UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERIA DEL SOFTWARE

Sistema de control y monitorización de sillas subescaleras

System of control and monitoring of stairlifts

Realizado por

Carlos Andrés Billordo Pérès Campagnale

Tutorizado por

Luis Manuel Llopis Torres

Departamento

Desarrollo de sistemas empotrados con Arduino

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2019

Fecha defensa:

Fdo. El/la Secretario/a del Tribunal

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

D/Dª.: Carlos Andrés Billordo Pérès Campagnale, con DNI 77797907Q, estudiante del Grado en Ingeniería del software, de la Universidad de Málaga.

DECLARO QUE:

El Trabajo Fin de Grado/Máster denominado: Sistema de control y monitorización de sillas subescaleras

es de mi autoría, inédito (no ha sido difundido por ningún medio, incluyendo internet) y original (no es copia ni adaptación de otra), no habiendo sido presentado anteriormente por mí ni por ningún otro autor ni en parte ni en su totalidad. Así mismo, se ha desarrollado respetando los derechos intelectuales de terceros, para lo cual se han indicado las citas necesarias en las páginas donde se usan, y sus fuentes originales se han incorporado en la bibliografía. Igualmente se han respetado los derechos de propiedad industrial o intelectual que pudiesen afectar a cualquier institución o empresa.

Para que así conste, firmo la presente declaración en Málaga, a 27 de septiembre de 2019.

Fdo.: D/Dª Carlos Andrés Billordo Pérès Campagnale

Resumen

En los tiempos del internet de las cosas, cada vez más dispositivos de nuestro hogar tienen la habilidad de conectarse a internet para darnos información, permitirnos controlarlos desde lejos y así hacernos la vida más fácil. Un dispositivo que ayuda a las personas con movilidad reducida no puede quedar fuera de esta tendencia ya que ahora mismo en el mercado europeo ninguna silla subescaleras o plataforma para minusválidos tiene este requisito, se ha diseñado un software para controlar una silla subescaleras para obtener la información, enviarla a un servidor que la guarda y permite a los técnicos poder ver los errores o hacer un mantenimiento preventivo para que los usuarios no se queden sin poder moverse libremente.

Además, se ha desarrollado una aplicación web que permite añadir nuevas sillas subescaleras, ver un listado con la información más relevante o ver el detalle e histórico de una silla en particular.

Se ha optado por un Raspberry Pi por su potencia, versatilidad, conectividad y precio reducido para manejar el hardware de la silla subescaleras, para el servidor se ha seleccionado NodeJS ya que es un *framework* orientado a eventos y es ideal para manipular datos en tiempo real y para la aplicación web se ha optado por Angular que es perfecto para mostrar datos en tiempo real junto con Socket.io.

Palabras clave: control, subescaleras, IoT, Raspberry, Nodejs, Python

Abstract

Nowadays, in the times of the internet of things, an increasing number of home devices can be connected to the internet to give us information, allow us to control them from a distance and thus make our lives easier. A device that helps people with reduced mobility cannot be left out of this trend, since right now there are no stair lifts or platforms for the disabled in the European market with this requirement. A software has been designed to control a stair lift to gather information , send it to a server that saves it and allow technicians to see the errors or take preventive maintenance measures so that users are not left without being able to move freely.
In addition, a web application has been developed which allows new stair lifts to be added, to see a list with the most relevant information or to see the detail and history of a particular stair lift.
Raspberry Pi has been chosen for its power, versatility, connectivity and reduced price to handle the hardware of the stairlift. As for the server, NodeJS has been selected since it is an event-oriented framework and is ideal for manipulating data in real time. And as regards the web application, Angular has been chosen, which is perfect for displaying real time data together with Socket.io.

Keywords: control, stairlift, IoT, Raspberry, Nodejs, Python

Índice

1

Introducción

1.1 Motivación

Ahora mismo en el mercado europeo, no existe ningún otro sistema que permita conectar estos dispositivos a internet por lo tanto hay una demanda de los clientes y fabricantes, algunos permiten añadir como extra un teléfono 3G para poder llamar en caso de emergencia.
Estamos en el momento del Internet de las cosas (IoT) para domotizar los hogares por la comodidad y seguridad que esto implica. Un producto tan valioso para las personas con movilidad reducida no puede quedar fuera de esta tendencia, cuando un componente falla, un sensor queda presionado por accidente o simplemente se queda sin batería, el cliente se queda atrapado sin la posibilidad de subir o bajar, cuando esto sucede se debe llamar a un técnico especializado que puede tardar días en acudir.
Con el sistema que se quiere desarrollar, se solucionarían muchos de estos fallos en minutos o se podría avisar al cliente que su dispositivo necesita un cambio de baterías con antelación.

1.2 Objetivos

Los objetivos de este proyecto son:

- Realizar el diseño hardware y software para controlar los motores, sensores y actuadores de las sillas para así poder leer la información y abaratar costes
- Leer el estado actual de la silla y enviarlo por internet a un servidor
- Guardar el historial de todas las sillas subescaleras
- Servir una aplicación web que permita ver tanto el historial como el estado actual de las sillas configuradas y añadir nuevas sillas
- Realizar un pequeño control remoto del dispositivo subescaleras mediante la aplicación web que permita reiniciar el sistema, mover la silla si algún sensor no permite su movimiento por razones de seguridad o alguna otra característica que se detecte en la toma de requisitos.

1.3 Estructura de la memoria

La memoria del trabajo fin de grado consta de cinco capítulos, las referencias y tabla de imágenes, los capítulos referentes a la realización del proyecto (2,3 y 4) contienen una introducción y la información referente a cada apartado:

- El capitulo 2: Fase previa. Describe todos los pasos e información previas al trabajo para ayudar a dar una idea global de lo que se tiene y lo que se quiere y como se quiere conseguir.
- El capitulo 3: Fase de desarrollo. Describe la solución que se ha diseñado y como se ha hecho para cumplir con los objetivos fijados.
- El capitulo 4: Fase de pruebas. Describe las pruebas por las que han pasado las partes desarrolladas para detectar los fallos y que no se vuelvan a producir.
- El capitulo 5: Conclusiones y mejoras futuras. Describe como su nombre indica a las conclusiones que se llega después de terminar el trabajo y las posibles mejoras que se han encontrado durante su desarrollo para mejorar el producto.

1.4 Tecnologías a utilizar

Python

Es un lenguaje de programación interpretado, interactivo y orientado a objetos, utiliza tipos de datos dinámicos por lo que no hay que identificar el tipo de las variables para que el código sea más limpio, pero ello puede conllevar a errores difíciles de solucionar. Tiene la ventaja de ser multiplataforma aunque en este trabajo solo se utiliza en GNU/Linux. Tiene una sintaxis que favorece la legibilidad del código ya que no utiliza llaves para delimitar los bloques, en su defecto utiliza la indentación (espacios o tabuladores). Es el principal lenguaje que promueve la fundación Raspberry para ser utilizada en su placa.

Node.js

Es un entorno de ejecución (framework) de JavaScript, basado en ECMAScript, código abierto y orientado a eventos asíncronos que es ideal para aplicaciones que consuman datos en tiempo real, como es el caso.
Todo el programa se ejecuta en un solo hilo, pero no se bloquea nunca, ya que utiliza entradas y salidas asíncronas, cuando una llamada se ejecuta, esta llama a su *callback* para devolver los datos de la operación y esto permite que el hilo siga corriendo por ejemplo en la lectura del disco duro o de la base de datos.
Contiene muchos módulos básicos que permite iniciar un proyecto en muy poco tiempo por ejemplo *fs* permite el acceso a disco o *http* el acceso a red para crear un servidor web con pocas líneas de código.

MongoDB

Es una base de datos NoSQL orientado a documentos escrita en C++, en vez de utilizar tablas para guardar datos, utiliza una estructura llamada BSON, que es una representación binaria de JSON, con un esquema dinámico, esto quiere decir que cada objeto (registro en SQL) puede tener una estructura diferente al resto dentro de la misma colección (tabla en SQL). Para hacer consultas tiene una consola escrita en JavaScript pero existen librerías para la mayoría de los lenguajes utilizados actualmente. Este tipo de bases de datos se utiliza especialmente en sistemas distribuidos gracias a su facilidad para escalar horizontalmente.

AngularJS

Es un entorno de ejecución (framework) de JavaScript que se ejecuta del lado del cliente para crear aplicaciones web dinámicas. Mediante etiquetas HTML personalizadas se puede definir acciones que se llevarán a cabo en el navegador y que se van actualizando en tiempo real, de esta manera se extiende las posibilidades de HTML para conseguir páginas web más dinámicas y con más posibilidades. Angular utiliza los patrones MVC (Modelo Vista Controlador) y MVVM (Modelo Vista Modelo de vista). Además permite añadir etiquetas asociadas a directivas para añadir más funcionalidades a las aplicaciones desarrolladas.

MQTT/Mosquitto

MQTT[5] es un protocolo desarrollado por IBM para el envío de mensajes asíncronos entre dispositivos de poca capacidad de procesamiento ya que es muy liviano, por lo tanto es perfecto para el internet de las cosas (en adelante IoT). Utiliza un sistema de publicación y suscripción, donde existe un *broker* que centraliza las comunicaciones y cada interesado puede suscribirse a un tema, todos los mensajes que lleguen a ese tema son reenviados a los demás dispositivos.
Como los mensajes en MQTT se organizan en temas, es fácil especificar quien va a recibir la información o del otro lado, recibir mucha información de distintos temas.

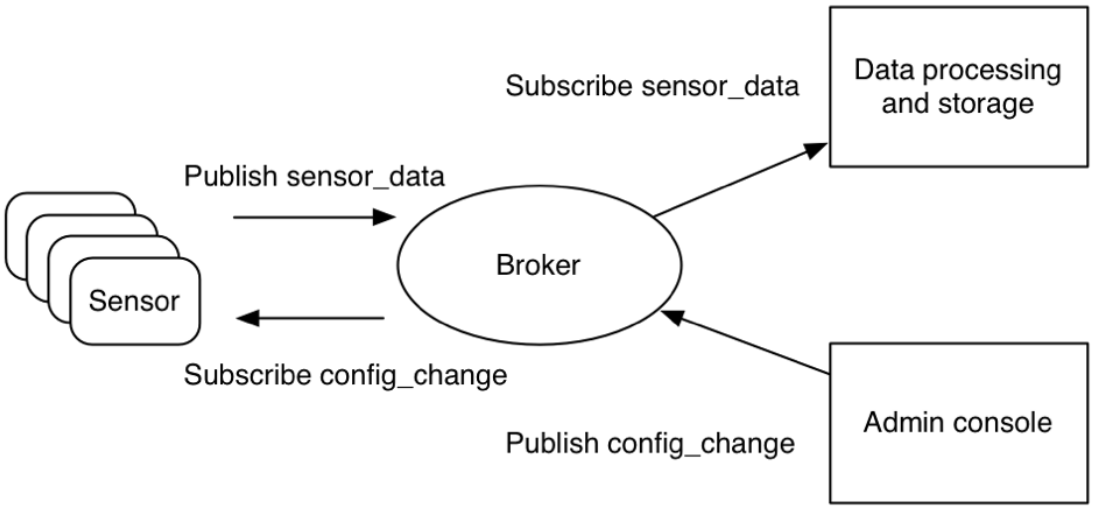


Ilustración 1: Ejemplo MQTT

Para manejar estos mensajes se ha seleccionado el broker Mosquitto[6] desarrollado por Eclipse y código libre, está escrito en C y es muy ligero.

Socket.IO

Es una librería JavaScript para aplicaciones web en tiempo real, permite la comunicación bilateral entre el navegador y el servidor, al igual que NodeJS, es orientado a eventos. Su funcionamiento es parecido a MQTT, el cliente se registra en los canales que necesita y desde ellos puede enviar y recibir mensajes por lo que es perfecto para aplicaciones que esperan información sin tener que recargar la página cada vez que se produzca un nuevo evento.

2

Fase previa

2.1 Introducción

En este capitulo se indicará los pasos e información del estudio previo del trabajo para ayudar a dar una idea global de los dispositivos que se van a tratar, las funcionalidades que se deben tener en cuenta y los pasos a seguir para llevarlo a cabo.
Se describe que es una silla subescaleras, se muestran las partes más importantes y su funcionamiento básico, las partes utilizadas en este trabajo, los componentes añadidos que ayudarán a que la silla tenga las funcionalidades previas y nuevas.

2.2 Requisitos

- La silla debe moverse a izquierda y derecha
- La silla debe detenerse en caso de activarse un sensor
- La silla debe activar el motor cuando se pulse el botón de movimiento
- La silla debe activar el motor desde la aplicación web
- La silla debe bajar/subir el apoyapié al bajar/subir el asiento
- La silla debe bajar la velocidad en las curvas abiertas
- La silla debe detenerse en el punto de carga
- La silla debe apagar todo en caso de presionarse el sensor de emergencia
- La silla debe gestionar la carga de las baterías
- La silla debe conectarse al servidor
- La silla debe enviar al servidor el estado actual de las baterías

- La silla debe enviar al servidor cualquier interacción que se haga sobre ella
- La silla debe mostrar por pantalla el estado actual de las baterías
- La silla debe mostrar por pantalla cualquier interacción que se haga sobre ella
- La silla debe actualizar la lista de mensajes cuando se actualice en la aplicación web
- La aplicación web debe permitir registrar nuevas sillas
- La aplicación web debe permitir editar las sillas que hay en el sistema
- La aplicación web debe permitir añadir códigos de estado
- La aplicación web debe permitir editar código de estado
- La aplicación web debe permitir eliminar código de estado
- La aplicación web debe permitir visualizar todas las sillas registradas mostrando información básica, nivel de batería y último estado en tiempo real
- La aplicación web debe permitir visualizar el detalle de una silla y su historial en tiempo real
- La aplicación web debe permitir mover la silla a distancia

2.3 Sillas subescaleras

Una silla o plataforma subescaleras son dispositivos mecánicos donde una persona con movilidad reducida se puede sentar (silla) o entrar con una silla de ruedas, muletas (plataforma) para salvar escaleras o desniveles, utilizan un riel o guía para desplazarse tanto en vertical como por la escalera, generalmente utilizan engranajes para realizar el movimiento a través del riel, aunque algunas marcas tienen otros tipos de mecanismos.
Cuentan con un motor alimentado por baterías para que esta sea independiente de la corriente eléctrica, además contienen sensores final de carrera para detenerse en caso de que algo obstruya el camino.
En este trabajo solo se tratarán las sillas subescaleras, el funcionamiento es similar pero para acotar los componentes y funciones se prefiere descartar las plataformas subescaleras.
Como se puede ver en la Ilustración 2 como mínimo todas las sillas poseen, un asiento, la unidad central que alberga el motor, placa, estructura y mecanismo de desplazamiento, un sensor a cada lado de la unidad central, una botonera para bajar y subir por la escalera, un sensor de emergencia y un apoyapiés con sensores de presión.
Existen sillas curvas y rectas, su funcionamiento son equiparables pero no iguales, las sillas curvas tienen más complejidad ya que deben cambiar su velocidad en las curvas para cuidar a la persona que se está transportando.

Como se ha comentado anteriormente, existe un sensor de emergencia que determina si la silla va a demasiada velocidad, por ejemplo, en el caso que se rompa el engranaje y caiga libre, tiene un sistema mecánico que muerde el riel frenando la silla para proteger al ocupante y apaga el sistema por si es un error software o si llega al final del riel y la silla se pasa del punto final, en este caso se corta la electricidad tanto de las baterías como del transformador por si el software se queda congelado y no para el motor.

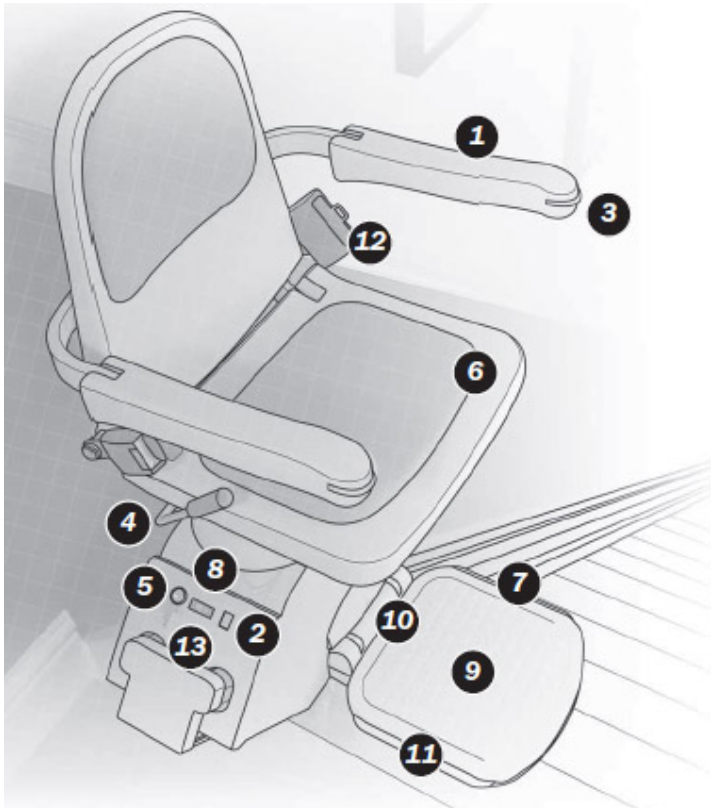


Ilustración 2: Partes externas más importantes

Las partes más importantes de una silla subescaleras son:

1. Apoyabrazos
2. Pantalla informativa
3. Botonera de movimiento
4. Palanca para girar el asiento
5. Llave de seguridad
6. Asiento
7. Sensor de presión derecho del apoyapiés
8. Interruptor general
9. Apoyapiés
10. Bisagra del apoyapiés
11. Sensor de presión izquierdo del apoyapiés

12. Cinturón de seguridad
13. Sensor de presión izquierdo

Internamente se pueden ver otras partes importantes que también son genéricas a todos los tipos de sillas:

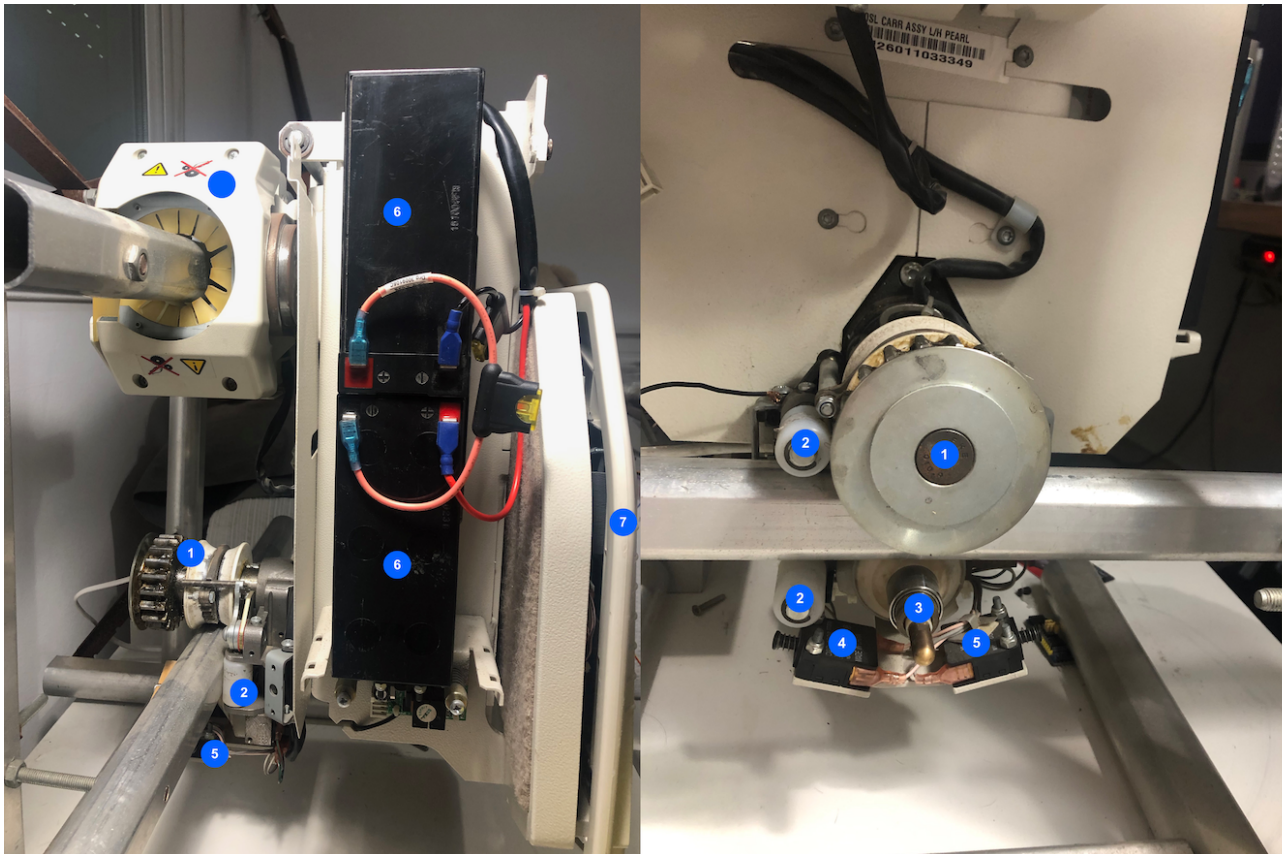


Ilustración 3: Partes internas más importantes

1. Piñón de giro
2. Rodillos estabilizadores
3. Positivo de carga
4. Sensor de presión del riel inferior derecha
5. Sensor de presión del riel inferior izquierda
6. Baterías
7. Sensor de presión del apoyapiés izquierda

Su funcionamiento es muy sencillo, se bajan los apoyabrazos, se baja el asiento y el apoyapiés, cuando se presiona uno de los botones del apoyabrazos, la silla tarda 1 segundo en empezar a moverse por si la presión fue involuntaria, se debe mantener el botón presionado para que, en caso de mareo o cualquier otro problema, la silla se detenga. Al pasar por una curva abierta, la silla disminuye la velocidad para que no despida al pasajero o de sensación de mareo. Durante el trayecto si algo o alguien presiona algún sensor, la silla se detiene hasta que el

sensor este libre nuevamente. Cuando llega al punto de carga superior o inferior, la silla se detiene automáticamente y pone las baterías a cargar.

2.3 Silla subescaleras utilizada

Para el trabajo fin de grado se ha seleccionado una silla subescaleras de la marca Stannah, el modelo Solus curvo ya que es un dispositivo ampliamente extendido, probado y contiene todos los componentes que luego se pueden extrapolar a otras marcas y modelos.

En la ilustración 4 se puede ver la silla utilizada y sus partes más importantes

Ilustración 4: Silla subescaleras Stannah Solus

En el modelo concreto que se ha utilizado tiene la función de bajar el apoyapiés cuando el usuario baja el asiento, de esta manera no necesita agacharse para bajarlo, se ha optado por incluirlo en el trabajo, por lo que se controla el movimiento de la peana y se informa cuando baja o sube al servicio web.

2.4 Componentes añadidos

2.4.1 Controlador

En el trabajo se ha escogido el Raspberry Pi, Ilustración 5, se analizó el Intel Galileo y el Arduino Due.

El Intel Galileo se ha descartado porque es un controlador de un procesador con un núcleo, en las pruebas realizadas previamente, se detectó que la velocidad de respuesta entre los sensores de presión y la detención del motor eran muy altas (del orden de segundos) ya que al tener un solo núcleo, las tareas de multihilo no se podían llevar a cabo y el sistema intercalaba las operaciones de lectura del sensor, escritura del relé del motor, envío del mensaje al servidor y los mensajes mostrados en la pantalla, todo ello hacía que la respuesta no sea inmediata, no atendiendo al requisito que la detención del motor debe ser inmediata tras ser presionado un sensor. Además, la documentación y la comunidad que hay detrás de este controlador son escasas comparado con las otras dos alternativas.

El Arduino se descartó por la necesidad de hardware complementario para su conexión a internet, su poca potencia, ya que cuenta con un microcontrolador, aunque la velocidad de respuesta entre la presión del sensor y la detención del motor eran buenas, al no ejecutar un sistema operativo, limitaba bastante las opciones a la hora de realizar el trabajo.

Raspberry Pi[1], en este caso es un ordenador de placa reducida que tiene un procesador Broadcom BCM2837B0 de cuatro núcleos de 64bits a 1,4GHZ, lo cual es perfecto para poder lanzar varios hilos que se encarguen de las tareas y así dejar el principal para la gestión de los sensores y el motor. Además tiene Wi-Fi integrado, 40 GPIO y 1GB DDR2 de RAM.

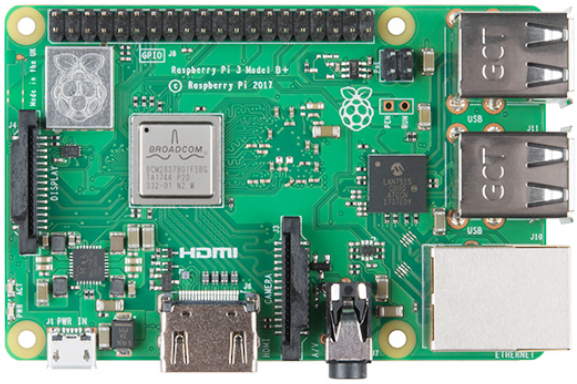


Ilustración 5: Raspberry Pi 3B+

2.4.2 Shield de gestión de corriente

Se ha seleccionado el Zero2Go Omini para la gestión de la corriente ya que el motor trabaja a 24V de corriente continua y el Raspberry trabaja a 5V de corriente continua.

Entre las características más importantes están: la conversión de entre 3.3V y 28V a 5V y proporciona el voltaje que se tiene en la entrada para poder calcular el porcentaje de carga y saber cuando está cargando.

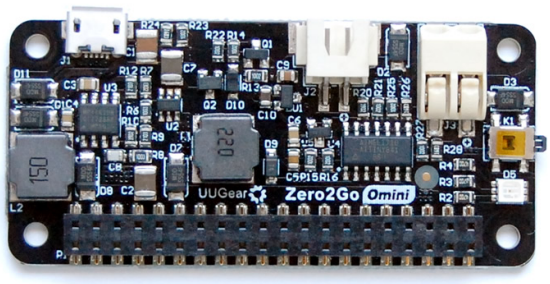


Ilustración 6: Zero2Go Omini

2.4.3 Pantalla OLED

Para mostrar información se ha seleccionado una pantalla OLED de 1,5" y 128x128 pixeles, en poco tamaño permite mostrar mucha información y hay librerías de código abierto para hacerla funcionar en Python.
Otra opción barajada fue una pantalla táctil HDMI pero se descartó por su alto precio y uso de GPIO.

Ilustración 7: Pantalla OLED

2.4.4 Regulador de tensión

Se han utilizado tres reguladores de tensión, ya que las dos entradas que tiene la silla proveen 24V y algunos componentes necesitan menos, los reguladores aceptan hasta 38V y devuelven hasta 36V y una tensión de 3A.
El primero se ha utilizado para el circuito de emergencia.

El segundo se utiliza para bajar el voltaje a 12V para el motor del apoyapiés.
El tercero es el encargado de bajar a 21V la velocidad del motor principal para la velocidad de curva.

Ilustración 8: Regulador de tensión

2.4.5 Modulo de relés

Se ha utilizado un módulo de 8 relés, son alimentados por la Raspberry, se conectan a 5V y para activar cada relé existe una entrada independiente de 5V, este modulo se ha utilizado para seleccionar el sentido de giro y velocidad del motor, el sentido de giro del motor del apoyapiés y la carga de las baterías.

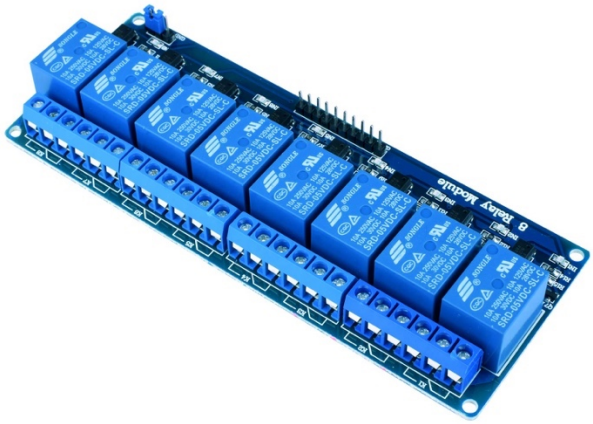


Ilustración 9: Módulo de relés

2.5 Componentes reutilizados

Además de la estructura principal, apoyapiés y asiento, se han reutilizado para completar lo anteriormente citado:

2.5.1 Motores

- Motor central: utiliza es de 24V de corriente continua, con una fuerza de 0,32KW y un torque de 60Nm, permite mover a una persona de hasta 120Kg

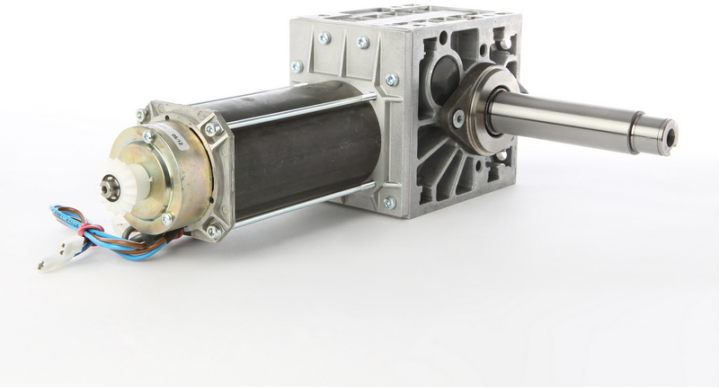


Ilustración 10: Motor de 24V

- Motor apoyapiés: Es un pequeño motor de 12V que se acciona al subir/bajar el asiento para ayudar al usuario y así no tener que agacharse.

Ilustración 11: Motor de 12V

2.5.2 Sensores

Se utilizan microinterruptores para detectar si la silla choca con algo o alguien, para detectar cuando llega al final del recorrido, para detectar cuando está a punto de salirse del riel o para detectar cuando el usuario baja el asiento para bajar el apoyapiés automáticamente.

Ilustración 12: Microinterruptor

2.5.3 Baterías

Para alimentar el motor se utilizan 2 baterías de 12V conectadas en serie. Además mediante los conversores de tensión y el Zero2Go Omini se alimenta al Raspberry Pi y el motor del apoyapiés.

Ilustración 13: Batería de 12V

3

Fase de desarrollo

3.1 Introducción

En esta sección se explica tanto el desarrollo hardware como el software, en el caso del hardware, se explicará como se han conectado los componentes anteriormente citados, sus funciones y en el caso del software se explicará el desarrollo del script encargado del manejo de la silla, los programas usados en el servidor, el desarrollo del software encargado de gestionar los mensajes, conexiones y proporcionar la API para la web y la aplicación web encargada de brindar la información al usuario.

3.2 Desarrollo Hardware

3.2.1 Hardware de la silla

Todas las conexiones pasan por una placa de pruebas y una cinta conecta los GPIO del Raspberry, a través de ésta se alimentan la pantalla, los reguladores de tensión, el módulo de relés y se pasa 3.3V a los microinterruptores para detectar cuando se presiona. De esta manera el Raspberry recibe una interrupción en el momento que hay una bajada o subida de tensión en los GPIO configurados para ello, esto se llama Pull-up o Pull-down.

Para interconectar los componentes que se han reutilizado de la silla se han reutilizado también los conectores que iban a la antigua placa base, como se

puede ver en la ilustración 14 se ha fabricado los conectores con cables debido al alto costo que tienen este tipo de conectores.

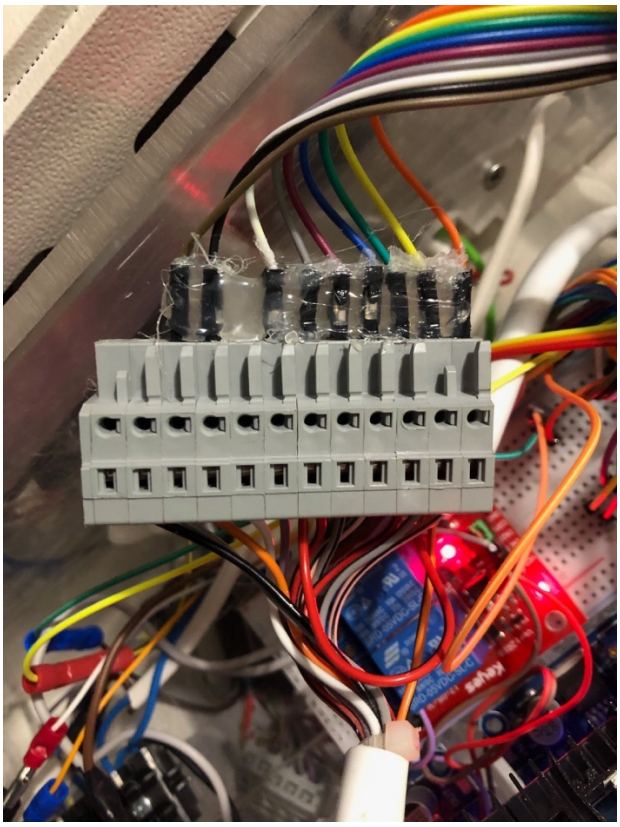


Ilustración 14: Ejemplo de conexión con la silla

La pantalla, como se ha indicado en el capitulo anterior, se alimenta mediante la salida del Raspberry de 5V y la información la recibe mediante el protocolo I^2C, el cual permite conectar varios dispositivo al mismo pin GPIO y para diferenciarlos tienen distintas direcciones, la información se envía por un pin (SDA) en serie, la velocidad máxima en el Raspberry es de 400kbit/s pero la velocidad normal es de 100kbit/s, además utiliza otro pin a modo de reloj para controlar las comunicaciones con los periféricos.

Para controlar el movimiento de la silla se detectan 2 microinterruptores alojados en el apoyabrazos del asiento, cuando alguno es presionado, el Raspberry lee los sensores por si hay algún obstáculo y si no se activa unos de los relés del módulo instalado, este modulo se alimenta de los 5V que suministra el Raspberry, y tiene una entrada para activar cada uno de los relés, en este caso la conexión del motor utiliza tres relés: los dos primeros permiten seleccionar cual de los dos polos del motor recibirá el positivo, de esta manera se indica el sentido de rotación del motor y el tercero permite seleccionar si al

motor se le conecta el positivo de las baterías (24V) o el positivo que sale del regulador de tensión (21V) para la velocidad de curva.

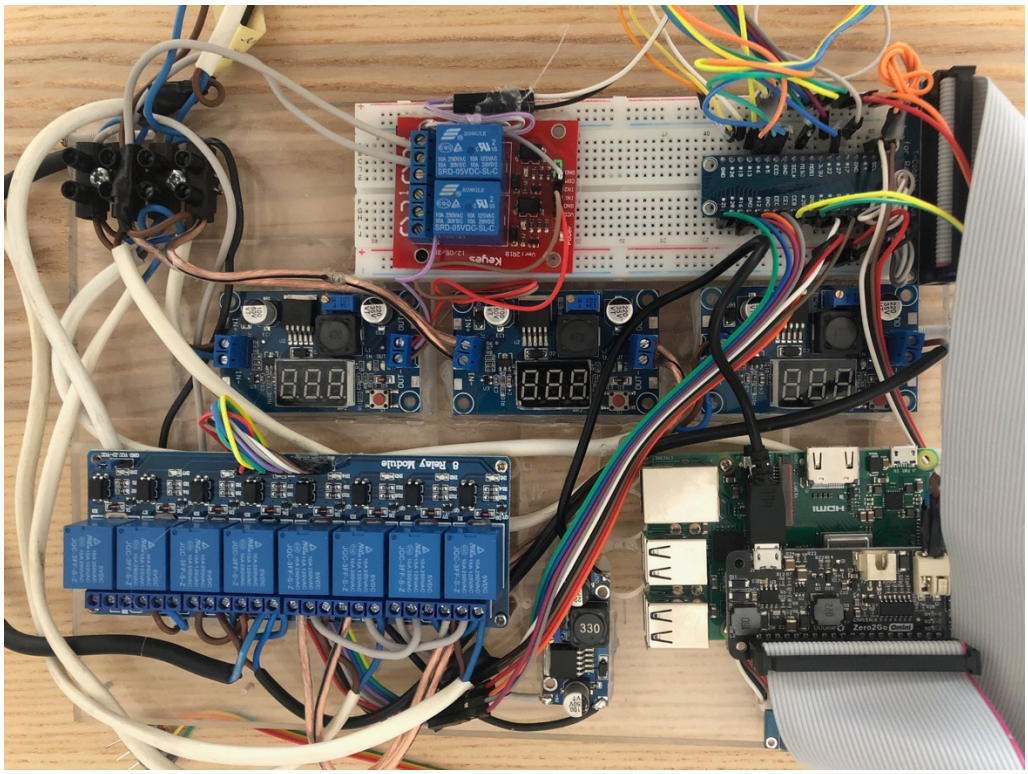
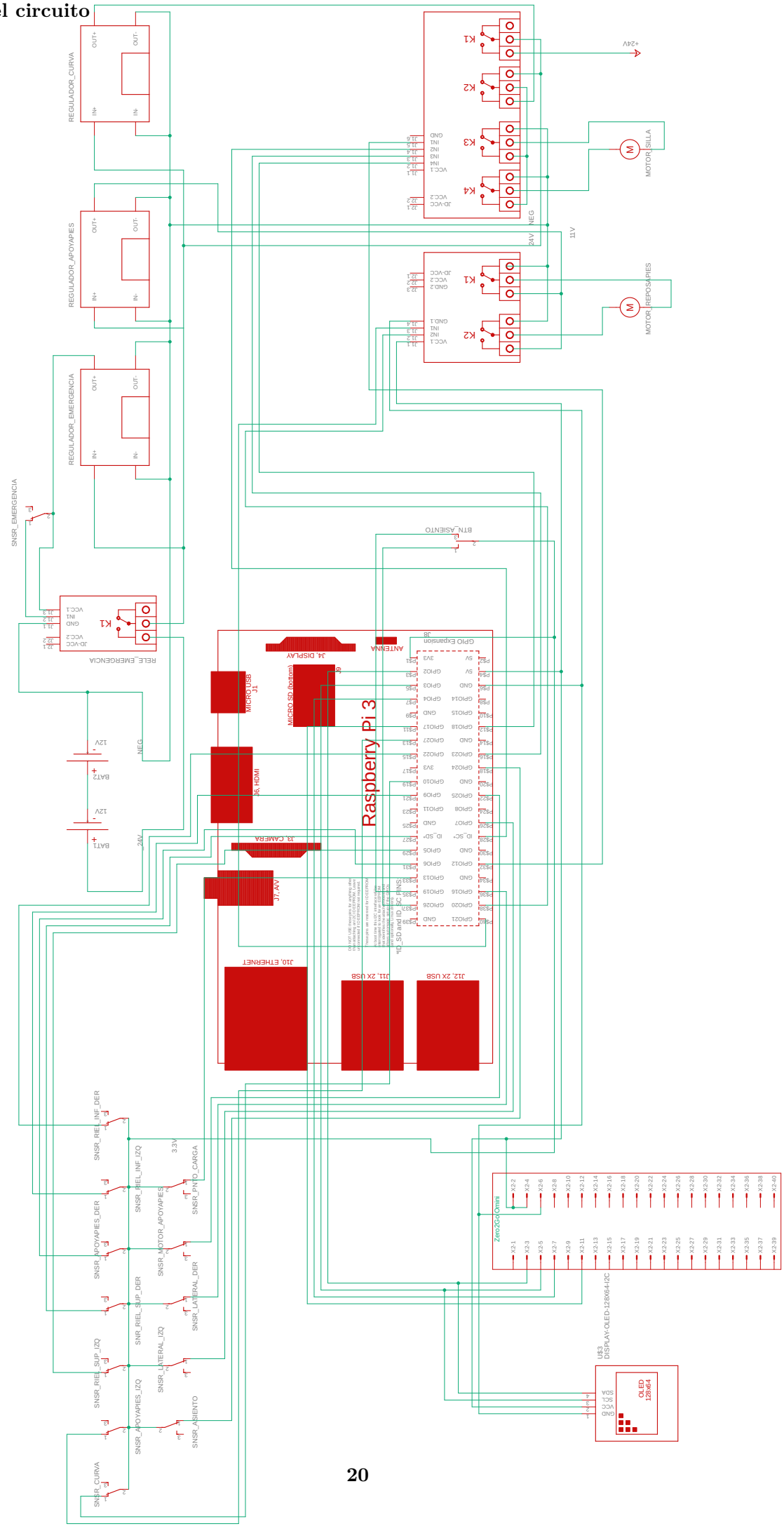


Ilustración 15: Componentes añadidos

La gestión de la carga de las baterías se encarga un hilo que lee el estado de las baterías cada 10 segundos, en el caso en que las baterías bajen de 25V, se activa el relé que comunica el positivo del cargador y el del circuito por lo que el Raspberry y todos los componentes pasan a ser alimentados por el cargador hasta que se detecte 27.5V, ya que el Zero2Go Omini soporta hasta 28V. En el caso en el que la silla esté fuera del punto de carga, el script no activaría el relé por seguridad, por ejemplo en el caso en el que algo conectase el positivo de carga con el riel (negativo) e hiciese un corto circuito.
Como sistema se seguridad se ha diseñado un circuito de emergencia, en él se alimenta un relé con 5V que se obtienen de las baterías pasando por uno de los reguladores de tensión y esos mismos 5V entran y salen del sensor de emergencia, así cuando el sensor corta esta corriente, todo el sistema se apaga. Este sensor se acciona cuando el positivo de carga de la silla se presiona más de lo normal, eso quiere decir que el sensor de el punto de carga ha fallado y la silla continúa su marcha para salirse del riel. Como este circuito no depende del software, si el script que controla la silla o el sistema operativo tienen algún error, la silla se apagaría por completo, pasaría lo mismo si algún componente de este circuito falla, todo se apagaría por precaución.

3.2.2 Diseño del circuito



3.3 Desarrollo Software

Como se puede ver en la Ilustración 16, el sistema se compone de tres nodos principales:

- Sillas subescaleras, en las cuales el software implementado gestiona el hardware y a su vez envía por MQTT el estado de cada silla al servidor.
- Servidor, el cual contiene una aplicación principal hecha en Node.JS que gestiona toda la información, un servidor MQTT para gestionar la comunicación y un servidor MongoDB para guardar el histórico de cada silla. Además sirve la aplicación web.

Aplicación web, la cual permite a los técnicos gestionar las sillas y comprobar el estado de éstas en tiempo real.

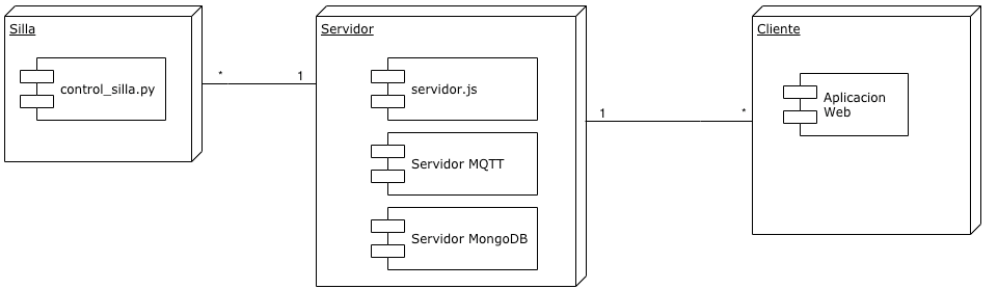


Ilustración 16: Diagrama de despliegue

3.3.1 Controlador silla subescaleras

Para el control de la silla se ha desarrollado un script en Python, ya que es el lenguaje de programación recomendando por la Fundación RaspberryPi[2]. Este script contiene un hilo principal encargado de la gestión de los sensores, motores y botonera, además lanza dos hilos, uno para la administración de la pantalla y otro para la lectura periódica de las baterías y el estado de carga.

Al iniciar el programa, se inicializan las variables globales, la pantalla, se conecta al servidor MQTT, se lanzan los dos hilos mencionados anteriormente y se inicializan el estado de los GPIOs añadiendo la configuración de las interrupciones.

Se ha utilizado la librería oficial de Raspberry para el control de los GPIO[3], esto permite interactuar con los GPIO de entrada y salida, en los de entrada mediante el método *add_event_detect* se detecta cuando un pin cambia de estado (botonera o sensor presionado), por ejemplo:

```
GPIO.setup(snsrRielSupDer, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.add_event_detect(snsrRielSupDer, GPIO.BOTH, callback=stopMotor, bouncetime=300)
```

Gracias a esto se ahorra tener un bucle infinito leyendo cada una de las entradas en busca de un cambio, lo cual se traduce en ahorro de CPU y una respuesta más inmediata cuando hay algún evento.
Además, se descarga del servidor la lista de códigos con sus mensajes correspondientes, de esta manera, si hay algún cambio no haría falta actualizar todos los sistemas, también se suscribe en el servidor de MQTT al tema *sillas/codigos* y cuando se recibe un mensaje, la silla recarga la lista de mensajes sin necesidad de reiniciar el script.
El hilo encargado de la gestión de las baterías y la carga está diseñado con un bucle infinito que lee el voltaje a través del Zero2Go Omini y dependiendo de esta lectura y si la silla está en el punto de carga o no, acciona el relé correspondiente para iniciar o detener la carga. Las condiciones que se debe dar para cargar son: que el voltaje leído sea menor que 25V y la silla esté en el punto de carga, hay que tener en cuenta también que en sucesivas lecturas la silla tenga menos de 27.5V y ya esté cargando, además de, obviamente, estar en el punto de carga. En caso contrario se detiene la carga para proteger el Zero2Go Omini. Como dato importante, la carga se desactiva cuando la silla se va a mover, dado que puede darse el caso que haya un fallo y la silla se mueva hacia el final del riel, se active el sensor de emergencia pero el cargador continúe alimentando el circuito y la silla caiga del riel. Cada vez que el hilo lee y trata la información de las baterías, envía al servidor el porcentaje de carga actual para mostrarlo al usuario.
El otro hilo, como se ha dicho anteriormente, se encarga de gestionar los mensajes que se muestran en la pantalla, mediante un bucle infinito, se espera el evento que actualiza la variable global del porcentaje de la batería o que haya un mensaje nuevo procedente de una nueva interacción con la silla. Luego se dibuja en la pantalla la batería y se rellena con el porcentaje obtenido del hilo anterior y si la silla está en el punto de carga, se dibuja también un icono de un enchufe, antes de mostrar el mensaje, se le da formato mediante el método `formatMensaje(mensaje)` el cual adapta el mensaje al límite de caracteres que puede mostrar la pantalla.
Cuando existe una nueva interacción con la silla, cualquier método llama a `mostrarMensaje(_codigo)` éste busca en el diccionario el mensaje asociado al código que recibe y activa el evento para que el hilo encargado de manejar la pantalla muestre el nuevo mensaje, además envía al servidor MQTT el código al tema *sillas/{id}* donde id es el identificador único que devuelve la página al dar de alta la silla en el sistema.

La silla se puede mover desde la botonera del apoyabrazos o de forma remota por la aplicación web, en los dos casos el funcionamiento es parecido, en el momento que se presiona uno de los botones, si es físico, se lee el puerto GPIO por si es una interrupción fantasma, si realmente el botón está presionado o es remoto, se espera un segundo por si fue un accionamiento accidental, si durante ese segundo el botón sigue presionado y no hay ningún sensor presionado, se acciona el relé correspondiente al sentido de giro que se desea. Si durante el trayecto, se presiona el sensor de curva, el método encargado de tratar esa interrupción activa un relé que cambia la fuente de energía desde las baterías al regulador de tensión para bajar la velocidad. Si un sensor se activa o se suelta el botón la silla se detiene. En todos estos casos, se muestra la información por pantalla y se envía el estado al servidor.
Puede darse el caso en el que si un sensor es presionado y soltado muy rápidamente, el software no detecte este cambio y no permita mover el motor, por eso cada vez que se vaya a mover la silla, se leen el estado de todos los sensores para asegurarse si se puede llevar a cabo esta acción.
En la Ilustración 17 se pueden ver los casos de uso de este script, el usuario puede mover la silla a izquierda o derecha (subida o bajada dependiendo del lado de instalación) y presionar algún sensor, lo cual detiene la silla inmediatamente, todo ello envía un mensaje al servidor y muestra la información por pantalla.

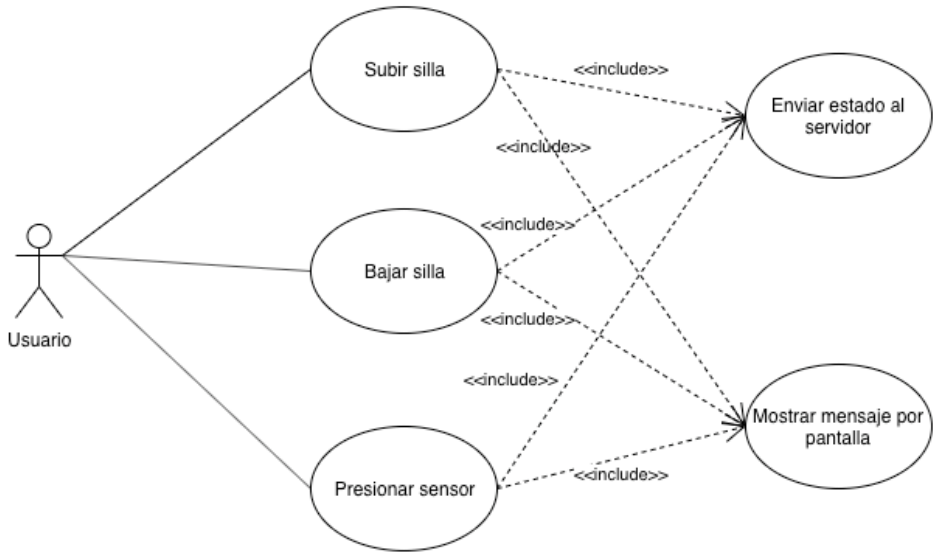


Ilustración 17: Casos de uso silla subescaleras

En el caso del envío de mensajes al servidor, se utiliza la siguiente codificación:

000#[Valores…#…]

Por ejemplo el mensaje generado para enviar el estado de las baterías:

400#98,5

En el servidor, dependiendo del código que llega al principio del mensaje, se transforma el resto a la forma necesaria para almacenarlo en la base de datos o enviarlo a la aplicación web, en el caso anterior sería *400* el código de la acción y *98,5* el porcentaje de carga actual.
En la ilustración 18 se puede ver un ejemplo de cómo trabaja el módulo de la silla interactuando con los hilos, el servidor MQTT y el hardware.

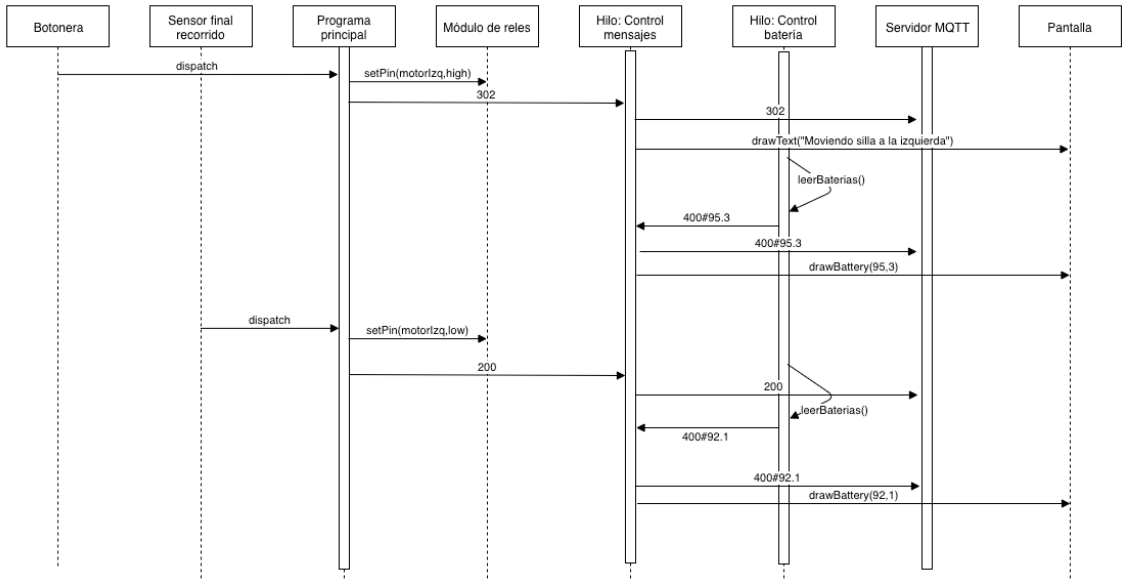


Ilustración 18: Ejemplo módulo silla subescaleras

3.3.2 Servidor

En el servidor hay tres servicios corriendo, el primero y más importante es el desarrollado en NodeJS que es el encargado interconectar todas las partes, segundo el broker MQTT, que gestiona el envío y la recepción de los mensajes y tercero la base de datos MongoDB.

3.3.2.1 Servicio NodeJS

Este servicio, como se ha indicado antes, es el encargado de hacer que todo se comunique entre sí.

Para brindar todas las funcionalidades, se crea un servidor http mediante la librería ExpressJS, este servidor escucha las conexiones en el puerto 3000, además, se ha configurado para servir las aplicación web en el mismo puerto.
Cuando se inicia, además de levantar el servidor http, se conecta a la base de datos, se conecta al broker Mosquitto y se suscribe al canal *sillas/#* para obtener todos los mensajes provenientes de las distintos estados y se crea el lado del servidor para las conexiones de socket.io. Cuando se recibe un mensaje de una silla, ésta se analiza y separa en las partes explicadas anteriormente, se insertan en el historial de la silla en la base de datos y se envía por socket.io a los temas *sillas/#* y *sillas/{id},* en el primer caso es para la vista general de las sillas, donde la aplicación debe separar el mensaje del id al que pertenece para actualizar la información necesaria y en el segundo caso es para la vista de detalle de la silla en concreto al que pertenece el id para igualmente mostrar la información e insertar al principio del historial mostrado.
La otra tarea importante de este servicio es brindar la API que consume la aplicación web y los scripts de las sillas, en esta API se consultan los mensajes de estado y toda la información correspondiente a las sillas, para esto último se necesita haber iniciado sesión en la aplicación, donde se genera un token, que es indispensable para consultar la información, de esta manera, se protegen los datos de los clientes y permite no alterar la información por terceros.

3.3.2.2 Servicio MQTT

Se ha instalado el bróker Mosquitto, además para proteger los mensajes se ha configurado para necesitar usuario y contraseña para conectarse a él, no se necesita definir la estructura de los temas, cuando un cliente se registra en un tema, si nadie más lo hace, esos mensajes no llegarán a ningún sitio, en este trabajo la estructura utilizada es la siguiente:

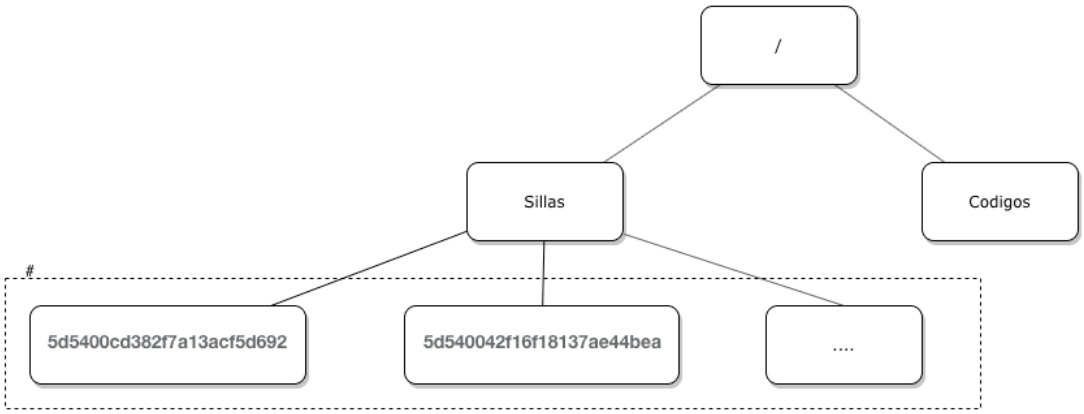


Ilustración 19: Diagrama de temas MQTT

3.3.2.3 Servidor MongoDB

Se ha seleccionado MongoDB para ser el gestor de base de datos para el presente trabajo debido a que todo el entorno web se ha desarrollado en JavaScript y la información se guarda en el mismo formato que utilizan tanto el servidor NodeJS como la vista en AngularJS. Además es muy rápido cuando hay mucha cantidad de datos como es el caso del historial de cada silla y en caso de necesitarse se podría escalar horizontalmente con cierta facilidad.
El diagrama de la base de datos utilizada se puede ver en la ilustración 20.

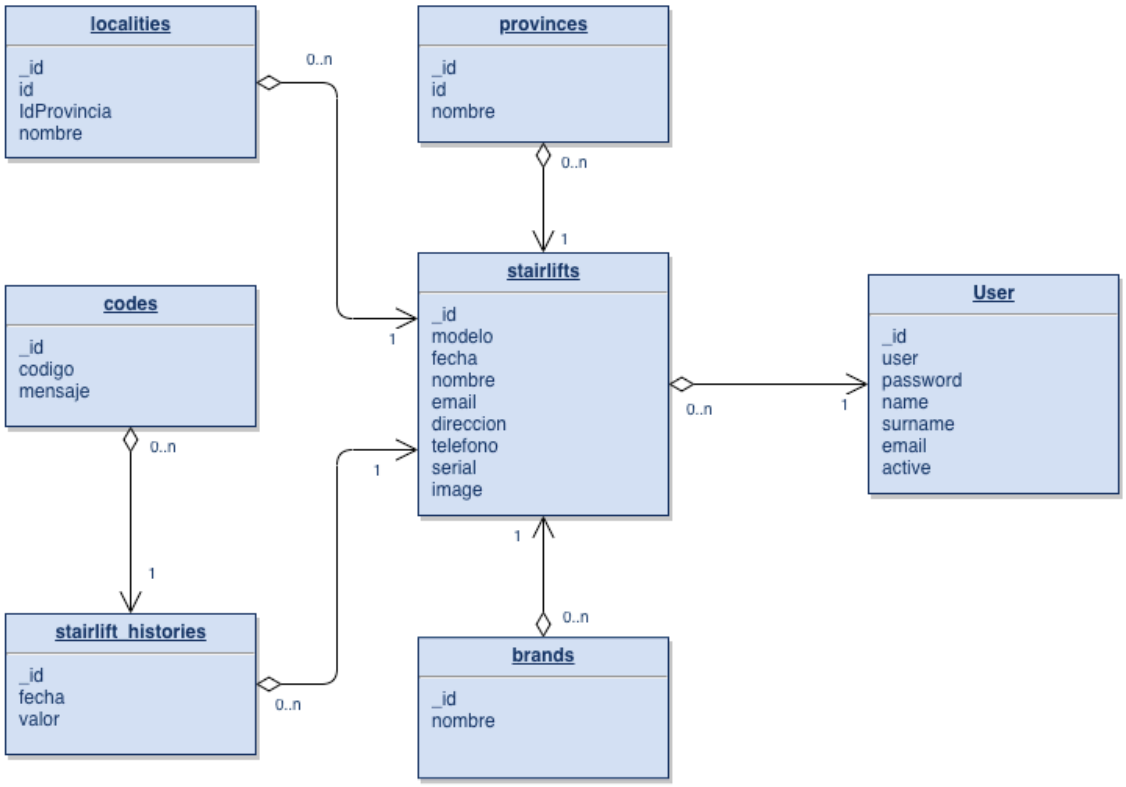


Ilustración 20: Diagrama de base de datos

3.3.3 Aplicación web

La aplicación web permite a los técnicos ver el estado de todas las sillas que tiene asociadas, los datos de los clientes y gestionar los mensajes que se muestran en las pantallas de las sillas.
La aplicación web está desarrollada con HTML5, CSS3 y AngularJS, esta ultima tecnología ayuda a que la aplicación sea más dinámica y con ayuda de Socket.IO permite visualizar los datos en tiempo real sin necesidad de recargar el navegador.
El primer paso para acceder es realizar la autenticación mediante usuario y contraseña, la contraseña se encripta con un salt y se compara con la que está

en la base de datos, si coincide, se genera un token que contiene el id del usuario, la fecha en formato unix del momento en el que se genera el token y la fecha en formato unix del momento en el que el token expira, en este caso está configurado para que caduque a las 8 horas. Una vez se inicia sesión, el servidor devuelve el token y la aplicación web guarda además de éste el usuario en el *Local Storage* del navegador web porque en cada petición que se hace a la API se debe enviar el token, éste se decodifica para poder autenticar que el usuario tiene permiso para hacer esa petición, además se comprueba la fecha de caducidad, si el token ya está caducado, se devuelve un código 401 y la página web redirige al usuario otra vez a la página de iniciar sesión. También puede darse el caso en el que el token que se recibe en el servidor sea corrupto o una persona sin acceso intente hacer peticiones a la API, en ese caso se devuelve un error 500 y también se redirige al usuario a la página de iniciar sesión. Si todo es correcto, se continúa el camino al método que hace la acción y devuelve el resultado con un código 200 y los datos en formato JSON.

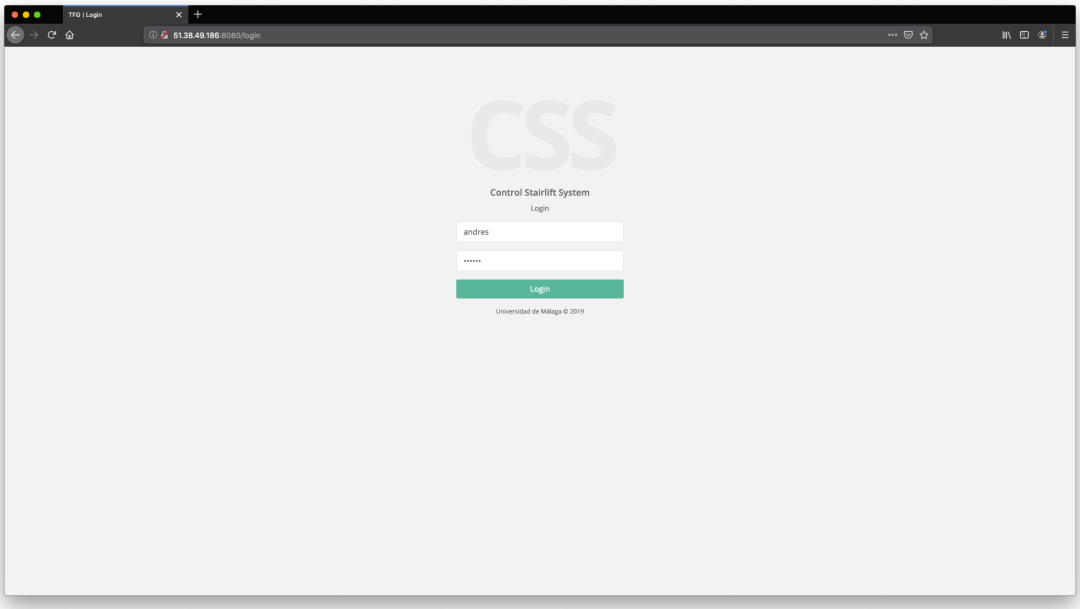


Ilustración 21: Pantalla de inicio de sesión

En la sección de códigos de estado, el usuario puede crear, editar y eliminar los códigos y su correspondientes mensajes, cuando alguna de estas acciones ocurre el servidor la ejecuta y envía un mensaje MQTT al tema *sillas/codigos* para que las sillas puedan llamar a la API para obtener la lista de códigos en formato diccionario. Como las sillas no inician sesión en el sistema esta acción es la única que no necesita un token para devolver información, porque además no contiene información sensible, a esta sección solo tienen acceso los usuarios de tipo *Administrador*.

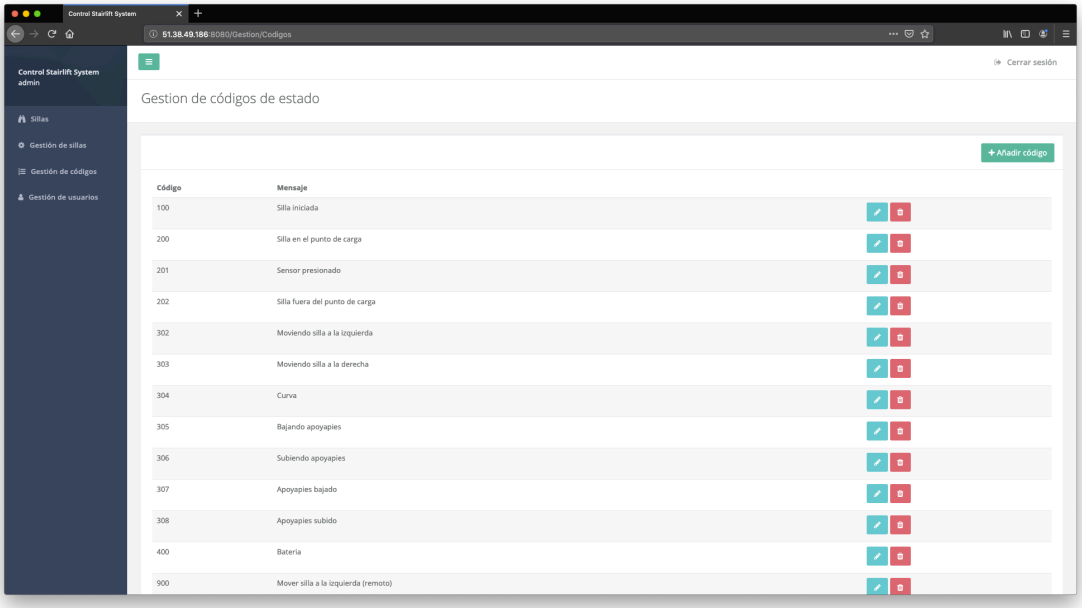


Ilustración 22: Pantalla de gestión de códigos

Otra sección donde sólo tienen acceso los usuarios de tipo Administrador es la gestión de usuarios, en ella se pueden dar de alta nuevos usuarios, editarlos o desactivarlos, en el caso en el que se vaya a añadir un usuario deben completar todos los campos como nombre de usuario, contraseña, tipo de usuario, etc. En el caso en el que se vaya a editar el usuario, la contraseña no es obligatoria, si no se indica una contraseña, el sistema no hará ningún cambio en ella. Al desactivar un usuario, se cambiar una variable de tipo booleano en la base de datos, la cual es comprobada en el momento del inicio de sesión, si la variable es *false*, el usuario recibirá un mensaje avisando que su usuario ha sido desactivado y no podrá acceder.

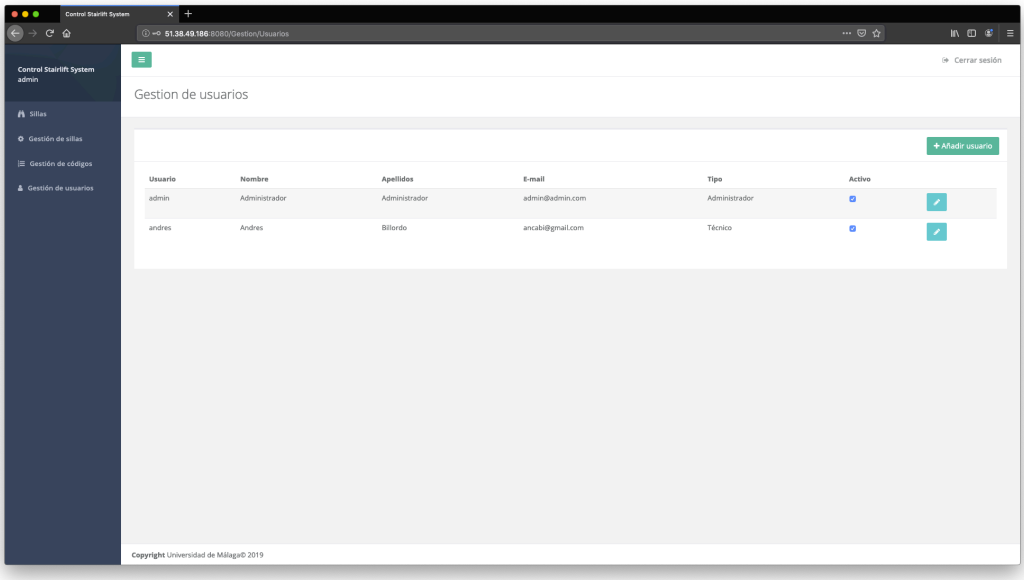


Ilustración 23: Pantalla de gestión de usuarios

La sección de gestión de sillas permite al usuario crear y editar las sillas, sólo se pueden gestionar las sillas que pertenecen al usuario que ha iniciado sesión, además de los datos de tipo texto como nombre, marca, modelo, teléfono, etc se solicita la provincia y localidad guardadas en la base de datos y una imagen para identificar más fácilmente de que silla se trata. En el caso en el que el usuario no suba ninguna imagen, el sistema utiliza una foto predeterminada para cada marca y modelo. Cuando el usuario crea una silla en la aplicación, al terminar el proceso, se le devuelve un ID que debe copiar en una variable dentro del script de la silla que va a instalar para poder identificar los mensaje MQTT que llegan desde ella.

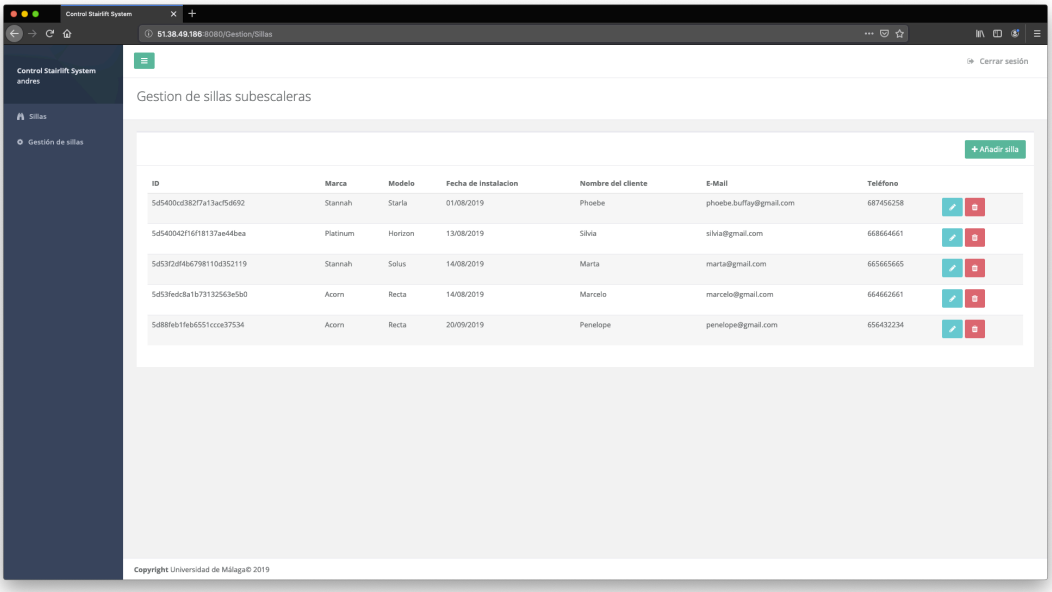


Ilustración 24: Pantalla de gestión de sillas

Cuando el usuario inicia sesión, se le redirige a la página donde puede ver el listado de todas las sillas que tiene configuradas, esta página se suscribe al tema sillas/# para recibir todos los mensajes de todas las sillas, cuando se recibe un mensaje se llama a un método encargado de buscar en todas las sillas que tiene el usuario y una vez que la encuentra, actualiza el mensaje o el estado de la batería dependiendo el código que se recibe. Además del estado se puede ver información básica del dispositivo como su ID, imagen, marca y modelo, para ver la información detallada se debe hacer clic en una de ellas.

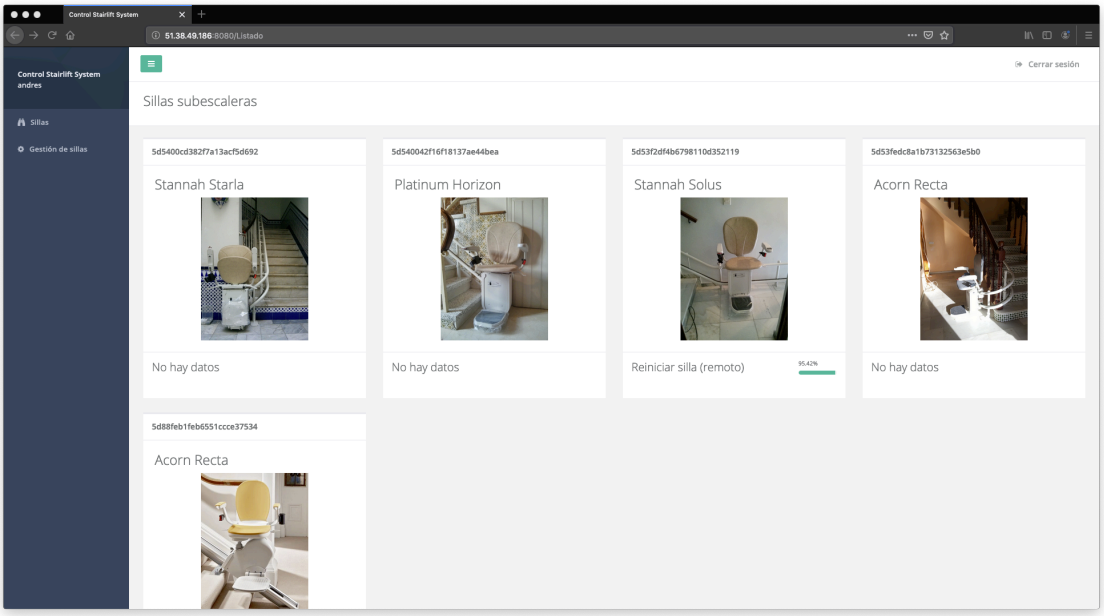


Ilustración 25: Pantalla de listado de sillas

En el detalle de las sillas, se ve toda la información referente a la silla y el cliente, el último estado y el nivel de batería, además de esto, tiene una sección de historial donde se puede filtrar por códigos, un intervalo de fechas y un número de registros que se desea ver, por defecto se ven los últimos 100 registros de todos los códigos de los últimos treinta días. Esta página al iniciarse se registra al tema *sillas/{id}* para obtener sólo los mensajes referentes a la silla que se está visualizando, cuando un nuevo mensaje llega se llama a un método encargado de actualizar el último estado o el nivel de batería dependiendo del código que se recibe, además si este mensaje está incluido en los filtros definidos, se inserta como un nuevo registro en la tabla del historial.

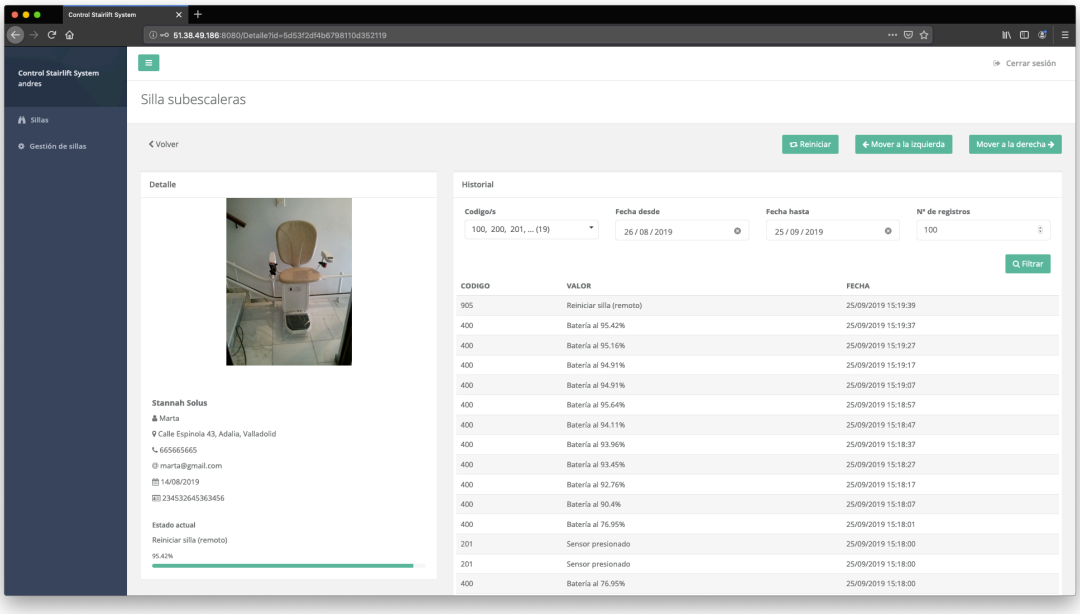


Ilustración 26: Pantalla de detalle de silla

Desde esta última página se puede controlar la silla a distancia, esto puede ser útil por ejemplo si el interruptor se rompe, para ello el usuario tiene dos botones en la parte superior, los cuales en el momento en el que se presionan envían un código a la silla por MQTT (900 o 901 para mover la silla a izquierda o derecha respectivamente) mediante el evento ng-mousedown de AngularJS, la silla se comporta de igual manera que si se hubiese presionado el botón físico, los sensores funcionan de igual manera por seguridad, para que si algo se pusiese en el camino no se lastime la silla o el obstáculo, aunque en el anteproyecto se haya indicado lo contrario. Cuando el usuario suelta el botón, se dispara el evento ng-mouseup que envía otro código a la silla (902 en este caso) para detener el motor. Cuando la silla se mueve de forma remota, éste envía un mensaje de vuelta (903 o 904) para avisar al usuario que se ha recibido la orden y la silla se está moviendo ya que el técnico puede estar a mucha distancia y no tener forma de comprobarlo, en el momento en el que la silla deja de moverse envía un mensaje si la silla está o no en el punto de carga para que el usuario sepa si ha quedado cargando o no. A la izquierda de estos botones hay un tercer botón que permite reiniciar el sistema operativo del Raspberry pi, esta función es útil por si el software se ha quedado en algún error. Para lograr esto, la aplicación web envía un código 905 a la silla, la cual cuando lo recibe, vuelve todos los GPIO's al estado inicial y reinicia el sistema.

4

Fase de pruebas

4.1 Introducción

En este capítulo se explican las pruebas por las que ha pasado este trabajo para asegurarse el correcto funcionamiento de todos los componentes desarrollados.
Se han seleccionado las siguientes pruebas:

- Pruebas unitarias
- Pruebas de integración
- Pruebas de rendimiento

4.1 Pruebas unitarias

Se ha diseñado una batería de pruebas unitarias para comprobar el correcto funcionamiento de la API para ello se ha utilizado la librería *chai* que es una librería de aserciones que se puede emparejar con cualquiera de las librerías de pruebas de JavaScript, en este caso se ha elegido Mocha[9] porque es el más extendido y fácil de utilizar, además de ser muy potente.
En el diseño de las pruebas se llama a los servicios y se comprueba que el estado de la petición sea el correcto para los datos proporcionados, el siguiente ejemplo es uno de los más complejos:

```
describe('Editar una silla: ', () => {

    it('Debería editar una silla en la base de datos', (done) => {
        chai.request(url)
            .get('/sillas')
            .set('Authorization', 'Bearer '+token)
            .end(function (err, res) {
                expect(res.body.sillas).to.have.lengthOf(1);
                expect(res).to.have.status(200);
                res.body.sillas[0].nombre = 'Editado';
                chai.request(url)
                    .put('/silla')
                    .send({ silla: res.body.sillas[0] })
                    .set('Authorization', 'Bearer '+token)
                    .end(function (err, res) {
                      expect(res).to.have.status(200);
                      chai.request(url)
                        .get('/sillas')
                        .set('Authorization', 'Bearer '+token)
                        .end(function (err, res) {
                          expect(res.body.sillas[0].nombre).to.eq("Editado");
                          expect(res).to.have.status(200);
                          done();
                        });
                    });
            });
    });
});
```

Se puede ver como se define el caso de prueba mediante la palabra *describe* y luego el caso mediante la palabra *it*, una vez dentro se hace una petición al servicio y se compara con lo que se espera, por ejemplo en el primer caso se comprueba que el array devuelto tiene exactamente un objeto y que el código de la respuesta fue 200 (OK), luego se llama al servicio que se quiere probar y por último se vuelve a llamar al primer servicio para comprobar que el cambio se ha hecho con éxito.
Como se puede ver en la ilustración 27 se han hecho 27 pruebas unitarias y todas han sido satisfactorias.

```
[ancabi@debianProyecto:~/sillaServer$ mocha app/tests/*.js --timeout 15000


  Hacer login:
    ✓ debería hacer login (114ms)

  Usuario no existe:
    ✓ no debería hacer login

  Contraseña incorrecta:
    ✓ no debería hacer login (83ms)

  Obtener sillas (Comprobar token):
    ✓ No debería permitir hacer la consulta

  Obtener las sillas:
    ✓ Debería devolver el listado de sillas

  Obtener silla (Comprobar token):
    ✓ No debería permitir hacer la consulta

  Obtener las silla:
    ✓ Debería devolver el detalle de una silla

  Obtener provincias (Comprobar token):
    ✓ No debería permitir hacer la consulta

  Obtener las provincias:
    ✓ Debería devolver el listado de provincias

  Obtener localidades (Comprobar token):
    ✓ No debería permitir hacer la consulta

  Obtener las localidades:
    ✓ Debería devolver el listado de las localidades

  Obtener marcas (Comprobar token):
    ✓ No debería permitir hacer la consulta

  Obtener las marcas:
    ✓ Debería devolver el listado de las marcas

  Obtener codigos:
    ✓ Debería permitir hacer la consulta y devolver los codigos

  Obtener codigos en formato diccionario:
    ✓ Debería permitir hacer la consulta y devolver los codigos

  Añadir una silla:
    ✓ Debería crear una silla en la base de datos

  Añadir una silla (Comprobar token):
    ✓ Debería crear una silla en la base de datos

  Editar una silla:
    ✓ Debería editar una silla en la base de datos (98ms)

  Editar una silla (Comprobar token):
    ✓ No debería editar una silla en la base de datos

  Eliminar una silla:
    ✓ Debería eliminar una silla en la base de datos (44ms)

  Eliminar una silla (Comprobar token):
    ✓ No debería eliminar una silla en la base de datos

  Añadir un codigo:
    ✓ Debería crear un codigo en la base de datos

  Añadir un codigo (Comprobar token):
    ✓ Debería crear un codigo en la base de datos

  Editar un codigo:
    ✓ Debería editar un codigo en la base de datos

  Editar un codigo (Comprobar token):
    ✓ No debería editar un codigo en la base de datos

  Eliminar un codigo:
    ✓ Debería eliminar el unico codigo que hay en base de datos

  Eliminar un codigo (Comprobar token):
    ✓ No debería eliminar un codigo en la base de datos


  27 passing (619ms)
```

Ilustración 27: Resultado de las pruebas unitarias

4.2 Pruebas de integración

Las pruebas de integración se utilizan para comprobar la correcta interconexión de las distintas partes de un sistema y su correcto funcionamiento.
Para este trabajo se han diseñado las pruebas de integración con la librería *Mocha*, sobre ella se han creado tres pruebas para comprobar la conexión con la base de datos, con el bróker MQTT y con una silla respectivamente. La forma de probar cada una fue:

- En el caso de la base de datos, se crea una nueva conexión, cuando ésta es exitosa, se da por buena la prueba.
- En el caso del bróker MQTT se crea la conexión, se suscribe al tema *test/sillas* y se envía un mensaje, si ese mensaje se recibe es que la conexión es correcta.
- En el caso de la conexión con una silla subescaleras, se crea una conexión con el bróker MQTT, se suscribe al tema test/sillas/{id}, donde id corresponde con el id de una silla en concreto y se envía un mensaje 'Prueba'.
  Para que la prueba se pueda llevar a cabo, se ha añadido un pequeño bloque de código en el script de las sillas para que, en el caso de recibir un mensaje en el tema mencionado anteriormente y que el mensaje sea 'Prueba', enviar por ese mismo tema un mensaje 'OK'. En el caso de recibir ese mensaje en el servidor, la prueba sería satisfactoria.

En la ilustración 28 se pueden ver las tres pruebas realizadas satisfactoriamente.

```
[ancabi@debianProyecto:~/sillaServer$ mocha app/tests/testsIntegración.js --timeout 15000


  Conexión con la base de datos:
    ✓ Debería conectar (47ms)

  Conexión con el broker MQTT:
    ✓ Debería conectar (99ms)

  Envío de mensajes MQTT:
    ✓ Debería enviar y recibir un mensaje (56ms)

  Envio y recepcion de mensajes MQTT con una silla:
    ✓ Debería enviar un mensaje a la silla y recibir un mensaje de la silla (47ms)


  4 passing (284ms)
```

Ilustración 28: Resultado de las pruebas de integración

4.3 Prueba de rendimiento

Las pruebas de rendimiento se utilizan para ver cómo se comporta un sistema cuando tiene el número esperado de peticiones y cómo se comporta en un caso en el que hay más peticiones de las esperadas, de esa manera se puede probar y mejorar el sistema para responder en un caso extremo.
Para probar esto se ha generado un pequeño script que genera 200 sillas, las guarda en la base de datos y envía mensajes MQTT cada pocos segundos para así poder comprobar cómo reaccionan todos los componentes, es un caso extremo ya que es muy difícil llegar a tener 200 sillas para un sólo usuario pero además es muy raro que las 200 sillas envíen mucha información a la vez.
El rendimiento del sistema tras estar funcionando durante cinco horas fue excelente, no hubo ni una sola caída de rendimiento, los mensajes se han actualizado en tiempo real y el servidor respondió a todas las peticiones en el mismo tiempo que en las pruebas anteriores.
La única parte negativa fue por parte del navegador web, ya que al mostrar tantas imágenes, la velocidad de respuesta se vio comprometida, como futura mejora se debería añadir una opción para poder deshabilitar la vista de las imágenes.
En la ilustración 29 se puede ver la consola del servidor respondiendo a las peticiones y el navegador mostrando la mayor cantidad posible de sillas.

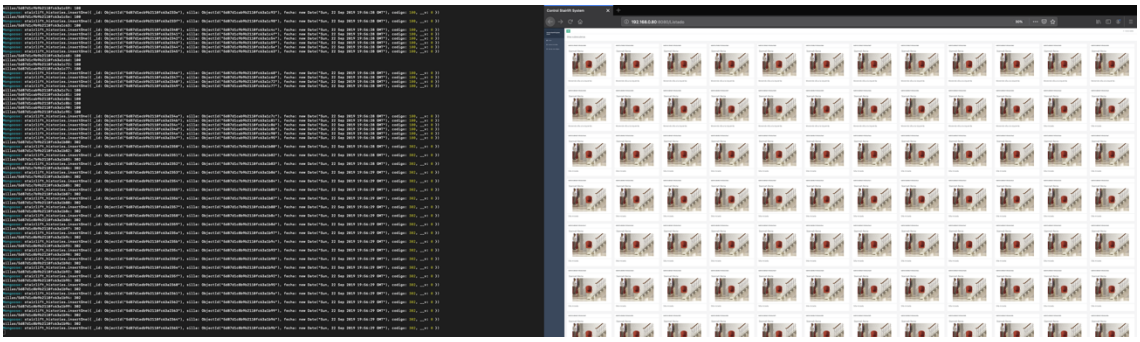


Ilustración 29: Resultado de las pruebas de rendimiento

5

Conclusiones

Una vez finalizado el trabajo puedo valorar lo importante que ha sido, me ha permitido afianzar muchos conocimientos que he adquirido en la carrera en materias como electrónica, física, pruebas software, sistemas empotrados, ingeniería web, etc. Además de aprender muchos conceptos nuevos para poder adaptar la idea inicial a un producto tangible.

Como se fue demostrando en la memoria, se ha logrado llevar a cabo todo lo que se ha indicado en el anteproyecto, primero se ha analizado el problema actual y se ha diseñado una solución acorde a los requisitos planteados, luego, se ha estudiado el producto actual para ver qué reutilizar y cómo interactuar con esos componentes y por último se ha estudiado varias posibilidades para abordar una solución óptima.

El trabajo, como se ha indicado, logra una solución a la necesidad actual pero eso no significa que sea un producto sin ninguna carencia, durante el trabajo se han identificado posibles mejoras para desarrollos futuros, como por ejemplo:

- Un sistema que permita actualizar el script de forma remota
- Buscar una alternativa al Raspberry Pi. La elección de este componente creo que fue la correcta para el trabajo pero se han utilizado casi todos los GPIO por lo que si se necesita poner algún otro sensor o alguna nueva funcionalidad habría que añadir componentes adicionales pero se han probado y no tienen la misma velocidad que los GPIO nativos.
- Añadir una pantalla táctil con un menú para que el cliente pueda configurar su red Wi-Fi por si cambia de compañía o hacer algún tipo de chequeo del estado de la silla.

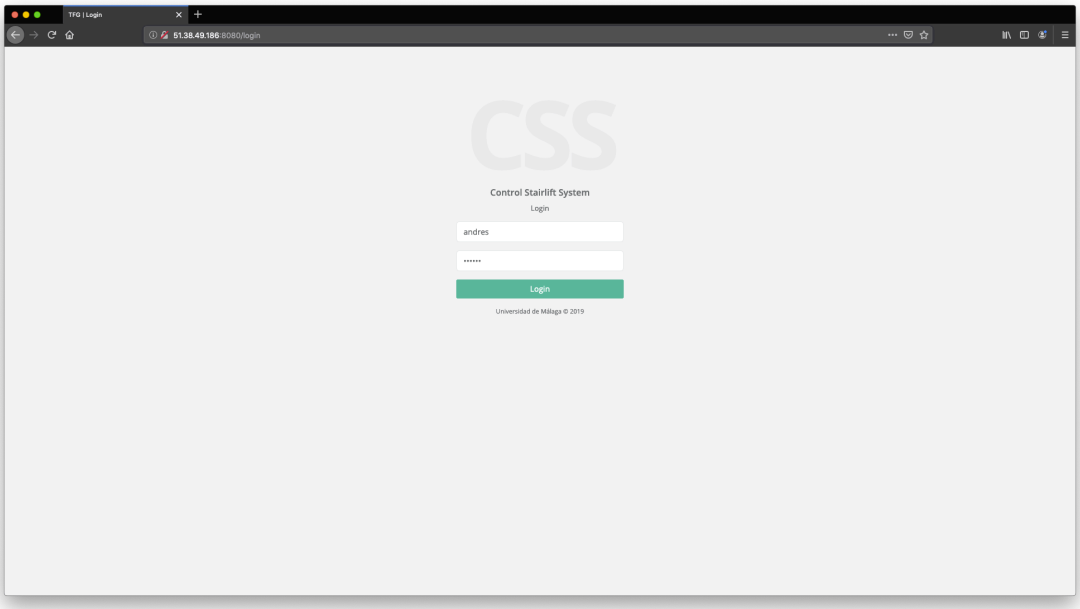
Referencias

[1] Raspberry Pi [en línea] https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/ [Consulta 02/08/2019]
[2] Wikipedia [en linea] https://es.wikipedia.org/wiki/Raspberry_Pi [Consulta 07/08/2019]
[3] GPIO en RaspberryPi [en linea] https://www.raspberrypi.org/documentation/usage/gpio/ [Consulta 08/08/2019]
[4] MQTT.js [en linea] https://github.com/mqttjs/MQTT.js [Consulta 08/08/2019]
[5] MQTT [en línea] https://www.ibm.com/developerworks/ssa/library/iot-mqtt-why-good-for-iot/index.html [Consulta 10/08/2019]
[6] Mosquitto [en línea] https://mosquitto.org/ [Consulta 13/09/2019]
[7] Python [en línea] https://docs.python.org [Consulta 14/09/2019]
[8] Ilustración 2: Partes de una silla subescaleras [en línea] http://www.nystairliftinstall.com/parts_stairlift.html [Consulta 17/09/2019]
[9] MochaJS [En línea] https://mochajs.org/ [Consulta 20/09/2019]

Apéndice A

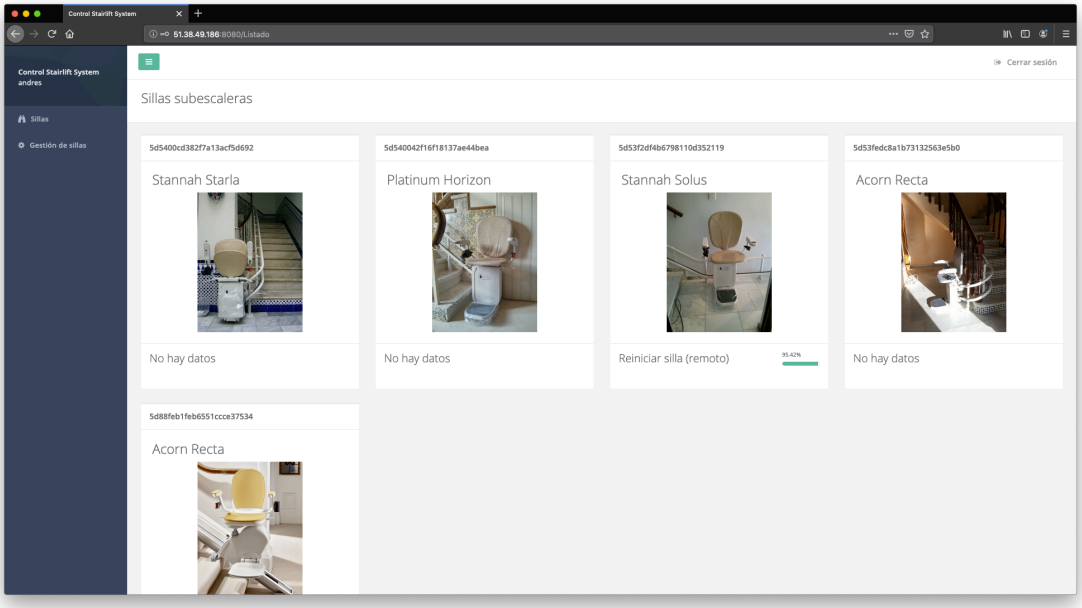
Manual de la aplicación web

1. Inicio de sesión



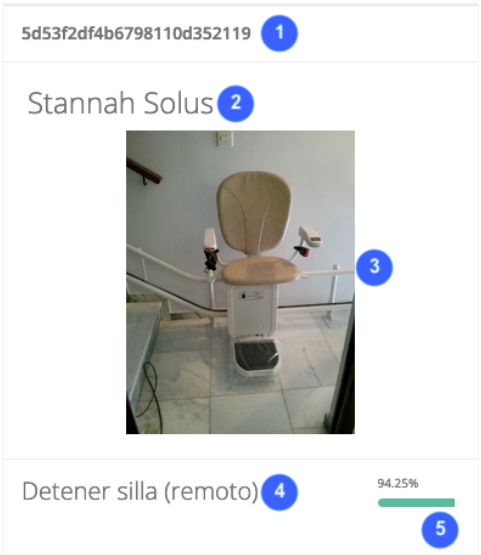
Al entrar en la aplicación se ingresa a la página de inicio de sesión, en ella se debe rellenar usuario y contraseña, una vez relleno, se debe presionar el botón Login. Si los datos son correctos, se accede a la página del listado de las sillas configuradas.

2. Listado de sillas



En todas las sucesivas páginas el usuario tendrá disponible a la izquierda el menú principal para poder acceder a las distintas secciones.

En esta página el usuario puede ver las sillas que tiene configuradas con los datos básicos para poder diferenciarlas y el estado actual de cada silla.

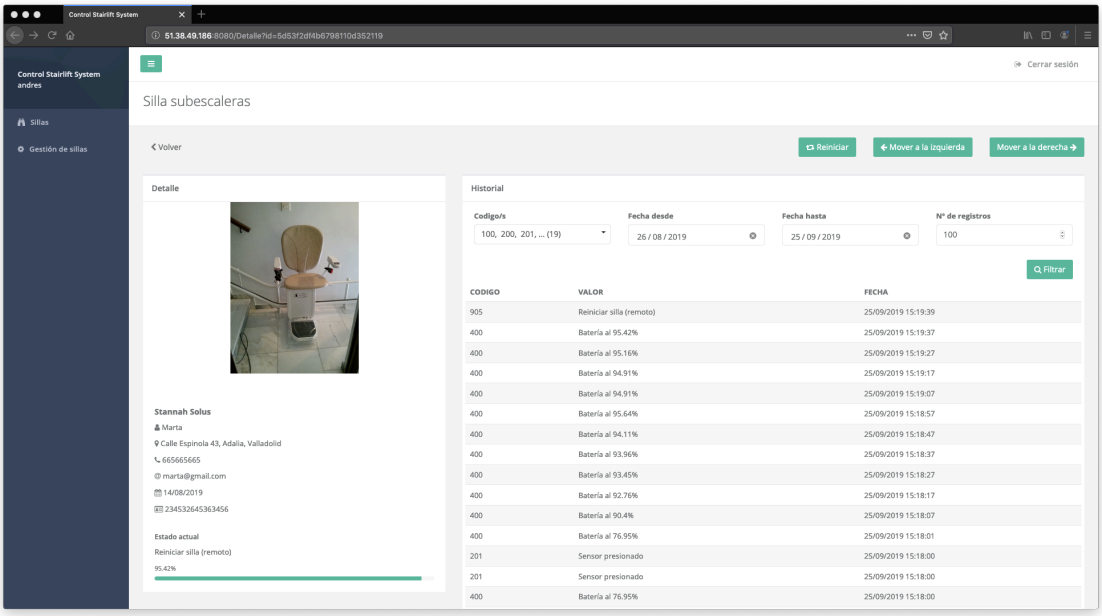


1. Id único de la silla
2. Marca y modelo
3. Imagen
4. Último estado
5. Nivel de batería

Los elementos 4 y 5 se actualizan en tiempo real a medida que la silla envía su información por internet.

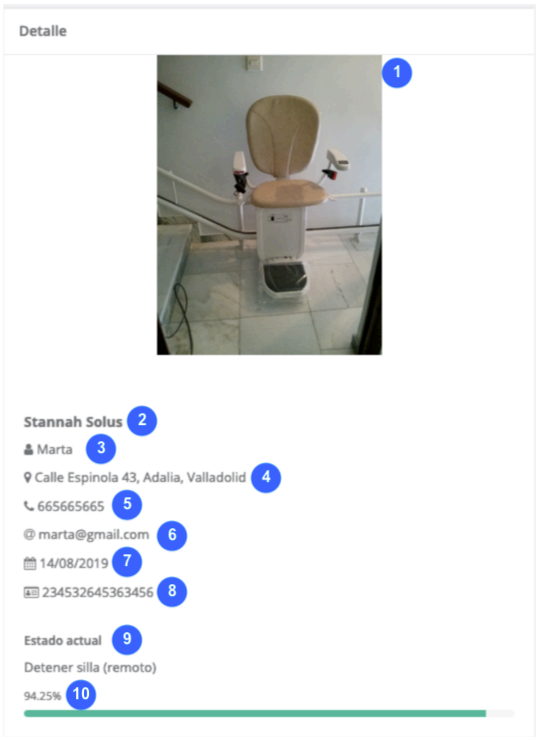
Si el usuario hace clic en uno de estos elementos se redirige a Detalle de silla subescaleras (3)

3. Detalle de silla subescaleras



En esta página el usuario puede ver los detalles de la silla, su historial y los botones de control remoto, estos últimos están situados en la parte superior derecha de la pantalla. Para controlar la silla remotamente el usuario debe presionar y mantener presionado el botón para que la silla se mueva para el lado indicado, una vez suelte el botón, la silla se detendrá. Para saber si la orden se está ejecutando en el estado actual aparece un mensaje indicando que la silla se está moviendo de forma remota, una vez se detiene, indica si está o no en el punto de carga. Cuando la silla se mueve de forma remota, los sensores funcionan de igual manera a cuando se mueve con los botones situados en el apoyabrazos de la silla.
A la izquierda de estos botones está situado el botón de Reiniciar, si el usuario presiona este botón, la silla seleccionada reiniciará el sistema operativo y volverá estar operativa.

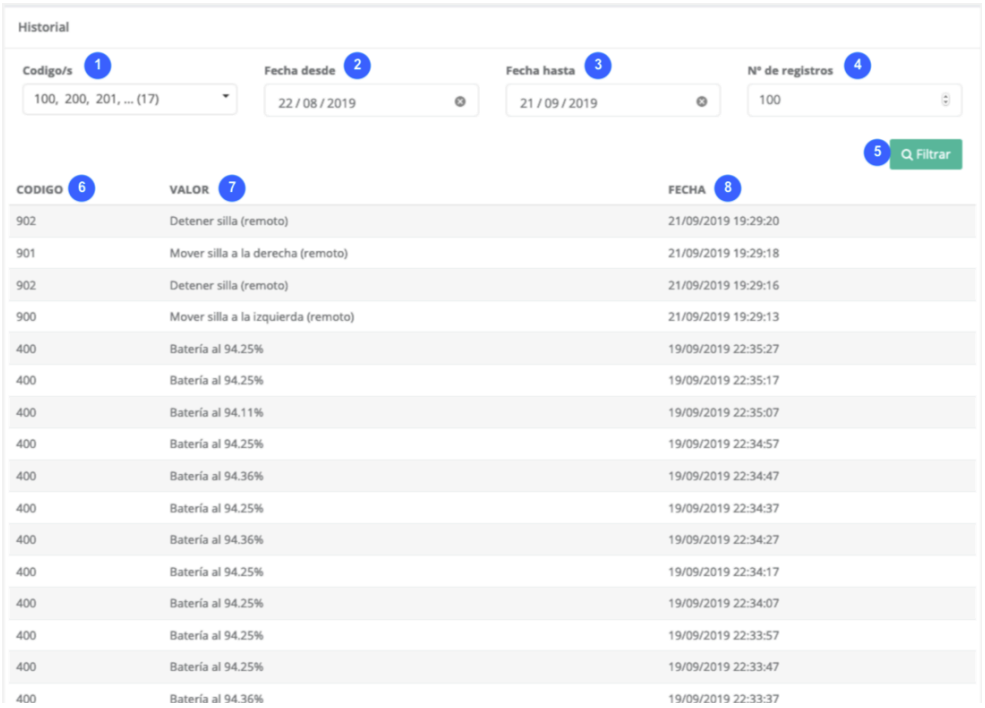
3.1 Detalle



En el lado izquierdo está el detalle de la silla seleccionada, el usuario puede ver los siguientes datos:

1. Imagen
2. Nombre del cliente
3. Dirección de la silla
4. Teléfono del cliente
5. E-Mail del cliente
6. Número de serie de la silla
7. Estado actual
8. Nivel de batería

3.2 Historial



En este bloque el usuario puede ver el historial de la silla, en la parte superior tiene los filtros (1,2,3 y 4) en el control 1 puede seleccionar los códigos de los mensajes que desea filtrar, en el control 2 puede seleccionar la fecha de inicio de los eventos, en el control 3 puede seleccionar la fecha de fin de los eventos y en el control 4 puede indicar el número de registros que desea visualizar. Una vez indicada la información el usuario debe presionar el botón Filtrar(5).

IMPORTANTE

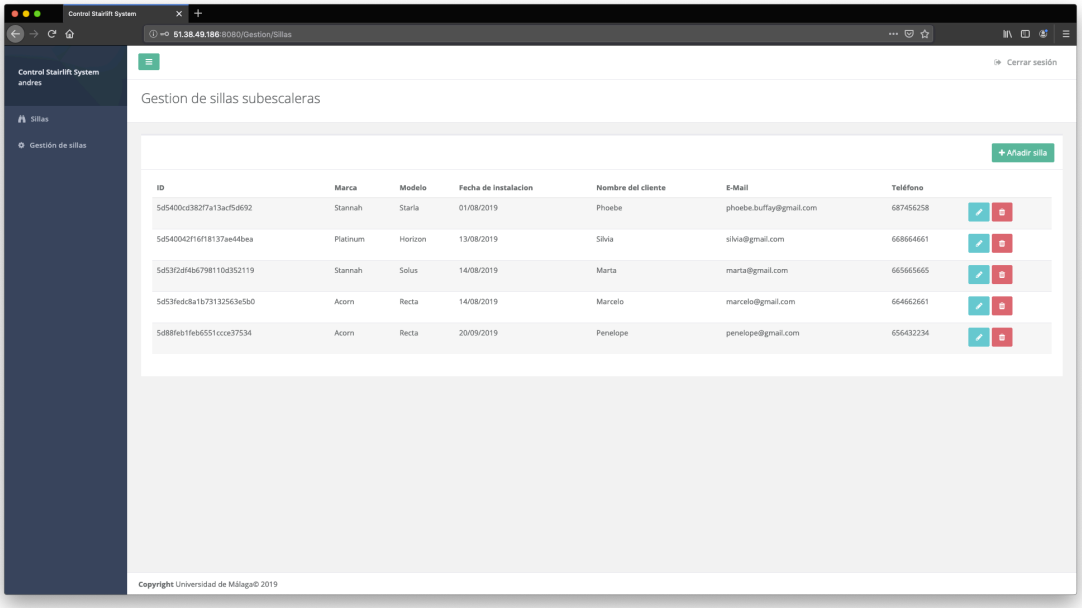
- Al menos debe seleccionar un código
- Las fechas son obligatorias
- La fecha de fin no pude ser anterior a la fecha de inicio
- El número de registros mínimo es 1

El resultado del filtro anterior se puede ver en la tabla inferior, en ella el usuario visualiza:

6. Los códigos de los eventos
7. El mensaje asociado a ese código y su valor si procede
8. La fecha del evento

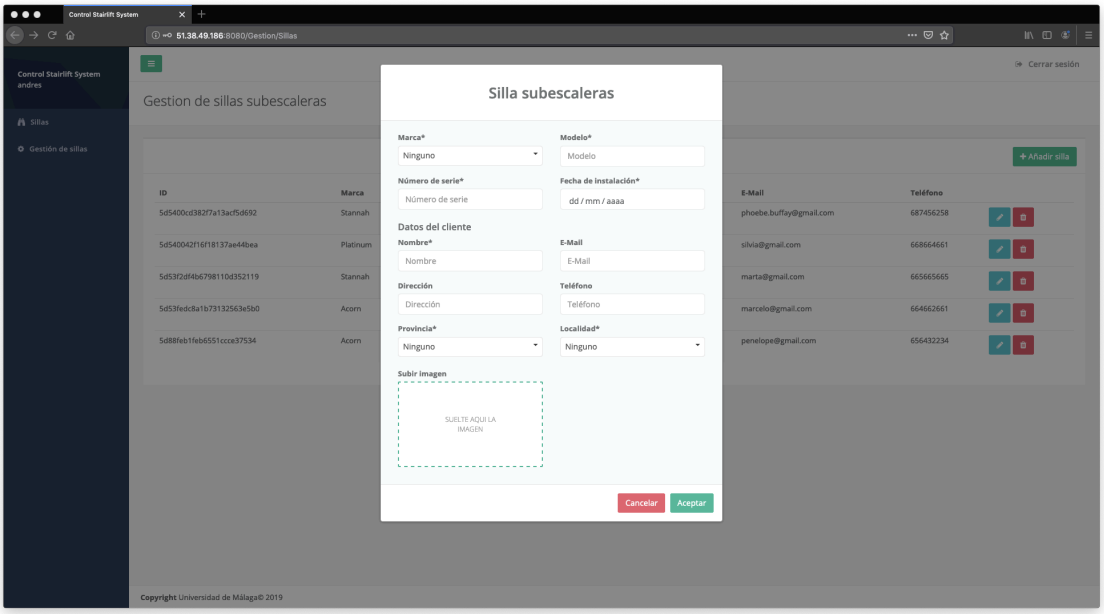
En la parte inferior de la tabla el usuario puede cambiar de página, la página muestra un máximo de 20 registros.

4 Gestión de sillas subescaleras



En esta sección el usuario puede gestionar sus sillas, en la parte superior derecha está situado el botón Añadir silla (4.1), en la parte central de la pantalla está situada la tabla con la información básica de las sillas y los botones de editar (4.2) y eliminar (4.3).

4.1 Añadir silla

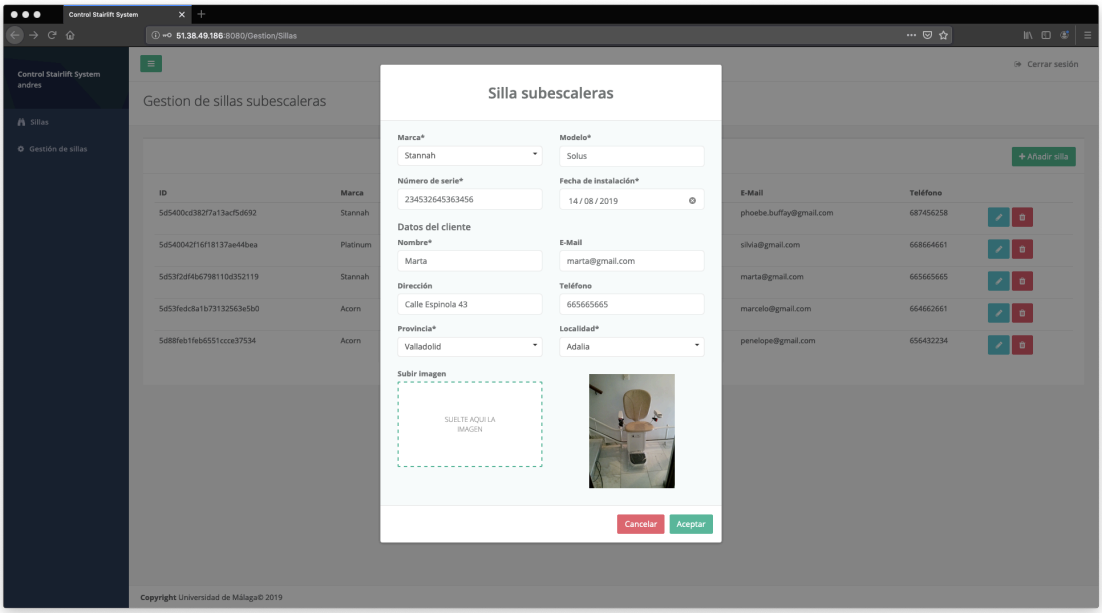


En esta ventana emergente el usuario debe rellenar los datos obligatorios marcados con un asterisco, los datos son:

- Marca
- Modelo
- Número de serie
- Fecha de instalación
- Nombre del cliente
- Provincia
- Localidad

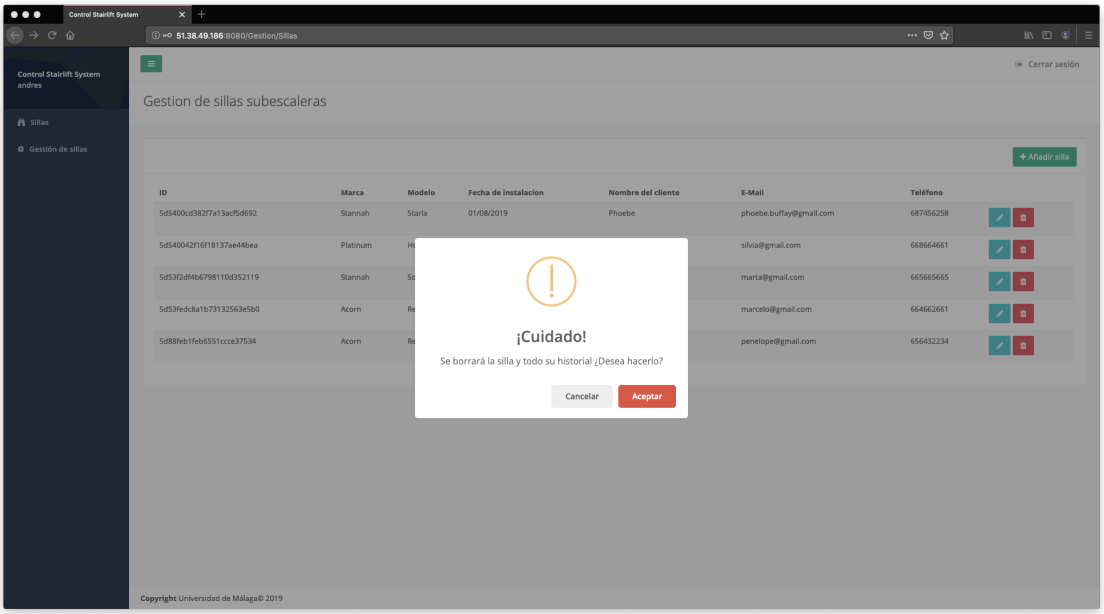
Además en el cuadro inferior izquierdo se puede arrastrar o seleccionar una imagen para asociarla a la silla y que sea más fácil identificarla.

4.2 Editar silla



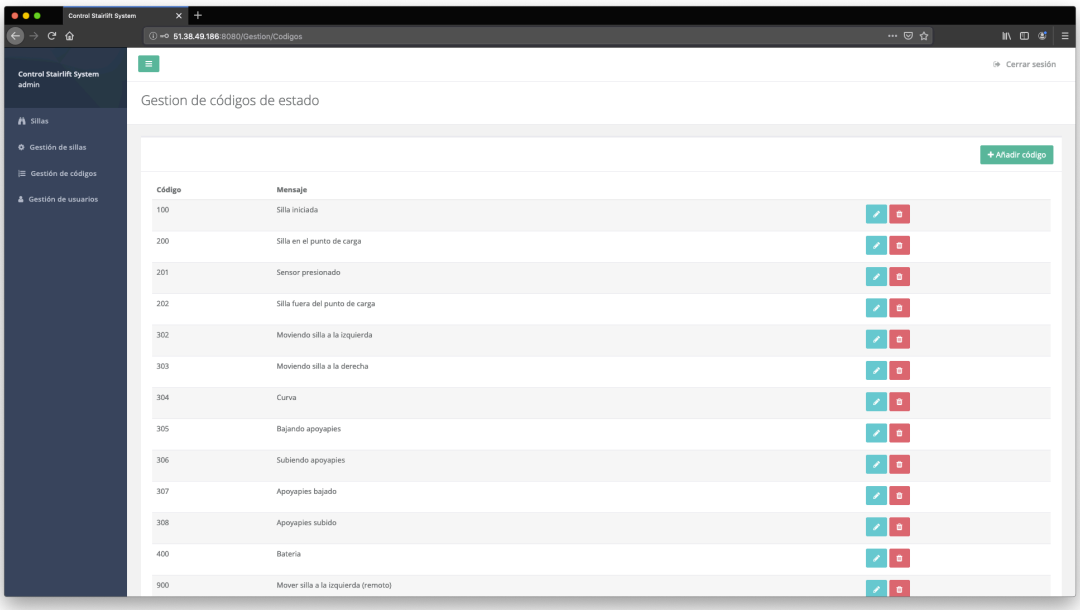
En esta sección el usuario puede editar la información relativa a una silla, los campos obligatorios, marcados con un asterisco no se pueden dejar en blanco (Véase la sección 4.1 para ver la lista de los datos obligatorios). La imagen destacada se puede cambiar pero no eliminar.
Si el usuario presiona el botón rojo (Cancelar) todos los cambios no se guardarán.

4.3 Eliminar silla



Cuando el usuario presiona el botón de eliminar correspondiente a una de las sillas, se lanza una ventana emergente para confirmar que el usuario desea eliminar esta silla, una vez eliminada los datos no se pueden recuperar.
El borrado de una silla implica quitar los datos de la silla y todo su historial.
Si el usuario presiona el botón gris (Cancelar) no se lleva a cabo ninguna acción, en cambio al presionar el botón rojo (Aceptar) el sistema elimina por completo toda la información referente a la silla seleccionada.

5 Gestión de códigos

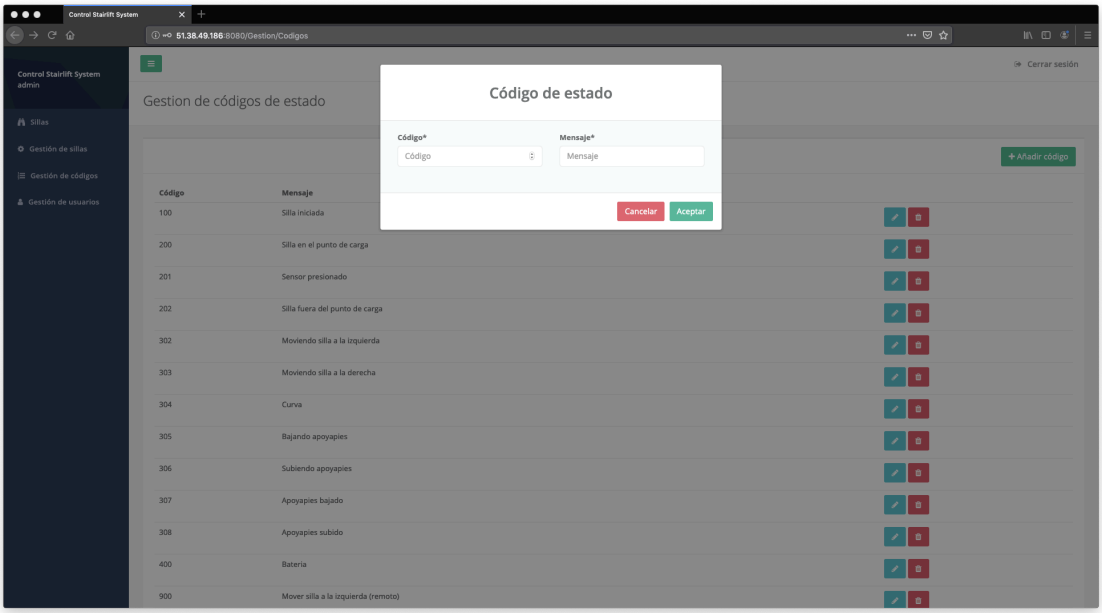


En esta sección el usuario puede gestionar los códigos, en la parte superior derecha está situado el botón Añadir código (5.1), en la parte central de la pantalla está situada la tabla con los códigos existentes y los botones de editar (5.2) y eliminar (5.3).

IMPORTANTE

- El código es único, si se intenta repetir un código existente, informará de un error y no permitirá guarda el nuevo código.

5.1 Añadir código

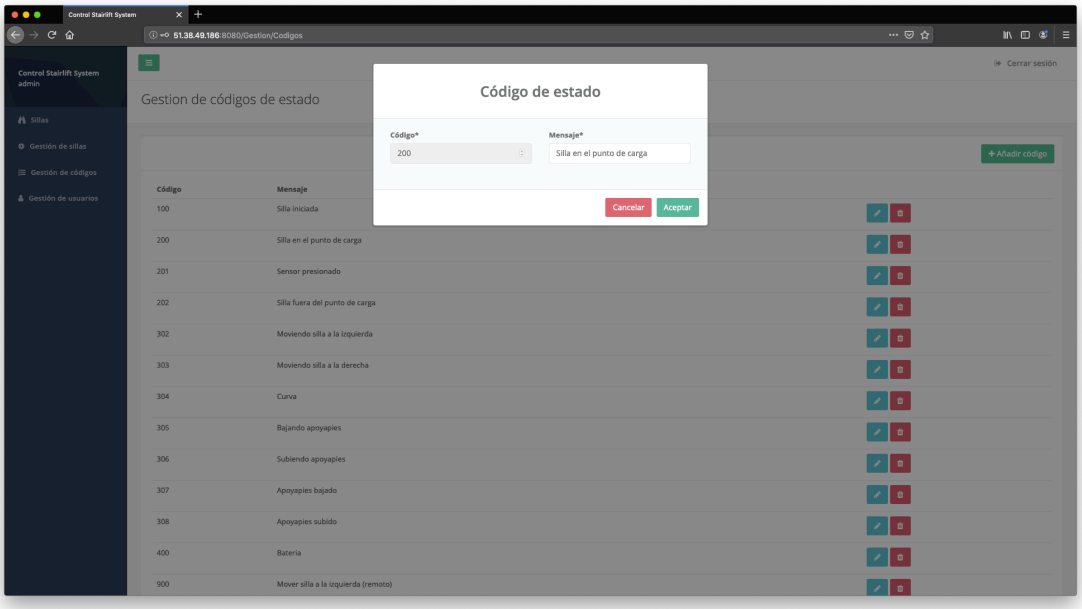


En esta ventana emergente el usuario debe rellenar los datos referentes al nuevo código que se va a insertar:

- Código: Es un valor numérico que identifica al código
- Mensaje: Es una cadena que ayuda al usuario a identificar el mensaje recibido por una silla.

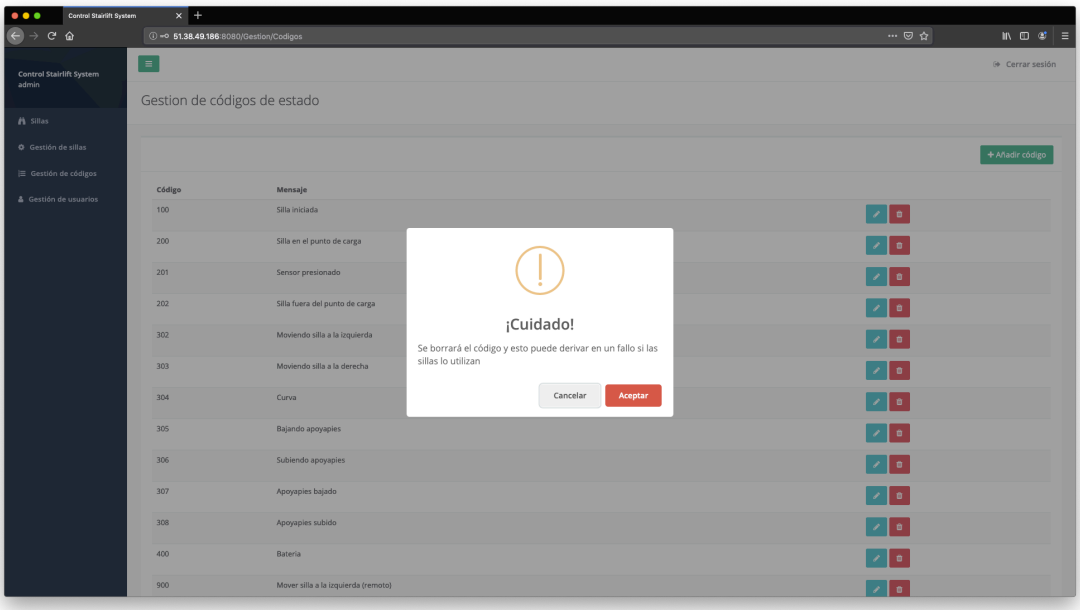
Ambos campos son obligatorios.

5.2 Editar código



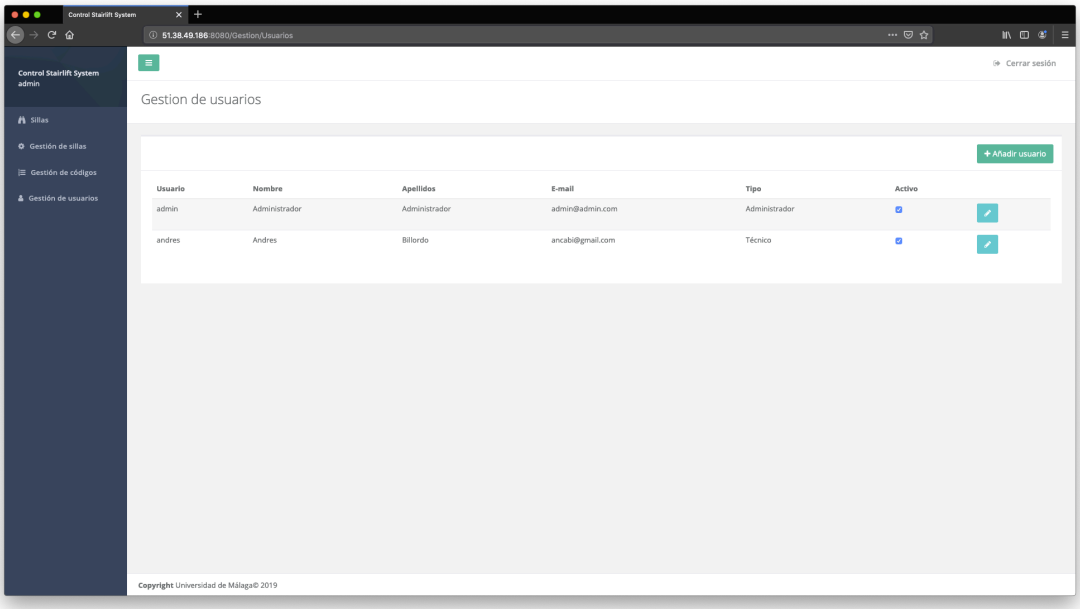
En esta sección el usuario puede editar el mensaje asociado a un código.
Si el usuario presiona el botón rojo (Cancelar) los cambios no se guardarán.

5.3 Eliminar código



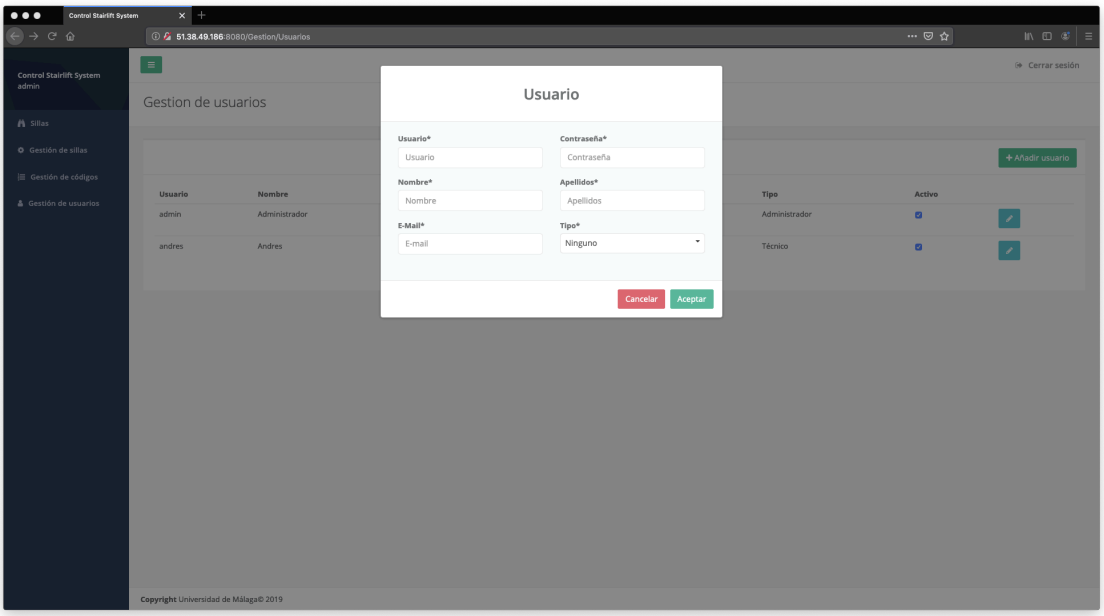
Cuando el usuario presiona el botón de eliminar correspondiente a uno de los códigos, se lanza una ventana emergente para confirmar que el usuario desea eliminar este código, como se puede ver en la imagen, si se elimina un código que las sillas puedan utilizar, causaría un fallo en las mismas.
Si el usuario presiona el botón gris (Cancelar) no se lleva a cabo ninguna acción, en cambio al presionar el botón rojo (Aceptar) el sistema elimina el código seleccionado.

6 Gestión de usuarios



Los usuarios tipo administrador tienen acceso a la sección para gestionar los usuarios, arriba a la derecha esta situado el botón Añadir usuario (6.1) y en cada línea hay una casilla para activar/desactivar un usuario y el botón para editar(6.2).
Al hacer clic en la casilla, se desactiva el usuario, impidiendo el acceso a la plataforma, para volver a permitir el acceso debe marcar la casilla.

6.1 Añadir usuario



Al añadir un usuario se deben rellenar todos los datos requeridos:

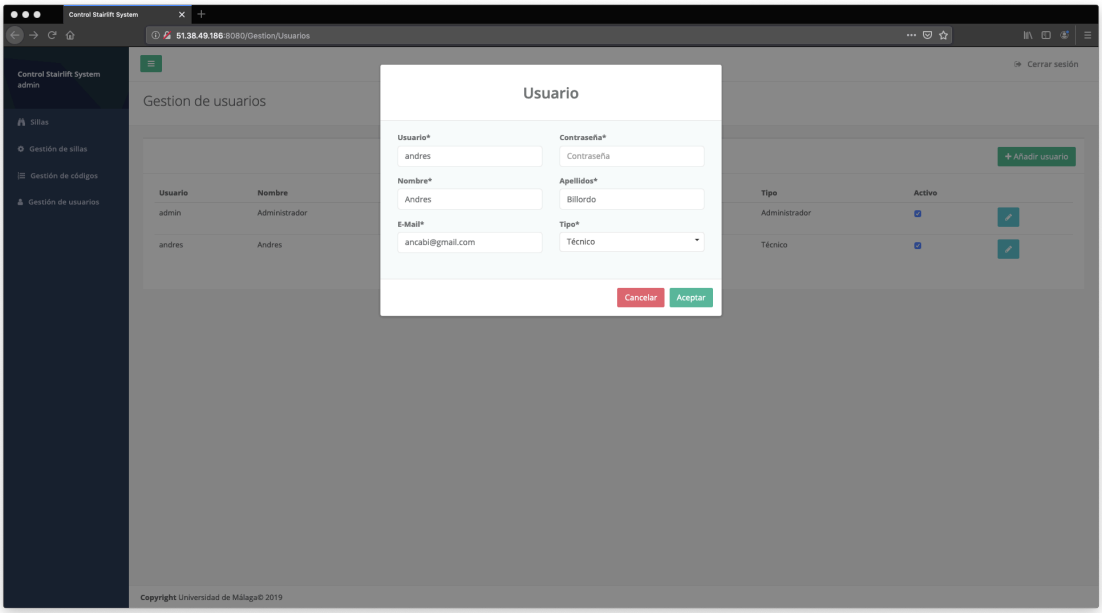
- Usuario
- Contraseña
- Nombre
- Apellidos
- Email
- Tipo de usuario

Existen dos tipos de usuario:

- Administrador: Tiene acceso total a la configuración de la aplicación y además puede añadir sillas subescaleras para monitorizar.
- Técnico: Sólo tiene acceso a la gestión de sus sillas y monitorización de éstas.

Si el usuario presiona el botón rojo (Cancelar) los cambios no se guardarán.

6.2 Editar usuario



Al editar un usuario no se pueden dejar campos en blanco a excepción de la contraseña, si no se especifica este campo, el sistema mantendrá la contraseña que el usuario tenía hasta ese momento.
Si el usuario presiona el botón rojo (Cancelar) los cambios no se guardarán.