UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

# SMARTCART

## SISTEMA DE ASISTENCIA EN SUPERMERCADOS

GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

26 DE AGOSTO DE 2020

AUTOR: IGNACIO PAYÁ SILLA
TUTOR: PASCUAL PÉREZ BLASCO

**DOCUMENTOS**

1. **MEMORIA**

    1.1. ANEXO Nº1: MANUAL DE LA APLICACIÓN
    1.2. ANEXO Nº2: CÓDIGO ANDROID STUDIO
    1.3. ANEXO Nº3: CÓDIGO GOOGLE APPS SCRIPTS
    1.4. ANEXO Nº4: DATA-SHEETS

2. **PLANOS**

3. **PLIEGO DE CONDICIONES**

4. **PRESUPUESTO**

**AGRADECIMIENTOS**

Me gustaría agradecer en primer lugar a mi tutor, Pascual Pérez Blasco, la oportunidad que me ha brindado, haciendo posible la realización de este proyecto.

Por otro lado, también me gustaría agradecer todo el apoyo incondicional que he recibido por parte de mi familia; Mis padres Eduardo y Rosa María, mi hermana María y mi pareja Laura.

Por último, quisiera mencionar especialmente a mi compañero de trabajo, Fran Soler, por ayudarme con el diseño del concepto visual del carrito SMARTCART.

**RESUMEN**

Durante los últimos años hemos podido comprobar como nuestras tareas más cotidianas se han visto superpuestas por la tecnología. El proceso del avance del ser humano pasa actualmente por la digitalización de la información y la automatización de las actividades realizadas por éste. Se busca innovar y encontrar utilidades que revolucionen nuestra manera de vivir, pero en otros casos se procede a mejorar lo que ya tenemos. Es el caso de este proyecto.

Se hizo un estudio de cuál podría ser un caso como el explicado anteriormente. El resultado fue el siguiente: Digitalizar la compra en un supermercado.

En este trabajo se va a desarrollar una **aplicación** informática que contendrá todos los productos de un supermercado, con la información necesaria de cada producto (incluyendo su localización). El sistema estará acoplado a los carros del centro que a su vez dispondrán de un sistema de **sensorización** aplicado en la estructura para poder llevar el cálculo de la compra. Los carros recibirán el nuevo nombre de **SMARTCART**.

Además, para hacer más eficiente la compra, el centro dispondrá de localizadores de posición con los que se habilitará un listado de productos dividido por zonas y se informará del cajero al que acudir en caso de finalización.

Por último, estos localizadores llamados "***beacons"*** (balizas electrónicas) enviarán mensajes a los SMARTCARTS de los clientes, como pueda ser, por ejemplo, nuevas ofertas disponibles.

**Palabras clave:** SMARTCART, tecnología BLE, beacons, aplicación

**ABSTRACT**

In recent years we have been able to see how our most daily tasks have been overlapped by technology. The process of human advancement currently goes through the digitization of information and the automation of the activities carried out by it. We seek to innovate and find utilities that revolutionize our way of life, but in other cases we proceed to improve what we already have. It is the case of this project.

A study was made of what could be a case like the one explained above. The result was the following: Digitize a supermarket purchase.

In this work, it will be developed a mobile **app** which will contain all the products of a supermarket, with the necessary information for each product (including its location). The system will be coupled to the trolleys of the center, which in turn will have a **sensorization** system applied to the structure in order to carry out the purchase calculation. The carts will be renamed **SMARTCART**.

In addition, to make the purchase more efficient, the center will have position locators that will enable a list of products divided by zones and the cashier that the client must coming up will be informed to in case of completion.

In conclusion, these locators called "***beacons***" (electronic beacons) will send messages to the SMARTCARTS, such as, for example, new offers available.

**Keywords:** SMARTCART, BLE technology, beacons, App

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

# MEMORIA

SMARTCART

GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

26 DE AGOSTO DE 2020

AUTOR: IGNACIO PAYÁ SILLA
TUTOR: PASCUAL PÉREZ BLASCO

**ÍNDICE**

**ÍNDICE IMÁGENES**

**ÍNDICE TABLAS**

## 1. OBJETIVO

El objetivo de este trabajo consiste en el desarrollo de una aplicación informática insertada en un prototipo, implementado en los carros de un supermercado.

El prototipo dispondrá de un sistema de **posicionamiento** en **interiores** desarrollado para interiores que funciona mediante aplicaciones **bluetooth.** Este método permitirá que la aplicación controle la localización del carro, proporcionando el listado de compra correspondiente y las diferentes ofertas disponibles. Además, los carros tendrán en su estructura un sensor acoplado para leer los productos que se incorporen en la cesta y así realizar el cálculo de la compra. Una vez finalizada la compra, la aplicación informará al cliente del cajero al que deberá acudir y a su vez, ésta enviará a dicho cajero el resultado de la compra con la información necesaria de la operación. Al llegar, en las bases de los cajeros, se realizará un pesaje del carro mediante una báscula industrial y se comprobará que la información enviada al servidor es compatible con el resultado para proceder al pago.

Como es obvio, la aplicación requerirá de dos **servidores**. Una para los empleados del supermercado, con la que podrán insertar un usuario y contraseña establecidos en la base de datos del supermercado y una vez registrados, utilizar cualquiera de los cajeros del centro; el otro servidor es para los clientes. Los servidores deberán estar conectados entre sí, para que se actualicen al tiempo los cambios realizados.

Como objeto opcional, se añadirá un sistema de pago con tarjeta al carro para agilizar en mayor medida el resultado de la compra. Además de un registro de los clientes en la base de datos de la aplicación que almacene los resultados de las compras y los recorridos seguidos para proporcionar información al cliente en las siguientes visitas.

## 2. ANTECEDENTES

Durante los últimos años hemos podido comprobar que las aplicaciones para móviles suponen una comodidad en todas nuestras tareas diarias, por ejemplo:

- Aplicaciones bancarias para controlar los gastos domésticos
- Aplicaciones para realizar compras por internet
- Regulación de la luz de casa, los televisores, el coche, etc.
- Avisos por salto de alarmas

Tenemos que agradecer estos sistemas a la innovación en la telefonía móvil, en concreto a los **Smartphones**. Hay que remontarse hasta el 9 de enero de 2007, donde Steve Jobs, creador de Apple presentaba por primera vez al mundo un Smartphone, el primer **Iphone** y donde dejó en constancia una de las frases más importantes hasta la fecha del siglo XXI: “Hemos reinventado el teléfono”. Actualmente podemos afirmar que estaba en lo cierto.

La revolución tecnológica ha llegado a nuestras vidas cotidianas y esto conduce hacia una evolución que propicia reestructuraciones en ciertas actividades, en lo que a este proyecto se refiere, la realización de la compra en los supermercados. Cada vez se producen más compras por internet diarias, pero sigue siendo mayor la cantidad de gente que prefiere el método tradicional. Es el motivo principal por el cual una solución apropiada para avanzar es la de trasladar la tecnología a los supermercados.

Uno de los inconvenientes que se presentan es el de la localización interior del supermercado. Durante los últimos años, son varias las empresas que se han centrado en encontrar sistemas que trabajen en

zonas cerradas de una manera correcta, sin producir errores u obstrucciones por "zonas muertas". Otra vía ha sido la de mejorar los ya existentes, como son el caso de la localización por señales bluetooth. Una de las empresas con las que he contactado para recibir información apropiada sobre este sistema es la empresa **WITRAC**, en Valencia capital, que se dedica en su mayor medida a trabajar en sistemas de geolocalización con otras industrias.

## 3. ESTUDIO DE NECESIDADES Y FACTORES A CONSIDERAR

En el siguiente apartado, encontraremos una serie de condiciones o normas que deben ser cumplimentadas por las dos partes del proyecto (**Hardware** y **Software**):

### 3.1. HARDWARE:

- Sistema del carro:
  - La carcasa del carro debe cubrir y proteger la pantalla, sensores, placas de carga y cableado interior. La normativa debe estar cumplimentada por el proveedor del material.
  - Se controlará mediante la aplicación Software a través de la pantalla.
  - Dispondrá de un código de numeración para disponer de un listado de todos los carros.
  - Los conectores deben de ser los homologados por los fabricantes de cada componente, cumpliendo también con los valores de carga estipulados en las hojas de características.
  - Debe cumplir con las medidas estipuladas para la carga de productos del supermercado en los documentos oficiales:
    - **UNE-EN 1929-1:1998 "Carros de supermercado. Parte 1: Requisitos y ensayos para carros de supermercado con o sin asiento para niños**".
    - **UNE-EN 1929-2:2005 "Carros de supermercado. Parte 2: Requisitos, ensayos e inspección para carros de supermercado con o sin asiento para niños, destinados a ser utilizados en cintas transportadoras de pasajeros".**

- Sistema de Geolocalización:
  - El sistema **BLE** (Bluetooth Low Energy) debe estar repartido por la zona comercial, localizado en las diferentes secciones del centro y en las salidas. El proveedor proporcionará el equipamiento con sus respectivas protecciones y asumirá la normativa establecida.

- Estación de carga:
  - Se debe cumplir con el **Real Decreto-ley 15/2018,** del 5 de octubre, de medidas urgentes para la transición energética y la protección de los consumidores.
  - Deben establecerse con el supermercado zonas específicas para el almacenaje de los carros, donde se colocará la estación de carga.

### 3.2. SOFTWARE

- Aplicación Móvil:
    - Debe contener dos servidores: uno para los trabajadores y otro para los clientes.
    - Debe tener registrado el código numérico del carro en el que se encuentra.
    - Las interfaces que aparezcan en todo momento deben ser claras y sencillas para facilitar el proceso al cliente.
    - Las acciones de las que debe disponer son las siguientes: Listado de productos con su información correspondiente; Localización del carro; Lista de la compra realizada en todo momento, con la posibilidad de modificarla en cualquier momento; Envíos de ofertas disponibles en las secciones diferentes del supermercado; Cálculo del precio y peso de la compra; Finalización de compra; Sugerencia de cajero más óptimo y envío de la información de la compra.
    - Debe disponer de un sistema de ahorro de energía en caso de no uso.

## 4. PLANTEAMIENTO DE SOLUCIONES ALTERNATIVAS Y JUSTIFICACIÓN DE LA SOLUCIÓN ADOPTADA

### 4.1. CARCASA DE PROTECCIÓN DEL MATERIAL DEL CARRO

Los prototipos que van a ir colocados en los carros requieren de un aislamiento protector que sea resistente, transmita bien el calor y sobre todo que no contamine. Es por ello que se ha realizado un estudio exhaustivo con varios de los posibles materiales a poder utilizarse, realizando cálculos concisos para obtener la mejor elección. A continuación, se expondrán las opciones acompañadas por una tabla argumentativa:

#### 4.1.1. PLÁSTICO TÉCNICO

Es un tipo de plástico que ofrece un mayor rendimiento que los plásticos de tipo estándar, pues ofrece unas características con las que se mejora su rendimiento frente a cambios de temperatura, oposición a daños por productos químicos, resistencia a impactos y fuerza mecánica. Durante los últimos años ha ido imponiéndose a otros materiales tradicionales en el ámbito técnico como lo son la madera o el metal, no solo por su relación peso/fuerza, sino también por su agilidad en la fabricación ya que permite ser moldeado con estéticas de mayor dificultad.

Es frecuente en las aplicaciones industriales, en el transporte, la construcción y la fabricación. Además, es utilizado también para productos eléctrico/electrónicos.

*Imagen 1: Plásticos Técnicos*

### 4.1.2. RESINAS EPOXI

Este material tiene una gran propiedad sobre el resto (por lo menos en mayor medida) y es una gran cantidad de propiedades con las que se puede fabricar. Puede ser fabricado como un líquido con baja viscosidad o perfectamente como un sólido muy resistente.

Este material ya es utilizado para el revestimiento en los carros de supermercado, por lo que supondría una ventaja frente al resto de materiales, además de que se usa para pinturas especiales, con lo cual protege de forma excelente.

Son resistentes frente al calor y a propiedades químicas, se adhieren ágilmente y contienen un correcto aislamiento eléctrico. Gracias a sus posibles combinaciones en la creación, pueden añadirse elementos que permitan la conductividad eléctrica con la que mejoren el funcionamiento del sistema y lo conviertan en una excelente propiedad para las aplicaciones electrónicas.

### 4.1.3. ALUMINIO

Sin lugar a dudas, este es probablemente el material más demandado en todos los ámbitos. Presenta unas características atractivas, como su ligereza, resistencia, maleabilidad y conductividad térmica y eléctrica. En este último caso, está capacitado para competir frente al cobre en prestaciones y en el precio.

Además, es un material completamente reciclable con el que no se pierden sus propiedades y, por tanto, no reduce sus cualidades con su uso.

*Imagen 2: Aluminio*

### 4.1.4. COMPARACIÓN Y ELECCIÓN

A continuación, se expone una tabla con las valoraciones de los tres materiales para fabricar la carcasa para una serie de factores que influyen en el resultado final. Los factores basados para encontrar los resultados óptimos son los siguientes:

- **Temperatura**: Se refiere a la capacidad que tiene el material para soportar temperaturas altas y aislar del calor.
- **Resistencia**: Estudia la fuerza y dureza que posee el material.
- **Punto de influencia**: Es el límite elástico que soporta el material frente a cambios de presión. Estudia la capacidad que tiene de retornar a su estado normal o al punto que es capaz de llegar hasta que se rompe.
- **Ductilidad**: La capacidad de deformación sin llegar a la ruptura.

- **Maleabilidad**: Es un estudio que se realiza para comprobar la variedad de formas o cambios en la estructura que se pueden realizar con el material.
- **Peso**: El exceso de peso penaliza. Por tanto, es relevante conocer cuál de los materiales pesa menos con la misma cantidad.
- **Precio**: Es el factor más importante, según el criterio de fabricación y venta, por lo que es relevante conocer los precios que disponen los materiales a exponer.
- **Fabricación**: Factor que evalúa la complejidad, duración y manipulación a la hora de fabricar productos que contengan estos materiales.
- **Reciclaje**: Importante para el medioambiente conocer la capacidad de reciclado de los materiales.
- **Disponibilidad**: Estudia la facilidad de obtención del material para realizar el proyecto.

| FACTORES | VALOR RELATIVO (%) | MATERIALES | | |
|---|---|---|---|---|
| **TÉCNICOS** | **67%** | **P.TÉCNICOS** | **R.EPOXI** | **ALUMINIO** |
| **Temperatura** | **15%** | 10 | 8 | 5 |
| **Resistencia** | **15%** | 8 | 6 | 10 |
| **Ductilidad** | **12%** | 8 | 5 | 9 |
| **Pto de influencia** | **12%** | 7 | 5 | 8 |
| **Maleabilidad** | **8%** | 9 | 9 | 7 |
| **Peso** | **5%** | 10 | 9 | 6 |
| **OTROS** | **33%** | | | |
| **Precio** | **20%** | 9 | 8 | 9 |
| **Fabricación** | **5%** | 8 | 8 | 7 |
| **Reciclaje** | **5%** | 9 | 9 | 10 |
| **Disponibilidad** | **3%** | 8 | 10 | 8 |
| **RESULTADO** | **100%** | **8,61** | **7,22** | **8,04** |

*Tabla 1: Carcasa de protección*

La justificación de los resultados es la siguiente:

- **Temperatura**: El metal se calienta con mayor facilidad que el plástico y puede provocar incomodidad a la hora de ser utilizado. En el caso de las Resinas Epoxi, absorben más rápidamente el calor y esto provoca que se mantenga a una temperatura mayor durante más tiempo. Los plásticos técnicos están preparados con material específico que resiste al calor y no permite su traspaso o absorción, al menos en menor medida que el resto.
- **Resistencia**: Frente a golpes o arañazos, el metal es más resistente, aunque en el momento en el que queda marcado es más visible que en el plástico, pues recibe el mismo color por dentro que por fuera. Las resinas Epoxi se componen de materiales más débiles que los plásticos técnicos, por lo que reciben menor puntuación.
- **Ductilidad**: El metal es más duro y, por tanto, más complicado de romper, aunque más susceptible de soportar golpes y abolladuras. El plástico técnico es más frágil y puede romperse en varios trozos, al igual que las resinas Epoxi, solo que estas últimas con mayor facilidad.

- **Punto de influencia**: El plástico técnico tiene una capacidad inédita para regresar a su estado original tras sufrir alguna deformación, con lo que tras sufrir un impacto no mayor al de su límite elástico, no puede apreciarse cambio alguno. Con las resinas Epoxi, aumenta su capacidad de deformación según su viscosidad en el material, pero en el caso de los plásticos sólidos, poseen menor resistencia que los técnicos.
- **Maleabilidad**: En general, los plásticos, debido a su viscosidad, permiten recibir formas con mayor agilidad que los metales. El aluminio, por suerte, es un metal que permite ser moldeado con mayor facilidad que otros metales, aunque no lo suficiente para superar la capacidad de los plásticos.
- **Peso**: El plástico es más ligero que el metal, no tiene mayor cabida. En el caso de los plásticos técnicos, sus propiedades permiten reducir el peso frente a otros tipos, lo que supone que reciba la mayor puntuación en este apartado.
- **Precio**: Los metales son más caros que los plásticos, pero en el caso del aluminio, al ser un recurso muy práctico y con un reciclaje del 100%, supone una reducción en efectivos frente a otros materiales. Las resinas Epoxi resultan ser un poco más caras debido a la gran cantidad y variedad de productos que se emplean en su fabricación.
- **Fabricación**: Los plásticos son más agiles de fabricar una vez se obtengan los productos con los que darles forma. Las temperaturas y las condiciones son mejores frente a la fabricación de los metales, con lo que desembocan en un confort mayor.
- **Reciclaje**: En los 3 casos nos encontramos con facilidad de reciclaje, pero los plásticos pierden algunas propiedades en sus condiciones conforme su reutilización, al contrario que el aluminio, pues en cualquier situación, sus propiedades siguen intactas y disponibles con la misma capacidad que en su primer uso.
- **Disponibilidad**: Es el factor con menos valor, pues no es muy relevante, ya que los 3 tipos de materiales a escoger tienen total disponibilidad. Sin embargo, los carros de los supermercados se fabrican con plástico hecho mediante resinas Epoxi, con lo que se podría trabajar en el proyecto con materiales reciclados de carros de supermercado.

### 4.2. SISTEMA HARDWARE PARA CONTENER EL CÓDIGO SOFTWARE

Para la elección del sistema Hardware se han consultado a expertos en la materia, con los que se ha razonado ciertos aspectos que demandan las empresas hoy en día.

En este apartado, simplemente se han barajado dos ideas con las que trabajar, pero las opciones reales son más amplias. Para explicar la decisión tomada, se van a presentar a continuación las dos opciones escogidas y en adición, una tabla con las ventajas e inconvenientes de cada uno de ellos:

### 4.2.1. SMARTPHONE PROPIO

El mismo título lo indica. Simplemente trataría de suponer una solución práctica para los clientes. Consistiría en instalar la aplicación en un Smartphone propio y acoplarlo en un enganche que se situaría en la estructura de los carros del supermercado.

*Imagen 3: Smartphone "Iphone X"*

### 4.2.2. TABLET INCORPORADA EN EL CARRO

En este caso, se trataría de un sistema incorporado ya en el prototipo. Los clientes no tienen la necesidad de proporcionar nada. Además, en todos los carros iría acoplado el mismo sistema y no requeriría de cambios específicos. La Tablet escogida para este sistema, es el modelo Tablet **PIPO X4 64GB Intel Cherry Trail T3 Z8350** con cuatro núcleos y **10.1 pulgadas.** Sistema **Windows 10 IP67.**

*Imagen 4: Tablet industrial PIPO X4*

### 4.2.3. COMPARACIÓN Y ELECCIÓN

Una vez presentadas las dos opciones a escoger, pasamos a exponer las ventajas e inconvenientes que nos encontramos para el proyecto:

| | SMARTPHONE PROPIO | SMARTPHONE FIJO |
|---|---|---|
| **VENTAJAS** | - Uso práctico y cómodo para cada cliente<br>- Resultado del proyecto más económico | - Comodidad para el cliente, no necesita utilizar su propio Smartphone<br>- Seguridad de la empresa respecto al uso del sistema por parte de los clientes<br>- Fabricación de los prototipos universal, sin excepciones o especificaciones variadas |
| **INCONVENIENTES** | - Exigencia hacia los clientes de tener un Smartphone para utilizar el sistema<br>- Problemas de señal o cobertura dentro del centro<br>- Obligación de tener batería en el móvil en su momento de uso<br>- Requerimiento de descarga de la App | - Aumento del presupuesto en el proyecto<br>- Necesidad de baterías para recargar las tablets |

*Tabla 2: Ventajas e inconvenientes SMARTPHONE*

Por los resultados marcados en la tabla, se deduce que la mejor opción (a pesar de sus inconvenientes), es la de incorporar a los prototipos una Tablet fija que funcione, única y exclusivamente, para la aplicación del supermercado.

Aunque suponga un aumento en el presupuesto, a la larga supone una ventaja para los intereses del supermercado.

## 4.3. PROGRAMA Y CÓDIGO

El diseño de la aplicación móvil puede realizarse con diversos programas preparados para ello. Son muchos los casos, cada uno con un nivel de resultados variado. Un valor importante para escoger correctamente u óptimamente es el tipo de lenguaje con el que trabaje el programa. Con lo que, para este caso concreto, disponemos de dos opciones para escoger, de las cuales vamos a exponer las características a continuación.

#### 4.3.1. PROGRAMA Qt

El programa Qt es un sistema que se utiliza para desarrollar programas que trabajan con interfaces gráficas de usuario. Se escribe mediante lenguaje en C, con lo que supone una herramienta práctica para los diseñadores electrónicos de softwares informáticos, utilizado por los desarrolladores de la empresa nórdica NOKIA.


*Imagen 5: Logotipo Qt creator*

#### 4.3.2. PROGRAMA ANDROID STUDIO

Consiste en una Plataforma desarrollada para Google para Android que trabaja con un entorno de desarrollo integrado con software libre de código abierto. Es bastante cómodo para trabajar, pues permite programar con diversos lenguajes. En concreto, mediante Java, Kotlin y C++. Sin embargo, para el desarrollo de Apps, es recomendable el uso de Java para la escritura del código informático.

#### 4.3.3. ELECCIÓN Y JUSTIFICACIÓN

En este caso, la elección del programa no ha dependido de las características de los sistemas, simplemente ha sido por comodidad frente a la obtención del programa. La plataforma de Android Studio se encuentra disponible para Microsoft Windows con total libertad, descarga gratuita y con información y documentación completa disponible para su uso.

Además, la Tablet escogida para el prototipo del carro contiene sistema software Android, por tanto, agiliza la elección del programa con el que trabajar.

### 4.4. BASE DE DATOS

El proyecto requiere de una base de datos con toda la información disponible sobre los productos que hay disponibles en el supermercado. Los propios supermercados son los responsables de realizarla y proporcionarla para poder completar el diseño de la aplicación. Sin embargo, este estudio se realiza para el caso en el cual el supermercado no disponga de una base de datos propia con sus productos. Además de implementar el diseño de la App, se añade una base de datos práctica.

Las dos opciones a escoger son las siguientes:

#### 4.4.1. SISTEMA MySQL

Consiste en un programa que realiza gestiones de bases de datos, desarrollado por la empresa **Oracle Corporation** que trabaja con el sistema de lenguaje C++ y que resulta ser de los más demandados para crear bases de datos. Es realmente práctico, pues permite relacionar varios documentos creados y

contiene una protección específica que ayuda a mantener la privacidad de la información que contiene. Requiere de un servidor propio donde almacenar los archivos.


*Imagen 6: Logotipo MySQL*

#### 4.4.2. SISTEMA GOOGLE SHEETS + GOOGLE APPS SCRIPTS

Es otro programa habilitado por la empresa **Google** con el que se pueden crear hojas de cálculo que hagan la función de bases de datos almacenadas en el Google **Drive**. Para poder manipular estas hojas de cálculo desde la aplicación, se necesita otra plataforma que permita utilizar esas funciones, que es mediante el programa **Google Apps Scripts.** Trabaja de una forma parecida a MySQL, pero la gran ventaja se encuentra en su plataforma de almacenaje. Sube a la nube todos los archivos que se completan.

#### 4.4.3. ELECCIÓN Y JUSTIFICACIÓN

Para la decisión de cuál sería el programa más conveniente a escoger en este proyecto, toda la balanza se ha decantado por el sistema GOOGLE SHEETS. Y el motivo no es otro que la comodidad en los costes de mantenimiento. En cuestión al tema de la programación y calidad, el resultado sería diferente, pero como para este proyecto no se precisa de una base de datos con urgencia, no es necesario buscar el mejor programa disponible.

### 4.5. GEOLOCALIZACIÓN PARA INTERIORES

En este proyecto, la clave de todo es el sistema de localización de la que disponga el centro. La aplicación dependerá de este sistema, por lo que es muy importante acertar en el método que se va a utilizar. Se realiza un estudio completo de tres teorías diferentes con las que se puede trabajar, comprobando así cuál de los tres es la opción más óptima.

#### 4.5.1. TECNOLOGÍA BLE (BLUETOOTH LOW ENERGY)

Consiste en una tecnología ***wireless*** que se constituye por sistema Bluetooth 4.0 y funciona en el rango de las frecuencias de 2,4 GHz. Trabaja con baja energía usando **transponders** que emiten un **ping rate** ajustable. Se basa en un proceso de triangulación frecuencial, con el que la información se transmite a través de ***beacons*** (balizas electrónicas) trabajando con el modelo **Eddystone**. Es un recurso muy utilizado durante los últimos años, sobre todo en almacenes, donde se controlan los productos que

entran o salen, registrando la información en una base de datos. Su punto fuerte lo encontramos en la distancia que puede llegar a albergar (hasta 100 metros).


*Imagen 7: Beacon (baliza electrónica)*

### 4.5.2. SISTEMA ZIGBEE

Esta tecnología se basa en protocolos de alto nivel de comunicación, se usa en la radiodifusión digital de datos, que al igual que la tecnología BLE, trabaja con baja energía. Trabaja en la banda **ISM**, a 2,4 GHz de frecuencia. Funciona mediante envío de datos a través de nodos móviles. Debe haber nodos fijos que controlen los pasos de estos nodos móviles y en adición un nodo base, que proporcione la señal a todos los nodos fijos. Utilizar este recurso supone un ahorro en costes significativo, pero nos encontramos ante la tesitura de que la tecnología BLE ha pasado por encima de ésta. Es una tecnología obsoleta, por así decirlo.


*Imagen 8: Tecnología Zigbee*

### 4.5.3. FOTOEMISIÓN (LED)

Probablemente, este sea el recurso más complicado y menos eficaz. Se trata de un sensor óptico de tipo semiconductor y que produce un efecto fotoeléctrico. Requiere de un optoacoplador, que consiste en un dispositivo de emisión y recepción que funciona como un interruptor siempre activado debido a la luz emitida por un diodo led. Esta luz actúa sobre el dispositivo, saturando un **fototriac**, que es un fototransistor (componente optoelectrónico).

El sistema se encuentra acoplado en un circuito integrado con el cual se transmite la información de la posición al dispositivo que contiene la aplicación, mostrando así en la imagen la posición del cliente dentro del supermercado.

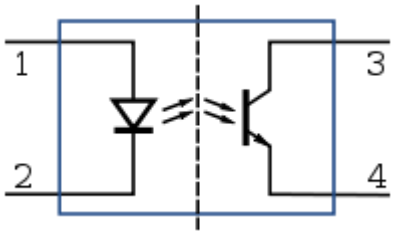



*Imagen 9: Símbolo electrónico del optoacoplador*

### 4.5.4. COMPARACIÓN Y ELECCIÓN

Al igual que en casos anteriores, se ha completado una tabla para visualizar la comparativa de los resultados en los materiales para diversos factores a cumplimentar. Estos factores que se han seguido se presentarán a continuación, comentando brevemente el motivo por el que se utilizan en la valoración:

- **Distancia**: Evalúa la distancia a la que puede llegar a recibir la señal del prototipo al cual debe transmitir la información.
- **Cantidad**: El número de *beacons*, nodos o fotoreceptores que deben incorporarse en la zona del supermercado para geolocalizar los carros.
- **Interferencias**: Factores internos o externos que afectan a las señales de los receptores, como por ejemplo: ruido, polvo, altas temperaturas, toxicidad de los materiales, etc.
- **Tiempo de respuesta**: Tiempo que tarda en recibir la señal el receptor o en enviarla el emisor.
- **Temperatura**: La cantidad de calor que llega a transmitir el prototipo.
- **Precio**: Coste del montante.
- **Montaje**: Dificultad frente al montaje del sistema y cantidad involucrada.
- **Disponibilidad**: Facilidad a la hora de encontrar el sistema en el mercado por un precio adecuado al comprador.

| FACTORES | VALOR RELATIVO (%) | MATERIALES | | |
|---|---|---|---|---|
| **TÉCNICOS** | **60%** | **T.BLE** | **T. ZIGBEE** | **FOTOEMISOR** |
| **Distancia** | **15%** | 10 | 8 | 3 |
| **Cantidad** | **15%** | 8 | 8 | 5 |
| **Interferencias** | **10%** | 8 | 6 | 5 |
| **T. de respuesta** | **10%** | 10 | 9 | 7 |
| **Temperatura** | **10%** | 9 | 9 | 6 |
| **OTROS** | **40%** | | | |
| **Precio** | **20%** | 7 | 8 | 6 |
| **Montaje** | **15%** | 8 | 8 | 5 |
| **Disponibilidad** | **5%** | 9 | 7 | 8 |
| **RESULTADO** | **100%** | **8,45** | **7,95** | **5,35** |

*Tabla 3: Sistema de Geolocalización*

Una vez completada la valoración, encontramos como mejor posicionado para la decisión del montante de la geolocalización al sistema basado en tecnología BLE, con un resultado de 8,45. Las justificaciones que respaldan esta decisión son las siguientes:

- **Distancia**: La tecnología BLE destaca sobre el resto por sus avances en la ampliación de la distancia de localización, llegando a alcanzar valores cercanos a los 100 metros. La tecnología Zigbee se acerca, pero no llega a superar los 80 metros y los sensores fotoeléctricos no destacan precisamente por esta característica, pues requieren de distancias en torno a los 2-3 metros.
- **Cantidad**: En el caso de la tecnología BLE y Zigbee nos encontramos con el mismo resultado, pues requieren del mismo número de aparatos repartidos por el centro (en el primer caso de *beacons* y en el segundo de nodos). Sin embargo, el sistema formado por los leds fotoemisores requiere de uno individualizado por cada carro y eso conlleva un mayor número de receptores.
- **Interferencias**: Es una de las razones por las cuales la tecnología BLE se encuentra en mayor demanda los últimos años, pues gracias al sistema Bluetooth consigue evitar interferencias externas que puedan afectar al funcionamiento de la localización del carrito. Es por lo que la tecnología Zigbee, que utiliza balizas fijas y móviles mediante wifi se ve obsoleta frente a la anterior. Los fotoemisores sufren en mayor medida, sobre todo con el polvo, la suciedad y cualquier obstáculo que se interponga en la trayectoria de la emisión.
- **Tiempo de respuesta**: En este apartado, la diferencia no es tan grande con respecto al sistema BLE frente al Zigbee. De hecho, el valor difiere en microsegundos, no es muy significativo, pero sí que es mayor la diferencia con respecto al fotoemisor, pues es uno de los mayores inconvenientes en los que se encuentra este prototipo. Dependen mucho de la iluminación.
- **Temperatura**: El módulo BLE y el módulo Zigbee no sufren con respecto a los cambios de temperaturas que llegan a alcanzar ya que trabajan con comodidad, se encuentran en zonas de cierta estabilidad ambiental y su sistema está protegido frente al calor. No es el caso así de los fotoemisores, pues contienen tecnología que puede llegar a alcanzar niveles de temperatura poco óptimos para el sistema. Además, cualquier avería o fallo que pudiera producirse en la placa, evolucionaría en un muy probable aumento de la temperatura.

- **Precio**: La clave de la tecnología Zigbee son sus costes bajos frente al resto de tecnologías geolocalizadoras. El sistema BLE se acerca bastante a estos precios y es por ello por lo que su puntuación no difiere mucho de la mencionada anteriormente. Sin embargo, los fotoemisores requieren de un trabajo conjunto con un montante conformado por varios componentes, por lo que el precio se ve aumentado. Además, su cantidad, como bien se ha explicado anteriormente, es mayor.
- **Montaje**: En el caso de las dos primeras, al tener la misma cantidad y ser sistemas semejantes, su montaje también consiste en un método similar, por lo que sus diferencias son escasas. Los fotoemisores, al requerir ser montados en cada uno de los carros y en varios puntos del supermercado mayores que los otros, conlleva un mayor tiempo en su montaje.
- **Disponibilidad**: El sistema BLE está bastante demandado en la actualidad, por lo que son varias las empresas que se dedican a su fabricación. El sistema Zigbee, al encontrarse obsoleto, es más complicado encontrarlo en la actualidad. Los fotoemisores contienen otras piezas en fila para el acabado en su montaje, pero tienen facilidad de encuentro en cualquier tienda de electrónica.

### 4.6. SENSORES DE LECTURA DE LOS PRODUCTOS

En este último apartado, podemos descubrir una variedad de métodos con los que analizar el producto que entra dentro del carro y apuntarlos en la lista de compra del cliente. La información del producto se encuentra en la base de datos que contenga todo el listado y documentación, por lo que los sensores deben ser precisos y efectivos. Las opciones a estudiar van a ser las tres siguientes:

#### 4.6.1. LDRs (FOTORESISTENCIAS)

Se trata de un tipo de sensor óptico que consiste en una resistencia semiconductora dependiente de la iluminación. Tiene varias aplicaciones en la tecnología, como son las cámaras de fotos, la conexión del alumbrado público o cualquier detección en general. Sin embargo, presenta una variedad de desventajas que afectan en su elección final, como la dependencia de la temperatura, su tiempo de respuesta (que es bastante lento), la toxicidad de algunos materiales que afectan a su funcionamiento o su irregularidad (no es lineal). Además, otro inconveniente que puede ser muy importante es la obligación de montar en cada producto del supermercado dicho sensor.

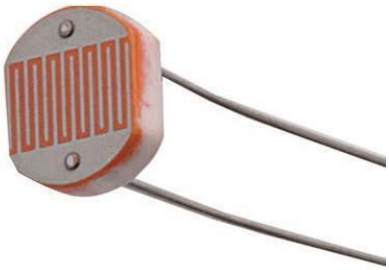


*Imagen 10: LDR*

### 4.6.2. LECTORES DE CÓDIGO DE BARRAS

Consisten en la codificación de caracteres alfanuméricos mediante el uso de códigos binarios a través de combinaciones como blanco/negro, brillo/mate o claro/oscuro. Se basan en un fotoemisor-fotorreceptor que luego amplifica la señal y se procesa por un sistema analógico. Pueden ser de Código EAN-L, distribución modular o bidimensionales. Los utilizados en los supermercados son de haz móvil, que consiste en una luz láser + un fotodiodo, y son de tipo omnidireccionales holográficos.

La mayor ventaja que encontramos en este sistema es que éstos son utilizados ya por los supermercados y, por lo tanto, ya se encuentran incorporados en todos los productos.

*Imagen 11: Lector de código de barras*

### 4.6.3. REALIDAD AUMENTADA (RA)

Esta tecnología permite al usuario visualizar los productos en el dispositivo tecnológico con la información gráfica incorporada sobre la información física ya existente. Esto provoca que las partes físicas tangibles y las virtuales se combinen, formando así una realidad aumentada en tiempo real. Es un proceso todavía en desarrollo, pues ya está disponible en algunos dispositivos, pero presenta fallos y tiene carencias existentes que producen errores en el sistema o en la lectura del producto. Sin embargo, es el sistema del futuro, sin lugar a dudas, debido a su comodidad en la práctica.

### 4.6.4. COMPARACIÓN Y ELECCIÓN

Los factores que se van a seguir para evaluar las tres opciones son los siguientes:

- **Precisión**: Ajuste completo sobre el producto a medir.
- **Distancia**: Valor máximo al que puede detectarse el producto.
- **Interferencias**: Factores internos o externos que afectan a las señales de los receptores, como por ejemplo: ruido, polvo, altas temperaturas, toxicidad de los materiales, etc.
- **Tiempo de respuesta**: Tiempo que tarda en recibir la señal el receptor o en enviarla el emisor.
- **Comodidad**: Incorporación del material del sensor en el producto.
- **Temperatura**: La cantidad de calor que llega a transmitir el prototipo.
- **Precio**: Coste del montante.
- **Montaje**: Implementación del lector en la carcasa de los carritos y sensores en los productos.
- **Disponibilidad**: Facilidad a la hora de encontrar el sistema en el mercado por un precio adecuado al comprador.

| FACTORES | VALOR RELATIVO (%) | MATERIALES | | |
|---|---|---|---|---|
| **TÉCNICOS** | **72%** | **LDR** | **L. C. DE BARRAS** | **RA** |
| **Precisión** | **20%** | 8 | 9 | 6 |
| **Distancia** | **15%** | 7 | 8 | 9 |
| **Interferencias** | **15%** | 5 | 8 | 7 |
| **T. de respuesta** | **12%** | 9 | 10 | 8 |
| **Comodidad** | **5%** | 5 | 10 | 10 |
| **Temperatura** | **5%** | 3 | 10 | 10 |
| **OTROS** | **28%** | | | |
| **Precio** | **15%** | 7 | 8 | 10 |
| **Montaje** | **8%** | 5 | 9 | 10 |
| **Disponibilidad** | **5%** | 9 | 10 | 7 |
| **RESULTADO** | **100%** | **6,78** | **8,82** | **8,21** |

*Tabla 4: Sensores de lectura*

Analizando los resultados finales obtenidos, comprobamos que los lectores de código de barras son la opción idónea a escoger, por lo menos a corto plazo. Supone una comodidad añadida que los supermercados contengan ya una base de datos con los productos del centro y sus respectivos códigos de barras incorporados. La nota obtenida en nuestra valoración es superior a las otras dos opciones por los siguientes motivos:

- **Precisión**: Los códigos de barras consisten en códigos binarios que almacenan caracteres alfanuméricos, por lo que la precisión es muy exacta. No puede haber errores, ya que los códigos binarios son diferentes para cada producto. Los fallos pueden darse a la hora de no leer bien el código, con lo que el resultado obtenido sería un simple error de lectura, no de cálculo. Las LDR dependen de la temperatura y toxicidad de los materiales, y sus resultados no son lineales, por lo que estos datos pueden afectar a su precisión. En el caso de la Realidad Aumentada, es un sistema que todavía se encuentra en fase de desarrollo y hay que mejorar en muchos aspectos, como es el caso de la precisión, pues las lecturas pueden proporcionar gran variedad de desaciertos en múltiples ocasiones.
- **Distancia**: En los tres casos nos encontramos con distancias muy cortas. Las LDR necesitan captar el sensor con una distancia menor a las otras, pues dependen de la temperatura, y para encontrar el cambio, requieren de mucha cercanía. Los lectores de código de barras tienen un rango de lectura, dependiendo del tipo que sean:
    - **CLV 412**: 35 a 95 mm
    - **CLV 410**: 50 a 400 mm

  En el caso de la Realidad Aumentada, el rango es mayor a los otros, pero depende mucho de la precisión del sistema, por lo que puede verse afectado a la hora de analizar el producto.

- **Interferencias**: Las LDR, como se ha comentado anteriormente, dependen mucho de la temperatura, por lo que cualquier cambio puede afectar en la medida, al igual que algunas toxicidades en los materiales (como suciedad o polvo). Estos dos últimos factores afectan también a los otros dos tipos, pero la Realidad aumentada sufre además con la luz, pues dependiendo de ésta puede afectar a la cámara que analiza el producto.

- **Tiempo de respuesta**: Los tres presentan un tiempo de respuesta bastante parejo. Sin embargo, el más rápido es el lector de códigos de barras. Las LDR sufren más con este aspecto, y la Realidad Aumentada tiene que analizar detalladamente el producto y buscar en la base de datos una imagen que sea equiparable a la analizada, por lo que se ve más perjudicada en este aspecto.
- **Comodidad**: En este caso, los códigos de barras ya vienen incorporados de por sí, por lo que no supone ningún cambio que se traduzca en un esfuerzo externo. La Realidad Aumentada no requiere de ninguna incorporación sobre los productos. La complejidad la encontramos con las LDR, pues se tendría que llegar a un acuerdo con los proveedores de los productos para que incorporaran el sensor en sus productos antes de enviarlos a los supermercados. Esto resulta un inconveniente exagerado para los proveedores.
- **Temperatura**: Los códigos de barras son códigos binarios impresos y la Realidad Aumentada trabaja con imágenes, por lo que este factor no afecta para nada en estos dos casos, al contrario que las LDR, que requieren de material que depende de la temperatura y consisten en fotorresistencias semiconductoras dependientes de la iluminación, motivo por el cual pueden sufrir sobrecalentamientos.
- **Precio**: Exige cierta inversión proporcionar componentes para todos los productos y para el sistema de análisis de los sensores en las LDR. Los lectores de código de barras pueden llegar a ser bastante económicos, pero aun así aumentan considerablemente el presupuesto final. La ventaja la encontramos con el ahorro que supone implementar los códigos de barras en los productos. En el caso de la Realidad Aumentada, simplemente se necesitaría la cámara de la Tablet para analizar el producto, por lo que no supone ningún incremento en el coste.
- **Montaje**: Aquí la dificultad la encontramos de nuevo con las LDR, ya que se deben colocar los sensores en cada producto, añadir en cada carro el sistema de análisis, etc. En los códigos de barras solamente es necesario implementar los lectores en los carros. Por último, la Realidad Aumentada estaría ya implementada en la cámara de la Tablet.
- **Disponibilidad**: En el mercado industrial encontramos una gran variedad de lectores de códigos de barras con muchas opciones para escoger y con precios competentes. Las LDR son fáciles de encontrar en tiendas de electrónica, pero cada vez se utilizan en menor medida. El sistema de Realidad Aumentada sigue en proceso de desarrollo, pero últimamente es más demandado por la industria y, como es el caso en servidores de la empresa Google, ya lo están implementando en sus plataformas para smartphones.

## 5. DESCRIPCIÓN DETALLADA DE LAS SOLUCIONES ADOPTADAS

Una vez escogidos los materiales idóneos para completar el proyecto, vamos a proceder a describir detalladamente los procesos de fabricación, diseño y montaje.

Como se ha descrito en apartados anteriores, el proyecto requiere de un prototipo acoplado en los carros del supermercado, que consiste en:

- Carcasa protectora de plástico técnico Arnite negro, **PET** (Polietileno Tereftalato).
- Tablet **PIPO X4** que contenga la aplicación móvil diseñada por la plataforma **Android Studio**.
- Lector de código de barras **DATALOGIC – GRYPHON$^{TM}$|GFS4400 2D**
- Módulo para cargar las baterías modelo **TP4056**.
- Cableado: 1 m de cable rojo para carga, 1 m de cable negro de fase.
- 2 planchas metálicas de 50 x 15 x 20 mm.

Por otro lado, tenemos el sistema de geolocalización. Consiste en la tecnología **BLE** (Bluetooth Low Energy), que trabaja mediante el uso de balizas electrónicas llamadas ***beacons***. Necesitaremos:

- Dos ***beacons*** por cada zona específica del supermercado: uno para localizar y otro para enviar notificaciones al **SMARTCART.**
- ***Beacons*** en todas las salidas del supermercado para controlar los carros en todo momento.
- Biblioteca para Android Studio sobre la tecnología ***beacon***.
- Modelo **Confidex Viking™ Classic.**

Por último, tenemos la base de carga de los carros, que consistirá (dependiendo de cada centro) en módulos divididos por tres parcelas con capacidad de abastecimiento de 10 carros por parcela. Contendrá los siguientes materiales:

- Estructura de acero inoxidable.
- Vías de aluminio fijadas al suelo.
- Fuentes de alimentación conmutadas (una por parcela), modelo **TRIAD MAGNETICS AWSP100-5**.
- Cableado por parcela: 10 m de cable rojo para carga; 10 m de cable negro de fase.
- Planchas metálicas por parcela: 20 planchas de 100 x 20 x 2 mm.

### 5.1. PLÁSTICO TÉCNICO PET PARA LA CARCASA

La carcasa del prototipo acoplado al carro requiere de un material protector resistente que cubra los dispositivos que vayan a necesitarse en los SMARTCARTS. Debe protegerlos de golpes, humedad o polvo. Del mismo modo, debe proteger al cliente de temperaturas altas, cortocircuitos o sobrecargas y posibles daños físicos. Con estas condiciones, se ha decidido instalar una carcasa hecha por un material plástico técnico llamado **Arnite**, de color negro, de propiedades químicas basadas en Polietileno Tereftalato.

Se trata de un poliéster termoplástico que posee una gran rigidez, favoreciendo así la tenacidad del material. Como mayor consecuencia a esta característica tenemos su menor resistencia a los impactos, pero ya que no se encontrará en zonas peligrosas ni de riesgo mayoritario, no supone un inconveniente extremo.

Dispone de una gran estabilidad dimensional que agiliza su transporte y, además, presenta unas condiciones aptas para el contacto con productos alimentarios. Protege adecuadamente del calor y de ácidos químicos. Sus propiedades de deslizamiento son correctas y no se desgasta con facilidad, por lo que perdura con el tiempo.

Otro factor importante, es su condición frente al reciclaje. Pierde pocas propiedades frente a reutilizaciones y su disponibilidad en el mercado es absoluta. En cuanto al precio, los plásticos técnicos se encuentran en desventaja frente a otros materiales como el aluminio. Aun así, su coste es bastante competitivo frente a otros plásticos.

| | PROPIEDADES |
|---|---|
| **PET** | Compacto y rígido |
| | Duro |
| | Flexible |
| | Resistencia al desgaste |
| | Resistencia a ácidos diluidos, productos de limpieza y disolventes |
| | Buen aislante eléctrico |
| | Mecanización ágil |
| | Agua superior a 65 ºC no resiste durante período largo de tiempo |
| | Material soldable |
| | Adhesividad aceptable |

*Tabla 5: Propiedades del PET*

Dentro de la carcasa se tiene que acoplar: la pantalla que va a albergar la aplicación (Tablet PIPO X4), un lector de código de barras que analizará los productos que entrarán en la cesta del carro (DATALOGIC – GRYPHON™|GFS4400 2D), el módulo TP4056 para cargar (o en este caso mantener y proteger) la batería de la Tablet, el cableado para los materiales y dos planchas metálicas en los laterales con las que alimentaremos al módulo y por consiguiente a la Tablet y al lector.

Sobre estas planchas, en la parte interior, se soldarán los cables correspondientes: en la plancha izquierda el cable positivo (rojo) y en la derecha el cable negativo (negro).

**Amperaje que soportan los cables de cobre**

| Nivel de temperatura: | 60°C | 75°C | 90°C | 60°C | |
|---|---|---|---|---|---|
| Tipo de aislante: | TW | RHW, THW, THWN | THHN, XHHW-2, THWN-2 | SPT | |
| Medida / calibre del cable | Amperaje soportado | | | Medida / calibre del cable | Amperaje soportado |
| 14 AWG | 15 A | 15 A | 15 A | 20 AWG | 2 A |
| 12 AWG | 20 A | 20 A | 20 A | | |
| 10 AWG | 30 A | 30 A | 30 A | 18 AWG | 10 A |
| 8 AWG | 40 A | 50 A | 55 A | | |
| 6 AWG | 55 A | 65 A | 75 A | 16 AWG | 13 A |
| 4 AWG | 70 A | 85 A | 95 A | | |
| 3 AWG | 85 A | 100 A | 115 A | 14 AWG | 18 A |
| 2 AWG | 95 A | 115 A | 130 A | | |
| 1 AWG | 110 A | 130 A | 145 A | 12 AWG | 25 A |
| 1/0 AWG | 125 A | 150 A | 170 A | | |
| 2/0 AWG | 145 A | 175 A | 195 A | | |
| 3/0 AWG | 165 A | 200 A | 225 A | | |
| 4/0 AWG | 195 A | 230 A | 260 A | | |

*Imagen 12: Tabla sobre el amperaje que soportan los cables de cobre*

Como se puede comprobar en la imagen anterior, la medida de cableado escogida es la marcada en rojo. El nivel de temperatura al que se va a trabajar no superará los 60 grados. El tipo de material es **TW** (T-*Thermoplastic* y W-*Water resistant*). El calibre del cable es de 12 **AWG** (*American Wire Gauge*) y el amperaje que es capaz de soportar es de 20 Amperios.

Los cables, como se ha explicado anteriormente, se soldarán a las planchas metálicas con soldador y estaño.

*Imagen 13: Soldador de estaño para cables de cobre*

Una vez aclarado cuál va a ser el material de la carcasa y estudiadas sus propiedades, se diseña en el programa **SOLIDWORKS** la forma, medidas y piezas de ésta. Los planos resultantes se encuentran en el documento de planos del proyecto.

Cuando los planos del diseño están listos, se envían al fabricante para cotejar los datos y se establecen las bases del pedido que se vaya a encargar por parte del supermercado.

El resultado final de la carcasa se enviará a las instalaciones para realizar el montaje de los SMARTCARTS y así comprobar que todo está correcto y que se superen las pruebas de servicio estipuladas en el pliego de condiciones.

### 5.2. TABLET PIPO X4

Esta Tablet es la escogida para abastecer el sistema software que se va a diseñar. La aplicación estará instalada en todas las Tablets colocadas en los **SMARTCARTS.** La Tablet irá incorporada a la carcasa por unas sujeciones que se señalarán en el documento de los planos, así como la distancia a la que se encontrará de los otros dispositivos y la posición. Como se ha comentado anteriormente, el modelo escogido es el del título y sus especificaciones técnicas son las siguientes:

- Sistema Windows 10, con el que Android permite trabajar.
- UPC – Intel Cherry Trail Z8350, que proporciona una velocidad de 1.44 – 1.92 GHz.
- GPU – Intel HD Graphics Gen8, que alcanza hasta 500 MHz.
- Pantalla Capacitiva de 10.1 pulgadas y una resolución de 1920 x 1200.
- Material IPS para pantallas capacitivas, antilluvia.
- Contiene 4 Gb de memoria RAM (con lo que mantiene el sistema funcionando de manera correcta) y 64 Gb de ROM (que permite trabajar con todas sus aplicaciones y realizar diversas funciones).
- Contiene una batería de 3.7 V / 12000 mAh.
- Su peso es de 1080 gramos y sus medidas son: 280 x 185 x 26.5 milímetros.
- Trabaja con tecnología Bluetooth 4.0, para el sistema de geolocalización.
- Conectividad de 4 puertos (micro USB/USB 2.0/USB 3.0/Micro HD) para poder conectar el lector de código de barras.
- Un conector Jack de 3.5 mm DC para cargar.

*Imagen 14: Parte frontal de la Tablet PIPO X4*

Al trabajar con Android como sistema operativo, vamos a poder instalar nuestra aplicación diseñada con el programa Android Studio, el cual explicaremos más adelante.

### 5.3. MÓDULO TP4056 PARA CARGAR BATERÍAS

Se trata de un circuito integrado que permite conectar baterías a su salida (a la Tablet y al sensor). A su entrada se conecta una fuente de energía eléctrica, en este caso, se tratará de una fuente de alimentación conmutada.

Los SMARTCARTS no siempre van a estar en la base donde se encontrará la fuente de energía. La batería de la Tablet está capacitada para aguantar durante más de 24 horas. Además, tiene que proporcionar energía al lector de código de barras. Sin embargo, este dispositivo (TP4056) tiene la capacidad de:

- Estabilizar la batería de la Tablet
- Proteger los otros componentes de sobrecargas o sobretensiones
- Mantener la temperatura

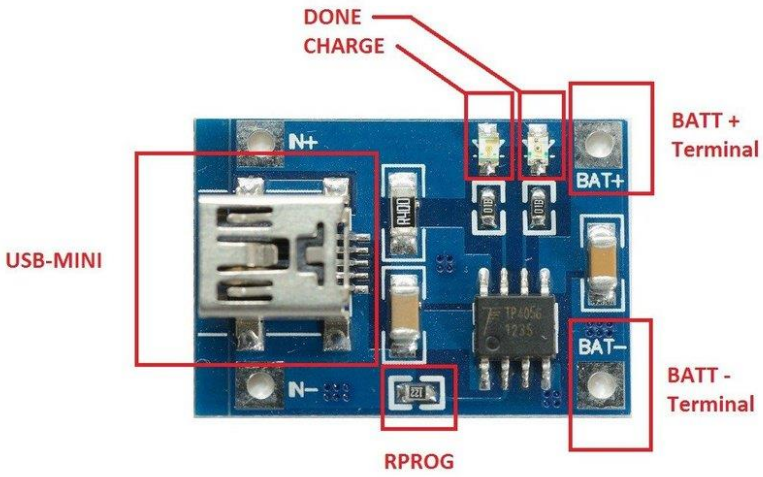



*Imagen 15: Componentes del módulo TP4056*

Del esquema presentado en la imagen de arriba, hay que analizar lo siguiente:

- Puerto **miniUSB**: Es uno de los conectores por los que se puede alimentar el módulo.
- Bornes **IN+/IN-**: Son los bornes que se encuentran a los lados del puerto miniUSB. Son la alternativa al puerto para cargar el módulo.
- **LEDs** de **Carga** y **Completado**: Su función es simple:
    - **Carga**: El LED correspondiente estará encendido en caso de estar cargando baterías.
    - **Completado**: Se encenderá su LED correspondiente cuando el proceso de carga de batería se haya completado.
- **BAT+/BAT**-: Son los bornes de salida para cargar las baterías externas.

En la siguiente imagen tenemos un ejemplo de cómo se debería hacer la conexión en caso de querer cargar una batería externa:

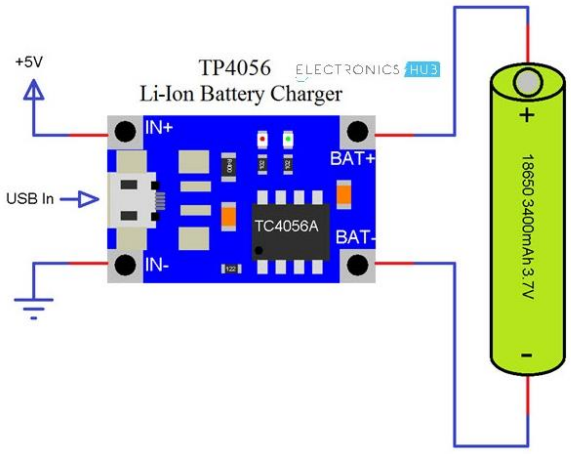



*Imagen 16: Esquema de uso correcto del módulo TP4056*

## 5.4. PROGRAMA ANDROID STUDIO

Todos los dispositivos que funcionan con el sistema operativo Android, permiten la instalación de aplicaciones diseñadas a través del programa Android Studio. Android se basa en un núcleo de sistema operativo libre, **Linux**, que además es gratuito y multiplataforma.

El programa Android Studio es una herramienta de trabajo que utilizan los programadores dentro de la plataforma Android. Se trata de un entorno de desarrollo integrado de código libre que permite diseñar aplicaciones móviles. Las aplicaciones se pueden escribir con tres tipos de lenguaje diferentes: **Kotlin** (que es el lenguaje oficial), **Java** o **C++**. Para el desarrollo de esta aplicación se ha escogido Java, ya que es el tipo de lenguaje más sencillo y práctico con el que se puede trabajar. También es el más utilizado.

Una de las múltiples ventajas que posee este programa es que permite traducir códigos escritos en Kotlin, Java o C++ a cualquiera de estos tres.

*Imagen 17: Logotipo del programa Android Studio*

El diseño de las aplicaciones se divide en dos partes:

- **Parte gráfica**: Es la parte en la que se diseñan las interfaces, donde encontramos un panel gráfico y por otro lado su código XML correspondiente.

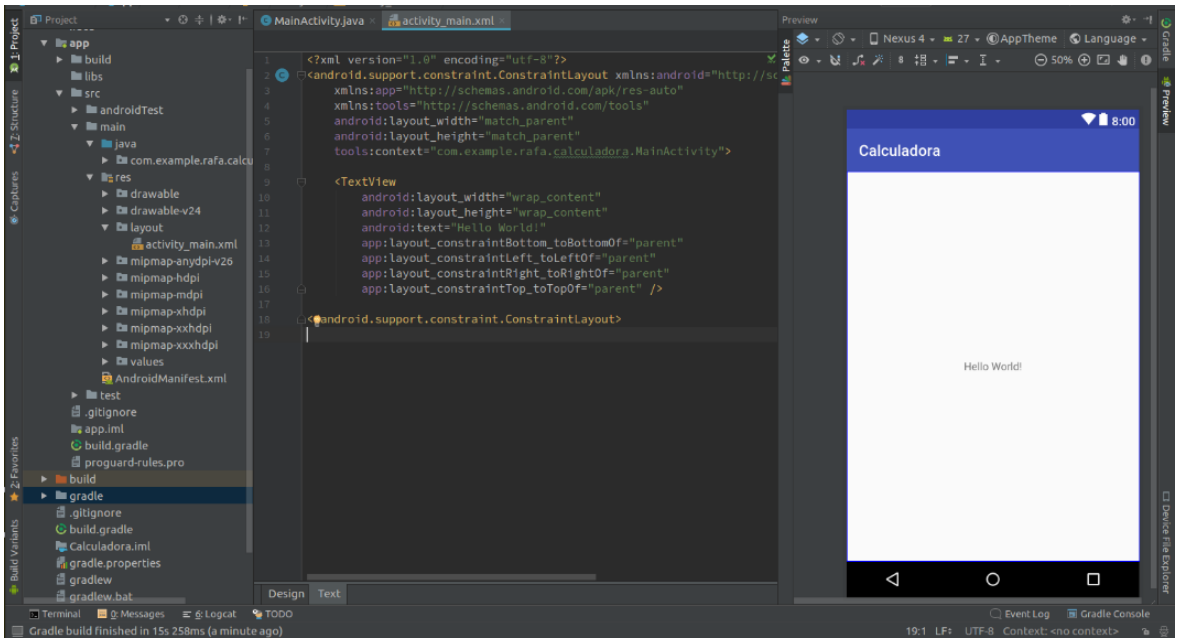



*Imagen 18: Ejemplo de la parte gráfica de una App en Android Studio*

- **Parte lógica**: Es la parte donde se escribe el código en Java que trabajará sobre las interfaces de la parte gráfica.

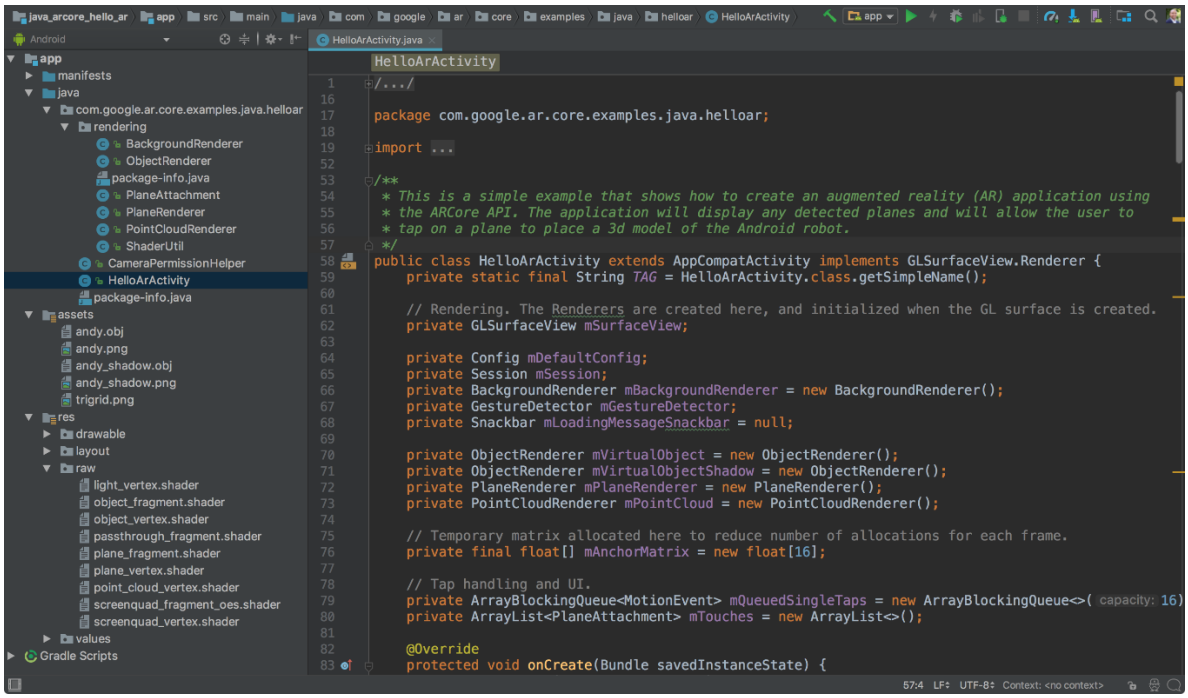



*Imagen 19: Ejemplo de la parte lógica de una App en Android Studio*

Dentro de la parte gráfica existen diferentes tipos, formas y diseños que podemos insertar dentro de las interfaces. Estos elementos son los siguientes:

- **Bloques de texto**: Dentro de este bloque, encontramos grandes variedades como textos numéricos, textos escritos, textos planos que permiten añadir **URLs**, contraseñas, etc.
- **Botones**: En este caso, disponemos de botones, botones de imagen, **CheckBox**, etc.
- **Widgets**: Estos sirven para poder ver o hacer imágenes y videos, crear listas, completar bases de datos a través de la plataforma **SQLite**, etc.
- **Layouts**: Con este método podemos organizar la interfaz ordenando los elementos de diversas maneras, en horizontal, vertical, unos encima de otros, etc.
- **Contenedores**: Sirven para colocar controles y elementos visuales que superan el campo de visión de la interfaz. Utiliza por ejemplo listas desplegables, **ScrollView**, etc.

En lo que respecta a la aplicación que vamos a diseñar para el SMARTCART, contendrá dos servidores distintos:

- El primero permitirá la inscripción de los trabajadores dentro de la aplicación para poder realizar una serie de tareas en las que completará las siguientes acciones:
    - Escoger un cajero
    - Añadir productos
    - Eliminar productos
    - Modificar productos

- El segundo tendrá las interfaces que utilizarán los clientes del supermercado. Las acciones que podrán completar son las siguientes:
    - Identificar la localización en el interior del supermercado
    - Inspeccionar el listado de productos por zonas
    - Realizar compra
    - Visualizar el listado de la compra + el cálculo del precio + el peso de la compra
    - Finalizar (indica el cajero al que acudir)

Además, los clientes recibirán notificaciones de ofertas cada vez que pasen por uno de los ***beacons*** destinados a esta misión.

### 5.5. BASE DE DATOS EN GOOGLE SHEETS + GOOGLE APPS SCRIPTS

Las bases de datos que se van a necesitar para completar la aplicación del SMARTCART van a realizarse a través de **GOOGLE SHEETS**.

- Para poder utilizar esta plataforma tenemos que crear un directorio dentro del Google Drive y en éste, vamos a completar una hoja de cálculo donde diseñaremos las distintas tablas que vamos a necesitar.
- Una vez estemos dentro, publicaremos estas hojas como páginas web. Esto se hace con el siguiente comando:

**Archivo / Publicar en la web**

- En cuanto se haya completado este proceso, generaremos una dirección URL para las hojas de cálculo. Utilizaremos estas direcciones URL más tarde en el diseño de la parte lógica de la aplicación para así poder interactuar con las tablas. Podremos recoger datos o modificarlos.

Las bases de datos se pueden diseñar en varios modelos. En este caso, las que vamos a diseñar en este proyecto son de tipo relacional, que consisten en unas tablas que se componen de registros (distribuidos por filas) y una serie de campos (orientados por columnas) que almacenan los registros anteriormente mencionados. Como las tablas no van a tener relación alguna entre sí, ninguno de los campos va a ser común entre ellas.

Las tablas que se van a necesitar para este proyecto son las siguientes:

- **Tabla empleados**: Esta tabla contiene el registro de los empleados que trabajan en el supermercado. La tabla dispondrá de las columnas:
    - **Nombre**
    - **Contraseña**

| ID | USUARIO | CONTRASEÑA |
|---|---|---|
| 1 | roberto | rObErtO |
| 2 | susana | 19susana95 |
| 3 | raquel | raquelita27 |
| 4 | alfonso | trombon333 |
| 5 | Laura | MiYoMayor |

*Imagen 20: Hoja de datos de los empleados*

Las tablas que se añaden a continuación, dependerán del supermercado al que le estemos diseñando la aplicación. Cada supermercado se divide en zonas, en las que se distribuyen los diferentes productos que venden. Por ejemplo, la sección de frutas y verduras, la sección de congelados, la sección de charcutería, etc. Además, cada supermercado deseará incorporar elementos adicionales de información que deberán ser establecidos en el pedido sobre el diseño de la aplicación.

Para este proyecto teórico, vamos a establecer cinco secciones diferentes y por cada producto, los datos: **nombre**, **fecha de caducidad**, **peso** y **precio**. Cada una de las secciones contendrá una tabla con los datos establecidos por producto que se aplicará en la hoja de cálculo:

- **Tabla frutas y verduras**: En la tabla aparecerá el nombre del producto, la fecha de caducidad, el peso y el precio.

| ID | NOMBRE | FECHA CAD. | PESO | PRECIO |
|---|---|---|---|---|
| 1 | Melón | 28-08-2020 | 4.110 | 3,5 |
| 2 | Sandía | 30-08-2020 | 6780 | 3,62 |
| 3 | Berenjena | 29-08-2020 | 304 | 0,37 |

*Imagen 21: Hoja de datos de fruta y verdura*

- **Tabla congelados**: En la tabla aparecerá el nombre del producto, la fecha de caducidad, el peso y el precio.

| ID | NOMBRE | FECHA CAD. | PESO | PRECIO |
|---|---|---|---|---|
| 1 | Pizza 4 quesos | 30-10-2021 | 386 | 2,29 |
| 2 | Judías verdes | 07-11-2021 | 750 | 2,81 |
| 3 | Polos limón | 14-03-2022 | 1330 | 1,75 |

*Imagen 22: Hoja de datos de congelados*

- **Tabla charcutería**: En la tabla aparecerá el nombre del producto, la fecha de caducidad, el peso y el precio.

| ID | NOMBRE | FECHA CAD. | PESO | PRECIO |
|---|---|---|---|---|
| 1 | Solomillo | 22-09-2020 | 500 | 3,69 |
| 2 | Pechugas pavo | 25-10-2020 | 330 | 2,95 |
| 3 | Jamón ibérico | 30-10-2020 | 100 | 2,98 |

*Imagen 23: Hoja de datos de charcutería*

- **Tabla licores**: En la tabla aparecerá el nombre del producto, la fecha de caducidad, el peso y el precio.

| ID | NOMBRE | FECHA CAD. | PESO | PRECIO |
|---|---|---|---|---|
| 1 | Vino Tinto | - | 750 | 4,5 |
| 2 | Vino Blanco | - | 750 | 4,12 |
| 3 | Sidra | - | 750 | 0,89 |

*Imagen 24: Hoja de datos de licores*

- **Tabla droguería**: En la tabla aparecerá el nombre del producto, la fecha de caducidad, el peso y el precio.

| ID | NOMBRE | FECHA CAD. | PESO | PRECIO |
|---|---|---|---|---|
| 1 | Lejía | - | 1.000 | 2,81 |
| 2 | Lote bayetas | - | 200 | 1,27 |
| 3 | Estropajo verde | - | 50 | 0,9 |

*Imagen 25: Hoja de datos de droguería*

## 5.6. LECTOR DE CÓDIGO DE BARRAS DATALOGIC – GRYPHON™ | GFS4400 2D

El sensor que vamos a utilizar para analizar los productos que se introduzcan al carro y, por consiguiente, se añadan a la lista de la compra del cliente, es el mencionado en el título. Este sensor se conectará mediante el cable USB que lleva incorporado a uno de los puertos de la Tablet, en este caso el puerto USB 3.0. Además, irá acoplado a la carcasa por unas sujeciones que se señalarán en el documento de los planos, así como la distancia a la que se encontrará de los otros dispositivos y la posición.

El funcionamiento de este sistema es el siguiente:

- **Paso 1**: El sensor leerá el código del producto.
- **Paso 2**: A través de la conexión con la Tablet por el puerto USB 3.0, la aplicación recibirá la información.
- **Paso 3**: La información será enviada a la base de datos donde se cotejará el producto que responderá al código leído.
- **Paso 4**: Cuando la base de datos haya encontrado el producto referido, enviará a la aplicación la información y ésta la añadirá a la lista de la compra.

Los datos más relevantes del lector son los siguientes:

- Contiene un marcador de 4 puntos de visibilidad alta con cruz central que sirve para un escaneo dirigido.
- La lectura es omnidireccional, permite leer todo tipo de código de barras en cualquier orientación.
- Tiene una óptica de tolerancia de movimiento muy avanzada.
- Capacidad para capturar imágenes y escanear documentos.
- Es capaz de leer códigos de barras de hasta 4 milésimas de pulgada.
- Lee códigos postales, además de códigos 1D, 2D, apilados y compuestos.
- Las conexiones pueden ser de dos tipos: RS-232 o por USB.
- Contiene un zumbador integrado de buena lectura.
- Para activarse tiene 2 opciones: Vía manual o por detección automática.
- Su clasificación de sellado de agua y partículas es IP54

*Imagen 26: Lector de código de barras DATALOGIC – GRYPHON™ | GFS4400 2D*

## 5.7. TECNOLOGÍA BLE PARA EL SISTEMA DE GEOLOCALIZACIÓN

El sistema **bluetooth 4.0** emite señales de 2.4 GHz con un alcance que llega hasta los 100 metros de distancia. Su gran ventaja es el bajo consumo de energía, por lo que permite emitir señales durante mucho tiempo (varios años) sin necesidad de recargar las baterías de los ***beacons*** (balizas electrónicas).


*Imagen 27: Icono del sistema Bluetooth 4.0*

Al tratarse de un sistema Bluetooth, la tecnología es inalámbrica y esto posibilita conectar varias balizas electrónicas por el centro y transmitir información entre ellas y los prototipos SMARTCARTS.

Otra de sus mejoras frente a otras tecnologías geolocalizadoras es su velocidad de transferencia, que es capaz de alcanzar los 1Mbps (Mega Bit por Segundo). Otras características importantes son:

- **Modulación**: GFSK de 0.45 a 0.55.
- **Nº de canales**: 40.
- **Separación**: 2 MHz.
- **Tipología de red**: Scatternet.
- **Consumo de pico de corriente**: <15 mA.
- **Corriente en Standby**: <2 µA.
- **Grado de seguridad de las transmisiones**: cifrado AES 128 con CCM.

Para poder utilizar esta tecnología se necesita del uso de unos dispositivos que, como hemos apuntado en apartados anteriores, se llaman ***beacons***, o también balizas electrónicas. Estos aparatos son los que transmiten a las Tablets de los SMARTCARTS las señales BLE con la información necesaria. Para este proyecto, se necesitan dos ***beacons*** por sección de supermercado, y cada uno tendrá una función:

- Localizar el carro en un mapa del centro
- Enviar diferentes ofertas que ofrezca el centro a los clientes

Las balizas que se van a instalar en el centro para este proyecto son el modelo **Confidex Viking™ Classic**. Son balizas de grado industrial que trabajan con el sistema **Eddystone** (Es el formato que utiliza Android para programar los *beacons* en sus aplicaciones). Están diseñadas para soportar entornos severos y sus características son las siguientes:

- **Memoria**: Flash de 512 kB y RAM de 64 kB.
- **Duración**: hasta los 10 años.
- **Dimensiones**: 55 x 54 x 19 mm.
- **Rango de lectura**: Hasta 200 m.
- **Temperatura**: -20 a 60 ˚C.
- **Clasificación del IP**: IP68


*Imagen 28: Beacon Confidex Viking™ Classic*

Como se ha expuesto anteriormente, la tecnología con la que se va a trabajar sobre los *beacons* es **Eddystone™**. Se trata de un sistema de Google con código abierto con el que trabaja Android desde hace mucho tiempo. Los formatos que permite transmitir son los siguientes:

- **UID**: Identificador de Usuario.
- **URL**: Localizador de Recursos Uniforme (para páginas web).
- **TLM**: Telemetría, permite activar diferentes acciones.
- **EID**: Identificadores efímeros, cambian periódicamente.

La plataforma Android Studio posee una biblioteca específica para trabajar con los *beacons* y así poder acoplarlos al diseño de la aplicación:

Enlace: https://github.com/AltBeacon/android-beacon-library

En el anexo del código para la aplicación se mostrará el código escrito para aplicar los *beacons* al diseño de la aplicación.

## 5.8. FUENTE DE ALIMENTACIÓN CONMUTADA TRIAD MAGNETICS AWSP100-5

El modelo AWSP100-5 es el escogido para abastecer la energía eléctrica que se proporcionará a los SMARTCARTS. Se colocará en las bases de carga donde se almacenarán los prototipos.

Esta fuente de alimentación está diseñada para una amplia variedad de aplicaciones donde se desea tener una alta confiabilidad. Se utiliza para recursos industriales, al igual que para mercados de telecomunicaciones. Su diseño le caracteriza por poseer un gran rendimiento.

*Imagen 29: Fuente de alimentación AWSP100-5*

Algunas de sus características más relevantes son las siguientes:

| ESPECIFICACIONES TÉCNICAS | DATOS | UNIDADES |
|---|---|---|
| **Altura** | 42 | mm |
| **Ancho** | 98 | mm |
| **Largo** | 198 | mm |
| **PCB cambiado** | 7 | - |
| **Tipo de componente** | Transpuesta | - |
| **Estilo del dispositivo** | Adjunto | - |
| **Tipo de Entrada/Salida** | AC/DC | - |
| **Número de salidas** | 1 | - |
| **Tipo de salidas** | Fijo | - |
| **Voltaje de salida** | 5 | V |
| **Potencia de salida** | 100 | W |
| **Voltaje min. de entrada de CA** | 88 | V |
| **Voltaje máx. de entrada de CA** | 264 | V |
| **Voltaje min. de aislamiento** | 3000 | VCA |
| **Regulación de línea** | 0.5 | % |
| **Regulación de carga** | 1 | % |
| **Corriente de entrada máx.** | 1.8 | A |
| **Frecuencia de entrada** | 47-63 | Hz |
| **Eficiencia** | 75 | % |
| **Tipo de conector de entrada** | Bloque de terminales tornillo | - |
| **Tipo de conector de salida** | Bloque de terminales tornillo | - |
| **Ondulación y ruido** | 100 | mV |
| **Temp. min. de funcionamiento** | -10 | ˚C |
| **Temp. máx. de funcionamiento** | 60 | ˚C |
| **Peso** | 560 | g |
| **Número de pines** | 7 | - |

*Tabla 6: Especificaciones técnicas de la fuente de alimentación AWSP100-5*

## 5.9. BASE DE CARGA PARA LOS SMARTCARTS

Las bases de carga, como se informa al principio de este apartado, son unas estructuras diseñadas por nosotros y presentan las siguientes características:

- Estructura de acero inoxidable.
- Vías de aluminio para los carros en el suelo.
- División en 3 parcelas (dependiendo del supermercado).
- En cada parcela hay una fuente de alimentación AWSP100-5.
- 10 carros por parcela.

Los motivos por los que decidimos colocar 10 carros por parcela son los siguientes:

- La fuente de alimentación AWSP100-5 posee unas características de salida de 5 V de tensión y 100 W de potencia. Esto significa que suministra 20 A a la salida según la siguiente operación:

$$Corriente\ de\ salida\ (A) = \frac{Potencia\ de\ salida\ (W)}{Tensión\ de\ salida\ (V)} = \frac{100}{5} = 20\ A$$

- La batería de la Tablet es de 3,7 V y 12000 mAh. Esto supone un consumo de 44,4 Wh de potencia según la operación:

$$Potencia\ (Wh) = Voltaje\ (V) \cdot Intensidad\ (Ah) = 3{,}7 \cdot 12 = 44{,}4\ Wh$$

- El cargador de la Tablet es de 5 V/3 A.

Con 3 A de carga, la Tablet es capaz de cargarse incluyendo la pantalla encendida. Sin embargo, con la pantalla apagada, la carga se reduce a 1.5 A. De modo que la fuente de alimentación sería capaz de abastecer en torno a 10 SMARTCARTS según la siguiente operación:

$$\frac{Corriente\ de\ salida\ de\ fuente\ de\ alimentación\ (A)}{Corriente\ de\ carga\ de\ Tablet\ con\ pantalla\ apagada\ (A)} = \frac{20}{1.5} = 13{,}33 \cong 10\ SMARTCARTS$$

El motivo por el cual se redondea hacia abajo, es simple: Cuantos menos carros tenga que cargar la fuente de alimentación, más corriente podrá proporcionar a cada uno, lo que se traduce en mayor velocidad de carga.

Una vez aclarada la cantidad de carros que habrá por parcela, se debe establecer la distancia a la que se colocarán las planchas metálicas de carga. Estas planchas metálicas de 100 x 20 x 2 mm tendrán un sistema con el cual se producirá la carga de los carros. Este sistema es el siguiente:

- La plancha estará sujeta por dos muelles. Estos muelles amortiguarán el movimiento de la plancha, que solo podrá ser hacia dentro o hacia fuera.
- Los laterales de la plancha se meterán por dos huecos y solo podrán moverse hacia dentro o hacia fuera (para evitar desplazamientos de las planchas o roturas).
- Los cantos de la plancha por la parte de fuera serán redondos para deslizar mejor con las planchas que irán incorporadas en los carros.
- Cuando la plancha sea empujada hacia dentro, hará contacto con una pieza metálica.
- La pieza metálica al ser empujada hacia dentro, hará contacto con el pin del cable que le corresponda.
- El cable recibe la corriente de la fuente de alimentación.

Los cables que salgan de la fuente de alimentación para cargar los carros, se distribuirán de la siguiente manera:

- En paralelo.
- Los cables positivos (rojo) irán por el módulo de la izquierda.
- Los cables negativos (negro) irán por el módulo de la derecha.

| Amperaje que soportan los cables de cobre | | | | | |
|---|---|---|---|---|---|
| Nivel de temperatura: | 60°C | 75°C | 90°C | 60°C | |
| Tipo de aislante: | TW | RHW, THW, THWN | THHN, XHHW-2, THWN-2 | SPT | |
| Medida / calibre del cable | Amperaje soportado | | | Medida / calibre del cable | Amperaje soportado |
| 14 AWG | 15 A | 15 A | 15 A | 20 AWG | 2 A |
| 12 AWG | 20 A | 20 A | 20 A | | |
| 10 AWG | 30 A | 30 A | 30 A | 18 AWG | 10 A |
| 8 AWG | 40 A | 50 A | 55 A | | |
| 6 AWG | 55 A | 65 A | 75 A | 16 AWG | 13 A |
| 4 AWG | 70 A | 85 A | 95 A | | |
| 3 AWG | 85 A | 100 A | 115 A | 14 AWG | 18 A |
| 2 AWG | 95 A | 115 A | 130 A | | |
| 1 AWG | 110 A | 130 A | 145 A | 12 AWG | 25 A |
| 1/0 AWG | 125 A | 150 A | 170 A | | |
| 2/0 AWG | 145 A | 175 A | 195 A | | |
| 3/0 AWG | 165 A | 200 A | 225 A | | |
| 4/0 AWG | 195 A | 230 A | 260 A | | |

*Imagen 30: Tabla sobre el amperaje que soportan los cables de cobre*

Al igual que los cables que irán por dentro de la carcasa del carro, el cableado de la base de los carros será del mismo modelo: 60 grados de nivel de temperatura, tipo de cable TW, calibre 12 AWG y soporte de 20 Amperios.

Los pines que se colocarán son los del modelo ETX-5004x0001 y para colocarlos se necesita un pelacables y una Crimpadora industrial.

Los planos del diseño de la base de carga se encuentran en el documento de planos de este proyecto. En ellos podremos encontrar toda la información referente a los materiales, distancias y peso.

## 6. RESULTADOS

Completados todos los puntos del proyecto, es hora de realizar los montajes y las comprobaciones pertinentes establecidas en el Pliego.

En primer lugar, tenemos el diseño de la aplicación con la plataforma Android Studio. Los resultados se comprueban en la Tablet y son los siguientes:

- El icono de la aplicación debe ser visible dentro de la interfaz de la Tablet, con el título incorporado.

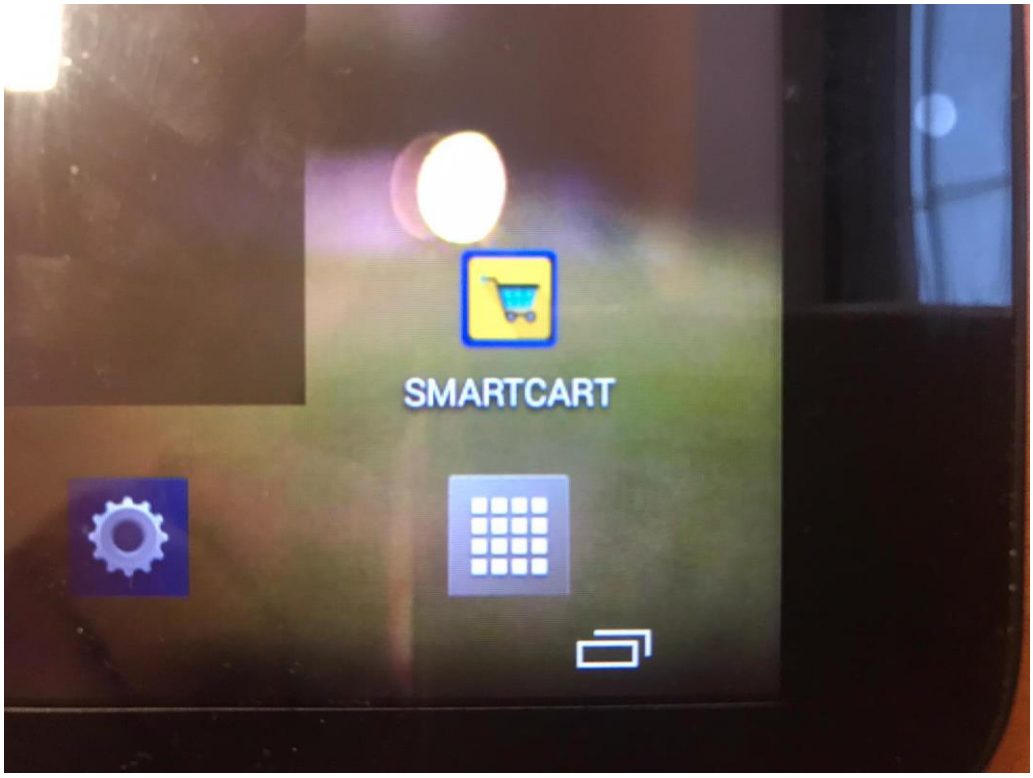



*Imagen 31: Icono de la aplicación SMARTCART*

- Al pulsar el icono, la primera interfaz que debe aparecer tiene que darnos a escoger entre los dos servidores disponibles: EMPLEADO y CLIENTE.

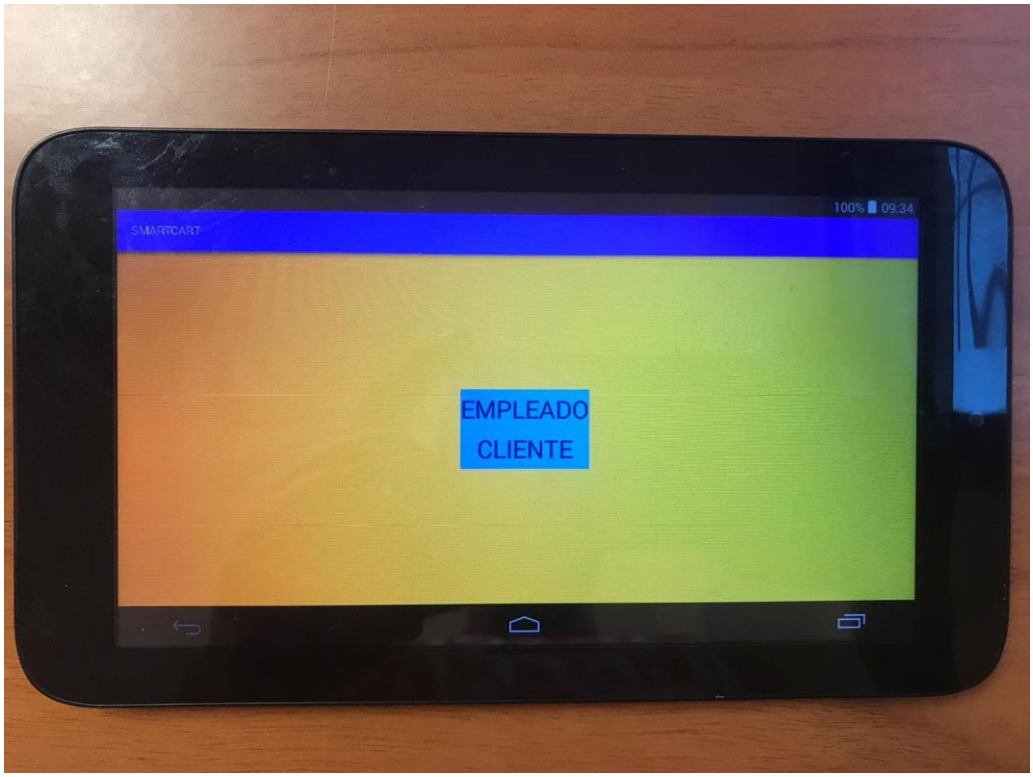



*Imagen 32: Interfaz de Inicio*

- El empleado activará la interfaz **EMPLEADO,** donde le pedirán que se registre para poder entrar. El usuario y la contraseña estarán almacenados en la base de datos:

| ID | USUARIO | CONTRASEÑA |
|---|---|---|
| 1 | roberto | rObErtO |
| 2 | susana | 19susana95 |
| 3 | raquel | raquelita27 |
| 4 | alfonso | trombon333 |
| 5 | Laura | MiYoMayor |

*Imagen 33: Hoja de datos de los empleados*

Si los datos **usuario** y **contraseña** son introducidos correctamente, se habilitará la interfaz EMPLEADO 2 donde le dará a escoger entre los cajeros del supermercado.

- Al escoger uno de los cajeros, se abre una interfaz que nos pedirá el número del carro que se encuentre en la cola para añadir en los visores de texto los datos del **PESO** y el **PRECIO**.

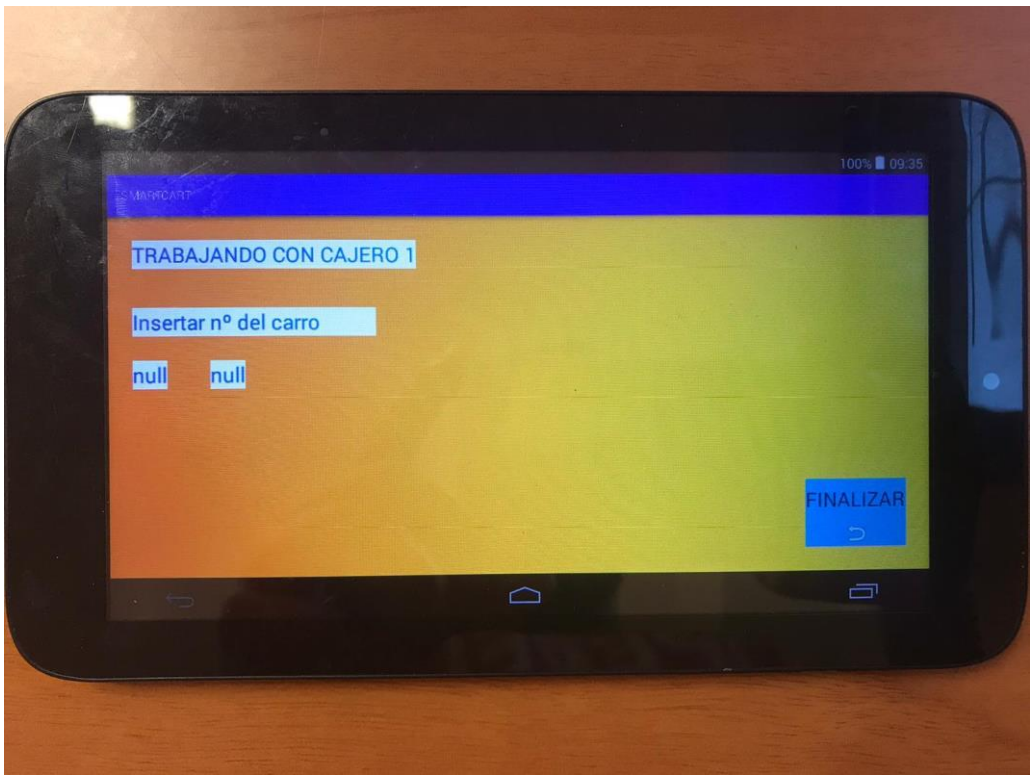



*Imagen 34: Interfaz CAJERO 1*

- En el caso de escoger el servidor CLIENTE, la interfaz que aparece ofrecerá la información necesaria del carro:
    - Nº del carro.
    - Posición actual.
    - Sección actual.
    - Lista de productos.
    - Lista de la compra.
    - Peso y Precio de la compra.

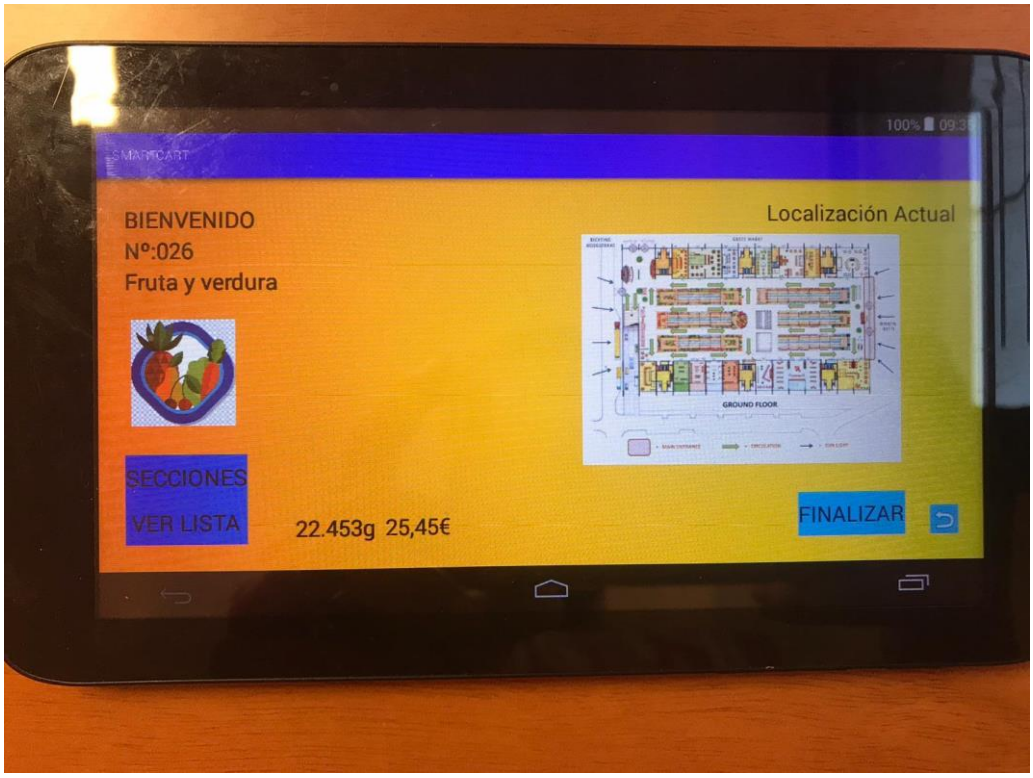



*Imagen 35: Interfaz CLIENTE*

- Una vez finalizada la lista de compra, pulsamos **Finalizar** y muestra una interfaz con el nombre del cajero al que se debe acudir. En el cajero, los responsables apuntan el número correspondiente del carro y en la interfaz del empleado aparecerán los datos **Peso** y **Precio**. Si todo coincide, se realiza el pago y se termina con el proceso.

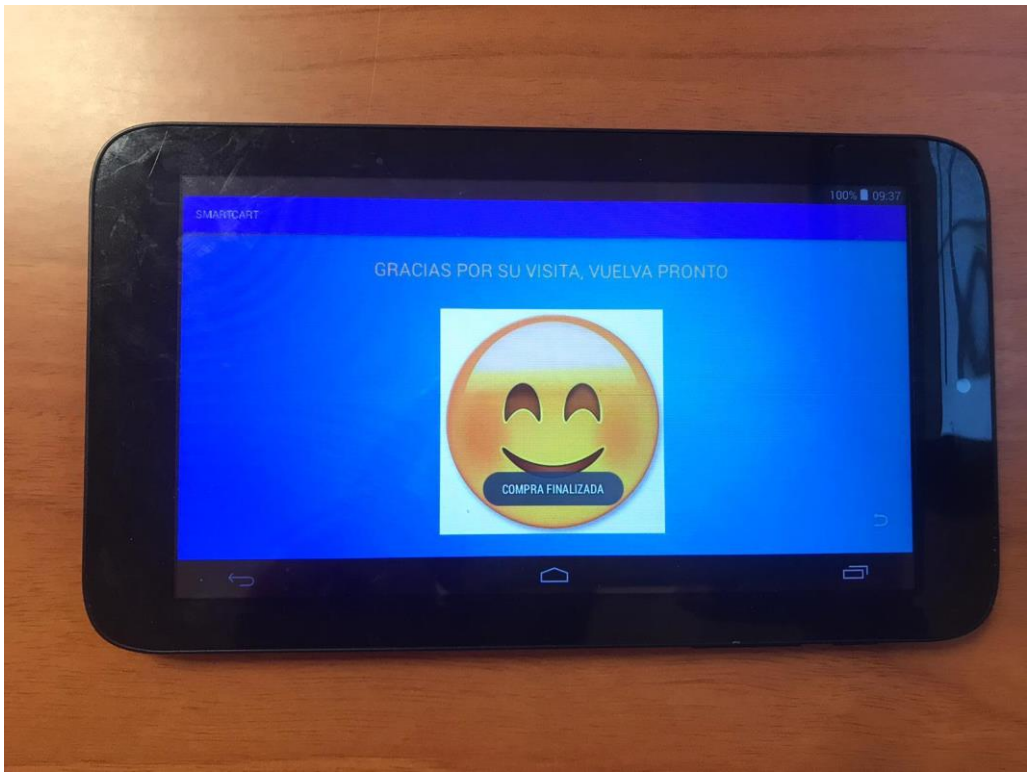



*Imagen 36: Interfaz FINAL*

En segundo lugar, tenemos el diseño del SMARTCART. Incorporamos todos los componentes que debe llevar dentro de la carcasa diseñada. Éstos son:

- La Tablet con la aplicación.
- El lector de código de barras.
- El módulo para cargar baterías.
- El cableado y las planchas metálicas de los laterales.

*Imagen 37: SMARTCART*

En tercer lugar, tenemos todo el sistema de geolocalización con la tecnología BLE. Los *beacons* o balizas electrónicas se instalarán en las estanterías de entrada de cada sección del supermercado. Habrá dos *beacons* por sección:

- Uno para la localización.
- Otro para enviar las ofertas.

También habrá *beacons* en todas las entradas y salidas del centro para controlar los carros en todo momento. En caso de salida del centro, se mandará un aviso a los encargados de seguridad.

Por último, tenemos la base de carga de los carros. Se compondrá de tres parcelas, donde cada una podrá abastecer a 10 SMARTCARTS.

En estas parcelas, tendremos dos paredes en las cuales encontraremos el sistema de carga. Por ellas circularán los cables que alimentarán los carros desde la fuente de alimentación. Se conectarán a unas planchas metálicas. Cuando hagan contacto con las planchas de los laterales de la carcasa del SMARTCART, éste recibirá la energía para cargarse.

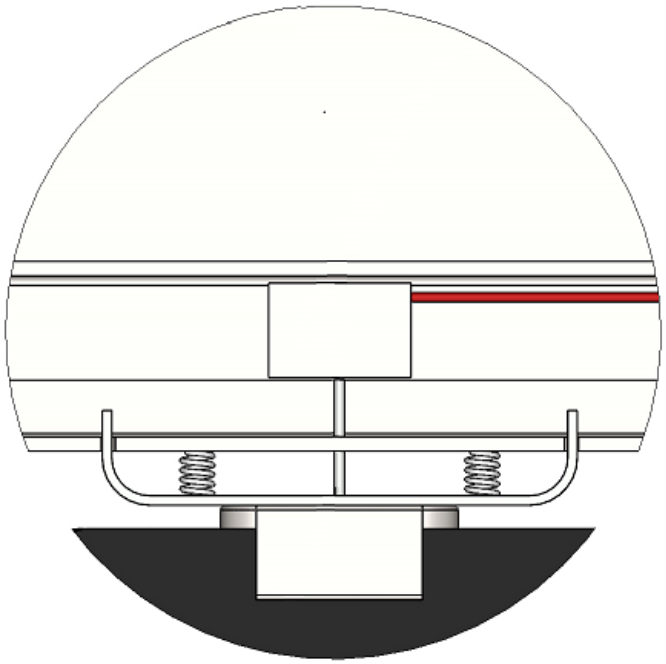


*Imagen 38: Sistema de carga*

Las ruedas traseras de los carros se introducirán por unas vías que estarán atornilladas al suelo de la base.

El resultado podemos verlo en la siguiente imagen:

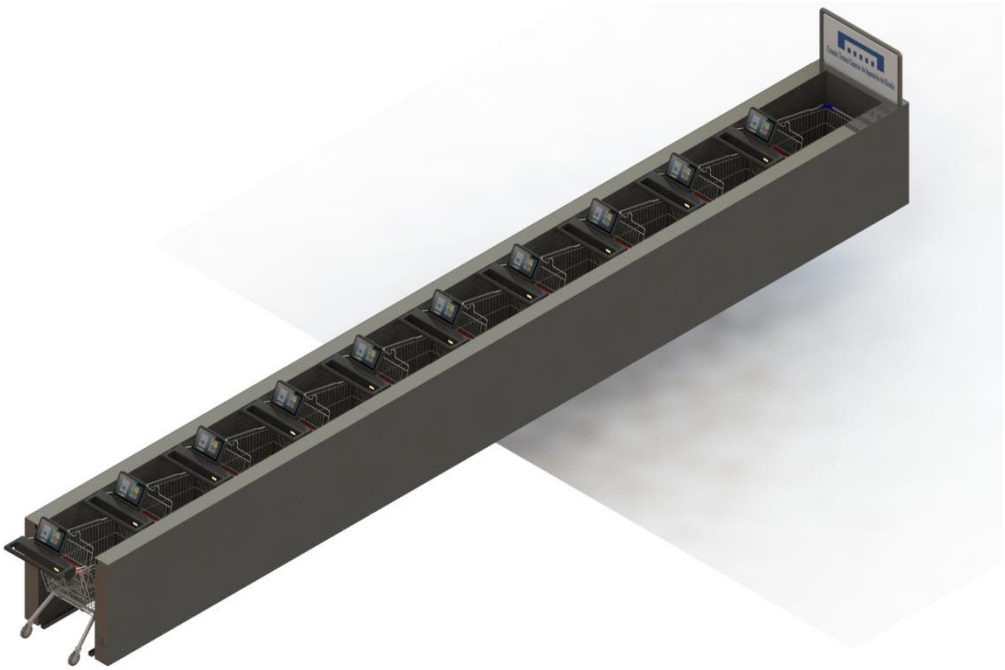


*Imagen 39: Base de carga*

## 7. CONCLUSIONES

Realizar este proyecto me ha supuesto una gran esfuerzo físico y mental que se ha traducido en una satisfacción plena. Poder plasmar mis ideas en un proyecto de este calibre tras los conocimientos recibidos a lo largo de estos años universitarios me hace sentir preparado para afrontar nuevos retos.

Durante los últimos meses, he tenido que realizar varias tareas que han requerido de todos mis esfuerzos y conocimientos: tareas de investigación, tareas de innovación, tareas de programación, etc., además de tareas complejas que precisan de esfuerzo, proyección y sobre todo serenidad, pues es importante mantener la calma en situaciones arduas.

Uno de los momentos más peliagudos que ha condicionado la realización del proyecto ha sido la pandemia global en la que nos hemos visto envueltos, el Coronavirus, que ha obligado a los países a decretar estados de alarma y ha provocado que toda la nación se viera envuelta por cuarentenas de varios meses. Esta situación ha sentenciado la posibilidad de poder completar mi proyecto de forma práctica.

Es por esto por lo que me he visto en la situación de tener que realizar algunas partes del trabajo de manera teórica, como viene a ser todo el proceso de montaje del prototipo. Sin embargo, todo lo que supone el diseño de la aplicación móvil se ha podido finalizar de forma correcta.

En lo que al proyecto se refiere, la mayor dificultad la encontramos en la escritura del código. El tiempo que he necesitado para aprender a programar con Android Studio y a escribir código Java ha sido la tarea más costosa y peliaguda. Todo lo que se refiere al montaje y material ha requerido de horas de investigación y comparación, pero gracias a mis conocimientos en ingeniería electrónica industrial lo pude solventar con más facilidad.

Sin embargo, he de aclarar que algunas decisiones tomadas sobre el tipo de material o recursos utilizados, pueden mejorarse. Es el caso de las bases de datos, por ejemplo. El programa utilizado es costoso de manejar y en ocasiones poco seguro, por lo que otros programas mejor preparados solventarían esta serie de problemas con los que nos encontramos. También podría ser un ejemplo el método de sensorización de los productos, aunque esta decisión se tomó puramente por comodidad.

De todos modos, los resultados obtenidos son del todo gratificantes y considero que la idea planteada puede suponer un método interesante y cómodo para los supermercados. Además, el sistema permite añadir mejoras e incluso avanzar aún más en el sistema de compra, al igual que el proyecto SMARTCART puede ser implementado para otras utilidades como bien podría ser en aeropuertos, almacenes industriales, laboratorios de investigación, etc. Siempre hay que tratar de superarse.

## 8. BIBLIOGRAFÍA


[1] – Agencia Estatal BOE (12 de noviembre de 2007). *Boletín Oficial del Estado. Legislación.* [31 de diciembre de 2008]. Información obtenida en: https://www.boe.es/legislacion/legislacion.php

[2] – Comité de Educación de la IAPD (s.f.). *Termoplásticos de Ingeniería Amorfos y Semicristalinos, Módulo 4.* [2 de marzo de 2012]: Información obtenida en: https://www.plasticseurope.org/es/about-plastics/what-are-plastics/large-family

[3] – Salvat Editores (1869). *Enciclopedia de Ciencia y Técnica. Tomo 1, Aluminio.* [1984]. Información obtenida en: https://es.wikipedia.org/wiki/Aluminio

[4] – PIPO (2015). *Tablet PIPO X4*. [agosto de 2018]. Información obtenida en: https://www.pipo-store.com/pipo-x4.html

[5] – Blanchette, Jasmin; Summerfield, Mark (s.f.). *A Brief History of Qt.* [junio de 2006]. Información obtenida en: https://es.wikipedia.org/wiki/Qt_(biblioteca)

[6] – Sinicki, Adam (s.f.). *I want to develop Android Apps. What languages should I learn?* [10 de agosto de 2019]. Información obtenida en: https://es.wikipedia.org/wiki/Android_Studio

[7] – Urlocker, M. Zack (s.f.). *Google Runs MySQL.* [13 de diciembre de 2005]. Información obtenida en: https://www.mysql.com/

[8] – Kienle, Holger (s.f.). *It's About Time to Take JavaScript (More) Seriously.* [mayo de 2010]. Información obtenida en: https://developers.google.com/apps-script/guides/sheets

[9] – Gómez, Carles; Oller, Joaquim; Paradells, Josep (s.f.). *Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Thecnology.* [2012]. Información obtenida en: https://www.areatecnologia.com/nuevas-tecnologias/bluetooth-le.html

[10] – Bfonics (2014). *Low Energy Bluetooth | bfonics Smart Beacons | BLE.* [8 de noviembre de 2014]. Información obtenida en: https://es.wikipedia.org/wiki/Baliza_electr%C3%B3nica

[11] – Soria López, José Manuel (s.f.). *Cuadro Nacional de Atribución de Frecuencias.* [7 de agosto de 2015]. Información obtenida en: https://www.xataka.com/seleccion/zigbee-z-wave-que-que-se-diferencian-que-marcas-domotica-compatibles

[12] – Malvino, Albert Paul (s.f.). *Principios de Electrónica.* [2000]. Información obtenida en: https://es.wikipedia.org/wiki/Optoacoplador

[13] – Innovae (s.f.). *Realidad Aumentada. Historia de la realidad aumentada.* [2018]. Información obtenida en: https://es.wikipedia.org/wiki/Realidad_aumentada

[14] – Sals, Benjamín Valdez (s.f.). *Tecnología en la UABC.* [2006]. Información obtenida en: https://poliformat.upv.es/access/content/group/GRA_12160_2019/Presentaciones/SeIV_Sensores%C3%93pticos_13_14.pdf

[15] – Isaac (s.f.). *TP4056: el módulo para cargar baterías.* [8 de agosto de 2017]. Información obtenida en: https://www.hwlibre.com/tp4056/

[16] – Masvoltaje (s.f.). *¿Qué tipos de cables eléctricos existen?* [27 de abril de 2016]. Información obtenida en: https://masvoltaje.com/blog/tipos-de-cables-electricos-que-existen-n12

[17] – Amadeo, Ron (s.f.). *Conoce a "Eddystone" de Google, un iBeacon fighter flexible y de código abierto.* [14 de julio de 2015]. Información obtenida en: http://www.usingbeacons.com/eddystone-de-google-nuevas-posibilidades-para-la-tecnologia-beacon/

[18] – González Verdugo, David (s.f.). *Beacons en Android.* [27 de julio de 2017]. Información obtenida en: https://solidgeargroup.com/beacons-en-android/

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

# ANEXO Nº1

## MANUAL DE LA APLICACIÓN

GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

26 DE AGOSTO DE 2020

AUTOR: IGNACIO PAYÁ SILLA
TUTOR: PASCUAL PÉREZ BLASCO

**ÍNDICE**



**ÍNDICE IMÁGENES**

**ÍNDICE DIAGRAMAS**

## 1. INTRODUCCIÓN

En este anexo encontramos el manual para entender cómo es el funcionamiento de la aplicación móvil que hemos diseñado para los SMARTCARTS.

Para ello, vamos a explicar detalladamente lo que tenemos en cada interfaz de la aplicación. Nos vamos a ayudar de unos diagramas de flujo que en su mayor medida, nos van a facilitar la comprensión.

Como en la Tablet no habrá más aplicaciones instaladas, el programa no tiene por qué cerrarse, pero en caso de ser así, no habrá problema para el cliente, pues el título de la aplicación vendrá acompañado del siguiente icono:

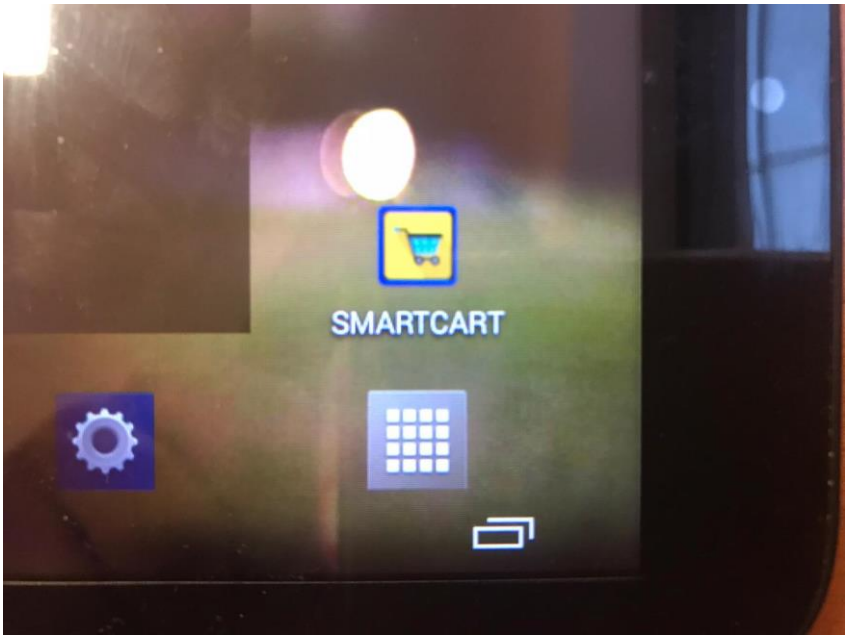



*Imagen 1: Icono de la aplicación*

## 2. INTERFAZ DE INICIO

Esta interfaz corresponde a la primera Activity con la que nos vamos a encontrar nada más iniciemos nuestra aplicación. En ella vamos a encontrar el título de la aplicación y dos comandos, que son los siguientes:

- Botón **EMPLEADO**: Este botón servirá para dar paso a la interfaz de registro de los empleados del supermercado. Sin embargo, en caso de activar sin ser un empleado, no habría ningún problema, ya que tendremos opción de retornar al inicio.

- Botón **CLIENTE**: Este botón servirá para dar paso a la interfaz de los clientes. En ella encontrarán todo lo necesario para realizar la compra correctamente. No requerirá de inscripción o registro alguno. Además, tendrán la opción de regresar al inicio si así lo desean.

Para ayudar a la comprensión, acompañamos la explicación anterior con un diagrama de flujo:

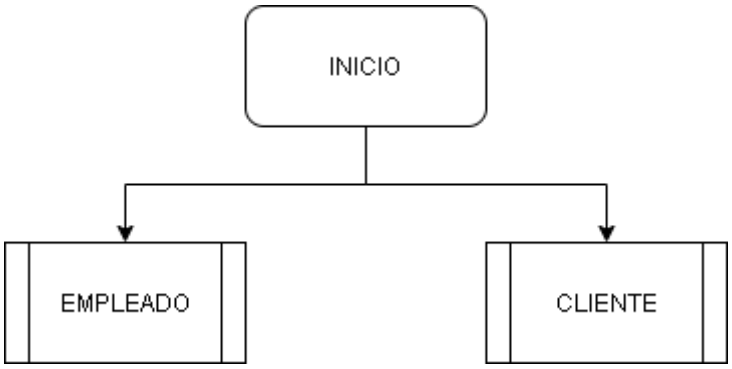



*Diagrama 1: Inicio aplicación del SMARTCART*

Como podemos ver en el diagrama, es muy sencillo el proceso. Iniciamos la aplicación y tenemos dos opciones a escoger: **EMPLEADO** o **CLIENTE**.

En la siguiente imagen podemos comprobar el resultado visual:

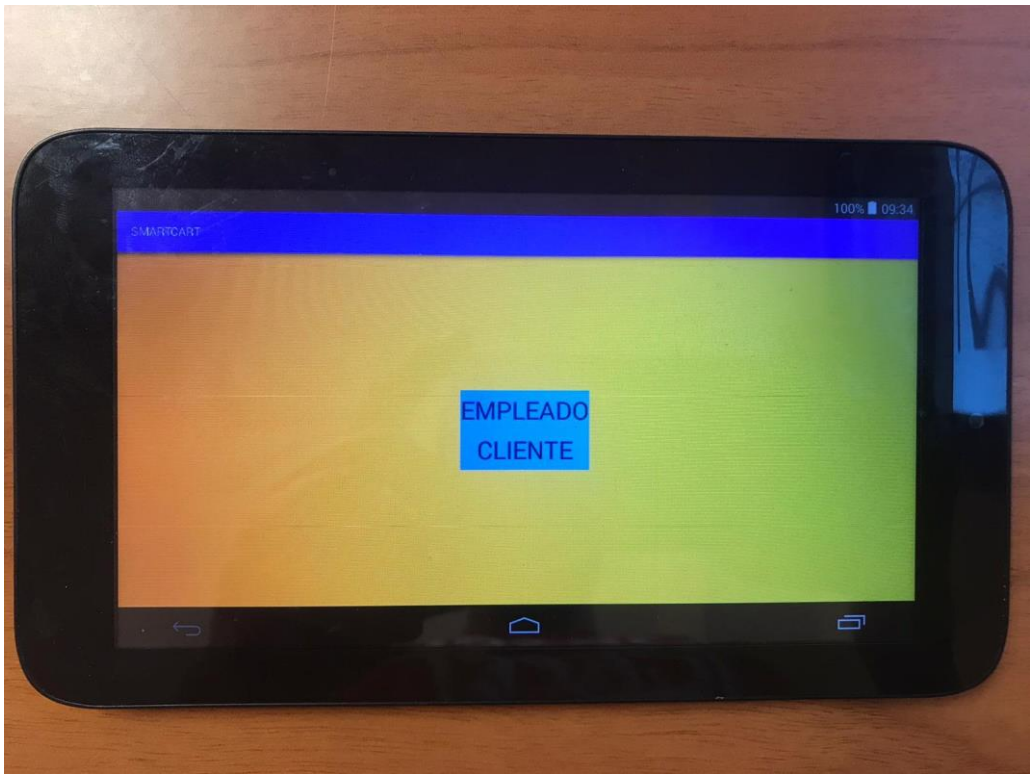



*Imagen 2: Interfaz de inicio*

## 3. INTERFAZ EMPLEADO

Este apartado involucra solamente a los trabajadores del supermercado. Éstos deben clicar en el botón "**EMPLEADO**" para poder iniciar sesión en la aplicación.

En ese momento, se abrirá una nueva interfaz en la que encontrarán los siguientes comandos:

- Editor de texto **USUARIO**: En este comando deben escribir el usuario que hayan escogido y se encuentre almacenado en la base de datos.
- Editor de texto **CONTRASEÑA**: En este comando deben escribir la contraseña que hayan escogido y se encuentre almacenada en la base de datos.
- Botón **ENTRAR**: Este botón les permitirá dirigirse a la interfaz con la que podrán trabajar. En caso de no introducir correctamente el usuario, la contraseña o ambos, la aplicación enviará un mensaje de error y les pedirá de nuevo la introducción del usuario y la contraseña.
- Botón **VOLVER**: Este botón les permitirá regresar a la interfaz de inicio.

En la siguiente imagen podemos comprobar el resultado visual:

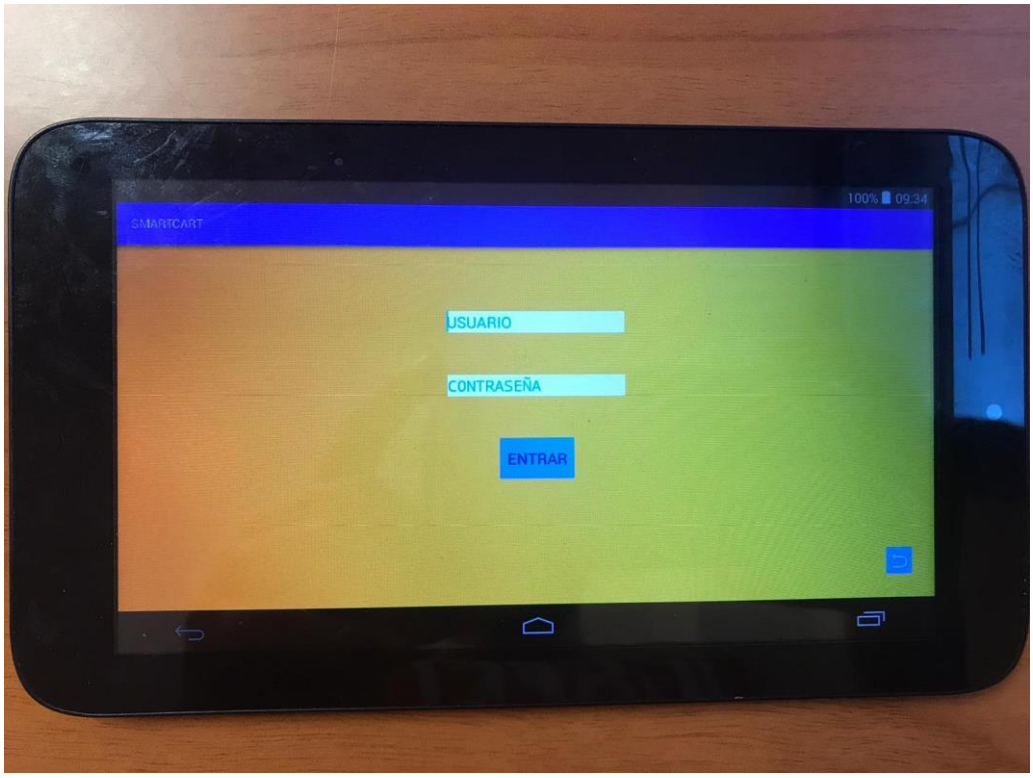



*Imagen 3: Interfaz EMPLEADO*

Para muestra visual, acompañamos la explicación anterior con un diagrama de flujo:

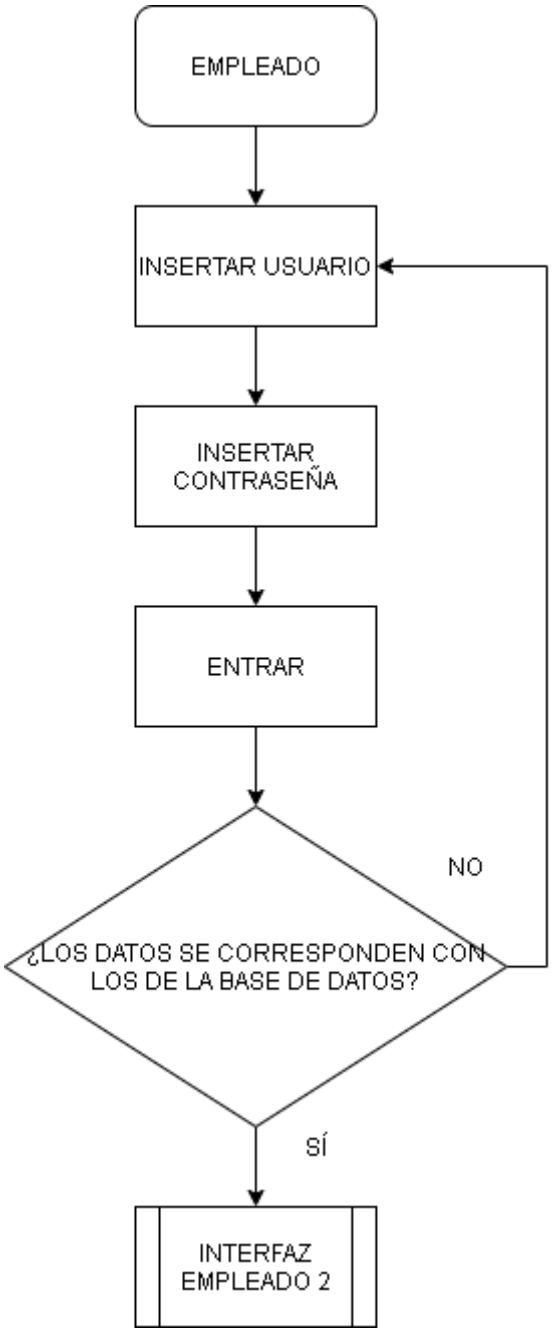



*Diagrama 2: Interfaz del Empleado*

Como podemos ver en el diagrama, una vez habilitada la interfaz del empleado, nos conduce a la introducción del **usuario** y la **contraseña**, por ese orden. Luego pulsamos el botón **ENTRAR** y una vez corroborado en la base de datos, nos habilita la siguiente interfaz. En caso de no ser así, nos conduce de nuevo a la introducción del usuario y la contraseña.

Los botones **VOLVER** no es necesario añadirlos a los diagramas, pues todas las interfaces contienen uno.

## 4. INTERFAZ EMPLEADO 2

Es la continuación de la interfaz anterior. En caso de aprobarse el registro, los empleados se encontrarán con la segunda interfaz, donde tendremos los siguientes comandos:

- Botón **CAJERO 1**: Este botón les permitirá entrar dentro de la interfaz con la que podrán trabajar en el cajero nº1. Recibirán dentro de esta interfaz la información necesaria sobre la compra de los clientes: **Nº** del **carro**, **Precio** y **Peso** de la compra.
- Botón **CAJERO 2**: Este botón les permitirá entrar dentro de la interfaz con la que podrán trabajar en el cajero nº2. Recibirán dentro de esta interfaz la información necesaria sobre la compra de los clientes: **Nº** del **carro**, **Precio** y **Peso** de la compra.
- Botón **CAJERO 3**: Este botón les permitirá entrar dentro de la interfaz con la que podrán trabajar en el cajero nº3. Recibirán dentro de esta interfaz la información necesaria sobre la compra de los clientes: **Nº** del **carro**, **Precio** y **Peso** de la compra.
- Botón **CAJERO 4**: Este botón les permitirá entrar dentro de la interfaz con la que podrán trabajar en el cajero nº4. Recibirán dentro de esta interfaz la información necesaria sobre la compra de los clientes: **Nº** del **carro**, **Precio** y **Peso** de la compra.
- Botón **VOLVER**: Este botón les permitirá regresar a la interfaz anterior, que en este caso es la interfaz **EMPLEADO**.

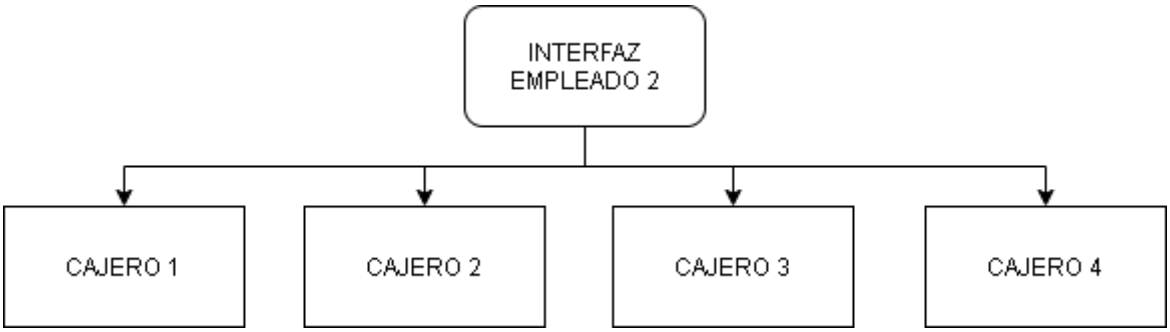



*Diagrama 3: Interfaz Empleado 2*

Como podemos observar, una vez habilitada la interfaz **EMPLEADO 2**, nos encontramos con 4 opciones. Cada opción corresponde a un botón y cada botón corresponde a un cajero distinto. Estos botones abrirán una interfaz diferente donde nos encontraremos la información necesaria sobre la compra de los clientes.

En la siguiente imagen podemos comprobar el resultado visual:

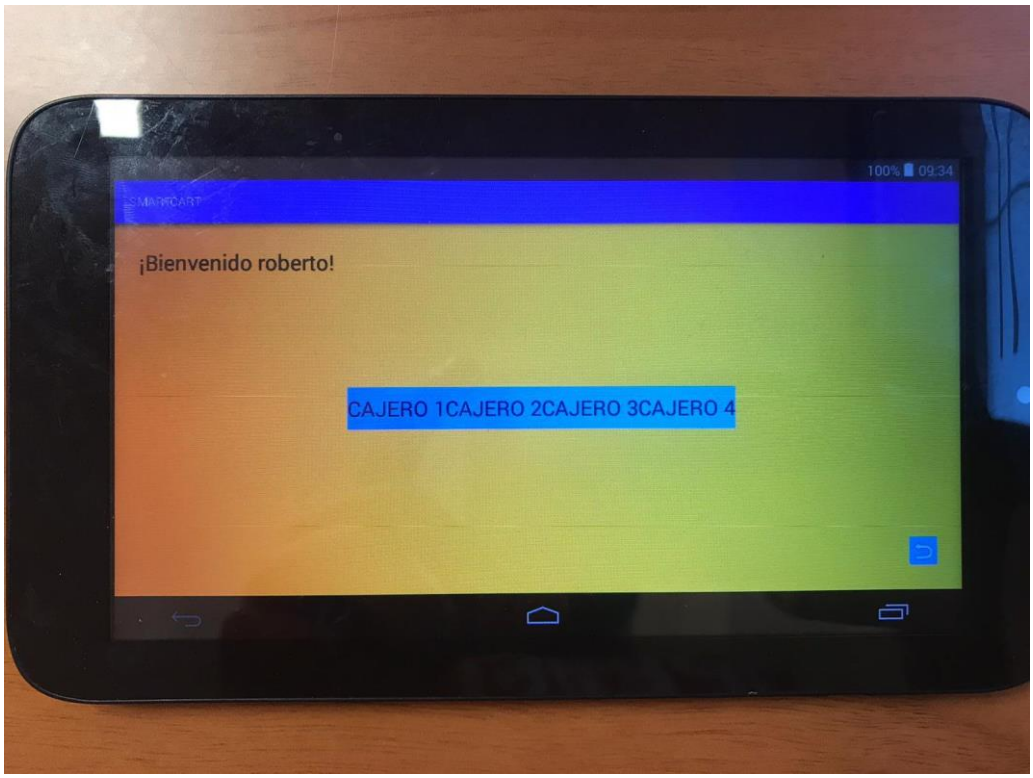



*Imagen 4: Interfaz EMPLEADO 2*

## 5. INTERFAZ CAJERO 1

El Empleado tiene 4 opciones donde escoger, pero en todas vamos a encontrarnos con la misma interfaz. Simplemente, el cambio lo encontramos en el número de cajero que hemos escogido. Para este ejemplo, hemos utilizado la interfaz del cajero 1 y a continuación explicaremos lo que nos vamos a encontrar y cuáles son sus funciones:

- Visor de texto: En él aparece la frase: "**TRABAJANDO CON CAJERO 1**". Si estuviéramos en otro de los cajeros disponibles, cambiaría, únicamente, el número del cajero.
- Editor de texto: En él se debe insertar el número del carro que se quiera atender. Una vez introducido, en los dos visores de texto de abajo aparecerán los valores de **PESO** y **PRECIO** correspondientes al carro seleccionado.
- Visor de texto **PESO**: En él aparecerá el valor del peso de la compra.
- Visor de texto **PRECIO**: En él aparecerá el valor del precio de la compra.
- Botón **FINALIZAR**: Se debe pulsar en caso de finalizar el trabajo con el cajero. Nos retornará a la interfaz de inicio.
- Botón **VOLVER**: Pulsaremos esta opción en caso de querer regresar a la interfaz anterior, que en este caso es la interfaz de selección de cajero.

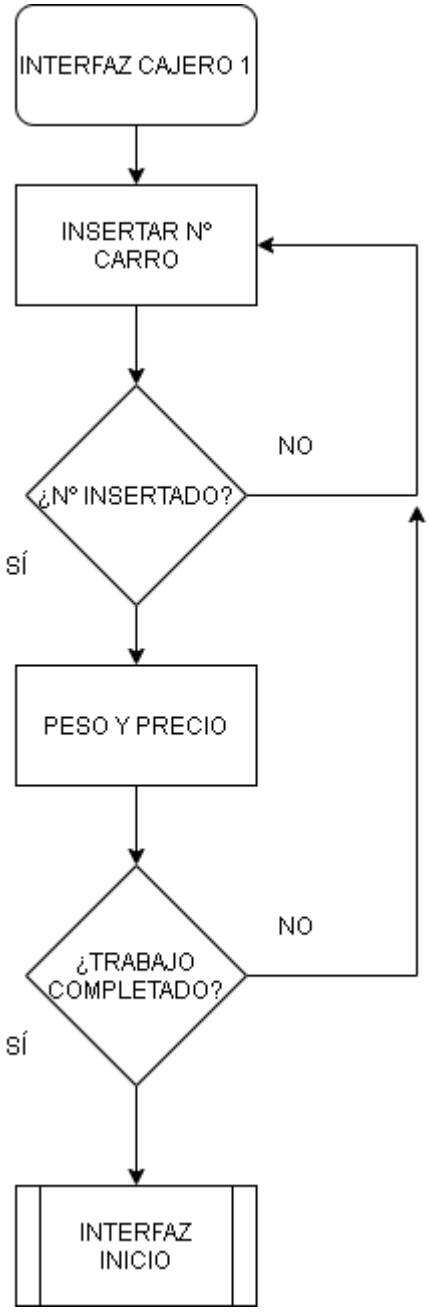



*Diagrama 4: Interfaz Cajero 1*

Como comenté anteriormente, todas las interfaces de los cajeros completan la misma tarea. Por tanto, el diagrama de flujo (4) y la interfaz del ejemplo sirven para todos los cajeros.

En la siguiente imagen podemos comprobar el resultado visual:

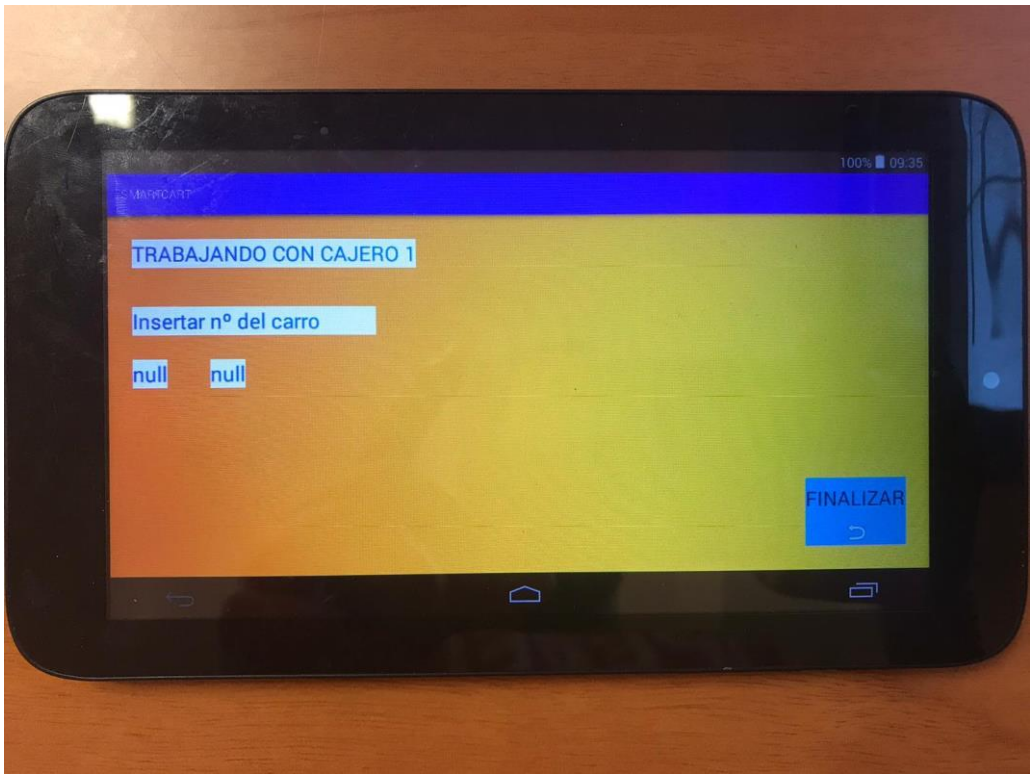



*Imagen 5: Interfaz CAJERO 1*

## 6. INTERFAZ CLIENTE

Para este apartado tenemos que regresar al principio. En este caso, pulsamos el botón **CLIENTE** y nos conduce a la interfaz **CLIENTE**. En ella vamos a encontrar una serie de comandos que nos servirán para realizar todo el proceso de compra, visualizar nuestra posición en el supermercado al momento, etc.

Aquí los describimos:

- Visor de texto **BIENVENIDO**: Nos dan la bienvenida al centro (Arriba izquierda).
- Visor de texto **Nº DEL CARRO**: Nos dan el número del carro (debajo del visor de texto BIENVENIDO).
- Visor de texto **SECCIÓN**: Nos dicen la sección en la que se encuentra en ese momento el carro (debajo del visor de texto Nº DEL CARRO).
- Icono **SECCIÓN**: Depende de la sección en la que se encuentre el carro, nos aparece el icono de dicha sección (debajo del visor de texto SECCIÓN).
- Botón **SECCIONES**: Al pulsarlo nos abre otra interfaz en la que encontramos las secciones que hay en el supermercado (Abajo izquierda).
- Botón **VER LISTA**: Al pulsarlo nos muestra otra interfaz en la que aparece la lista de la compra que estamos realizando (debajo del botón SECCIONES).
- Visor de texto **PESO**: En el momento que finalicemos la lista de la compra aparecerá en el visor el peso de la compra (abajo a la derecha del botón VER LISTA).

- Visor de texto **PRECIO**: En el momento que finalicemos la lista de la compra aparecerá en el visor el precio de la compra (abajo a la derecha del visor de texto PESO).
- Visor de texto **LOCALIZACIÓN**: Nos informan de que el mapa de abajo es la ubicación actual (Arriba derecha).
- Visor **MAPA**: Aparece un plano del supermercado en el que se indica la posición actual del carro (debajo del visor de texto LOCALIZACIÓN).
- Botón **FINALIZAR**: Una vez finalizada la compra, pulsamos este botón, que enviará al cliente a otra interfaz en la que se le indicará a qué cajero debe acudir para realizar las comprobaciones del peso y el precio (Abajo derecha).
- Botón **VOLVER**: Este botón les permitirá regresar a la interfaz de inicio (Abajo derecha).

Aquí podemos visualizar cuáles son los iconos de las secciones del centro:

- Sección FRUTA Y VERDURA

*Imagen 6: Icono FRUTA Y VERDURA*

- Sección CONGELADOS

*Imagen 7: Icono CONGELADOS*

- Sección CHARCUTERÍA

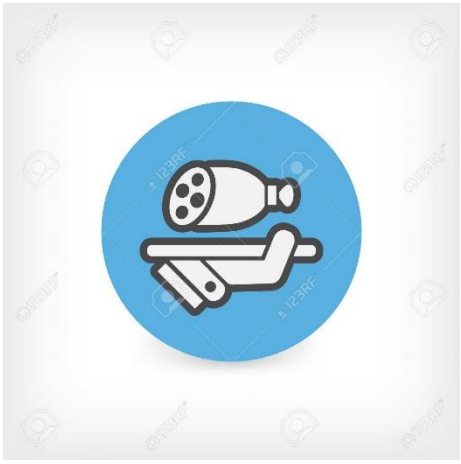


*Imagen 8: Icono CHARCUTERÍA*

- Sección LICORES

*Imagen 9: Icono LICORES*

- Sección DROGUERÍA:

*Imagen 10: Icono DROGUERÍA*

En la siguiente imagen podemos comprobar el resultado visual de la interfaz **CLIENTE** cuando éste entra por primera vez:

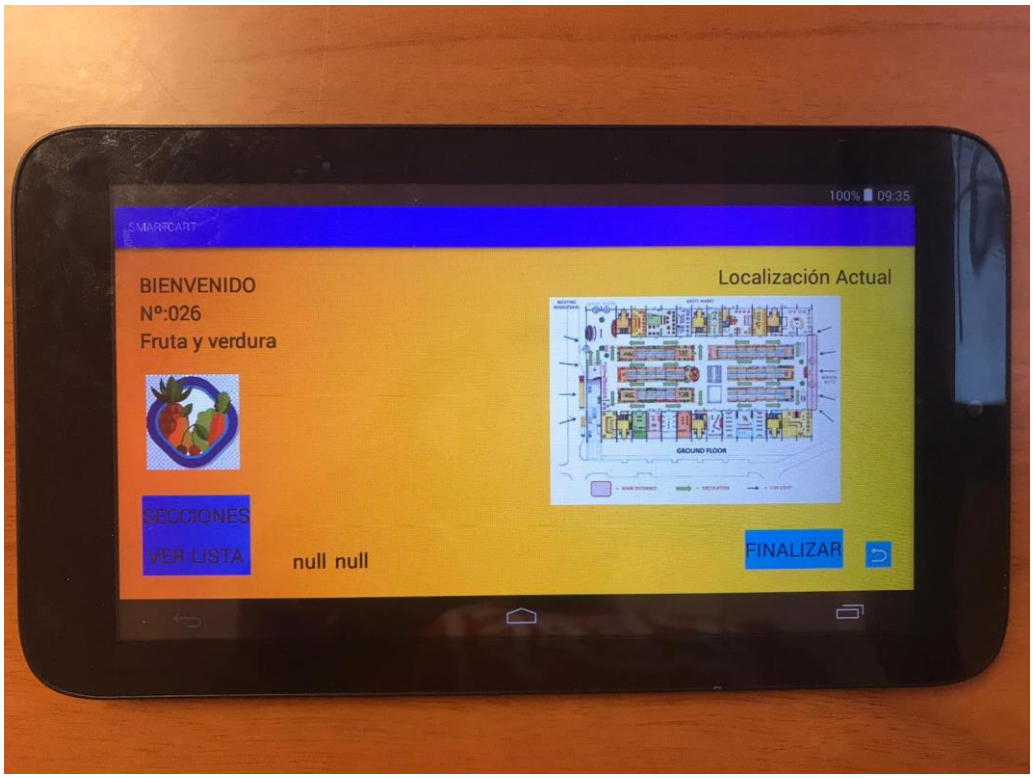



*Imagen 11: Interfaz CLIENTE*

## 7. INTERFAZ SECCIONES

El cliente, desde la interfaz **CLIENTE**, ha pulsado el botón **SECCIONES**. Aparece la interfaz **SECCIONES** y encontramos en ella una serie de botones con los títulos de las secciones de las que dispone el supermercado. Al escoger alguna de ellas, nos aparece todo el listado de productos que hay en dicha sección.

Aquí describimos los comandos de la interfaz:

- Botón **FRUTA Y VERDURA**: Al pulsarlo nos abre otra interfaz con la lista de productos de esta sección.
- Botón **CONGELADOS**: Al pulsarlo nos abre otra interfaz con la lista de productos de esta sección.
- Botón **CHARCUTERÍA**: Al pulsarlo nos abre otra interfaz con la lista de productos de esta sección.
- Botón **LICORES**: Al pulsarlo nos abre otra interfaz con la lista de productos de esta sección.
- Botón **DROGUERÍA**: Al pulsarlo nos abre otra interfaz con la lista de productos de esta sección.
- Botón **VOLVER**: Este botón les permitirá regresar a la interfaz anterior, que en este caso es la interfaz CLIENTE.

En la siguiente imagen podemos comprobar el resultado visual:

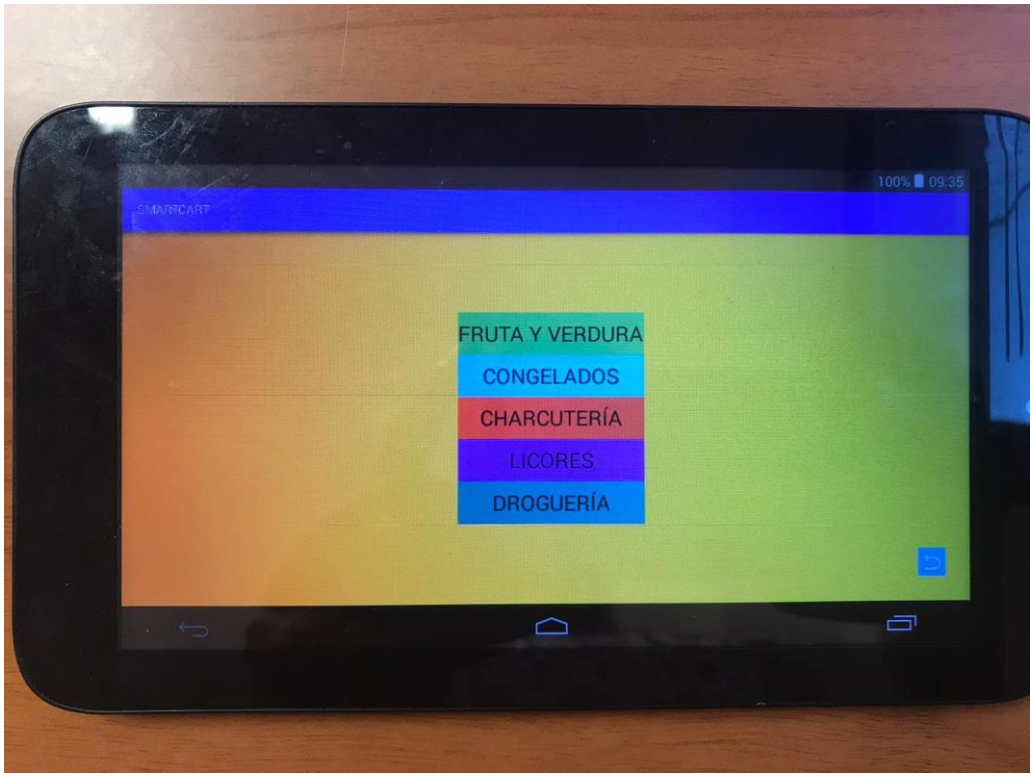



*Imagen 12: Interfaz SECCIONES*

## 8. INTERFAZ FRUTA Y VERDURA

El cliente, desde la interfaz **SECCIONES**, ha pulsado el botón **FRUTA Y VERDURA**. Aparece la interfaz **FRUTA Y VERDURA** y encontramos en ella un listado de todos los productos que hay en esta sección. En caso de pulsar otra sección en la interfaz **SECCIONES**, aparecerá la misma interfaz y solamente cambiará el listado de productos, que encontraremos el de la sección correspondiente.

Aquí describimos los comandos de la interfaz utilizada de ejemplo:

- Botones de productos **FRUTA Y VERDURA**: Al pulsarlos recibimos información específica de cada producto.
- Botón **VOLVER**: Este botón les permitirá regresar a la interfaz anterior, que en este caso es la interfaz SECCIONES.

En la siguiente imagen podemos comprobar el resultado visual:

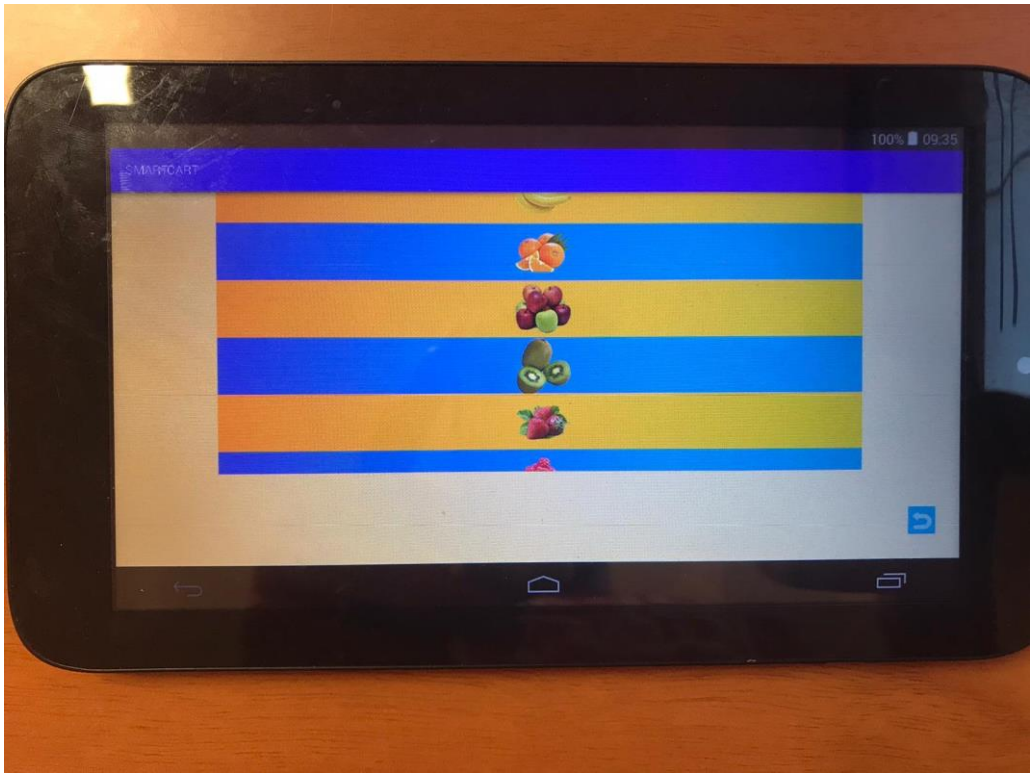



*Imagen 13: Interfaz FRUTA Y VERDURA*

Como se ha comentado anteriormente, pulsando cualquier sección en la interfaz **SECCIONES** se muestra otra interfaz con el listado de productos correspondiente a dicha sección.

## 9. INTERFAZ VER LISTA

Regresamos a la interfaz **CLIENTE** y pulsamos el botón **VER LISTA**. Aparece la interfaz **VER LISTA** y en ella encontramos un cajetín vacío, dos visores vacíos (uno para el PESO y otro para el PRECIO) y dos botones. Conforme el cliente vaya añadiendo productos al carro a través del lector de código de barras, el cajetín se irá completando con los productos que el lector esté detectando. Los visores PESO y PRECIO recogerán los datos correspondientes de cada producto. Una vez finalizada la compra, se pulsa el botón **ENVIAR** y aparece de nuevo la interfaz **CLIENTE**, donde aparecerán los datos PESO y PRECIO correspondientes al de la lista de la compra.

Estos son los comandos:

- Cajetín **PRODUCTOS**: Este comando, al inicio, aparecerá vacío. En cuanto pasemos un producto por el lector de código de barras y éste lo detecte, el producto aparecerá en el cajetín.
- Visor **PESO**: Aquí aparecerá la suma de los datos PESO de cada producto que se introduzca en la cesta del carro.

- Visor **PRECIO**: Aquí aparecerá la suma de los datos PRECIO de cada producto que se introduzca en la cesta del carro.
- Botón **VOLVER**: Este botón les permitirá regresar a la interfaz anterior, que en este caso es la interfaz CLIENTE.

En la siguiente imagen podemos comprobar el resultado visual:

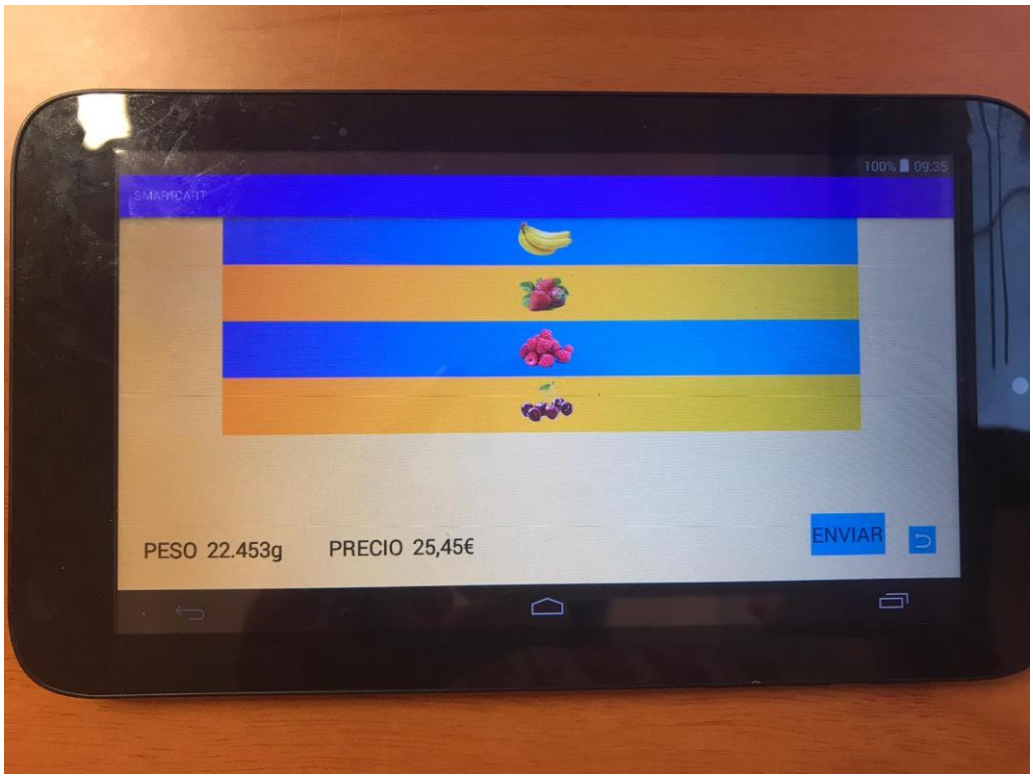



*Imagen 14: Interfaz VER LISTA*

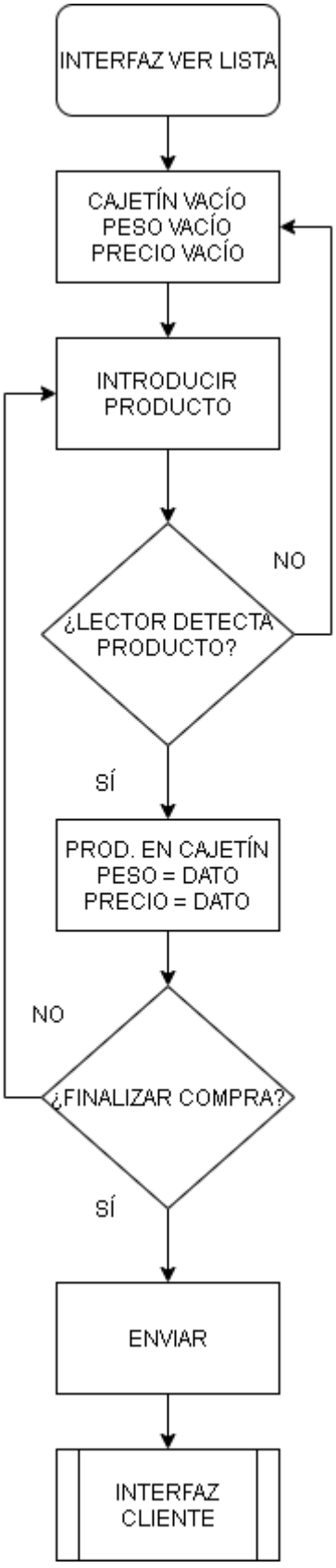



*Diagrama 5: Interfaz Ver lista*

En la siguiente imagen, veremos la interfaz **CLIENTE** una vez se haya pulsado el botón **ENVIAR** de la interfaz **VER LISTA**. Encontraremos los valores de **PESO** y **PRECIO** con los datos proporcionados desde la interfaz **VER LISTA**:

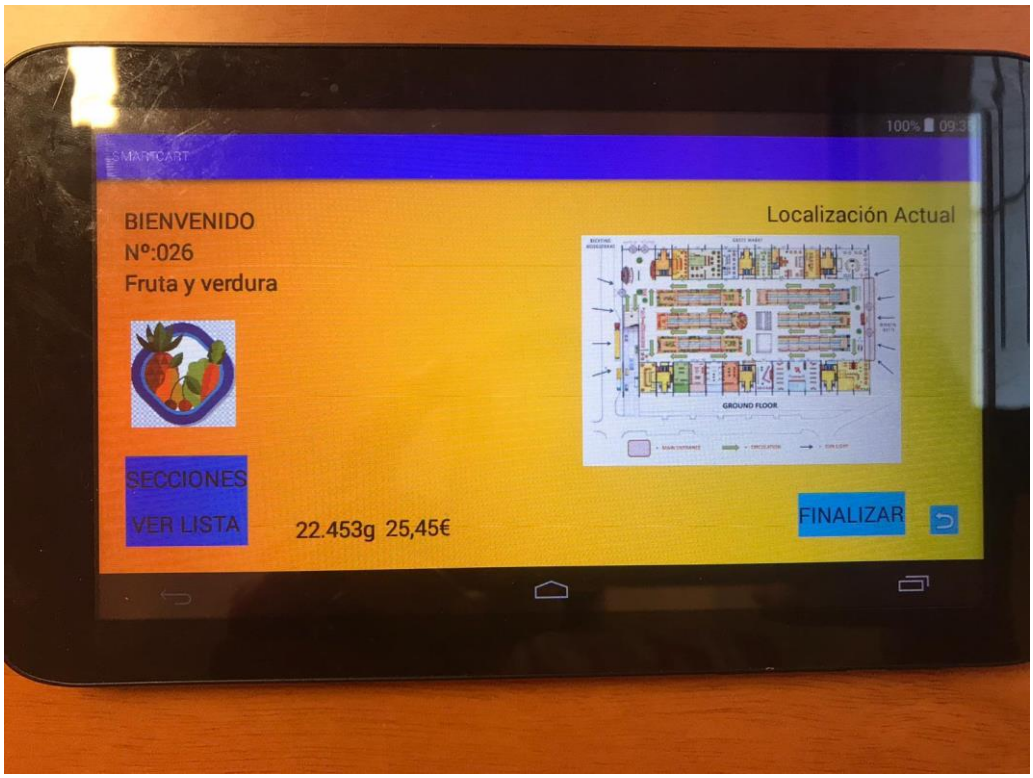



*Imagen 15: Interfaz CLIENTE con datos PESO y PRECIO*

## 10. INTERFAZ ESPERA

Hemos regresado a la interfaz **CLIENTE,** pero esta vez con la lista de la compra finalizada y los datos **PESO** y **PRECIO** incorporados en los visores de la interfaz **CLIENTE**. Pulsamos el botón **FINALIZAR** y aparece una interfaz nueva llamada **ESPERA**, donde se nos indica a qué cajero tenemos que acudir para que nos atiendan.

Estos son los comandos:

- Visor de texto **CAJERO**: Nos muestra el cajero al que debemos acudir con nuestro carro.
- Botón **FINALIZAR**: Este botón se pulsará en cuanto el empleado del cajero nos haya atendido y se haya completado el pago. Aparecerá la interfaz **FINAL**.

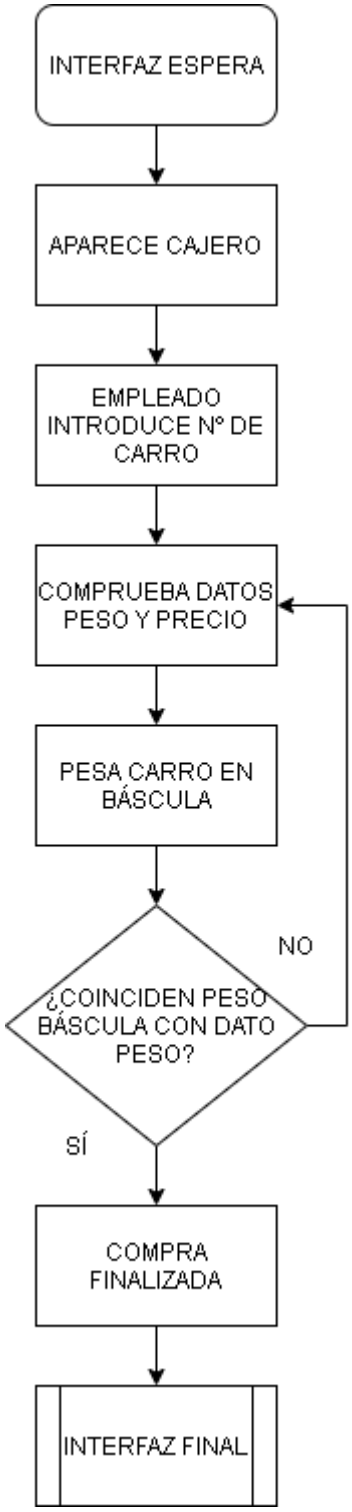



*Diagrama 6: Interfaz Espera*

En la siguiente imagen podemos comprobar el resultado visual:


*Imagen 16: Interfaz ESPERA*

### 11. INTERFAZ FINAL

Nos encontramos con la última interfaz de la aplicación. En ella simplemente aparece un mensaje de agradecimiento por la compra, un *Emoji* sonriente y un botón que nos permite regresar a la interfaz de **INICIO**.

Los comandos son los siguientes:

- Visor de Texto **DESPEDIDA**: Aparece un mensaje de agradecimiento y despedida.
- Imagen **EMOJI**: Es una imagen de un *Emoji* sonriente.
- Botón **VOLVER**: Nos permite regresar al inicio y comenzar el proceso de nuevo.

En la siguiente imagen podemos comprobar el resultado visual:


*Imagen 17: Interfaz FINAL*

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

# ANEXO Nº2

## CÓDIGO ANDROID STUDIO

GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

26 DE AGOSTO DE 2020

AUTOR: IGNACIO PAYÁ SILLA
TUTOR: PASCUAL PÉREZ BLASCO

**ÍNDICE**

**ÍNDICE IMÁGENES**

## 1. INTRODUCCIÓN

Para este anexo se va a establecer el código Java de la aplicación diseñada en el programa Android Studio. Se indicará en todo momento a qué interfaz corresponde cada escritura, además de la explicación de cada comando.

También se va a compartir el código ***.xml*** correspondiente a la parte gráfica de la aplicación. A continuación, presentaremos todos los resultados.

## 2. INTERFAZ DE INICIO

Se trata de la interfaz que aparece al momento de abrir la aplicación en el dispositivo móvil. Nos encontramos con el título de la aplicación y dos botones a escoger:

- Botón **EMPLEADO**: Habilita la interfaz para los empleados.
- Botón **CLIENTE**: Habilita la interfaz para los clientes.

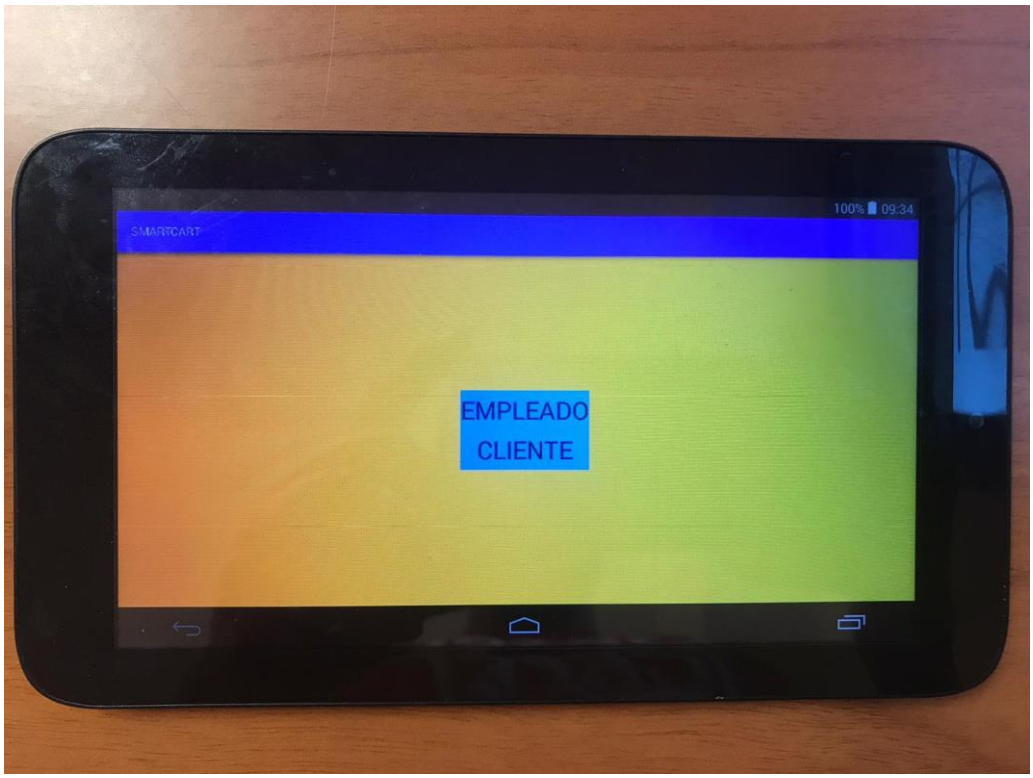



*Imagen 1: Interfaz de inicio*

### 2.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
```

```
import android.view.View;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    //Método para el botón EMPLEADO
    public void Empleado (View view){
        Intent empleado = new Intent (this, EmpleadoActivity.class);
        startActivity(empleado);
    }
    //Método para el botón CLIENTE
    public void Cliente (View view){
        Intent cliente = new Intent (this, ClienteActivity.class);
        startActivity(cliente);
    }
}
```

2.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <Button
            android:id="@+id/b_empleado"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_empleado"
            android:textSize="30sp" />

        <Button
            android:id="@+id/b_cliente"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_cliente"
            android:textSize="30sp" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 3. INTERFAZ EMPLEADO

En esta interfaz, nos encontramos en la situación siguiente:

- El operario ha escogido la opción “EMPLEADO” en la primera interfaz.
- El operario, supuestamente, es un empleado del supermercado y quiere entrar en la aplicación, por lo que va a encontrarse con una interfaz en la cual le van a pedir que se registre.

La interfaz contiene los siguientes comandos:

- Editor de texto: Para escribir el usuario.
- Editor de texto: Para escribir la contraseña.
- Botón **ENTRAR**: Una vez verificado que es un empleado del supermercado, habilita la segunda interfaz para los empleados.
- Botón **VOLVER**: Para regresar a la primera interfaz.

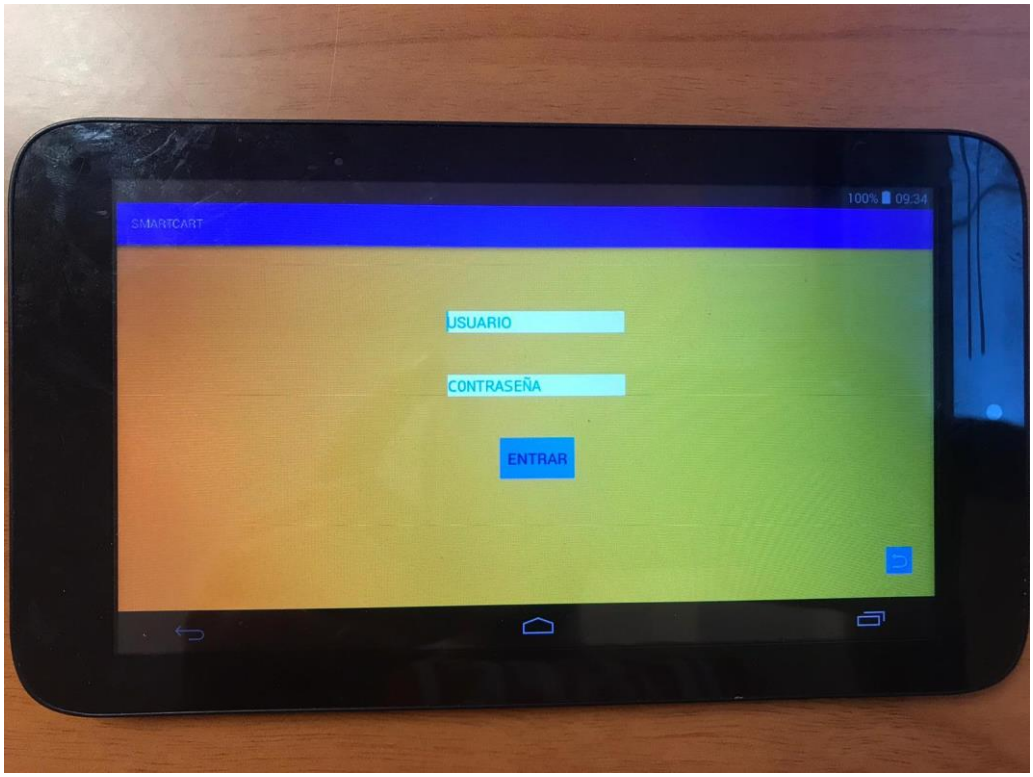



*Imagen 2: Interfaz EMPLEADO*

### 3.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import android.app.ProgressDialog;
import com.android.volley.Response;
import com.android.volley.VolleyError;

import org.json.JSONException;
import org.json.JSONObject;

public class EmpleadoActivity extends AppCompatActivity {
    private EditText etu, etc;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_empleado);

        etu = (EditText)findViewById(R.id.et_usuario);
        etc = (EditText)findViewById(R.id.et_contraseña);
    }
    //Método para el botón entrar, que va a leer la base de datos del Google
Sheets
    public void Registro (View view){

        String usuario = etu.getText().toString();
        String password = etc.getText().toString();

        ProgressDialog cargar = ProgressDialog.show(this,"Cargando...","un
momentito",false,true);
        String url =
"https://script.google.com/macros/s/1d9RgX9nY8t0qLH9tIx1_mamM5B3_DQNsWni7CYKG3K7C6
A9LnTNXfocp/exec?tarea=leer&usuario="+usuario+"&password"+password;
        url = url.replace(" ","%20");

        StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
new Response.Listener<String>() {


            @Override
            public void onResponse(String response) {
                try {
                    JSONObject jobj = new JSONObject(response);
                    String result = jobj.getString ("response");
                    String usuario = jobj.getString("usuario");
                    String password = jobj.getString("contraseña");
                    if(result.equals("registro correcto")){
                        Intent intent = new
Intent(getApplicationContext(),Empleado2Activity.class);
                            intent.putExtra("usuario", usuario);
                            intent.putExtra("contraseña", password);
                            startActivity(intent);
                            Toast.makeText(this, "Has iniciado sesión como " +
usuario, Toast.LENGTH_LONG).show();
```

```
                    } if(result.equals("usuario o contraseña no son correctos")){
                        Toast.makeText(this, "Usuario o contraseña incorrectos",
Toast.LENGTH_LONG).show();
                        etu.setText("");
                        etc.setText("");
                    }
                }
                catch (JSONException r){
                    r.printStackTrace();
                }
            }
        } new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
            loading.dismiss();
            }
        }
    }

    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, MainActivity.class);
        startActivity(volver);
    }
}
```

3.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".EmpleadoActivity">

    <ImageButton
        android:id="@+id/b_volver"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        android:contentDescription="@string/b_volver"
        android:onClick="Volver"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_menu_revert" />

    <EditText
        android:id="@+id/et_usuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="75dp"
        android:ems="10"
        android:hint="@string/et_usuario"
        android:inputType="textPersonName"
```

```
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.499"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/et_contraseña"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:ems="10"
        android:hint="@string/et_contraseña"
        android:inputType="textPassword"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et_usuario" />

    <Button
        android:id="@+id/b_registro"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:onClick="Registro"
        android:text="@string/b_registro"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et_contraseña" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

### 4. INTERFAZ EMPLEADO 2

En esta interfaz, nos encontramos en la situación siguiente:

- El operario se ha registrado con éxito.
- El operario ahora tiene que escoger el cajero en el que va a trabajar.

La interfaz contiene los siguientes comandos:

- Visor de texto: Aparece el usuario del empleado.
- Botón **CAJERO 1**: Habilita la interfaz del cajero 1.
- Botón **CAJERO 2**: Habilita la interfaz del cajero 2.
- Botón **CAJERO 3**: Habilita la interfaz del cajero 3.
- Botón **CAJERO 4**: Habilita la interfaz del cajero 4.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

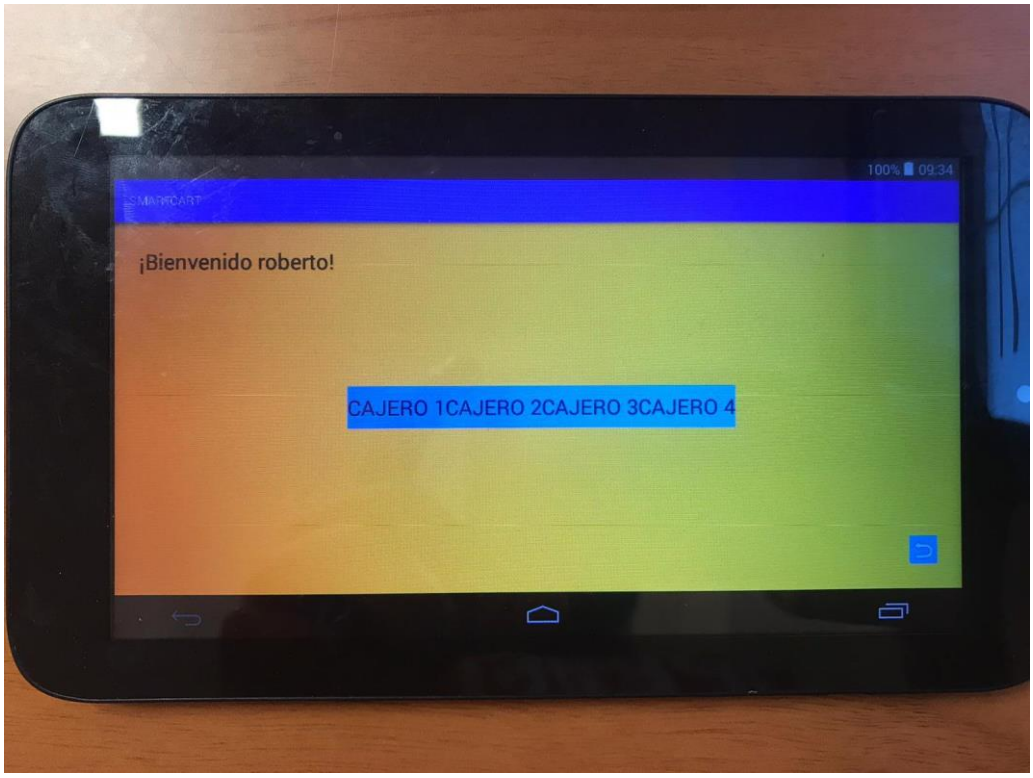



*Imagen 3: Interfaz EMPLEADO 2*

4.1. Código Java:

```
 package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class Empleado2Activity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_empleado2);
    }

    //Método para el botón cajero 1
    public void Cajero_uno (View view) {
        Intent cajero = new Intent (this, Cajero1Activity.class);
        startActivity(cajero);
    }

    //Método para el botón cajero 2
    public void Cajero_dos (View view) {
        Intent cajero = new Intent (this, Cajero2Activity.class);
        startActivity(cajero);
    }
```

```
    //Método para el botón cajero 3
    public void Cajero_tres (View view) {
        Intent cajero = new Intent (this, Cajero3Activity.class);
        startActivity(cajero);
    }

    //Método para el botón cajero 4
    public void Cajero_cuatro (View view) {
        Intent cajero = new Intent (this, Cajero4Activity.class);
        startActivity(cajero);
    }

    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, EmpleadoActivity.class);
        startActivity(volver);
    }
}
```

4.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Empleado2Activity">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_1"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <Button
            android:id="@+id/b_cajero1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
```

```
            android:onClick="Cajero_uno"
            android:text="@string/b_cajero1" />

        <Button
            android:id="@+id/b_cajero2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="Cajero_dos"
            android:text="@string/b_cajero2" />

        <Button
            android:id="@+id/b_cajero3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="Cajero_tres"
            android:text="@string/b_cajero3" />

        <Button
            android:id="@+id/b_cajero4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="Cajero_cuatro"
            android:text="@string/b_cajero4" />
    </LinearLayout>

    <ImageButton
        android:id="@+id/b_volver3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        android:contentDescription="@string/b_volver3"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_menu_revert" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

## 5. INTERFAZ CAJERO 1

En esta interfaz, nos encontramos en la situación siguiente:

- El operario ha escogido el cajero 1.
- El operario introducirá el número del carro al que tenga que atender y recibirá la información del peso y precio de la compra.

La interfaz contiene los siguientes comandos:

- Visor de texto: Aparece el cajero que está utilizando.
- Editor de texto: Para introducir el número del carro.

- Visor de texto: Aparece el peso de la compra.
- Visor de texto: Aparece el precio de la compra.
- Botón **FINALIZAR**: Finaliza el trabajo hecho en el cajero 1 y regresa al principio de la aplicación.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

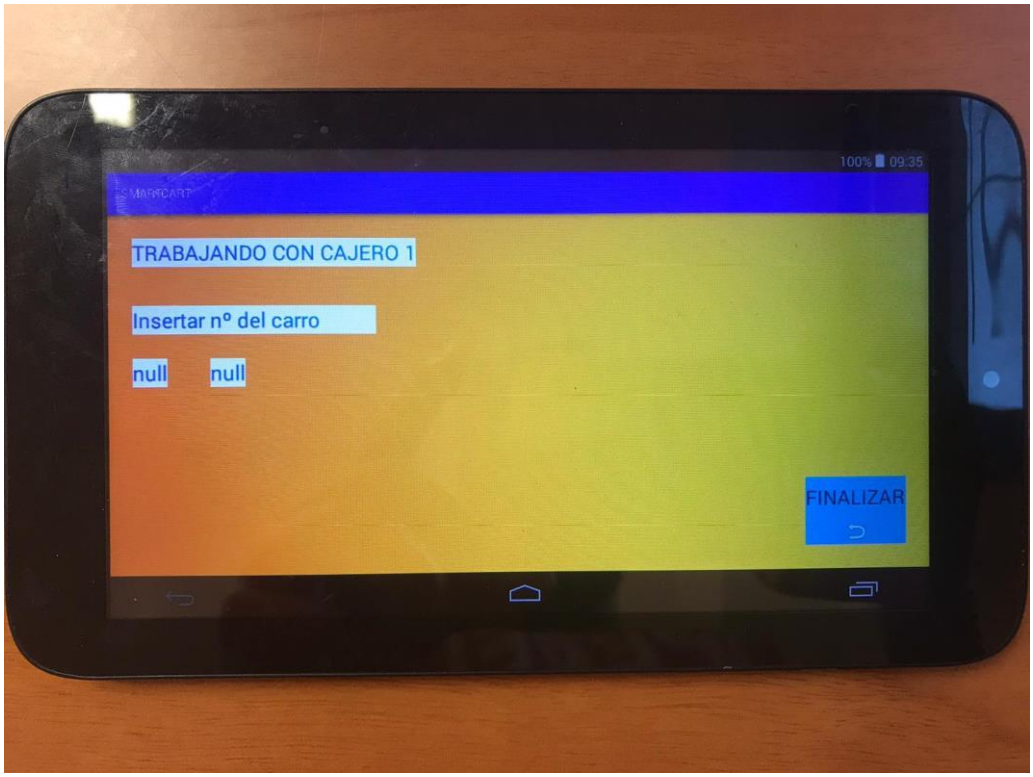



*Imagen 4: Interfaz CAJERO 1*

5.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;


public class Cajero1Activity extends AppCompatActivity {

    private TextView tvpeso;
    private TextView tvprecio;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cajero1);

        tvpeso = (TextView)findViewById(R.id.tv_3);
        tvprecio = (TextView)findViewById(R.id.tv_4);
```

```
        //Método de envio de datos del peso y el precio
        String dato3 = getIntent().getStringExtra("dato3");
        tvpeso.setText("" + dato3);
        String dato4 = getIntent().getStringExtra("dato4");
        tvprecio.setText("" + dato4);
    }

    //Método para el botón finalizar
    public void Finalizar (View view) {
        Intent finalizar = new Intent (this, MainActivity.class);
        startActivity(finalizar);
    }

    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, Empleado2Activity.class);
        startActivity(volver);
    }
}
```

5.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Cajero1Activity">

    <TextView
        android:id="@+id/tv_2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_2"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/et_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="45dp"
        android:ems="10"
        android:hint="@string/et_1"
        android:inputType="textPersonName"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/tv_2" />

    <TextView
        android:id="@+id/tv_3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_3"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et_1" />

    <TextView
        android:id="@+id/tv_4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginLeft="50dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_4"
        android:textSize="24sp"
        app:layout_constraintStart_toEndOf="@+id/tv_3"
        app:layout_constraintTop_toBottomOf="@+id/et_1" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent">

        <Button
            android:id="@+id/b_finalizar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_finalizar"
            android:textSize="24sp" />

        <ImageButton
            android:id="@+id/b_volver4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:contentDescription="@string/b_volver4"
            app:srcCompat="@android:drawable/ic_menu_revert" />
    </LinearLayout>


</androidx.constraintlayout.widget.ConstraintLayout>
```

## 6. INTERFAZ CAJERO 2

En esta interfaz, nos encontramos en la situación siguiente:

- El operario ha escogido el cajero 2.
- El operario introducirá el número del carro al que tenga que atender y recibirá la información del peso y precio de la compra.

La interfaz contiene los siguientes comandos:

- Visor de texto: Aparece el cajero que está utilizando.
- Editor de texto: Para introducir el número del carro.
- Visor de texto: Aparece el peso de la compra.
- Visor de texto: Aparece el precio de la compra.
- Botón **FINALIZAR**: Finaliza el trabajo hecho en el cajero 2 y regresa al principio de la aplicación.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

### 6.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;


public class Cajero1Activity extends AppCompatActivity {

    private TextView tvpeso;
    private TextView tvprecio;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cajero1);

        tvpeso = (TextView)findViewById(R.id.tv_3);
        tvprecio = (TextView)findViewById(R.id.tv_4);

        //Método de envio de datos del peso y el precio
        String dato3 = getIntent().getStringExtra("dato3");
        tvpeso.setText("" + dato3);
        String dato4 = getIntent().getStringExtra("dato4");
        tvprecio.setText("" + dato4);
    }

    //Método para el botón finalizar
    public void Finalizar (View view) {
        Intent finalizar = new Intent (this, MainActivity.class);
        startActivity(finalizar);
    }
```

```
    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, Empleado2Activity.class);
        startActivity(volver);
    }
}
```

6.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Cajero1Activity">

    <TextView
        android:id="@+id/tv_2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_2"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/et_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="45dp"
        android:ems="10"
        android:hint="@string/et_1"
        android:inputType="textPersonName"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tv_2" />

    <TextView
        android:id="@+id/tv_3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_3"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et_1" />

    <TextView
```

```
        android:id="@+id/tv_4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginLeft="50dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_4"
        android:textSize="24sp"
        app:layout_constraintStart_toEndOf="@+id/tv_3"
        app:layout_constraintTop_toBottomOf="@+id/et_1" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent">

        <Button
            android:id="@+id/b_finalizar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_finalizar"
            android:textSize="24sp" />

        <ImageButton
            android:id="@+id/b_volver4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:contentDescription="@string/b_volver4"
            app:srcCompat="@android:drawable/ic_menu_revert" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

### 7. INTERFAZ CAJERO 3

En esta interfaz, nos encontramos en la situación siguiente:

- El operario ha escogido el cajero 3.
- El operario introducirá el número del carro al que tenga que atender y recibirá la información del peso y precio de la compra.

La interfaz contiene los siguientes comandos:

- Visor de texto: Aparece el cajero que está utilizando.
- Editor de texto: Para introducir el número del carro.
- Visor de texto: Aparece el peso de la compra.
- Visor de texto: Aparece el precio de la compra.
- Botón **FINALIZAR**: Finaliza el trabajo hecho en el cajero 3 y regresa al principio de la aplicación.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

7.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;


public class Cajero1Activity extends AppCompatActivity {

    private TextView tvpeso;
    private TextView tvprecio;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cajero1);

        tvpeso = (TextView)findViewById(R.id.tv_3);
        tvprecio = (TextView)findViewById(R.id.tv_4);

        //Método de envio de datos del peso y el precio
        String dato3 = getIntent().getStringExtra("dato3");
        tvpeso.setText("" + dato3);
        String dato4 = getIntent().getStringExtra("dato4");
        tvprecio.setText("" + dato4);
    }

    //Método para el botón finalizar
    public void Finalizar (View view) {
        Intent finalizar = new Intent (this, MainActivity.class);
        startActivity(finalizar);
    }

    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, Empleado2Activity.class);
        startActivity(volver);
    }
}
```

7.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Cajero1Activity">

    <TextView
        android:id="@+id/tv_2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_2"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/et_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="45dp"
        android:ems="10"
        android:hint="@string/et_1"
        android:inputType="textPersonName"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tv_2" />

    <TextView
        android:id="@+id/tv_3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_3"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et_1" />

    <TextView
        android:id="@+id/tv_4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginLeft="50dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_4"
        android:textSize="24sp"
        app:layout_constraintStart_toEndOf="@+id/tv_3"
        app:layout_constraintTop_toBottomOf="@+id/et_1" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent">

        <Button
```

```
            android:id="@+id/b_finalizar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_finalizar"
            android:textSize="24sp" />

        <ImageButton
            android:id="@+id/b_volver4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:contentDescription="@string/b_volver4"
            app:srcCompat="@android:drawable/ic_menu_revert" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 8. INTERFAZ CAJERO 4

En esta interfaz, nos encontramos en la situación siguiente:

- El operario ha escogido el cajero 4.
- El operario introducirá el número del carro al que tenga que atender y recibirá la información del peso y precio de la compra.

La interfaz contiene los siguientes comandos:

- Visor de texto: Aparece el cajero que está utilizando.
- Editor de texto: Para introducir el número del carro.
- Visor de texto: Aparece el peso de la compra.
- Visor de texto: Aparece el precio de la compra.
- Botón **FINALIZAR**: Finaliza el trabajo hecho en el cajero 4 y regresa al principio de la aplicación.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

### 8.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;


public class Cajero1Activity extends AppCompatActivity {

    private TextView tvpeso;
    private TextView tvprecio;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cajero1);
```

```
        tvpeso = (TextView)findViewById(R.id.tv_3);
        tvprecio = (TextView)findViewById(R.id.tv_4);

        //Método de envio de datos del peso y el precio
        String dato3 = getIntent().getStringExtra("dato3");
        tvpeso.setText("" + dato3);
        String dato4 = getIntent().getStringExtra("dato4");
        tvprecio.setText("" + dato4);
    }

    //Método para el botón finalizar
    public void Finalizar (View view) {
        Intent finalizar = new Intent (this, MainActivity.class);
        startActivity(finalizar);
    }

    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, Empleado2Activity.class);
        startActivity(volver);
    }
}
```

8.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Cajero1Activity">

    <TextView
        android:id="@+id/tv_2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_2"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/et_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="45dp"
        android:ems="10"
        android:hint="@string/et_1"
        android:inputType="textPersonName"
        android:textSize="24sp"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tv_2" />

    <TextView
        android:id="@+id/tv_3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_3"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/et_1" />

    <TextView
        android:id="@+id/tv_4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginLeft="50dp"
        android:layout_marginTop="28dp"
        android:text="@string/tv_4"
        android:textSize="24sp"
        app:layout_constraintStart_toEndOf="@+id/tv_3"
        app:layout_constraintTop_toBottomOf="@+id/et_1" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent">

        <Button
            android:id="@+id/b_finalizar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_finalizar"
            android:textSize="24sp" />

        <ImageButton
            android:id="@+id/b_volver4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:contentDescription="@string/b_volver4"
            app:srcCompat="@android:drawable/ic_menu_revert" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 9. INTERFAZ CLIENTE

En esta interfaz, nos encontramos en la situación siguiente:

- Regresamos al inicio y el cliente escoge la opción **CLIENTE**.
- Visualizamos la interfaz del cliente.

La interfaz contiene los siguientes comandos:

- Visor de texto: Aparece la palabra **BIENVENIDO**.
- Visor de texto: Aparece el nº del carro.
- Visor de texto: Aparece la sección en la que se encuentra el carro.
- Visor de imagen: Aparece el icono de la sección.
- Visor de texto: Aparece la frase **LOCALIZACIÓN ACTUAL**.
- Mapa: Imagen con la localización actual del carro.
- Botón **SECCIONES**: Habilita la interfaz de las secciones del supermercado.
- Botón **VER LISTA**: Habilita la interfaz en la que se encuentra la lista de la compra que se está realizando.
- Visor de texto: Aparece el peso de la compra.
- Visor de texto: Aparece el precio de la compra.
- Botón **FINALIZAR**: Finaliza el trabajo hecho en el cajero 1 y regresa al principio de la aplicación.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

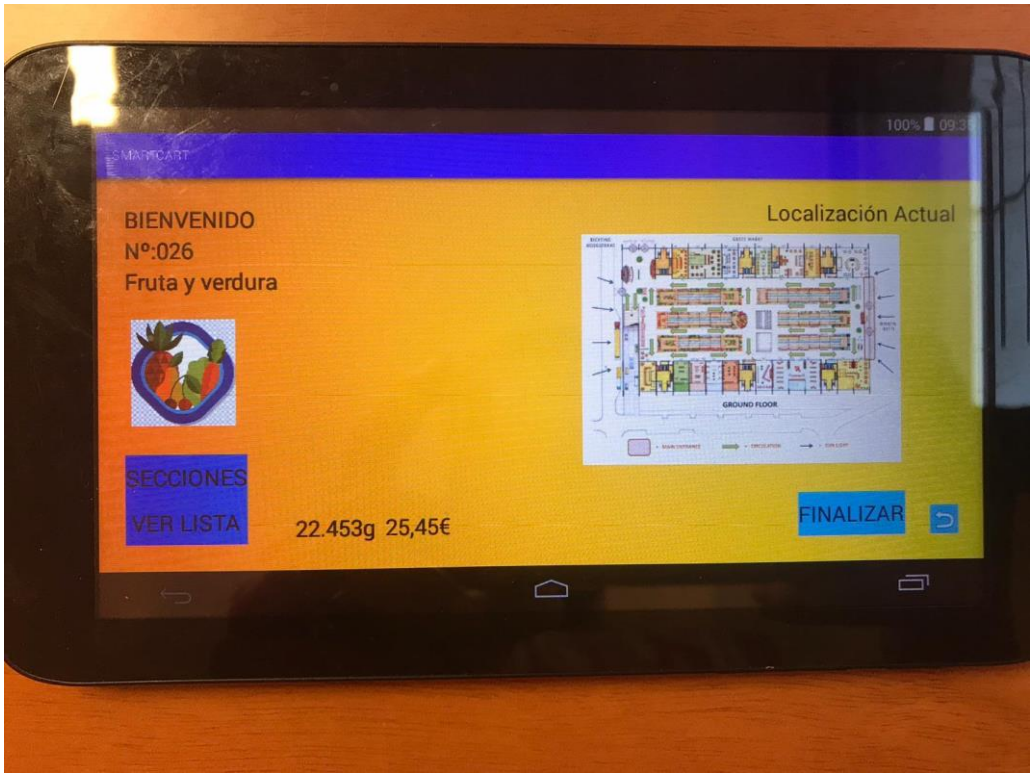



*Imagen 5: Interfaz CLIENTE*

9.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class ClienteActivity extends AppCompatActivity {

    private TextView tvpeso;
    private TextView tvprecio;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cliente);

        tvpeso = (TextView)findViewById(R.id.tv_7);
        tvprecio = (TextView)findViewById(R.id.tv_8);

        //Método de envio de datos del peso y el precio
        String dato = getIntent().getStringExtra("dato");
        tvpeso.setText("" + dato);
        String dato2 = getIntent().getStringExtra("dato2");
        tvprecio.setText("" + dato2);
    }

    //Método para el botón finalizar
    public void Fin (View view) {
        Intent fin = new Intent (this, EsperaActivity.class);
        fin.putExtra("dato3", tvpeso.getText().toString());
        fin.putExtra("dato4", tvprecio.getText().toString());
        startActivity(fin);
    }

    //Método para el botón secciones
    public void Secciones (View view) {
        Intent secciones = new Intent (this, SeccionesActivity.class);
        startActivity(secciones);
    }

    //Método para el botón ver lista
    public void Lista (View view) {
        Intent lista = new Intent (this, ListaActivity.class);
        startActivity(lista);
    }

    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, MainActivity.class);
        startActivity(volver);
    }
}
```

9.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ClienteActivity">

    <ImageButton
        android:id="@+id/b_volver2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        android:contentDescription="@string/b_volver2"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_menu_revert" />

    <ImageView
        android:id="@+id/i_plano"
        android:layout_width="480dp"
        android:layout_height="251dp"
        android:layout_marginTop="60dp"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:contentDescription="@string/i_plano"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@mipmap/plano" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:text="@string/tv_plano"
        android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginBottom="28dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent">
```

```
    <Button
        android:id="@+id/b_secciones"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/b_secciones"
        android:textSize="24sp" />

    <Button
        android:id="@+id/b_lista"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/b_lista"
        android:textSize="24sp" />
</LinearLayout>

<LinearLayout
    android:id="@+id/linearLayout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="28dp"
    android:layout_marginLeft="28dp"
    android:layout_marginTop="28dp"
    android:orientation="vertical"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:id="@+id/tv_bienvenido"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/tv_bienvenido"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/tv_carro"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/tv_carro"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/tv_seccion_actual"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/tv_seccion_actual"
        android:textSize="24sp" />
</LinearLayout>

<ImageView
    android:id="@+id/i_fruta_verdura"
    android:layout_width="109dp"
    android:layout_height="97dp"
    android:layout_marginTop="29dp"
    android:layout_marginBottom="29dp"
    android:contentDescription="@string/i_fruta_verdura"
    app:layout_constraintBottom_toTopOf="@+id/linearLayout2"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout"
    app:srcCompat="@mipmap/icono_fruta_verdura"
```

```
        tools:layout_editor_absoluteX="48dp" />

    <Button
        android:id="@+id/b_fin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        android:text="@string/b_fin"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/b_volver2" />

    <TextView
        android:id="@+id/tv_7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginLeft="50dp"
        android:layout_marginBottom="28dp"
        android:text="@string/tv_7"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/linearLayout2" />

    <TextView
        android:id="@+id/tv_8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginLeft="10dp"
        android:layout_marginBottom="28dp"
        android:text="@string/tv_8"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/tv_7" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

### 10. INTERFAZ SECCIONES

En esta interfaz, nos encontramos en la situación siguiente:

- El cliente quiere visualizar los productos que hay en el supermercado.
- Pulsa el botón **SECCIONES** y encuentra las 5 opciones que hay.
- En cada una de estas secciones hay una variedad de productos.

La interfaz contiene los siguientes comandos:

- Botón **FRUTA Y VERDURA**: Habilita la interfaz de frutas y verduras.
- Botón **CONGELADOS**: Habilita la interfaz de congelados.
- Botón **CHARCUTERÍA**: Habilita la interfaz de charcutería.
- Botón **LICORES**: Habilita la interfaz de licores.
- Botón **DROGUERÍA**: Habilita la interfaz de droguería.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

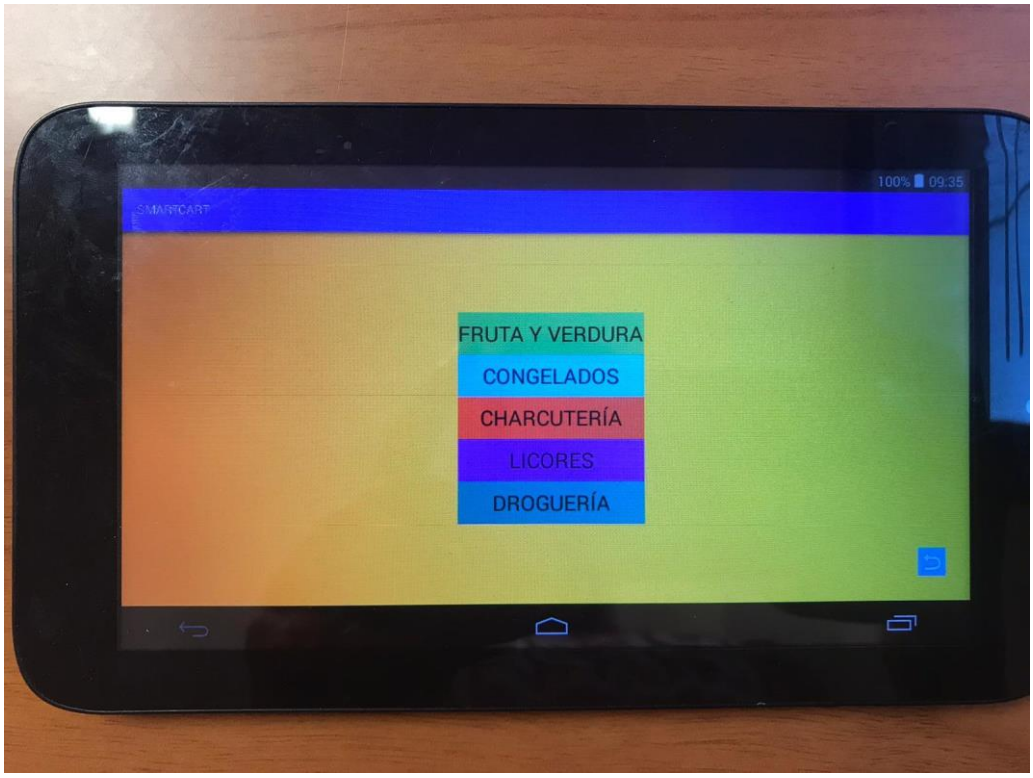



*Imagen 6: Interfaz SECCIONES*

10.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class SeccionesActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_secciones);
    }

    //Método para el botón fruta y verdura
    public void FrutaVerdura (View view) {
        Intent fruta = new Intent (this, Seccion_F_VActivity.class);
        startActivity(fruta);
    }

    //Método para el botón congelados
    public void Congelados (View view) {
        Intent congelados = new Intent (this, CongeladosActivity.class);
        startActivity(congelados);
    }
```

```
    //Método para el botón charcuteria
    public void Charcuteria (View view) {
        Intent charcuteria = new Intent (this, CharcuteriaActivity.class);
        startActivity(charcuteria);
    }

    //Método para el botón licores
    public void Licores (View view) {
        Intent licores = new Intent (this, LicoresActivity.class);
        startActivity(licores);
    }

    //Método para el botón drogueria
    public void Drogueria (View view) {
        Intent drogueria = new Intent (this, DrogueriaActivity.class);
        startActivity(drogueria);
    }

    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, ClienteActivity.class);
        startActivity(volver);
    }
}
```

10.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SeccionesActivity">

    <ImageButton
        android:id="@+id/b_volver6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        android:contentDescription="@string/b_volver6"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_menu_revert" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
```

```
        <Button
            android:id="@+id/b_fv"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_fv"
            android:textSize="24sp" />

        <Button
            android:id="@+id/b_con"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_con"
            android:textSize="24sp" />

        <Button
            android:id="@+id/b_ch"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_ch"
            android:textSize="24sp" />

        <Button
            android:id="@+id/b_lic"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_lic"
            android:textSize="24sp" />

        <Button
            android:id="@+id/b_dr"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/b_dr"
            android:textSize="24sp" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 11. INTERFAZ FRUTA Y VERDURA

En esta interfaz, nos encontramos en la situación siguiente:

- El cliente quiere visualizar los productos de fruta y verdura que hay en el supermercado.
- Pulsando cada producto, aparece información relacionada con él.

La interfaz contiene los siguientes comandos:

- Botones de frutas.
- Botones de verduras.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

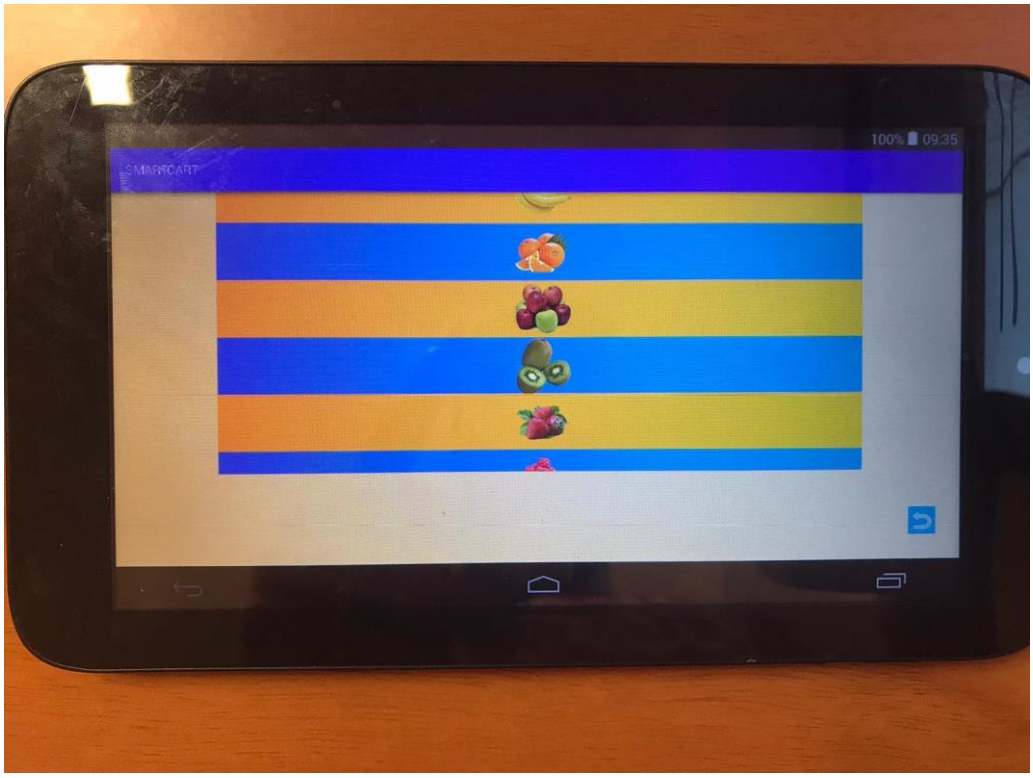



*Imagen 7: Interfaz FRUTA Y VERDURA*

### 11.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class Seccion_F_VActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_seccion__f__v);
    }

    public void Seleccion (View view) {
        switch (view.getId()) {
            case R.id.abrillantador:
              Toast.makeText(this,"Abrillantador de muebles",
Toast.LENGTH_LONG).show();
              break;
            case R.id.estropajo:
                Toast.makeText(this,"Estropajo verde", Toast.LENGTH_LONG).show();
                break;
            case R.id.balleta:
                Toast.makeText(this,"Pack de 6 Balletas",
```

```
Toast.LENGTH_LONG).show();
                break;
            case R.id.lejia:
                Toast.makeText(this,"Lejía", Toast.LENGTH_LONG).show();
                break;
            case R.id.guantes:
                Toast.makeText(this,"Guantes de látex", Toast.LENGTH_LONG).show();
                break;
            case R.id.cristales:
                Toast.makeText(this,"Limpia cristales", Toast.LENGTH_LONG).show();
                break;
            case R.id.muebles:
                Toast.makeText(this,"Limpia muebles", Toast.LENGTH_LONG).show();
                break;

        }
    }

    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, SeccionesActivity.class);
        startActivity(volver);
    }
}
```

11.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#00000000"
    tools:context=".Seccion_F_VActivity">

    <ScrollView
        android:layout_width="729dp"
        android:layout_height="409dp"
        android:layout_marginBottom="200dp"
        android:background="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <ImageButton
                android:id="@+id/naranja"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:background="#2196F3"
                android:contentDescription="@string/naranja"
```

```
            android:onClick="Seleccion"
            app:srcCompat="@drawable/naranjas" />

        <ImageButton
            android:id="@+id/manzana"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/manzana"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/manzanas" />

        <ImageButton
            android:id="@+id/kiwi"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/kiwi"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/kiwis" />

        <ImageButton
            android:id="@+id/fresa"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/fresa"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/fresas" />

        <ImageButton
            android:id="@+id/frambuesa"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/frambuesa"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/frambuesas" />

        <ImageButton
            android:id="@+id/cereza"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/cereza"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/cerezas" />

        <ImageButton
            android:id="@+id/banana"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/banana"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/bananas" />
    </LinearLayout>
</ScrollView>
```

```
    <ImageButton
        android:id="@+id/b_volver7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_menu_revert" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 12. INTERFAZ CONGELADOS

En esta interfaz, nos encontramos en la situación siguiente:

- El cliente quiere visualizar los productos de congelados que hay en el supermercado.
- Pulsando cada producto, aparece información relacionada con él.

La interfaz contiene los siguientes comandos:

- Botones de productos congelados.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

### 12.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class Seccion_F_VActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_seccion__f__v);
    }

    public void Seleccion (View view) {
        switch (view.getId()) {
            case R.id.limon:
              Toast.makeText(this,"Helados de limón", Toast.LENGTH_LONG).show();
              break;
            case R.id.naranja:
                Toast.makeText(this,"Helados de naranja",
Toast.LENGTH_LONG).show();
                break;
            case R.id.hielos:
                Toast.makeText(this,"Bolsa de hielos", Toast.LENGTH_LONG).show();
                break;
```

```
            case R.id.pizza:
                Toast.makeText(this,"pizza 4 quesos", Toast.LENGTH_LONG).show();
                break;
            case R.id.judias:
                Toast.makeText(this,"judias verdes", Toast.LENGTH_LONG).show();
                break;
            case R.id.croquetas:
                Toast.makeText(this,"Croquetas de carne",
Toast.LENGTH_LONG).show();
                break;
            case R.id.guisantes:
                Toast.makeText(this,"Guisantes pequeños",
Toast.LENGTH_LONG).show();
                break;

        }
    }
    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, SeccionesActivity.class);
        startActivity(volver);
    }
}
```

12.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#00000000"
    tools:context=".Seccion_F_VActivity">

    <ScrollView
        android:layout_width="729dp"
        android:layout_height="409dp"
        android:layout_marginBottom="200dp"
        android:background="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <ImageButton
                android:id="@+id/naranja"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:background="#2196F3"
                android:contentDescription="@string/cereza"
                app:srcCompat="@drawable/cerezas" />
```

```
        <ImageButton
            android:id="@+id/limon"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/banana"
            app:srcCompat="@drawable/bananas" />

        <ImageButton
            android:id="@+id/guisantes"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/naranja"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/naranjas" />

        <ImageButton
            android:id="@+id/croquetas"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/manzana"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/manzanas" />

        <ImageButton
            android:id="@+id/judias"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/kiwi"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/kiwis" />

        <ImageButton
            android:id="@+id/pizza"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/fresa"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/fresas" />

        <ImageButton
            android:id="@+id/hielos"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/frambuesa"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/frambuesas" />

    </LinearLayout>
</ScrollView>

<ImageButton
    android:id="@+id/b_volver7"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_menu_revert" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 13. INTERFAZ CHARCUTERÍA

En esta interfaz, nos encontramos en la situación siguiente:

- El cliente quiere visualizar los productos de charcutería que hay en el supermercado.
- Pulsando cada producto, aparece información relacionada con él.

La interfaz contiene los siguientes comandos:

- Botones de productos de charcutería.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

### 13.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class Seccion_F_VActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_seccion__f__v);
    }

    public void Seleccion (View view) {
        switch (view.getId()) {
            case R.id.longaniza:
              Toast.makeText(this,"Longanizas", Toast.LENGTH_LONG).show();
              break;
            case R.id.morcilla:
                Toast.makeText(this,"Morcilla de cebolla",
Toast.LENGTH_LONG).show();
                break;
            case R.id.solomillo:
                Toast.makeText(this,"Solomillo de cerdo",
Toast.LENGTH_LONG).show();
                break;
            case R.id.pavo:
```

```
                Toast.makeText(this,"Pechugas de pavo", Toast.LENGTH_LONG).show();
                break;
            case R.id.jamon:
                Toast.makeText(this,"Jamón Ibérico", Toast.LENGTH_LONG).show();
                break;
            case R.id.huevos:
                Toast.makeText(this,"Huevos grandes", Toast.LENGTH_LONG).show();
                break;
            case R.id.chorizo:
                Toast.makeText(this,"Chorizo picante", Toast.LENGTH_LONG).show();
                break;

        }
    }
    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, SeccionesActivity.class);
        startActivity(volver);
    }
}
```

13.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#00000000"
    tools:context=".Seccion_F_VActivity">

    <ScrollView
        android:layout_width="729dp"
        android:layout_height="409dp"
        android:layout_marginBottom="200dp"
        android:background="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <ImageButton
                android:id="@+id/morcilla"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:background="#2196F3"
                android:contentDescription="@string/cereza"
                app:srcCompat="@drawable/cerezas" />

            <ImageButton
                android:id="@+id/longaniza"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/banana"
            app:srcCompat="@drawable/bananas" />

        <ImageButton
            android:id="@+id/chorizo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/naranja"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/naranjas" />

        <ImageButton
            android:id="@+id/huevos"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/manzana"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/manzanas" />

        <ImageButton
            android:id="@+id/jamon"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/kiwi"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/kiwis" />

        <ImageButton
            android:id="@+id/pavo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/fresa"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/fresas" />

        <ImageButton
            android:id="@+id/solomillo"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/frambuesa"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/frambuesas" />

    </LinearLayout>
</ScrollView>

<ImageButton
    android:id="@+id/b_volver7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="28dp"
```

```
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_menu_revert" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 14. INTERFAZ LICORES

En esta interfaz, nos encontramos en la situación siguiente:

- El cliente quiere visualizar los productos de licores que hay en el supermercado.
- Pulsando cada producto, aparece información relacionada con él.

La interfaz contiene los siguientes comandos:

- Botones de productos de licores.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

### 14.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class Seccion_F_VActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_seccion__f__v);
    }

    public void Seleccion (View view) {
        switch (view.getId()) {
            case R.id.ginebra:
              Toast.makeText(this,"Ginebra Larios", Toast.LENGTH_LONG).show();
              break;
            case R.id.vodka:
                Toast.makeText(this,"Vodka de arándanos",
Toast.LENGTH_LONG).show();
                break;
            case R.id.blanco:
                Toast.makeText(this,"Vino Blanco", Toast.LENGTH_LONG).show();
                break;
            case R.id.tinto:
                Toast.makeText(this,"Vino Tinto", Toast.LENGTH_LONG).show();
                break;
```

```
            case R.id.rosado:
                Toast.makeText(this,"Vino Rosado", Toast.LENGTH_LONG).show();
                break;
            case R.id.sidra:
                Toast.makeText(this,"Sidra", Toast.LENGTH_LONG).show();
                break;
            case R.id.whisky:
                Toast.makeText(this,"Whisky Ballantines",
Toast.LENGTH_LONG).show();
                break;

        }
    }
    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, SeccionesActivity.class);
        startActivity(volver);
    }
}
```

14.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#00000000"
    tools:context=".Seccion_F_VActivity">

    <ScrollView
        android:layout_width="729dp"
        android:layout_height="409dp"
        android:layout_marginBottom="200dp"
        android:background="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <ImageButton
                android:id="@+id/vodka"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:background="#2196F3"
                android:contentDescription="@string/cereza"
                app:srcCompat="@drawable/cerezas" />

            <ImageButton
```

```
            android:id="@+id/ginebra"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/banana"
            app:srcCompat="@drawable/bananas" />

        <ImageButton
            android:id="@+id/whisky"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/naranja"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/naranjas" />

        <ImageButton
            android:id="@+id/sidra"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/manzana"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/manzanas" />

        <ImageButton
            android:id="@+id/rosado"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/kiwi"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/kiwis" />

        <ImageButton
            android:id="@+id/tinto"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/fresa"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/fresas" />

        <ImageButton
            android:id="@+id/blanco"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/frambuesa"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/frambuesas" />

    </LinearLayout>
</ScrollView>

<ImageButton
    android:id="@+id/b_volver7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_menu_revert" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 15. INTERFAZ DROGUERÍA

En esta interfaz, nos encontramos en la situación siguiente:

- El cliente quiere visualizar los productos de droguería que hay en el supermercado.
- Pulsando cada producto, aparece información relacionada con él.

La interfaz contiene los siguientes comandos:

- Botones de productos de droguería.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

### 15.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class Seccion_F_VActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_seccion__f__v);
    }

    public void Seleccion (View view) {
        switch (view.getId()) {
            case R.id.abrillantador:
              Toast.makeText(this,"Abrillantador de muebles",
Toast.LENGTH_LONG).show();
              break;
            case R.id.estropajo:
                Toast.makeText(this,"Estropajo verde", Toast.LENGTH_LONG).show();
                break;
            case R.id.balleta:
                Toast.makeText(this,"Pack de 6 Balletas",
Toast.LENGTH_LONG).show();
                break;
            case R.id.lejia:
                Toast.makeText(this,"Lejía", Toast.LENGTH_LONG).show();
                break;
```

```
            case R.id.guantes:
                Toast.makeText(this,"Guantes de látex", Toast.LENGTH_LONG).show();
                break;
            case R.id.cristales:
                Toast.makeText(this,"Limpia cristales", Toast.LENGTH_LONG).show();
                break;
            case R.id.muebles:
                Toast.makeText(this,"Limpia muebles", Toast.LENGTH_LONG).show();
                break;

        }
    }
    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, SeccionesActivity.class);
        startActivity(volver);
    }
}
```

15.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#00000000"
    tools:context=".Seccion_F_VActivity">

    <ScrollView
        android:layout_width="729dp"
        android:layout_height="409dp"
        android:layout_marginBottom="200dp"
        android:background="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <ImageButton
                android:id="@+id/estropajo"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:background="#2196F3"
                android:contentDescription="@string/cereza"
                app:srcCompat="@drawable/cerezas" />

            <ImageButton
                android:id="@+id/abrillantador"
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/banana"
            app:srcCompat="@drawable/bananas" />

        <ImageButton
            android:id="@+id/muebles"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/naranja"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/naranjas" />

        <ImageButton
            android:id="@+id/cristales"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/manzana"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/manzanas" />

        <ImageButton
            android:id="@+id/guantes"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/kiwi"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/kiwis" />

        <ImageButton
            android:id="@+id/lejia"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/fresa"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/fresas" />

        <ImageButton
            android:id="@+id/balleta"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/frambuesa"
            android:onClick="Seleccion"
            app:srcCompat="@drawable/frambuesas" />

    </LinearLayout>
</ScrollView>

<ImageButton
    android:id="@+id/b_volver7"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="28dp"
```

```
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_menu_revert" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 16. INTERFAZ VER LISTA

En esta interfaz, nos encontramos en la situación siguiente:

- El cliente desde la interfaz **CLIENTE**, pulsa el botón **VER LISTA**.
- Dentro de la nueva interfaz, el cliente visualizará la lista de productos que estará leyendo con el lector de código de barras e incorporando en la cesta del carro.
- Visualizará el peso y el precio de la lista.
- Podrá modificar la lista en todo momento, en caso de error.

La interfaz contiene los siguientes comandos:

- Botones de productos de cualquier sección.
- Visor de Texto: Para calcular el peso de la compra.
- Visor de Texto: Para calcular el precio de la compra.
- Botón **ENVIAR**: Habilita la opción de compartir el resultado de la compra al cajero que corresponda.
- Botón **VOLVER**: Para regresar a la interfaz anterior.

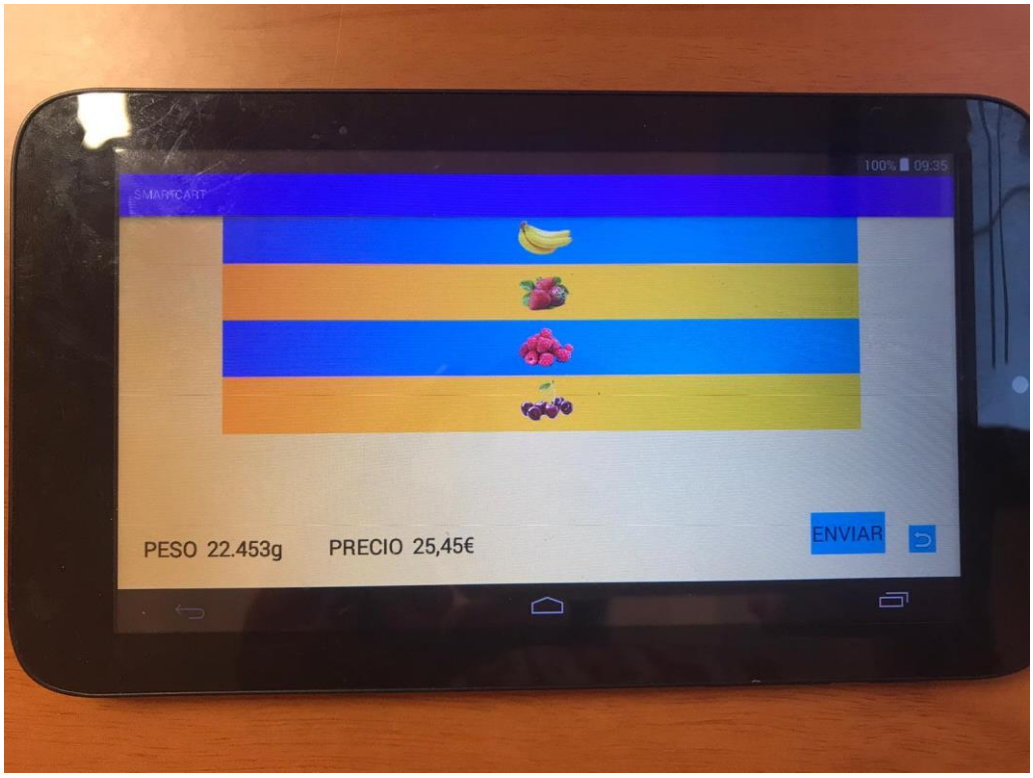



*Imagen 8: Interfaz VER LISTA*

16.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class ListaActivity extends AppCompatActivity {

    private TextView tvpeso;
    private TextView tvprecio;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lista);

        tvpeso = (TextView)findViewById(R.id.tv_peso);
        tvprecio = (TextView)findViewById(R.id.tv_precio);

    }
    //Método para el botón enviar
    public void Enviar (View view) {
        Intent enviar = new Intent (this, ClienteActivity.class);
        enviar.putExtra("dato", tvpeso.getText().toString());
        enviar.putExtra("dato2", tvprecio.getText().toString());
        startActivity(enviar);
    }
    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent (this, ClienteActivity.class);
        startActivity(volver);
    }

    public void Seleccion (View view) {
        switch (view.getId()) {
            case R.id.banana:
                Toast.makeText(this,"Banana de canarias",
Toast.LENGTH_LONG).show();
                break;
            case R.id.fresa:
                Toast.makeText(this,"Fresas", Toast.LENGTH_LONG).show();
                break;
            case R.id.frambuesa:
                Toast.makeText(this,"Frambuesas rojas", Toast.LENGTH_LONG).show();
                break;
            case R.id.cereza:
                Toast.makeText(this,"Cerezas", Toast.LENGTH_LONG).show();
                break;
        }
    }
}
```

16.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ListaActivity">

    <TextView
        android:id="@+id/tv_peso"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginLeft="10dp"
        android:layout_marginBottom="28dp"
        android:text="@string/tv_peso"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/tv_5" />

    <TextView
        android:id="@+id/tv_precio"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginLeft="10dp"
        android:layout_marginBottom="28dp"
        android:text="@string/tv_precio"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/tv_6" />

    <TextView
        android:id="@+id/tv_5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="28dp"
        android:layout_marginLeft="28dp"
        android:layout_marginBottom="28dp"
        android:text="@string/tv_5"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/tv_6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginLeft="50dp"
        android:layout_marginBottom="28dp"
        android:text="@string/tv_6"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/tv_peso" />
```

```
<ImageButton
    android:id="@+id/b_volver5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="28dp"
    android:layout_marginRight="28dp"
    android:layout_marginBottom="28dp"
    android:contentDescription="@string/b_volver5"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:srcCompat="@android:drawable/ic_menu_revert" />

<Button
    android:id="@+id/b_enviar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="28dp"
    android:layout_marginRight="28dp"
    android:layout_marginBottom="28dp"
    android:text="@string/b_enviar"
    android:textSize="24sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/b_volver5" />

<ScrollView
    android:layout_width="729dp"
    android:layout_height="330dp"
    android:layout_marginBottom="50dp"
    app:layout_constraintBottom_toTopOf="@+id/tv_precio"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <ImageButton
            android:id="@+id/banana"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
            android:contentDescription="@string/banana"
            app:srcCompat="@drawable/bananas" />

        <ImageButton
            android:id="@+id/fresa"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/fresa"
            app:srcCompat="@drawable/fresas" />

        <ImageButton
            android:id="@+id/frambuesa"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#2196F3"
```

```
            android:contentDescription="@string/frambuesa"
            app:srcCompat="@drawable/frambuesas" />

        <ImageButton
            android:id="@+id/cereza"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="#FFC107"
            android:contentDescription="@string/cereza"
            app:srcCompat="@drawable/cerezas" />

    </LinearLayout>
</ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

### 17. INTERFAZ ESPERA

En esta interfaz, nos encontramos en la situación siguiente:

- El cliente desea terminar con la compra.
- Aparece un mensaje en el que se indica a qué cajero debe acudir.
- Una vez se realice el pago, se pulsa finalizar.

La interfaz contiene los siguientes comandos:

- Visor de Texto: Aparece el mensaje que indica a qué cajero tiene que acudir.
- Botón **FINALIZAR**: Finaliza el proceso de la aplicación.


*Imagen 9: Interfaz ESPERA*

### 17.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class CompraActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_compra);
    }

    //Método para el botón finalizar
    public void Final (View view) {
        Intent final = new Intent (this, FinalActivity.class);
        Toast.makeText(this, "COMPRA FINALIZADA", Toast.LENGTH_LONG).show();
        startActivity(final);
    }
}
```

### 17.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#2196F3"
    tools:context=".CompraActivity">

    <TextView
        android:id="@+id/tv_escoge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#2196F3"
        android:text="@string/tv_escoge"
        android:textColor="#FFC107"
        android:textSize="30sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/b_final"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_marginBottom="28dp"
        android:background="#FFEB3B"
        android:text="@string/b_final"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

## 18. INTERFAZ FINAL

En esta interfaz, nos encontramos en la situación siguiente:

- El cliente ha finalizado la compra.

La interfaz contiene los siguientes comandos:

- Visor de Texto: Aparece un mensaje de despedida.
- Botón **VOLVER**: Regresa a la interfaz del principio.

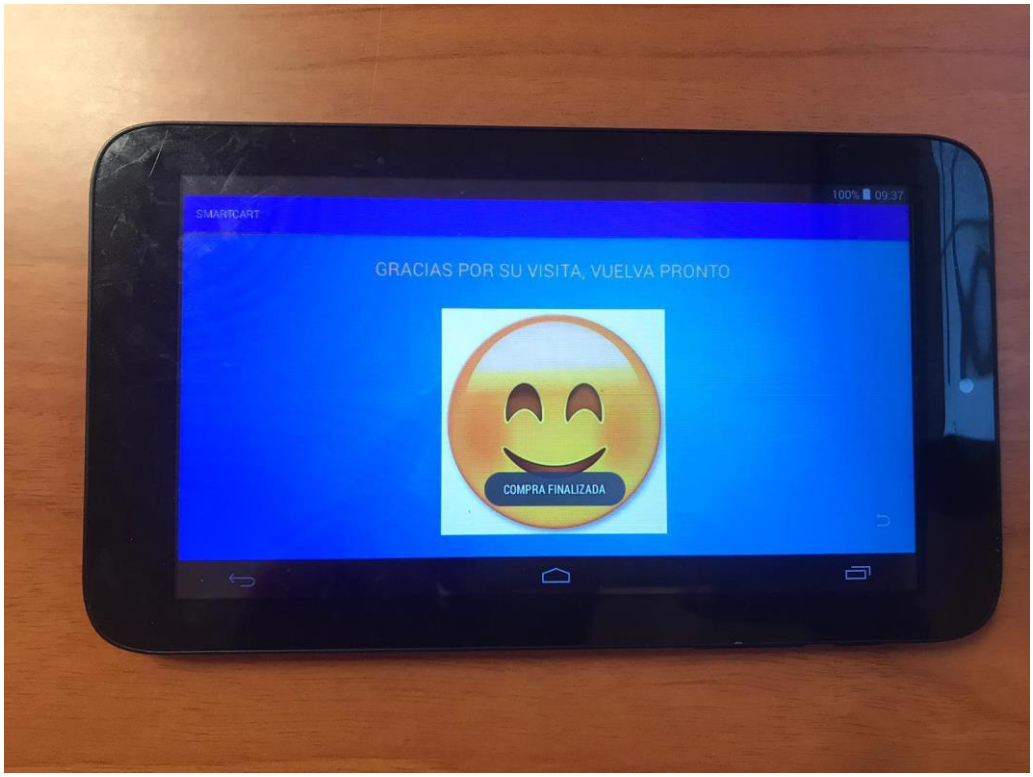



*Imagen 10: Interfaz FINAL*

### 18.1. Código Java:

```
package com.example.smartcart;

import androidx.appcompat.app.AppCompatActivity;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class FinalActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_final);
    }

    //Método para el botón volver
    public void Volver (View view) {
        Intent volver = new Intent(this, MainActivity.class);
        startActivity(volver);
    }
}
```

18.2. Código .xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#2196F3"
    tools:context=".FinalActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="380dp"
        android:layout_height="295dp"
        android:layout_marginBottom="28dp"
        android:background="#2196F3"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:srcCompat="@mipmap/emoji" />

    <TextView
        android:id="@+id/tv_final"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="28dp"
        android:text="@string/tv_final"
        android:textColor="#FFC107"
        android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
    <ImageButton
        android:id="@+id/b_volver7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="28dp"
        android:layout_marginRight="28dp"
        android:layout_marginBottom="28dp"
        android:contentDescription="@string/b_volver7"
        android:onClick="Volver"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_menu_revert" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

## 19. FUNCIONES STRINGS

Son las funciones que utilizamos para que aparezcan las escrituras en los comandos de las interfaces.

### 19.1. Código Java:

```
<resources>
    <string name="app_name">SMARTCART</string>
    <string name="b_empleado">empleado</string>
    <string name="b_cliente">cliente</string>
    <string name="b_volver">volver</string>
    <string name="et_usuario">USUARIO</string>
    <string name="et_contraseña">CONTRASEÑA</string>
    <string name="b_registro">entrar</string>
    <string name="b_volver2">volver</string>
    <string name="i_plano">plano</string>
    <string name="tv_plano">Localización Actual</string>
    <string name="tv_bienvenido">BIENVENIDO</string>
    <string name="b_secciones">secciones</string>
    <string name="b_lista">ver lista</string>
    <string name="tv_carro">Nº:026</string>
    <string name="tv_seccion_actual">Fruta y verdura</string>
    <string name="i_fruta_verdura">Fruta y verdura</string>
    <string name="tv_1">¡Bienvenido roberto!</string>
    <string name="b_cajero1">cajero 1</string>
    <string name="b_cajero2">cajero 2</string>
    <string name="b_cajero3">cajero 3</string>
    <string name="b_cajero4">cajero 4</string>
    <string name="b_volver3">volver</string>
    <string name="tv_2">TRABAJANDO CON CAJERO 1</string>
    <string name="et_1">Insertar nº del carro</string>
    <string name="tv_3">PESO</string>
    <string name="tv_4">PRECIO</string>
    <string name="b_finalizar">finalizar</string>
    <string name="b_volver4">volver</string>
    <string name="tv_peso">22.453g</string>
    <string name="tv_precio">25,45€</string>
    <string name="tv_5">PESO</string>
    <string name="tv_6">PRECIO</string>
    <string name="b_volver5">volver</string>
```

```
    <string name="b_fin">finalizar</string>
    <string name="b_enviar">enviar</string>
    <string name="tv_7">PESO</string>
    <string name="tv_8">PRECIO</string>
    <string name="b_volver6">volver</string>
    <string name="b_fv">fruta y verdura</string>
    <string name="b_con">congelados</string>
    <string name="b_ch">charcutería</string>
    <string name="b_lic">licores</string>
    <string name="b_dr">droguería</string>
    <string name="banana">descripción</string>
    <string name="cereza">descripción</string>
    <string name="frambuesa">descripción</string>
    <string name="fresa">descripción</string>
    <string name="kiwi">descripción</string>
    <string name="manzana">descripción</string>
    <string name="naranja">descripción</string>
    <string name="tv_escoge">ACUDA AL CAJERO 1 PARA FINALIZAR</string>
    <string name="b_final">finalizar</string>
    <string name="tv_final">GRACIAS POR SU VISITA, VUELVA PRONTO</string>
    <string name="b_volver7">volver</string>

</resources>
```

## 20. FUNCIONES PARA EL SISTEMA DE GEOLOCALIZACIÓN

Al no tener un simulador de ***beacons***, no podemos escribir un código que pueda interactuar con la aplicación. Sin embargo, sí conocemos las funciones que tendríamos que utilizar para poder manejarlos, gracias a la página web de https://solidgeargroup.com.

Podemos utilizar los *beacons* a través de **EDDYSTONE** (desarrollador de Google para balizas electrónicas de código abierto) para estas funciones:

- **Eddystone-UID**: Trabaja con identificadores de un *beacon*.
- **Eddystone-URL**: Trabaja enviando una URL.
- **Eddystone-TLM**: Trabaja emitiendo con los paquetes anteriores y contiene el estado de salud de un *beacon*, como el nivel de batería.
- **Eddystone-EID***:* Trabaja con un identificador encriptado que cambia periódicamente.

### 20.1. Código Java:

```
compile 'org.altbeacon:android-beacon-library:${altbeacon.version}'


public class RangingActivity extends Activity implements BeaconConsumer,
RangeNotifier {
    protected static final String TAG = "RangingActivity";
    private BeaconManager mBeaconManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ranging);

        mBeaconManager = BeaconManager.getInstanceForApplication(this);
```

```
        //vamos a usar el protocolo Eddystone, así que tenemos que
definirlo aquí
        mBeaconManager.getBeaconParsers().add(new BeaconParser().
                setBeaconLayout(BeaconParser.EDDYSTONE_UID_LAYOUT));

        // Bindea esta actividad al BeaconService
        mBeaconManager.bind(this);
    }
}


@Override
public void onBeaconServiceConnect() {
    // Encapsula un identificador de un beacon de una longitud arbitraria
de bytes
    ArrayList<Identifier> identifiers = new ArrayList<>();

    // Asignar null para indicar que queremos buscar cualquier beacon
    identifiers.add(null);

    // Representa un criterio de campos utilizados para buscar beacons
    Region region = new Region("AllBeaconsRegion"), identifiers);

    try {
        // Ordena al BeaconService empezar a buscar beacons que coincida
con el objeto Region pasado
        mBeaconManager.startRangingBeaconsInRegion(region);
    } catch (RemoteException e) {
        e.printStackTrace();
    }
    // Especifica una clase que debería ser llamada cada vez que
BeaconsService obtiene datos, una vez por segundo por defecto
    mBeaconManager.addRangeNotifier(this);
}


@Override
public void didRangeBeaconsInRegion(Collection<Beacon> beacons, Region
region) {
    if (beacons.size() > 0) {
        Log.i(TAG, "El primer beacon detectado se encuentra a una distancia
de "+beacons.iterator().next().getDistance()+" metros.");
    }
}
```

### 21. FUNCIONES PARA TRABAJAR CON EL LECTOR DE CÓDIGO DE BARRAS

Con el lector de código de barras, podríamos establecer la relación con las bases de datos de los productos en el *Google Sheets* y así, cada vez que leyera un producto, este aparecería en la lista del cliente de la interfaz **VER LISTA**.

Al no disponer de un simulador para comprobarlo, no podemos acoplarlo a nuestra aplicación teórica, pero sabemos cuáles son las funciones con las que tendríamos que trabajar gracias a la plataforma ***developers*** de Android en la que nos indican los ejemplos. Estas funciones son las siguientes:

Obtenemos el objeto ***UsbManager***:

```
UsbManager manager = UsbManager.getInstance(this);
```

Si no utilizamos la biblioteca, es así:

```
UsbManager manager = (UsbManager)
getSystemService(Context.USB_SERVICE);
```

Obtenemos el objeto ***UsbAccessory***:

```
UsbAccessory accessory = (UsbAccessory)
intent.getParcelableExtra(UsbManager.EXTRA_ACCESSORY);
```

El Código quedaría de esta forma:

```
<manifest ...>
        <uses-feature android:name="android.hardware.usb.accessory" />

        <uses-sdk android:minSdkVersion="<version>" />
        ...
        <application>
          <uses-library android:name="com.android.future.usb.accessory"
/>
            <activity ...>
                ...
                <intent-filter>
                    <action
android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED" />
                </intent-filter>

                <meta-data
android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
                    android:resource="@xml/accessory_filter" />
            </activity>
        </application>
    </manifest>
```

El archivo debe guardarse y filtrar todos los datos:

```
<?xml version="1.0" encoding="utf-8"?>

    <resources>
        <usb-accessory model="DemoKit" manufacturer="Google"
version="1.0"/>
    </resources>
```

Una vez guardado, ya se puede trabajar con él:

```
<activity ...>
        ...
        <intent-filter>
            <action
android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED" />
        </intent-filter>

        <meta-data
android:name="android.hardware.usb.action.USB_ACCESSORY_ATTACHED"
            android:resource="@xml/accessory_filter" />
    </activity>
```

Luego declaramos el archivo:

```
<?xml version="1.0" encoding="utf-8"?>

    <resources>
        <usb-accessory manufacturer="Google, Inc." model="DemoKit"
version="1.0" />
    </resources>
```

Obtenemos el objeto de la Activity:

```
UsbAccessory accessory = UsbManager.getAccessory(intent);
```

Con este comando podemos enumerar los accesorios:

```
UsbManager manager = (UsbManager)
getSystemService(Context.USB_SERVICE);
    UsbAccessory[] accessoryList = manager.getAccessoryList();
```

Para obtener permiso para establecer una comunicación con el accesorio:

```
private static final String ACTION_USB_PERMISSION =
        "com.android.example.USB_PERMISSION";
    private final BroadcastReceiver usbReceiver = new
BroadcastReceiver() {

        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if (ACTION_USB_PERMISSION.equals(action)) {
                synchronized (this) {
                    UsbAccessory accessory = (UsbAccessory)
intent.getParcelableExtra(UsbManager.EXTRA_ACCESSORY);
```

```
                    if
(intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)) {
                        if(accessory != null){
                            //call method to set up accessory
communication
                        }
                    }
                    else {
                        Log.d(TAG, "permission denied for accessory " +
accessory);
                    }
                }
            }
        }
    };
```

Registramos el receptor de emisión:

```
UsbManager usbManager = (UsbManager)
getSystemService(Context.USB_SERVICE);
    private static final String ACTION_USB_PERMISSION =
        "com.android.example.USB_PERMISSION";
    ...
    permissionIntent = PendingIntent.getBroadcast(this, 0, new
Intent(ACTION_USB_PERMISSION), 0);
    IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
    registerReceiver(usbReceiver, filter);
```

Para que aparezca el diálogo:

```
UsbAccessory accessory;
    ...
    usbManager.requestPermission(accessory, permissionIntent);
```

Con el siguiente Código, establecemos una comunicación con el accesorio:

```
UsbAccessory accessory;
    ParcelFileDescriptor fileDescriptor;
    FileInputStream inputStream;
    FileOutputStream outputStream;
    ...

    private void openAccessory() {
        Log.d(TAG, "openAccessory: " + accessory);
        fileDescriptor = usbManager.openAccessory(accessory);
        if (fileDescriptor != null) {
```

```
            FileDescriptor fd = fileDescriptor.getFileDescriptor();
            inputStream = new FileInputStream(fd);
            outputStream = new FileOutputStream(fd);
            Thread thread = new Thread(null, this, "AccessoryThread");
            thread.start();
        }
    }
```

Y con esta función finalizamos la comunicación:

```
BroadcastReceiver usbReceiver = new BroadcastReceiver() {
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();

            if
(UsbManager.ACTION_USB_ACCESSORY_DETACHED.equals(action)) {
                UsbAccessory accessory =
(UsbAccessory)intent.getParcelableExtra(UsbManager.EXTRA_ACCESSORY);
                if (accessory != null) {
                    // call your method that cleans up and closes
communication with the accessory
                }
            }
        }
    };
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

# ANEXO Nº3

## CÓDIGO GOOGLE APPS SCRIPTS

GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

26 DE AGOSTO DE 2020

AUTOR: IGNACIO PAYÁ SILLA
TUTOR: PASCUAL PÉREZ BLASCO

**ÍNDICE**



**ÍNDICE IMÁGENES**

## 1. INTRODUCCIÓN

Las bases de datos se han realizado con la plataforma ***Google Sheets***, habilitada desde *Google Drive*, siendo una herramienta que permite diseñar hojas de datos, hojas de cálculo y tablas estructuradas con total libertad.

No obstante, para poder establecer una relación con las bases de datos diseñadas en Google Sheets y la aplicación móvil en Android Studio, es necesaria la plataforma ***Google Apps Scripts*** que, gracias a su formato de código libre, posibilita la opción de que todo el mundo pueda utilizarla.

Como se describió en la memoria, son herramientas complejas de manejar y requieren mucho trabajo y dedicación para completar su diseño lógico, pero al ser programas gratuitos (al contrario que otros de mejor calidad), condicionan favorablemente su demanda.

A continuación, se van a exponer los pasos que han sido necesarios para completar la tarea. Por un lado, tenemos el diseño de las hojas de datos en Google Sheets (podemos encontrarlos en el documento **MEMORIA**), y por el otro, el código escrito en Google Apps Scripts que establecerá la relación entre bases de datos y aplicación móvil.

## 2. FUNCIONES Y VARIABLES

Un dato importante para entender la estructura del código es conocer las funciones que vamos a necesitar. Por ejemplo:

- Función ***doGet***: Esta función sirve para leer desde la aplicación los datos de las bases de datos del Google Sheets.
- Función ***doPost***: Esta función sirve para leer y realizar cambios en las bases de datos desde la aplicación.

En este proyecto, tratándose de un trabajo teórico, no tenemos la opción de poder utilizar las bases de datos con el lector de código de barras. Por tanto, el único recurso que podemos exprimir es el de leer las bases de datos desde la aplicación, con lo que solo podremos trabajar con el *doGet*.

Con respecto a las variables, hay que indicar en el código a qué hojas de datos tiene que acudir la aplicación. Por este motivo, mediante la URL, llamamos a la base de datos del Google sheets. Luego, tenemos que indicar cuáles de las hojas que hay dentro del documento vamos a utilizar. En este caso son todas y son las siguientes:

- Tabla "**empleados**": Contiene la información de los empleados del supermercado.
- Tabla "**fruta y verdura**": Contiene un ejemplo de frutas y verduras que venden en el supermercado, además de información relevante de cada producto.
- Tabla "**congelados**": Contiene un ejemplo de congelados que venden en el supermercado, además de información relevante de cada producto.
- Tabla "**charcutería**": Contiene un ejemplo de charcutería que venden en el supermercado, además de información relevante de cada producto.
- Tabla "**licores**": Contiene un ejemplo de licores que venden en el supermercado, además de información relevante de cada producto.
- Tabla "**droguería**": Contiene un ejemplo de artículos de droguería que venden en el supermercado, además de información relevante de cada producto.

## 3. CÓDIGO JAVA EN GOOGLE APPS SCRIPTS

En las siguientes líneas de código podemos comprobar cómo se escribiría todo lo mencionado en el apartado anterior:

```
var ss = SpreadsheetApp.openByUrl("https://docs.google.com/spreadsheets/d/1GCYAEjtgXxt7TmJ8QkUsZR3Q4G6wzQqBDYfOG15wsVk/edit#gid=0");
```

*Imagen 1: Llamada a la URL de la base de datos del Google Sheets*

```
var empleados = ss.getSheetByName('empleados');
var fruta_verdura = ss.getSheetByName('fruta y verdura');
var congelados = ss.getSheetByName('congelados');
var charcuteria = ss.getSheetByName('charcutería');
var licores = ss.getSheetByName('licores');
var drogueria = ss.getSheetByName('droguería');
```

*Imagen 2: Llamada a las hojas de la base de datos*

En la siguiente imagen vemos cómo se completaría la función *doGet*:

```
function doGet (valor) {
  var tarea = valor.parameter.action;

  if(tarea == 'leerUsuario')return leer_usuario(valor);
  if(tarea == 'leerFrutaVerdura')return leer_fruta_verdura(valor);
  if(tarea == 'leerCongelados')return leer_congelados(valor);
  if(tarea == 'leerCharcuteria')return leer_charcuteria(valor);
  if(tarea == 'leerLicores')return leer_licores(valor);
  if(tarea == 'leerDrogueria')return leer_drogueria(valor);

}
```

*Imagen 3: Función doGet*

Se crea la variable "**tarea**" para leer los parámetros de la función *doGet*. Dichos parámetros se devuelven en una serie de funciones en las que se diseñará su propósito, que en este caso es leer las tablas que tenemos en el documento de la base de datos en el Google Sheet.

Para este proyecto teórico solo va a ser necesaria la función ***leer_usuario***, debido a la imposibilidad de simular el lector de código de barras. Como no se pueden analizar los productos que entran al carro, no podemos establecer una conexión entre la aplicación y las tablas de productos. Sin embargo, el formato del código escrito resultante de lectura sería el mismo que vamos a mostrar ahora:

```
function leer_usuario(request){
  var records={};
  var u = personas.getLastRow();
  var data = [];
  for (var i=0, i<=u; i++){
    record = {};
    record['ID'] = empleados.getRange(i,1).getValue();
    record['USUARIO'] = empleados.getRange(i,2).getValue();
    record['CONTRASEÑA'] = empleados.getRange(i,3).getValue();
    data.push(record);
  }
  records.info = data;
  var result = JSON.stringify(records);
  return ContentService.createTextOutput(result).setMimeType(ContentService.MimeType.JSON);
}
```

*Imagen 4: Función leer_usuario*

En esta función leemos los datos que se encuentran en la hoja de datos "**empleados**". Recordemos que esta hoja de datos tenía como misión almacenar los usuarios y sus respectivas contraseñas de los empleados del supermercado.

La aplicación llamará a esta hoja de datos en el momento en el que un empleado se adentre en la **interfaz de empleado** y trate de iniciar sesión para poder trabajar con el resto de la aplicación. Este paso, simplemente tiene cabida para la lectura de datos, porque no se podrán realizar cambios desde la aplicación sobre la base de datos del Google Sheets.

## 4. POSIBLES FUNCIONES CON EL "DOPOST"

Antes mencionaba que la utilidad de la función *doPost* es la de permitir modificar las bases de datos del Google Sheets desde la aplicación móvil de Android Studio. Sin embargo, comentaba que, al no tener la opción de simular la utilidad del lector de código de barras, no iba a ser necesario utilizar el recurso del *doPost*.

De todas maneras, en caso de tener la opción, la escritura del *doPost* se completaría de la siguiente manera:

```
function doPost(valor){
  var tarea = valor.parameter.action;

  if(tarea == 'insertProduct')return insert_product(valor);
  if(tarea == 'deleteProduct')return delete_product(valor);
  if(tarea == 'modifyProduct')return modify_product(valor);
}
```

*Imagen 5: Función doPost*

Se crea la variable "**tarea**" para leer los parámetros de la función *doPost*. Dichos parámetros se devuelven en una serie de funciones en las que se diseñará su propósito, que en este caso es leer y

modificar las tablas que tenemos en el documento de la base de datos en el Google Sheet desde la aplicación móvil del programa Android Studio.

Las funciones que devuelve en este caso son las siguientes:

- Función "**insert_product**": Esta función tiene la misión de permitir al empleado del supermercado insertar un producto nuevo en la base de datos y, por consiguiente, en la aplicación.
- Función "**delete_product**": Esta función tiene la misión de permitir al empleado del supermercado eliminar un producto en la base de datos y, por consiguiente, en la aplicación.
- Función "**modify_product**": Esta función tiene la misión de permitir al empleado del supermercado modificar un producto en la base de datos y, por consiguiente, en la aplicación.

Una vez aclarada cuál es el cometido de cada función, se procede a su escritura lógica, como en el caso anterior de la función "**leer_usuario**", pero como en este proyecto no se necesita, no es necesario adjuntar el código.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

# ANEXO Nº4

## DATA-SHEETS

GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

26 DE AGOSTO DE 2020

AUTOR: IGNACIO PAYÁ SILLA
TUTOR: PASCUAL PÉREZ BLASCO

**ÍNDICE**

# Carro supermercado 180 l.

Carro de Hipermercado con unas dimensiones de 1.000 x 610 x 1.035, con una capacidad de 180 litros. Acabado zincado brillante. Dotado con 4 ruedas de goma pivotantes Ø 125 mm.

| | |
|---|---|
| **Referencia** | 9445 |
| **Dimensiones (mm) largo x ancho x alto** | 1.000 x 610 x 1.035 |
| **Ruedas Fijas mm** | |
| **Ruedas Giratorias mm** | 125 |
| **Bandaje** | Goma |
| **Peso Kg** | 15 |
| **Carga** | 180 |

# ARNITE

Poliéster termoplástico sobre la base de tereftalato de polietileno. Las propiedades específicas de este PET cristalino en estado virgen hacen de el un material especialmente indicado para la mecanización de piezas mecánicas de precisión que deban soportar grandes cargas y/o sujetas a desgaste. Sin embargo no es adecuado para aplicaciones con impacto.

## Principales características del ARNITE:

- ✓ Elevada resistencia mecánica, rigidez y dureza.
- ✓ Muy buena resistencia a la fluencia.
- ✓ Coeficiente de rozamiento bajo y uniforme.
- ✓ Excelente resistencia al desgaste (mejor que la de los nylon)
- ✓ Muy buena estabilidad dimensional (mejor que la del delrin)
- ✓ Muy baja absorción de humedad.
- ✓ Mejor resistencia a los ácidos que el nylon y el delrin.
- ✓ Buenas propiedades dieléctricas y de aislamiento eléctrico.

| | |
|---|---|
| Densidad | 1,44 g/cm3 |

| | |
|---|---|
| Absorción de agua hasta saturación | 0,5% |

| Propiedades térmicas | |
|---|---|
| Temperatura de fusión | 245ºC |
| Temperatura Máx. en periodos cortos | 160ºC |
| Temperatura Máx. de continuo | 110-115º C |
| Temperatura mín. de servicio | -20ºC |
| Coeficiente de dilatación térmica | 60-80x$10^{-6}$ m/(m.k.) |

| Propiedades Mecánicas | |
|---|---|
| Elongación a la rotura | 15% |
| Módulo de elasticidad | 3.500 Mpa |
| Dureza con bola | 170 N/mm2 |
| Dureza Rockwell | M96 |

**PiPO X4 Windows 10 10.1 Inch IP67 Rugged Tablet Intel Cherry Trail Z8350 4GB 64GB Waterproof**

| | |
|---|---|
| **BASIC** | |
| **Modelo** | PIPO X4 |
| **UPC** | Intel Cherry Trail Z8350 |
| **GPU** | Intel HD Gráfico (Gen8) |
| **Núcleo** | Quad Núcleo hasta 1.92GHZ |
| **Sistema** | Windows 10 |
| **Idioma** | Multi lenguaje |
| **RAM** | 4GB DDR3 |
| **ROM** | 64GB eMMC |
| **Pantalla** | 10.1 Inch IPS Pantalla capacitiva |
| **Resolución** | 1920 x 1200 |
| **Cámara** | El frente es 2.0MP trasero es 5.0MP |
| **Batería** | 3.7V/12000mAh |
| **Peso** | 1080g |
| **Talla** | 280*185*26.5mm |
| **FUNCIÓN** | |
| **WIFI** | 802.11 ac / b / g / n |
| **Bluetooth** | 4.0 |
| **OTG** | Apoyo |
| **GPS** | Apoya |
| **G-sensor** | Apoyo |
| **4G** | No Apoyo |
| **Frecuencia** | NO |
| **AUDIO Y VIDEO** | |

| | |
|---|---|
| **Micrófono** | Incorporado |
| **Altavoz** | Incorporado |
| **Audio** | MP3, WMA, WAV, OGG, FLAC, ALAC, APE, AAC, AC-3 |
| **Imagen** | BMP, JPG, GIF, PNG |
| **Vídeo** | MEPG1 / 2/4, H.263 / H.264, RMVB, WMV / VC-1, MVC, AVS, MJPEG |
| **CONECTIVIDAD** | |
| | 1 x puerto micro USB |
| | 1 x puerto USB 2.0 |
| | 1 x puerto USB 3.0 |
| | 1 x puerto Micro HD |
| | 1 x RJ45 puerto de red |
| | 1 x RS232 puerto |
| | 1 x jack de 3,5 mm DC |
| | 1 x 3.5 mm Auricular puerto |
| | 1 x ranura Micro SD (soporte máximo 128GRAMO) |
| **PAQUETE INCLUIDO** | |
| | 1 x PIPO X4 Tableta |
| | 1 x cargador(5V/3A) |
| | 1 x adaptador (el adaptador depende del país del comprador) |

# GRYPHON™ I GFE4400 2D

DATALOGIC

## ADVANCED IMAGING TECHNOLOGY

Using advanced imaging technology, the Gryphon™ GFE4400 2D engine provides a compact and easy-to-use solution for OEM use in self-service kiosks, price verifiers, ticket readers, document handling and medical laboratory applications, as well as vending machines and other automated equipment.

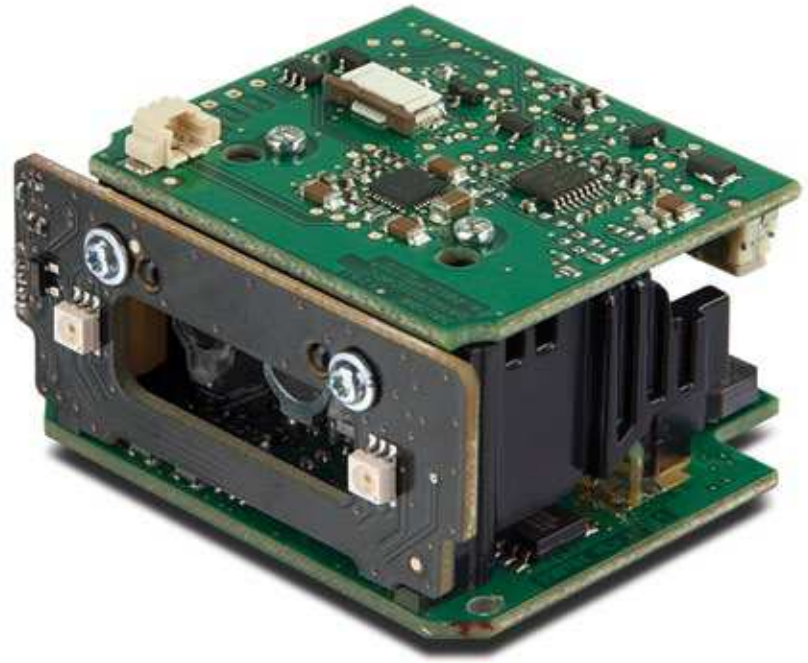
The GFE4400 2D OEM engine features outstanding near-field reading, a wide angle field-of-view, high motion tolerance, snappy reading, and also offers excellent performance on decoding poor or damaged bar codes. The GFE4400 2D OEM engine also offers good-read visual confirmation with Datalogic's patented 'Green Spot' technology.

## RED ILLUMINATING LIGHT AND 4-DOT AIMER

User comfort is maximized with the GFE4400 2D OEM engine's steady, deep red illumination light. Easier on the eyes than competitive products with flicker illumination, the highly visible 4-Dot aimer defines a precise reading zone and reduces accidental reads. The aimer's center cross provides a locator for targeted scanning in a multiple bar code environment.

## 1D AND 2D CAPABILITIES

The GFE4400 2D OEM engine provides snappy reading performance on all common 1D and 2D codes as well as postal, stacked and composite codes like PDF417. For mobile marketing or ticketing applications, this imager also offers excellent performance when reading bar codes from mobile devices.



GREEN SPOT · DATALOGIC aladdin · 4-DOT AIMER · GS1

## FEATURES

- Highly visible 4-Dot aimer with center cross for targeted scanning
- Omnidirectional reading
- Advanced motion tolerance optics
- Image capture and document scanning capabilities
- Reads bar codes down to 4 mils
- Reads 1D, 2D and postal codes plus stacked and composite codes
- Interface options: RS-232 or USB
- Datalogic's patented 'Green Spot' technology for good-read feedback
- Automatic sensing or manual trigger options available
- EASEOFCARE Service Plans offer a wide range of service options to protect your investment, ensuring maximum productivity and ROI

## INDUSTRY-APPLICATIONS

- OEM Applications:
  - Service Kiosks
  - Price Verifiers
  - Ticket Readers
  - Document Handling
  - Medical Laboratories
  - Vending Machines
  - Additional Automated Equipment

# GRYPHON™ I GFE4400 2D

DATALOGIC

TECHNICAL SPECIFICATIONS

## DECODING CAPABILITY

| | |
|---|---|
| 1D / LINEAR CODES | Autodiscriminates all standard 1D codes including GS1 DataBar™ linear codes. |
| 2D CODES | Aztec Code; China Han Xin Code; Data Matrix; MaxiCode; Micro QR Code; QR Code |
| POSTAL CODES | Australian Post; British Post; China Post; IMB; Japanese Post; KIX Post; Korea Post; Planet Code; Postnet; Royal Mail Code (RM4SCC) |
| STACKED CODES | EAN/JAN Composites; GS1 DataBar Composites; GS1 DataBar Expanded Stacked; GS1 DataBar Stacked; GS1 DataBar Stacked Omnidirectional; MacroPDF; MicroPDF417; PDF417; UPC A/E Composites |

## ELECTRICAL

| | |
|---|---|
| CURRENT | Operating (Typical): < 180 mA<br>Standby/Idle (Typical):<br>Automatic Object Sense Mode: 115 mA<br>Online & Serial OnLine Modes: 65 mA |
| INPUT VOLTAGE | 5 VDC +/- 5% |

## ENVIRONMENTAL

| | |
|---|---|
| AMBIENT LIGHT | 0 - 100,000 lux |
| HUMIDITY (NON-CONDENSING) | 5-95% |
| TEMPERATURE | Operating: -20 to 50 °C / -4 to 122 °F<br>Storage/Transport: -20 to 70 °C / -4 to 158 °F |

## INTERFACES

| | |
|---|---|
| INTERFACES | OEM (IBM) USB; RS-232; USB: USB COM; USB HID Keyboard |

## PHYSICAL CHARACTERISTICS

| | |
|---|---|
| DIMENSIONS | 2.8 x 4.2 x 4.8 cm / 1.1 x 1.7 x 1.9 in |
| WEIGHT | USB: 51.2 g / 1.8 oz<br>RS-232: 51.2 g / 1.8 oz |

## READING PERFORMANCE

| | |
|---|---|
| IMAGE CAPTURE | Graphic Formats: BMP, JPEG, TIFF;<br>Greyscale: 256, 16, 2 |
| IMAGER SENSOR | Wide VGA: 752 x 480 pixels |
| LIGHT SOURCE | Aiming: 650 nm VLD |
| PRINT CONTRAST RATIO (MINIMUM) | 25% |
| READING ANGLE | Pitch: +/- 40°; Roll (Tilt): 180°; Skew (Yaw): +/- 40° |
| RESOLUTION (MAXIMUM) | 1D Linear: 0.102 mm / 4 mils<br>Data Matrix: 0.178 mm / 7 mils<br>PDF417: 0.102 mm / 4 mils |

## READING RANGES

| | |
|---|---|
| TYPICAL DEPTH OF FIELD | Minimum distance determined by symbol length and scan angle.<br>Printing resolution, contrast, and ambient light dependent.<br>Code 39: 5 mil: 4.7 to 17.7 cm / 1.8 to 7.0 in<br>Code 39: 10 mil: 1.7 to 33.2 cm / 0.7 to 13.1 in<br>Data Matrix: 10 mil: 2.7 to 17.1 cm / 1.0 to 6.7 in<br>Data Matrix: 15 mil: 1.2 to 24.6 cm / 0.5 to 9.7 in<br>EAN: 13 mil: 2.5 to 41.9 cm / 1.0 to 16.5 in<br>PDF417: 10 mil: 2.2 to 23.9 cm / 0.9 to 9.4 in<br>QR Code: 10 mil: 3.5 to 16.0 cm / 1.4 to 6.3 in |

## SAFETY & REGULATORY

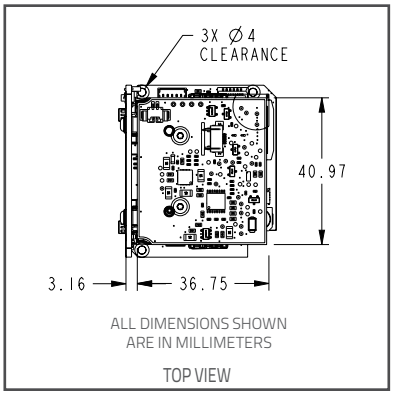
| | |
|---|---|
| AGENCY APPROVALS | The product meets necessary safety and regulatory approvals for its intended use.<br>The Quick Reference Guide for this product can be referred to for a complete list of certifications. |
| ENVIRONMENTAL COMPLIANCE | Complies to China RoHS; Complies to EU RoHS |
| LASER CLASSIFICATION | Caution Laser Radiation - Do not stare into beam; IEC 60825, Class 2 |

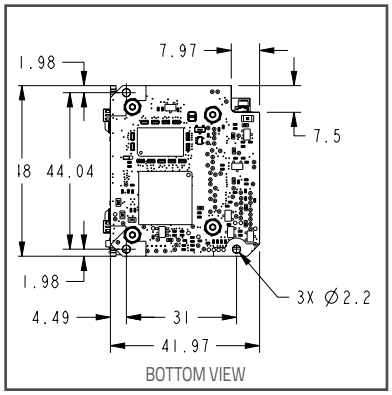
## UTILITIES

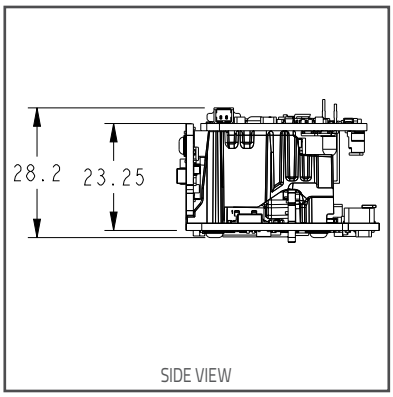
| | |
|---|---|
| DATALOGIC ALADDIN™ | Datalogic Aladdin configuration program is available for download at no charge. |
| OPOS / JAVAPOS | JavaPOS Utilities are available for download at no charge.<br>OPOS Utilities are available for download at no charge. |
| REMOTE HOST DOWNLOAD | Lowers service costs and improves operations. |

## WARRANTY

| | |
|---|---|
| WARRANTY | 18-Month Factory Warranty |





DS-GRYPHONIGFE4400-EN Revision C 20150123

# GRYPHON™ I GFS4400 2D

DATALOGIC

## SIMPLE TO INTEGRATE

Designed for easy integration, the Gryphon GFS4400 2D module provides a 'plug-and-play' solution for system designers. Constructed using high-impact resin to withstand repeated cleanings with solvents and disinfectant solutions, the sealed enclosure meets industrial standards against water and particulates to provide dependable scanning performance.

## ADVANCED IMAGING TECHNOLOGY

The GFS4400 2D OEM module features outstanding near-field reading, a wide angle field-of-view, high motion tolerance, snappy reading, and also offers excellent performance on decoding poor or damaged bar codes. As with all Gryphon readers, the GFS4400 reader offers good-read visual confirmation with Datalogic's patented 'Green Spot' technology.

## RED ILLUMINATING LIGHT AND 4-DOT AIMER

User comfort is maximized with the GFS4400 OEM module's steady, deep red illumination light. Easier on the eyes than competitive products with flicker illumination, the highly visible 4-Dot aimer defines a precise reading zone and reduces accidental reads. The aimer's center cross provides a locator for targeted scanning in a multiple bar code environment.


GREEN SPOT · DATALOGIC aladdin · GS1 · 4-DOT AIMER · DATALOGIC 5 YEAR WARRANTY

## FEATURES

- Highly visible 4-Dot aimer with center cross for targeted scanning
- Omnidirectional reading
- Advanced motion tolerance optics
- Image capture and document scanning capabilities
- Reads bar codes down to 4mils
- Reads 1D, 2D and postal codes plus stacked and composite codes
- Interface options: RS-232 or USB
- Datalogic's patented Green Spot for good-read feedback
- Integrated good-read beeper
- Automatic sensing or manual trigger options available
- High-impact resin creates a solvent-tolerant enclosure
- Water and Particulate Sealing Rating: IP54
- EASEOFCARE Service Plans offer a wide range of service options to protect your investment, ensuring maximum productivity and ROI

## INDUSTRY-APPLICATIONS

- OEM Applications:
  - Service Kiosks
  - Price Verifiers
  - Ticket Readers
  - Document Handling
  - Medical Laboratories
  - Vending Machines
  - Additional Automated Equipment

# GRYPHON™ I GFS4400 2D

DATALOGIC

TECHNICAL SPECIFICATIONS

## DECODING CAPABILITY

| | |
|---|---|
| 1D / LINEAR CODES | Autodiscriminates all standard 1D codes including GS1 DataBar™ linear codes. |
| 2D CODES | Aztec Code; China Han Xin Code; Data Matrix; MaxiCode; Micro QR Code; QR Code |
| POSTAL CODES | Australian Post; British Post; China Post; IMB; Japanese Post; KIX Post; Korea Post; Planet Code; Postnet; Royal Mail Code (RM4SCC) |
| STACKED CODES | EAN/JAN Composites; GS1 DataBar Composites; GS1 DataBar Expanded Stacked; GS1 DataBar Stacked; GS1 DataBar Stacked Omnidirectional; MacroPDF; MicroPDF417; PDF417; UPC A/E Composites |

## ELECTRICAL

| | |
|---|---|
| CURRENT | Operating (Typical): < 180 mA<br>Standby/Idle (Typical):<br>Automatic Object Sense Mode: 115 mA<br>Online & Serial OnLine Modes: 65 mA |
| INPUT VOLTAGE | 5 VDC +/- 5% |

## ENVIRONMENTAL

| | |
|---|---|
| AMBIENT LIGHT | 0 - 100,000 lux |
| DROP RESISTANCE | Withstands repeated drops from 0.76 m / 2.50 ft onto a concrete surface |
| HUMIDITY (NON-CONDENSING) | 5-95% |
| PARTICULATE AND WATER SEALING | IP54 |
| TEMPERATURE | Operating: -20 to 50 °C / -4 to 122 °F<br>Storage/Transport: -20 to 70 °C / -4 to 158 °F |

## INTERFACES

| | |
|---|---|
| INTERFACES | OEM (IBM) USB; RS-232; USB: USB COM; USB HID Keyboard |

## PHYSICAL CHARACTERISTICS

| | |
|---|---|
| COLORS AVAILABLE | Grey; Other colors and custom logo options are available for minimum quantity purchase. |
| DIMENSIONS | 3.9 x 5.7 x 5.8 cm / 1.5 x 2.2 x 2.3 in |
| WEIGHT (WITH CABLE) | USB: 170 g / 6.3 oz<br>RS-232: 204 g / 7.2 oz |

## READING PERFORMANCE

| | |
|---|---|
| IMAGE CAPTURE | Graphic Formats: BMP, JPEG, TIFF;<br>Greyscale: 256, 16, 2 |
| IMAGER SENSOR | Wide VGA: 752 x 480 pixels |
| LIGHT SOURCE | Aiming: 650 nm VLD |
| PRINT CONTRAST RATIO (MINIMUM) | 25% |
| READING ANGLE | Pitch: +/- 40°; Roll (Tilt): 180°; Skew (Yaw): +/- 40° |
| READING INDICATORS | Beeper (Adjustable Tone); Datalogic 'Green Spot' Good Read Feedback; Good Read LED |
| RESOLUTION (MAXIMUM) | 1D Linear: 0.102 mm / 4 mils<br>Data Matrix: 0.178 mm / 7 mils<br>PDF417: 0.102 mm / 4 mils |

## READING RANGES

| | |
|---|---|
| TYPICAL DEPTH OF FIELD | Minimum distance determined by symbol length and scan angle.<br>Printing resolution, contrast, and ambient light dependent.<br>Code 39: 5 mil: 4.7 to 17.7 cm / 1.8 to 7.0 in<br>Code 39: 10 mil: 1.7 to 33.2 cm / 0.7 to 13.1 in<br>Data Matrix: 10 mil: 2.7 to 17.1 cm / 1.0 to 6.7 in<br>Data Matrix: 15 mil: 1.2 to 24.6 cm / 0.5 to 9.7 in<br>EAN: 13 mil: 2.5 to 41.9 cm / 1.0 to 16.5 in<br>PDF417: 10 mil: 2.2 to 23.9 cm / 0.9 to 9.4 in<br>QR Code: 10 mil: 3.5 to 16.0 cm / 1.4 to 6.3 in |

## SAFETY & REGULATORY

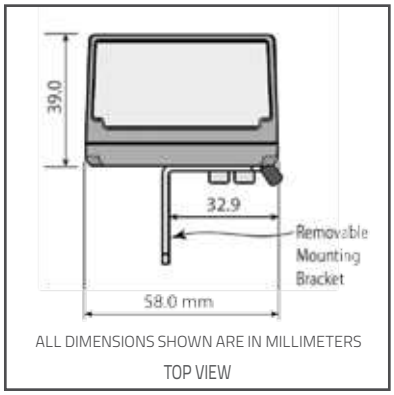
| | |
|---|---|
| AGENCY APPROVALS | The product meets necessary safety and regulatory approvals for its intended use.<br>The Quick Reference Guide for this product can be referred to for a complete list of certifications. |
| ENVIRONMENTAL COMPLIANCE | Complies to China RoHS; Complies to EU RoHS |
| LASER CLASSIFICATION | Caution Laser Radiation - Do not stare into beam; IEC 60825, Class2 |

## UTILITIES

| | |
|---|---|
| DATALOGIC ALADDIN™ | Datalogic Aladdin configuration program is available for download at no charge. |
| OPOS / JAVAPOS | JavaPOS Utilities are available for download at no charge.<br>OPOS Utilities are available for download at no charge. |
| REMOTE HOST DOWNLOAD | Lowers service costs and improves operations. |

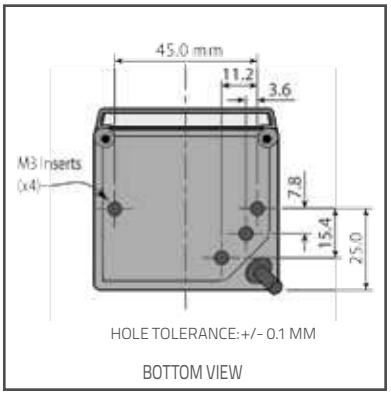
## WARRANTY

| | |
|---|---|
| WARRANTY | 3-Year Factory Warranty |



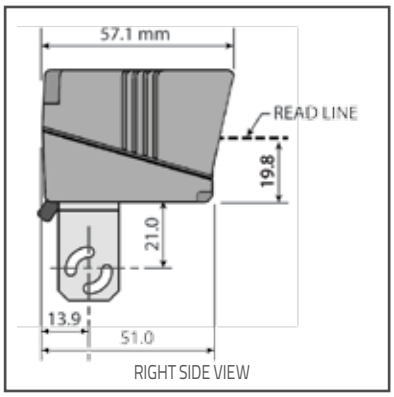
ALL DIMENSIONS SHOWN ARE IN MILLIMETERS

TOP VIEW



HOLE TOLERANCE: +/- 0.1 MM

BOTTOM VIEW



RIGHT SIDE VIEW



www.datalogic.com

# Solutions for Automatic Identification, Barcodes and Mobility

*More than 15,000 customers in 15 countries • More than 60,000 products in stock*


800 729 0236
www.logiscenter.us

0241 9550 9225
www.logiscenter.de

01 82 88 44 70
www.logiscenter.fr

020 3318 8265
www.logiscenter.co.uk

0720 115814
www.logiscenter.at

02 58 80 583
www.logiscenter.be

911 433 048
www.logiscenter.com

02 4792 1697
www.logiscenter.it

308 800 842
www.logiscenter.pt


## About LOGISCENTER

Logiscenter is the leading provider of automatic identification and data capture equipments. Our products include barcode printing and reading devices, mobile computing devices, wireless access points, identification cards, and consumables.

**All Brands Technical Service**

## Reasons to work with us

**Our customers come first.** Our objective is not only selling our products but also holding a long-lasting relationship with our customers, based on the benefits and advantages we create for you and your company.

**The best brands.** We offer products from the best manufacturers worldwide.

**Expert technical support team.** Our support staff are experts in the products we sell. But not just the products, they can tell you the best solution for your particular problem or need.

**Largest stock.** We have a permament stock of more than 60,000 products.

**Same-day shipping.** For all orders placed and confirmed before 16:00 on working days.

**Best prices.** With Logiscenter, you can be sure that you are paying a fair price for your products, we will not overcharge you.

**Easy returns.** Damaged or defective products can be returned easily, for an exchange or reimbursement.

## Our Brands

ZEBRA
Honeywell
DATALOGIC
elo
EPSON
unitech
star
PRINTRONIX
CITIZEN
Getac
TSC
TOSHIBA
MOTOROLA

## Solutions for: Industry • Consumer • Mobility Transportation and Logistics • Retail • Health

Label printers

Barcode readers

Data terminals & PDA

PVC card printers


RFID

Consumables
(labels, ribbons, PVC cards)

# TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8

## DESCRIPTION

The TP4056 is a complete constant-current/constant-voltage linear charger for single cell lithium-ion batteries. Its SOP package and low external component count make the TP4056 ideally suited for portable applications. Furthermore, the TP4056 can work within USB and wall adapter.

No blocking diode is required due to the internal PMOSFET architecture and have prevent to negative Charge Current Circuit. Thermal feedback regulates the charge current to limit the die temperature during high power operation or high ambient temperature. The charge voltage is fixed at 4.2V, and the charge current can be programmed externally with a single resistor. The TP4056 automatically terminates the charge cycle when the charge current drops to 1/10th the programmed value after the final float voltage is reached.

TP4056 Other features include current monitor, under voltage lockout, automatic recharge and two status pin to indicate charge termination and the presence of an input voltage.

## FEATURES

- Programmable Charge Current Up to 1000mA
- No MOSFET, Sense Resistor or Blocking Diode Required
- Complete Linear Charger in SOP-8 Package for Single Cell Lithium-Ion Batteries
- Constant-Current/Constant-Voltage
- Charges Single Cell Li-Ion Batteries Directly from USB Port
- Preset 4.2V Charge Voltage with 1.5% Accuracy
- Automatic Recharge
- two Charge Status Output Pins
- C/10 Charge Termination
- 2.9V Trickle Charge Threshold (TP4056)
- Soft-Start Limits Inrush Current
- Available Radiator in 8-Lead SOP Package, the Radiator need connect GND or impending

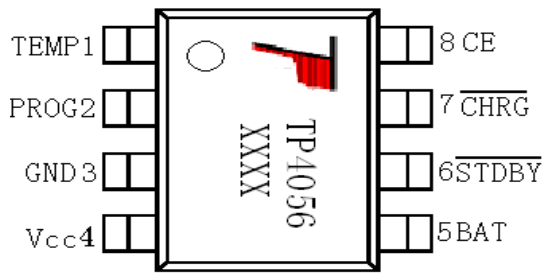
## ABSOLUTE MAXIMUM RATINGS

- Input Supply Voltage($V_{CC}$)：-0.3V～8V
- TEMP：-0.3V～10V
- CE：-0.3V～10V
- BAT Short-Circuit Duration：Continuous
- BAT Pin Current：1200mA
- PROG Pin Current：1200uA
- Maximum Junction Temperature：145℃
- Operating Ambient Temperature Range：-40℃～85℃
- Lead Temp.(Soldering, 10sec)：260℃

## APPLICATIONS

- Cellular Telephones, PDAs, GPS
- Charging Docks and Cradles
- Digital Still Cameras, Portable Devices
- USB Bus-Powered Chargers,Chargers

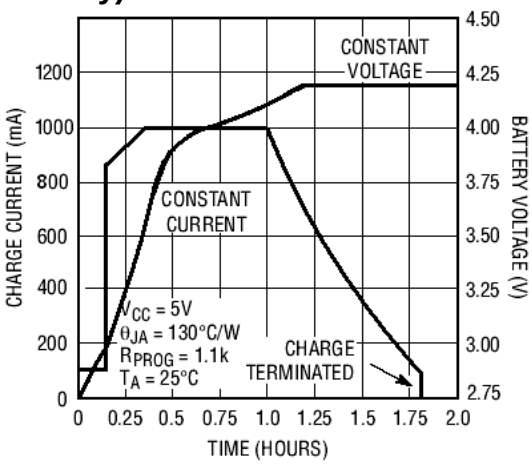
## PACKAGE/ORDER INFORMATION




SOP－8

| photo | ORDER PART NUMBER |
|---|---|
|  | TP4056-42-SOP8-PP |
| | **PART MARKING** TP4056 |

## Complete Charge Cycle (1000mAh Battery)



**TEMP(Pin 1) :Temperature Sense Input** Connecting TEMP pin to NTC thermistor's output in Lithium ion battery pack. If TEMP pin's voltage is below 45% or above 80% of supply voltage $V_{IN}$ for more than 0.15S, this means that battery's temperature is too high or too low, charging is suspended. The temperature sense function can be disabled by grounding the TEMP pin.

**PROG(Pin 2): Constant Charge Current Setting and Charge Current Monitor Pin** charge current is set by connecting a resistor $R_{ISET}$ from this pin to GND. When in precharge mode, the ISET pin's voltage is regulated to 0.2V. When in constant charge current mode, the ISET pin's voltage is regulated to 2V.In all modes during charging, the voltage on ISET pin can be used to measure the charge current as follows:

$$I_{BAT} = \frac{V_{PROG}}{R_{PROG}} \times 1200 \quad (V_{PROG}=1V)$$

**GND(Pin3): Ground Terminal**

**Vcc(Pin 4): Positive Input Supply Voltage** $V_{IN}$ is the power supply to the internal circuit. When $V_{IN}$ drops to within 30mv of the BAT pin voltage, TP4056 enters low power sleep mode, dropping BAT pin's current to less than 2uA.

**BAT(Pin5): Battery Connection Pin.** Connect the positive terminal of the battery to BAT pin. BAT pin draws less than 2uA current in chip disable mode or in sleep mode. BAT pin provides charge current to the battery and provides regulation voltage of 4.2V.

**$\overline{\text{STDBY}}$(Pin6): Open Drain Charge Status Output** When the battery Charge Termination, the $\overline{\text{STDBY}}$ pin is pulled low by an internal switch, otherwise $\overline{\text{STDBY}}$ pin is in high impedance state.

**$\overline{\text{CHRG}}$ (Pin7): Open Drain Charge Status Output** When the battery is being charged, the $\overline{\text{CHRG}}$ pin is pulled low by an internal switch, otherwise $\overline{\text{CHRG}}$ pin is in high impedance state.

**CE(Pin8): Chip Enable Input.** A high input will put the device in the normal operating mode. Pulling the CE pin to low level will put the YP4056 into disable mode. The CE pin can be driven by TTL or CMOS logic level.

## ELECTRICAL CHARACTERISTICS

The ● denotes specifications which apply over the full operating temperature range, otherwise specifications are at **$T_A$=25℃， $V_{CC}$=5V**， unless otherwise noted.

| SYMBOL | PARAMETER | CONDITIONS | | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|---|
| $V_{CC}$ | Input Supply Voltage | | ● | 4.0 | 5 | 8.0 | V |
| $I_{CC}$ | Input Supply Current | Charge Mode, $R_{PROG}$ = 1.2k | ● | | 150 | 500 | μA |
| | | StandbyMode(Charge | ● | | 55 | 100 | μA |
| | | Terminated) | ● | | 55 | 100 | μA |
| | | Shutdown Mode ($R_{PROG}$ Not Connected,$V_{CC}$ < $V_{BAT}$, or $V_{CC}$ < $V_{UV}$) | | | | | |
| $V_{FLOAL}$ | Regulated Output (Float) Voltage | 0℃≤$T_A$≤85℃， $I_{BAT}$=40mA | | 4.137 | 4.2 | 4.263 | V |
| $I_{BAT}$ | BAT Pin Current | $R_{PROG}$ = 2.4k, Current Mode | ● | 450 | 500 | 550 | mA |
| | Text condition:VBAT=4.0V | $R_{PROG}$ = 1.2k, Current Mode | ● | 950 | 1000 | 1050 | mA |
| | | Standby Mode, $V_{BAT}$ = 4.2V | ● | 0 | −2.5 | −6 | μA |
| $I_{TRIKL}$ | Trickle Charge Current | $V_{BAT}$<$V_{TRIKL}$, $R_{PROG}$=1.2K | ● | 120 | 130 | 140 | mA |
| $V_{TRIKL}$ | Trickle Charge Threshold Voltage | $R_{PROG}$=1.2K, $V_{BAT}$ Rising | | 2.8 | 2.9 | 3.0 | V |
| $V_{TRHYS}$ | Trickle Charge Hysteresis Voltage | $R_{PROG}$=1.2K | | 60 | 80 | 100 | mV |
| $T_{LIM}$ | Junction Temperature in Constant Temperature Mode | | | | 145 | | ℃ |

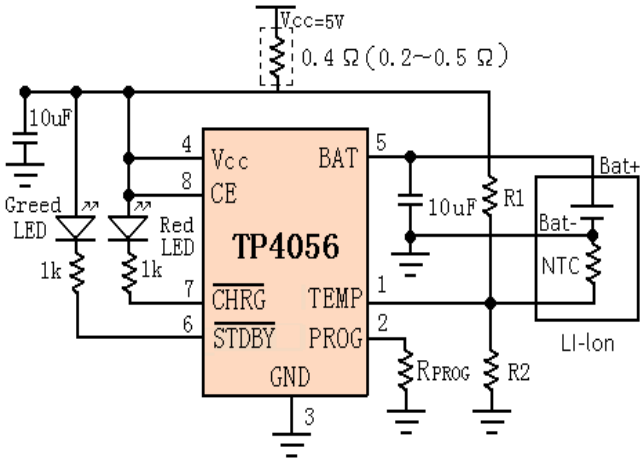
## indicator light state

| Charge state | Red LED $\overline{CHRG}$ | Greed LED $\overline{STDBY}$ |
|---|---|---|
| charging | bright | extinguish |
| Charge Termination | extinguish | bright |
| Vin too low; Temperature of battery too low or too high; no battery | extinguish | extinguish |
| BAT PIN Connect 10u Capacitance; No battery | Greed LED bright，Red LED Coruscate T=1-4 S | |

## Rprog Current Setting

| $R_{PROG}$ (k) | $I_{BAT}$ (mA) |
|---|---|
| 10 | 130 |
| 5 | 250 |
| 4 | 300 |
| 3 | 400 |
| 2 | 580 |
| 1.66 | 690 |
| 1.5 | 780 |
| 1.33 | 900 |
| 1.2 | 1000 |

## TYPICAL APPLICATIONS



# PRODUCT DATASHEET

## Confidex Viking™ Classic

Reliable industrial grade Bluetooth® Low Energy beacon for industrial identify, sense and locate applications.

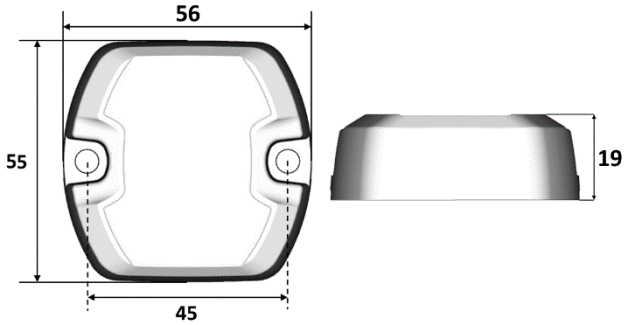
### ELECTRICAL SPECIFICATION

**Device type**
Bluetooth® Low Energy beacon
**Air interface protocol**
Bluetooth® 4.2
NFC: ISO/IEC 14443A
**Compliancy**
CE
FCC-ID: 2AMK9-CFXBLE-1
**Operational frequency**
ISM: 2402 - 2480 MHz
NFC: 13,56 MHz
**SOC**
Nordic Semiconductor NRF52832
**Memory configuration**
512 kB FLASH, 64 kB RAM
**Configuration interface**
NFC interface for fast deployments and configuration
**Sensor***
Built-in temperature sensor; other sensors by request
**Sensitivity**
-96 dBm sensitivity (Bluetooth® Low Energy)
**Read range****
Up to 200 m / 650 ft. Measured on and off metal.
**Battery type**
2 x CR2477 coin battery for maximal lifetime
**Applicable surface materials**
Can be attached to any surface

* Sensor is located inside the plastic housing, which limits real time measurement of the ambient temperature. Sensor calibration is available as additional service.
** Read ranges are measured in laboratory environment and there can be some variation in real application.

### MECHANICAL SPECIFICATION

**Encapsulation materials**
High quality PC/ASA
**Weight**
45 g
**Delivery format**
Single
**Tag dimensions**
56 x 54 x 19 mm / 2.20 x 2.13 x 0.75 in




### ENVIRONMENTAL RESISTANCE

**Operating temperature**
-20°C to +60°C / -4°F to +140°F
**Water resistance**
Good, tested 5 hours in 1m deep water (IP68)
**Vibration resistance**
JESD22-B103B, service condition 2; 3-axis vibration with 10G acceleration
**ESD immunity**
±8 kV according to EN 61000-4-2 (air discharge)
±4 kV according to EN 61000-4-2 (contact discharge)
**Chemical resistance**
No physical or performance changes in:
- 168h Motor oil exposure
- 48h Salt water (salinity 10%) exposure
- 48h Sulfuric acid (10%, pH 2) exposure
- 48h NaOH (10%, pH 13) exposure

Generally good resistance with moderate concentrations of acids, alcohols, alkalis, detergents and cleaners.
Acetone should be avoided
**Expected lifetime*****
Up to 10 years (depending on operating mode)

*** Beacon lifetime in nominal mode is optimized for long term use and is highly affected by operating mode parameters and ambient temperature. Configured TX power level and advertisement interval have influence on lifetime.

*Values in the table are the best recommendations; resistance against environmental conditions depends on the combination of all influencing factors, exposure duration and chemical concentrations. Thus, product's final suitability for certain environmental conditions is recommended to be tested. Contact Confidex for more specific information.*




contact.rfid@confidex.com www.confidex.com

## OPERATING MODES

Eddystone

Confidex Viking™ beacon utilizes Eddystone™ open standard frame format ensuring straightforward implementation on a wide range of Bluetooth® Low Energy devices. Following Eddystone™ frame types are supported:

**Eddystone™-UID** frame broadcasts 16-byte Beacon ID composed of a 10-byte namespace and a 6-byte instance.

**Eddystone™-URL** frame broadcasts a URL using a compressed encoding format.

**Eddystone™-TLM** frame broadcasts telemetry information as temperature.

**Eddystone™-EID** frame broadcasts and encrypted ephemeral identifier that changes periodically.

Eddystone™-EID frame operating mode requires service for resolving the ephemeral ID. Please contact Confidex for more information.

Confidex Viking™ Classic beacon can be factory configured to be compatible with Quuppa, AirFinder by LinkLabs, HERE, and Trimble operating modes.

## INSTALLATION INSTRUCTIONS

Confidex Viking™ Classic can be attached with several fixing methods:

1. Mechanical fixing – Screw

Mechanical fixing is recommended to be used in every application that includes risk for high mechanical stress or low temperature during tag fixing. Screw size M5 is to be used for fixing.

2. Mechanical fixing – Cable tie

Plastic or metallic cable ties can also be used for fixing Confidex Viking™. Maximum width of cable tie is 4 mm.

3. High performance acrylic adhesive

When mounting the tag with adhesive, clean and dry the surface for obtaining the maximum bond strength. Ideal application temperature is from +21°C to +38°C (+70°F to +100°F), bond strength can be improved with firm application pressure and moderate heating from +38°C to +54°C (+100°F to +130°F). Installation at temperatures below +10°C (+50°F) is not recommended.

## ORDER INFORMATION

**Product number:** 3002063, 3001896 (with accelerometer)

**Product name:** Confidex Viking™ Classic

Availability: Please contact Confidex Smart Industries Sales at www.confidex.com/contact-us

For additional information and technical support, please contact Confidex Ltd.







contact.rfid@confidex.com www.confidex.com

TRIAD MAGNETICS

**Switch Mode Power Supplies**
**Single Output**
**AC/DC Power Supply with PFC**

# AWSP100-5

## Description:

The AWSP100-5 is a single output power supply. This power supply is designed for a wide variety applications where high reliability is desired, including applications for the industrial and telecommunications markets. Excellent performance specifications are provided, together with compliance to European EMC (EN55022, Class B and EN61000-3-2), and Low Voltage directive (TUV EN60950).

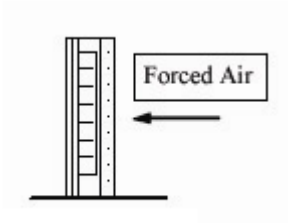
## Specifications (@25C)

### Input Characteristics:

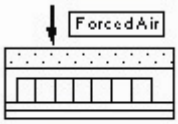
| | |
|---|---|
| **Input Voltage:** | 88-264Vac |
| **Input Frequency Range:** | 47-63Hz |
| **Max Input Current:** | 1.8A @ Vin (rated) |
| **Max Inrush Current:** | 20A@110Vac, 40A@220Vac at cold start |
| **Power Factor:** | >0.92 |
| **Leakage Current:** | <3.5mA/240Vac |

### Output Characteristics:

| | |
|---|---|
| **Output Voltage:** | 5Vdc |
| **Output Current:** | 20.0A |
| **Output Power:** | 100W |
| **Adjustable Output Range:** | ±10% |
| **Ripple & Noise:** | 100mV |
| **Load Regulation:** | ±1.0% |
| **Line Regulation:** | ±0.5% |
| **Efficiency:** | 75% |
| **Temperature Drift:** | <0.03%/°C (10-50°C) |
| **Start-up Time:** | 300ms max @ 230VAc |
| **Hold-up Time:** | 20ms min, 100% Load@230Vac |
| **Rise-up Time:** | 600ms max, 100% Load@230Vac |
| **Over Current Protection:** | At 105%<Load<150% |
| **Over Voltage Protection:** | At 125%<Load<145%@I/P:115/230VAC. Reset by recycle AC ON/OFF |



Mounting A



Mounting B

### General Specifications:

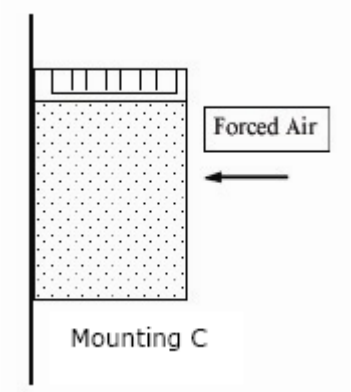
| | |
|---|---|
| **Switching Frequency:** | 134kHz (PWM) / 67kHz (PFC) |
| **Dimension (LxWxH):** | 199x98x42 |
| **Weight:** | 560g net, 610g gross |
| **Cooling:** | Natural Convection |
| **Isolation Resistance:** | I/P—O/P, I/P—FG, O/P—FG: 500VDC/100Mohms |
| **Dielectric Strength:** | I/P—O/P: 4.3KVdc; I/P—FG: 2.5KVac; O/P—FG: 0.5KVac |
| **Warranty:** | 2 years |

### Environmental Specifications:

| | |
|---|---|
| **Operating Temperature:** | -10° to 60°C |
| **Operating Humidity:** | 20 to 90% RH, non-condensing |
| **Storage Temperature:** | -25 to 75°C |
| **Storage Humidity:** | 10 to 95% RH, non-condensing |
| **Vibration:** | 10-55Hz, 2G 1min/cycle, period of 60min, each X, Y & Z axis |

### EMC & Safety Specifications:

| | |
|---|---|
| **EMC Emissions:** | Conforms to EN55022,VCCI,CISPR22 Class B (Conducted & Radiated); IEC-61000-4-2, 61000-4-4, 61000-4-5, 61000-4-11 |
| **ESD Susceptibility:** | EN61000-4-2 (15kV/8kV) |
| **Radiated Susceptibility:** | EN61000-4-3 (3Vm¹) |
| **Fast Burst:** | EN61000-4-4 (2kV) |
| **Surge:** | EN61000-4-5 (1kV/2kV) |
| **Safety Approval:** | UL 60950 (UL File No: E204980)<br>TUV EN60950 (TUV No: 50058679) |



Mounting C

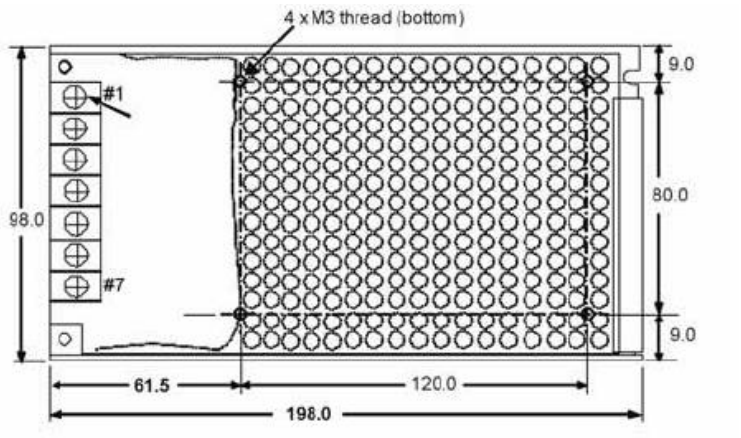
Web: www.TriadMagnetics.com
Phone 951-277-0757
Fax 951-277-2757

460 Harley Knox Blvd.
Perris, California 92571

Publish Date: April 15, 2019

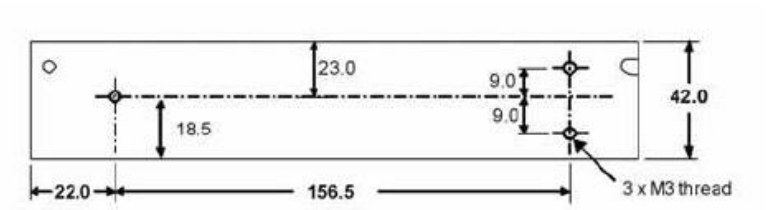
# Switch Mode Power Supplies
# Single Output
# AC/DC Power Supply with PFC

## Outline Dimensions (mm):



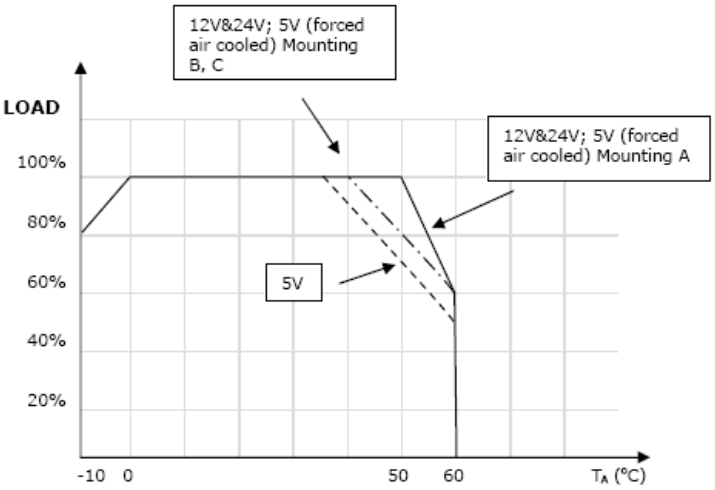

TERMINAL SCREW ASSIGNMENT

# 1,2 : AC INPUT
# 3 : FG
# 4,5 : DC OUTPUT -V
# 6,7 : DC OUTPUT +V




Unit : mm
Tolerance : +/-1.0mm

## Derating Curve:




## Connector Types:

**Input:** Screw Terminals
**Output:** Screw Terminals

**RoHS Compliance:** As of manufacturing date February 2016, all standard products meet the requirements of 2015/863/EU, known as the RoHS 3 initiative.

* Upon printing, this document is considered “uncontrolled”. Please contact Triad Magnetics’ website for the most current version.


Web: www.TriadMagnetics.com
Phone 951-277-0757
Fax 951-277-2757

460 Harley Knox Blvd.
Perris, California 92571



Publish Date: April 15, 2019

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

# PLANOS

SMARTCART

GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

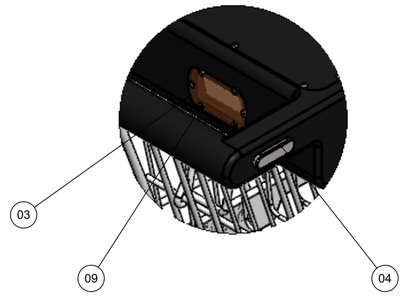
26 DE AGOSTO DE 2020

AUTOR: IGNACIO PAYÁ SILLA
TUTOR: PASCUAL PÉREZ BLASCO

**ÍNDICE**

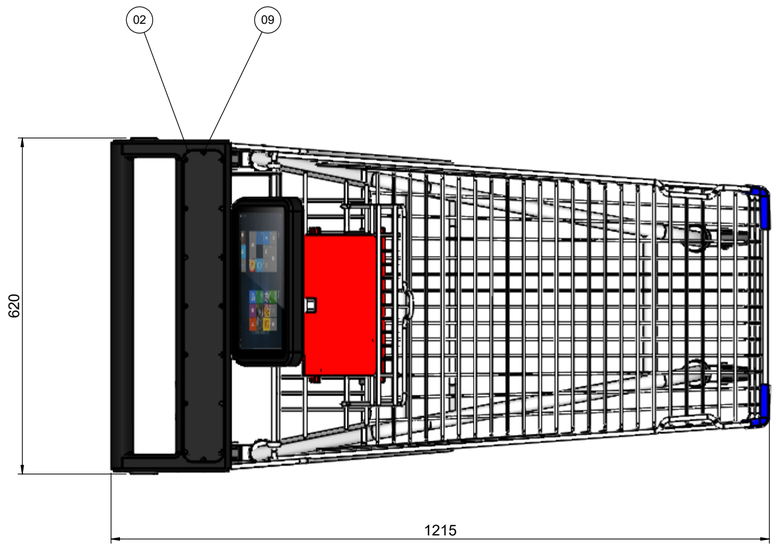


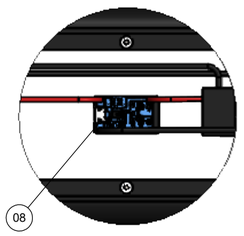
DETALLE A
ESCALA 1 : 4








DETALLE B
ESCALA 1 : 2

LISTADO DE PIEZAS

| ELEM | CTDA | Nº DE PIEZA | DESCRIPCION | MATERIAL |
|---|---|---|---|---|
| 01 | 1 | 01.001.001 | SOPORTE EMPUJE CARRO | Arnite negro (PET) |
| 02 | 1 | 01.001.002 | TAPA SUPERIOR | Arnite negro (PET) |
| 03 | 1 | 01.001.003 | TAPA LECTOR | Vidrio |
| 04 | 2 | 01.001.004 | CHAVETA | Inoxidable 316 |

LISTADO ELEMENTOS COMERCIALES

| ELEM | CTDA | REFERENCIA | DESCRIPCION | PROVEEDOR |
|---|---|---|---|---|
| 05 | 1 | | Carro supermercado | |
| 06 | 1 | GFS4400 2D | Lector código de barras | |
| 07 | 1 | PIPO X4 | Pantalla táctil | |
| 08 | 1 | TP4056 | Módulo para cargar baterias | |

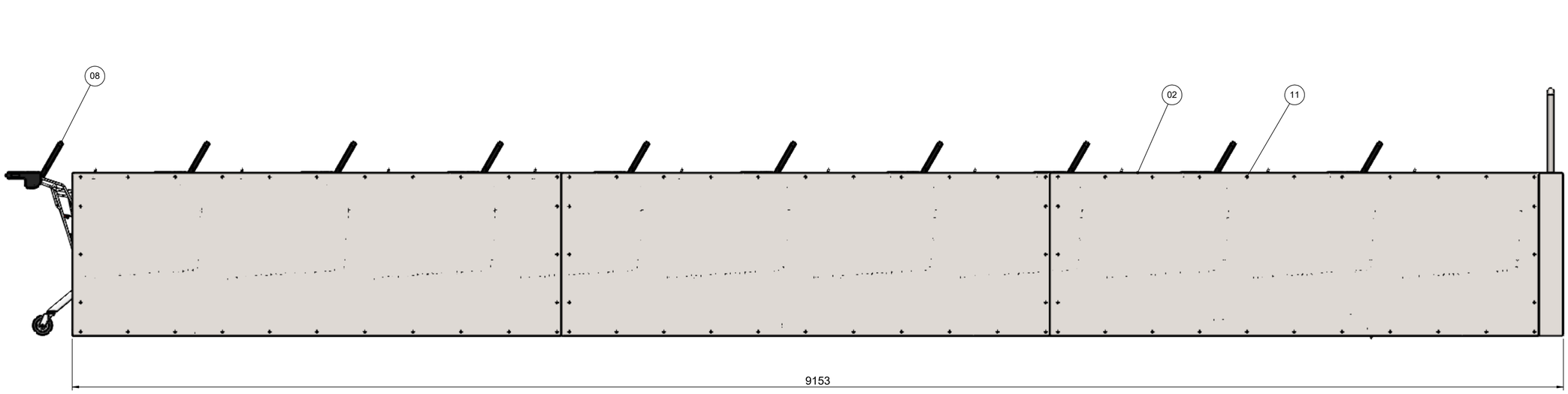
LISTADO DE TORNILLERIA

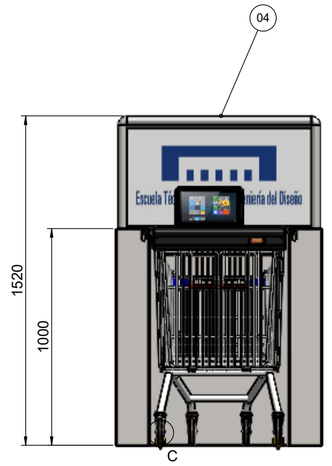
| ELEM | CTDA | REFERENCIA | DESCRIPCION | PROVEEDOR |
|---|---|---|---|---|
| 09 | 22 | M3x8 | Tornillo allen avellanado DIN 7991 | |

| | | | | | |
|---|---|---|---|---|---|
| 01 | | 1215 x 620 x 1000 mm | Montaje | | Peso (kg) 13.96 |
| CANT. | MATERIAL | DIMENSIONES | ELABORACION | TRA. TERMICO o SUPERFICIAL | OBSERVACIONES |

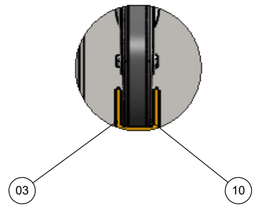
Escuela Técnica Superior de Ingeniería del Diseño

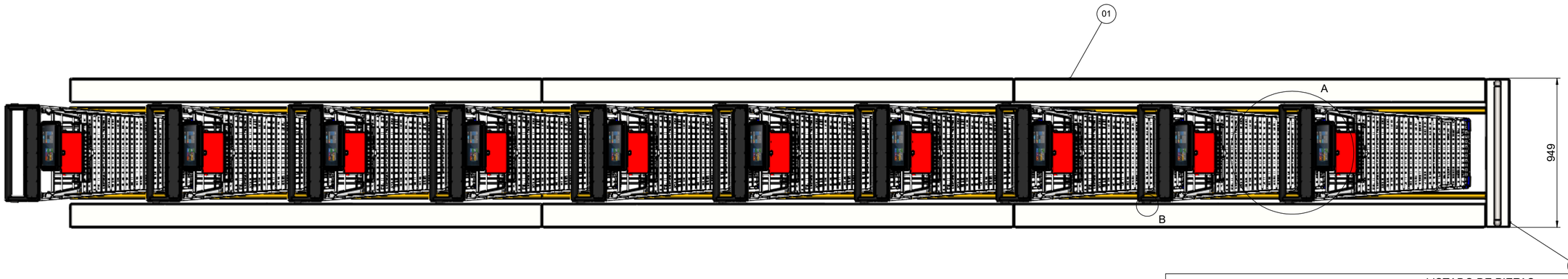
| | | | |
|---|---|---|---|
| Dibujado | Ignacio Payá | | TOLERANCIAS GENERALES UNE-EN 22768 (ISO 2768) |
| Comprobado | Ignacio Payá | | |
| Aprobado | Ignacio Payá | | Dimensionales m |
| Fecha 26/08/2020 | Descripción Plano: | | Geométricas k |
| Formato A2 / Escala 1:8 | SMARTCART | | Plano Nº: 01.001.C000 |
| | Subconjunto: | | |
| | Proyecto: SMARTCART | | HOJA 1/1 |

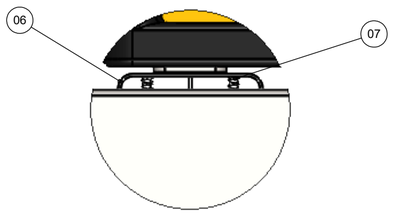


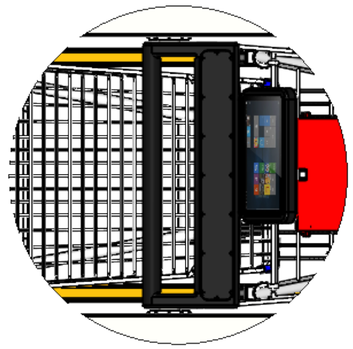

DETALLE C
ESCALA 1 : 4




DETALLE B
ESCALA 1 : 3



DETALLE A
ESCALA 1 : 10

## LISTADO DE PIEZAS

| ELEM | CTDA | Nº DE PIEZA | DESCRIPCION | MATERIAL |
|---|---|---|---|---|
| 01 | 6 | 01.001.010 | ESTRUCTURA LATERAL | AISI 316 |
| 02 | 6 | 01.001.011 | TAPA ESTRUCTURA LATERAL | AISI 316 |
| 03 | 6 | 01.001.012 | CARRIL RUEDAS CARRO | Plancha Fe |
| 04 | 1 | 01.001.013 | ESTRUCTURA FRONTAL | AISI 316 |
| 05 | 1 | 01.001.014 | TAPA ESTRUCTURA FRONTAL | AISI 316 |
| 06 | 18 | 01.001.015 | BALANCIN | AISI 316 |
| 07 | 36 | 01.001.016 | MUELLE DE COMPRESION | Comercial |
| 08 | 10 | 01.001.C000 | SMARTCART | |

## LISTADO ELEMENTOS COMERCIALES

| ELEM | CTDA | REFERENCIA | DESCRIPCION | PROVEEDOR |
|---|---|---|---|---|
| 09 | 1 | TP4056 | Módulo para carga de baterias | |
| 10 | 2 | Ø25.5 x 22 (110016) | Tope simple macho | AMC Mecanocaucho |

## LISTADO DE TORNILLERIA

| ELEM | CTDA | REFERENCIA | DESCRIPCION | PROVEEDOR |
|---|---|---|---|---|
| 11 | 101 | M5x8 | Tornillo allen avellanado DIN 7991 | |

| 03 | | 9153 x 949 x 1520 mm | Montaje | | Peso (kg) 1469.48 |
|---|---|---|---|---|---|
| CANT. | MATERIAL | DIMENSIONES | ELABORACION | TRA. TERMICO o SUPERFICIAL | OBSERVACIONES |

Escuela Técnica Superior de Ingeniería del Diseño

| | |
|---|---|
| Dibujado | Ignacio Payá |
| Comprobado | Ignacio Payá |
| Aprobado | Ignacio Payá |
| Fecha | 26/08/2020 |
| Formato | A2 |
| Escala | 1:20 |

TOLERANCIAS GENERALES UNE-EN 22768 (ISO 2768)
Dimensionales m
Geométricas k

Descripción Plano: BASE DE CARGA

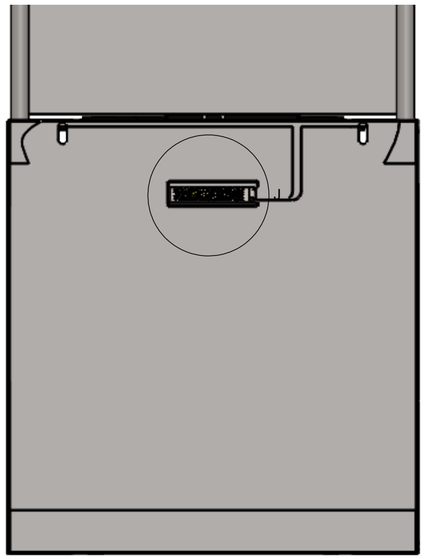
Subconjunto: Base de Carga

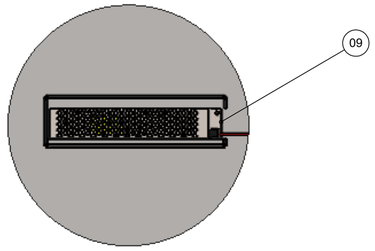
Proyecto: SMARTCART

Plano Nº: 01.001.C001

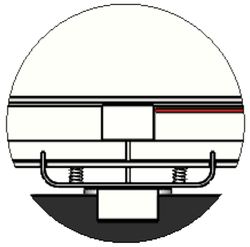
HOJA 1/2

**DETALLE CABLEADO**

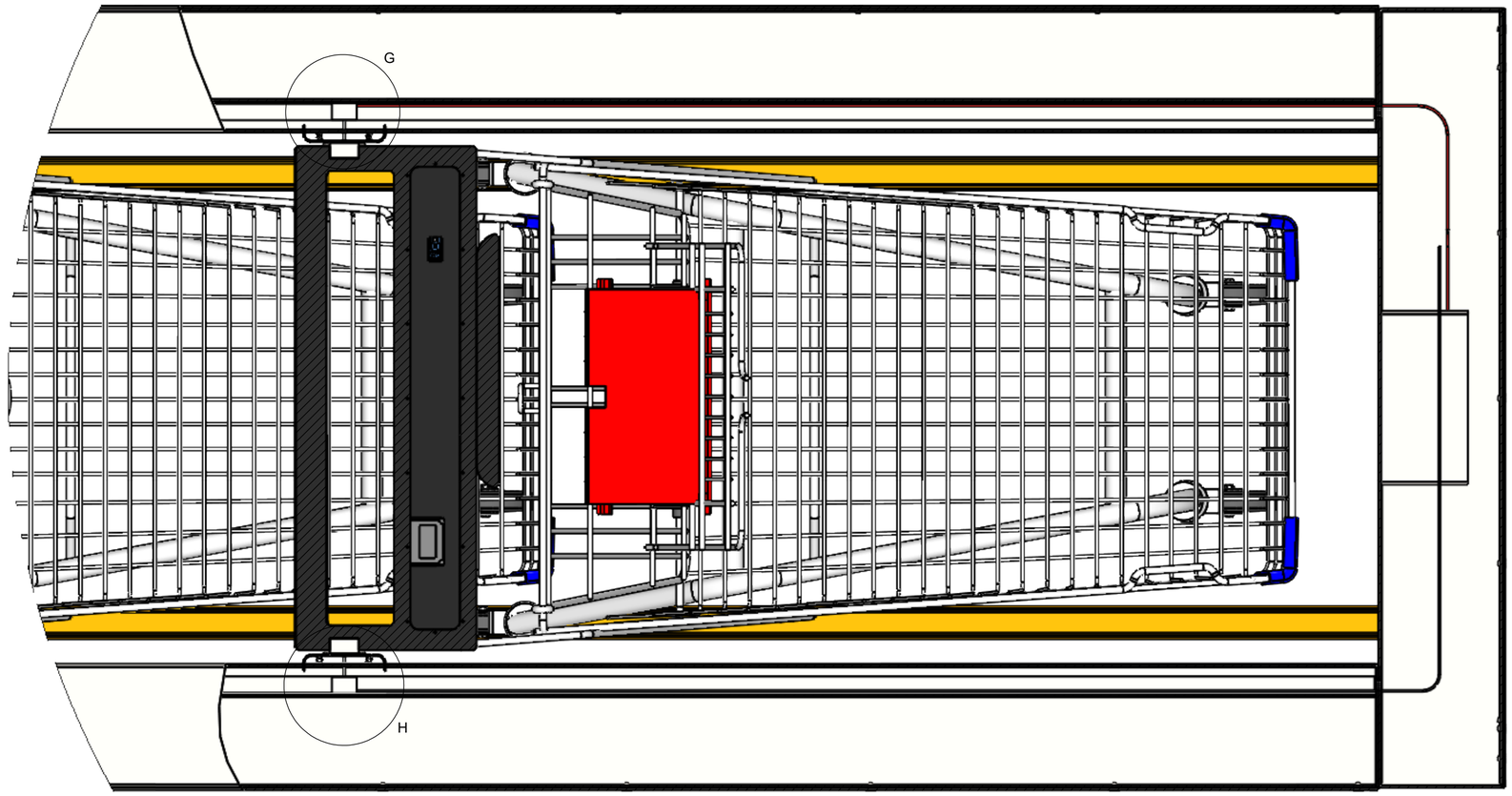




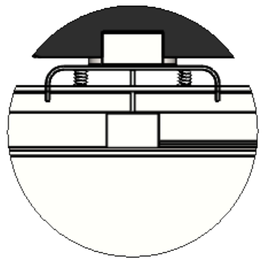

DETALLE J
ESCALA 1 : 5



DETALLE G
ESCALA 2 : 5




DETALLE CABLEADO



DETALLE H
ESCALA 2 : 5

| | | | | | |
|---|---|---|---|---|---|
| 03 | | 9153 x 949 x 1520 mm | Montaje | | Peso (kg) 1469.48 |
| CANT. | MATERIAL | DIMENSIONES | ELABORACION | TRA. TERMICO o SUPERFICIAL | OBSERVACIONES |

Escuela Técnica Superior de Ingeniería del Diseño

| | |
|---|---|
| Dibujado | Ignacio Payá |
| Comprobado | Ignacio Payá |
| Aprobado | Ignacio Payá |
| Fecha 26/08/2020 | Descripción Plano: BASE DE CARGA |
| Formato A2 / Escala 1:10 | Subconjunto: Base de Carga |
| | Proyecto: SMARTCART |

TOLERANCIAS GENERALES UNE-EN 22768 (ISO 2768)
Dimensionales m
Geométricas k

Plano Nº: 01.001.C001

HOJA 2/2

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

# PLIEGO DE CONDICIONES

SMARTCART

GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

26 DE AGOSTO DE 2020

AUTOR: IGNACIO PAYÁ SILLA
TUTOR: PASCUAL PÉREZ BLASCO

**ÍNDICE**

**ÍNDICE TABLAS**



**ÍNDICE IMÁGENES**

## 1. DEFINICIÓN Y ALCANCE DEL PLIEGO

En estos documentos del proyecto, los objetivos son fijar las condiciones técnicas mínimas a cumplir en caso de que se proceda a la instalación del sistema SMARTCART. Los ámbitos físicos se establecerán para las partes Hardware y los ámbitos logísticos, para la parte Software.

Especificar el posible hecho de que las condiciones que se establecerán en dicho documento pueden verse variadas por ciertos motivos que dependen del pedido, instalación o características del centro. Todo lo descrito es debido a los pedidos personalizados. Por este motivo, las normas que se estipularán a continuación, serán de carácter general.

## 2. CONDICIONES Y NORMAS DE CARÁCTER GENERAL

Las normas vigentes a tener en cuenta para poder completar adecuadamente el sistema SMARTCART son las siguientes:

### 2.1. SISTEMA SOFTWARE

- **ISO/IEC 25021: Describe la evaluación de los requisitos para la calidad del producto software.**
- **ISO/IEC 25000: Describe los requisitos y la calidad del software.**
- **ISO/IEC 9126: Orientada a la funcionalidad y la satisfacción del cliente que utiliza el producto software.**

### 2.2. DISEÑO DE LA CARCASA

- **UNE-EN 1929-1: PARTE 1 "Carros de supermercado. Parte 1: Requisitos y ensayos para carros de supermercado con o sin asiento para niños".**
- **UNE-EN 1929-2: PARTE 2 "Carros de supermercado. Parte 2: Requisitos, ensayos e inspección para carros de supermercado con o sin asiento para niños, destinados a ser utilizados en cintas transportadoras de pasajeros".**
- **UNE-EN 1929-3: PARTE 3 "Requisitos y ensayos para carros de supermercado con dispositivos adicionales para el transporte de objetos, con o sin asiento para niños".**
- **UNE-EN 1929-4: PARTE 4 "Requisitos y ensayos para carros de supermercado con dispositivos adicionales para el transporte de objetos, con o sin asiento para niños, destinados a ser utilizados en cintas transportadoras de pasajeros".**
- **UNE-EN 1929-5: PARTE 5 "Requisitos y ensayos para carros de supermercado con asientos para bebés y niños".**

### 2.3. SISTEMA DE GEOLOCALIZACIÓN

- **UNE-EN 62657-1: 2017 "Redes de comunicaciones industriales. Red de comunicación inalámbrica. Parte 1: Requisitos de comunicación inalámbrica y consideraciones de espectro (Ratificada por la Asociación Española de Normalización en diciembre de 2017)".**
- **UNE-EN 62657-2: 2017 "Redes de comunicaciones industriales. Red de comunicación inalámbrica. Parte 2: Gestión de coexistencia (Ratificada por la Asociación Española de Normalización en agosto de 2017)".**

2.4. ESTACIÓN DE CARGA

- **Real Decreto-ley 15/2018, 5 de octubre "Medidas urgentes para la transición energética y la protección de los consumidores".**

## 3. CONDICIONES DE LOS MATERIALES

### 3.1. MATERIAL CARCASA

El plástico técnico utilizado para la fabricación de la carcasa es **Arnite** de color negro con Bisulfuro de Molibdeno. Su denominación química es **Polietileno tereftalato** (**PET**, de forma abreviada). Se escoge en especial por su resistencia al desgaste y por sus aptas características con productos alimentarios.

En las siguientes tablas van a presentarse los resultados de las características que condicionan a dicho material:

| CARACTERÍSTICAS MECÁNICAS | PRUEBA (DIN/ASTM) | VALOR | UNIDAD |
|---|---|---|---|
| **Densidad** | 53479 | 1,37 | $g/cm^3$ |
| **Elongación en pto de fluencia** | 53455 | 81 | MPa |
| **Resistencia al desgarre** | 53455 | - | MPa |
| **Resistencia a la rotura por alargamiento** | 53455 | 70 | % |
| **Módulo de elasticidad a la tracción** | 53457 | 2800 | MPa |
| **Módulo de elasticidad a la flexión** | 53457 | - | MPa |
| **Dureza Brinell** | 53456 | 145 | MPa |
| **Resistencia al impacto** | 53453 | o. Br. | $KJ/m^2$ |
| **Resistencia a la fluencia tras 1000 h. de carga estática** | - | 36 | MPa |
| **Resistencia al alargamiento, por 1%, tras 1000 horas** | - | 13 | MPa |
| **Coeficiente de fricción contra acero endurecido y afilado p=0,05 $N/mm^2$, v=0,6 m/s** | - | 0,25 | - |
| **Desgaste por fricción en las mismas condiciones** | - | 0,35 | m/km |

*Tabla 1: Características mecánicas del PET*

| CARACTERÍSTICAS TÉRMICAS | PRUEBA (DIN/ASTM) | VALOR | UNIDAD |
|---|---|---|---|
| **Temperatura de fusión** | 53736 | 255 | - |
| **Temperatura de vitrificación dinámica** | 53736 | 69 | - |
| **Resistencia a la deformación; Procedimiento A** | 95 | 170 | ºC |
| **Resistencia a la deformación; Procedimiento B** | ISO 75 | 170 | ºC |
| **Temperatura de empleo a corto tiempo** | - | 180/100 | ºC |
| **Capacidad de conductividad calorífica específica** | - | 0,24 | W/(m.K) |
| **Capacidad calorífica** | - | 1,1 | J/(g.K) |
| **Coeficiente de dilatación longitudinal** | - | 7-8 | 10(-5)/k |

*Tabla 2: Características térmicas del PET*

| CARACTERÍSTICAS ELÉCTRICAS | PRUEBA (DIN/ASTM) | VALOR | UNIDAD |
|---|---|---|---|
| **Coeficiente dieléctrico** | 53483 | 3,2 | - |
| **Factor de pérdida dieléctrica** | 53483 | 0,021 | - |
| **Resistencia específica de paso** | 53482 | 10(14) | W.cm |
| **Resistencia superficial** | 53482 | 10(14) | W |
| **Resistencia de chispa eléctrica** | 53481 | 60 | KV/mm |
| **Resistencia a las corrientes parásitas** | 53480 | KC 350 | - |

*Tabla 3: Características eléctricas del PET*

| DATOS ADICIONALES | PRUEBA (DIN/ASTM) | VALOR | UNIDAD |
|---|---|---|---|
| **Absorción de humedad hasta saturar** | 53714 | 0,2 | % |
| **Absorción de agua hasta saturar** | 53495 | 0,5 | % |
| **Resistencia al agua caliente** | - | Inestable | - |
| **Inflamabilidad** | UL 94 | HB | - |
| **Comportamiento a la intemperie)** | - | Inestable | - |
| **Certificación FDA** | - | Apto | - |

*Tabla 4: Datos adicionales del PET*

### 3.2. PANTALLA

La pantalla PIPO X4 es la Tablet escogida para el prototipo a diseñar. Las especificaciones técnicas completas y sus características son las siguientes:

| AUDIO Y VIDEO | VALOR |
|---|---|
| **Formatos de Imagen** | BMP; GIF; JPEG; JPG; PNG |
| **Formatos de reproductor de música** | AAC; AC3; ALAC; APE; FLAC; MP3; OGG; WAV; WMA |
| **Formatos de reproductor de video** | AVS; H.263; H.264; MPEG1; MPEG2; MPEG4; MVC; RMVB; VC-1; WMV |

*Tabla 5: Audio y video PIPO X4*

| BATERÍA | VALOR | UNIDAD |
|---|---|---|
| **Capacidad de la batería** | 3.7 V-12000 mA/h | V-mA/h |

*Tabla 6: Batería PIPO X4*

| CÁMARA DE FOTOS | VALOR | UNIDAD |
|---|---|---|
| **Resolución trasera** | 5 | Megapíxeles |
| **Resolución delantera** | 2 | Megapíxeles |

*Tabla 7: Cámara de fotos PIPO X4*

| CONEXIÓN | VALOR |
|---|---|
| **Conectividad** | Bluetooth 4.0; Wi-Fi 802.11 b/g/n |
| **Conectores** | HDMI |

*Tabla 8: Conexión PIPO X4*

| DISEÑO | VALOR | UNIDAD |
|---|---|---|
| **Colores** | Negro | - |
| **Dimensiones altura** | 280 | mm |
| **Dimensiones ancho** | 185 | mm |
| **Dimensiones profundidad** | 26.5 | mm |
| **Peso** | 1020 | g |

*Tabla 9: Diseño PIPO X4*

| MEMORIA | VALOR | UNIDAD |
|---|---|---|
| **RAM** | 4 | Gb |
| **Memoria teléfono** | 64 | Gb |

*Tabla 10: Memoria PIPO X4*

| PANTALLA | VALOR | UNIDAD |
|---|---|---|
| **Membrana** | Capacitiva | - |
| **Tipo** | IPS | - |
| **Resolución ancho** | 800 | Pixels |
| **Resolución largo** | 1280 | Pixels |
| **Tamaño** | 10.1 | Pulgadas |
| **Pantalla táctil** | Multitáctil | - |

*Tabla 11: Pantalla PIPO X4*

| SISTEMA | VALOR |
|---|---|
| **Procesador** | Intel Atom X5 Z8350, Intel HD Graphics, Quad-core 1.92 GHz |
| **Sistema Operativo** | Windows 10 |

*Tabla 12: Sistema PIPO X4*

### 3.3. LECTOR DE CÓDIGO DE BARRAS

El modelo escogido es el DATALOGIC GRYPHON™ GFS4400 2D. Sus condiciones son las siguientes:

| CAPACIDAD DE DECODIFICACIÓN | VALOR |
|---|---|
| **1D/Códigos lineales** | Auto discrimina todos los códigos 1D estándar |
| **Códigos 2D** | Aztec Code; China Han Xin Code; Data Matrix; MaxiCode; Micro QR Code; QR Code |
| **Códigos postales** | Australian Post; British Post; China Post; IMB; Japanese Post; KIX Post; Korea Post; Planet Code; Postnet; Royal Mail Code (RM4SCC) |
| **Códigos apilados** | EAN/JAN Composites; GS1 DataBar Composites; GS1 DataBar Expanded Stacked; GS1 DataBar Stacked; GS1 DataBar Stacked Omnidirectional; MacroPDF; MicroPDF417; PDF417; UPC A/E Composites |

*Tabla 13: Capacidad de decodificación del lector de código de barras*

| CARACTERÍSTICAS ELÉCTRICAS | VALOR |
|---|---|
| **Corriente** | Trabajando (típico): <180 mA<br>Standby (típico): modo de detección automática de objetos: 115 mA<br>Online & modo Serial Online: 65 mA |
| **Voltaje de entrada** | 5 VDC +/- 5% |

*Tabla 14: Características eléctricas del lector de código de barras*

| CARACTERÍSTICAS MEDIOAMBIENTALES | VALOR |
|---|---|
| **Luz ambiental** | 0-100000 lux |
| **Resistencia a caídas** | Soporta caídas repetidas desde 0.76 m / 2.50 ft sobre una superficie de hormigón |
| **Humedad (sin condensación)** | 5-95% |
| **Sellado de partículas y agua** | IP54 |
| **Temperatura** | Trabajando: -20 a 50 °C<br>Transporte: -20 a 70 °C |

*Tabla 15: Características medioambientales del lector de código de barras*

| INTERFACES | VALOR |
|---|---|
| **Interfaces** | OEM (IBM) USB; RS-232; USB: USB COM; USB HID Keyboard |

*Tabla 16: Interfaces del lector de código de barras*

| CARACTERÍSTICAS FÍSICAS | VALOR | UNIDAD |
|---|---|---|
| **Colores** | Gris | - |
| **Dimensiones** | 3.9 x 5.7 x 5.8 | cm |
| **Peso (con el cable)** | USB: 170<br>RS-232: 204 | g<br>g |

*Tabla 17: Características físicas del lector de código de barras*

| RENDIMIENTO DE LECTURA | VALOR |
|---|---|
| **Captura de imagen** | Formatos Gráficos: BMP, JPEG, TIFF;<br>Escala de grises: 256, 16, 2 |
| **Sensor de imágenes** | Wide VGA: 752 x 480 pixels |
| **Fuente de luz** | Aiming: 650 nm VLD |
| **Relación de contraste de impresión (mínimo)** | 25% |
| **Ángulo de lectura** | Pitch: +/- 40°; Roll (Tilt): 180°; Skew (Yaw): +/- 40° |
| **Indicadores de lectura** | Beeper (Adjustable Tone); Datalogic 'Green Spot' Good Read Feedback; Good Read LED |
| **Resolución (máxima)** | 1D Linear: 0.102 mm<br>Data Matrix: 0.178 mm<br>PDF417: 0.102 mm |

*Tabla 18: Rendimiento de lectura del lector de código de barras*

| RANGOS DE LECTURA | CONDICIONES | VALOR |
|---|---|---|
| **Profundidad típica de campo** | Distancia mínima determinada por la longitud del símbolo y el ángulo de exploración.<br><br>Depende de la resolución de impresión, contraste y la luz ambiental. | Código 39: 5 mil: 4.7 a 17.7 cm<br>Código 39: 10 mil: 1.7 a 33.2 cm<br>Matriz data: 10 mil: 2.7 a 17.1 cm<br>Matriz data: 15 mil: 1.2 a 24.6 cm<br>EAN: 13 mil: 2.5 a 41.9 cm<br>PDF417: 10 mil: 2.2 a 23.9 cm<br>Código QR: 10 mil: 3.5 a 16.0 cm |

*Tabla 19: Rangos de lectura del lector de código de barras*

| REGULACIÓN Y SEGURIDAD | CONDICIONES |
|---|---|
| **Aprobaciones de la agencia** | El producto cumple con las aprobaciones reglamentarias y de seguridad necesarias para el uso previsto<br><br>Puede consultarse la guía de referencia rápida de este producto para obtener una lista completa de certificaciones |
| **Cumplimiento ambiental** | Cumple con China RoHS; Cumple con EU RoHS |
| **Clasificación Láser** | Precaución Radiación Láser – No mirar al rayo; IEC 60825, Clase 2 |

*Tabla 20: Regulación y seguridad del lector de código de barras*

### 3.4. MÓDULO DE CARGA DE BATERÍAS

Se trata del módulo TP4056, un cargador para baterías de litio. Puede regular la temperatura del sistema al que esté conectado. Sus condiciones son las siguientes:

| CARACTERÍSTICAS | VALOR |
|---|---|
| **Carga programable de corriente** | Hasta 1000 mA |
| **MOSFET** | No, requiere de diodo de bloqueo o sensor resistivo |
| **Cargador** | Lineal de paquete SOP-8 para baterías de lítio-ion de celda única<br>Puerto USB |
| **Corriente, Voltaje** | Constantes |
| **Carga de Voltaje** | 4.2 V con 1.5% de precisión |
| **Recarga** | Automática |
| **Terminación de carga** | C/10 |
| **Umbral de Carga por goteo** | 2.9 V |
| **Límites de la corriente de irrupción** | Arranque suave |
| **Radiador** | Paquete SOP de 8 derivaciones, conectado al GND |

*Tabla 21: Características del TP4056*

| ÍNDICES ABSOLUTOS MÁXIMOS | VALOR | UNIDAD |
|---|---|---|
| **Vcc** | -0.3 - 8 | V |
| **TEMP** | -0.3 – 10 | V |
| **CE** | -0.3 – 10 | V |
| **BAT Corto-circuito Duración** | Continua | - |
| **BAT Pin Corriente** | 1200 | mA |
| **PROG Pin Corriente** | 1200 | uA |
| **Temperatura máx de unión** | 145 | °C |
| **Rango de temp. Ambiente** | -40 – 85 | °C |
| **Temperatura de plomo** | 260 | °C |

*Tabla 22: Índices absolutos máximos del TP4056*

### 3.5. SISTEMA DE GEOLOCALIZACIÓN

El recurso a utilizar para el sistema de localización ***indoor*** es la tecnología BLE. Sus condiciones presentes en cuanto a la arquitectura física son las siguientes:

| CARACTERÍSTICAS ARQUITECTURA | VALOR |
|---|---|
| **Modulación** | GFSK de 0.45 a 0.55 |
| **Tasa Mbit/s** | 1 Mbit/s |
| **Nº de Canales** | 40 |
| **Separación** | 2 MHz |
| **Banda de frecuencia** | 2.4 GHz |
| **Rango** | 100 m |
| **Tipología de Red** | Scatternet |
| **Tasa** | 1 Mbps |
| **Consumo de Pico de corriente** | < 15 mA |
| **Corriente en Standby** | < 2 µA |

*Tabla 23: Arquitectura BLE*

### 3.6. FUENTE DE ALIMENTACIÓN CONMUTADA

Modelo AWSP100-5 del fabricante **TRIAD MAGNETICS.** Condiciones:

| CARACTERÍSTICAS DE ENTRADA | VALOR |
|---|---|
| **Voltaje** | 88-264 Vac |
| **Rango de Frecuencia** | 47-63 Hz |
| **Corriente máxima** | 1.8 A @ Vin |
| **Corriente de irrupción máxima** | 20 A @ 110 Vac; 40 A @ 220 Vac con inicio frío |
| **Factor de Potencia** | >0.92 |
| **Corriente de fuga** | <3.5 mA / 240 Vac |

*Tabla 24: Características de entrada de la fuente de alimentación*

| CARACTERÍSTICAS DE SALIDA | VALOR |
|---|---|
| **Voltaje** | 5 Vdc |
| **Corriente** | 20.0 A |
| **Potencia** | 100 W |
| **Rango de salida ajustable** | ±10% |
| **Ondulación y ruido** | 100 mV |
| **Regulación de carga** | ±1.0% |
| **Regulación de línea** | ±0.5% |
| **Eficiencia** | 75% |
| **Deriva de temperatura** | <0.03%/°C (10-50°C) |
| **Tiempo de puesta en marcha** | 300 ms máx @230 Vac |
| **Tiempo de espera** | 20 ms min, 100% carga @ 230 Vac |
| **Tiempo de subida** | 600 ms máx, 100% carga @ 230 Vac |
| **Protección contra sobrecarga** | Al 105%<carga<150% |
| **Protección contra sobretensión** | Al 125%<@<145% @ I/P: 115/230 Vac |

*Tabla 25: Características de salida de la fuente de alimentación*

| ESPECIFICACIONES GENERALES | VALOR |
|---|---|
| **Frecuencia de cambio** | 134 kHz (PWM) / 67 kHz (PFC) |
| **Dimensiones (L x W x H)** | 199 x 98 x 42 |
| **Peso** | 560 g neto / 610 g bruto |
| **Enfriamiento** | Convección natural |
| **Resistencia de aislamiento** | I/P – O/P, I/P – FG, O/P – FG: 500 Vdc/100 Mohms |
| **Resistencia dieléctrica** | I/P – O/P: 4.3 kVdc; I/P – FG: 2.5 kVac; O/P – FG: 0.5 kVac |

*Tabla 26: Especificaciones generales de la fuente de alimentación*

| ESPECIFICACIONES AMBIENTALES | VALOR |
|---|---|
| **Temperatura de trabajo** | -10 a 60°C |
| **Humedad de trabajo** | 20 a 90% RH, sin condensación |
| **Temperatura de almacenamiento** | -25 a 75°C |
| **Humedad de almacenamiento** | 10 a 95% RH, sin condensación |
| **Vibraciones** | 10-55 Hz, 2G 1 min/ciclo, período de 60 min, para los ejes X, Y y Z |

*Tabla 27: Especificaciones ambientales de la fuente de alimentación*

| EMC Y ESPECIFICACIONES DE SEGURIDAD | VALOR |
|---|---|
| **Emisiones de EMC** | Conforme a EN55022, VCCI, CISPR22 Clase B (Conducta y Radiación); IEC-61000-4-2, 61000-4-4, 61000-4-5, 61000-4-11 |
| **Susceptibilidad de ESD** | EN61000-4-2 (15kV/8kV) |
| **Susceptibilidad de radiación** | EN61000-4-3 (3Vm) |
| **Ráfaga rápida** | EN61000-4-4 (2kV) |
| **Oleada** | EN61000-4-5 (1kV/2kV) |
| **Aprobación de seguridad** | UL 60950 (UL File No: E204980) TUV EN60950 (TUV No: 50058679) |

*Tabla 28: EMC y Especificaciones de seguridad de la fuente de alimentación*

## 4. CONDICIONES DE LA EJECUCIÓN

### 4.1. SISTEMA SOFTWARE

Para realizar la Aplicación móvil acorde a los requisitos del comprador, hay que tener en cuenta las siguientes condiciones que marcarán los materiales, diseño y tiempo de ejecución del sistema software:

- Diferenciar entre aplicación móvil y página web del centro correspondiente
- Diseño limpio y cuidado, simplificado y de fácil comprensión
- Sin cambios del diseño, mantener la misma identidad en todo momento
- Tipografía acordada con la empresa compradora, al igual que los colores, imágenes, etc.
- Seguir los requisitos de la plataforma con la que se trabaje, en este caso Android
- Conocer todos los productos del centro para implementarlos en el servidor
- Conocer las zonas específicas del centro para incorporarlas en el servidor
- Ligar un servidor de autoayuda para poder funcionar sin problemas
- Registrar a los trabajadores del centro
- Establecer un intervalo de tiempo entre 2-3 semanas para la ejecución de la aplicación

### 4.2. CARCASA

La fabricación de la carcasa del prototipo se pedirá a un proveedor con el que se establecerán los parámetros del material, diseño y tiempo de ejecución. Se creará una ficha técnica del resultado y se proporcionará (junto con el contrato) a la empresa cliente. Las condiciones que se acordarán con el proveedor son las siguientes:

- Enviar los planos con el diseño, las medidas y el material
- Establecer las cantidades a fabricar, consultando con la empresa compradora
- Establecer en el contrato el tiempo establecido para completar el pedido con el proveedor
- Informar de los posibles cambios antes de la fecha de inicio de ejecución
- Negociar una reducción de pago en caso de incumplimiento de los plazos o posibles errores en la fabricación
- Tramitar el pago de envío del material una vez finalizado el paquete

### 4.3. PANTALLA, LECTOR DE CÓDIGO DE BARRAS, MÓDULO DE CARGA DE BATERÍAS Y FUENTE DE ALIMENTACIÓN

Como todos estos materiales se compran a proveedores externos a la empresa, tienen que establecerse los pedidos por parte de la empresa cliente antes de realizarse. Una vez acordada la cantidad, se completan los pedidos y se establece el importe del montante en el presupuesto final del proyecto.

### 4.4. BASE DE CARGA

La cantidad de bases de carga que se instalarán en el centro dependerán, única y exclusivamente, del número de SMARTCARTS que quieran montar. Las bases se componen de tres parcelas de 10 carros cada uno.

### 4.5. MONTAJE PROTOTIPO

Una vez se disponga de todo el material para completar el montaje del prototipo SMARTCART, las condiciones de su ejecución son las siguientes:

- Comprobación del acoplamiento correcto de los materiales dentro de la carcasa
- Comprobación del acoplamiento correcto del sistema en el carro
- Comprobación del montaje de la base de carga con las medidas marcadas por la empresa cliente
- Comprobación del funcionamiento de las fuentes de alimentación acopladas en las bases de carga
- Comprobación de la debida distribución de la carga en todos los módulos en todas las bases de carga
- Comprobación de la medida idónea de las bases, los carros deben entrar, cargarse y salir sin problema
- Comprobación de la protección establecida en las bases de carga
- Establecer un intervalo de tiempo de ejecución de entre 5-7 días
- Aplicar el importe dentro del presupuesto final del proyecto

### 4.6. SISTEMA DE GEOLOCALIZACIÓN

Para poder completar este pedido, su ejecución depende de las instalaciones de la empresa cliente. Hay que realizar un estudio con las medidas del centro, la distribución de las estanterías donde se almacenan los productos, el cálculo de las posibles zonas muertas, zonas opacas e interferencias que puedan hallarse, etc. Además, debe completarse el pedido de prototipos por carro que establezca la empresa cliente para conocer la cantidad de carros que hay que localizar dentro del centro. La ejecución, por tanto, tendrá que tener en cuenta las siguientes condiciones:

- Conocer la cantidad de prototipos a localizar
- Conocer la distribución de los productos para establecer las plataformas dentro de la App
- Conocer la cantidad de cajas de pago para enviar el pedido en caso de finalización de compra
- Implementar dos ***beacons*** por cada zona específica del centro

- Implementar ***beacons*** a todas las salidas del centro para informar a seguridad
- Establecer un intervalo de entre 5-7 días para las instalaciones del sistema en el centro
- Aplicar el importe dentro del presupuesto final del proyecto

## 5. PRUEBAS DE SERVICIO

### 5.1. SISTEMA SOFTWARE

Las pruebas que deben completarse en el sistema software son abundantes, dando comienzo a la prueba de atributos de calidad:

Atributos de calidad

Se requiere el cumplimiento de las peticiones por parte de los usuarios, garantizando con ello la calidad del sistema a desarrollar. Es por ello por lo que se adjunta la siguiente imagen:


*Imagen 1: Requisitos de calidad de productos software*

Destacar como atributo más importante a tener en cuenta la Usabilidad, pues demuestra la sencillez con la que un cliente puede manejar la aplicación móvil.

Marcos de trabajo y niveles de pruebas

Encontramos dos marcos de trabajo principales. Por un lado, tenemos los secuenciales y, por otro, los iterativos (que son incrementales). Para ambos casos podemos encontrar ejemplos en la siguiente imagen:

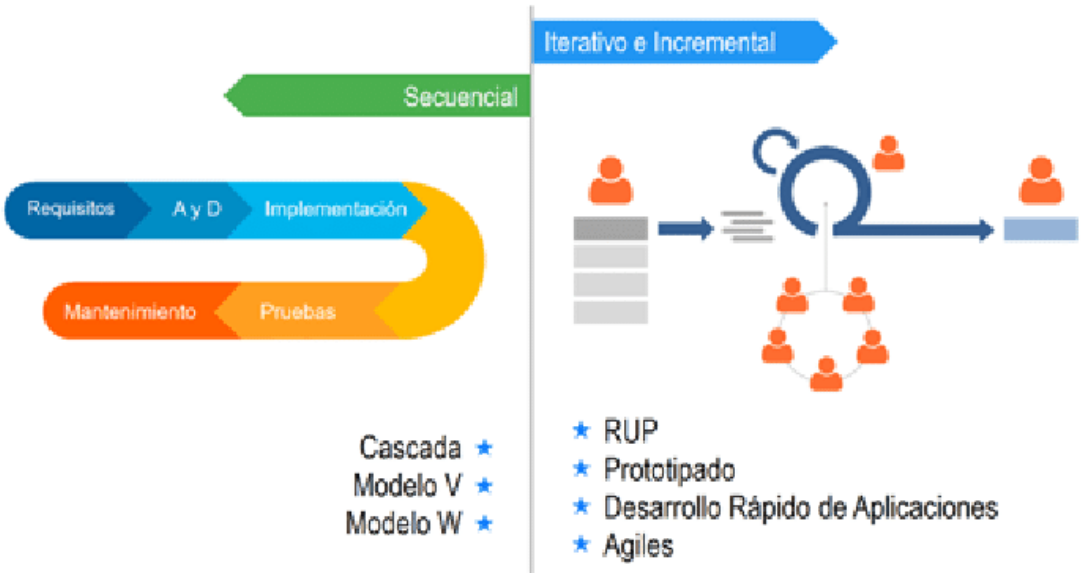



*Imagen 2: Marcos de trabajo de desarrollo software*

Durante los últimos años, se ha intentado agilizar el manejo de las aplicaciones para facilitar su uso y hacer sentir cómodo al cliente, pero no es un rasgo que deba tener en cuenta el operario que se encargue de realizar las pruebas de servicio.

Para analizar los niveles de pruebas se tiene en cuenta el alcance de evaluación del producto. Esto conduce a que dependa de su propósito, uso, comportamiento y estructura. Las pruebas que se realizarán son las siguientes:

| TIPO DE PRUEBA | DESCRIPCIÓN |
|---|---|
| **Pruebas unitarias** | Prueban las unidades más minúsculas del sistema software incluyendo el módulo de éste, con el objetivo de validarlas |
| **Pruebas de componentes** | Testean cada componente, independientemente del resto del software, para comprobar que funciona como es debido |
| **Pruebas de integración** | Una vez probada cada componente del software, se cercioran de su correcto ensamblado |
| **Pruebas de sistema** | Comprueban el sistema en su conjunto |
| **Pruebas de aceptación** | Los usuarios verifican que la aplicación está lista para salir al mercado |
| **Pruebas de regresión** | Trata de realizar pruebas señaladas anteriormente, en caso de cambios realizados en la aplicación |

*Tabla 29: Tipos de pruebas para la clasificación por niveles*

Para tener una idea clara de cuáles son los niveles que contienen este tipo de pruebas, se toma como referencia la imagen insertada a continuación:


*Imagen 3: Niveles de pruebas para el desarrollo software*

5.2. CARCASA

Para la carcasa, completaremos unas pruebas de servicio donde analizaremos las propiedades mecánicas de este material y así tener toda la información relevante acerca de su comportamiento óptimo.

Realizaremos las pruebas con una Máquina Universal de Ensayos (MUE), pues gracias a esta herramienta se podrán completar pruebas como la de tracción, compresión, etc.

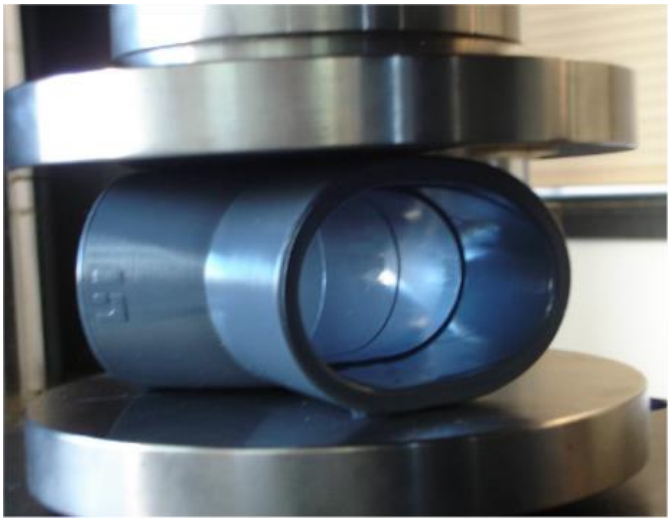


*Imagen 4: Prueba de compresión con la herramienta MUE*

| PRUEBAS MECÁNICAS | DESCRIPCIÓN |
|---|---|
| **Tracción** | Aplicación de esfuerzo en el mismo sentido que el eje longitudinal de las probetas |
| **Compresión** | Comportamiento del material a sometimientos de carga de compresión a una determinada velocidad uniforme |
| **Flexión** | Capacidad del material para resistir fuerzas aplicadas perpendicularmente a su eje longitudinal |
| **Coeficientes de fricción estático y dinámico** | Relación entre la fuerza de tracción exigible para comenzar el deslizamiento o continuarlo entre dos superficies y la fuerza de gravedad que actúa perpendicular sobre ellas |
| **Separación por pelado** | Determina la fuerza de unión de los materiales plásticos en forma de láminas |
| **Punción** | Es un ensayo común para láminas |
| **Ensayo de desgarro** | Determina la fuerza exigible para propagar el rasgado de un corte definido a partir de otro ya practicado |
| **Dureza** | Resistencia del material a ser rayado o incluso penetrado. Tiene en cuenta el módulo de Elasticidad del material |
| **Impacto** | Se estudia con diferentes modos: Por caída libre, caída de proyectil sobre el material o impacto pendular |

*Tabla 30: Pruebas mecánicas para materiales plásticos*

## 5.3. PANTALLA, LECTOR DE CÓDIGO DE BARRAS, MÓDULO DE CARGA DE BATERÍAS Y FUENTE DE ALIMENTACIÓN

Con respecto a todos estos dispositivos electrónicos, las pruebas de servicio se remiten a comprobar con multímetros, pinzas amperimétricas y vatímetros, que los valores de tensión, corriente y potencia de salida son los correspondidos con los que se mencionan en las hojas de características de cada uno de ellos.


*Imagen 5: Multímetro Eléctrico*

## 5.4. SISTEMA DE GEOLOCALIZACIÓN

En este sistema, la importancia de las pruebas recae sobre todo en la transmisión de la información de manera correcta, por lo que es importante comprobar la estructura de los paquetes de prueba, la potencia de salida, las características de modulación y el error de frecuencia.

Existe un modo de evaluación directo (DTM) para completar pruebas de transmisión y recepción a través de una interfaz de control cableada. Se activa mediante comandos externos.

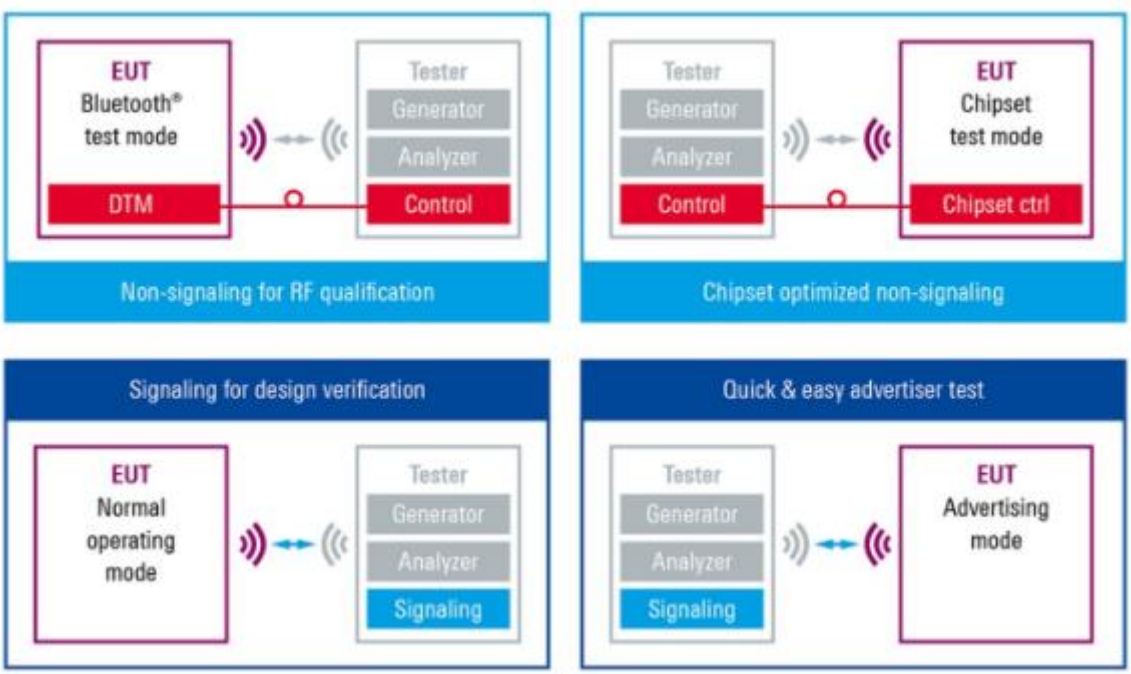



*Imagen 6: Pruebas de servicio para la tecnología BLE*

## 6. BIBLIOGRAFÍA


[1] – Agencia Estatal BOE (12 de noviembre de 2007). *Boletín Oficial del Estado. Legislación.* [31 de diciembre de 2008]. Información obtenida en: https://www.boe.es/legislacion/legislacion.php

[2] – Sanmetal, S.A. (1971). *Termoplásticos, ARNITE-PET.* [12 de febrero de 2010]. Información obtenida en: http://www.sanmetal.es/productos/termoplasticos/arnite-pet/10

[3] – PIPO (2015). *Tablet PIPO X4*. [agosto de 2018]. Información obtenida en: https://www.pipo-store.com/pipo-x4.html

[4] – Logiscenter (s.f.). *Condiciones del material para el lector de código de barras Datalogic Gryphon GFS4400 2D.* [21 de octubre de 2017]. Información obtenida en: https://www.logiscenter.com/lector-codigo-barras-datalogic-gryphon-i-gfs4400?gclid

[5] – Isaac (s.f.). *TP4056: el módulo para cargar baterías.* [8 de agosto de 2017]. Información obtenida en: https://www.hwlibre.com/tp4056/

[6] – Gómez, Carles; Oller, Joaquim; Paradells, Josep (s.f.). *Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Thecnology.* [2012]. Información obtenida en: https://www.areatecnologia.com/nuevas-tecnologias/bluetooth-le.html

[7] – UNITEL (s.f.). *Los 10+1 requisitos a tener en cuenta en el diseño de una APP.* [23 de septiembre de 2018]. Información obtenida en: https://unitel-tc.com/10-requisitos-diseno-crear-app/

[8] – B. Bruegge, A. H. Dutoit (s.f.). *Ingeniería de software orientado a objetos.* [2002]. Información obtenida en: file:///C:/Users/chuki/Downloads/Dialnet-PruebasAAplicacionesMoviles-7019200.pdf

[9] – AIMPLAS (1990). *Ensayos de propiedades mecánicas.* [2011]. Información obtenida en: https://www.aimplas.es/tipos-ensayos/propiedades-mecanicas-de-los-materiales-plasticos/

[10] – Gómez Martín, José Antonio (s.f.). *Análisis de los mecanismos de posicionamiento en interiores basados en huellas.* [2014]. Información obtenida en: https://www.rohde-schwarz.com/es/soluciones/test-and-measurement/wireless-communication/wireless-connectivity/bluetooth/tests-de-bluetooth/tests-de-bluetooth_230304.html

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

# PRESUPUESTO

SMARTCART

GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

26 DE AGOSTO DE 2020

AUTOR: IGANCIO PAYÁ SILLA
TUTOR: PASCUAL PÉREZ BLASCO

**ÍNDICE**



**ÍNDICE TABLAS**

## 1. INTRODUCCIÓN

En lo que respecta al presupuesto del proyecto, se incluyen los materiales, el diseño de la aplicación móvil y la mano de obra. Con respecto a algunos casos, hay que tener en cuenta que el precio es establecido por los proveedores a los que se les pide el material. Otro de los factores a tener en cuenta es la personalización de los pedidos, que depende de las cantidades que el cliente estipule por contrato. Los fabricantes son escogidos por la empresa vendedora, por lo que los precios se ajustarán a esta medida.

## 2. PRESUPUESTO DE LOS MATERIALES

En la siguiente tabla tenemos reflejados los materiales que se van a utilizar para el desarrollo del sistema, con sus correspondientes precios por unidad y en adición, una columna que incluye el precio total de la suma de las cantidades. En su mayoría serán precios establecidos por parte de los proveedores, con lo que se ajustarán a su medida.

*Los materiales que contengan en la columna de **UNIDADES** el símbolo (#), significa que la cantidad de unidades dependerá del pedido estipulado por el comprador, pero para poder completar una simulación del presupuesto vamos a suponer que los prototipos demandados son un total de 100.

**Para la cantidad de *beacons* a utilizar, dependemos de las dimensiones del centro, la distribución de los productos y la cantidad de salidas que contenga el supermercado. Para completar esta simulación de presupuesto, vamos a suponer que el supermercado divide sus instalaciones en 5 zonas clasificadas (por cada zona se necesitan dos *beacons*) y que además dispone de 2 salidas especificadas.

| MATERIALES | UNIDADES | PRECIO (€/ud) | PRECIO TOTAL |
|---|---|---|---|
| **Carcasa PET** | (#) 100 | 15,95 | 1595 |
| **Pantalla PIPO X4** | (#) 100 | 272,03 | 27203 |
| **Módulo de carga de baterías TP4056** | (#) 100 | 1,30 | 130 |
| **Lector de código de barras GFS4400 2D** | (#)100 | 183,80 | 18380 |
| **Fuente de alimentación AWSP100-5** | (#) 10 | 42,40 | 424 |
| ***Beacons* CONFIDEX VIKING CLASSIC** | 12 | 29,90 | 358,8 |
| **Planchas acero inoxidable 800 x 600 x 2 mm** | (#) 200 | 31,37 | 3137 |
| **Vías de aluminio 3000 x 75 mm** | (#) 60 | 3,37 | 202,2 |
| **Topes para carros** | (#) 20 | 0,79 | 15,8 |
| **Muelle de compresión** | (#) 400 | 0,59 | 236 |
| **Tornillos allen avellanado DIN 7991** | (#) 1300 | 0,09 | 117 |
| **Cableado 12 AWG Rojo 3 m** | (#) 74 | 5,99 | 443,26 |
| **Cableado 12 AWG Negro 3 m** | (#) 74 | 5,99 | 443,26 |
| **Pines para cables 90 x 64 x 24 mm** | (#) 400 | 0,07 | 28 |
| | | **SUMA TOTAL** | 52613,32 |

*Tabla 1: Presupuesto de los materiales*

## 3. PRESUPUESTO DE LA APLICACIÓN MÓVIL

El presupuesto de la aplicación móvil es aplicado para su codificación y su parte gráfica (el diseño). Además, las bases de datos se crean en base a las peticiones del supermercado, en este caso necesitamos 6 (una para los empleados y las otras 6 para cada sección).

El mantenimiento de la aplicación corre a cargo del vendedor. En caso de pretender cambios, la empresa cliente debe consultar a la empresa vendedora. Ésta enviará a un ingeniero que completará las exigencias del cliente.

Respecto a las bases de datos, el mantenimiento corre a cargo de la empresa cliente. Los cambios se visualizarán en la aplicación inmediatamente. Sin embargo, si la exigencia corresponde a añadir una nueva base de datos, se deberá consultar a la empresa vendedora.

| APLICACIÓN MÓVIL | PRECIO TOTAL |
|---|---|
| **Software** | 1800 |
| **Diseño personalizado** | 500 |
| **Mantenimiento software** | 100 |
| **Base de datos registro trabajadores** | 400 |
| **Base de datos registro productos** | 1000 |
| **Mantenimiento Bases de datos** | 100 |
| **SUMA TOTAL** | 3900 |

*Tabla 2: Presupuesto de la aplicación móvil*

## 4. PRESUPUESTO DE LA MANO DE OBRA

Con respecto al montante del sistema, la tabla contendrá el número de operarios necesarios para realizar los diferentes montajes, inspecciones y diseños de los 100 carros y sus bases. También incluye el tiempo requerido para cada ejercicio, el precio por horas y el total.

| MANO DE OBRA | OPERARIO (o) | TIEMPO (h/o) | PRECIO (€/h/o) | PRECIO TOTAL |
|---|---|---|---|---|
| **Montaje en carro** | 6 | 40 | 35,12 | 8428,8 |
| **Diseño de planos** | 1 | 24 | 48,23 | 1157,52 |
| **Diseño software** | 1 | 80 | 68,25 | 5460 |
| **Montaje base de carga** | 2 | 16 | 35,12 | 1123,84 |
| **Montaje beacons** | 2 | 40 | 35,12 | 2809,6 |
| **SUMA TOTAL** | | | | 18979,76 |

*Tabla 3: Presupuesto de la mano de obra*

## 5. PRESUPUESTO COMPLETO

Una vez completados todos los presupuestos anteriores, se realiza una suma de todo el montante para enviar los resultados al cliente. En esta tabla también se añadirá el resultado al aplicar el I.V.A. (21%).

| PRESUPUESTOS | PRECIO |
|---|---|
| **Presupuesto de materiales** | 52613,32 |
| **Presupuesto de Aplicación móvil** | 3900 |
| **Presupuesto de mano de obra** | 18979,76 |
| **Total** | 75493,08 |
| **I.V.A. (21%)** | 15853,55 |
| **Total con I.V.A.** | **91346,63** |

*Tabla 4: Presupuesto completo*

Como resultado de la operación, para un pedido concreto de 100 prototipos SMARTCART y unas instalaciones que requieren de 12 balizas "*beacons*" electrónicas, el presupuesto asciende a un total de **91346,63**€, incluyendo el I.V.A.