# Documentación del Proyecto - Huellitas Felices (Segunda Entrega)

Esta documentación detalla el funcionamiento técnico y funcional de la nueva versión del sistema backend de Huellitas Felices. Se trata de una aplicación desarrollada con Node.js y Express, que gestiona integralmente una veterinaria y petshop. A las funcionalidades ya existentes se le suman varias mejoras, incluyendo el uso de MongoDB Atlas, control de acceso con roles, autenticación segura, gestión de historias clínicas y despliegue en Render.com.

# Nuevas funcionalidades incorporadas

- Login seguro con bcrypt y sesiones.
- Almacenamiento de usuarios, productos, mascotas e historias clínicas en MongoDB Atlas.
- Control de acceso por roles: administrador, empleado, veterinario.
- Prevención de eliminación del último administrador.
- Restricción de cambio de rol a administradores.
- Módulo completo de historias clínicas vinculado a mascotas.
- Observaciones médicas con historial y fecha.
- Despliegue online del sistema con Render.com.
- Conexión a MongoDB Atlas mediante variables de entorno.

# **Módulos principales**

### **Usuarios**

Los usuarios se almacenan en una colección MongoDB. Las contraseñas se almacenan de forma segura con bcrypt y se validan durante el inicio de sesión. Cada usuario tiene un rol definido que habilita o restringe ciertas funciones.

#### Mascotas

Permite registrar, editar, buscar y eliminar mascotas. Se introdujo la opción de búsqueda por nombre y especie. Ahora también es posible asociar historias clínicas a cada mascota.

#### Historias clínicas

Este módulo es exclusivo para usuarios con rol de veterinario. Se pueden registrar nuevos datos clínicos, incluyendo veterinario a cargo, motivo de consulta y observaciones con fecha. Las observaciones anteriores se mantienen visibles como historial.

### Turnos

Se gestionan turnos para atención veterinaria y peluquería. Se pueden crear, editar y eliminar. Las vistas fueron adaptadas para integrarse con MongoDB.

### Autenticación y control de acceso

El sistema utiliza sesiones de Express ('express-session') para mantener la sesión activa. Cada acción protegida verifica el rol del usuario antes de ejecutar la acción. El middleware 'autorizar.js' verifica que el usuario tenga el rol necesario para acceder a ciertas rutas.

### Despliegue en Render.com

El proyecto fue subido a una cuenta de Github, para luego ser vinculado y desplegado en Render.com, permitiendo accederlo desde cualquier navegador. Para ello, se definieron variables de entorno como la URL de conexión a MongoDB Atlas (`MONGO\_URI`) y el secreto de sesión (`SESSION\_SECRET`).

# Tecnologías utilizadas

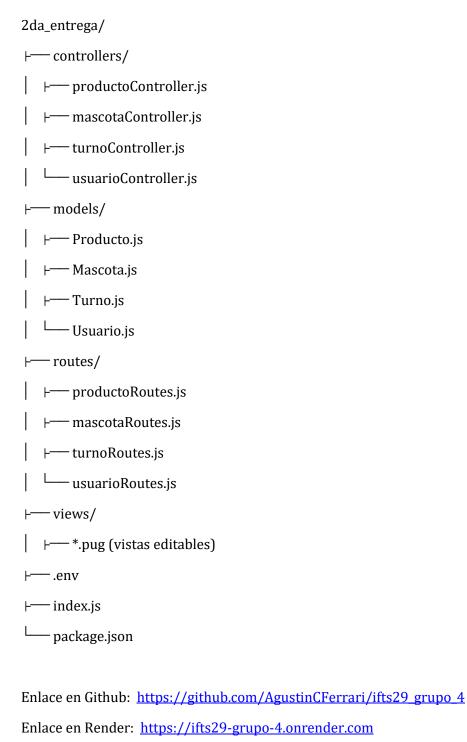
- Node.js
- Express
- MongoDB Atlas
- Mongoose
- bcrypt
- express-session
- dotenv
- Pug (motor de vistas)
- Github
- Render.com

### Tabla comparativa entre ambas entregas del proyecto

Función	1ª Entrega	2ª Entrega
Login con JSON		□ (ahora Mongo)
Hasheo de contraseñas		□ bcrypt
Control por rol		□ Mejorado
Historias clínicas		
Observaciones con historial		
MongoDB Atlas		
Despliegue online (Render)		
Protección último administrador		

### Estructura general del proyecto

El proyecto está dividido en carpetas por responsabilidad: modelos, controladores, rutas y vistas. Cada módulo cuenta con sus propias rutas, controladores y vistas específicas.



Enlace en Drive: <a href="https://drive.google.com/drive/folders/1y1-kYj-">https://drive.google.com/drive/folders/1y1-kYj-</a>

0kDpP\_1pyRDVHg9c0\_ciyKl7d?usp=drive\_link