UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Automatización y asistencia remota para cuidados de mascotas

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Cañamas, Jeremie

Tutor: José H. Canós Cerdá

2017/2018

Resumen

En este proyecto se desarrollará un sistema de control de alimentación para mascotas, el sistema se gestionará de manera automática y remota gracias a un cliente móvil que se comunicará con este. Para realizar este proyecto se tendrá que desarrollar todos los componentes del producto, pasando por el dispensador, un servidor y un cliente para la comunicación remota. Con el producto el usuario podrá gestionar todos los aspectos de alimentación, tanto los dispensadores y raciones de la mascota.
Para conseguir que el producto cumpla las necesidades del usuario, se analizara el mercado y las necesidades del mismo.

Palabras clave: Raspberry; Node.js; Arduino; Sensores; Android; MongoDB; API REST.

Abstract

A pet control feed system will be developed in this project. the system will be managed automatically and remotely thanks to a mobile client that will communicate with it. To carry out this project, you will have to develop all the components of the product, passing through the dispenser, a server and a client for remote communication. With this product, the user can manage all aspects about food, both the dispensers and rations of the pet.
To get that the product acomplish the user's needs, the market and its needs will be analyzed.

Keywords : Raspberry, Nodejs, Arduino, Sensor, Android, MongoDB, API REST.

Tabla de contenidos

Tabla de ilustraciones

1. Introducción

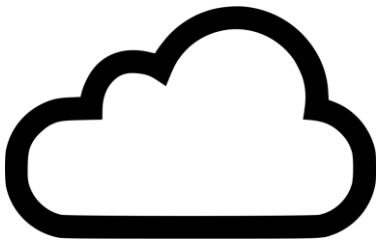
La elaboración de este documento surge de una necesidad, mantener a las mascotas correctamente alimentadas durante la ausencia de sus dueños. Para mantener nuestras mascotas correctamente alimentadas hay que tener control sobre la cantidad de comida y a qué hora tienen que comer.

El público objetivo es el que posea mascotas y que tenga que ausentarse durante un periodo de tiempo, pudiendo ser una jornada laboral hasta varios días. Estos usuarios tienen varias opciones para cubrir las necesidades. Una opción es depender de una tercera persona para que cuide de la mascota y otra opción es adquirir un dispensador de comida. La opción más sencilla es adquirir un dispensador, pero hay que elegir el adecuado. Dentro de la variedad de los dispensadores, los más adecuados tienen un precio elevado a causa de todas las funcionalidades extras.

En vista de la situación del mercado y de las necesidades, surge este proyecto con la intención de cubrir las funcionalidades imprescindibles sin aumentar el precio. El producto deseado por estos usuarios tiene que poder realizar la configuración de las raciones con cantidades de comida y horas. Además de poder realizarlas a través de una aplicación móvil. Para poder elaborar el producto más adecuado para los usuarios se harán los siguientes apartados:

- Un análisis de mercado.
- Un análisis de requisitos no funcionales.
- Establecer la estructura del sistema.
- Desarrollar el producto.

La memoria estará ordenada de la siguiente forma. Se empezará analizando la situación actual del mercado y un análisis de los casos de uso. Después de analizar la situación y los requisitos se establecerá la estructura del producto que mejor se adapte a los requisitos. Una vez establecida la estructura se detallará cada componente de la estructura. Se concluirá la memoria con relación a los estudios cursados y la bibliografía.


2. Estado del arte

En el mercado existen varios tipos de dispensadores automáticos. En este apartado se analizan las ventajas y desventajas de los mismos. Para ello se han dividido en tres grupos:

- Dispensadores de gravedad.
- Dispensadores automáticos con raciones limitadas.
- Dispensadores automáticos con depósito.

Dispensadores de gravedad

En el primer grupo están los dispensadores de gravedad. Su funcionamiento se basa en un depósito donde se alberga la comida y una bandeja en donde se repone la comida a medida que es consumida por el animal. Unos ejemplos en la figura 2. El precio de estos dispensadores se sitúa entre 10 y 30 euros.

Ventajas:

- No necesitan electricidad para funcionar.
- Tienen una buena capacidad en relación con su tamaño.
- El mantenimiento mínimo.
- Precio muy económico.

Desventajas

- Ningún control sobre las raciones.
- Únicamente comida solida como el pienso.
- Posibilidad media de atasco en el conducto entre el depósito y la bandeja.

El principal atractivo de estos dispensadores de comida es el precio, siendo estos los más económicos. Sin embargo, destacan como puntos negativos el inexistente control de las raciones y la posibilidad de atasco en el conducto que provee el alimento a la bandeja. Todo esto hace que la salud de la mascota pueda verse perjudicada.

Ilustración 1 dispensadores gravedad

Dispensadores electrónicos

El siguiente grupo de dispensador se basa en la electrónica para ofrecer raciones. Estas raciones son configurables por el usuario tanto en cantidad como en cuanto a horario de dispensación.

En este grupo se diferencian dos clases de dispensadores. En primer lugar, los dispensadores electrónicos circulares. En segundo lugar, los dispensadores electrónicos con depósito. A continuación, se analizarán las ventajas y desventajas de cada uno.

Dispensadores electrónicos circulares

Esta clase de dispensadores poseen funcionalidades como programar la hora a la que se sirve. Estos disponen de una bandeja circular con 6 u 8 huecos donde cada hueco es una ración. Un ejemplo en la figura 4Sobre la bandeja de las raciones se sitúa una pieza de plástico que cubre todas las raciones excepto una. Para realizar los servicios un motor hace girar la cubierta descubriendo la siguiente ración. Los precios de estos dispensadores son de 30 a 50 euros.

Ventajas:

- Control de los intervalos de tiempo entre raciones.
- Acepta cualquier tipo de comida tanto solida como húmedas.
- Autonomía de 3 a 6 meses.

Desventajas

- Raciones limitadas.
- Mantenimiento alto.

- Batería basada en pilas alcalinas.

El racionamiento de la comida y el precio ajustado, hacen de estos dispensadores de comida una opción a tener en cuenta. A pesar de esto sus limitaciones son importantes, ya que este tipo de dispensadores ofrecen menos raciones que sus alternativas. Además, el mantenimiento es mayor que en los dispensadores de gravedad ya que la tapa es móvil y fácil de atascar.

Ilustración 2 Dispensadores circulares

Dispensadores con depósito

Por último, los dispensadores electrónicos con depósito. En estos se puede programar tanto la hora de servir como la cantidad. Su funcionamiento se basa en un depósito de almacenado de comida y de un mecanismo que la deposita en la bandeja en el momento que especifique él usuario. Un ejemplo de su forma en la ilustración 3

Ventajas:

- Control sobre la hora de servicio.
- Control sobre la cantidad servida.
- Numero de raciones dependiente de la capacidad del depósito.

Desventajas:

- Partes móviles.
- Mantenimiento medio.
- Precio elevado.

- Únicamente comida solida como el pienso.

Dentro de este tipo de dispensadores hay un gran abanico de productos. Los más básicos ofrecen control sobre las raciones, la hora del servicio y la cantidad a servir. Mientras que los más complejos ofrecen funcionalidades extra como control remoto, retransmisión de video, mensajes de voz, etc. Todas estas ventajas aumentan el precio por lo que los hace menos accesibles.

Ilustración 3 Dispensadores con deposito

3. Análisis casos de uso

Tras realizar entrevistas a clientes potenciales, se analizará las necesidades de los usuarios. Para ello establecemos el contexto del proyecto. El cliente móvil es utilizado por los usuarios para realizar las acciones. Para que se conecte de manera remota el cliente móvil con el servidor para llevar a cabo las acciones en los distintos dispensadores, es necesario tener el servidor desplegado y accesible desde internet. El dispensador tendrá que actuar con el servidor para recoger datos del usuario. Se observa la estructura en la figura 4.

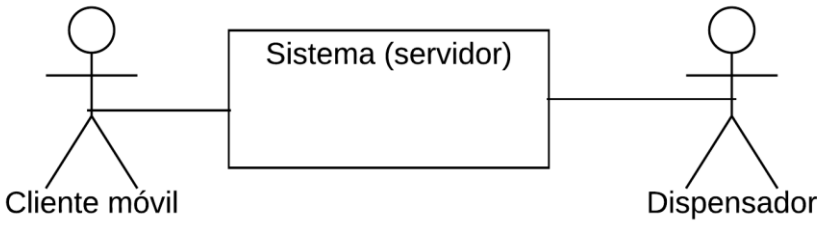


Ilustración 4 Contexto de la aplicación

Cliente móvil

El cliente móvil es la aplicación donde el usuario podrá realizar los cambios, añadir, eliminar y gestionar los dispensadores, sin embargo, el usuario debe estar registrado en la aplicación para realizar dichas acciones. Hay dos posibilidades para entrar en la aplicación, si es la primera vez el usuario tendrá que registrarse. Si ya está registrado podrá iniciar sesión. En la ilustración 5 se observan los casos

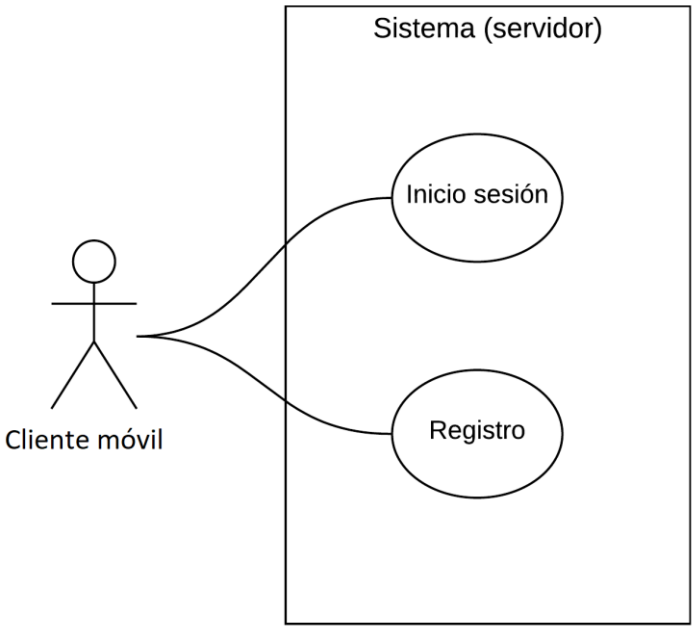


Ilustración 5 Login

Una vez realizados los pasos de inicio de sesión se accederá a la sección de los dispensadores. En esta sección podremos consultar, eliminar, crear y gestionar los dispensadores. En la ilustración 6 se muestran las acciones posibles.

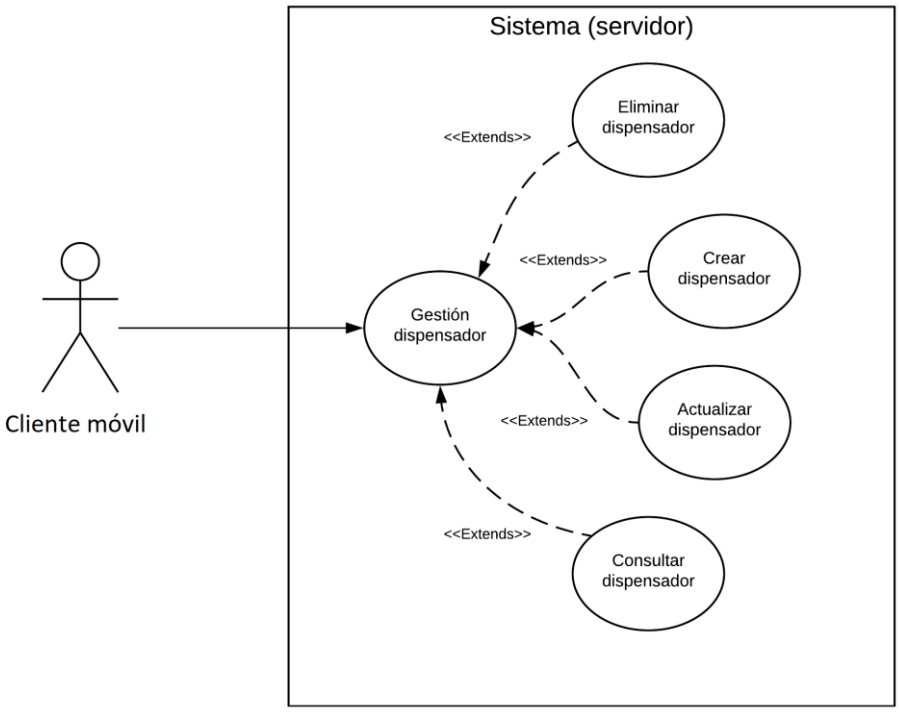


Ilustración 6 Casos uso con dispensador

Cuando accedemos a consultar el dispensador, accederemos a la sección de las raciones. En esta sección podremos consultar, actualizar, eliminar y crear las raciones. los casos se muestran en la figura 7.

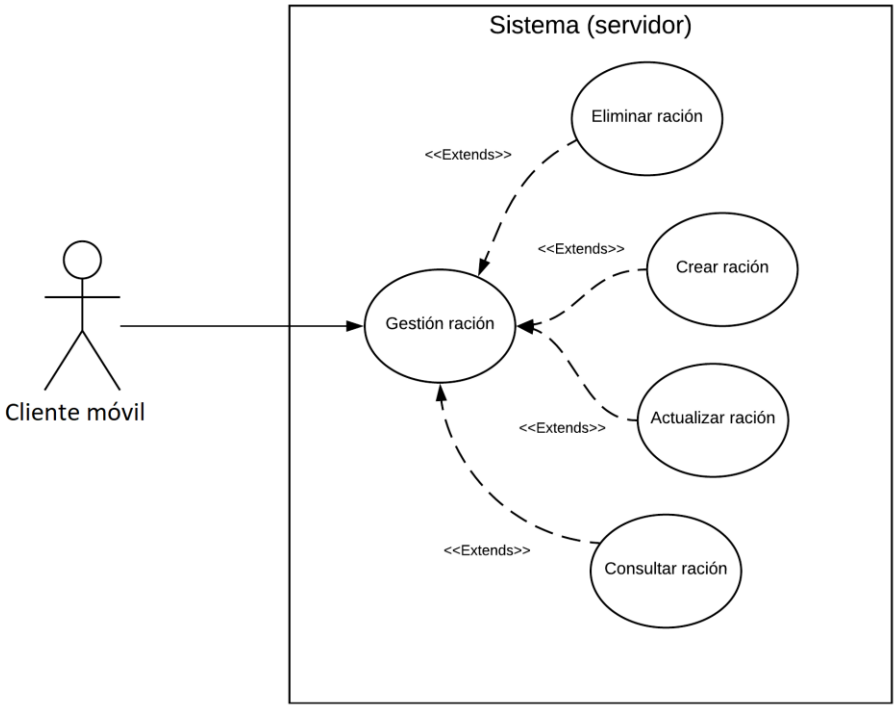


Ilustración 7 Casos uso ración

Dispensador

Este caso el actor es el dispensador el cual comunica con el servidor, que hace de sistema principal. El primer caso es obtener las raciones. Pedirá al servidor los datos de las raciones que estén relacionadas con este. El segundo caso es actualizar peso y esta acción solo la puede llevar a cabo el dispensador, ya que su estado varia. Observamos los casos en la figura 8.

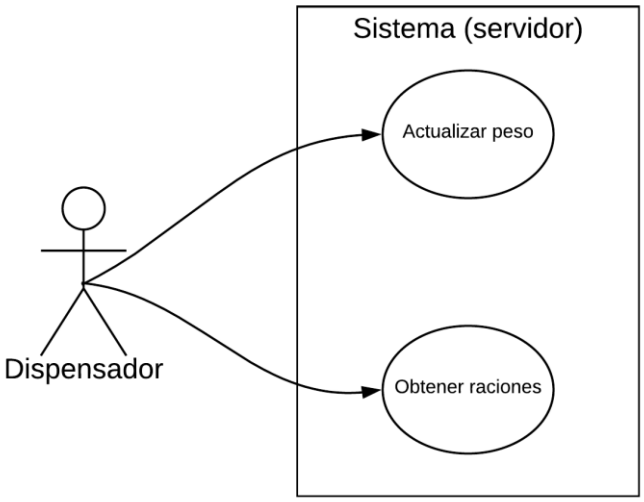


Ilustración 8 Casos uso dispensador

4. Diseño de la solución

Analizando el mercado, se desarrollará un dispensador de tipo electrónico con depósito. Porque este ofrece el mejor control en las raciones. Para evitar un sobre coste en el dispensador se implementará las funcionalidades justas para el control de las raciones. Para cumplir las necesidades del usuario se implementará un servidor. La creación del servidor permitirá al usuario consultar el estado del dispensador y las raciones programadas en cualquier lugar. Con la necesidad de abarcar el mayor número posible de usuarios, se implementará una API REST [1], esta ofrecería soporte multiplataforma, serian compatibles todos los dispositivos que soporten el protocolo HTTP [2]. En este caso solo se implementará una aplicación Android, pero en un futuro se podrá implementar una aplicación para IOS o una aplicación de escritorio para Windows. En la ilustración 9 podemos observar en esencia el producto resultante de este proyecto. Una aplicación Android, un servidor y un dispensador.

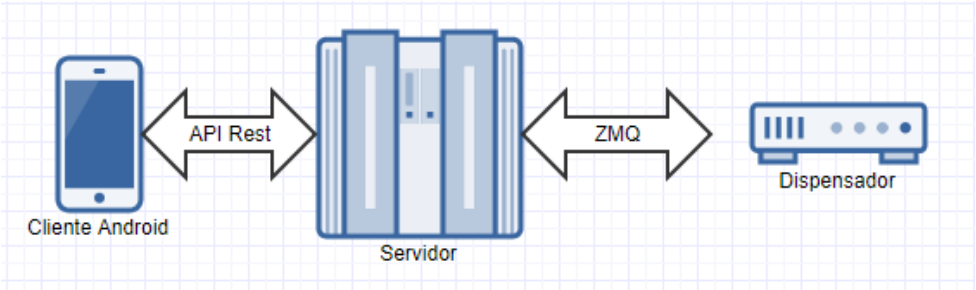


Ilustración 9 Concepto

4.1. Servidor

El servidor surge de una necesidad, específicamente para satisfacer la necesidad de realizar las acciones desde cualquier lugar. Además, para no limitar la creación de aplicaciones, es decir, poder desarrollar aplicaciones para ordenadores, navegadores web, etc. Para ello se desarrolla una API REST así cualquier dispositivo que acepte el protocolo HTTP podrá usar esta API REST. Para analizar los de manera más sencilla se ha dividido en dos partes. La primera parte, que es una API REST, que se comunica con la aplicación móvil. La segunda parte, es la que se comunica con los dispensadores, utilizando los sockets de la librería ZMQ [3].

4.1.1. API REST

REST es una arquitectura basada en el protocolo de comunicación HTTP (HyperText Transfer Protocol). La implementación de una arquitectura REST implica:

- Cada recurso disponible tendrá que tener una URL única.
- Implica el uso explícito de métodos de HTTP. Estos métodos son indicados en la cabecera HTTP por parte del cliente y son los siguientes.
    - GET: recoge información de un recurso.
    - PUT: modifica o actualiza el estado de un recurso.
    - POST: crea un nuevo recurso.
    - DELETE: elimina un recurso.

- Los datos de estas peticiones, tanto de envió como de recepción, están en formato JSON (JavaScript Object Notation) [4] o XML (eXtensible Markup Language).
- No mantiene estado.

Una API (aplication program interface) [5] es una interfaz que permite la comunicación entre programas. En este caso la API REST que se desarrolla es la interfaz que ofrecerá los recursos a los dispositivos móviles.

Cualquier petición que se realice tendrá un código de estado en la respuesta, los códigos siguientes son los utilizados en la API REST creada:

- 200, la operación se realizó con éxito.
- 404, no se encontró resultado de la petición.
- 500, error en la petición.
- 201, operación de nuevos datos realizada con éxito.
- 204, operación de eliminación realizada con éxito.

Detalle de los recursos

A continuación, se exponen las URLs que se utilizan en la API REST. Para facilitar la comprensión se han dividido en tres grupos. Los recursos de los usuarios, los dispensadores y las raciones.

Usuario

El usuario tiene dos posibilidades sobre la URL que es http://localhost/api/usuario. La primera con el verbo GET se utilizará para iniciar sesión. Dentro del cuerpo de esta petición hay que añadir dos valores el Email y la contraseña con el que se registró en la aplicación. En la segunda opción tenemos la de registro de usuario. Esta tiene el verbo POST, dentro de esta petición hay que enviar los datos Email, contraseña y nombre que tiene el usuario. Como resultado las dos peticiones devuelven un token [6] de autenticación para poder acceder al resto de las peticiones. Cuando se registra el usuario se genera un Id automáticamente. Este Id se utilizará en otras peticiones. Pero está incluido en el token.

Inicio sesión

4.2. verbo	4.3. Ruta	4.4.
GET	http://localhost/api/usuario	
DATOS		
Nombre	**Tipo**	**Descripción**
Contraseña	String	Se pondrá la contraseña con la que se registró el usuario.
Email	String	Se pondrá el email con el que registro el usuario
Respuesta		

Status	200
Token	

Registro usuario

verbo	Ruta
POST	http://localhost/api/usuario

Datos

Nombre	Tipo	Descripción
Contraseña	String	Se pondrá la contraseña con la que se registró el usuario
Email	String	Se pondrá el email con el que registro el usuario
Nombre	String	El nombre del usuario

Respuesta	
Status	200
Token	

Dispensador

En los dispensadores hay cinco posibilidades. Para poder acceder a la URL es necesario el token de autenticación recibido en el inicio de sesión o en el registro. El token se ubica en la sección de cabeceras de la petición. La URL es http://localhost/api/dispensador .

Consultar dispensadores

Con esta petición se obtendrá la lista de los dispensadores que este asociado al usuario. Para ello se tiene que enviar el token que estará incluido en la cabecera.

verbo	Ruta
Get	http://localhost/api/dispensador

Datos

Nombre	Tipo
Token	String

Respuestas	
Status	200
body	Lista de dispensadores JSON

Crear dispensador

En la petición de crear dispensador hay que enviar los siguientes datos el nombre que tendrá el dispensador. Además del token del usuario para recuperar el Id y poder asignarle el nuevo dispensador. Como resultado obteneos el nuevo dispensador en formato JSON. Al guardar el dispensador en la base de datos se genera automáticamente un Id que lo asigna la base de datos.

verbo	Ruta	
POST	http://localhost/api/dispensador	
Datos		
Nombre	**Tipo**	**Descripción**
Nombre	String	El nombre que se le quiere asignar al comedero
Token	Sting	

Respuestas	
Status	201
body	El nuevo dispensador en formato JSON

Eliminar dispensador

Para eliminar un dispensador hay que enviar el Id del dispensador. El Id del dispensador se genera cuando se creó. Este tendrá que ir en el cuerpo de la petición. Como resultado obtendremos una respuesta 204.

verbo	Ruta
DELETE	http://localhost/api/dispensador
Datos	
Nombre	**Tipo**
IdDispensador	String

Respuestas	
Status	204

Actualizar Dispensador

Para actualizar un dispensador es necesario el id generado por la base de datos cuando se creó, y el valor de los atributos. Todos estos estarán en el cuerpo de la petición.

verbo	Ruta
PUT	http://localhost/api/dispensador
Datos	
Nombre	**Tipo**
IdDispensador	String
Nombre	String

Respuestas	
Status	200

Ración

Para las raciones tenemos cuatro posibilidades en sobre la URL http://localhost/api/racion para acceder a estas peticiones también es necesario el token de autenticación.

Consultar ración

Para acceder a las raciones se utiliza el Id del dispensador ya que cada ración esta asignada a un dispensador. El id estará ubicado en el cuerpo de la petición. Como resultado de la petición obtenemos una lista con las raciones asignadas al dispensador.

verbo	Ruta
GET	http://localhost/api/racion
Datos	
Nombre	**Tipo**
IdDispensador	String

Respuestas	
Status	200
Body	Lista raciones formato JSON

Crear ración

Para crear una ración necesitamos todos los atributos de una ración que son: Id dispensador, nombre, hora, minutos y cantidad. Los atributos de la ración estarán ubicados en el cuerpo de la petición. Una vez realizada la petición obtenemos una ración en formato JSON, además con el Id asignado.

verbo	Ruta
POST	http://localhost/api/racion

Datos

Nombre	Tipo	Descripción
IdDispensador	String	
Nombre	String	Un nombre que elija el usuario
Hora	number	La hora a la que se sirve la ración
minutos	number	En que minuto de la hora se sirve la ración
Cantidad	number	Que cantidad se tendrá que repartir

Respuestas	
Status	201
Body	La nueva ración formato JSON

Eliminar ración

Para eliminar la ración se necesita la Id de la ración. El Id se ubicará en el cuerpo de la petición. Una vez finalizada la operación obtenemos el código 204 para indicar que se eliminó correctamente.

verbo	Ruta
DELETE	http://localhost/api/racion

Datos		
Nombre	**Tipo**	**Descripción**
IdRacion	String	

Respuestas	
Status	204

Actualizar ración

Para actualizar una ración se necesita el Id de la ración y todos los atributos. Los atributos se ubican en el cuerpo de la petición. Como resultado obtenemos el código 200.

verbo	Ruta
PUT	http://localhost/api/racion

Datos enviados		
Nombre	**Tipo**	**Descripción**
IdRacion	String	
Nombre	String	Nombre asignado a la ración
Hora	number	La hora a la que se sirve la ración
minutos	number	En que minuto de la hora se sirve la ración
Cantidad	number	Que cantidad se tendrá que repartir

Respuestas	
Status	200

4.4.1. Middleware

El middleware se implementa para proteger las rutas. La protección consiste en la autenticación mediante tokens [7]. Hay varios tipos de token, el utilizado es el JWT(JSON Web Token). Como se ha visto en el punto anterior todas las peticiones exceptuando las de usuario requieren el token para acceder. Para obtener el token hay que iniciar sesión o registrarse en la aplicación. Con los datos recibidos de las peticiones crea un token que contiene los siguientes datos: Id usuario, fecha creación [8] y fecha expiración. En el token también se establece la codificación utilizada, en este caso es la HS256. Para codificar también se utiliza claves secretas. Se visualiza un ejemplo en la ilustración 10.

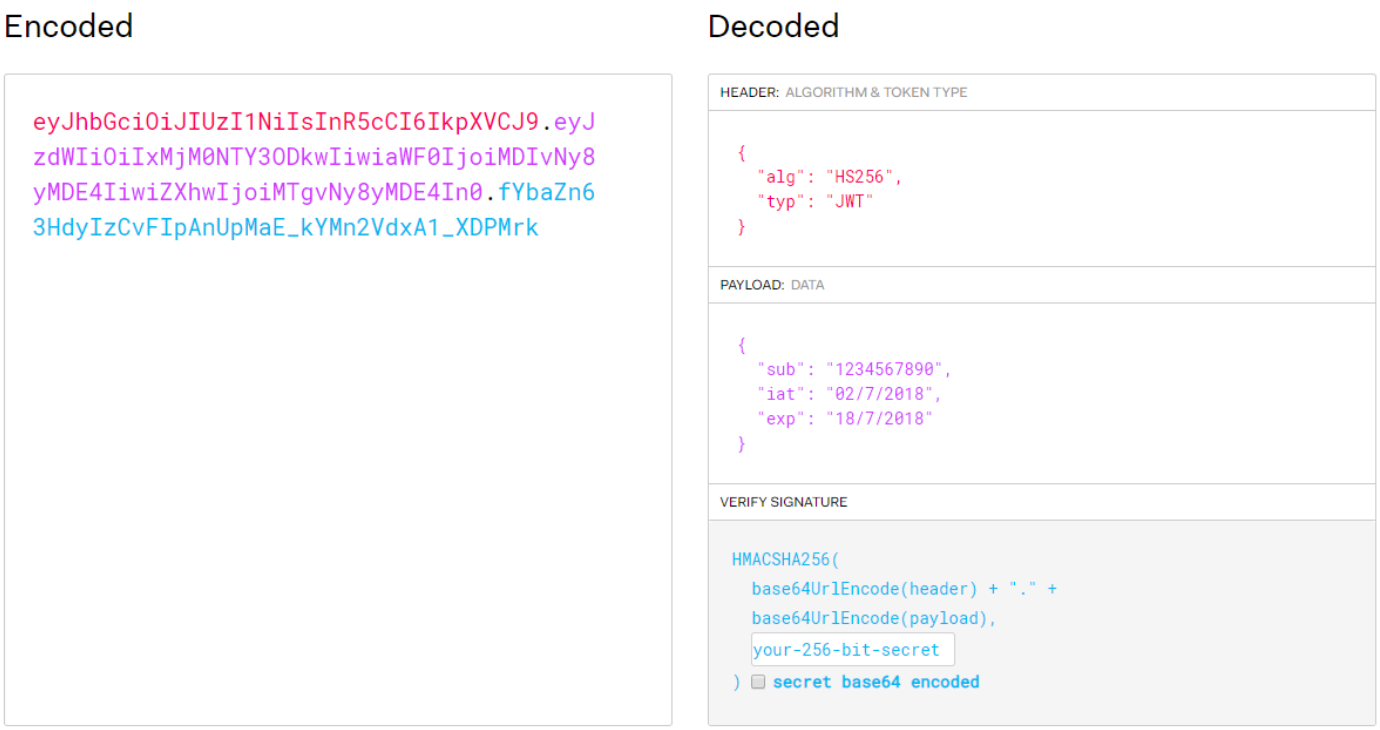


Ilustración 10 JWT Token

4.4.2. Controladores

Se implementan los controladores para: persistir los datos, consultarlos, actualizarlos o eliminarlos. En esta sección se explicará los controladores, que son utilizados por las rutas y por la interfaz de los dispensadores. Los controladores procesan los datos que llegan de ambos sitios. Para poder registrar, consultar, actualizar o eliminar tiene que procesar las peticiones y mapearlas. Para ello utiliza el modelo de datos que se definirá en el punto 4.1.4.:

- Error al realizar la petición, significa que no hay conexión con la base de datos.
- ítem inexistente, en este caso la consulta se ha hecho de manera satisfactoria pero no existe un resultado, es decir, devuelve una consulta vacía.

- Consulta satisfactoria, en este caso la consulta se ha realizado y existe un resultada.

Los controladores se han agrupado en tres grupos los cuales se corresponden con: usuario, dispensador, ración.

Usuarios

El controlador de los usuarios, hay dos funciones las cuales son:

- **Registrar usuario**: creara un objeto usuario a partir de los datos recibidos. Para poder crear al nuevo usuario, se recoge los datos en la petición, que se ubican en el cuerpo de la petición. Una vez hecho el mapeado con los datos de la petición y el modelo de datos, se ejecutará la consulta para persistirlo en la base de datos.
- **Inicio sesión:** comprobará si el usuario existe en la base de datos, para poder comprobar lo correctamente recupera el email y la contraseña del cuerpo de la petición. Con los datos extraídos se realizará la búsqueda. En la búsqueda tiene que coincidir el Email y la contraseña del usuario. Si la búsqueda resulta vacía se le notificara al usuario. Si existe el usuario generara un nuevo token para enviarlo.

URL	Función
api.get('/usuario', userCtrl.singIn)	Inicio sesión
api.post('/usuario',userCtrl.singUp)	Registro usuario

Dispensadores

En el controlador de los dispensadores hay 4 funciones que son:

- **ObtenerDispensadores:** la función se recuperará todos los dispensadores que estén relacionados con el usuario, para ello se realiza una búsqueda con el id del usuario. El Id del usuario se recupera por el token, que es necesario para acceder

a estas rutas. En la búsqueda se utilizará para identificar todos los dispensadores con el mismo usuario.

- **EliminarDispensadores:** esta función eliminará un dispensador, para ello se recupera el Id del dispensador que estará en el cuerpo de la petición. Con el Id se ejecutará la función de buscar y eliminar.
- **CreateDispensador:** esta función crea un dispensador. Los datos necesarios son el Id del usuario recuperado del token, un nombre y el código del dispensador que serán recuperados del cuerpo de la petición.
- **ActualizarDispensador:** con esta función se actualizará el dispensador. Para ello recupera todos los datos del dispensador y los actualiza. El peso no se puede actualizar por el usuario.
- **GuardarPeso:** esta función es particular de la interfaz de los dispensadores. En el mensaje recibido del dispensador habrá el código con el que se registró y el peso de la comida restante en el depósito.

URL	Función
Api.get('/dispensador', auth.IsAuth, dispCtrl.getDispensador)	ObtenerDispensador
Api.delete('/dispensador', auth.IsAuth, dispCtrl.deleteDispensador)	EliminarDispensador
Api.post('/dispensador', auth.IsAuth, dispCtrl.postDispensador)	CuardarDispensador
Api.put('/dispensador', auth.IsAuth, dispCtrl.putDispensador)	ActualizarDispensador
Interfaz dispensador	GuardarPeso

Ración

El controlador de las raciones hay 5 funciones:

- **CreateRacion:** para crear una nueva ración es necesario el Id del dispensador. Además de los atributos: nombre, cantidad, hora y minutos, todos estos datos se recuperan del cuerpo de la petición. Con ellos se realizará el mapeado con el

modelo de dato de las raciones y se lanzará la operación de guardado a la base de datos.

- **DeleteRacion**: Para la eliminación de una ración únicamente es necesario el id de la ración este se obtiene del cuerpo de la petición y se hará una búsqueda para eliminar el objeto guardado.
- **UpdateRacion:** esta función hace los mismo que en crear la ración únicamente cambia la operación a realizar.
- **GetRacion** con esta función obtenemos todas las raciones relacionadas con un dispensador, por lo tanto, es necesario tener el Id del dispensador, se realiza la búsqueda de las raciones que tengan el mismo id de Dispensador y se enviaran al usuario.
- **Extrac** esta función es únicamente para la interfaz de Arduino, cuando se recibe una petición a la interfaz se realiza con el id del dispensador por lo tanto se realiza la búsqueda y se extrae las raciones con el mismo Id. Pero además se da un formato específico para que el dispensador pueda interpretarlos.

URL	Función
api.get('/racion', auth.IsAuth, racionCtrl.getRacion)	**getRacion**
api.post('/racion', auth.IsAuth, racionCtrl.postracion)	**postRacion**
api.delete('/racion', auth.IsAuth, racionCtrl.deleteRacion)	**deleteRacion**
api.put('/racion', auth.IsAuth, racionCtrl.putRacion)	**putRacion**
Interfaz dispensador	**Extrac**

4.4.3. Modelos de base de datos

En este apartado se define el modelo de datos que sigue los objetos: usuarios, dispensador y ración. Se establecen estos modelos porque la base de datos utilizada es MongoDB [9]. Esta base de datos es de tipo NoSQL, es decir, que no tiene tablas ni entidades. Lo que le permite tener una gran flexibilidad. Pero a la vez puede ser perjudicial porque no sigue ninguna estructura y ser difícil de mantener. Los datos se almacenan en texto plano con formato JSON. Los modelos de los objetos son:

Modelo Dispensador

Nombre	Tipo
Nombre	String
Usuario	String
Cantidad	Number
IdFisico	String

Modelo Ración

Nombre	Tipo
Nombre	String
Cantidad	Number
Hora	Number
Minuto	Number
Dispensador	String

Para la tabla de los usuarios hay precondiciones de guardado y de búsqueda. Esto se debe a que la contraseña de los usuarios esta encriptada [10] y por lo tanto hay que procesar los datos antes de cualquier operación.

Modelo Usuario

Nombre	Tipo
Email	String
Nombre	String
SingnUpDate	Date
LastLogin	Date
Contraseña	String

Una vez visto todos los componentes que conforman la API REST se mostrara con un diagrama de proceso la ruta que sigue una petición ilustración 12.

DIAGRAMA SECUENCIA PETICIÓN

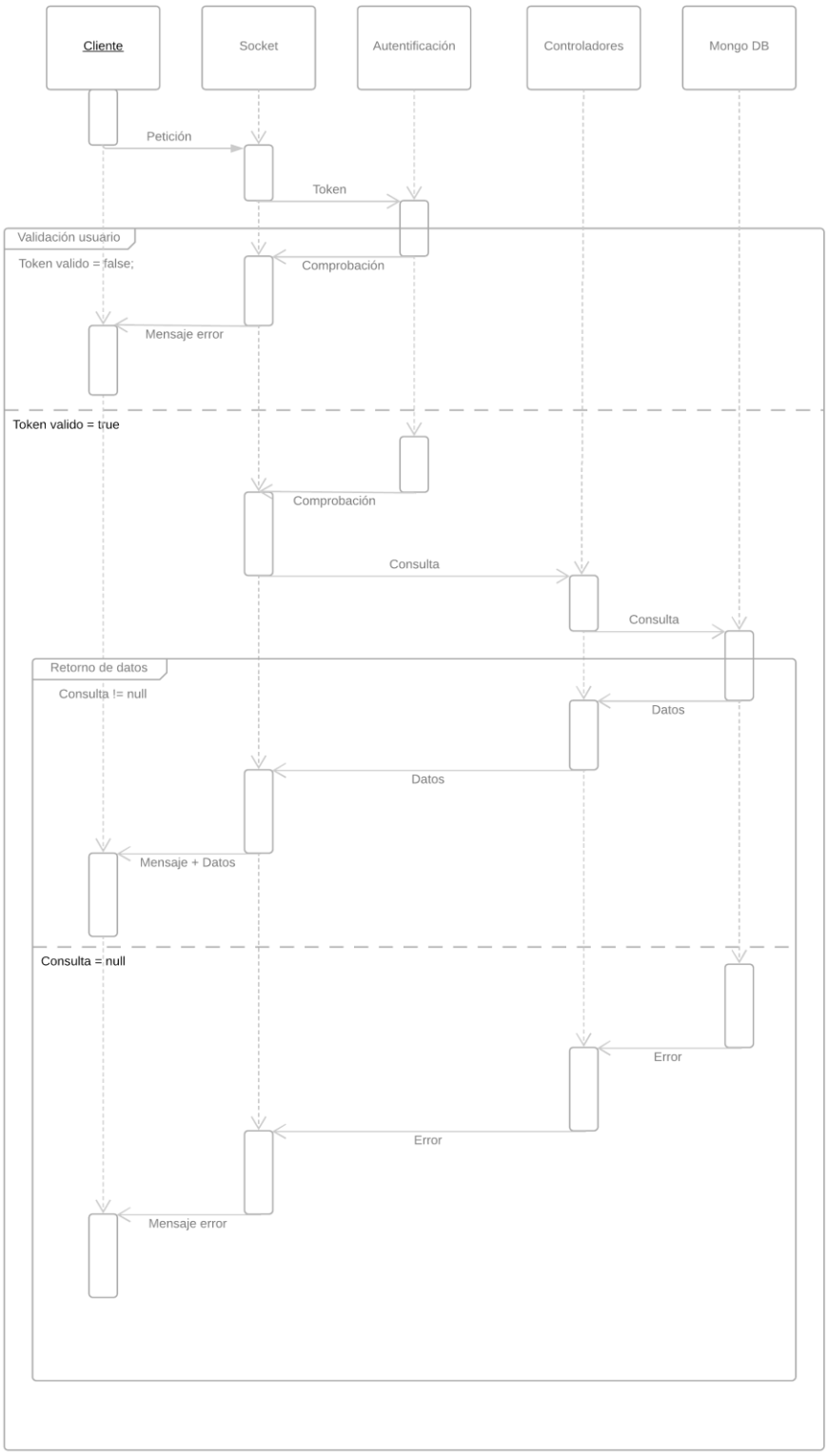


Ilustración 11 Secuencia petición

4.4.3.1. Interfaz para los dispensadores

Esta interfaz se ha creado únicamente para los dispensadores. Esta interfaz usa los sockets de la librería ZeroMQ. Se ha elegido esta tecnología porque el socket router [11] distingue entre agentes. Para distinguir los dispensadores que se conecten se utiliza el primer segmento del mensaje. En este segmento se ubica el identificador del dispensador. Este coincidirá con el Id físico que se utiliza para el registro.

En el paso de mensaje se realizan varias acciones, primero el dispensador pedirá una actualización de datos, en este mensaje viajan dos datos, el primero es el identificador del dispensador y el segundo dato corresponde al peso de la comida restante, con estos datos el servidor guardara el peso con el correspondiente dispensador. para ello utilizara la función vista antes guardar cantidad que requiere. de estos datos, una vez almacenados se procede a realizar él envió de datos. En él envió de datos el servidor extraerá todas las raciones relacionadas con el dispensador. Utiliza la función extrac visto en el punto 4.1.3. Con los datos extraídos se genera una estructura descartando datos que no utilizara él dispensador. Únicamente enviara los datos: cantidad, hora y minuto. se genera el mensaje con el mismo Id físico en el primer segmento y la estructura generada en el tercer segmento.

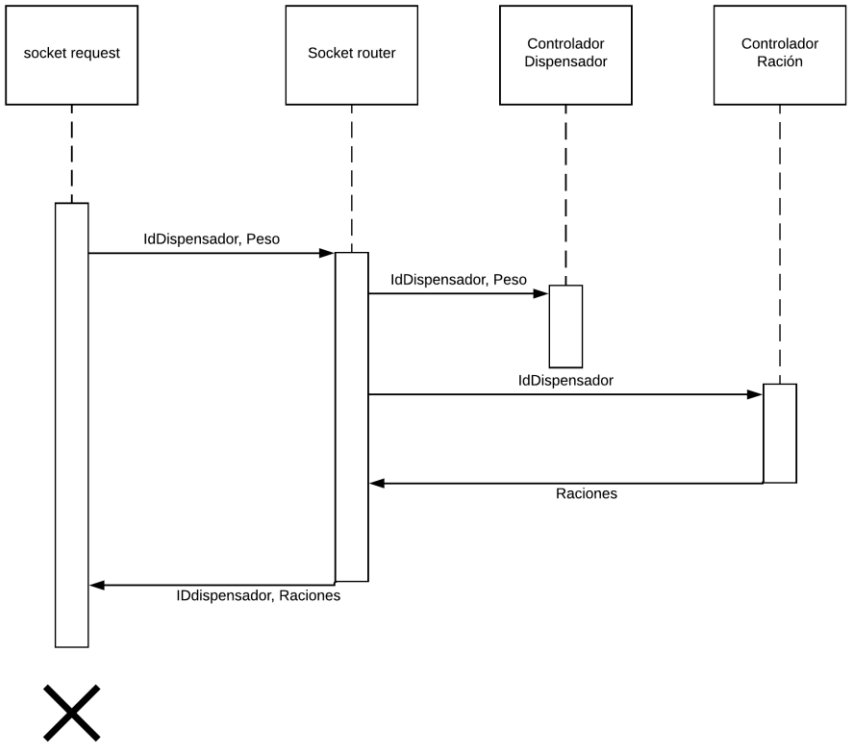


Ilustración 12 Socket router

4.5. Dispensador

En esta sección se explicará el hardware utilizado y el software necesario para su funcionamiento. En primer lugar, se expone la arquitectura física del dispensador y que hardware es utilizado en cada parte del mismo. Seguidamente se explica el software de cada componente.

4.5.1. Hardware

El dispensador está compuesto por varios componentes hardware los cuales son: Raspberry pi Zero w [12], Arduino NANO [13], amplificador HX711 [14], célula de carga de 3 kg [15]y un motor. A continuación, el diagrama físico, de las conexiones entre ellos.

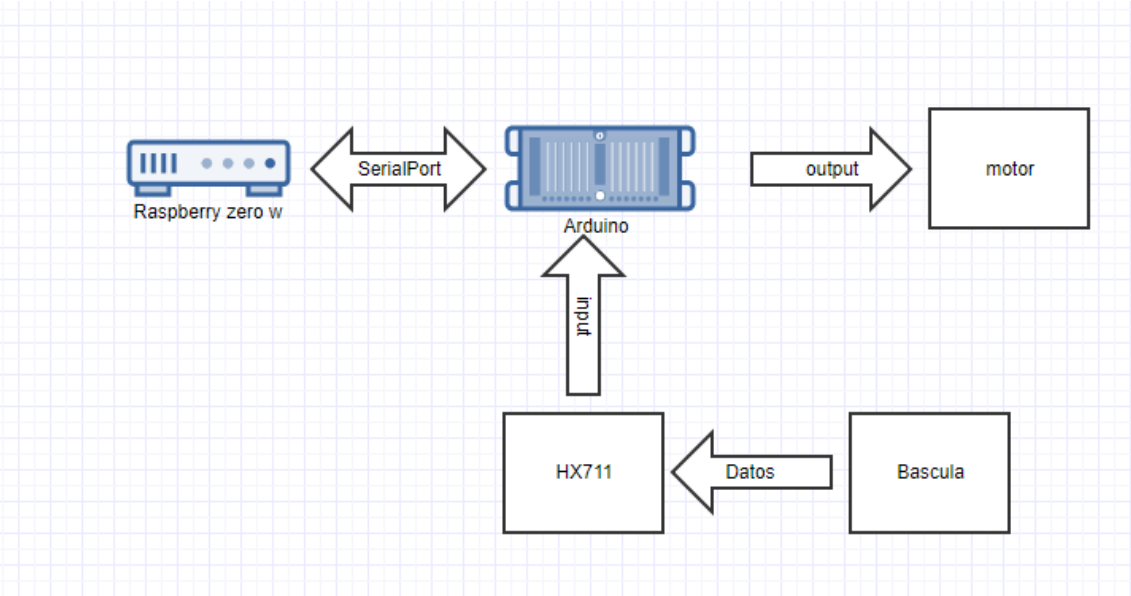


Ilustración 13 Estructura dispensador

4.5.2. Raspberry pi Zero W

Este miniordenador se utilizará para conectarse al servidor y actualizar la base de datos. Para asegurar su funcionamiento si no dispone de una conexión a internet. También tiene la función de controlar el Arduino. Se eligió este dispositivo por varios motivos. Dispone de conexión wifi, el tamaño reducido para incluir lo en el dispensador y un precio económico.

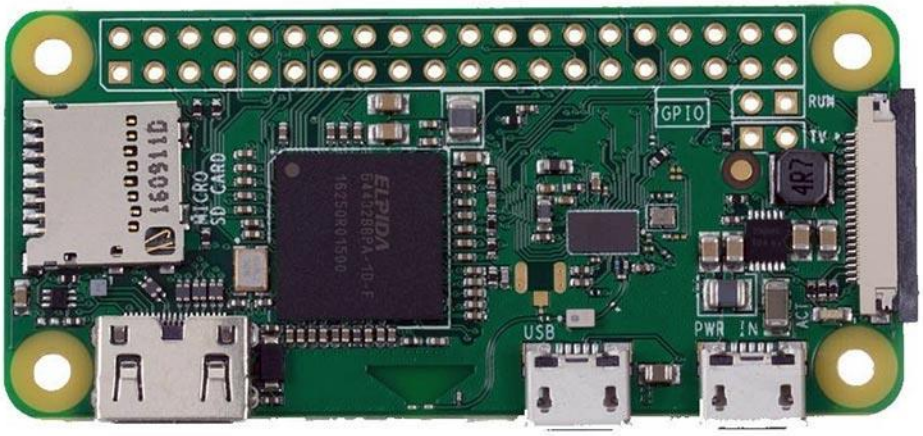


Ilustración 14 Raspberry pi zero

Procesador	1GHz single-Core CPU
RAM	512MB RAM
Puertos	• Mini HDMI port • Micro USB OTG port • Micro USB power • CSI conector de cámara
Pines	HAT-compatible 40-pin header
Precio	11,10 euros

4.5.2.1. Arduino

El Arduino NANO es el encargado de las entradas y salidas del dispensador, es decir, de la báscula para la entrada de datos y del motor como salida de datos. Se eligió este dispositivo por su tamaño, el precio y la facilidad de controlar los sensores. Esta placa posee las siguientes dimensiones 18,5mm x 43,2mm, tiene una memoria para el programa de 32 Kb. Además, dispone de 30 pines que para trabajos futuros pueden tener utilidad.

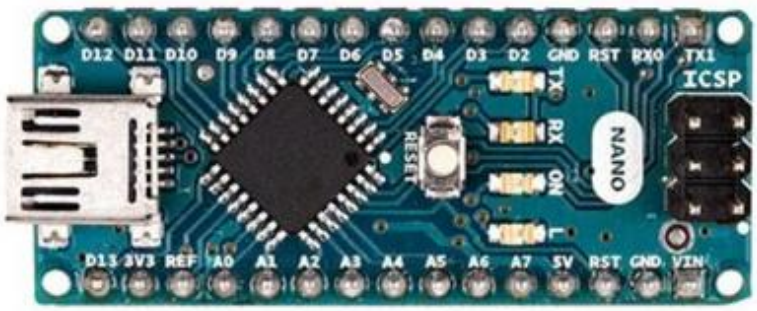


Ilustración 15 Arduiono NANO

Las especificaciones de este Arduino son:

Microcontrolador	ATMega328
Voltaje de operación	5V
I/O digitales	14 (6 son PWM)
Memoria flash	32KB
Frecuencia de trabajo	16MHz
Precio	3 euros

4.5.2.2. Amplificador hx711

Este módulo nos permitirá trabajar de forma sencilla con las células de carga, ya que es capaz de detectar las variaciones en las resistencias de la célula, no solamente hay que conectar lo al dispositivo para obtener el peso hay que realizar calibraciones y con ellas podemos tener una gran precisión, ya que devuelve valores con una precisión de 24 bits lo que significa 16,8 millones de valores. En la ilustración se visualiza el chip con las conexiones a la placa de Arduino y a la célula de carga.

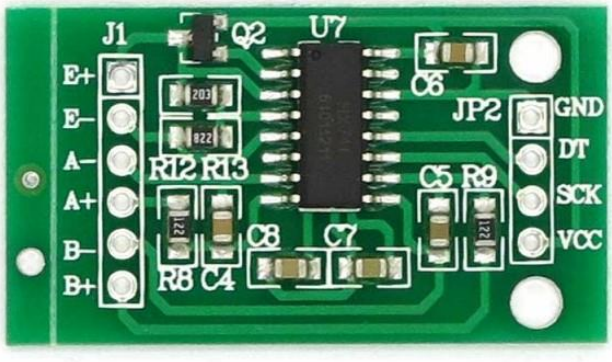


Ilustración 16 Amplificador HX711

4.5.2.3. Célula de carga

El funcionamiento de estos dispositivos es el siguiente se les suministra una señal eléctrica y devuelve una energía directamente proporcional al peso del objeto. La señal eléctrica recorre unas resistencias que están situadas en la estructura de la célula, al ejercer una fuerza esta barra sufre una deformación al igual que las resistencias que están situadas en el dispositivo. Esta deformación hace variar la corriente que se le suministra, a partir de esta se calcula el peso. Este circuito y la ubicación de las resistencias se llama Puente de Wheatstone.

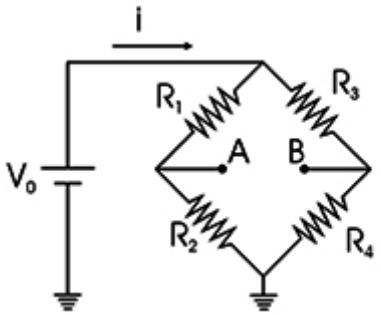


Ilustración 17 Puente Wheatstone

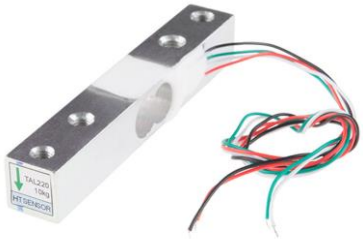


Ilustración 18 Célula de carga

4.5.2.4. Motor

El motor tendrá la función de generar movimiento para que el dispensador mueva los engranes que conducen la comida. los datos de este motor es que necesita al menos 5 voltios para funcionar y a la velocidad de 6000 revoluciones por minuto. Junto a este habrá un sistema de engranajes que reducirán las vueltas por minuto y proporcionara más fuerza al sistema de reparto de comida.

Ilustración 19 motor

4.5.3. Software

El siguiente apartado estará descompuesto en varias secciones la primera corresponde a la Raspberry pi Zero, en esta se habla del programa que posee. En la segunda parte se hablará del Arduino, de cómo ha sido programado para realizar el reparto de la comida y recuperar el peso restante en el depósito.

La arquitectura de los dispensadores es la siguiente la parte principal es la Rapsberry. Esta se encargará de mandar al Arduino lo que tiene que hacer, esta orden será enviada a través del puerto USB o puerto serie, además de controlar el Arduino tendrá que comunicarse con el servidor para poder actualizar la base de datos. La segunda sección del dispensador es el Arduino, este recibirá las órdenes y las ejecutará. En último lugar se encuentran los actuadores y los sensores que en este caso solo hay dos la báscula que es un sensor y el motor que es el actuador.

4.5.3.1. Raspberry pi Zero

En esta sección se describirá este la funcionalidad, en primer lugar, se describirá el funcionamiento del socket con el correspondiente controlador y la base de datos que utilizan para almacenar los datos, en segundo lugar, se explicara el funcionamiento y la comunicación que hay con el Arduino.

Comunicación y actualización del sistema

La comunicación con el servidor se realiza a través de un socket de ZMQ en este caso se utiliza el socket de tipo ‘request’, estos sockets tienen un atributo llamado ‘identity’ el cual será distinto para cada dispensador. El dispensador tendrá una rutina que mandará que se actualice cada hora. Con la demanda de la actualización también se enviarán datos como el peso restante en el depósito del dispensador, este peso se obtendrá a través del controlador del Arduino. Con la respuesta del servidor se procederá a descomponer lo ya que en el servidor se le dio un formato con los datos únicamente necesario para el dispensador que son la hora, la cantidad a servir y el ID de la ración que genera MongoDB con estos datos se realizara el guardado.

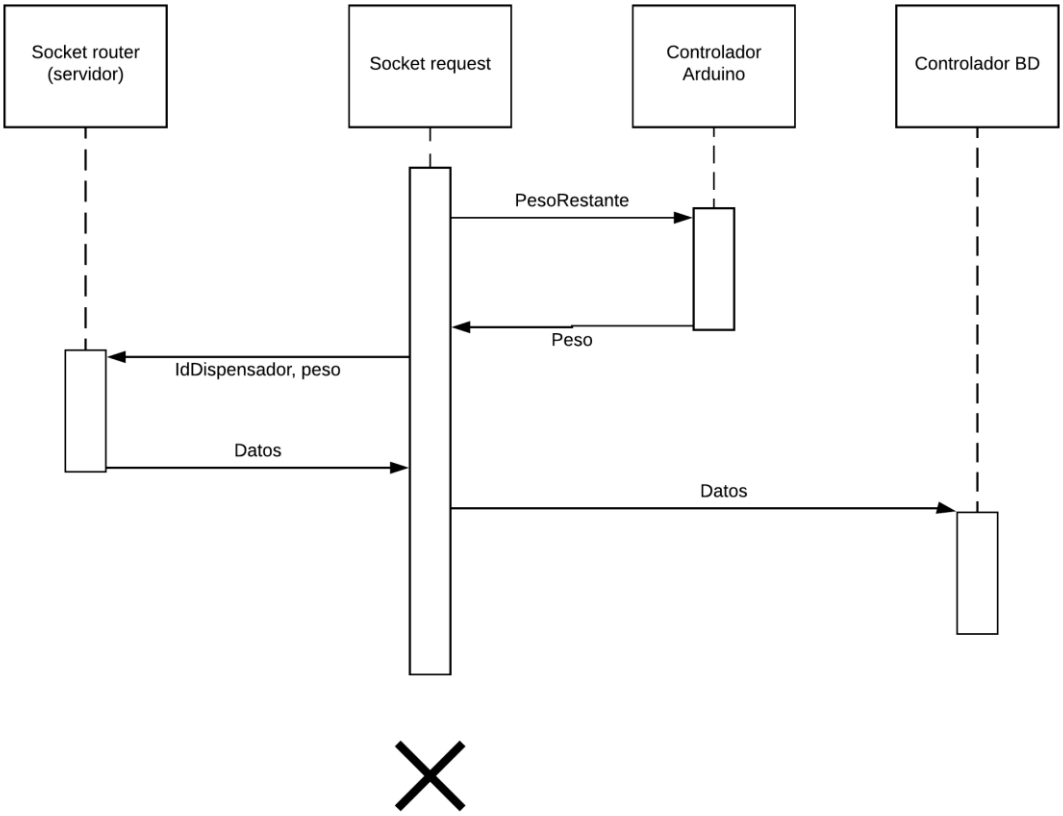


Ilustración 20 Conexión Dispensador

Rutina Arduino y comunicación

Esta rutina indicara en que momento tiene que servir una ración y cuanta cantidad. Primero, hay que crear la rutina que verifique la hora a la que tiene que servir, la rutina se ejecutara cada hora, en cada ejecución tiene que acceder al controlador de la base de datos para sacar las raciones que tiene guardadas, una vez obtenidos comprobara la hora de todas las raciones, en el caso de no haber ninguna en la próxima hora volverá a la espera y si hay una raciones se ejecutara una nueva rutina, esta comprobara cada minuto y cuando llegue al minuto en que se tenga que servir se lanzara la orden al controlador del Arduino esta orden únicamente contendrá la cantidad que debe servir, una vez mandada la orden al controlador generara el mensaje y lo emitirá por el puerto serie [17] al que esté conectado el Arduino.

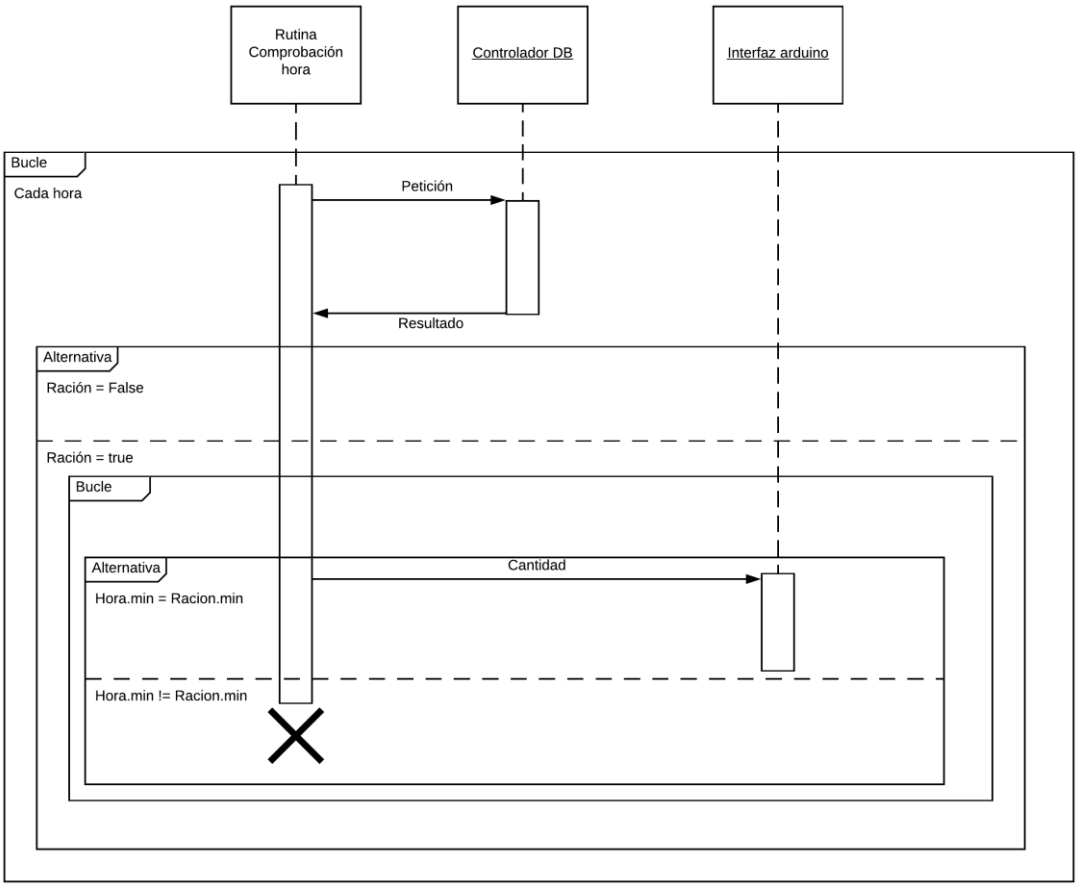


Ilustración 21 Rutina comprobación Hora

4.5.3.2. Arduino

El Arduino es el componente encargado de controlar el hardware, este se comunica a través de los pines. El actuador es un motor, que será el encargado de sacar la comida. El sensor es el amplificador hx711 que a la vez estará conectado a la célula de carga, estos dos componentes conforman la báscula, la ilustración 22 muestra la estructura de las conexiones con él Arduino.

El Arduino actuará cuando se lo comunique la Raspberry pi. En el mensaje enviado contendrá dos secciones primero la orden, y si la orden es de servir comida el segundo datos será la cantidad a servir. Las ordenes pueden ser dos, la primera es recuperar el peso de la báscula. El valor recuperado se enviará a la Raspberry pi. La segunda es servir comida. Con esta función también se enviará la cantidad que tiene que servir.

Si la operación a realizar es servir primero obtendrá el peso de la báscula y empezará a servir, cuando el depósito haya perdido el peso equivalente a la ración, parará el motor y el servicio habrá finalizado.

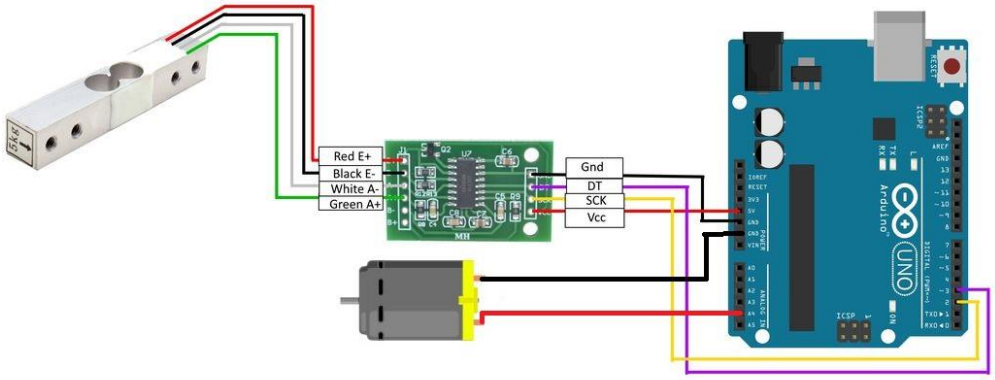


Ilustración 22 Conexiones Arduino

4.6. Aplicación Android

En esta sección se explicará la aplicación Android [18] y que casos de uso se utiliza en cada pantalla. Se muestra el mockup y la pantalla de la aplicación final. Las pantallas se agrupan en tres grupos: usuarios, dispensadores, raciones.

Pantallas Usuario

En las pantallas del usuario se utilizan los casos de uso de inicio de sesión y la de registro de usuario.

Inicio sesión: Como vemos en la ilustración 23, hay dos campos de texto y dos botones, el primer campo de texto es para poner el email con él que se registró el usuario y el segundo con la contraseña utilizada. En el primer botón será para iniciar la sesión y el segundo para que usuario pueda registrarse en la aplicación.

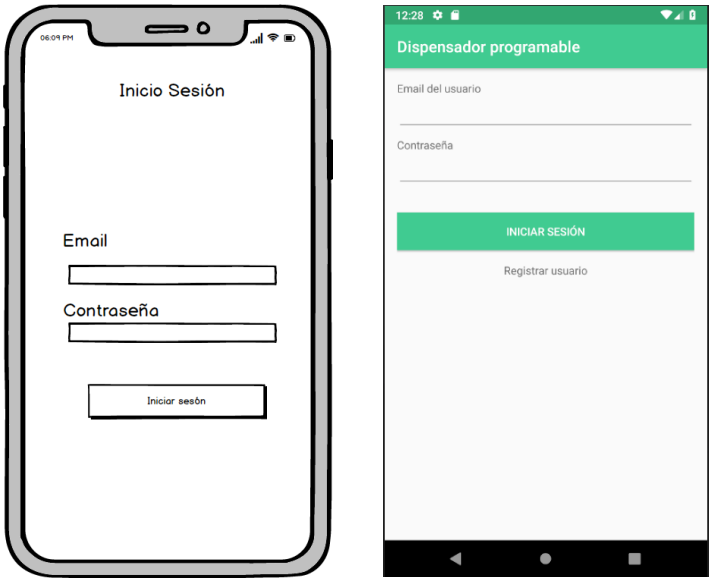
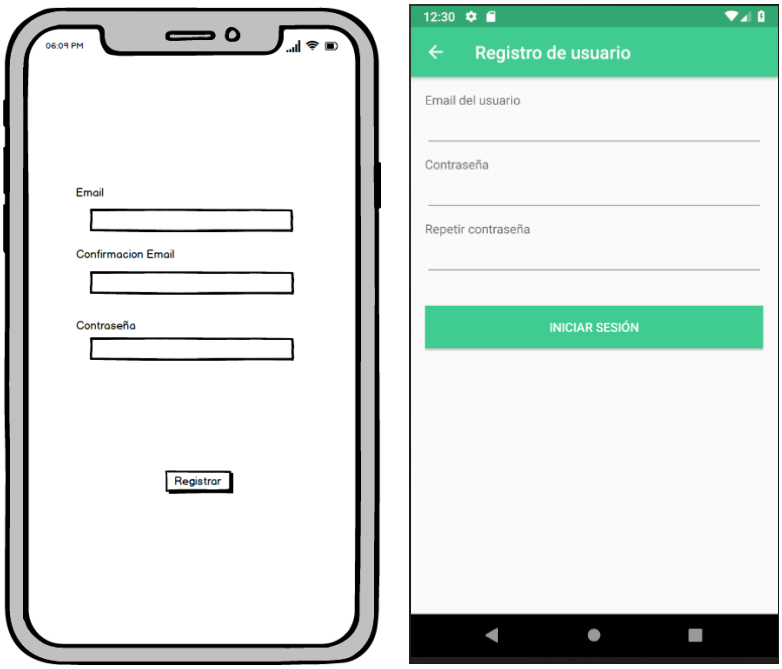


Ilustración 23 Inicio sesión

Registro usuario: En esta pantalla hay tres campos de texto el primero es para el email, los dos campos siguientes es para la contraseña, hay dos para que se asegure que es la correcta. Seguidamente hay un botón para finalizar el registro.



Pantallas Dispensadores

Las pantallas mostradas a continuación se utilizan para la gestión de los dispensadores que son: consultar dispensador, crear dispensador, eliminar dispensador y actualizar dispensador.

- **Consultar dispensador**: En la ilustración 24 observamos todos los dispensadores que tiene un usuario con el nombre que se le asigno y el peso restante en el depósito. Desde esta pantalla podemos ir a la de registrar dispensador, actualizar dispensador, eliminar dispensador y si se selecciona un dispensador iremos a la pantalla de las raciones del dispensador seleccionado. Si se selecciona el botón editar iremos a actualizar dispensador. Si se selecciona eliminar iremos a la pantalla de eliminar dispensador. Si se selecciona el botón con el símbolo de '+' iremos a crear dispensador.

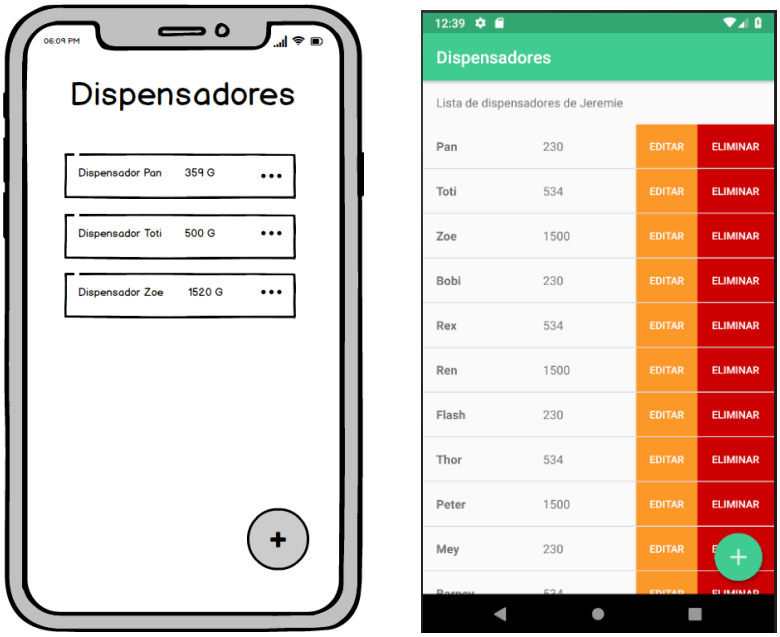


Ilustración 24 Consultar dispensadores

- **Crear dispensador:** En la ilustración 25 se observa dos campos. El primero, corresponde al nombre que se le quiere dar al nuevo dispensador. El segundo corresponde con el Id físico que tiene cada dispensador. Una vez finalizado se pulsará crear dispensador.

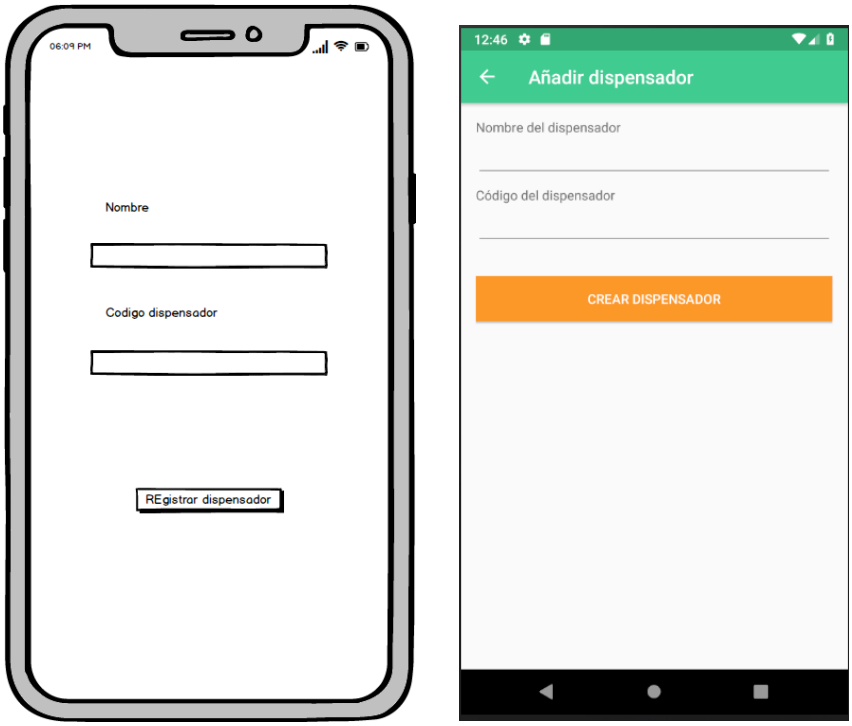


Ilustración 25 Añadir dispensador

- **Eliminar dispensador:** En la ilustración 26, cuando pulsemos el botón eliminar de la lista aparecerá una notificación de si el usuario está seguro de eliminar el dispensador. Se pulsará el deseado y volverá a la pantalla de consultar dispensadores.

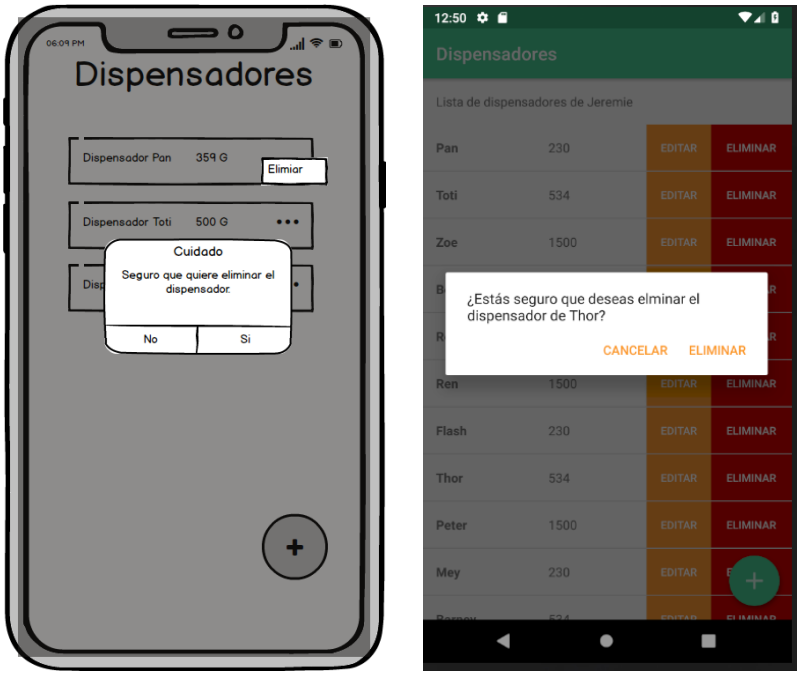


Ilustración 26 Eliminar dispensador

Pantallas raciones

En el siguiente conjunto de pantallas se gestionará las raciones**.**

- **Consultar las raciones:** En la ilustración 27 se observa las raciones que tiene asignado el dispensador seleccionado. Desde esta pantalla se puede acceder a eliminar ración, editar ración, añadir ración. Pulsando el botón de color naranja accedemos a editar ración. Pulsando el botón rojo acedemos a eliminar ración. y se pulsa el botón con el símbolo '+' accedemos a añadir ración.

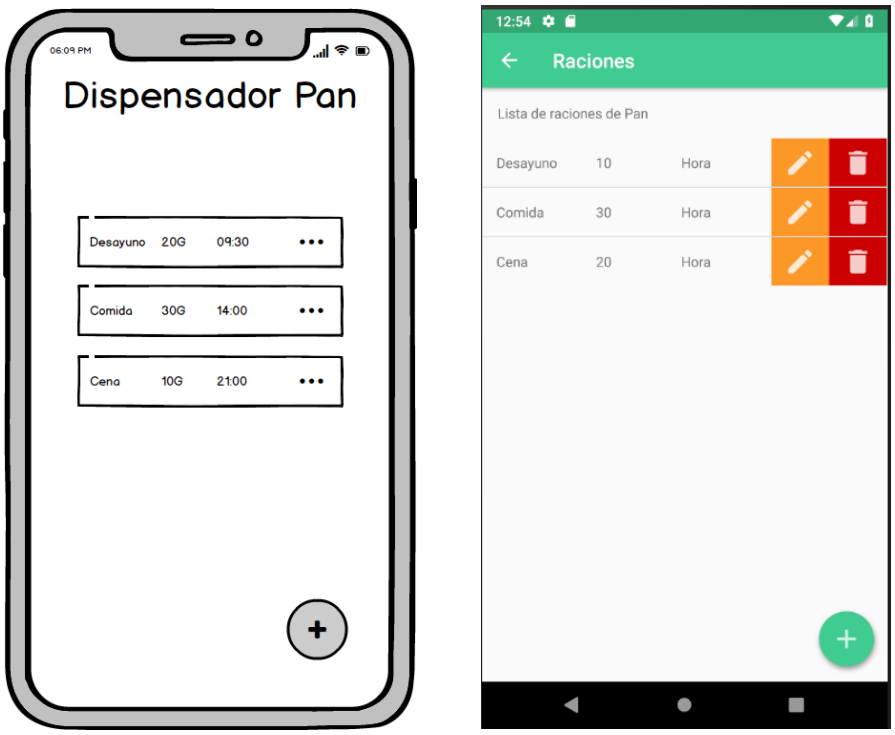


Ilustración 27 Consultar raciones

- **Actualizar ración:** En esta pantalla nos aparece los datos de la ración seleccionada en la cual podremos modificar los. Una vez actualizados se pulsará el botón de actualizar para confirmar los cambios, una vez echo los cambios volveremos a la pantalla de consulta de raciones.

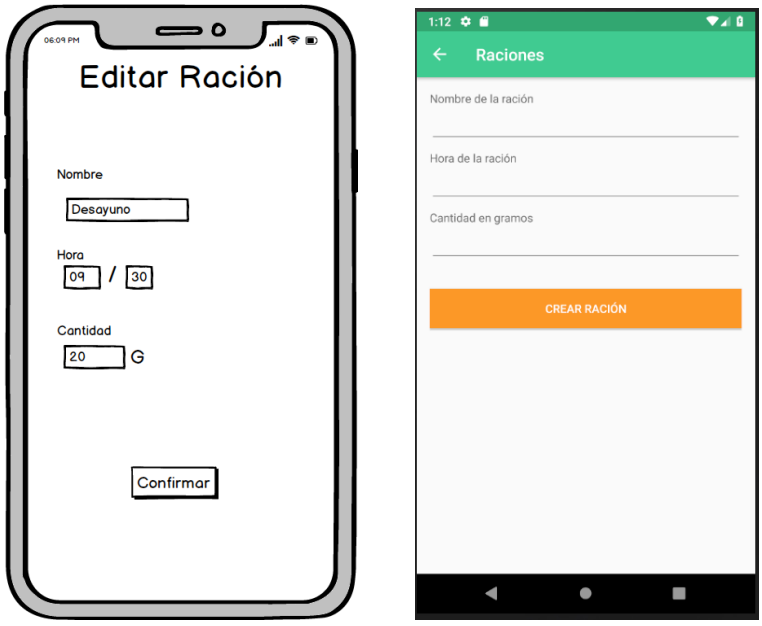


Ilustración 28 Actualizar ración

- **Eliminar ración:** En la ilustración 29, se observa que nos aparece una notificación de si se quiere eliminar la ración seleccionada. Una vez realizada la acción se volverá a la pantalla de consultar raciones.

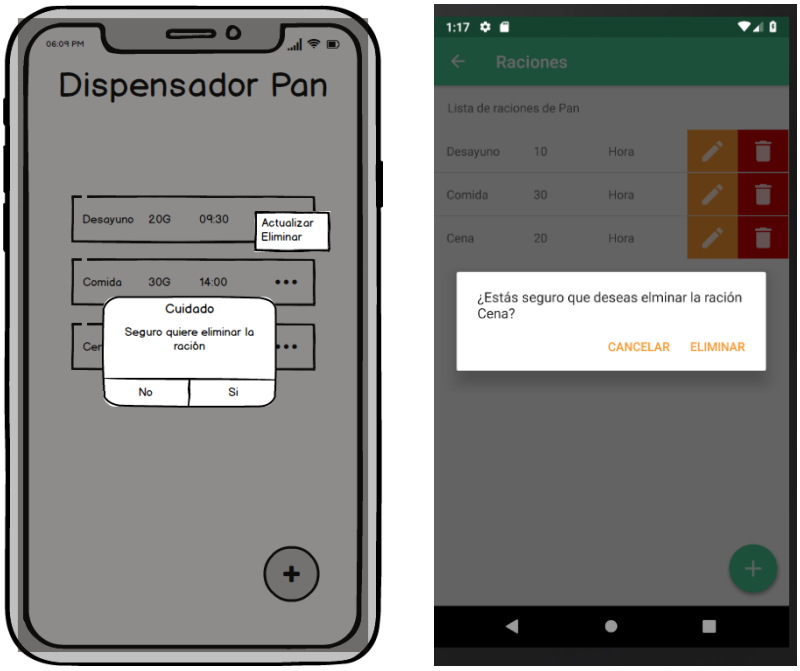


Ilustración 29 Eliminar ración

- **Nueva ración:** En la ilustración 30 nos aparecen los campos necesarios para crear la que son: el nombre, la hora, la cantidad a servir. Una vez finalizado se pulsa a crear ración y volveremos a la pantalla de consulta de ración.

Ilustración 30 Nueva ración

5. Tecnologías utilizadas

En este apartado se describirán las tecnologías utilizadas para el desarrollo de la solución, esta sección se dividirá en apartados: lenguaje, librerías, entornos de desarrollo y programas externos. Además, se comentará porque se eligieron estas tecnologías.

Ilustración 31 NODe.js

Node.js [19] es un entorno de ejecución para JavaScript, está orientado a eventos, por lo que es un lenguaje adecuado para el propósito ya que todas las conexiones se tratan como eventos. Además, se ha utilizado el framework express [20], que es un framework extensible de servidores HTTP. Las virtudes de Node.js gran rendimiento con pocos recursos, ya que por cada conexión se ejecutará un callback, estos permiten al entorno de ejecución seguir con el programa. De esta manera también se evitan bloqueos de entrada y salida, al comparar con el modelo de concurrencia que por cada llamada se ejecutara un hilo de ejecución nuevo el cual consume más recursos y además esperara las operaciones de entrada y salida haciendo la ejecución más lenta. También a tener en cuenta que Node tiene versiones estables para sistemas ARM, los dispositivos usados tienen esta arquitectura. Otros lenguajes que podrían hacer las mismas funciones serian C++, Ruby o Java.

Ilustración 32 NPM

Npm es un gestor de paquetes el cual facilita trabajar con Node.js, porque con una simple línea en la terminal instala las librerías disponibles y además ayuda a administrar módulos y agregar dependencias de manera sencilla, otra opción existente seria Yarn el cual funciona de manera similar, cualquiera de estas es buena opción. Pero se eligió Npm por llevar más tiempo en el mercado y más documentación.

Ilustración 33 ZMQ

Zmq es una librería de mensajería ligera, esta ofrece un alto rendimiento y baja latencia, además admite muchos escenarios ya que está disponible en más de 40 lenguajes de programación pueden comunicarse varios sistemas que no compartan el mismo lenguaje de programación de manera sencilla y rápida, otras opciones disponibles son MQTT o RabbitMQ.

Ilustración 34 Mongo

Las bases de datos utilizadas en el servidor o en los dispensadores es MongoDB, esta base de datos es de tipo no relacional, lo que conlleva ciertas ventajas y desventajas. Las ventajas son una velocidad de respuesta superior a una relacional, también los recursos consumidos son inferiores. Las desventajas serian que no tienen mucha fiabilidad en la atomicidad de los datos, pero al tratarse de un proyecto con pocas tablas no conlleva problemas. Posibles alternativas son Cassandra o Redis.

Ilustración 35 Visual Code

El editor de texto utilizado es visual studio code, este editor tiene una gran compatibilidad con Node.js pudiendo hacer autocompletado e incluso debugger [21]. Todo teniendo en cuenta que es gratuito. Otros editores a tener en cuenta son sublimeText o Atom.

Ilustración 36 Postman

Portman es la herramienta que se ha utilizado para testear la API ya que permite realizar consultas con todo tipo de detalle, pudiendo incluir cabeceras y atributos al cuerpo.

Ilustración 37 Arduino IDE

Para la programación del Arduino se ha utilizado su lenguaje propietario y el entorno de programación que proporciona la Web oficial llamado Genuino Arduino. La ventaja de este idle es que tiene completa compatibilidad con las placas ya que existen versiones. Y la carga de datos integrada.

Ilustración 38 Android studio

Para la programación de la aplicación se ha utilizada el entorno de desarrollo Android estudio. Este incorpora las herramientas necesarias para el desarrollo. Editor de interfaz, máquina virtual, etc.

LA siguiente sección se comentará las librerías más importantes usadas en el proyecto

- Serial-port para la comunicación con el Arduino.
- Momento para el manejo de las fechas y horas.
- Jwt-simple para la creación de los tokens.

- Moongose driver para la base de datos.
- Hx711 driver para la báscula.
- Bycrip encriptar datos.

6. Conclusiones

En este apartado de la memoria se presenta una evaluación global de los resultados. Se analizará el producto creado y se planteará una visión de futuro del mismo. Finalizando con una valoración de parte del autor.

En este documento se ha analizado la situación de los dispensadores en el mercado. Los cuales han permitido a los usuarios facilitar la tarea de mantener bien alimentadas a sus mascotas, mientras se ausentan durante el día e incluso durante varios. En el análisis se vieron los tipos de dispensadores con sus ventajas y desventajas. Con el resultado se eligió la opción que más se adaptó a las necesidades.

En general la solución que se ha aportado en el proyecto tiene un gran potencial, gracias a las funcionalidades que se han implementado. Siempre intentando mantener el coste del dispositivo lo más bajo posible.

Los trabajos futuros para este proyecto pueden ser muy variados. Se pueden mejorar aspectos del dispensador como poder incluir pilas o baterías. Otra opción se puede impermeabilizar para poder estar al aire libre o poner un depósito de agua. Como alternativa se puede adaptar para más aplicaciones, como un aviario, un acuario y para mascotas de mayor tamaño.

En la sección del cliente móvil se complementaría con una aplicación para IOS o crear una aplicación de escritorio para cuando se trate de empresas como protectoras y zoológicos.

Con relación a los estudios hechos durante la carrera, se han aplicado conocimientos sobre todo de redes y TSR(Tecnología de sistemas de información en la red) en el que se aprendieron Node.js, ZMQ y MongoDB. En la parte de análisis de requisitos no funcionales se han aplicado conocimientos adquiridos en AER (Análisis y Especificación de Requisitos).

La conclusión del trabajo a nivel personal es muy buena. Con el proyecto he tenido que aprender tecnologías que no se ven durante la carrera o al menos no de forma obligatoria las cuales son Android y Arduino. Además de profundizar en Node.js. Otro gran reto ha sido aprender a usar el hardware y cómo conectarlos con Node.js y las rutinas, todo esto pasando por el sistema operativo. Todo el conjunto de conocimientos

adquiridos durante la carrera hace que merezca la pena buscar otras alternativas tecnológicas para dar solución a un problema.

7. Bibliografía

[1] J. Ordóñez, «¿Qué es una API REST?», *Idento*, 04-may-2018. [En línea]. Disponible en: https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/.

[2] «Características Http», *Scribd*. [En línea]. Disponible en: https://es.scribd.com/document/302010917/Caracteristicas-Http.

[3] «ØMQ - The Guide - ØMQ - The Guide». [En línea]. Disponible en: http://zguide.zeromq.org/page:all.

[4] «JSON». [En línea]. Disponible en: https://www.json.org/.

[5] «Intro to API documentation». [En línea]. Disponible en: https://www.getpostman.com/docs/v6/postman/api_documentation/intro_to_api_documentation.

[6] K. Hokamura, *JWT(JSON Web Token) encode and decode module for node.js: hokaccha/node-jwt-simple*. 2018.

[7] «JSON Web Tokens - jwt.io». [En línea]. Disponible en: https://jwt.io/.

[8] «moment - npm». [En línea]. Disponible en: https://www.npmjs.com/package/moment.

[9] «MongoDB Documentation». [En línea]. Disponible en: https://docs.mongodb.com/.

[10] «bcrypt», *npm*. [En línea]. Disponible en: https://www.npmjs.com/package/bcrypt.

[11] «Using DEALER and ROUTER Sockets - zeromq». [En línea]. Disponible en: http://zeromq.org/tutorials:dealer-and-router.

[12] «Raspberry Pi Downloads - Software for the Raspberry Pi», *Raspberry Pi*.

[13] «Arduino Nano». [En línea]. Disponible en: https://store.arduino.cc/usa/arduino-nano.

[14] «Tutorial trasmisor de celda de carga HX711, Balanza Digital». [En línea]. Disponible en: https://naylampmechatronics.com/blog/25_tutorial-trasmisor-de-celda-de-carga-hx711-ba.html.

bogde, *An Arduino library to interface the Avia Semiconductor HX711 24-Bit Analog-to-Digital Converter (ADC) for Weight Scales.: bogde/HX711*. 2018.

[15] «¿Qué es una célula de carga y cómo funciona? Tipos de celda». [En línea]. Disponible en: https://es.omega.com/prodinfo/celulas-de-carga.html.

[17] «Serial port terminal > Cannot open /dev/ttyS0: Permission denied», *Ask Ubuntu*. [En línea]. Disponible en: https://askubuntu.com/questions/210177/serial-port-terminal-cannot-open-dev-ttys0-permission-denied.

«serialport - npm». [En línea]. Disponible en: https://www.npmjs.com/package/serialport.

[18] «Document», *Android Developers*. [En línea]. Disponible en: https://developer.android.com/reference/org/w3c/dom/Document.

[19] F. de Node.js, «Documentación», *Node.js*. [En línea]. Disponible en: https://nodejs.org/es/docs/.

[20] «express», *npm*. [En línea]. Disponible en: https://www.npmjs.com/package/express.

[21] «Node.js Extension Pack - Visual Studio Marketplace». [En línea]. Disponible en: https://marketplace.visualstudio.com/items?itemName=waderyan.nodejs-extension-pack.

8. Anexos

Es este apartado se expone los modelos 3D mas importantes para la creación del dispensador. Estos modelos se crearon, con la herramienta Blender.

Cuerpo

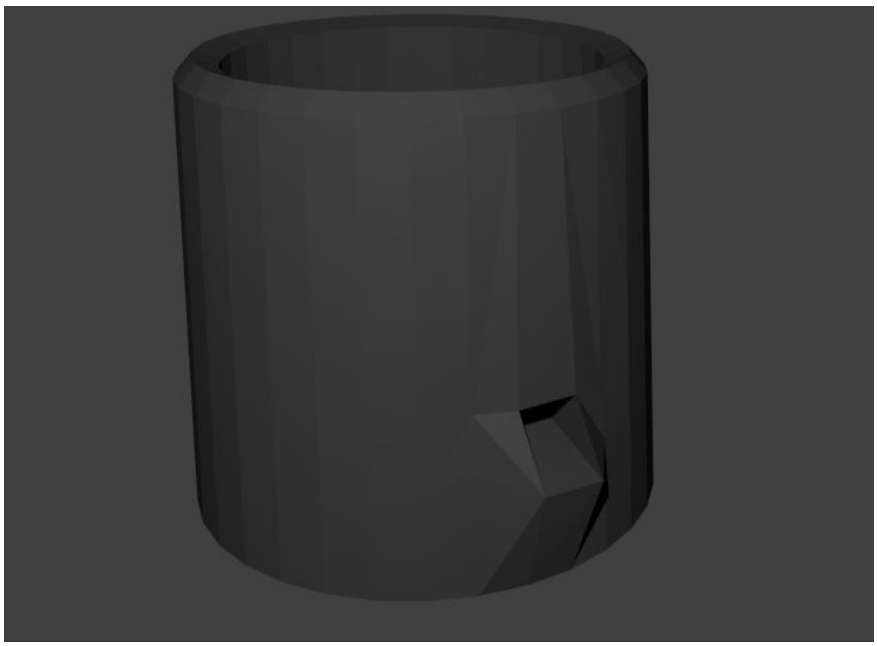


Ilustración 39 Cuerpo dispensador

Bandeja

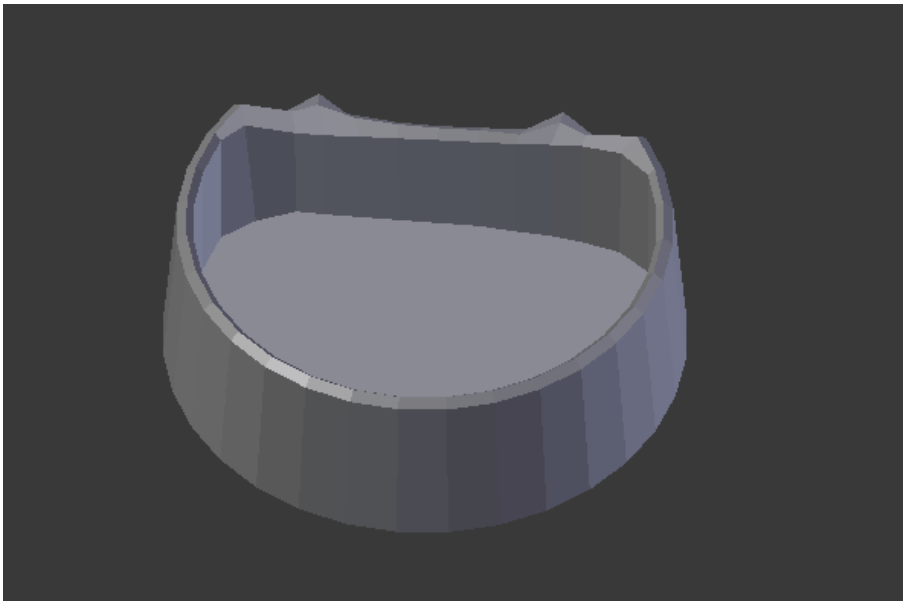


Ilustración 40 Bandeja

Rotor

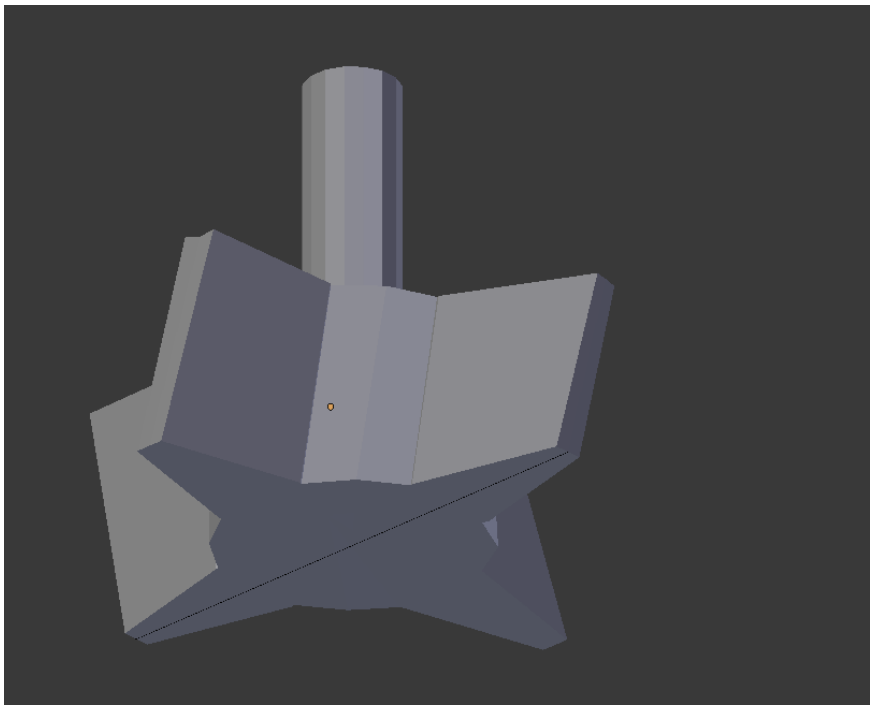


Ilustración 41 Hélice

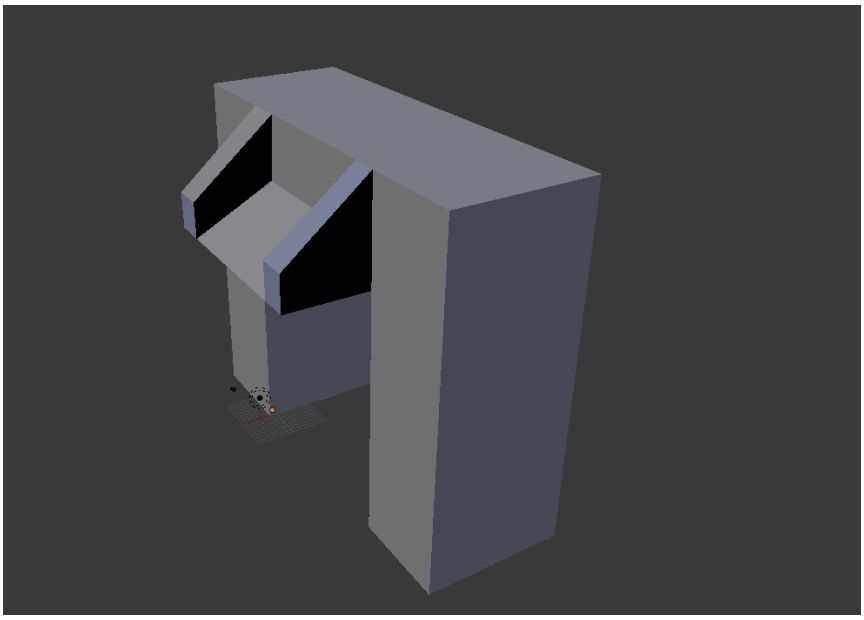


Ilustración 42 Soporte rotor

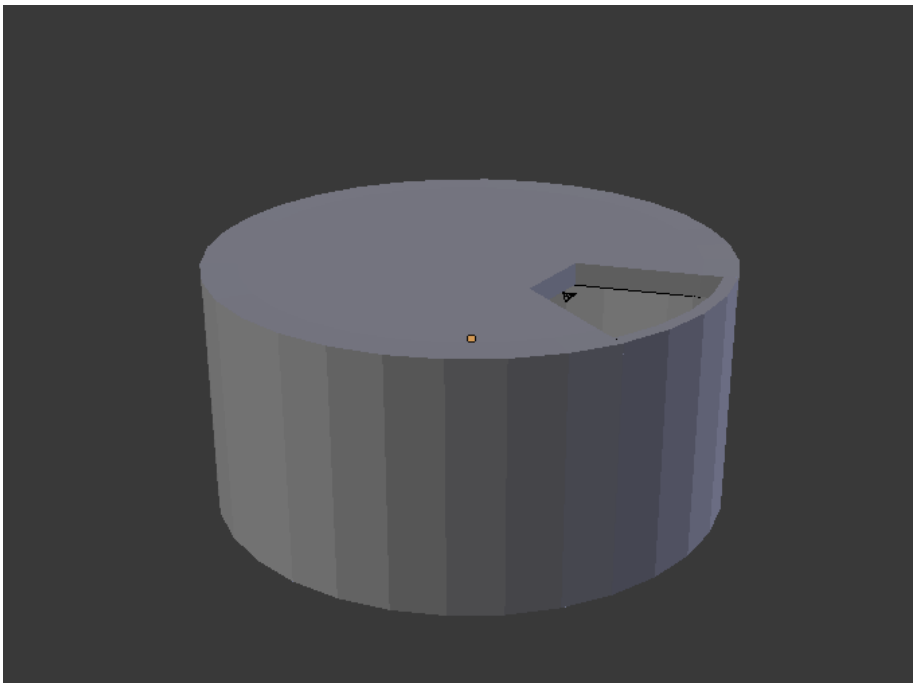


Ilustración 43 Cuerpo rotor

Soporte bascula

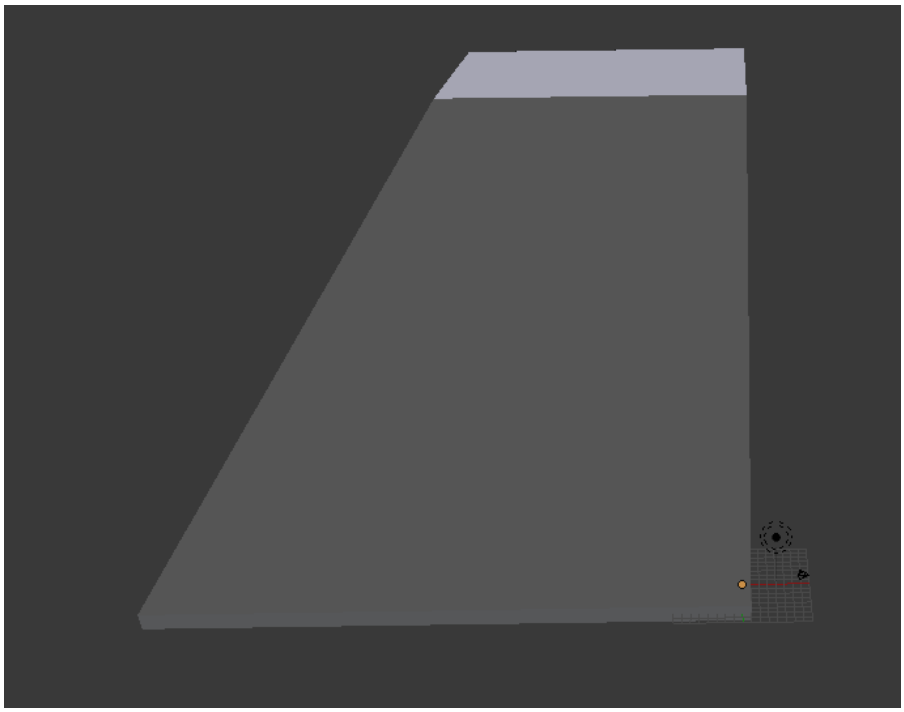


Ilustración 44 Soporte bascula

Cuenco

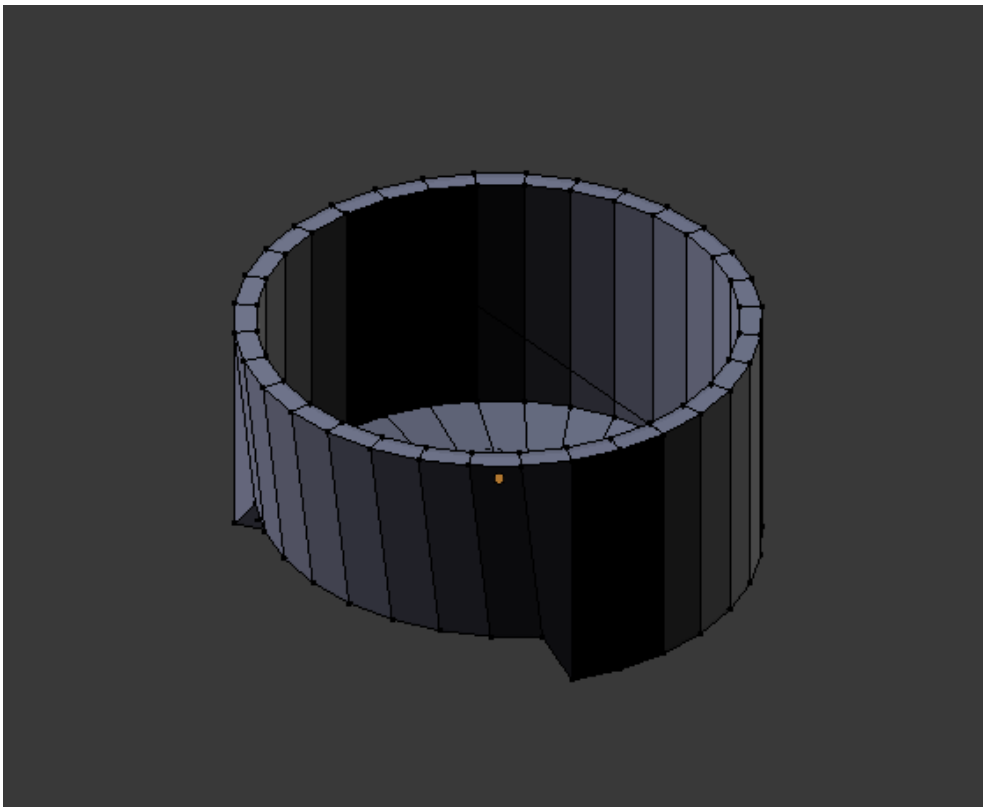


Ilustración 45 Cuenco

Fotos dispensador

Ilustración 46 Foto externa dispensador

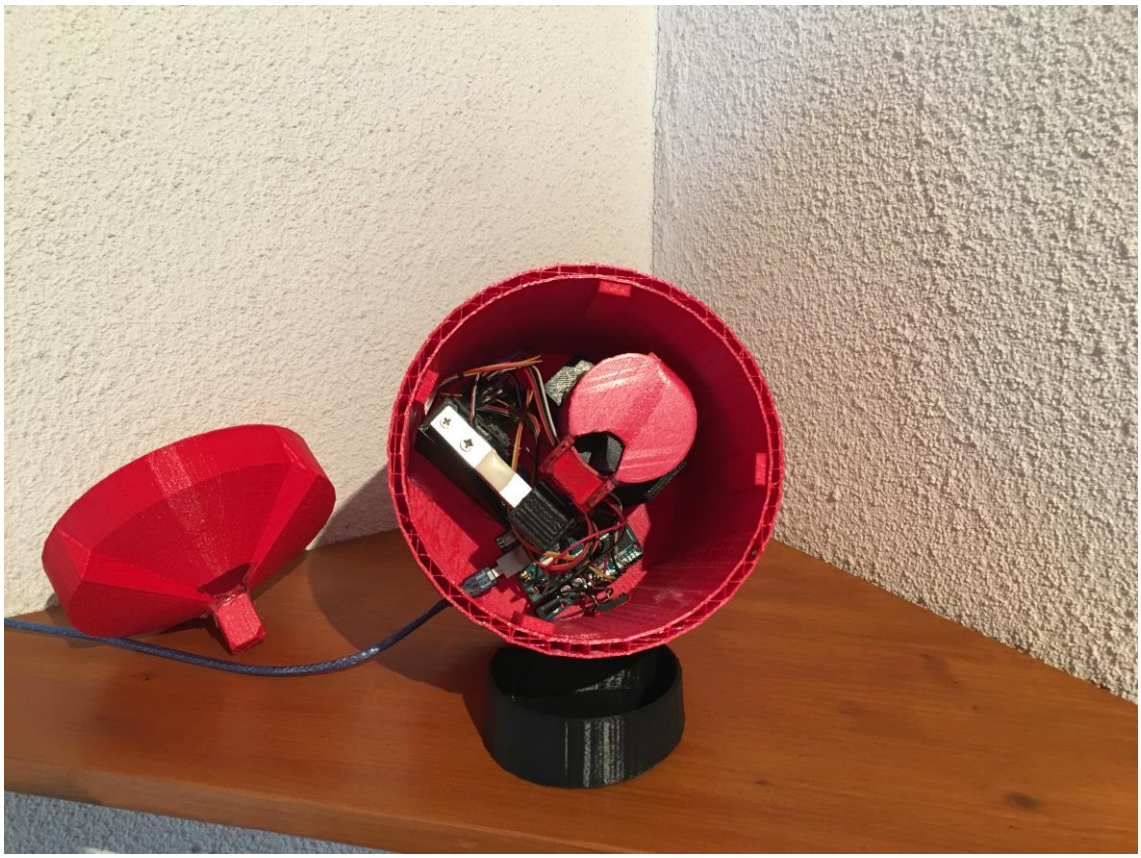


Ilustración 47 Foto interna dispensador