de usuarios, junto con el restablecimiento de contraseña, se ha utilizado la librería externa Djoser. Esta librería también soporta autenticación por token basada en la funcionalidad básica de autenticación por token de Django, pero esta posibilidad tiene las siguientes limitaciones:

- No soporta autenticación mediante redes sociales, lo cual, en etapas posteriores del proyecto, será necesario incorporar.
- No soporta múltiples tokens por usuario, es decir, un usuario no puede tener sesiones abiertas en diferentes dispositivos, de forma independiente, sino que por cada autenticación del usuario el servidor mantiene un único token que sirve para todos los dispositivos.

Por estas razones, se ha descartado la funcionalidad de autenticación de la librería Djoser y sólo se utilizarán los siguientes servicios web que implementa:

Servicio web	Endpoint	Características

Registro de usuarios	/register/	Utilizado para la solicitud de alta en el sistema de los usuarios.
Activación de cuenta	/activate/	Éste no es un endpoint que se exponga directamente a los usuarios, sino que se enviará un enlace al correo electrónico de este usuario a este endpoint para que se realice la activación.
Cambio de e-mail	/email/	Utilizado para cambiar el campo de e-mail del usuario, que es el que identifica de forma unívoca su cuenta.
Solicitud de cambio de contraseña	/password/reset/	Se utiliza para solicitar el envío al e-mail del usuario de un correo que contendrá un link al servicio de cambio de contraseña, con todos los datos necesarios para que se lleve a cabo correctamente.
Confirmación de solicitud de cambio de contraseña	/password/reset/ confirm	Es un endpoint que no se expone directamente a los usuarios, sino que se envía como parte del correo del servicio web anterior. Este enlace tiene asociado un identificador de usuario y un token para que sólo el usuario que solicitó el cambio de contraseña la pueda cambiar.

Tabla 11: Endpoints Djoser

En el Anexo E se da una explicación pormenorizada de los pasos para instalar y configurar la librería Djoser en Django.

Autenticación

Para validar las credenciales de los usuarios se utilizará la librería externa Django OAuth Toolkit, cuya instalación y configuración se detallan en el Anexo E.

Aunque esta librería contiene muchísimas más funcionalidades, sólo se van a utilizar los siguientes servicios web:

Servicio web	Endpoint	Características
Autenticación	/o/token/	Utilizado para la autenticación inicial mediante e-mail y contraseña, y para refrescar posteriormente el token si ya ha caducado.
Activación de cuenta	/o/revoke_token/	Este endpoint es utilizado para eliminar el token de acceso en el lado del servidor. Esto sólo afecta al token cuya validez se solicita terminar, independientemente de que el usuario tenga otros dispositivos con otros tokens válidos.

Tabla 12: Endpoints Django Oauth Toolkit

Estos endpoints no se publican en el REST API navegable de Django Rest Framework, con lo que habría que tener desarrolladas las pantallas del frontend correspondientes para poder probar esta funcionalidad.

2. Front-end

En este apartado se da una visión de los desarrollos implicados en la implementación del frontend con Angular 4, así como el estado en el que quedan al finalizar este TFM.

2.1. Inventario

A continuación se listan los desarrollos de que consta el frontend.

A nivel de proyecto, se han modificado los siguientes ficheros:

- index.html: aquí se incluye el título de la página principal, así como los links a los ficheros css del tema Yeti utilizado (basado en Bootstrap) y Font Awesome.
- styles.css: en este fichero se han incluido estilos que afectan a todo el proyecto.

Módulo básico

MÓDULOS	
Nombre	**Descripción**
basic.module	Este módulo se encarga de realizar lo siguiente: - Declaración de los componentes propios del módulo. - Exportación de los componentes que podrán ser reutilizados por otros módulos. - Declaración y exportación de los servicios que podrán ser utilizados por los demás módulos. - Importación del módulo de enrutamiento basic-routing.module
basic-routing.module	Este módulo se encarga de definir la relación entre los componentes y las rutas en las que serán mostrados en el navegador. También establece si se utiliza algún guardián que realice cualquier tipo de comprobación antes de acceder a la ruta.
COMPONENTES	
Nombre	**Descripción**
navbar.component	En este componente se definen las opciones principales de navegación en el menú superior de la aplicación. Mostrará unas opciones u otras en función de si el usuario está o no autenticado, y de los privilegios de que disponga dicho usuario. El menú mostrará todas las opciones en su versión escritorio, y mostrará de forma compactada sólo algunas de ellas en la versión móvil. Este componente también se encargará de invocar al webservice de cierre de sesión, mediante la opción de menú correspondiente, y borrar finalmente las opciones del localstorage creadas en el proceso de login.

pagenotfound. component	A este componente se redigirá cualquier petición mediante una URL que no se encuentre en la lista de enrutamiento del módulo basic-routing.module.
servererror.component	Este componente se mostrará cuando se produzca cualquier error del tipo 40X proveniente del servidor.
home.component	A este componente se redirigirá cualquier petición a iocells.com, y constituirá el componente de inicio de la aplicación.
user-registration. component	Este componente se utiliza en combinación con el endpoint de registro de usuarios proporcionado por la librería Djoser de Django, y mediante el cual el usuario introducirá su e-mail y la contraseña elegida, y tras sumisión del formulario se mostrará cualquier error que se haya producido.
user-registration-confirmation.component	El usuario recibirá un e-mail con un enlace tras el registro de usuario. Pulsando sobre dicho enlace se invocará a este componente, que procesará la petición de confirmación de cuenta en combinación con el backend.
password-reset-request. component	Este componente permite la solicitud de nueva contraseña al backend, tras lo cual el usuario recibirá un e-mail con un nuevo enlace para restablecer su contraseña. Sólo será necesario introducir el e-mail del usuario que solicita la nueva contraseña.
password-reset. component	El restablecimiento de contraseña se llevará a cabo en este componente, introduciendo dos veces la nueva contraseña deseada.
login.component	Mediante este componente el usuario introducirá sus credenciales, las cuales se gestionarán en combinación con el backend. Cualquier error de autenticación se capturará en el lado del servidor y se mostrará al usuario en el navegador. Tras la autenticación exitosa del usuario, se almacenará en el localstorage del navegador el perfil del usuario (superusuario o usuario normal). Cualquier cambio que se produzca en los privilegios del usuario requerirá que haya de cerrar la sesión actual y autenticarse nuevamente para que se refresquen las opciones de navegación.
panel.component	A este componente se accederá por defecto tras la autenticación exitosa del usuario, y mostrará diferentes botones con iconos, cada uno correspondiente a un módulo de la aplicación. En este proyecto sólo se muestran tres módulos habilitados, los otros tres (Balance, Consultas, Importación/Exportación) quedan fuera del alcance de este TFM.
logout-confirmation.component	Éste será el componente encargado de confirmar la cierre de sesión exitosa, juntamente con la librería Djoser en el backend.
user-details.component	Este componente se utiliza para el mantenimiento de los datos básicos del usuario autenticado.
myuser.component	Formulario en el que se podrán modificar los datos del usuario, interactuando con el backend y mostrando los mensajes de error correspondientes.

changeemail.component	Este componente será invocado por el componente anterior mediante una ventana modal, y permitirá el cambio del campo clave del usuario, el e-mail. El usuario, para mayor seguridad, tendrá que introducir de nuevo sus credenciales de autenticación.
lov.component	Éste es un componente reutilizable por otros componentes y módulos, cuando se requiera una lista de valores asociada a un campo relacionado con otro modelo.
pagination.component	Componente reutilizable que muestra los controles de navegación para implementar la correcta paginación de registros devueltos por el backend.
search.component	Componente que implementa un campo de búsqueda con sus eventos asociados, reutilizable en otros componentes.
searchlist.component	Este componente utiliza los dos anteriores para implementar un componente de búsqueda de registros con paginación.
GUARDIANES	
Nombre	**Descripción**
auth.guard	Comprueba que el usuario actual esté autenticado, y en caso contrario redirigirá la navegación a la pantalla de login.
isnotloggedin.guard	Este guardián comprueba que el usuario actual no esté autenticado, y en caso de estarlo redigirá la navegación al panel principal de la aplicación.
MODELOS	
Nombre	**Descripción**
change_email.model	Representa el modelo utilizado en el formulario de cambio de e-mail.
password-reset.model	Representa el modelo utilizado en el formulario de restablecimiento de contraseña.
usuario.model	Representa el modelo utilizado en el formulario de mantenimiento de usuario.
SERVICIOS	
Nombre	**Descripción**
authentication.service	Gestiona la interacción con los servicios del backend de login, logout, restablecimiento de contraseña, confirmación de restablecimiento de contraseña, registro de usuario, cambio de e-mail, activación de usuario, almacenamiento y refresco de token, y recuperación de los datos del usuario actual.
users.service	Gestiona la interacción con los servicios del backend de consulta de lista de usuarios, de un usuario en particular, y actualización de usuario.
CLASES	
Nombre	**Descripción**
searchpaginationmultirow.class	Implementa las variables y métodos comunes utilizados por cualquier componente que utilice el componente searchlist.component.
INTERFACES	

Nombre	Descripción
pagination.interface	Este interfaz se utiliza en combinación con la clase searchpaginationmultirow.class para asegurar que el componente que extiende dicha clase define determinados parámetros de entrada necesarios por el componente searchlist.component.
CONFIGURACIÓN	
Nombre	**Descripción**
apis	Define una clase donde se recogen todos los endpoints del backend con los que interactuará este módulo.
parameters	Clase que agrupa los parámetros de paginación y la URL del backend, para el paso a producción.

Tabla 13: Desarrollos front-end módulo básico

Módulo de administración

MÓDULOS	
Nombre	**Descripción**
admin.module	Este módulo se encarga de ralizar lo siguiente: - Declaración de los componentes propios del módulo. - Exportación de los componentes que podrán ser reutilizados por otros módulos. - Declaración y exportación de los servicios que podrán ser utilizados por los demás módulos. - Importación del módulo de enrutamiento admin-routing.module
admin-routing.module	Este módulo se encarga de definir la relación entre los componentes y las rutas en las que serán mostrados en el navegador. También establece si se utiliza algún guardián que realice cualquier tipo de comprobación antes de acceder a la ruta.
COMPONENTES	
Nombre	**Descripción**
especiescontainer.component	Controla la navegación entre la búsqueda y listado de especies, y el formulario de especies.
especies.component	Búsqueda y listado de especies.
especieform.component	Formulario de mantenimiento y creación de especies.
estacionescontainer.component	Controla la navegación entre la búsqueda y listado de estaciones, y el formulario de estaciones.
estaciones. component	Búsqueda y listado de estaciones.
estacionform.component	Formulario de mantenimiento y creación de estaciones.
localidadescontainer.compon	Controla la navegación entre la búsqueda y listado de localidades, y el

ent	formulario de localidades.
localidades. component	Búsqueda y listado de localidades.
localidadform.component	Formulario de mantenimiento y creación de localidades.
municipioscontainer.component	Controla la navegación entre la búsqueda y listado de municipios, y el formulario de municipios.
municipios. component	Búsqueda y listado de municipios.
municipioform.component	Formulario de mantenimiento y creación de municipios.
paisescontainer.component	Controla la navegación entre la búsqueda y listado de países, y el formulario de países.
paises. component	Búsqueda y listado de países.
paisesform.component	Formulario de mantenimiento y creación de países.
paneladmin.component	Panel de navegación del módulo de administración.
provinciascontainer.component	Controla la navegación entre la búsqueda y listado de provincias, y el formulario de provincias.
provincias. component	Búsqueda y listado de provincias.
provinciaform.component	Formulario de mantenimiento y creación de provincias.
remitentescontainer.component	Controla la navegación entre la búsqueda y listado de remitentes, y el formulario de remitentes.
remitentes. component	Búsqueda y listado de remitentes.
remitenteform. component	Formulario de mantenimiento y creación de remitentes.
subespeciescontainer.component	Controla la navegación entre la búsqueda y listado de subespecies, y el formulario de subespecies.
subespecies. component	Búsqueda y listado de subespecies.
subespecieform. component	Formulario de mantenimiento y creación de subespecies.
users.component	Búsqueda y listado de usuarios.
userform.component	Formulario de mantenimiento de usuarios.
especielov.component	Componente que implementa una lista de valores para el modelo de especie.
localidadlov.component	Componente que implementa una lista de valores para el modelo de localidad.
municipiolov.component	Componente que implementa una lista de valores para el modelo de municipio.
paislov.component	Componente que implementa una lista de valores para el modelo de pais.
provincialov.component	Componente que implementa una lista de valores para el modelo de provincia.

GUARDIANES	
Nombre	**Descripción**
superuser.guard	Este guardián comprueba que el usuario que realiza la petición tiene los privilegios de superusuario correspondientes.
MODELOS	
Nombre	**Descripción**
especie.model	Representa el modelo utilizado en el formulario de especie.
estaciones.model	Representa el modelo utilizado en el formulario de estación.
localidad.model	Representa el modelo utilizado en el formulario de localidad.
municipio.model	Representa el modelo utilizado en el formulario de municipio.
pais.model	Representa el modelo utilizado en el formulario de país.
provincia.model	Representa el modelo utilizado en el formulario de provincia.
remitente.model	Representa el modelo utilizado en el formulario de remitente.
subespecie.model	Representa el modelo utilizado en el formulario de subespecie.
SERVICIOS	
Nombre	**Descripción**
especies.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de especie.
estaciones.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de estación.
localidades.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de localidad.
municipios.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de municipio.
paises.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de país.
provincias.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de provincia.
remitentes.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de remitente.
subespecies.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de subespecie.
CONFIGURACIÓN	
Nombre	**Descripción**
apis	Define una clase donde se recogen todos los endpoints del backend con los que interactuará este módulo.

Tabla 14: Desarrollos front-end módulo administración

Módulo de anillamiento

MÓDULOS	
Nombre	**Descripción**
anilla.module	Este módulo se encarga de ralizar lo siguiente: - Declaración de los componentes propios del módulo. - Exportación de los componentes que podrán ser reutilizados por otros módulos. - Declaración y exportación de los servicios que podrán ser utilizados por los demás módulos. - Importación del módulo de enrutamiento anilla-routing.module
anilla-routing.module	Este módulo se encarga de definir la relación entre los componentes y las rutas en las que serán mostrados en el navegador. También establece si se utiliza algún guardián que realice cualquier tipo de comprobación antes de acceder a la ruta.
COMPONENTES	
Nombre	**Descripción**
panelanilla.component	Panel de navegación del módulo de anillamiento.
anillas.component	Mantenimiento y creación de anillas por año.
jornadas.component	Este componente se encarga de gestionar la búsqueda, mantenimiento y creación de jornadas de anillamiento, así como de enlazar con los detalles de subzonas, meteorología y reclamos.
jornada-form.component	Formulario de mantenimiento y creación de jornadas.
meteorologia-form.component	Formulario de mantenimiento y creación de variables meteorológicas.
subzonas.component	Gestión de subzonas de anillamiento.
subzona-form.component	Formulario de mantenimiento y creación de subzonas.
reclamos.component	Gestión de subzonas de reclamos.
reclamo-form.component	Formulario de mantenimiento y creación de reclamos.
capturas.component	Componente que gestiona la búsqueda, creación y mantenimiento de capturas, así como la navegación a los detalles de fichas de muda y variables personalizadas.
captura-form.component	Formulario de mantenimiento y creación de capturas.
fichas-muda.component	Formulario de mantenimiento y creación de fichas de muda.
variables-form.component	Formulario de mantenimiento y creación de variables personalizadas.
MODELOS	
Nombre	**Descripción**
anilla.model	Representa el modelo utilizado en el formulario de anilla.
jornada.model	Representa el modelo utilizado en el formulario de jornada.
subzona.model	Representa el modelo utilizado en el formulario de subzona.
meteorologia.model	Representa el modelo utilizado en el formulario de meteorología.

reclamo.model	Representa el modelo utilizado en el formulario de reclamo.
captura.model	Representa el modelo utilizado en el formulario de captura.
variable.model	Representa el modelo utilizado en el formulario de variable.
ficha-muda.model	Representa el modelo utilizado en el formulario de ficha de muda.
SERVICIOS	
Nombre	**Descripción**
anillas.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de anilla.
jornadas.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de jornada.
subzonas.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de subzona.
meteorologia.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de meteorología.
reclamos.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de reclamo.
capturas.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de captura.
variables.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de varibale.
fichas-muda.service	Gestiona la interacción de los servicios CRUD del backend, del modelo de ficha de muda.
CONFIGURACIÓN	
Nombre	**Descripción**
apis	Define una clase donde se recogen todos los endpoints del backend con los que interactuará este módulo.

Tabla 15: Desarrollos front-end módulo anillamiento

2.2. Estado

En este apartado se realiza una estimación del grado de avance de los bloques de que se compone el proyecto con Angular 4.

Criterios

A continuación se detallan los criterios establecidos para determinar el estado de cada tipo de desarrollo, para poder así establecer el avance en términos de porcentaje:

MÓDULOS		
#	**Criterio**	**Descripción**
1	Programación	Definición del módulo en lenguaje Typescript
COMPONENTES		

#	Criterio	Descripción
1	Programación	Codificación de la lógica del componente
2	Plantilla HTML y CSS	Marcado de la plantilla HTML del componente y estilos CSS
3	Utilización de servicios	Integración con REST API
4	Tratamiento de errores	Tratamiento de errores con el servidor remoto y de interacción con el usuario.
5	Usabilidad	Aplicación de los criterios de usabilidad para la mejor experiencia del usuario.
6	Pruebas	Pruebas unitarias del componente.
GUARDIANES		
#	**Criterio**	**Descripción**
1	Programación	Definición del módulo en lenguaje Typescript
2	Pruebas	Pruebas unitarias del desarrollo
MODELOS		
#	**Criterio**	**Descripción**
1	Programación	Definición del módulo en lenguaje Typescript
SERVICIOS		
#	**Criterio**	**Descripción**
1	Programación	Definición del módulo en lenguaje Typescript
2	Pruebas	Pruebas unitarias del desarrollo
CLASES		
#	**Nombre**	**Descripción**
1	Programación	Definición del módulo en lenguaje Typescript
INTERFACES		
#	**Criterio**	**Descripción**
1	Programación	Definición del módulo en lenguaje Typescript
CONFIGURACIÓN		
#	**Criterio**	**Descripción**
1	Programación	Definición del módulo en lenguaje Typescript

Tabla 16: Criterios para medir el avance de los desarrollos

NOTA: para tener una visión más aproximada del avance del proyecto, se ponderará el peso de cada desarrollo atendiendo al número de criterios establecidos para determinar su estado. Así, por ejemplo, un componente tendrá un peso de 6, mientras que un interfaz tendrá un peso de 1.

Módulo básico

El grado de avance de este módulo es del **100%**.

Módulo de administración

El grado de avance de este módulo es del **100%**.

Módulo de anillamiento

MÓDULOS						
Nombre	**Criterios**					
	Programación					
anilla.module	100%					
anilla-routing.module	100%					
COMPONENTES						
Nombre	**Criterios**					
	Program.	**HTML**	**Servicios**	**Errores**	**Usabilid.**	**Pruebas**
panelanilla.component	50%	90%	0%	0%	50%	20%
anillas.component	50%	90%	0%	0%	50%	20%
jornadas.component	50%	90%	0%	0%	50%	20%
jornada-form.component	50%	90%	0%	0%	50%	20%
meteorologia-form.component	50%	90%	0%	0%	50%	20%
subzonas.component	50%	90%	0%	0%	50%	20%
subzona-form.component	50%	90%	0%	0%	50%	20%
reclamos.component	50%	90%	0%	0%	50%	20%
reclamo-form.component	50%	90%	0%	0%	50%	20%
capturas.component	50%	90%	0%	0%	50%	20%
captura-form.component	50%	90%	0%	0%	50%	20%
fichas-muda.component	50%	90%	0%	0%	50%	20%
muda-form.component	50%	90%	0%	0%	50%	20%
variables-form.component	50%	90%	0%	0%	50%	20%
MODELOS						
Nombre	**Criterios**					
	Programación					
anilla.model	0%					
jornada.model	0%					
subzona.model	0%					
meteorologia.model	0%					
reclamo.model	0%					
captura.model	0%					
variable.model	0%					
ficha-muda.model	0%					
SERVICIOS						
Nombre	**Criterios**					
	Programación			**Pruebas**		
anillas.service	0%			0%		

jornadas.service	0%	0%
subzonas.service	0%	0%
meteorologia.service	0%	0%
reclamos.service	0%	0%
capturas.service	0%	0%
variables.service	0%	0%
fichas-muda.service	0%	0%
CONFIGURACIÓN		
Nombre	**Criterios**	
	Programación	
apis	0%	

Tabla 17: Avance front-end módulo anillamiento

Por tanto, el grado de avance de este módulo es del **28,29%**.

2.3. Decisiones tomadas

Decisiones de desarrollo

Las decisiones de desarrollo más importantes tomadas durante el desarrollo del frontend han sido las siguientes:

- **Decisión 1**: En cuanto al control de errores en formularios, se ha decidido llevar la carga de las comprobaciones al backend, de forma que sea éste quien realice todas las validaciones y devuelva al fronend un array con todos los errores que se han producido. Esto se ha decidido de esta forma para que el peso de las traducciones de dichos mensajes de error recaiga también en el backend, que dispone de más mecanismos para poder traducir los valores textuales a diferentes idiomas.
- **Decisión 2**: El uso de las pantallas modales se ha restringido a determinados casos, como la visualización de resúmenes o formularios sencillos. La razón de esta decisión se fundamenta en la restricción de la librería Bootstrap de mostrar más de un nivel de ventanas modales, lo que hace que sea más adecuado utilizar modales en caso de un sólo nivel de detalle.
- **Decisión 3**: Se ha decidido que, aunque los privilegios del usuario cambien durante una sesión, éstos se recuperen en el momento de inicio de sesión. El usuario debería, de este modo, reiniciar sesión para poder ver los cambios en el menú de navegación.
- **Decisión 4**: Se permite a superusuarios acceder al menú de administración, sin tener habilitado el flag de staff, lo cual habilitaría también el acceso al backoffice de Django.
- **Decisión 5**: Se ha decidido utilizar también la librería jQuery para mejorar determinados aspectos de navegación en los formularios. Aunque Angular 4 dispone de muchas utilidades para esto, aún no tiene disponibles tantas librerías externas para poder conseguir determinados efectos.
- **Decisión 6**: Aunque el volumen de la aplicación aún no es tan grande como para cargar las diferentes partes según el método Lazy loading, es una buena práctica definir desde el

principio las distintas partes de que va a constar y prever un posible crecimiento según módulos.

- **Decisión 7**: Aunque existen diferentes extensiones en el gestor npm que implementan la librería Bootstrap para Angular 4, se ha decidido incluir los ficheros CSS y JavaScript necesarios con enlaces en el fichero index.html a los CDNs correspondientes, para poder realizar de esta forma pruebas de forma más sencilla con diferentes temas y librerías.

Decisiones de diseño y usabilidad

- **Decisión 1**: Acorde con la planificación inicial de este TFM, se ha priorizado el diseño para dispositivos móviles, aunque se ha intentado en la medida del tiempo disponible establecer el layout para la versión escritorio de la aplicación.
- **Decisión 2**: Aunque no se vayan a desarrollar los módulos de Balance, Consultas y Importación/Exportación, se ha decidido incluirlos en el panel principal de la aplicación como referencia para futuras fases del proyecto, y para mostrar también una mejor idea del volumen de trabajo total necesario para completar el proyecto.
- **Decisión 3**: La forma de implementar los campos relacionados con otros modelos ha sido mediante campos con listas de valores asociadas. Esta es una posible mejora para el futuro ya que Angular 4 permite la utilización de determinados mecanismos para mostrar valores coincidentes al tiempo que el usuario introduce valores en un campo. Posiblemente la solución futura utilizará una combinación de ambas.
- **Decisión 4**: Se restringe el acceso al usuario a la parte activa de la aplicación (home, login…) si está autenticado.
- **Decisión 5**: Para introducción de datos más intuitiva en los formularios complejos, se ha decidido introducir bloques expandibles que oculten o muestren parte de dicho formulario.
- **Decisión 6**: Se ha decidido utilizar los estilos por defecto del tema Yeti, de Bootswatch, bajo licencia MIT.
- **Decisión 7**: Utilización de iconos en botones para una navegación más intuitiva y mejorar así la experiencia del usuario.
- **Decisión 8**: Para la versión móvil de determinadas pantallas se han ocultado campos no determinantes para una correcta identificación de los objetos por parte del usuario.

3. Despliegue

Para la puesta en producción de este proyecto, se parte de un servidor VPS contratado en www.ovh.es, en el que se han instalado previamente las herramientas Webmin (para una gestión más amigable del entorno), CSF (como firewall), Apache (servidor web) y PostgreSQL (gestor de bases de datos). Estas instalaciones quedan fuera del alcance de este proyecto.

Tanto el frontend como el backend del sistema se van a instalar en el mismo servidor VPS, aunque se va a utilizar una configuración que sería también válida en el caso de que cada parte se instalase en servidores diferentes.

Se requieren dos dominios para este proyecto: uno para el backend (Django) y otro para el frontend (Angular 4). Estos 2 dominios se contratan, para este proyecto, en la compañía de hosting www.ovh.es :

- iocells.xyz: para el backend
- iocells.com: para el frontend

La razón por la que se necesitan dos dominios es para poder configurar correctamente las restricciones del estándar CORS en Django.

En el panel de administración de OVH, se modifica el registro A de la zona DNS de ambos dominios para que apunten a la IP del servidor VPS:

Dominio	TTL	Tipo	Destino
Puede añadir un nuevo registro haciendo clic en el botón de la derecha.			
iocells.com.	0	NS	dns111.ovh.net.
iocells.com.	0	NS	ns111.ovh.net.
iocells.com.	0	MX	1 redirect.ovh.net.
_autodiscover._tcp.iocells.com.	0	SRV	0 0 443 mailconfig.ovh.net.
_imaps._tcp.iocells.com.	0	SRV	0 0 993 ssl0.ovh.net.
_submission._tcp.iocells.com.	0	SRV	0 0 465 ssl0.ovh.net.
iocells.com.	0	A	213.32.21.190
autoconfig.iocells.com.	0	CNAME	mailconfig.ovh.net.
autodiscover.iocells.com.	0	CNAME	mailconfig.ovh.net.
ftp.iocells.com.	0	CNAME	iocells.com.

1 2 Ver Página 1/2 OK

Figura 38: Configuración DNS

(Se hace lo mismo para el dominio iocells.xyz)

Además se crea una dirección de correo electrónico con la que se configurarán las opciones de correo en el fichero settings.py: klugapps@gmail.com

Es preciso habilitar esta cuenta de correo para que se pueda utilizar por aplicaciones menos seguras, en el siguiente enlace, una vez en la cuenta de correo correspondiente:

https://myaccount.google.com/lesssecureapps

Todas las operaciones realizadas para el despliegue se detallan en el Anexo J.

Capítulo 5: Demostración

En este capítulo se muestra la navegación por las distintas pantallas de la aplicación que se han desarrollado.

1. Módulo básico

En un primer esquema se muestra la navegación en una parte de este módulo:

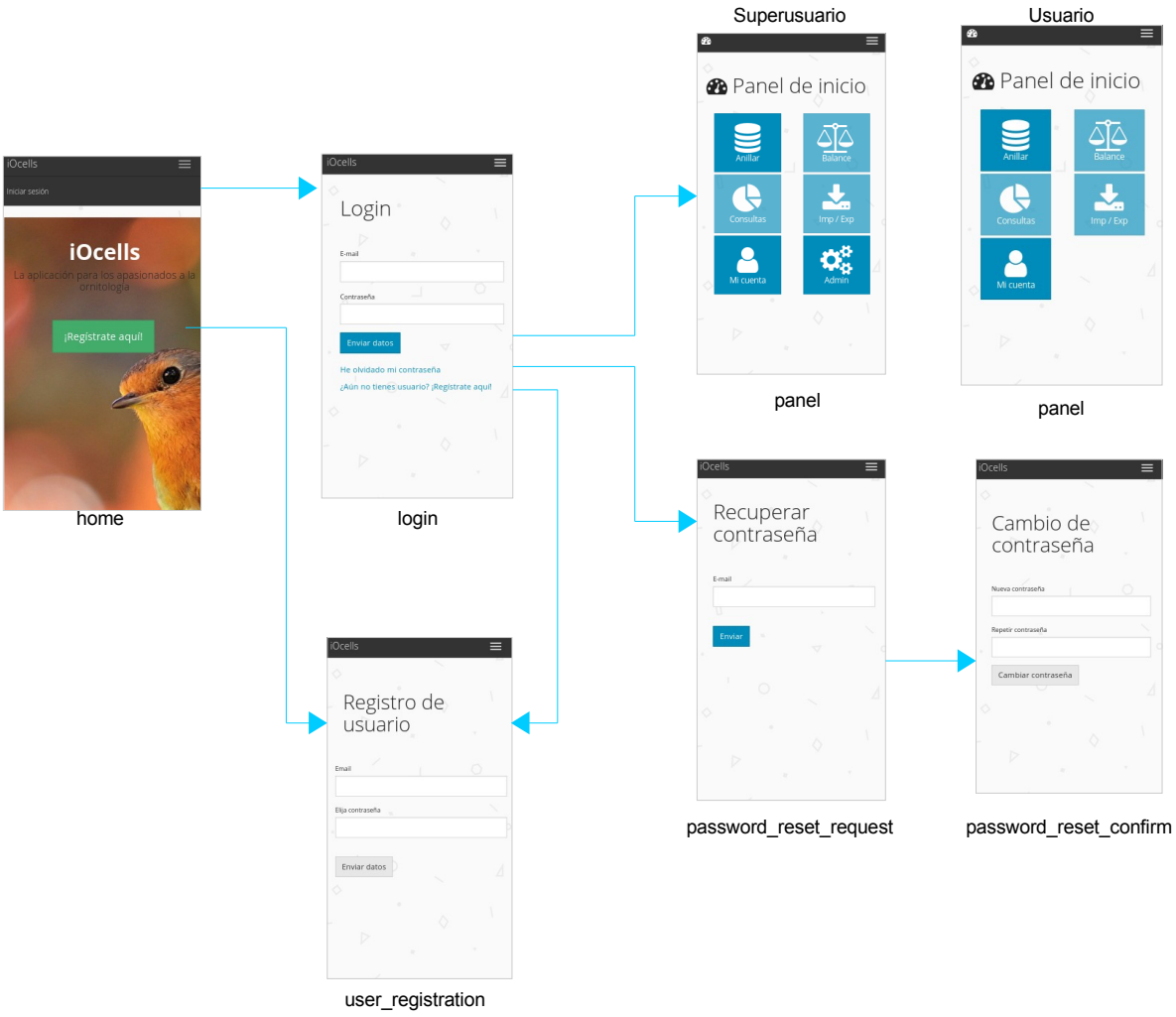


Figura 39: Navegación módulo básico

Se parte de la página “home”, desde la cual se puede ir a la página de registro de usuario o la de login. Siguiendo el flujo de login, se puede acceder también a la pantalla de solicitud de restablecimiento de contraseña, a la de confirmación de nueva contraseña, o también enlazar con la pantalla de registro de usuario.

También pertenece a este módulo el submenú “Mi cuenta”, cuya navegación se muestra a continuación:

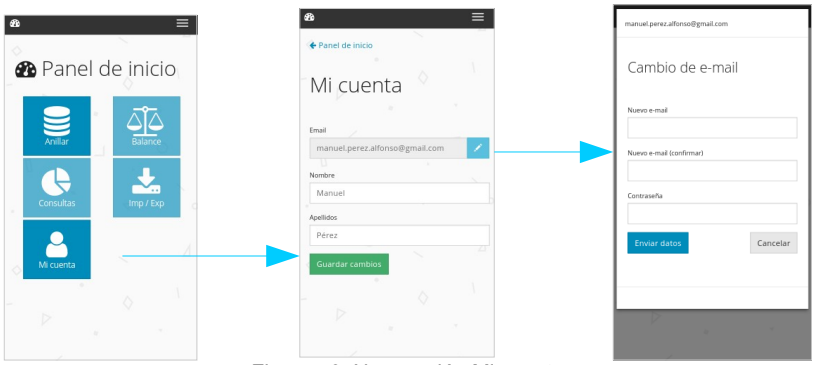


Figura 40: Navegación Mi cuenta

Este submenú lleva a la pantalla de mantenimiento de datos de usuario, que enlaza a su vez con la pantalla de cambio de e-mail.

En el Panel de Inicio se muestran como deshabilitados los botones que acceden a las funcionalidades fuera del alcance de este TFM.

2. Módulo de administración

Desde el panel de inicio, se accede al panel de administración, con una entrada por cada una de las pantallas de mantenimiento de datos maestros:

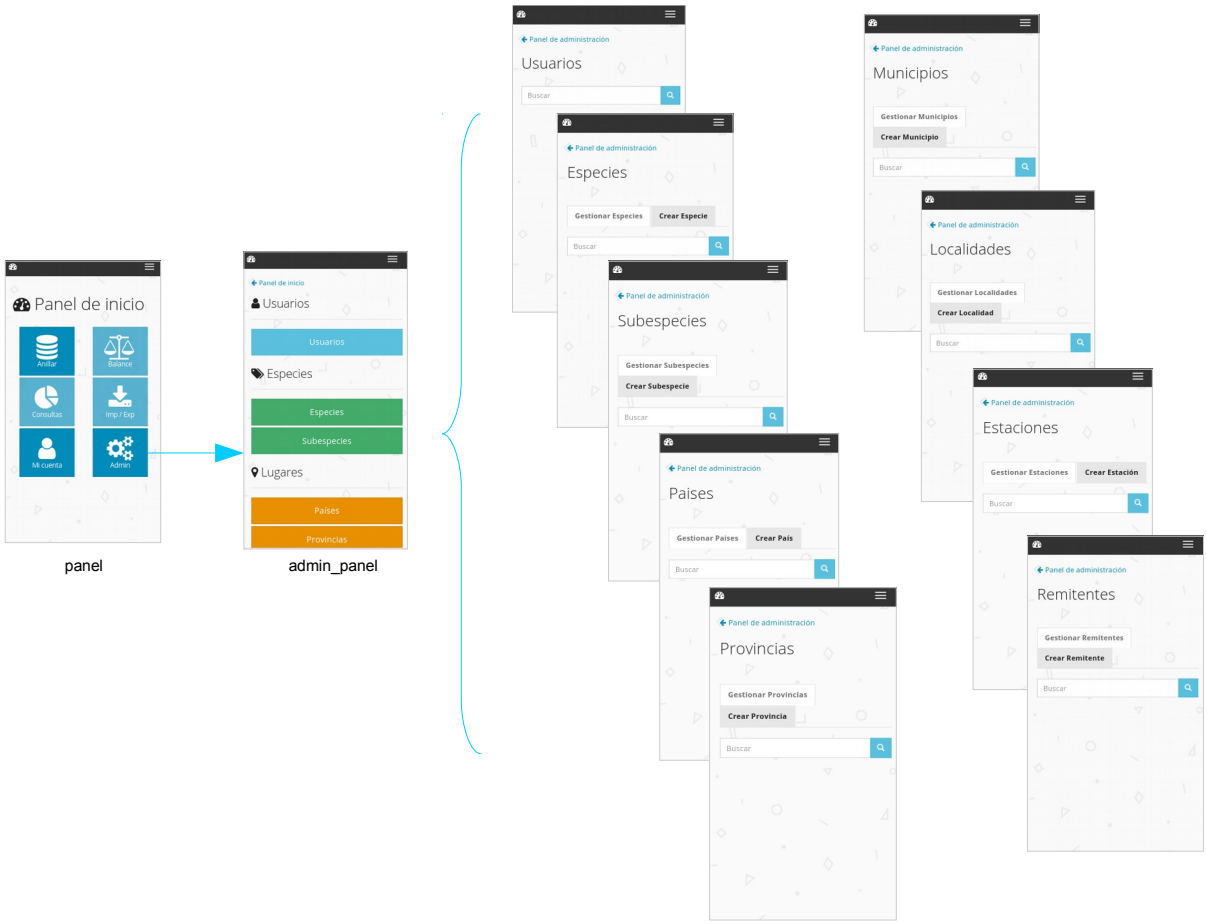


Figura 41: Navegación módulo administración

Cada pantalla de mantenimiento consta de dos pestañas: una para buscar y modificar datos existentes, y otra para crear nuevos registros.

3. Módulo de anillamiento

Desde el Panel de Inicio se accede al Panel de anillamiento, el cual da acceso a los submenús de Anillas, Jornadas, Capturas y Fichas de muda:

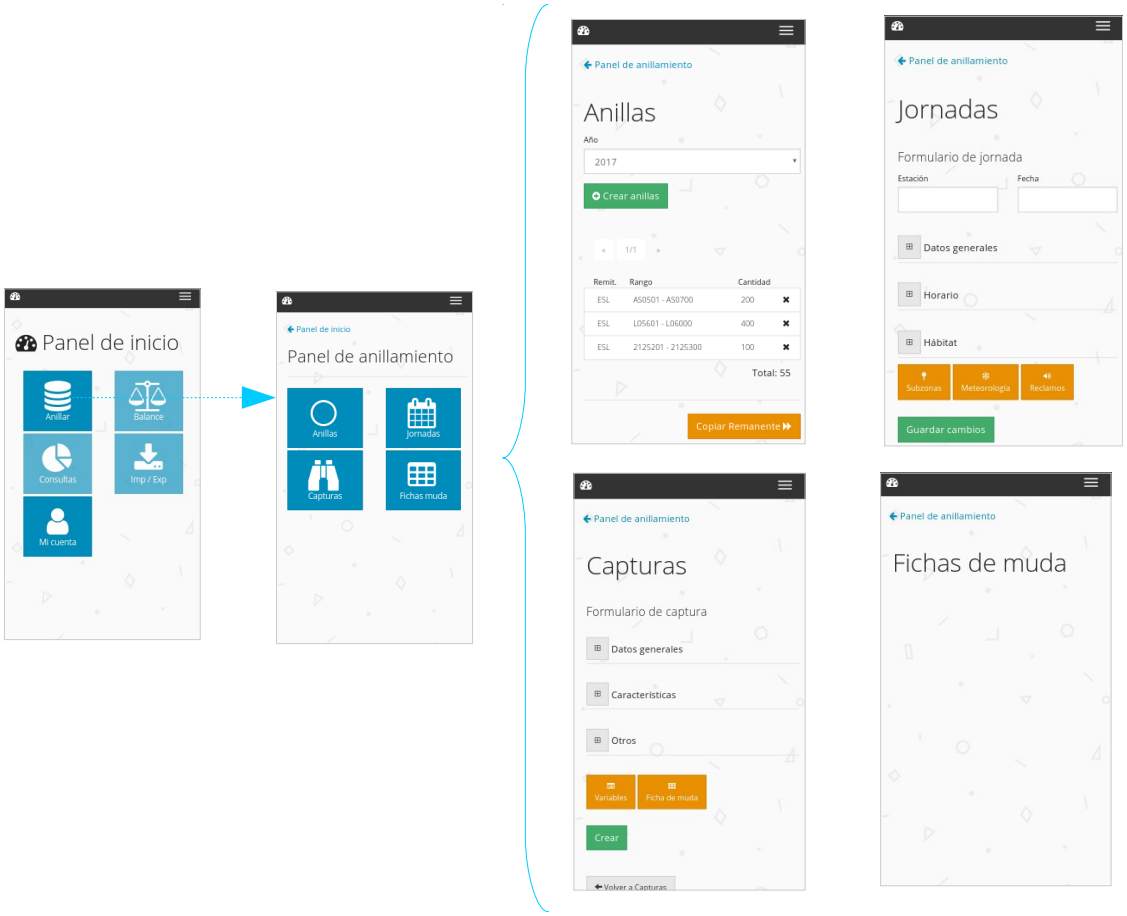


Figura 42: Navegación módulo anillamiento

A su vez, desde la pantalla de Jornadas se puede acceder a Subzonas, Meteorología y Reclamos:

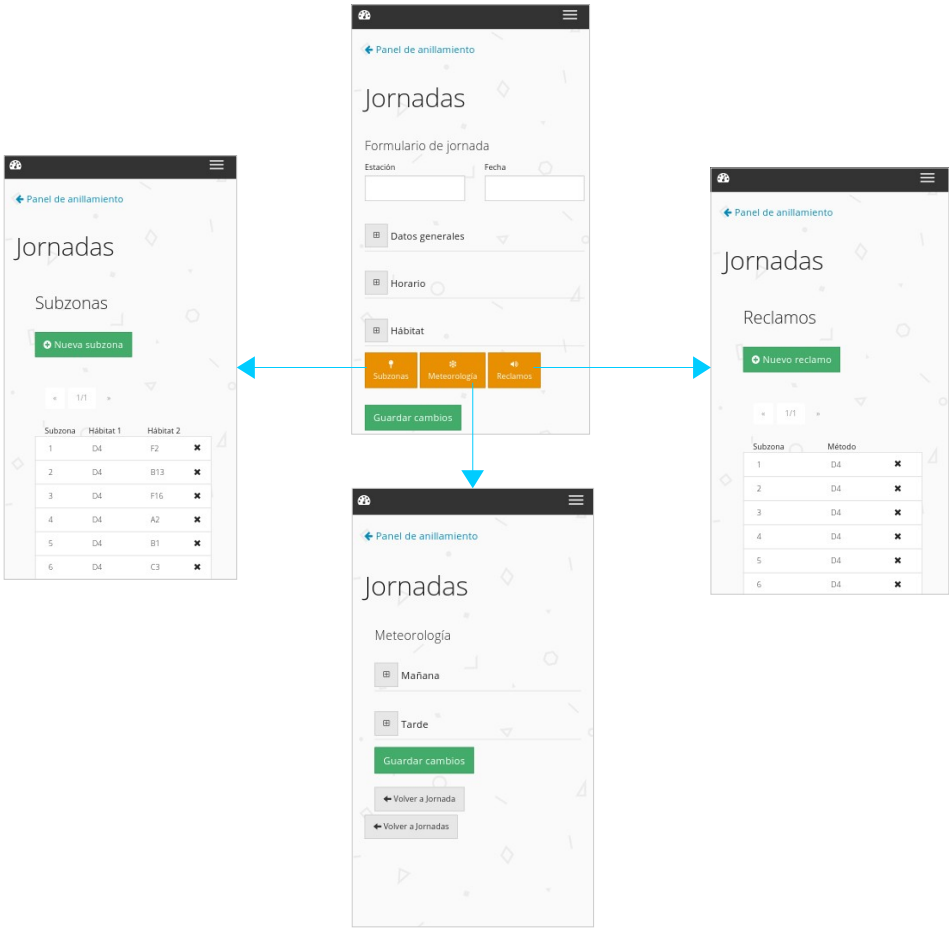


Figura 43: Navegación jornadas

Del mismo modo, desde la pantalla de Capturas se accede a las pantallas de Variables y a la de Ficha de muda:

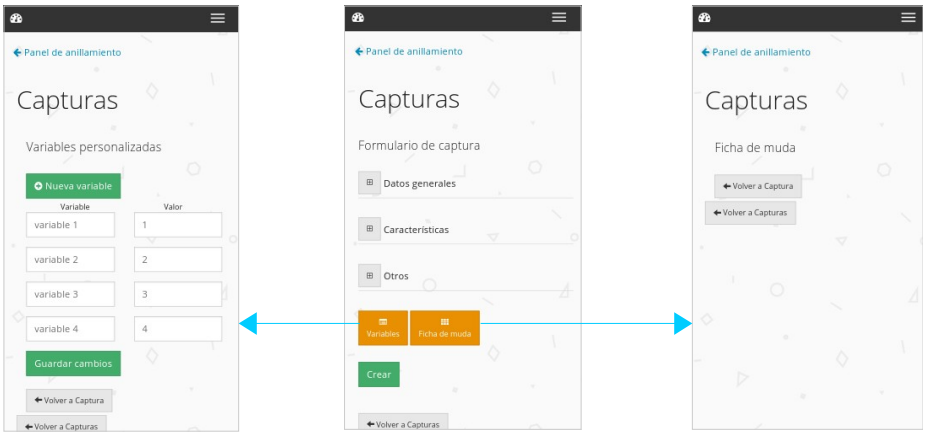


Figura 44: Navegación capturas

Capítulo 6: Conclusiones y líneas de futuro

1. Conclusiones

Las conclusiones más importantes que se han podido extraer en la realización de este proyecto son las siguientes:

- El ámbito del desarrollo de aplicaciones (tanto web, híbridas o nativas) es un área en constante desarrollo en el que es necesario estar constantemente informado de las nuevas tendencias que surgen. También es muy importante, una vez se escoge una determinada tecnología, conocer las características de la versión a utilizar, y sobre todo saber el tipo de soporte que se establece para la versión (es más conveniente utilizar una versión LTS, Long Term Support, que una que no lo es). En el desarrollo del presente proyecto han surgido nuevas versiones de Django (versión 1.11, LTS) y Angular (versión 4) que, aunque no han supuesto rehacer el trabajo realizado para poder utilizar las últimas características de estos frameworks, podrían haber supuesto un riesgo para el desarrollo del proyecto.
- Aunque se ha realizado un primer esfuerzo para poder definir gráficamente el front-end, el trabajo realizado en el back-end no se ha trasladado completamente al potencial usuario, ya que el front-end no se ha podido desarrollar en su totalidad. Habría sido más apropiado desarrollar el front-end en su totalidad, con fuentes de datos de prueba no remotas, para que los potenciales usuarios hubiesen tenido una percepción más concreta de las funcionalidades que ofrece la aplicación, y más tarde sustituir esas fuentes de datos de prueba por servicios web que conectasen con el back-end.
- Los frameworks utilizados, Django y Angular 4, son frameworks muy complejos para los que es necesario dedicar mucha dedicación si se pretende conocer todas sus funcionalidades y saber en cada momento qué técnica o método utilizar, ya que disponen de diferentes posibilidades para llevar a cabo la misma tarea. Es por ello que, muy posiblemente, determinadas partes del sistema se rehagan en futuras fases del proyecto, bien porque la experiencia ha demostrado que existen formas más escalables de realizarlas, o bien simplemente porque se carecía de este conocimiento al empezar el proyecto.
- El proceso de despliegue del sistema puede llegar a ser tan complejo como el desarrollo de cualquiera de las partes de que se compone, y es por tanto algo a tener en cuenta desde el inicio propio del proyecto, ya que determinadas decisiones de despliegue pueden tener incidencia en el proceso de desarrollo.

2. Líneas de futuro

A continuación se destacan las principales líneas de desarrollo en el futuro más cercano a este proyecto, que tratarían de perfeccionar determinados aspectos que, por cuestiones de planificación, han debido ser pospuestas.

Back-end

A continuación se citan de forma breve puntos de mejora o desarrollos pendientes en la parte de back-end que se abordarán en futuras fases del proyecto. Aun fuera del alcance de este proyecto, se citan aquí para agilizar la realización de las siguientes fases:

- Separación de los serializadores, vistas y URLs de los webservices para el módulo "customuser" y el módulo "anilla".
- Creación de endpoints específicos de sólo lectura y con los campos mínimos, para todas aquellas partes del frontend que requieran listados de objetos.
- Rediseño de las vistas de los servicios web para que eviten la utilización de la clase ModelViewSet, que implementa todos los métodos posibles. Es necesario acotar los métodos mínimos que una vista necesita utilizar para evitar posibles riesgos de seguridad.
- Traducción de los mensajes proporcionados por la librería Oauth2. Esta librería no implementa por defecto la traducción de los mensajes de error, con lo que se hace necesario copiarla como módulo local del proyecto y modificar aquellas partes del código susceptibles de ser traducidas. A continuación, es necesario aplicar la funcionalidad de Django para traducción de textos.
- Completar la configuración del backoffice de Django, con el alta de todos los modelos creados y la instalación de librerías adicionales como Grappelli, para la mejora de determinados aspectos de usabilidad de este backoffice, útil en determinadas ocasiones para realizar labores de mantenimietno de datos.
- Utilización de librerías adicionales para poder disponer de traducciones en los textos almacenados en la base de datos. Posibilidades: django-parler con django-parler-test, django-hvad, django-modeltranslation.

Front-end

A continuación se citan de forma breve los principales puntos de mejora o desarrollos pendientes en la parte de frontend, que se abordarán en futuras fases del proyecto:

- Traducción de los diferentes textos particulares a cada idioma, mediante la funcionalidad de traducción por defecto de Angular 4.
- En las pantallas de mantenimiento, se pretende habilitar el botón de guardar cambios sólo cuando se haya producido un cambio en los datos recuperados.
- En el ciclo de cambio de e-mail, se pretende que en un futuro se envíe un e-mail a la antigua dirección de correo electrónico introducida por el usuario, avisando de que se ha producido un cambio, y requiriendo de nuevo confirmación.
- Introducir campos adicionales de búsqueda, adicionalmente al campo de búsqueda genérico (buscar por registros activos/inactivos, rangos de fechas, …).
- Mejorar la navegabilidad en las pantallas de anillamiento, en la navegación de hasta tres niveles de detalle.

Bibliografía

[1] **Angular** “Docs” [documentación en línea].
<https://angular.io/docs/ts/latest/>
[2] **Aranzadi** “Oficina de anillamiento” [artículo en línea]. [Fecha de consulta: 22 de marzo del 2017].
<http://www.aranzadi.eus/ornitologia-oficina-de-anillamiento>
[3] **Coding Entepreneurs** “Blog API with Django Rest Framework” [lista de reproducción en línea]
<https://www.youtube.com/playlist?list=PLEsfXFp6DpzTOcOVdZF-th7BS_GYGguAS>
[4] **DigitalOcean** “¿Cómo instalar Node.js en Ubuntu 16.04?” [artículo en línea]
[Fecha de consulta: 29 de abril del 2017]
<https://www.digitalocean.com/community/tutorials/como-instalar-node-js-en-ubuntu-16-04-es>
[5] **DigitalOcean** “How to Install the Django Web Framework on Ubuntu 14.04” [artículo en línea]. [Fecha de consulta: 29 de abril del 2017]
<https://www.digitalocean.com/community/tutorials/how-to-install-the-django-web-framework-on-ubuntu-14-04>
[6] **DigitalOcean** “How To Secure Apache with Let’s Encrypt on Ubuntu 16.04”[artículo en línea]. [Fecha de consulta: 29 de abril de 2017]
<https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-16-04>
[7] **Django Rest Framework** “API Guide”
<http://www.django-rest-framework.org/>
[8] **Evonove** “Django OAuth Toolkit” [documentación en línea]
<https://github.com/evonove/django-oauth-toolkit>
[9] **fulcrum** “Bird Ringing Notes” [artículo en línea]. [Fecha de consulta: 22 de marzo del 2017].
<http://www.fulcrumapp.com/apps/bird-ringing-notes>
[10] **Google Play** “Aves de España” [artículo en línea]. [Fecha de consulta: 22 de marzo del 2017].
<https://play.google.com/store/apps/details?id=com.alborgis.seo>
[11] **Google Play** “BIRDPAPER by SEOBirdLife” [artículo en línea]. [Fecha de consulta: 22 de marzo del 2017].
<https://play.google.com/store/apps/details?id=org.seo.seo_birdpaper>
[12] **Google Play** “BirdRing” [artículo en línea]. [Fecha de consulta: 22 de marzo del 2017].
<https://play.google.com/store/apps/details?id=nl.birdring>
[13] **Google Play** “BirdTrack” [artículo en línea]. [Fecha de consulta: 22 de marzo del 2017].
<https://play.google.com/store/apps/details?id=org.bto.btapp>
[14] **Google Play** “BTO Ringers Info” [artículo en línea]. [Fecha de consulta: 22 de marzo del 2017].
<https://play.google.com/store/apps/details?id=bto.android.ringing.info>
[15] **Google Play** “Merlin Bird ID de Cornell Lab” [artículo en línea]. [Fecha de consulta: 22 de marzo del 2017].
<https://play.google.com/store/apps/details?id=com.labs.merlinbirdid.app>

[16] **Institut Català d'Ornitologia** "NouBioPro09 - Gestió de dades d'anellament" [artículo en línea]. [Fecha de consulta: 22 de marzo del 2017].
<http://www.ornitologia.org/ca/quefem/anellament/noubiopro09.html>
[17] **Postman** "Postman" [documentación en línea]
<https://www.getpostman.com/>
[18] **SEO/BirdLife** (2016, 11 de enero) "Nuevas `apps´ para hacer ciencia con SEO/BirdLife" [artículo en línea]. [Fecha de consulta: 1 de marzo del 2017].
<http://www.seo.org/2016/01/11/nuevas-apps-para-hacer-ciencia-con-seobirdlife/>
[19] **StackOverflow** "404 error after configured mod_wsgi on Apache2 for django deployment" [foro de discusión en línea]
<http://stackoverflow.com/questions/29825745/404-error-after-configured-mod-wsgi-on-apache2-for-django-deployment>
[20] **Sunscrapers** "Djoser" [documentación en línea]
<https://github.com/sunscrapers/djoser>
[21] **Ubuntu Handbook** "How to Install The Latest PyCharm IDE in Ubuntu 16.04 via PPA" [artículo en línea]. [Fecha de consulta: 10 de abril del 2017]
<http://ubuntuhandbook.org/index.php/2016/07/latest-pycharm-ubuntu-16-04-ppa/>
[22] **Udemy** "Angular 4" [curso online]
<https://www.udemy.com/the-complete-guide-to-angular-2/>
[23] **Youtube** "CodingEntepreneurs – Try Angular 4" [lista de reproducción]
<https://www.youtube.com/playlist?list=PLEsfXFp6DpzQThMU768hTZInWUqfoyTEW>
[24] **Youtube** "Mindspace – Angular 2" [lista de reproducción]
<https://www.youtube.com/playlist?list=PL55RiY5tL51qIb5VW2ywbT12UZeqmzBAu>

Anexos

Anexo A: Recursos multimedia utilizados

Pixabay “Robin Bird On Branch In The Garden”
<https://pixabay.com/en/robin-bird-on-branch-in-the-garden-818126/>
Bootswatch “Theme Yeti”
<https://bootswatch.com/yeti/>
Font Awesome “The icons”
<http://fontawesome.io/icons/>
toptal “Subtle patterns”
<https://www.toptal.com/designers/subtlepatterns/inspiration-geometry/>

Anexo B: Herramientas software utilizadas

Balsamiq Mockups “Webdemo” [herramienta en línea].
<https://webdemo.balsamiq.com/>

PyCharm - JetBrains “Community Edition” [IDE de desarrollo].
<https://www.jetbrains.com/pycharm/>

Sublime Text “Sublime Text 3” [Editor de texto]
<https://www.sublimetext.com/3>

Ubuntu “Ubuntu 16.04” [Sistema operativo]
<http://releases.ubuntu.com/16.04/>

Anexo C: Instalación de Django, PyCharm y creación del proyecto Django

Para la correcta instalación del framework Django y sus dependencias en un entorno de desarrollo local, se ejecutan las siguientes instrucciones en el terminal de Ubuntu 16.04, que es el sistema operativo utilizado para el desarrollo del proyecto:

```
>> sudo apt-get install python-psycopg2
>> sudo apt-get install libpq-dev
>> pip install django pytz psycopg2
>> pip install django
>> pip install Pillow
```

Para ver la versión de django que se ha instalado:

```
>> django-admin --version
```

Y se obtiene 1.10.6

A continuación se instalará el IDE de desarrollo PyCharm, en su edición Community, ejecutando las siguientes instrucciones:

```
>> sudo add-apt-repository ppa:mystic-mirage/pycharm
>> sudo apt update
>> sudo apt install pycharm-community
```

Por último, en el directorio de instalación deseado, se crea un nuevo proyecto de Django mediante el siguiente comando:

```
>> django-admin startproject todobirds
```

Tras esto se habrá creado una estructura de proyectos donde el fichero manage.py controlará la mayoría de comandos que se necesita ejecutar para inicializar el servidor wsgi, crear aplicaciones nuevas (el equivalente en Django a módulos), preparar y ejecutar migraciones en los modelos de datos, etc.:

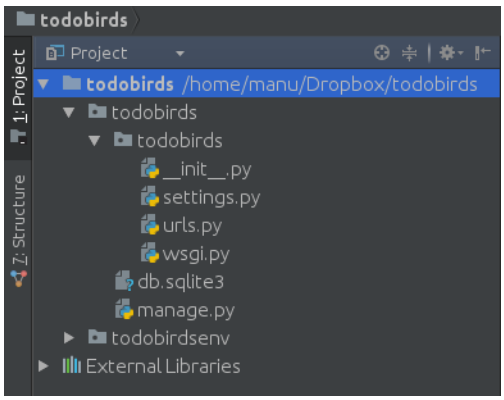


Figura 45: Proyecto Django

Posible incidencia: En caso de que se muestre un mensaje de error pidiendo permiso para editar el fichero workspace.xml del subdirectorio .idea, se ha de dar permisos suficientes al usuario de ubuntu que se utilice mediante un comando similar al siguiente:

```
>> sudo chown -R usuario:grupo /directorio_proyecto/.idea
```

Anexo D: Adaptación del modelo de usuarios en Django

En los requisitos del proyecto se ha definido el requerimiento de que los usuarios utilicen el campo e-mail para identificarse de forma unívoca en el sistema.

En el framework Django, en su versión 1.10, el campo que se utiliza por defecto para la autenticación es el nombre de usuario, con lo que es necesario realizar las siguientes operaciones para poder cambiar la funcionalidad por defecto:

1) Creación de una nueva aplicación en Django que redefina el modelo de usuarios (NOTA: una aplicación en Django equivale a un módulo independiente de los demás, con una definición de modelos y unas determinadas funcionalidades, que se puede reutilizar en otros proyectos). El comando necesario para crear la nueva apliación es el siguiente, y se ha de ejecutar en el directorio donde se encuentra el fichero manage.py del proyecto:

```
>> sudo python3 manage.py startapp customuser
```

2) Edición del fichero models.py de la aplicación customuser creada en el paso anterior, para poder adaptar la forma en la que Django gestiona el modelo de usuarios.

Primero se ha de crear un nuevo gestor de usuarios, que garantice que los usuarios se crean con los datos mínimos necesarios según los requerimientos del proyecto. Para ello se extiende la clase BaseUserManager para crear una nueva, MyUserManager. Esta nueva clase contendrá dos métodos que gestionarán, cada uno por separado, la creación de un usuario normal, y la creación de superusuarios:

- create_user: se encargará de gestionar la creación de usuarios normales, y validará que el campo e-mail se ha proporcionado.
- create_superuser: se encargará por su parte de validar que se ha proporcionado una dirección de e-mail válida, y además configurará todos aquellos campos que identifican a un usuario como superusuario (is_admin, is_staff, is_superuser).

En segundo lugar, dentro del mismo fichero models.py, se añadirá una nueva clase User que extenderá la clase AbstracUser, y mediante la cual se podrán definir los campos de que constará el modelo de usuario que se utilizará en el proyecto.

3) Por último, para que Django tenga en cuenta el nuevo modelo creado y no el que viene por defecto en el framework, hay que especificar en el fichero settings.py (en el subdirectorio “todobirds” del proyecto), la siguiente opción de configuración:

```
AUTH_USER_MODEL = 'customuser.User'
```

Anexo E: Instalación y configuración de librerías externas

En el entorno de desarrollo local se han instalado las siguientes librerías sin instalar previamente un entorno virtual, pero esto será necesario hacerlo en el entorno de producción para evitar posibles interferencias entre distintas instalaciones de python.

Django Rest Framework

Para poder instalar esta librería es necesario ejecutar las siguientes instrucciones:

```
>> pip install djangorestframework
>> pip install markdown
>> pip install django-filter
```

Además es necesario realizar las siguientes configuraciones:

setting.py – INSTALLED_APPS

Es necesario agregar la siguiente línea en negrita a la variable INSTALLED_APPS:

```
INSTALLED_APPS = (
    ...
    'rest_framework',
)
```

urls.py

Es necesario agregar la siguiente línea a la variable urlpatterns del fichero urls.py, a nivel de la aplicación base del proyecto, si se requiere utilizar las vistas de login y logout desde el REST API navegable:

```
url(r'^api-auth/', include('rest_framework.urls', namespace='rest_framework'))
```

settings.py - REST_FRAMEWORK

Por último, es necesario especificar una serie de configuraciones adicionales mediante la variable REST_FRAMEWORK en el fichero settings.py:

```
REST_FRAMEWORK = {
    'DEFAULT_FILTER_BACKENDS': (
        'rest_framework.filters.SearchFilter',
    ),
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.BasicAuthentication',
        'rest_framework.authentication.SessionAuthentication',
        'rest_framework.authentication.TokenAuthentication',
        'oauth2_provider.ext.rest_framework.OAuth2Authentication',
    ),
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 10
}
```

Se definen tres aspectos:

- Filtro de búsqueda: necesario para especificar en las vistas los campos por los que se podrá buscar en el frontend.
- Clases de autenticación: se especifican las básicas, y en último lugar la autenticación por Oauth2.
- Paginación: se especifica la clase por defecto para gestionar la paginación, y el número de elementos por página (que en el caso del entorno de desarrollo es de 10).

Djoser

Para instalar la librería Djoser se ha de ejecutar la siguiente instrucción:

```
>> pip install djoser
```

Además, es necesario realizar las siguientes configuraciones:

setting.py – INSTALLED_APPS

Es necesario agregar la siguiente línea en negrita a la variable INSTALLED_APPS:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    (...),
    'rest_framework',
    'djoser',
    (...),
)
```

urls.py

Es necesario agregar la siguiente línea a la variable urlpatterns del fichero urls.py, a nivel de la aplicación base del proyecto:

```
url(r'^auth/', include('djoser.urls')),
```

Migrar cambios

A continuación es necesario propagar los cambios a la base de datos mediante el siguiente comando:

```
>> sudo python3 manage.py makemigrations
>> sudo python3 manage.py migrate
```

setting.py – DJOSER

Por último, es necesario especificar una serie de configuraciones adicionales en el fichero settings.py, determinadas por la variable DJOSER. La configuración utilizada en el entorno de desarrollo es la siguiente:

```
DJOSER = {
    'DOMAIN': 'localhost:4200',
    'SITE_NAME': 'TodoBirds',
    'PASSWORD_RESET_CONFIRM_URL': 'password_reset_confirm/{uid}/{token}',
    'PASSWORD_RESET_CONFIRM_RETYPE': True,
    'SET_USERNAME_RETYPE': True,
    'LOGOUT_ON_PASSWORD_CHANGE': True,
    'ACTIVATION_URL': 'activate/{uid}/{token}',
    'SEND_ACTIVATION_EMAIL': True,
```

```
    'SEND_CONFIRMATION_EMAIL': True,
    'PASSWORD_VALIDATORS': [],
    'SERIALIZERS': {'user':'anilla.serializers.UserSerializer'},
}
```

Para agilizar la elección de la contraseña por parte del usuario en el entorno de desarrollo local, se han eliminado las restricciones de password válido. Esto se realiza en el fichero settings.py, comentando el contenido de la variable AUTH_PASSWORD_VALIDATORS:

```
AUTH_PASSWORD_VALIDATORS = [
    # {
    #     'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    # },
    # {
    #     'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    # },
    # {
    #     'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    # },
    # {
    #     'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    # },
]
```

Django OAuth Toolkit

Para instalar la librería Django OAuth Toolkit se ha de ejecutar la siguiente instrucción:

```
>> pip install django-oauth-toolkit
```

setting.py – INSTALLED_APPS

Es necesario agregar la siguiente línea en negrita a la variable INSTALLED_APPS:

```
INSTALLED_APPS = (
    ...
    'oauth2_provider',
)
```

urls.py

Es necesario agregar la siguiente línea a la variable urlpatterns del fichero urls.py, a nivel de la aplicación base del proyecto:

```
url(r'^o/', include('oauth2_provider.urls', namespace='oauth2_provider')),
```

setting.py – REST_FRAMEWORK

Es necesario agregar las siguientes líneas en negrita a la variable de configuración de Django Rest Framework REST_FRAMEWORK:

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'oauth2_provider.ext.rest_framework.OAuth2Authentication',
    )
}
```

setting.py – OAUTH2_PROVIDER

Por último, es necesario especificar una serie de opciones adicionales en la variable OAUTH2_PROVIDER:

```
OAUTH2_PROVIDER = {
    'CLIENT_ID_GENERATOR_CLASS': 'oauth2_provider.generators.ClientIdGenerator',
    'ALLOWED_REDIRECT_URI_SCHEMES': ["http", "https"], #A cambiar en producción a sólo https
    'ACCESS_TOKEN_EXPIRE_SECONDS': 5,
}
```

CLIENT_ID_GENERATOR_CLASS: Clase que se encargará de generar identificadores de los clientes que se autenticarán. En el caso de este proyecto, el único cliente será la aplicación web desarrollada en Angular.
ALLOWED_REDIRECT_URI_SCHEMES: Para reforzar la seguridad, se ha de especificar sólo el valor “https” en producción.
ACCESS_TOKEN_EXPIRE_SECONDS: Especifica cuántos segundos es válido el token de autenticación. En el caso del entorno de desarrollo se especifican 5 segundos para que se pongan de manifiesto de forma más evidente posibles incidencias en el proceso de autenticación.

Tras la instalación y configuración de esta librería, se podrán gestionar los tokens, privilegios y clientes desde el backoffice de Django:

Figura 46: Administración de Django OAuth Toolkit

Para poder conectar una aplicación en Angular 2 al backend de Django mediante Django OAuth Toolkit, será necesario crear un cliente desde el panel de administración de Django, con los siguientes detalles:

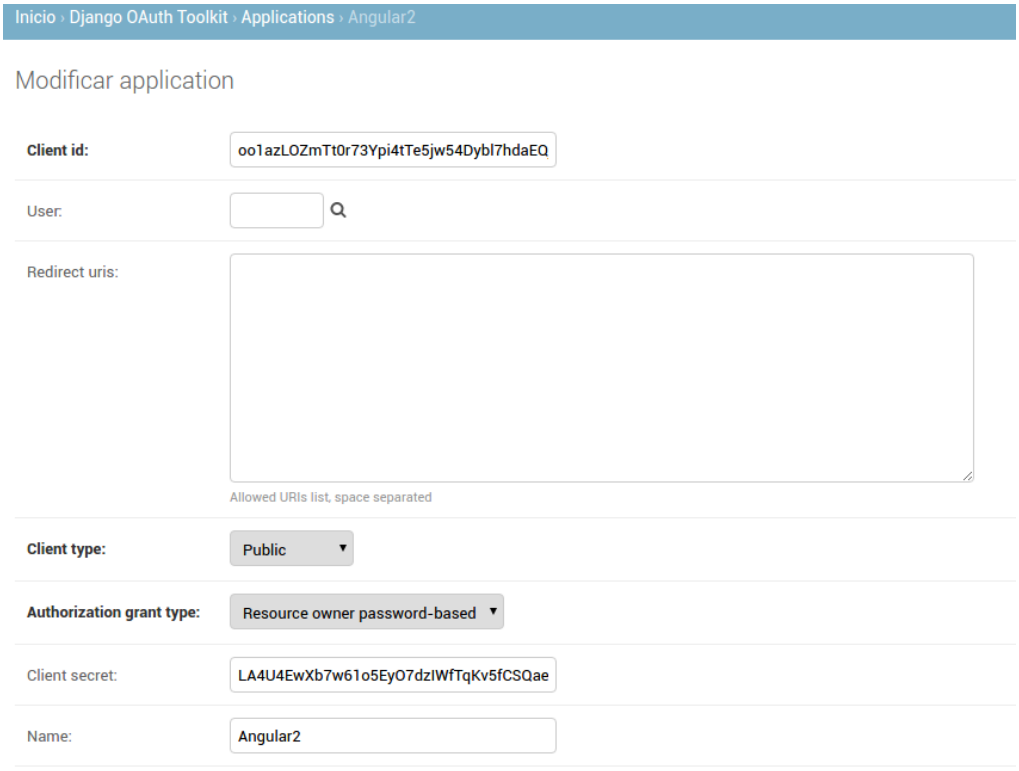


Figura 47: Configuración cliente Angular

django-cors-headers

Cuando se intente realizar pruebas en una máquina local con Angular y Django Rest Framework, surgirá la necesidad de habilitar Django para que funcione sin la restricción CORS. Por ello será necesario instalar la librería django-cors-headers que permitirá realizar peticiones HTTP en el mismo dominio, el local.

Para instalar la librería django-cors-headers se ha de ejecutar la siguiente instrucción:

```
>> pip install django-cors-headers
```

setting.py – INSTALLED_APPS

Es necesario agregar la siguiente línea en negrita a la variable INSTALLED_APPS:

```
INSTALLED_APPS = (
    ...
    'corsheaders',
)
```

setting.py – MIDDLEWARE

Es necesario agregar las siguientes líneas en negrita a la variable de configuración de Django Rest Framework MIDDLEWARE:

```
MIDDLEWARE= [
    ...
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    ...
```

```
]
```

Por último, es necesario configurar también las siguientes variables en settings.py:

```
CORS_ORIGIN_ALLOW_ALL = False
CORS_ALLOW_CREDENTIALS = True
CORS_ORIGIN_WHITELIST = (
        'localhost:8100',
)
```

Anexo F: Configuración del backoffice de Django

Para poder gestionar los modelos configurados en este proyecto desde el backoffice de Django, es necesario extender la clase ModelAdmin de Django para cada uno de los modelos, en el fichero admin.py de cada una de las aplicaciones de Django.

Así, como ejemplo, para poder dar de alta el modelo Especie en el backoffice, se crea una nueva clase EspecieAdmin (donde se especifican los campos a listar, entre otras opciones), y se registra esta clase posteriormente en el backoffice asociándola al modelo Especie:

```
class EspecieAdmin(admin.ModelAdmin):
    list_display = list_display_links =
('codigo','codigo_euring','nombre','nombre_latin','balance','is_active')

admin.site.register(Especie, EspecieAdmin)
```

Tras esto, es posible gestionar este modelo desde el backoffice, como se muestra en la siguiente imagen:

Figura 48: Back-office especies

Figura 49: Back-office, detalle de especie

Anexo G: Instalación del entorno de desarrollo de Angular 4

Para poder instalar el cliente de Angular 4 en el entorno de desarrollo, primero es necesario instalar Node.js y npm. Para ello, se ejecutan las siguientes instrucciones en la máquina de desarrollo (Ubuntu 16.04), desde el terminal de comandos :

```
>> sudo apt-get update
>> sudo apt-get install nodejs
>> sudo apt-get install npm
```

Una vez se ha instalado Node, se instala el cliente de Angular 4 mediante la siguiente instrucción, que instalará además todas las dependencias necesarias para el desarrollo del frontend:

```
>> npm install -g @angular/cli
```

Con esto ya se dispone de todo lo necesario para crear el proyecto en Angular 4, mediante la siguiente instrucción:

```
>> ng new iocells
```

Ahora sólo queda entrar al directorio creado, y ejecutar la siguiente instrucción:

```
>> ng serve
```

Consultando en el navegador la dirección http://localhost:4200/ se obendrá el mensaje “app works!”, lo cual será indicativo de que la aplicación se ha instalado correctamente.

Anexo H: Especificaciones de tablas

Tabla de usuarios	
Campos	- e-mail: tipo e-mail, requerido - Nombre: cadena de caracteres, longitud máxima de 128, no requerido - Apellidos: cadena de caracteres, longitud máxima de 128, no requerido. - Dirección: cadena de caracteres, longitud máxima de 256, no requerido. - Municipio: clave foránea a la tabla de municipios, no requerido. - Código postal: cadena de caracteres, longitud máxima de 5, no requerido. - Provincia: clave foránea a la tabla de provincias, no requerido. - Teléfono: cadena de caracteres, longitud máxima de 20, no requerido. - Anillador: booleano, verdadero por defecto - Activo: booleano, verdadero por defecto
Claves foráneas	Municipio, Provincia
Restricciones	e-mail: campo único.
Tabla de especies	
Campos	- Código: cadena de caracteres, longitud máxima de 15, requerido - Código Euring: cadena de caracteres, longitud máxima de 15, requerido - Nombre: cadena de caracteres, longitud máxima de 300, requerido. - Balance: booleano, falso por defecto - Activo: booleano, verdadero por defecto
Claves foráneas	No tiene
Restricciones	Código, Código Euring: campos únicos.
Tabla de subespecies	
Campos	- Código: cadena de caracteres, longitud máxima de 15, requerido - Nombre: cadena de caracteres, longitud máxima de 128, requerido. - Especie: clave foránea a la tabla de especies. - Activo: booleano, verdadero por defecto
Claves foráneas	Especie
Restricciones	Código: campo único.
Tabla de municipios	
Campos	- Código: cadena de caracteres, longitud máxima de 10, requerido - Nombre: cadena de caracteres, longitud máxima de 128, requerido. - Activo: booleano, verdadero por defecto
Claves foráneas	No tiene
Restricciones	Código: campo único.
Tabla de provincias	
Campos	- Código: cadena de caracteres, longitud máxima de 10, requerido - Nombre: cadena de caracteres, longitud máxima de 128, requerido. - Activo: booleano, verdadero por defecto

Claves foráneas	No tiene
Restricciones	Código: campo único.
Tabla de países	
Campos	- Código: cadena de caracteres, longitud máxima de 10, requerido - Nombre: cadena de caracteres, longitud máxima de 128, requerido. - Activo: booleano, verdadero por defecto
Claves foráneas	No tiene
Restricciones	Código: campo único.
Tabla de localidades	
Campos	- Código: cadena de caracteres, longitud máxima de 10, requerido - Nombre: cadena de caracteres, longitud máxima de 128, requerido. - Municipio: clave foránea a municipios, requerido. - Provincia: clave foránea a provincias, requerido. - País: clave foránea a países, requerido. - UTM: cadena de caracteres, longitud máxima de 128, no requerido. - Coordenadas: cadena de caracteres, longitud máxima de 128, no requerido. - Activo: booleano, verdadero por defecto
Claves foráneas	Municipio, Provincia, País
Restricciones	Código: campo único.
Tabla de estaciones	
Campos	- Nombre: cadena de caracteres, longitud máxima de 128, único, requerido. - Localidad: clave foránea a localidades, requerido. - Extensión: numérico, longitud máxima de 10, requerido. - Tipo de estación: cadena de caracteres, longitud máxima de 128, requerido. - Estándar utilizado: cadena de caracteres, longitud máxima de 128, no requerido. - Activo: booleano, verdadero por defecto
Claves foráneas	Localidad
Restricciones	Nombre: campo único.
Tabla de remitentes	
Campos	- Código: cadena de caracteres, longitud máxima de 15, único, requerido. - Nombre: cadena de caracteres, longitud máxima de 128, requerido. - País: clave foránea a paises, requerido. - Activo: booleano, verdadero por defecto
Claves foráneas	País
Restricciones	Código: campo único.
Tabla de variables	
Campos	- Código: cadena de caracteres, longitud máxima de 15, único, requerido. - Nombre: cadena de caracteres, longitud máxima de 128, requerido. - Descripción: cadena de caracteres, longitud máxima de 256, requerido.

	- Activo: booleano, verdadero por defecto
Claves foráneas	No tiene
Restricciones	Código: campo único.
Tabla de anillas	
Campos	- Usuario: clave foránea a usuarios, requerido. - Año: numérico, longitud máxima de 4. - Remitente: clave foránea a remitentes, requerido. - Anillo desde: cadena de caracteres, longitud máxima de 16, requerido. - Anillo hasta: cadena de caracteres, longitud máxima de 16, requerido. - Cantidad: numérico, verdadero por defecto
Claves foráneas	Usuario, Remitente
Restricciones	Anillo desde, anillo hasta: combinación única.
Tabla de jornadas	
Campos	- Usuario: clave foránea a usuarios, requerido. - Estación: clave foránea a estaciones, requerido. - Fecha: fecha, requerido. - Hora de inicio: campo temporal, requerido. - Hora de fin: campo temporal, requerido. - Hora de reinicio: campo temporal, requerido. - Hora de fin de reinicio: campo temporal, requerido. - Controles registrados: cadena de caracteres, longitud máxima de 32, requerido. - Metros redes: entero, requerido. - Uso de reclamos: booleano, falso por defecto. - Habitat 1: cadena de caracteres, longitud máxima de 32, requerido. - Habitat 2: cadena de caracteres, longitud máxima de 32, requerido.
Claves foráneas	Usuario, Estación
Restricciones	No tiene.
Tabla de subzonas	
Campos	- Jornada: clave foránea a jornadas, requerido. - Código: cadena de caracteres, longitud máxima de 16, requerido. - Forma de captura: cadena de caracteres, longitud máxima de 32, requerido. - Número de trampas: entero, requerido. - Metros redes: entero, requerido. - Hora de inicio: campo temporal, requerido. - Hora de fin: campo temporal, requerido. - Hora de reinicio: campo temporal, requerido. - Hora de fin de reinicio: campo temporal, requerido. - Habitat 1: cadena de caracteres, longitud máxima de 32, requerido. - Habitat 2: cadena de caracteres, longitud máxima de 32, requerido.

	- Agua: cadena de caracteres, longitud máxima de 16, requerido. - Fruta: cadena de caracteres, longitud máxima de 16, requerido. - Especie: clave foránea a especies, requerido.
Claves foráneas	Jornada, Especie.
Restricciones	No tiene.
Tabla meteorología	
Campos	- Jornada: clave foránea a jornadas, requerido. - Fuerza: cadena de caracteres, longitud máxima de 16, requerido. - Dirección: cadena de caracteres, longitud máxima de 16, requerido. - Nubosidad: cadena de caracteres, longitud máxima de 16, requerido. - Precipitación: cadena de caracteres, longitud máxima de 16, requerido. - Grados mínimo: entero, requerido. - Grados máximo: entero, requerido. - Horario: cadena de caracteres, longitud máxima de 16, requerido.
Claves foráneas	Jornada
Restricciones	No tiene.
Tabla de reclamos	
Campos	- Jornada: clave foránea a jornadas, requerido. - Subzona: clave foránea a subzonas, requerido. - Método de atracción: cadena de caracteres, longitud máxima de 16, requerido. - Especie: clave foránea a especies, requerido. - Hora de inicio: campo temporal, requerido. - Hora de fin: campo temporal, requerido.
Claves foráneas	Jornada, Subzona, Especie
Restricciones	No tiene.
Tabla de capturas	
Campos	- Jornada: clave foránea a jornadas, requerido. - Estación: clave foránea a estaciones, requerido. - Remitente: clave foránea a remitentes, requerido. - Modelo: cadena de caracteres, longitud máxima 16, requerido. - Especie: clave foránea a especies, requerido. - Subespecie: clave foránea a subespecies, requerido. - Tipo de captura: cadena de caracteres, longitud máxima 16, requerido. - Subzona: clave foránea a subzonas, requerido. - Hora: campo temporal, requerido. - Edad: cadena de caracteres, longitud máxima 16, requerido. - Sexo: cadena de caracteres, longitud máxima 16, requerido. - Ala: decimal de 8 enteros y 2 decimales, opcional. - Pr3: decimal de 8 enteros y 2 decimales, opcional.

	- Peso: decimal de 8 enteros y 2 decimales, opcional. - Grosor: decimal de 8 enteros y 2 decimales, opcional. - Músculo: decimal de 8 enteros y 2 decimales, opcional. - Estado Rep.: cadena de caracteres, longitud máxima 16, opcional. - Muda verano exterior: cadena de caracteres, longitud máxima 16, opcional. - Muda verano interior: cadena de caracteres, longitud máxima 16, opcional. - Muda invierno exterior: cadena de caracteres, longitud máxima 16, opcional. - Muda invierno interior: cadena de caracteres, longitud máxima 16, opcional. - Estado: cadena de caracteres, longitud máxima 16, opcional. - Tipo marca: cadena de caracteres, longitud máxima 16, opcional. - Codificación: cadena de caracteres, longitud máxima 128, opcional.
Claves foráneas	Jornada, Subzona, Especie
Restricciones	Jornada, Estación, Remitente, Especie, Subespecie, Subzona
Tabla de variables de captura	
Campos	- Captura: clave foránea a capturas, requerido. - Variable: clave foránea a variables, requerido. - Valor: cadena de caracteres, longitud máxima 16, requerido.
Claves foráneas	Balance
Restricciones	No tiene.
Tabla fichas muda	
Campos	- Captura: clave foránea a capturas, requerido. - Terciarias 7..10: cadena de caracteres, longitud máxima 1, opcional. - Secundarias 1..6: cadena de caracteres, longitud máxima 1, opcional. - Primarias 1..10: cadena de caracteres, longitud máxima 1, opcional. - Grandes cobertoras 1..10: cadena de caracteres, longitud máxima 1, opcional. - Cobertoras primarias 1..10: cadena de caracteres, longitud máxima 1, opcional. - Rectrius 1..10: cadena de caracteres, longitud máxima 1, opcional. - Alula 1..10: cadena de caracteres, longitud máxima 1, opcional. - Plumas cuerpo 1..10 (24 campos en total): cadena de caracteres, longitud máxima 1, opcional.
Claves foráneas	Jornada, Subzona, Especie
Restricciones	No tiene.
Tabla balance anual	
Campos	- Usuario: clave foránea a usuarios, requerido. - Año: numérico, longitud máxima de 4, requerido. - Anillas utilizadas: booleano, falso por defecto. - Controles: booleano, falso por defecto. - Recuperaciones: booleano, falso por defecto.

	- Especies: booleano, falso por defecto. - Subespecies: booleano, falso por defecto. - Asignación UTM: booleano, falso por defecto.
Claves foráneas	Usuario
Restricciones	No tiene.
Tabla errores balance anual	
Campos	- Balance: clave foránea a balances, requerido. - Tipo: cadena de caracteres, longitud máxima 16, requerido. - Descripción: cadena de caracteres, longitud máxima 256, opcional.
Claves foráneas	Balance
Restricciones	No tiene.

Tabla 18: Especificaciones de tablas

Anexo I: Especificaciones de servicios web

Alta de usuarios y autenticación	
Registro de usuario Activar usuario Cambiar e-mail Login Privilegios de usuario Recuperar contraseña Confirmar nueva contraseña Refrescar token Logout	Estos servicios web serán proporcionados por librerías externas y se abordan con más detalle en el apartado 5. Autenticación y alta de usuarios.
Modificación de usuario propio	
Campos	- URL del recurso - ID - Todos los campos definidos en el modelo.
Seguridad	- Campo Superusuario es de sólo lectura - Requiere autenticación
Registros afectados	Sólo se recupera el registro perteneciente al usuario que realiza la petición.
Campos de búsqueda	No tiene.
Administración de usuarios	
Campos	- URL del recurso - ID - E-mail - Nombre - Apellidos - Superusuario (Sí/No) - Activo (Sí/No)
Seguridad	- Campo e-mail es de sólo lectura - Requiere permisos de superusuario
Registros afectados	Todos.
Campos de búsqueda	- E-mail - Nombre - Apellidos
Administración de Especies	
Campos	- URL del recurso - ID - Todos los campos definidos en el modelo.

Seguridad	- Requiere autenticación y permisos de superusuario.
Registros afectados	Todos.
Campos de búsqueda	- Código - Código Euring - Nombre - Nombre en latín
Lista de Especies	
Campos	- ID - Código - Código Euring - Nombre
Seguridad	- Requiere autenticación - Permisos de sólo lectura
Registros afectados	Sólo registros activos
Campos de búsqueda	- Código - Código Euring - Nombre
Administración de Subespecies	
Campos	- URL - ID - ID de la especie. - Nombre de la especie. - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación y permisos de superusuario.
Registros afectados	Todos.
Campos de búsqueda	- Código de especie - Código de subespecie - Nombre de subespecie
Lista de Subespecies	
Campos	- ID - Código - Nombre - Nombre especie
Seguridad	- Requiere autenticación - Permisos de sólo lectura
Registros afectados	Sólo registros activos
Campos de búsqueda	- Código - Nombre - Nombre de especie
Administración de Municipios	

Campos	- URL - ID - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación y permisos de superusuario.
Registros afectados	Todos.
Campos de búsqueda	- Código - Nombre
Lista de Municipios	
Campos	- ID - Código - Nombre
Seguridad	- Requiere autenticación - Permisos de sólo lectura
Registros afectados	Sólo registros activos
Campos de búsqueda	- Código - Nombre
Administración de Provincias	
Campos	- URL - ID - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación y permisos de superusuario.
Registros afectados	Todos.
Campos de búsqueda	- Código - Nombre
Lista de Provincias	
Campos	- ID - Código - Nombre
Seguridad	- Requiere autenticación - Permisos de sólo lectura
Registros afectados	Sólo registros activos
Campos de búsqueda	- Código - Nombre
Administración de Países	
Campos	- URL - ID - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación y permisos de superusuario.
Registros afectados	Todos.
Campos de búsqueda	- Código

	- Nombre
Lista de Países	
Campos	- ID - Código - Nombre
Seguridad	- Requiere autenticación - Permisos de sólo lectura
Registros afectados	Sólo registros activos
Campos de búsqueda	- Código - Nombre
Administración de Localidades	
Campos	- URL - ID - ID y nombre de municipio - ID y nombre de provincia - ID y nombre de país - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación y permisos de superusuario.
Registros afectados	Todos.
Campos de búsqueda	- Código - Nombre
Lista de Localidades	
Campos	- ID - Código - Nombre
Seguridad	- Requiere autenticación - Permisos de sólo lectura
Registros afectados	Sólo registros activos
Campos de búsqueda	- Código - Nombre
Administración de Estaciones	
Campos	- URL - ID - ID y nombre de localidad - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación y permisos de superusuario.
Registros afectados	Todos.
Campos de búsqueda	- Código - Nombre
Lista de Estaciones	

Campos	- ID - Nombre - Nombre de localidad
Seguridad	- Requiere autenticación - Permisos de sólo lectura
Registros afectados	Sólo registros activos
Campos de búsqueda	- Nombre - Nombre de localidad - Tipo de estación - Estándar utilizado
Administración de Remitentes	
Campos	- URL - ID - ID y nombre de país - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación y permisos de superusuario.
Registros afectados	Todos.
Campos de búsqueda	- Código - Nombre - Nombre país
Lista de Remitentes	
Campos	- ID - Código - Nombre - Nombre de país
Seguridad	- Requiere autenticación - Permisos de sólo lectura
Registros afectados	Sólo registros activos
Campos de búsqueda	- Código - Nombre - Nombre de país - Extinto
Administración de Variables	
Campos	- URL - ID - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación y permisos de superusuario.
Registros afectados	Todos.
Campos de búsqueda	- Código - Nombre

	- Descripción
Lista de Variables	
Campos	- ID - Código - Nombre
Seguridad	- Requiere autenticación - Permisos de sólo lectura
Registros afectados	Sólo registros activos
Campos de búsqueda	- Código - Nombre - Descripción
Administración de Anillas	
Campos	- URL - ID - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación. - El usuario asociado al registro estará restringido al usuario que realiza la petición.
Registros afectados	Todos los pertenecientes al usuario que realiza la petición.
Campos de búsqueda	- Año - Nombre del remitente
Gestión de Jornadas	
Campos	- URL - ID - ID y nombre de estación - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación. - El usuario asociado al registro estará restringido al usuario que realiza la petición.
Registros afectados	Todos los pertenecientes al usuario que realiza la petición.
Campos de búsqueda	- Nombre de estación - Fecha
Gestión de Subzonas	
Campos	- URL - ID - Todos los campos definidos en el modelo.
Seguridad	Requiere autenticación.
Registros afectados	Todos los pertenecientes al usuario que realiza la petición.
Campos de búsqueda	No tiene.
Gestión de Metorología	

Campos	- URL - ID - Todos los campos definidos en el modelo.
Seguridad	Requiere autenticación.
Registros afectados	Todos los pertenecientes al usuario que realiza la petición.
Campos de búsqueda	No tiene.
Gestión de Reclamos	
Campos	- URL - ID - ID y nombre de especie - ID y código de subzona - Todos los campos definidos en el modelo.
Seguridad	Requiere autenticación.
Registros afectados	Todos los pertenecientes al usuario que realiza la petición.
Campos de búsqueda	No tiene.
Gestión de Capturas	
Campos	- URL - ID - ID y nombre de estación - ID y nombre de remitente - ID y nombre de especie - ID y nombre de subespecie - ID y código de subzona - Todos los campos definidos en el modelo.
Seguridad	- Requiere autenticación. - Las jornadas y subzonas posibles han de estar asociadas al usuario que realiza la petición.
Registros afectados	Todos los pertenecientes al usuario que realiza la petición.
Campos de búsqueda	No contiene.
Gestión de Variables de captura	
Campos	- URL - ID - Todos los campos definidos en el modelo.
Seguridad	Requiere autenticación.
Registros afectados	Todos los pertenecientes al usuario que realiza la petición.
Campos de búsqueda	No contiene.
Gestión de Fichas de muda	
Campos	- URL - ID - Todos los campos definidos en el modelo.

Seguridad	Requiere autenticación.
Registros afectados	Todos los pertenecientes al usuario que realiza la petición.
Campos de búsqueda	No contiene.
Gestión de Balance anual	
Campos	- URL - ID - Todos los campos definidos en el modelo.
Seguridad	Requiere autenticación.
Registros afectados	Todos los pertenecientes al usuario que realiza la petición.
Campos de búsqueda	Año.
Errores de Balance anual	
Campos	- URL - ID - Todos los campos definidos en el modelo.
Seguridad	Requiere autenticación.
Registros afectados	Todos los pertenecientes al usuario que realiza la petición.
Campos de búsqueda	No tiene.

Tabla 19: Especificaciones de servicios web

Anexo J: Pasos para el despliegue

1. Instalación del backend

Instalación del entorno virtual

Antes que nada, se crea un directorio iocells en el directorio home, donde albergar el proyecto de Django y todo lo relativo a éste. Una vez dentro de éste, se crea un nuevo directorio con el nombre del proyecto, quedando con la estructura: /home/ubuntu/iocells
Primero se instalan pip y las dependencias:

```
sudo apt-get install python3-pip python-dev build-essential
sudo pip3 install --upgrade pip
```

Y ahora se instala el entorno virtual:

```
sudo pip3 install --upgrade virtualenv
```

Ahora se creará el entorno virtual del proyecto:

```
cd /home/ubuntu/iocells
sudo python3 /usr/local/lib/python3.5/dist-packages/virtualenv.py iocellsenv
```

Instalación de Django y dependencias python

Para instalar las librerías de python que se necesitan se ha de activar primero el entorno virtual:

```
cd /home/ubuntu/iocells/iocellsenv/bin
source activate
```

La línea de comandos empezará con (iocellsenv) y esto indicará que el entorno virtual está activado y se pueden empezar a instalar las dependencias. Esto se hace para que no haya conflicto con el resto de las librerías python del servidor:

```
sudo apt-get install python-psycopg2
sudo apt-get install libpq-dev
pip install django pytz psycopg2
pip install Pillow
pip install djangorestframework markdown django-filter djoser django-oauth-toolkit django-cors-headers
```

Para ver la versión de django que se ha instalado:

```
django-admin --version
1.11
```

Ésta es la versión LTS que apareció a mitad de realización de este proyecto, aunque el código desarrollado para la versión 1.10 se ha podido reutilizar para la 1.11.

Creación de la base de datos PostgreSQL

- **Instalación**

Se ejecuta la siguiente instrucción:

```
sudo apt-get install postgresql postgresql-client-common
```

Si el servidor PostgreSQL no apareciese en la sección Servers sino en la de Un-used Modules, hay que pulsar sobre la opción Refresh Modules de Webmin.

- **Configuración**

Es necesario modificar el fichero postgresql.conf en la ruta /etc/postgresql/9.5/main, para que acepte conexiones de todas las IPs:

```
#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------
# - Connection Settings -
listen_addresses = '*'                  # what IP address(es) to listen on;
```

Ahora crearemos una base de datos específica para el proyecto de iocells, y un usuario también llamado iocells, con el mismo password de root:

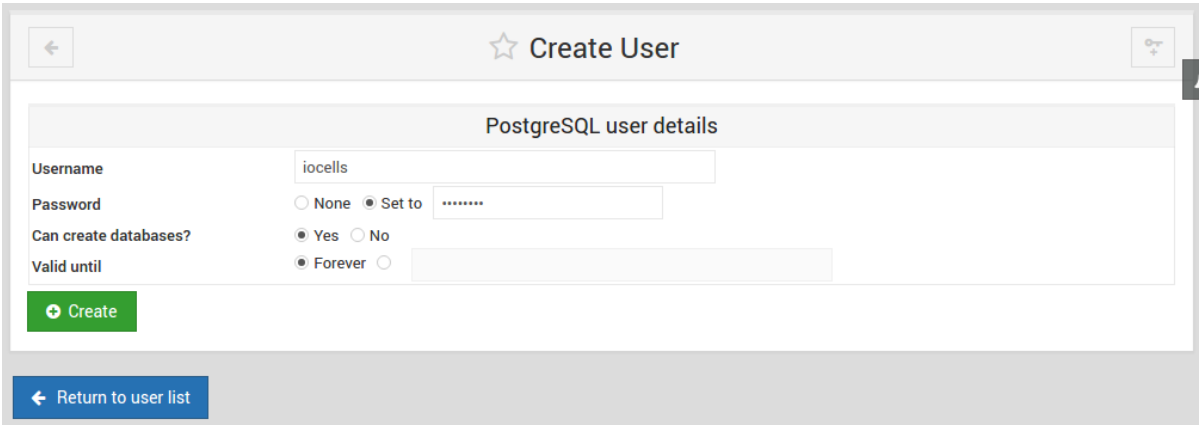


Figura 50: Creación usuario PostgreSQL

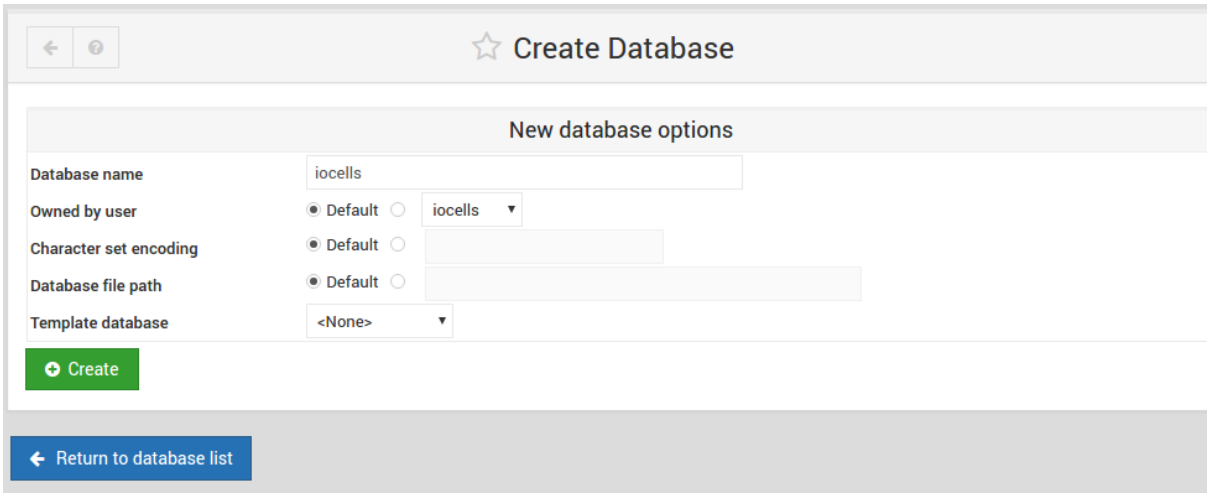


Figura 51: Creación base de datos

Y en la opción de allowed hosts se configura una nueva entrada para el usuario iocells y la base de datos iocells, sólo con acceso local y con autenticación MD5:

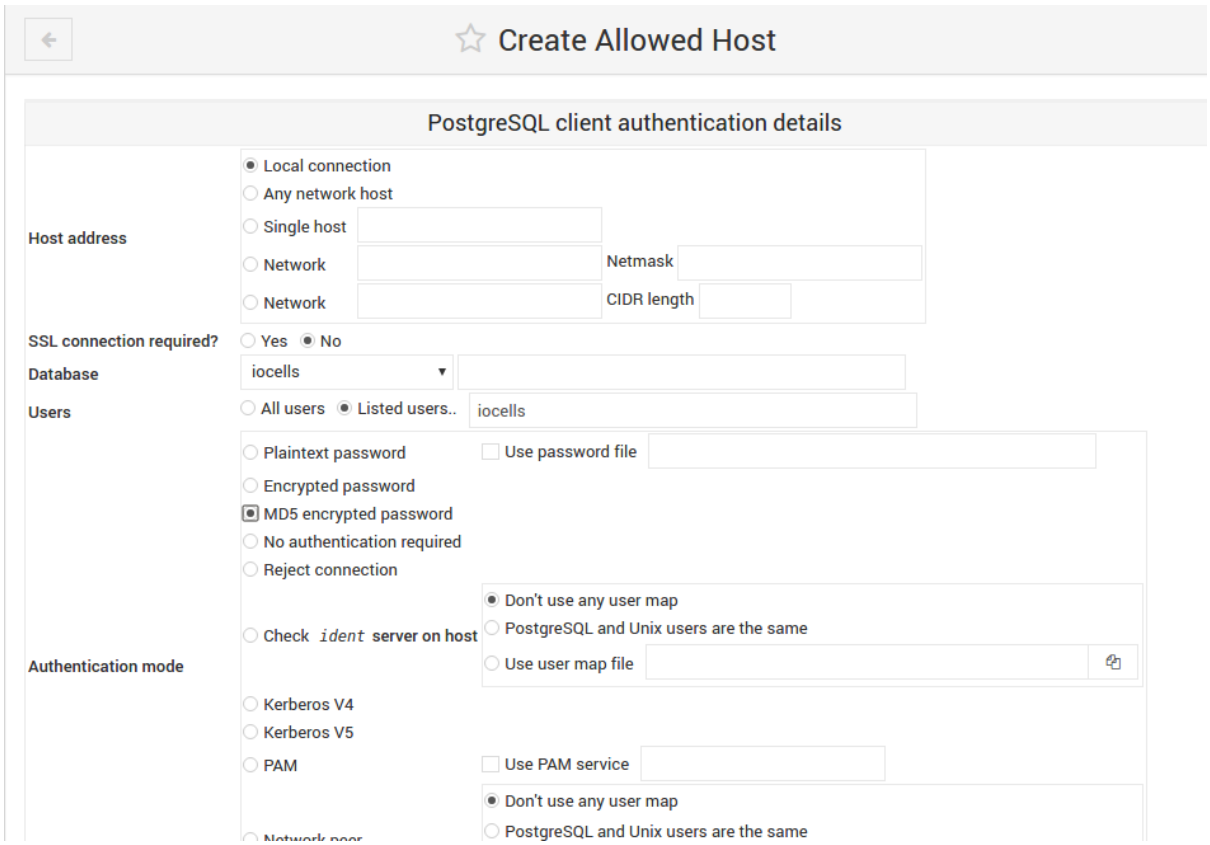


Figura 52: Configuración de hosts permitidos

Tras esto, se reinicia el servidor de postgreSQL, antes de dar el siguiente paso.

Ahora se comprueba que se puede acceder a la base de datos externamente, con el cliente pgAdmin, creado una nueva conexión:

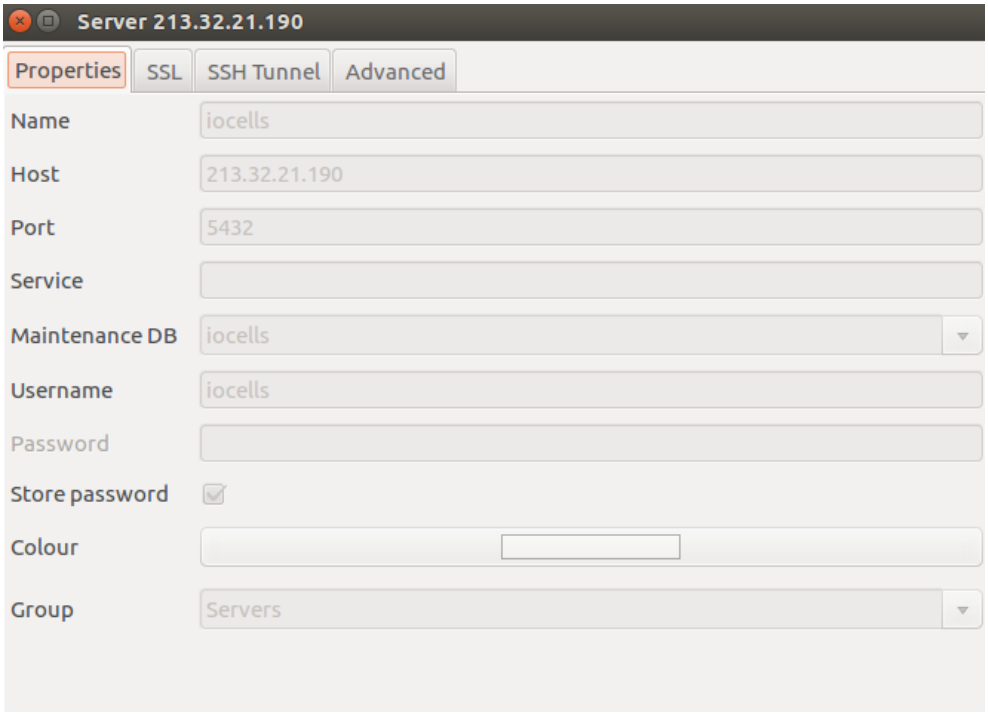


Figura 53: Acceso remoto a base de datos

Despliegue del proyecto

Antes de transferir al servidor, se borrarán del proyecto en local los directorios de migraciones de todas las aplicaciones que se han creado.

Con Filezilla se pasa el contenido del proyecto al directorio /home/ubuntu/iocells/iocells

Y ahora que el proyecto está en producción hay que modificar las siguientes opciones del fichero de configuración settings.py:

```
DEBUG = False
ALLOWED_HOSTS = ['iocells.xyz']
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'iocells',
        'USER': 'iocells',
        'PASSWORD': 'XXX',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
MEDIA_ROOT = 'var/www/media/'
```

Instalación de Apache

- **Instalación de mod_wsgi**

Para que Apache pueda gestionar el proyecto de Django como una aplicación web, se han de seguir los siguientes pasos:

1. Sin entrar en el entorno virtual: para python3 hay que quitar la librería libapache2-mod-wsgi-py3:

```
sudo apt-get remove libapache2-mod-wsgi-py3
sudo apt-get install apache2-dev
```

2. Dentro del entorno virtual, ejecutar:

```
pip install mod_wsgi
```

3. Fuera del entorno virtual, ir a /home/ubuntu/iocells/iocellsenv/bin y ejecutar:

```
sudo /home/ubuntu/iocells/iocellsenv/bin/mod_wsgi-express install-module
```

Esto creará el fichero /usr/lib/apache2/modules/mod_wsgi-py35.cpython-35m-x86_64-linux-gnu.so

4. Por último, hay que crear manualmente 2 ficheros que contendrán lo siguiente:

wsgi_express.load

LoadModule wsgi_module /usr/lib/apache2/modules/mod_wsgi-py35.cpython-35m-x86_64-linux-gnu.so

wsgi_express.conf

WSGIPythonHome /home/ubuntu/uocbackendpra/portfolio/portfolioenv

Estos dos ficheros hay que subirlos a /etc/apache2/mods-available, y ejecutar:

```
sudo a2enmod wsgi_express
sudo service apache2 restart
```

Con el siguiente comando se podrá comprobar si se ha producido algún error:

```
tail /var/log/apache2/error.log
```

- **Configuración de Apache – WSGI**

Para configurar apache y que gestione el proyecto en Django correctamente, hay que añadir un nuevo host virtual /etc/apache2/sites-available/iocellsxyz.conf con el siguiente contenido:

```
<VirtualHost iocells.xyz:80>
        ServerName iocells.xyz
        ServerAlias *.iocells.xyz
        ServerAdmin webmaster@localhost
        DocumentRoot /home/ubuntu/iocells
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

                WSGIDaemonProcess iocells processes=2 threads=12 python-path=/home/ubuntu/iocells/iocellsenv/lib/python3.5/site-packages:/home/ubuntu/iocells/iocells
   WSGIProcessGroup iocells
        WSGIPassAuthorization On
   WSGIScriptAlias / /home/ubuntu/iocells/iocells/todobirds/wsgi.py

        Alias /static /home/ubuntu/iocells/iocells/static
   <Directory /home/ubuntu/iocells/iocells/static>
      Order allow,deny
      Allow from all
        Require all granted
   </Directory>
```

```
        Alias /media /var/www/media
    <Directory /var/www/media>
        Require all granted
    </Directory>

    <Directory /home/ubuntu/iocells/iocells/todobirds>

      <Files wsgi.py>
            Require all granted
      </Files>
    </Directory>
</VirtualHost>
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Para proporcionar a Apache permisos de escritura y lectura en el directorio /media hay que ejecutar los siguientes comandos:

```
sudo mkdir /var/www/media
sudo groupadd varwwwusers
sudo adduser www-data varwwwusers
sudo chgrp -R varwwwusers /var/www/
sudo chmod -R 770 /var/www/media
```

- **Pasos finales**

En el directorio /home/ubuntu/iocells/iocells/ ejecutar, desde el entorno virtual:

```
python manage.py collectstatic
python manage.py makemigrations
```

Y ahora migrar las aplicaciones por separado:

```
python manage.py makemigrations customuser
python manage.py migrate
python manage.py makemigrations anilla
python manage.py migrate
```

Y a continuación hay que crear un superusuario con el siguiente comando:

```
python manage.py createsuperuser
```

Cliente Oauth2 para Angular4

Finalmente, para conectar correctamente (en términos de autenticación) la aplicación en Angular 4 con Django, es necesario crear un nuevo cliente en el backoffice de Django, según la siguiente captura de pantalla:

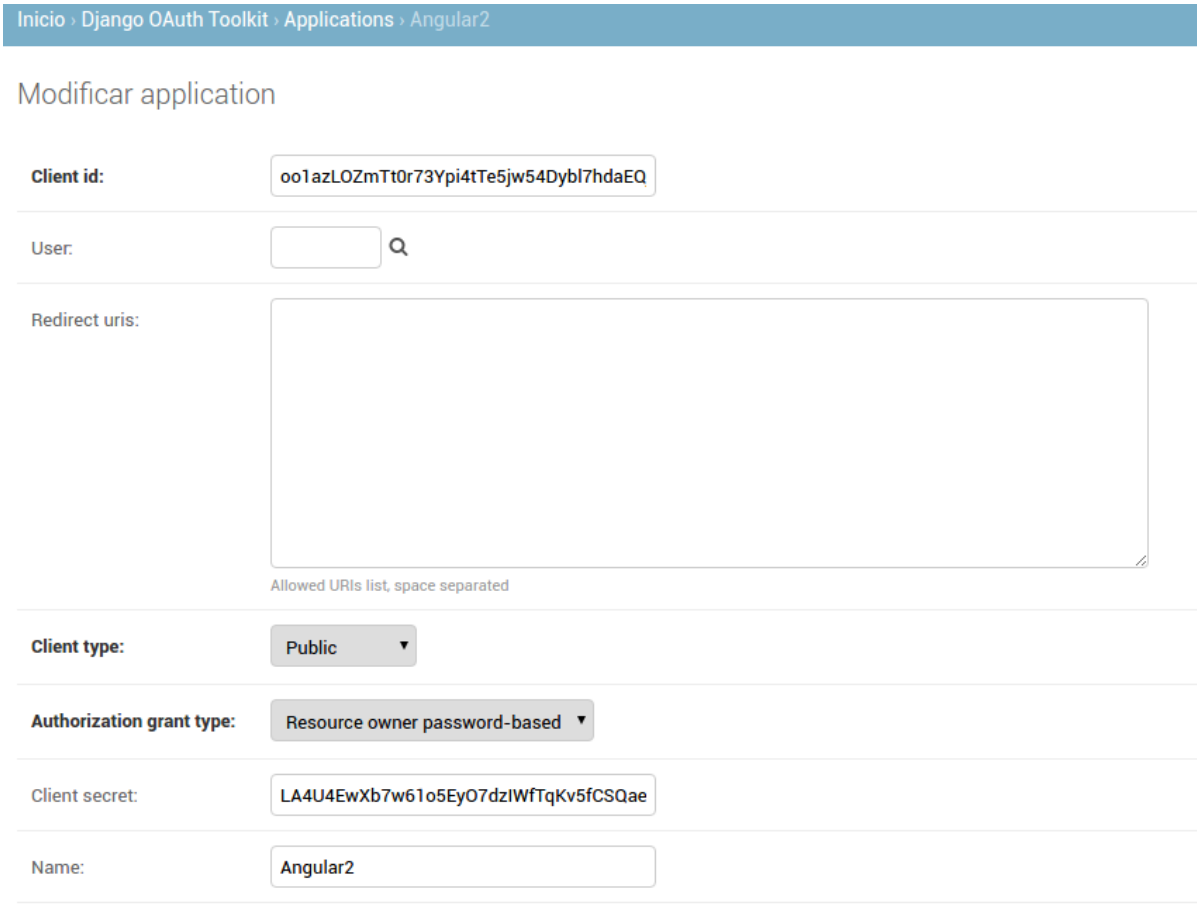


Figura 54: Configuración cliente Angular

NOTA: esta imagen es del entorno local, no del entorno de producción, que por razones de seguridad no se muestran.

2. Despliegue del frontend

Preparativos para el proyecto local de Angular 4

Es necesario previamente modificar algunas de las variables del proyecto en el entorno local:

1. En el fichero app/basic/services/authentication.service.ts, es necesario especificar en las variables client_id y client_secret los valores del cliente creado, según la captura de imagen anterior.
2. En el fichero app/basic/config/parameters.ts hay que modificar la variable backend_url al valor “https://iocells.xyz/”
3. En el fichero app/basic/config/parameters.ts hay que modificar la variable pagination a lo que finalmente se decida establecer en producción. En este caso se ha elegido 10 elementos por página.

Instalación de Angular 4

En una copia adaptada para producción, se ejecuta el siguiente comando en local:

```
ng build --prod --aot
```

El resultado del proceso se crea en el directorio dist, que se pasa a var/www/html/iocells.

3. Configuración Django – CORS

Para que se puedan aceptar peticiones de la url iocells.com al backend de Django, hay que configurar en settings.py las siguientes opciones de la librería CORS que se instaló en el backend:

```
# this disables Cross domain requests
CORS_ORIGIN_ALLOW_ALL = False
# this allows cookie being passed cross domain
CORS_ALLOW_CREDENTIALS = True
# this is the list of allowed origins for cross domain ajax
CORS_ORIGIN_WHITELIST = (
    'iocells.com',
)
```

4. Configuración Django – Djoser

Para que la librería Djoser quede correctamente configurada, hay que especificar las siguientes opciones en producción:

```
DJOSER = {
    'DOMAIN': 'iocells.com',
    'SITE_NAME': 'iOcells',
    'PASSWORD_RESET_CONFIRM_URL': 'password_reset_confirm/{uid}/{token}',
    'PASSWORD_RESET_CONFIRM_RETYPE': True,
    'SET_USERNAME_RETYPE': True,
    'LOGOUT_ON_PASSWORD_CHANGE': True,
    'ACTIVATION_URL': 'activate/{uid}/{token}',
    'SEND_ACTIVATION_EMAIL': True,
    'SEND_CONFIRMATION_EMAIL': True,
    'PASSWORD_VALIDATORS': [],
    'SERIALIZERS': {'user':'anilla.serializers.UserSerializer'},
}
```

5. Configuración Apache

Se crea un nuevo virtualhost en Apache con la siguiente configuración:

```
<VirtualHost iocells.com:80>
        DocumentRoot "/var/www/html/iocells"
        ServerName iocells.com
        ServerAlias *.iocells.com
        <Directory "/var/www/html/iocells">
                AllowOverride All
                Order allow,deny
                Allow from all
        </Directory>
</VirtualHost>
```

Además, para no tener problemas con el enrutado de Angular 4, hay que habilitar el módulo mod_rewrite en Apache, y añadir el siguiente fichero .htaccess a la ruta /var/www/html/iocells:

```
<IfModule mod_rewrite.c>
  RewriteEngine On
  RewriteBase /
  RewriteRule ^index\.html$ - [L]
  RewriteCond %{REQUEST_FILENAME} !-f
  RewriteCond %{REQUEST_FILENAME} !-d
  RewriteRule . /index.html [L]
</IfModule>
```

Esto es así porque al hacer el build del proeycto Angular 4 la única página html de que se dispone es index.html, por eso al realizar cualquier petición a iocells.com hay que redireccionar a index.html que se encargará de construir la ruta correcta.

6. Certificados digitales

Instalación

Se utilizan para este proyecto certificados digitales expedidos por la empresa Letsencrypt. Estos certificados son gratuitos y Letsencrypt es entidad certificadora, con lo que se pueden cifrar las comunicaciones entre el cliente, el frontend y el backend, sin coste adicional.

Se ejecutan las siguientes instrucciones:

```
sudo apt-get update
sudo apt-get install git
sudo git clone https://github.com/letsencrypt/letsencrypt /opt/letsencrypt
cd /opt/letsencrypt
```

Ahora, para añadir un certificado digital al dominio del frontend (iocells.com) se ejecutará la siguiente instrucción:

```
./letsencrypt-auto --apache -d iocells.com
```

Se siguen las instrucciones del asistente, que guía también en la elección del virtualhost correspondiente (creado en la parte de despliegue de Django).

Tras esto ya se ha creado un host para el puerto 443, llamado angular4iocells-le-ssl.conf, y con el siguiente contenido:

```
<IfModule mod_ssl.c>
	<VirtualHost iocells.com:443>
		DocumentRoot "/var/www/html/iocells"
		ServerName angular4iocells
		ServerAlias *.iocells.com
		<Directory "/var/www/html/iocells">
			AllowOverride All
			Order allow,deny
			Allow from all
		</Directory>
	SSLCertificateFile /etc/letsencrypt/live/iocells.com-0001/fullchain.pem
	SSLCertificateKeyFile /etc/letsencrypt/live/iocells.com-0001/privkey.pem
	SSLCertificateChainFile /etc/letsencrypt/live/iocells.com-0001/chain.pem
	Include /etc/letsencrypt/options-ssl-apache.conf
	ServerAlias *.iocells.com
	</VirtualHost>
</IfModule>
```

Pero falta redireccionar todas las peticiones del puerto 80 al 443. Para ello se instala primero el módulo redirect, y después se modifica el virtualhost angular4iocells.conf para llevar a cabo la redirección:

```
sudo a2enmod rewrite
sudo service apache2 restart
```

Y se añaden las siguientes líneas al final del virtualhost angular4iocells.conf, antes de la etiqueta </VirtualHost>:

```
RewriteEngine On
RewriteCond %{HTTPS} off [OR]
RewriteCond %{HTTP_HOST} ^www\. [NC]
RewriteCond %{HTTP_HOST} ^(?:www\.)?(.+)$ [NC]
RewriteRule ^ https://%1%{REQUEST_URI} [L,NE,R=301]
```

Y finalmente se reinicia el servidor de apache.

Estos mismos pasos se repiten para el dominio del backend:

./letsencrypt-auto --apache -d iocells.xyz -d www.iocells.xyz

En este caso obtenemos el siguiente error:

```
Error while running apache2ctl configtest.
Action 'configtest' failed.
The Apache error log may have more information.

AH00526: Syntax error on line 35 of /etc/apache2/sites-enabled/iocells.conf:
Name duplicates previous WSGI daemon definition.

Rolling back to previous server configuration...
```

Figura 55: Error de instalación Let's encrypt

Para solucionarlo se comenta la siguiente línea del fichero de configuración iocellsxyz.conf:

#WSGIDaemonProcess iocells processes=2 threads=12 python-path=/home/ubuntu/iocells/iocellsenv/lib/python3.5/site-packages:/home/ubuntu/iocells/iocells

Se vuelve a ejecutar el comando, eligiendo renovar el certificado creado anteriormente, y redireccionar todas las peticiones al puerto 443 (https):

```
root@vps384943:/opt/letsencrypt# ./letsencrypt-auto --apache -d iocells.xyz -d www.iocells.xyz
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Cert not yet due for renewal

You have an existing certificate that has exactly the same domains or certificate name you requested and isn't close to expiry.
(ref: /etc/letsencrypt/renewal/iocells.xyz.conf)

What would you like to do?
-------------------------------------------------------------------------------
1: Attempt to reinstall this existing certificate
2: Renew & replace the cert (limit ~5 per 7 days)
-------------------------------------------------------------------------------
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Renewing an existing certificate
Performing the following challenges:
tls-sni-01 challenge for iocells.xyz
tls-sni-01 challenge for www.iocells.xyz
Waiting for verification...
Cleaning up challenges
Generating key (2048 bits): /etc/letsencrypt/keys/0011_key-certbot.pem
Creating CSR: /etc/letsencrypt/csr/0011_csr-certbot.pem
Created an SSL vhost at /etc/apache2/sites-available/iocellsxyz-le-ssl.conf
Deploying Certificate for iocells.xyz to VirtualHost /etc/apache2/sites-available/iocellsxyz-le-ssl.conf
Enabling available site: /etc/apache2/sites-available/iocellsxyz-le-ssl.conf
Deploying Certificate for www.iocells.xyz to VirtualHost /etc/apache2/sites-available/iocellsxyz-le-ssl.conf

Please choose whether HTTPS access is required or optional.
-------------------------------------------------------------------------------
1: Easy - Allow both HTTP and HTTPS access to these sites
2: Secure - Make all requests redirect to secure HTTPS access
-------------------------------------------------------------------------------
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Redirecting vhost in /etc/apache2/sites-available/iocellsxyz.conf to ssl vhost in /etc/apache2/sites-available/iocellsxyz-le-ssl.conf
```

Figura 56: Instalación correcta certificado Let's encrypt

Como resultado se crea el virtualhost iocellsxyz-le-ssl.conf con el siguiente contenido:

```
<IfModule mod_ssl.c>
<VirtualHost iocells.xyz:443>
        ServerName iocells.xyz
        ServerAlias *.iocells.xyz
        ServerAdmin webmaster@localhost
        DocumentRoot /home/ubuntu/iocells
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

         #WSGIDaemonProcess iocells processes=2 threads=12 python-path=/home/ubuntu/iocells/iocellsenv/lib/python3.5/site-packages:/home/ubuntu/iocells/iocells
   WSGIProcessGroup iocells
        WSGIPassAuthorization On
   WSGIScriptAlias / /home/ubuntu/iocells/iocells/todobirds/wsgi.py

   Alias /static /home/ubuntu/iocells/iocells/static
   <Directory /home/ubuntu/iocells/iocells/static>
      Order allow,deny
      Allow from all
        Require all granted
   </Directory>
   Alias /media /var/www/media
   <Directory /var/www/media>
        Require all granted
   </Directory>

   <Directory /home/ubuntu/iocells/iocells/todobirds>

      <Files wsgi.py>
            Require all granted
      </Files>
   </Directory>
SSLCertificateFile /etc/letsencrypt/live/iocells.xyz/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/iocells.xyz/privkey.pem
Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
</IfModule>
```

Por último, se descomenta la línea que se había comentado antes en el fichero iocellsxyz.conf (no en iocellsxyz-le-ssl.conf, que ha de quedar comentada) y se añaden las siguientes líneas, como en el caso de angular4iocells.conf:

```
RewriteEngine On
RewriteCond %{HTTPS} off [OR]
RewriteCond %{HTTP_HOST} ^www\. [NC]
RewriteCond %{HTTP_HOST} ^(?:www\.)?(.+)$ [NC]
RewriteRule ^ https://%1%{REQUEST_URI} [L,NE,R=301]
```

Finalmente se reinicia apache de nuevo para que sean efectivos los cambios.

Autorenovación del certificado digital

Para autorenovar el certificado (caduca a los 3 meses), se ejecuta el siguiente comando:

```
sudo crontab -e
```

A continuación se selecciona el editor vim. Se modifica el fichero (primero se pulsa una tecla cualquiera para que entre en modo -Insert-), introduciendo la siguiente línea:

```
30 2 * * 1 /opt/letsencrypt/letsencrypt-auto renew >> /var/log/le-renew.log
```

y cuando se termina la edición se pulsa en la tecla Esc, y cuando desaparece -Insert- se escribe :wq .

Al acabar se recibe la confirmación mediante el siguiente mensaje:

crontab: installing new crontab

Se puede también comprobar desde webmin:

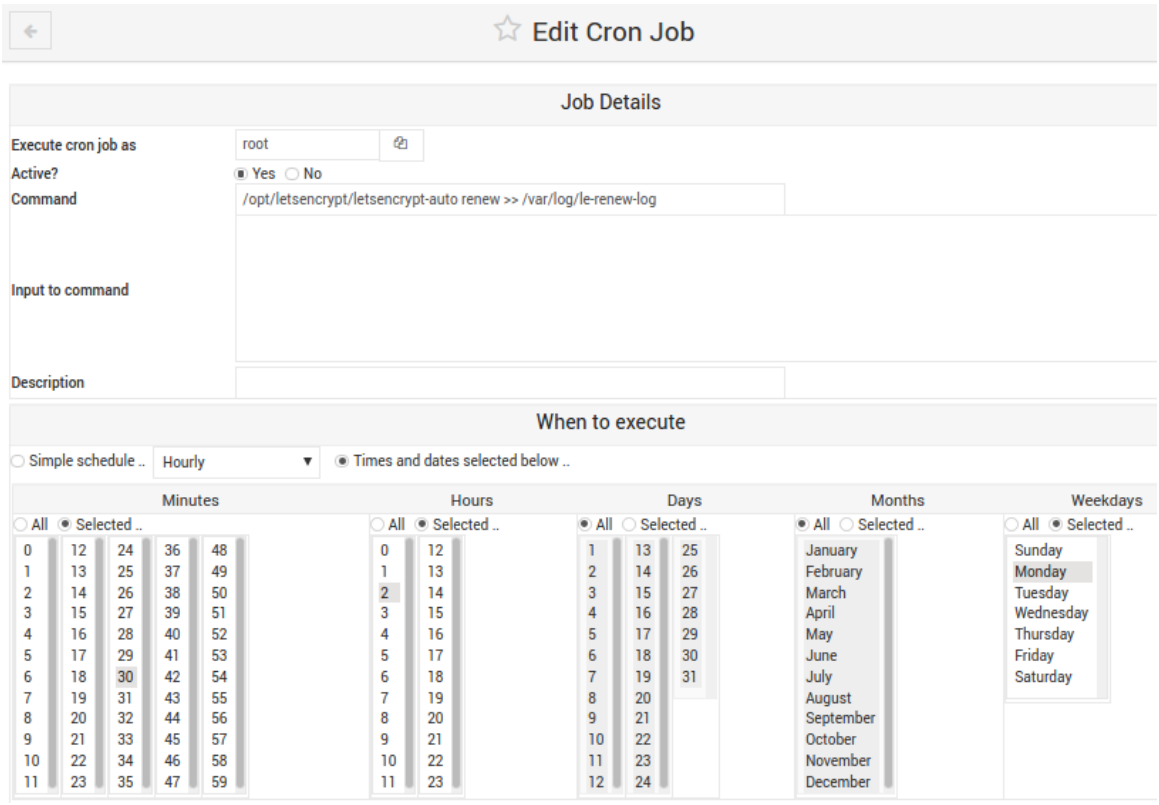


Figura 57: Renovación automática del certificado digital