



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



Universitat Politècnica de València

Departament de Sistemes Informàtics i
Computació

DESARROLLO DE UNA HERRAMIENTA DE GESTIÓN PARA SISTEMAS CONTRA INCENDIOS BASADA EN MODELOS CONCEPTUALES: DE REQUISITOS A CÓDIGO

Trabajo Fin de Máster

**Máster Universitario en Ingeniería y Tecnología de
Sistemas Software**

Alumno: Arnau Jordà Verdú

Tutores: Óscar Pastor López, Ignacio Panach Navarrete

2019-2020

Resumen

La idea del proyecto es aplicar un método de producción de software de requisitos a código. Se utilizarán modelos conceptuales en un entorno real, incluyendo un modelo organizacional basado en iStar (*) definiendo objetivos y actores de la organización estratégicos.

Tras ello, se elaborará un modelo de negocios con casos de uso para representar los procesos de la organización.

A partir de estos procesos y requisitos, se generará un modelo conceptual que represente esos requisitos. A partir de estos modelos, se implementará mediante la tecnología Flutter la aplicación final, que deberá ser coherente con todos los modelos desarrollados.

La finalidad del trabajo será ver cual es la trazabilidad del desarrollo de un sistema de información complejo, desde los requisitos hasta el código, observando qué aspectos son susceptibles de ser automatizados para ayudar posteriormente a otros proyectos en la mayor medida posible aprovechando los mapeados y correspondencias conceptuales de este proyecto.

La aplicación y el método se hará en base al desarrollo de una herramienta de gestión para una empresa de mantenimiento e instalaciones de sistemas contra incendios.

Palabras clave: Modelado Conceptual, Desarrollo Dirigido por Modelos, Desarrollo de Software centrado en Esquemas Conceptuales, Ingeniería de Sistemas de Información.

Abstract

The idea of the project is to apply a software production method from requirements to code. Conceptual models will be used in a real-world environment, including an organizational model based on iStar (i*) defining strategic organizational objectives and actors.

After that, a business model with notation of use case model will be developed to represent the processes of the organization.

Based on these processes and requirements, a conceptual model will be generated that represents those requirements. From these models, the final application will be implemented using Flutter technology, which must be consistent with all developed models.

The purpose of the work will be to see what is the traceability of the development of a complex information system, from requirements to code, observing which aspects are likely to be automated to then assist other projects to the greatest extent possible by taking advantage from the mappings and conceptual correspondences from this project.

The application and method will be made on the basis of the development of a management tool for a maintenance company and fire system installations.

Keywords: Conceptual Modeling, Model Driven Development, Conceptual Schema-centric Software Development, Information Systems Engineering

TABLA DE CONTENIDOS

RESUMEN	3
ABSTRACT	4
TABLA DE CONTENIDOS	5
ÍNDICE DE FIGURAS	7
CAP1. INTRODUCCIÓN	9
CAP2. MOTIVACIÓN Y OBJETIVOS	11
2.1 MOTIVACIÓN	11
2.2 OBJETIVOS	12
CAP3. ESTADO DEL ARTE	16
3.1 MODEL-DRIVEN DEVELOPMENT	16
3.2 HERRAMIENTAS MDD	17
3.3 PROBLEMAS Y BENEFICIOS DEL MDD	18
3.3 DETALLES DEL MÉTODO	20
3.4 TECNOLOGÍAS PARA LA IMPLEMENTACIÓN	21
3.4.1 Flutter	21
3.4.2 Server	22
3.4.3 Bases de Datos MySQL	22
3.4.4 Bases de Datos FireBase	22
3.4.5 PHP	22
3.4.6 Sketch	23
CAP4. ESPECIFICACIÓN	24
4.1 REQUISITOS DE USUARIO	24
4.1.1 Requisitos Funcionales	24
4.1.2 Requisitos No Funcionales	25
4.2 DESCRIPCIÓN DEL SISTEMA	26
CAP5. DESARROLLO DEL PROYECTO	29
5.1 CIM (ANÁLISIS DEL SISTEMA)	29
5.1.1 Modelado i* (iStar - iEstrella)	29
5.1.2 Casos de Uso	37
5.2 PIM (DISEÑO DEL SISTEMA)	45
5.2.1 DIAGRAMA DE CLASES	45
5.2.2 DIAGRAMAS DE ACTIVIDAD	48
5.2.2 Prototipado y Mockups	52
5.3 PSM (CÓDIGO A IMPLEMENTAR)	60
5.3.1 Tecnología.....	60
5.3.2 Bases de Datos	60
5.3.3 Código de Pantallas y Funciones.....	63
5.4 DISCUSIÓN DEL MÉTODO	77
5.4.1 CIM.....	77
5.4.2 PIM.....	78
5.4.3 PSM.....	79
CAP6. PRUEBAS Y RESULTADOS	80
6.1 PRUEBAS	80
6.1.1 Pruebas de Usabilidad.....	80
6.1.2 Cuestionario de Usabilidad.....	84
6.1.4 Prueba de compatibilidad	89
7. CONCLUSIONES	93

8. REFERENCIAS.....	94
9. ANEXO	97
9.1 RECURSOS SVG.....	97

ÍNDICE DE FIGURAS

figura 1:	esquema MDA	21
figura 2:	i* resumen actores	29
figura 3:	i* funcionamiento en la nube	30
figura 4:	i* controlar stock	31
figura 5:	i* realizar contabilidad	32
figura 6:	i* gestión clientes	33
figura 7:	i* mantenimientos pendientes	34
figura 8:	i* control de propiedades	35
figura 9:	i* modelo completo	36
figura 10:	T1 casos de uso	37
figura 11:	T2 casos de uso	38
figura 13:	diagrama final casos de uso	39
figura 14:	T2 clases	45
figura 15:	T3 clases	46
figura 16:	T5 clases	46
figura 17:	diagrama de clases	47
figura 18:	T2 actividad	48
figura 19:	DA registrarse	49
figura 20:	DA gestionar productos	49
figura 21:	DA gestionar fichas clientes	50
figura 22:	DA gestionar trabajadores	50
figura 23:	DA gastos e ingresos	50
figura 24:	DA controlar stock	50
figura 25:	DA chat	50
figura 26:	DA instalaciones	51
figura 27:	DA mantenimientos	51
figura 28:	mockups registro	53
figura 29:	mockups registro negocio	53
figura 30:	mockups menú inicial	54
figura 31:	mockups almacén	54
figura 32:	mockups clientes	55
figura 33:	mockups sucursales	55
figura 34:	mockups instalaciones	56
figura 35:	mockups incidencias	56
figura 36:	mockups instalaciones (añadir)	57
figura 37:	mockups instalaciones revisar	57
figura 38:	mockups mantenimientos	58
figura 39:	mockups trabajadores	58
figura 40:	mockups estadísticas	59
figura 41:	mockups chat	59
figura 42:	tablas bases de datos	62
figura 43:	captura registro	65
figura 44:	captura almacén	67
figura 45:	captura añadir producto	67
figura 46:	captura clientes	69

figura 47:	captura aviso incidencia	69
figura 48:	captura añadir cliente	69
figura 49:	captura añadir trabajador	70
figura 50:	captura estadísticas	72
figura 51:	captura listado ingresos	72
figura 52:	captura escoger mes	72
figura 53:	captura instalaciones	74
figura 54:	captura revisar instalación	74
figura 55:	captura lista bluetooth	74
figura 56:	etiqueta impresa	74
figura 57:	captura mantenimientos	75
figura 58:	captura escáner	75
figura 59:	captura seleccionar chat	76
figura 60:	captura sala de chat	76
figura 61:	dispositivo 1-1	89
figura 62:	dispositivo 1-2	89
figura 63:	dispositivo 1-3	89
figura 64:	dispositivo 1-4	89
figura 65:	dispositivo 2-1	90
figura 66:	dispositivo 2-2	90
figura 67:	dispositivo 2-3	90
figura 68:	dispositivo 2-4	90
figura 69:	dispositivo 3-1	91
figura 70:	dispositivo 3-2	91
figura 71:	dispositivo 3-3	91
figura 72:	dispositivo 3-4	91
figura 73:	dispositivo 4-1	92
figura 74:	dispositivo 4-2	92
figura 75:	dispositivo 4-3	92
figura 76:	dispositivo 4-4	92

CAP1. INTRODUCCIÓN

Las metodologías de desarrollo software son las estructuras que se utilizan para planificar y controlar los procedimientos de creación de un sistema software. En las organizaciones, el BPM (*Business Process Model*) se encarga de sus cambios en los modelos y procesos de negocio, de forma que un analista empresarial puede controlarlos. Aunque sigue habiendo un problema, la brecha entre los analistas y los desarrolladores sigue siendo extensa.

Para tratar de solucionar esto, *Model Driven Engineering* (MDE) constituye una propuesta útil a la hora de transferir los cambios en los procesos de la empresa a los sistemas que se encargan de la implementación de estos. Por lo tanto, el uso de ciertas recomendaciones de MDE como *Model Driven Architecture* (MDA), puede mejorar la interacción entre los analistas de negocios y los ingenieros de desarrollo software. [1]

Los objetivos que se pretenden abordar en este proyecto de fin de master, es la comprobación de la aplicación de un método de producción de software que, mediante objetivos y metas organizacionales, permitan materializar un modelo de requisitos el cual plasmaremos en un modelo conceptual y finalmente, pasarlo a código mediante la implementación de un software en dispositivos móviles que sirva para gestionar una empresa de sistemas de gestión para sistemas contra incendios.

La arquitectura del desarrollo estará basada en modelos conceptuales, de acuerdo con el estándar MDA. Estos modelos se irán transformando hasta la obtención del producto final (implementación del código). De esta forma, obtenemos unas soluciones como la automatización de partes del proceso y partes reutilizables por otras, con los beneficios que ello otorga.

En cuanto al sistema a implementar, los Sistemas de Gestión son programas o herramientas que han sido diseñados para permitir la optimización de recursos, mejorar la producción y reducir los costes gracias a la adaptación de estos a las necesidades propias de la industria que necesita de su uso:

- Automatización de procesos
- Elaboración de informes
- Facturación
- Cohesión del equipo mediante su uso
- Mayor movilidad

Con estos sistemas se obtiene un mayor control de un negocio, ya que se consiguen resultados que de otra forma requerirían de mas tiempo o desembolso de dinero, pudiendo automatizar tareas que hasta ahora eran manuales o unificarlas todas en una simple herramienta. Por ejemplo, se podrá tener a mano el apartado de stock, contabilidad, planificación, etc. de una empresa en vez de tenerlo todo en softwares separados.

Estos sistemas de gestión también son conocidos como ERP ("*Enterprise Resource Planning*"). Su definición es un conjunto de herramientas o sistemas de información que nos permite integrar las operaciones de una empresa: producción, logística, recursos, envíos, contabilidad, etc. Se adaptan según la empresa o se adquiere uno estándar en el cual se estaría pagando por cosas que no se usas. [2]

Una herramienta como la planteada para este proyecto desembocaría en un ahorro para el negocio, ya que evita depender de varios programas a la vez con sus correspondientes cuotas y mantenimiento, simplemente unificando todo en un único software que incluya todos los aspectos necesarios para este tipo de empresa o negocio.

La implementación se verá realizada para *smartphones* con sistema operativo Android e iOS mediante el SDK de Flutter [3], conexiones PHP [4] con uso de bases de datos MySQL [5] alojados en servidores web de *EditAJob*, unos servidores propios alojados en el proveedor de servicios host de *Hostinger*. Es decir, la aplicación estará en constante conexión a internet ya que los datos se encontrarán en la nube, siendo posible su acceso desde cualquier parte con tan solo iniciar sesión, de forma privada mediante encriptaciones Hash como se especificará a lo largo de este documento.

CAP2. MOTIVACIÓN Y OBJETIVOS

2.1 MOTIVACIÓN

Con el paso de los años, más y más comercios se adaptan a las nuevas tecnologías, recurriendo a software estándar, TPV con hardware físico con un mayor gasto, etc. [6]. ¿Lo malo? No todos los tipos de negocio cuentan con herramientas estandarizadas, por lo que pedir un software propio adaptado a ellos a una empresa puede ser un trabajo que requiera de un gasto elevado.

En pequeñas o medianas empresas puede suponer un gran esfuerzo económico y de aprendizaje e incluso, carecer de soluciones tecnológicas que satisfagan estos requisitos del comercio.

¿Cómo podríamos solucionar parte de esos problemas? El proyecto tratará de desarrollar una aplicación usando de forma rigurosa principios del desarrollo dirigido por modelos conceptuales, donde los modelos se convierten en el artefacto software esencial de la solución. La aplicación va a servir para realizar la gestión de una empresa, en este caso, se trata de una empresa que gestiona sistemas contra incendios, comprobando tras su implementación a partir de modelos conceptuales su factibilidad y posibles automatizaciones que podrían surgir para futuros desarrollos de herramientas tipo POS o ERP de distintos grupos de negocio: restaurantes, ferreterías, floristerías, etc.

Estamos en un momento donde más del 87% de los españoles usa un *smartphone* en su día a día (2019) [7], en cambio, no todos los trabajos contemplan adaptarse a estas nuevas tecnologías ya sea por motivos económicos, desconocimiento, falta de herramientas específicas, etc. Con esta aplicación diferentes empresas de un mismo sector podrían utilizar esta herramienta sin tener que desembolsar un gasto elevado en software propio.

Lo que se hará será establecer los aspectos comunes de este negocio, qué necesitan las empresas que se dedican al mismo sector.

Se pueden encontrar aplicaciones para gestionar negocios o pedir a empresas que se dedican a crear aplicaciones, hacer una ceñida al negocio pero esto presenta una serie de insuficiencias:

- Aplicaciones de gestión generales poco adaptadas al sector.
- Falta de aspectos fundamentales del negocio: control de stock, etiquetado, actualización de estados, recordatorios de mantenimiento.
- No distingue entre administradores y operarios (los primeros contarán con diferentes opciones y funciones).
- Falta de interconexión entre dispositivos con una BBDD conjunta al necesitar distintas apps para gestionar un sector.

Personalmente, el desarrollo de aplicaciones móviles ha sido una parte del desarrollo software que siempre me ha gustado y llamado la atención desde el instituto. En este, nos presentábamos a concursos llevados a cabo por la UPV de creación de aplicaciones móviles mediante MIT APP INVENTOR, en ambos años en los que me presenté, quedé en segunda posición. El primer año presenté una aplicación para el deporte, donde se ofrecían rutinas de entrenamiento, contador de pulsaciones, trazado de rutas en mapas, etc.) y en el segundo un organizador para estudiantes con agenda personal, calculadora de medias para la carrera elegida, perfil propio, panel de apuntes, etc. Tras descubrir las bases de datos en el grado de Ingeniería Multimedia, al finalizarlo, realicé una TPV para el sector de la restauración mediante JavaScript. Y actualmente, tras retomarlas en el master en diversas asignaturas, decidí trabajarlas en mi proyecto de fin de master ya que ambos aspectos (apps móviles y bases de datos) me motivan en el desarrollo de este.

Para ello, haré uso de Flutter, BBDD en un servidor propio en la nube contratado en Hostinger y contaré con la ayuda de la gran cantidad de documentación al alcance que hay en internet sobre Flutter, permitiendo el aprendizaje sobre este SDK y las diferentes funciones que dispone.

2.2 OBJETIVOS

Los objetivos son comprobar la aplicación de un método de producción de software extendido (que se está investigando en el Centro de Investigación del ProS), el cual sigue una arquitectura MDA:

1. Primero se estudiarán las necesidades del negocio, tomando un modelo que especifique los requisitos y necesidades del sistema de forma clara. Se detallarán los requisitos funcionales y no funcionales que, junto a la descripción del sistema, se podrán transformar en un modelo iEstrella (i*) [8] plasmando actores, metas y las tareas organizacionales a abarcar.
2. El modelo de requisitos guiará el desarrollo de un esquema conceptual mediante UML y la especificación del comportamiento del sistema.
3. El modelo conceptual guiará el código a la plataforma escogida.

Entre los modelos de requisitos, conceptuales y de código, será demostrable y visible una trazabilidad que nos lleve de uno a otro. En el capítulo tres, se detallará el método de forma mas clara indicando qué modelados, diagramas y herramientas se van a utilizar.

Como se ha mencionado, se tienen que especificar de forma clara las necesidades y objetivos del negocio:

Nos encontramos ante una empresa encargada de instalar y mantener sistemas de seguridad contra incendios en distintas instalaciones: ayuntamientos, centros escolares, tiendas, parques temáticos, restaurantes... Para ello hacen uso de distintos productos:

- Extintores
- Bocas de Incendio
- Sistemas de extinción automática de incendios
- Sistemas de detección de incendios
- Grupos de presión

Cada producto tendrá sus propias características de mantenimiento, algunas de ellas comunes para varios a la vez, por ejemplo:

- Extintores:
 - Recarga: cuando quedan bajos de presión
 - Retimbre: cada 5 años de mantenimiento
 - Revisión: cada año (puede que haya recargas o retimbres junto a esta)

Cada vez que se hace una recarga, retimbre o revisión, se necesita comprobar la presión y el peso del extintor y anotarlo.
- Bocas de Incendio:
 - Retimbre
 - Revisión
- Todos:
 - Contarán con numeración, junto a las deficiencias encontradas si las hay (en observaciones).

Cuando se realiza una instalación o mantenimiento, se genera una etiqueta según el producto que lo identifique, para saber los datos de este.

Para realizar el mantenimiento e instalación llevado a cabo por los operarios, será necesario una parte que gestione estas tareas, la administración. Sus objetivos serán los de controlar el stock (entradas, salidas, contacto con proveedores, controlar los gastos e ingresos, etc.) y gestionar las fichas de los clientes (las cuales incluirán también las instalaciones, mantenimientos y revisiones pendientes).

Las tareas organizacionales son:

- Controlar en todo momento el stock de los productos disponibles
- Contactar con los proveedores ante falta de stock
- Tener en cuenta las revisiones pendientes de ese mes
- Abrir y actualizar las fichas de los clientes: qué productos se han instalado, cuando se han instalado, cuando tienen la

próxima revisión, qué tipo de revisión toca (según los años de antigüedad de X productos), en qué sucursal se han instalado, nombre del operario que ha realizado la instalación, nombre de los operarios que han realizado el mantenimiento, y deficiencias encontradas.

- Calcular mediante las entradas y salidas de stock el gasto el ingreso en productos y ser posible la introducción de mano de obra.

Tras la aplicación del método, se pretende obtener la implementación de una aplicación que permita trabajar de forma más fácil y organizada a estos dos tipos de perfiles, ahorrando el jaleo entre distintas herramientas que se utilizan o anotaciones a mano con los problemas de cohesión y organización que ello puede conllevar:

Administradores:

- Configuran el negocio: nombre, CIF, ubicación, logo, etc.
- Configuran las propiedades: añadir trabajadores, stock, etc.
- Crearán y editarán los perfiles de los clientes: tendrán todas las opciones anteriormente descritas.
- Podrán observar la evolución de gastos e ingresos según el stock pudiendo añadir gastos o ingresos extra (sueldos, facturas...).
- Pueden manejar todas las funciones de los operarios.

Operarios:

- Podrán acceder a los perfiles de los clientes para añadir o mantener sus instalaciones y modificar los datos de estas (no los datos del cliente en sí, esto solo pueden los administradores).
- Se podrán añadir nuevas instalaciones vinculadas al perfil de un cliente: qué se ha instalado, su numeración, ubicación, etc. de forma que en la ficha del cliente sea posible ver y modificar dicha instalación (por ejemplo, en una revisión de mantenimiento).
- Añadir recordatorios de mantenimiento, estos servirán para observar con un simple vistazo que revisiones hay pendientes ese mes.
- Según el tipo de elemento instalado, que disponga de X características (si es un extintor, presión, tipo de extintor, etc. por ejemplo).
- Chat grupal entre todos los trabajadores (administradores incluidos) para notificar incidencias o lo que se quiera.
- Permitir el escaneo y realización de etiquetados (siendo posible su impresión en una impresora térmica) para los

elementos instalados, con un simple escaneo obtener su numeración, fecha de instalación, revisor, etc.

Esta herramienta ayudará tanto a los administradores como a los operarios a agilizar los procesos en el día a día del negocio al tener todo en un dispositivo móvil y en la nube, con los datos actualizados en todo momento. Para ello, será necesario:

1. Los usuarios podrán registrarse como dueños (administradores) o como trabajadores (operarios).
2. Habrá un sistema de inicio de sesión seguro.
3. Los dueños (administradores) podrán asignar trabajadores a su negocio, habilitando la conexión con la base de datos a estos, fichas de clientes, chat, etc.
4. El catálogo de productos de almacenado ("stockage") podrá ser editado en todo momento, indicando las cantidades, nombre del producto, coste, pvp, etc. y este stock se verá modificado manualmente y automáticamente según las instalaciones de estos productos.
5. La aplicación constará de una interfaz sencilla de utilizar y cohesionada.
6. Contará con un generador de etiquetas con conexión a impresoras térmicas bluetooth.
7. Habrá una conexión a una base de datos (aún por determinar, seguramente SQL mediante PHP y otras partes como el chat mediante *FireBase*) única para ese negocio. Cuando un dueño se registre y configure su negocio, se crearán tablas con un índice único para este, el cual se vinculará al perfil para poder identificar esas tablas con ese negocio en concreto.

Para producir una interfaz simple, se potenciará el uso de la similitud de los formatos en las diferentes pantallas, procurando no generar confusión debido al exceso de información o mucha navegación entre diferentes opciones, intentando separarse en mayor medida de una gran cantidad de pasos necesarios para completar una tarea. Se trabajará también en el soporte para dispositivos con grandes pantallas como tabletas, para casos en los que sea más cómodo este formato.

CAP3. ESTADO DEL ARTE

En este capítulo se introducirá el método de desarrollo *Model-Driven* (MDD) junto un estudio de las herramientas que hacen uso de él, indicando cuales son sus carencias a la hora de producir código a partir de unos requisitos y los beneficios que aporta a la hora de utilizarse.

Tras ello, elaboraremos una presentación de las herramientas a utilizar para la implementación del código final.

3.1 MODEL-DRIVEN DEVELOPMENT

A medida que aumentan las características de un producto, incluyendo la necesidad de adaptarse a distintos mercados (que varía según la localización, cultura y entorno), la producción de software se vuelve cada vez mas compleja. Además, el desarrollo software se enfrenta a la necesidad de acortar los tiempos de desarrollo a la vez que se solicitan unos mayores requisitos de calidad en el producto final.

El desarrollo basado en modelos conocido *Model-Driven Development*, es un enfoque que aspira a resolver el desafío mediante el uso de modelos como componentes de desarrollo para elevar el desarrollo de software a un nivel superior de abstracción. Este método de desarrollo utiliza modelos como especificaciones de software y los transforma hasta obtener el código fuente. MDD tiene como objetivos utilizar modelos para facilitar la visualización del futuro código y acelerar el desarrollo de software (haciéndolo más rentable), separando la lógica empresarial de la implementación [19].

La idea detrás del MDD, es poder crear un modelo del sistema y transformarlo en algo real. El sistema a desarrollar se modela desde múltiples niveles abstracción y perspectivas.

Desde hace unos años, la arquitectura basada en modelos para el desarrollo software ha ganado terreno en la ingeniería de software. Al tratarse de una nueva forma de desarrollo, hay dudas de la eficiencia y funcionalidad en su uso respecto a otros métodos mas comunes.

La arquitectura impulsada por modelos es una idea que otorga a los desarrolladores el poder de observar el ejercicio desde una forma mas general para resolverlo estableciendo unos modelos que permitan alcanzar el desarrollo del sistema.

3.2 HERRAMIENTAS MDD

MDD hace uso de modelos para representar artefactos de software, produciendo tanto modelos independientes de la plataforma como específicos de esta. En el MDD práctico, las herramientas tienen un panel importante para su desarrollo, reduciendo errores humanos y costes en el desarrollo.

En la actualidad, no existen herramientas universales que satisfagan las necesidades del MDD es por ello que cuando se habla de herramientas MDD, se habla de una colección de herramientas independientes entre sí. Al haber una gran multitud de estas, hablaremos de los diferentes tipos de herramientas que existen para el MDD:

- **Creación:** se usa para obtener modelos iniciales o editar modelos derivados.
- **Análisis:** para verificar los modelos en busca de su integridad, comprobar la consistencia, observar las condiciones en las que puede haber un error o calcular las métricas del modelo.
- **Transformación:** se utiliza para transformar modelos en otros modelos o para transformarlos en código.
- **Composición:** utilizada para componer, mediante una semántica de composición, varios modelos fuente.
- **Prueba:** para realizar pruebas a los modelos.
- **Simulación:** simulan la ejecución de los sistemas representados por modelos.
- **Metadatos:** manejan las relaciones entre diferentes modelos (metadatos y relaciones mutuas entre ellos).
- **Ingeniería Inversa:** se utilizan para transformar colecciones de artefactos software en modelos completos.

Algunas herramientas útiles para MDD: StateFlow [24], Visual Paradigm [25], Integranova [26], LabView [27] o WebRatio [28]

3.3 PROBLEMAS Y BENEFICIOS DEL MDD

En UHL [2008] y Hailpern and Tarr [2006] se discuten y presentan problemas relacionados con las aproximaciones del MDD [19][20][21]:

- **Mas experiencia necesaria:** Entender las interrelaciones entre los artefactos puede ser complicado y provocar quebraderos de cabeza sobretodo para usuarios casuales en el uso de estas. Las interrelaciones entre modelos, suelen aumentar la dificultad para los desarrolladores de modelos, estos tendrán que comprender que el impacto al realizar cambios en los modelos, afectarán en las diferentes notaciones.
- **Comparación y fusión de modelos:** todos los proyectos mantienen versiones de desarrollo de los artefactos, haciendo que la comparación entre modelos sea esencial. Comparar textos es fácil, pero comparar modelos, los cuales incluyen gráficas, diálogos, hojas de propiedades, etc. es más debido a que cuesta más ver las diferencias en términos de uso.
- **Depuración en el nivel de modelado:** cuando un usuario de una herramienta expresa los aspectos de comportamiento de una aplicación de modelos y los transforma en tiempo de ejecución, la depuración se vuelve un problema. Normalmente las herramientas de depuración de modelo solo admiten depuraciones para un modelo específico, por lo que se debería de mantener un registro de trazabilidad de qué elementos del modelo se asignan a qué elementos, en tiempo de ejecución.
- **Transformación de modelos:** cuantos mas modelos y niveles de abstracción se asocian a un sistema de software, más relaciones habrán entre los modelos. Muchas de estas interrelaciones son complejas, por lo que si un artefacto software interrelacionado cambia, afectará a determinados artefactos relacionados y la consistencia mutua no siempre se puede asegurar de forma automática, por lo que se tendrán que repasar y refinar todos los modelos afectados otra vez para asegurar su consistencia.
- **Mover la complejidad en vez de reducirla:** el técnico de desarrollo determinará si un MDD determinado reducirá la complejidad para el desarrollador o si simplemente la trasladará a otra parte del proceso. Al aumentar los artefactos software, aumenta la complejidad entre sus relaciones y en las herramientas que las visualizan o gestionan. Se tendrá que valorar si es mas eficiente gestionar cantidades mas pequeñas de artefactos grandes (menos relaciones) o cantidades mas grandes de artefactos pequeños (mas relaciones).

En cuanto a aspectos positivos, MDD proporciona los siguientes beneficios [22][23]:

- **Rapidez:** el modelo de una aplicación de software se especifica en un nivel de abstracción más alto que los lenguajes de programación tradicionales. Este modelo se transforma en una aplicación de software funcional generando código o interpretando/ejecutando el modelo. En otras palabras: cada elemento del modelo representa varias líneas de código. Por lo tanto, puede crear más funciones al mismo tiempo.
- **Mayor rentabilidad:** el tiempo de comercialización es más corto al ser más rápido. Todo esto depende del aprendizaje sobre MDD que se tengan o del modelo de negocio.
- **Empodera a los expertos en el dominio:** MDD introduce notaciones diferentes (textuales, gráficas), por lo que se puede modelar una solución software desde cualquier conocimiento del dominio a un modelo de alto nivel.
- **Soluciona la brecha entre negocios y desarrolladores:** Los expertos del dominio (analistas del negocio) se involucran más en el desarrollo.
- **Más sólido a cambios de los requisitos del negocio:** uno de los problemas en el desarrollo software es que los requisitos pueden cambiar de forma rápida. MDD proporciona la solución ya que al proporcionar un desarrollo rápido, permite un cambio más rápido en las aplicaciones.
- **Más sólido a cambios tecnológicos:** la evolución de las tecnologías crece de forma rápida. MDD permite no tener que cambiar el modelo por completo, simplemente cambiar el interpretador o generador de código, permitiendo adaptar los modelos antiguos al nuevo código.
- **Refuerza la arquitectura:** con el uso de MDD, se garantiza cumplir con la arquitectura escogida, estandarizando la tecnología a utilizar en esa arquitectura, ya que los principios de la arquitectura en el MDD se ven reforzados.
- **Permite enfocarse en problemas del negocio:** permite enfocarse en cómo resolver estos problemas mediante la TI en vez de enfocarse en la tecnología.

3.3 DETALLES DEL MÉTODO

Como se ha especificado anteriormente, se aplicará MDA en la arquitectura. El aspecto fundamental de la *Model Driven Architecture* es su habilidad en el control de todo el ciclo de desarrollo del proyecto, cubriendo el análisis y el diseño, la programación y las pruebas.

Los objetivos de la MDA son la portabilidad, reusabilidad e interoperabilidad. Al aplicar MDA, se dividirá el proyecto en tres modelos y las transformaciones necesarias que llevan de uno a otro, los modelos son los siguientes:

- **CIM (*Computation Independent Model*):** es aquí donde se modelan los requisitos que el sistema debe cumplir, definiendo bajo qué condiciones se utiliza el sistema. La MDA indica que el CIM deberá ser convertible a PIM y este PIM, en PSM.

Para este proyecto, se va a incluir en el CIM iEstrella, un modelo organizacional basado en metas que nos permite plasmar de forma sencilla cuales son los objetivos organizacionales del negocio, cuyos requisitos se verán representados en un modelo de Casos de Uso, indicando las transformaciones, similitudes y pérdidas desde el modelo en iEstrella en el que se basa.

- **PIM (*Platform Independent Model*):** muestra el nivel de independencia de plataforma necesario para poder ser utilizado en diferentes plataformas de implementación, es decir, tiene que tener un nivel de abstracción invariable representando la lógica del sistema y su interacción real sin detallar la tecnología que la implementará.
- **PSM (*Platform Specific Model*):** nos devuelve una vista del sistema para una plataforma determinada combinando el PIM con información detallada sobre cómo el sistema utiliza una plataforma en concreto.

En el proceso de MDA la idea principal es transformar niveles superiores (CIM, PIM) en niveles inferiores (PSM) que se utilizan para crear el código a implementar. La transformación de un modelo es un proceso que dado un modelo el cual se escoge como fuente, convertirlo a otro, este proceso de en medio es conocido como transformación. El esquema MDA sería el mostrado en la siguiente figura.

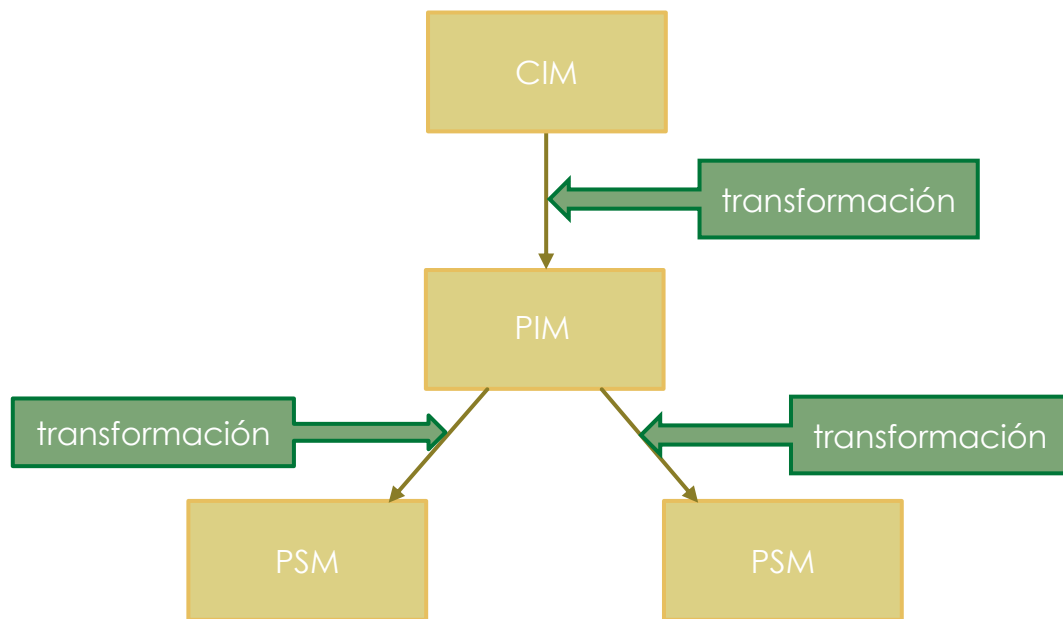


figura 1: esquema MDA

Las transformaciones describen cómo los elementos del modelo origen se transformarán en los elementos del modelo destino. Ambas partes pueden describirse mediante variables, plantillas, reglas lógicas, etc. [13][14][15]. Estas transformaciones pueden darse en horizontal dentro del mismo nivel si se requieren varios modelos (CIM-CIM, PIM-PIM) o en vertical (CIM-PIM, PIM-PSM), estas son las más comunes y son las llamadas como Transformación de Modelo (*Model Transformation*) en MDA [16].

3.4 TECNOLOGÍAS PARA LA IMPLEMENTACIÓN

3.4.1 Flutter

Flutter se trata de un SDK o *framework* desarrollado de la mano de Google y de código abierto el cual nos permite crear aplicaciones móviles y de escritorio de forma sencilla y rápida. Utiliza el lenguaje "*dart*" y el código que genera es nativo para todas las plataformas, es decir, que el rendimiento y la experiencia de usuario será idéntica a las aplicaciones nativas que se han desarrollado tradicionalmente. En este caso, se usará Flutter mediante la herramienta de Android Studio.

Como hemos mencionado antes, se compilan de forma nativa y multiplataforma con un solo código, es decir, un mismo código nos sirve para una aplicación en Android e iOS con las ventajas que ello conlleva.

Pros:

- Admite cambios en el código con observaciones directas sin necesidad de recompilar, conocido como Hot Reload
- Un mismo código sirve para distintas plataformas
- Utiliza Material Design de Google y Cupertino de Apple

- Permite accesos a las funciones nativas del dispositivo

Contras:

- Se necesita aprender *Dart*
- Herramienta aún joven
- Librerías algo limitadas debido a su poco tiempo en el mercado
- Las aplicaciones de escritorio son algo inestables

3.4.2 Server

El servidor escogido ha sido *Hostinger* [10], este nos permite la creación de bases de datos MySQL y almacenar en el gestor de archivos los PHP para poder realizar las conexiones desde la aplicación en todo momento. Ha sido escogido debido a que ya estoy familiarizado con él, ha su coste y a sus prestaciones, que mantienen una buena relación calidad-precio adaptándose a los requisitos que el proyecto necesita.

3.4.3 Bases de Datos MySQL

Este tipo de bases de datos relacionales serán utilizadas para el manejo de los datos de la aplicación casi en su totalidad. Ha sido escogido por sus prestaciones, solidez y cantidad de documentación alojada en la red. También porque ha sido el tipo de bases de datos que más he visto a lo largo del desempeño de mis estudios.

En nuestro caso, accederemos a estas bases de datos mediante PHP y se alojará en nuestro servidor.

3.4.4 Bases de Datos FireBase

Las bases de datos FireBase [11] nos permiten la observación de la actualización de los datos en tiempo real sin necesidad de re-consultar constantemente, esto nos proporciona gran fluidez para observar cambios en tiempo real.

Usaremos FireBase en el proyecto para la implementación de las pantallas que requieran de este tipo de bases de datos y velocidades, como por ejemplo, el chat entre trabajadores de la empresa.

3.4.5 PHP

PHP es un lenguaje de código abierto desarrollado de forma específica para el desarrollo web y el cual nos permite incrustarlo en ficheros HTML. En nuestro caso, utilizaremos PHP sobretodo para conectar con la base de datos y el servidor.

Se trata de un lenguaje que nos facilita el dinamismo y la necesidad de responder a las peticiones del proyecto de forma instantánea o en tiempo real.

Una de las ventajas de este tipo de ficheros es que no se puede acceder de forma externa al código del PHP ya que él se limita a interpretar las peticiones y réplicas del servidor.

3.4.6 Sketch

Sketch [12] es un programa desarrollado para los sistemas operativos *Macintosh* centrado en el diseño de gráficos vectoriales, de esta forma, nos ahorraremos la necesidad de hacer una búsqueda de imágenes o recursos que requieran de derechos de autor.

También, gracias a la cantidad de *plugins* gratuitos de los que dispone (de iconos por ejemplo), podremos alcanzar de una forma mas sencilla la cohesión visual de los elementos que compongan las distintas pantallas del proyecto.

CAP4. ESPECIFICACIÓN

4.1 REQUISITOS DE USUARIO

Los requisitos de usuario se han obtenido mediante el administrador de una empresa de gestión de sistemas de emergencias. Con estos requisitos conseguiremos hacernos una idea de todas las opciones que será necesario cubrir con la implementación de la herramienta. Hay dos tipos de requisitos: funcionales y no funcionales.

De aquí en adelante, se tratará a los trabajadores como operarios y a los dueños como administradores, para facilitar la comprensión en cuanto al tipo de empresa en el que nos encontramos. Los administradores serán, como hemos mencionado anteriormente, los encargados de crear el negocio incluyendo sus propiedades, etc. y manejando el stock, pudiendo añadir los trabajadores (operarios) registrados en la aplicación, vinculándolos a su negocio

4.1.1 Requisitos Funcionales

Los requisitos funcionales son los que especifican el manejo que requiere el sistema y las funciones que debe incluir.

Usuarios sin registrar:

- F1.** El uso de la aplicación requerirá que los usuarios tengan que estar registrados en la plataforma.
- F2.** Los usuarios sin registrar podrán registrarse desde la pantalla de inicio como operario o administradores.
- F3.** Se podrán registrar como operarios o administradores.

Usuarios registrados:

- F4.** Contarán con los datos necesarios para su identificación y creación de un perfil con identificación única.
- F5.** El menú inicial de la aplicación variará según el tipo de usuario que haya iniciado sesión (operario o administrador), desde él accederán a las opciones disponibles de la aplicación.
- F6.** Podrán acceder a un chat grupal los usuarios que estén vinculados al mismo negocio.
- F18.** Acceder a un calendario de revisiones el cual permita observar las revisiones necesarias ese mes o otros meses anteriores.

Usuario tipo administrador:

- F7.** Son los encargados de especificar las características del negocio, siendo estas editables solo por ellos.
- F8.** Tendrán una opción en el menú inicial para poder editar las características del negocio.
- F9.** Podrán observar la evolución del stock y obtener gráficas de gastos e ingresos.
- F10.** Son los encargados de introducir los productos junto sus características en el sistema.
- F14.** Los productos introducidos por los administradores podrán tener diversas características preestablecidas (por ejemplo, si registras un producto tipo extintor, tendrá unas características diferentes a las de un sistema de detección de incendios).
- F15.** Las estadísticas se calcularán de forma mensual y se compondrán por el movimiento de stock (entradas y salidas), salarios e ingresos o gastos extra pudiendo ser introducidos de forma manual.
- F19.** Se encargarán de la gestión de los clientes: darlos de alta, editarlos o darlos de baja.
- F20.** Podrán añadir trabajadores al negocio: darlos de alta (vincularlos), editarlos o darlos de baja.
- F21.** Tendrá la capacidad de anotar incidencias comunicadas por el cliente.

Usuario tipo operario:

- F11.** Tras registrarse, no podrán realizar ninguna acción hasta no quedar vinculados a un negocio.
- F12.** Tendrán opciones limitadas propias de los administradores: solo podrán ver el stock y no editarlo manualmente, solo podrán ver los clientes pero no editarlos (sí que pueden añadir instalaciones y hacer las revisiones), no podrán editar las propiedades del comercio y no podrán ver las estadísticas del comercio.
- F13.** Tendrán ligados a su perfil un sueldo, el cual servirá para la realización de las estadísticas.
- F16.** Cuando los operarios realicen una instalación o mantenimiento, quedarán vinculados a un perfil del cliente, pudiendo observar los distintos mantenimientos realizados anteriormente.
- F17.** Se permitirá la creación y escaneado de códigos de barras para identificar los productos registrados.

4.1.2 Requisitos No Funcionales

Son los requisitos que especifican las características del sistema, por ejemplo, seguridad, lenguaje utilizado, etc.

- NF1.** El desarrollo se hará con Flutter
- NF2.** Cuando un usuario quede registrado, accederá a la herramienta con un usuario y contraseña.

- NF3.** Los usuarios podrán mantener sus datos almacenados mediante *Shared Preferences* para evitar reintroducir los datos. Estos datos se almacenarán de forma segura.
- NF4.** Será una aplicación multiplataforma iOS e Android.
- NF5.** Al tratarse de una aplicación en la nube, será necesaria conectividad a la red constantemente para poder utilizar las funciones.
- NF6.** La conectividad con el server y MySQL se realizará mediante PHP.
- NF7.** Se utilizará un hash SHA-256 para almacenar las contraseñas en los servidores.
- NF8.** El lenguaje de programación para la implementación será *Dart*, el utilizado en Flutter.
- NF9.** Las bases de datos serán de tipo MySQL, utilizando consultas SQL para su manejo.
- NF10.** Se incluirán técnicas para evitar *SQL Injection* en los accesos y modificaciones de la base de datos.

4.2 DESCRIPCIÓN DEL SISTEMA

Tras estudiar las tecnologías tradicionales, actuales y los requisitos de usuario, se va a proceder a la realización de una herramienta para dispositivos Android e iOS que proporcione a este negocio una solución que facilite su gestión mediante una interfaz más atractiva y práctica. Se pasará de la utilización de programas de escritorio a una única herramienta, ahorrando costosos mantenimientos y actualizaciones.

Cuando un usuario nuevo desee usar la aplicación, procederá a realizar el registro en el sistema como operario o administrador.

- El administrador (propietario) pasará a continuación a configurar los atributos de la empresa (nif, dirección, teléfono de contacto, etc.), tras ello, podrá iniciar sesión y observar el menú inicial con todas las opciones (las suyas propias y las del operario), también podrá dar de alta a operarios, asignándoles un sueldo bruto para las estadísticas.
- El operario (trabajador) quedará registrado en el sistema a espera de quedar vinculado a un negocio. Una vez ha sido vinculado, este observará tras iniciar sesión, el menú de la aplicación con las opciones (solo las del operario).

El menú inicial dispondrá de las siguientes opciones:

- **CLIENTES**
 - Permite la creación y edición de los clientes por parte de los administradores.
 - Se tratará de una lista de los clientes dados de alta por los administradores, cada uno de ellos contará con una ficha la cual incluirá las instalaciones y

mantenimientos realizados, un listado de los productos que componen la instalación y sus características.

- Cuando un operario realice un mantenimiento o instalación, los datos de la ficha se actualizarán automáticamente.
- Podrá anotar incidencias comunicadas por el cliente.
- Tras la selección de un perfil, se podrá observar sus instalaciones o sucursales. Cada una de ellas contendrá la información de los productos instalados y su estado.
- Cada producto generará el código de barras imprimible mediante una impresora bluetooth para facilitar la identificación en el taller.

- **ALMACÉN (STOCK)**

- Se encarga de las salidas y entradas de stock.
- Permitirá dar de alta nuevos productos, indicando su tipología, nombre, modelo, coste, pvp, etc.
- Las salidas de stock se actualizarán de forma manual o automáticamente a través de las instalaciones o mantenimientos.

- **TALLER (MANTENIMIENTO)**

- Dispondrá de un lector de código de barras, con un escaneado se indicará qué producto es, de qué instalación, qué mantenimiento le toca, etc.
- Pone de forma fácil y al alcance de los usuarios los mantenimientos que tocarían ese mes
- Permitirá la selección de cualquier producto de forma manual si no se dispone de una etiqueta.
- Una vez realizado el mantenimiento permitirá guardar los cambios realizados al producto para que se vea tras un nuevo escaneado, los cambios realizados.
- Un operario puede seleccionar un mantenimiento o escanear el código de barras de un producto del lugar a realizar el mantenimiento, de forma que accederá a la ficha del cliente cuyo mantenimiento es necesario para poder modificar las tareas realizadas.

- **CHAT**

- Todos los usuarios vinculados a un mismo negocio tendrán acceso a un chat grupal para comentar de forma rápida cualquier problema.

- **ESTADÍSTICAS**

- Solo los administradores tendrán acceso a esta opción.

- Con solo acceder, se podrá observar una gráfica de los gastos derivados del stock, salarios y gastos o ingresos extra.
- Las estadísticas serán mensuales, siendo posible observar los gastos de meses anteriores.
- **AJUSTES**
 - Solo los administradores tendrán acceso a esta opción.
 - Permitirá el cambio en cualquier momento de los atributos del negocio.
 - Desde esta pantalla permitirá añadir trabajadores y modificar sus datos como el salario.

Las tablas de la base de datos serán únicas para cada negocio identificándose mediante un código único compuesto por números y letras gracias a las funciones UUID. Estas tablas se generan cuando un administrador se registre y configure los datos del negocio. Dentro de estas tablas de los propios negocios, se almacenarán los usuarios con su código único para poder identificarlos y averiguar de qué tipo son para habilitar o no las opciones a las que pueden acceder.

CAP5. DESARROLLO DEL PROYECTO

Como se ha especificado anteriormente, se va a verificación de la aplicación de la metodología en MDA. Para ello, primero se tendrán que establecer qué modelos representarán el nivel del CIM y cuales los del PIM, estos se presentan a continuación. Tras ello, se discutirán los beneficios de su uso y si se han encontrado pérdidas entre modelos.

5.1 CIM (ANÁLISIS DEL SISTEMA)

El CIM es un nivel que principalmente es representado por modelos de negocio, en este ámbito, nos encontramos una gran variedad de notaciones, entre ellos, el modelado en iEstrella (i* en adelante), el cual podremos modelar gracias a los requisitos y necesidades organizacionales ya especificadas en puntos anteriores. Tras el i*, haremos una transformación CIM-CIM para obtener los Casos de Uso del sistema, señalando los cambios para pasar de un modelo a otro.

5.1.1 Modelado i* (iStar - iEstrella)

El modelado en i* fue introducido para rellenar el vacío en los lenguajes típicos de modelado. Este modelo i* se centra en las dimensiones intencionales (¿por qué?), sociales (¿quiénes?) y las estratégicas (¿cómo?). Este modelo captura los requisitos organizacionales y los agrupa mediante metas y tareas, de forma que se permite ver el flujo entre actores y sus necesidades organizacionales mediante las tareas que le llevarán al cumplimiento de estas.

Pasaremos a documentar los distintos actores que intervienen junto a sus enlaces y dependencias, en este caso, encontramos:

- Un agente: Empresa de Sistemas Contra Incendios.
- Dos roles: Administradores y Operarios de la empresa.
- Dos actores: Clientes (actor secundario, no hace uso literal de la herramienta) y la herramienta en sí.

Los enlaces principales entre actores se indican en la Figura 2:

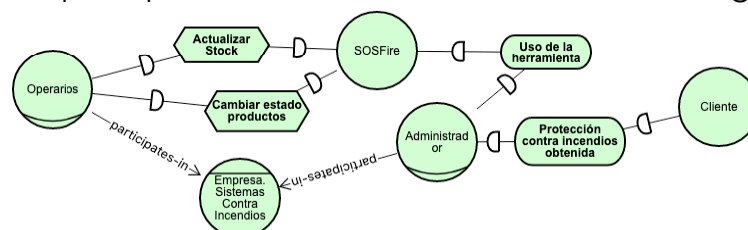


figura 2: i* resumen actores

Un cliente solicita unos servicios al administrador, el cual es un rol de la empresa. Para resolver las tareas que el cliente solicita, hará uso de la herramienta SOSFire junto a los operarios. Los objetivos y tareas que los actores desean conseguir para la informatización de la empresa a tecnologías actuales son las siguientes.

5.1.1.1 Funcionamiento en la nube:

- Este es el objetivo principal de la herramienta, tratará de adaptar y mejorar el sistema informático actual local de escritorio a tecnologías más modernas, eliminando las limitaciones actuales.
- Actualmente todo el sistema se trabaja de forma local. Se debe de presentar una solución que ofrezca portabilidad y seguridad.
- La portabilidad la obtendremos mediante el uso de PHP y el desarrollo con Flutter y el lenguaje Dart para dispositivos móviles como *Smartphones* y *Tablets*.
- La persistencia del proyecto se compone con el uso de MySQL y FireBase en el manejo de los datos.

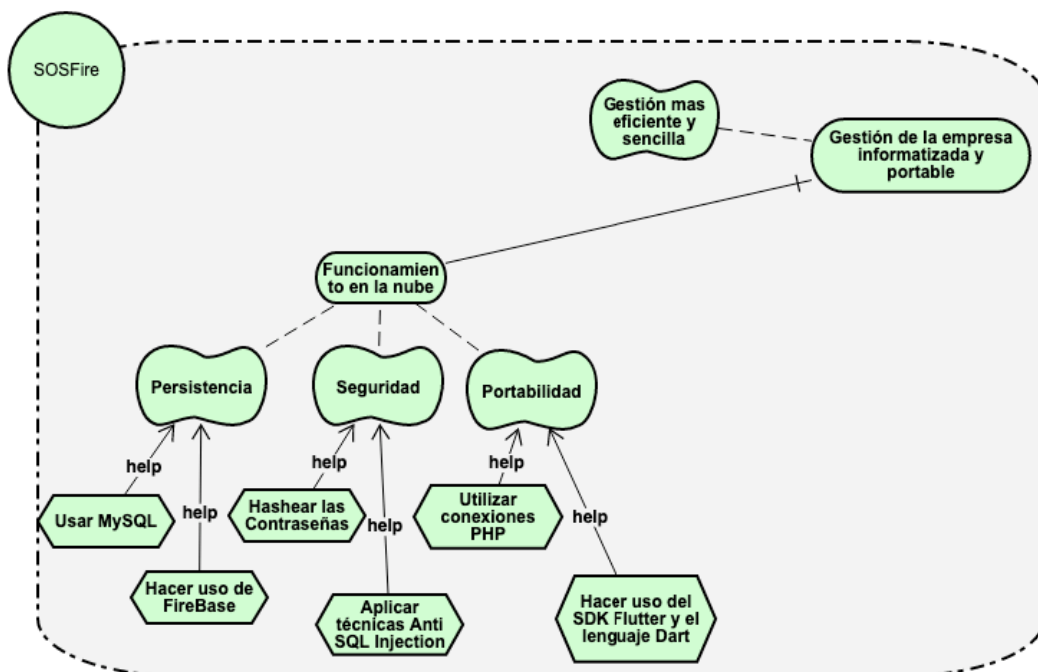


figura 3: i* funcionamiento en la nube

5.1.1.2 Controlar el Stock

- o Uno de los objetivos de este negocio es facilitar el control en todo momento del stock disponible, mejorando el mantenimiento de los productos debido a la facilidad para identificarlos y unas estadísticas más fáciles de elaborar.
- o Este control de stock se puede realizar de forma manual o automática. Esto último dependerá del mantenimiento o instalación que realice el operario, al introducir los productos utilizados en la ficha de instalación en la sucursal del cliente, el stock disponible variará, generando de forma automática el flujo de productos y por ende, controlar los gastos o ingresos que derivan de estos.
- o El administrador podrá dar de alta productos o editar el inventario de forma manual.

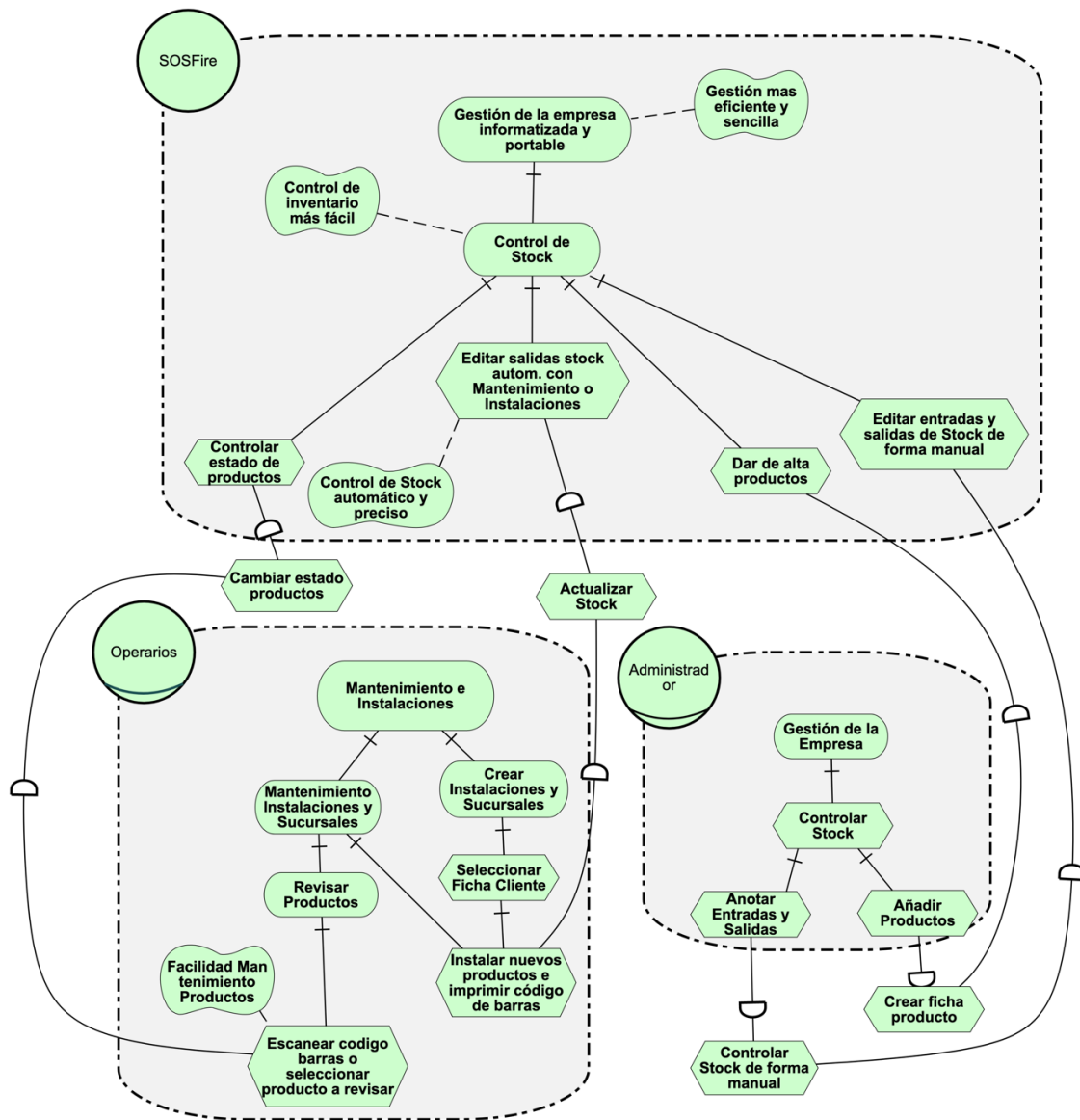


figura 4: i* controlar stock

5.1.1.3 Realizar la contabilidad

- o Una de las finalidades es que mediante la variación del stock se pueda obtener una estadística de los gastos e ingresos mensuales del negocio en materia de productos comprados y vendidos.
- o La contabilidad podrá observarse mediante el uso de gráficas circulares, dividida por gastos de stock, ingresos de stock, salarios, gastos extra e ingresos extra (estos últimos añadidos manualmente).
- o Al tratarse de información más sensible, solo tendrán acceso a esta pantalla los miembros de la administración.

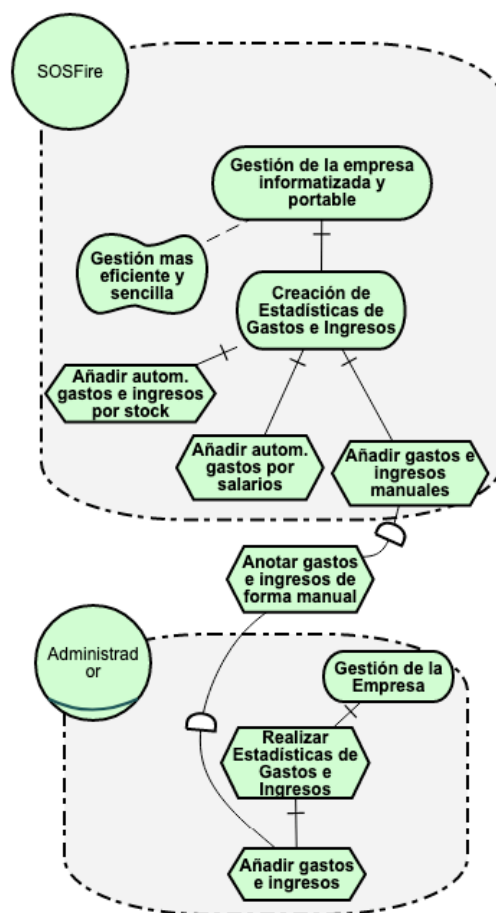


figura 5: i* realizar contabilidad

5.1.1.4 Gestión de clientes

- Cuando un cliente solicita un servicio y es nuevo, el administrador lo registra en la plataforma junto a sus sucursales.
- Mediante la sucursal abierta con la herramienta, los operarios al realizar instalaciones o mantenimiento, editarán esta sucursal o instalación y el stock variará automáticamente.
- Con estos objetivos la empresa obtendrá una cualidad de mejorar los servicios de mantenimiento, ya que con un simple vistazo se observarán las instalaciones o mantenimientos previstos vinculados a los perfiles de los clientes.

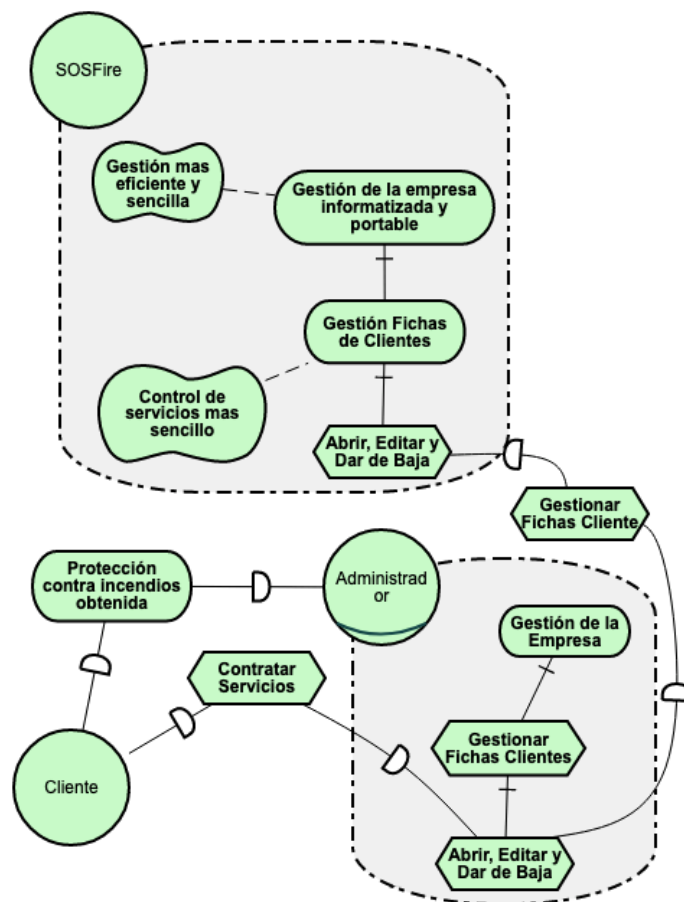


figura 6: i* gestión clientes

5.1.1.5 Mantenimientos Pendientes

- Cuando un producto instalado en una sucursal cumple un año desde su instalación, es necesaria una revisión. Se aportará la tecnología que se encargue de notificar de forma automática tras la instalación de un producto, las revisiones y mantenimientos pendientes.
- El administrador podrá anotar en cualquier momento una incidencia que el cliente notifique, quedando reflejada y al alcance de los operarios para poder resolverla.

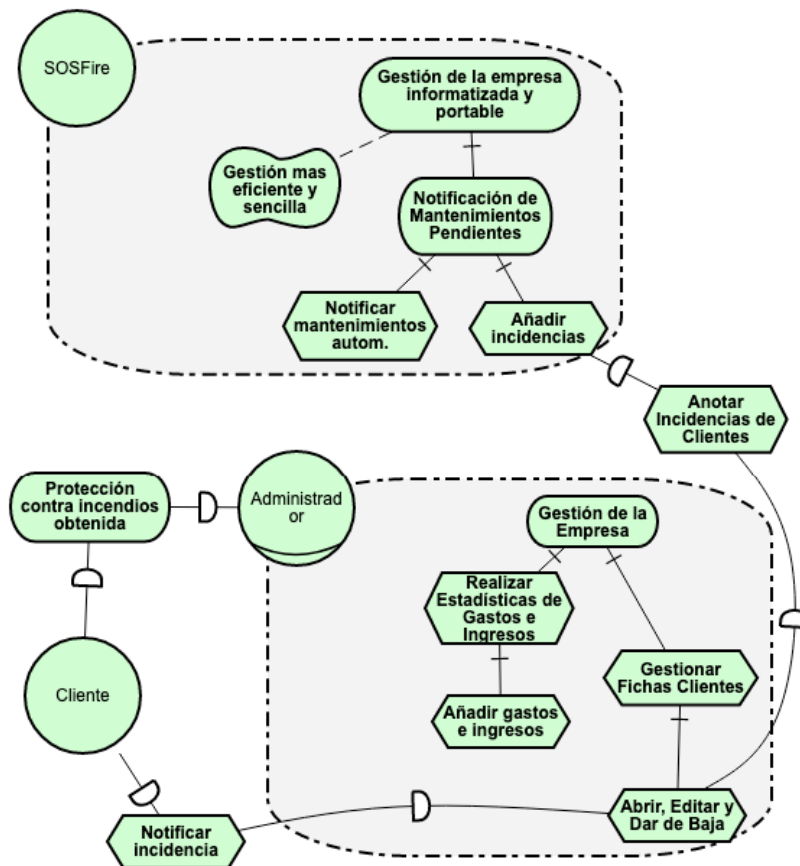


figura 7: i* mantenimientos pendientes

5.1.1.6 Control de Propiedades

- El administrador podrá editar las propiedades del negocio (CIF, nombre, dirección, etc.) siendo estos cambios reflejados instantáneamente.
- Incluirá la lógica que permita la vinculación de trabajadores (usuarios registrados) al negocio, indicando su rol y salario bruto (el cual computará en la elaboración de estadísticas mensual).

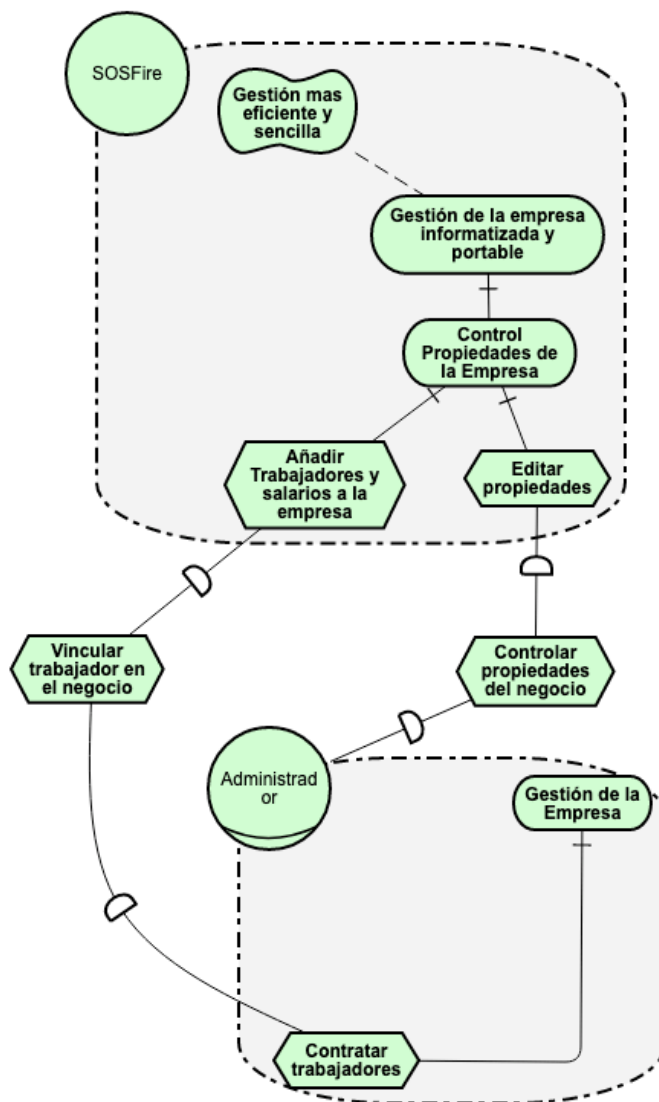


figura 8: i* control de propiedades

Estos han sido los objetivos organizacionales principales de la organización, poder gestionar todo desde cualquier lugar y de forma sencilla. El conjunto de todos los objetivos tiene la siguiente forma:

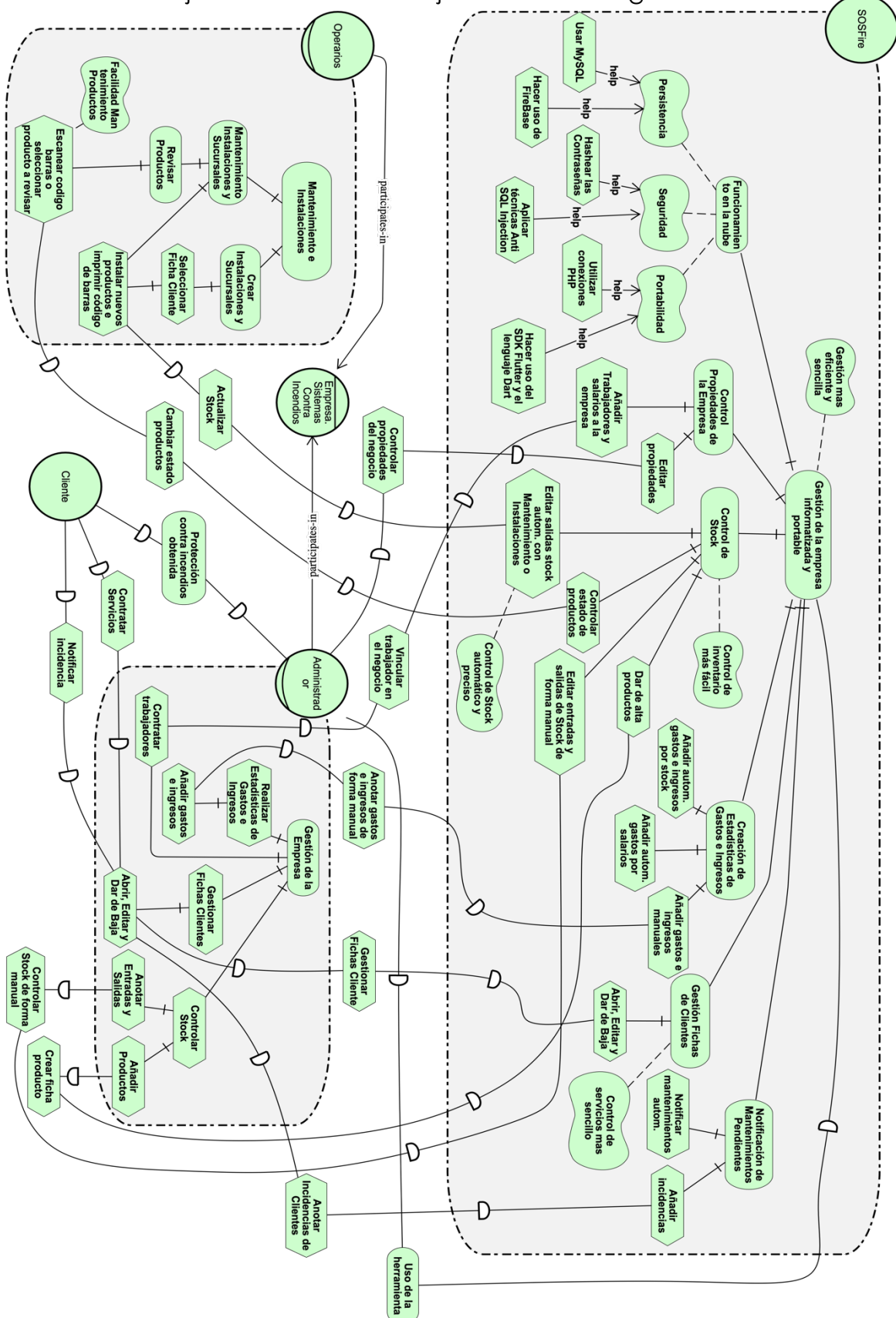


figura 9: i* modelo completo

Una vez realizada la descripción con el modelado i^* , será necesaria una transformación al modelo de Casos de Uso donde se observen los cambios en su estructura (ya que tal vez en el nuevo modelo falten o se añadan cosas debido a la tipología en sí del método).

5.1.2 Casos de Uso

El modelado de Casos de Uso (MCU en adelante) trata de describir cuales son las acciones que la herramienta puede llevar a cabo por parte de los actores, es decir, describe cual debería de ser el funcionamiento del sistema dados unos requisitos funcionales. [17]

Los casos de uso precisan de tres elementos:

1. Actores: son entidades externas, existen fuera del sistema y cumplen un rol específico como usuario.
2. Casos de uso: óvalos que contienen las acciones y funciones del sistema.
3. Sistema: rectángulo que delimita el sistema con los casos de uso dentro y los actores fuera de él.

T1: Respecto a los actores, agentes y roles, mientras que en i^* aparecen todos los actores que intervienen en la organización, en el MCU solo los que cumplen un rol en el sistema. El agente Empresa de Sistemas Contra Incendios no es necesario ya que en sí no cumple ninguna acción específica del sistema. El actor Cliente descrito en el modelo i^* , se trata de un actor secundario usado como recurso para la correcta descripción de las necesidades organizacionales de la herramienta, pero al no estar involucrado en la herramienta en sí (no es partícipe ni tiene ningún rol asignado dentro de esta) no aparecerá en el MCU.

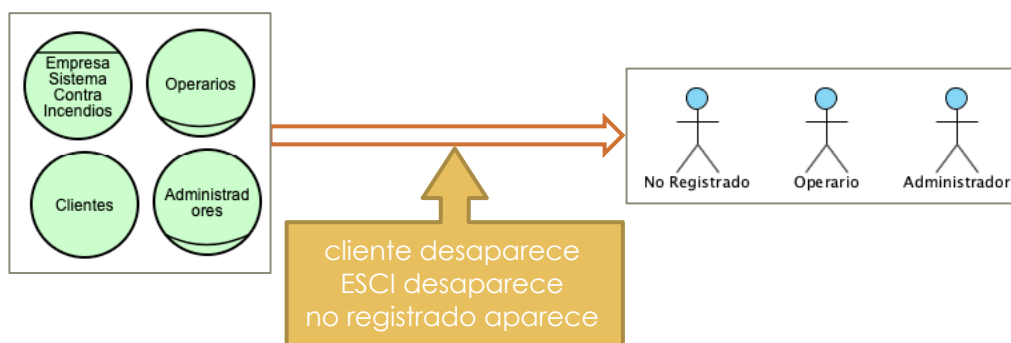


figura 10: T1 casos de uso

Actores del MCU:

- Usuarios sin registrar.
- Usuarios de tipo operario.
- Usuarios de tipo administrador.

Como los Administradores pueden realizar todas las tareas de un Operario del sistema, en el MCU los Administradores heredarán de los Operarios, esto no se puede mostrar en el i^* de forma literal.

T2: Los SoftGoals, que en i^* sirven para especificar metas donde no se tiene claro el criterio de cómo cumplirlos, se ven eliminados del MCU. En este caso concreto, se han mostrado representando parte de los requisitos no funcionales (uso de Flutter, PHP, etc.) por lo que en los Casos de Uso (CU) no van a aparecer al no ser necesarios para su especificación. Se podría relacionar con el caso de uso Registro, que incluye varios de los aspectos de estas SoftGoals (usa MySQL, hashlea las contraseñas, evita SQL Injection, PHP...)



figura 11: T2 casos de uso

T3: Los Casos de Uso que aparecerán en el MCU serán los del sistema, en el caso de i^* , se ven reflejadas las metas que cumplen los actores y roles que componen la organización, en el caso del MCU, solo interesarán las del sistema por lo que varias metas y tareas desaparecen.

T4: Otros de los cambios mas notorios entre ambos modelos, es el uso de las tareas en el i^* . En este, aparecen como acciones para conseguir llegar al objetivo de cumplir la meta, en el MCU, esto no sucede, ya que se especifican directamente las acciones que el sistema realiza, por lo que podríamos establecer una similitud entre Casos de Uso en el MCU con las Metas finales del i^* . Estas modificaciones entre modelos no son literales ya que lo que puede suponer una tarea organizacional menor en el i^* (como una tarea), puede aparecer como un Caso de Uso.

T5: Para la Gestión de Productos, Fichas de Cliente y Trabajadores, se comparten las acciones de Abrir, Editar y Dar de Baja. En MCU, estos casos de uso serán compartidos ya que la lógica es similar en todos ellos, a diferencia de en i^* , que cada meta o tarea la tiene como propia.

T6: Algunas tareas en i^* se verán reflejadas en los flujos normales o principales detallados en los casos de uso.

A continuación, se detalla una tabla con las transformaciones resumidas necesarias de i^* a MCU y después, el resultado de aplicarlas.

i^*	T	MCU
Agentes, roles y actores involucrados en toda la organización	T1	Actores que tienen un rol en el sistema
SoftGoals representando Requisitos No Funcionales de la organización	T2	Caso de Uso Registro que abarca parte de ellos
Metas y tareas de toda la organización	T3	Aparecen Casos de Uso de solo el sistema
Uso de tareas para cumplir metas	T4	Acciones como Casos de Uso, pueden ser Metas o Tareas según su lógica
Tareas de Añadir, Editar, Dar de baja por cada meta	T5	Casos de Uso compartidos para acciones que tienen una lógica similar
Tareas para cumplir metas	T6	Acciones en los flujos principales

Diagrama final de Casos de Uso tras aplicar las transformaciones:

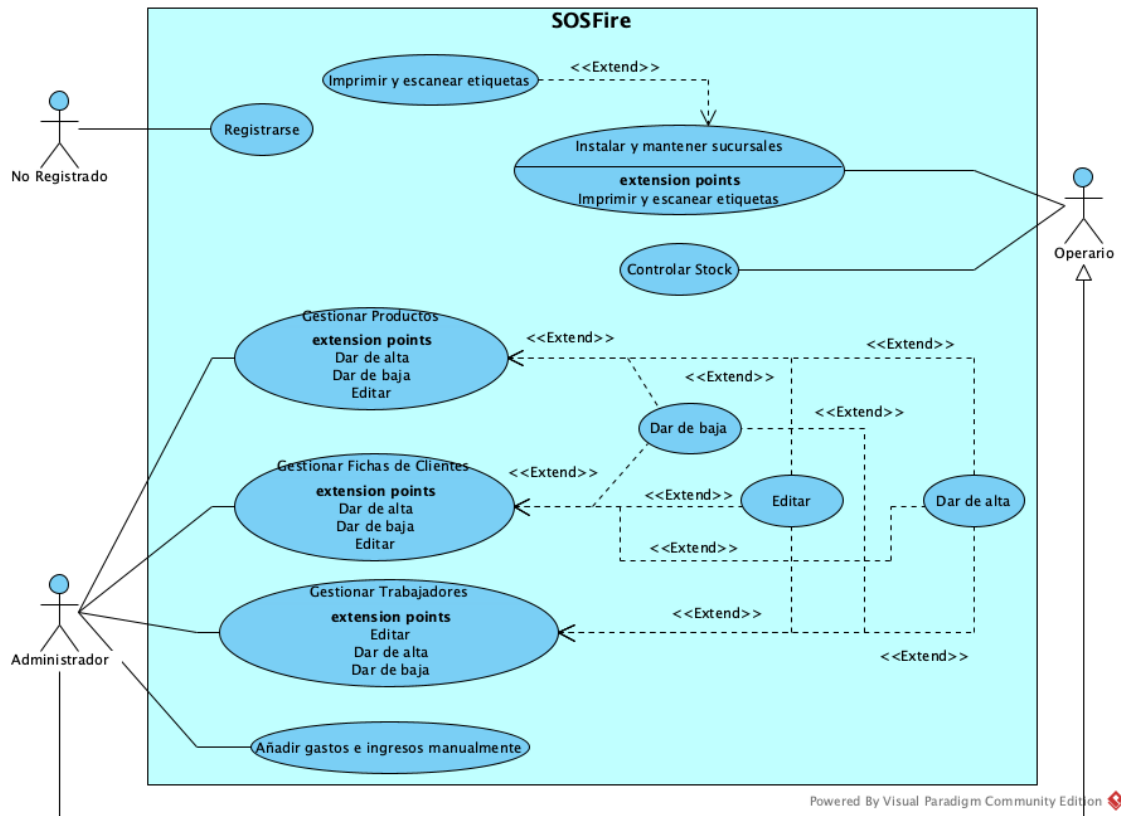


figura 13: diagrama final casos de uso

A continuación, detallaremos cada Caso de Uso, relacionándolo con la parte correspondiente en el modelo en i* incluyendo los Requisitos Funcionales que abarca junto los Flujos Normales o Principales (FN, el correcto desarrollo del caso de uso) y los Flujos Alternativos (FA, posibles excepciones y errores).

En la mayoría de casos de uso puede surgir un Flujo Alternativo el cual es carecer de internet y su excepción al no poder establecer conexión con el servidor, por lo que se va a omitir en todos ellos para evitar repetición.

5.1.2.1 Registrarse

Este Caso de Uso es llevado a cabo por los Usuarios Sin Registrar. Las acciones a realizar son rellenar las casillas de los datos requeridos según su tipo de rol escogido. Si el rol es Administrador, se incluirá un registro del comercio donde se añadan sus propiedades.

Requisitos Funcionales: 1, 2, 3, 4, 7

i*: 5.1.1.1 Funcionamiento en la Nube

FN:

1. Se accederá al registro desde el menú principal.
2. Aparecerá un menú para escoger el tipo de usuario.
3. Una vez escogido, se rellenarán los datos del usuario.
 - a. Si escoge administrador, irá a otra pantalla para registrar los datos del negocio y finalizará el registro.
 - b. Si el usuario es operario, finalizará el registro.
4. La herramienta realizará las comprobaciones de los datos, evitando usuarios duplicados.
5. Se ha completado el registro, ya se puede hacer Login

FA:

FN4. La herramienta notifica que faltan datos o el usuario ya existe, requiriendo introducir de nuevo los datos corregidos.

5.1.2.2 Gestionar Productos

El Administrador podrá controlar los productos que tengan disponibles en el negocio. Este Caso de Uso se ve extendido por otros tres los cuales se comparten con otros casos de uso por sus semejanzas:

- Dar de alta: elegirá un tipo de producto y lo configurará.
- Editar: permitirá la edición de un producto dado de alta.
- Dar de baja: eliminar un producto dado de alta.

Requisitos Funcionales: 10, 14

i*: 5.1.1.2 Control de Stock

FN:

1. Desde el menú inicial, el administrador accederá a la pantalla de Almacén
2. Seleccionará una opción:
 - a. Dar de alta: elegirá el tipo de producto y lo configurará
 - b. Dar de baja o editar: pulsa sobre un producto dado de alta y lo modifica (junto su stock) o elimina.
3. Se guardan los cambios, observables por cualquier usuario vinculado al negocio.

FA:

FN2b. No hay productos dados de alta, el administrador debe darlo de alta antes.

5.1.2.3 Gestionar Fichas de Clientes

Las Fichas de los Clientes y sus sucursales podrán gestionarse por parte del Administrador, como en el punto anterior, contará con tres casos de uso extendidos:

- Dar de alta: rellenará las propiedades del cliente (nombre, dirección, provincia, teléfono, etc.), tras ello, también podrá dar de alta las sucursales del cliente.
- Editar: permitirá la edición de un cliente o una de sus sucursales dada de alta.
- Dar de baja: otorga la función de eliminar un cliente o sucursal dada de alta.

Requisitos Funcionales: 19, 21

i*: 5.1.1.4 Gestión de Clientes

FN:

1. Desde el menú inicial, el administrador accederá a la pantalla de Clientes.
2. Seleccionará una opción:
 - a. Dar de alta: especificará los datos del cliente.
 - b. Dar de baja: pulsará sobre un cliente, editarlo y eliminarlo.
 - c. Editar: pulsará sobre un cliente dado de alta y gestionar sus sucursales, instalaciones, incidencias o pulsará en el botón de edición para editar sus datos.
3. Se guardarán los cambios, observables por cualquier usuario vinculado al negocio.

FA:

FN2bc. No hay clientes dados de alta, se debe hacer antes.

5.1.2.4 Gestionar Trabajadores

Existirá a disposición del administrador, la lógica suficiente como para que sea capaz de vincular trabajadores a su negocio, asignando su salario bruto, editarlo y darlo de baja. Cuenta con los mismos casos de uso comunes salvo el de dar de alta, que es diferente. Estos son:

- Dar de alta trabajador: vincula trabajador al negocio.
- Editar: permite editar el salario y rol del trabajador (pudiendo hacerlo administrador o operario en cualquier momento) .
- Dar de baja: desvincula al trabajador del negocio.

Requisitos Funcionales: 8, 11, 13, 21

i*: 5.1.1.6 Control de Propiedades

FN:

1. El Administrador accederá a la pantalla de Ajustes (no anotada como caso de uso por su similitud con el caso de uso Editar de los puntos anteriores).
2. Pulsará sobre el botón TRABAJADORES
3. Se mostrará la pantalla de trabajadores, y estas opciones:
 - a. Dar de alta Trabajador: para vincular un trabajador al negocio. Se introduce el correo electrónico del trabajador el cual recibirá un código de verificación. Se añade el sueldo (puede ser cero), se anota el código de verificación y se vincula (predeterminadamente con rol Operador).
 - b. Editar o dar de baja un trabajador: se permite editar el sueldo y rol (Operador-Administrador) de un trabajador, también es posible darlo de baja, desvinculándolo del negocio.
4. Se guardarán los cambios:
 - a. El trabajador quedará vinculado al negocio, ya puede iniciar sesión y acceder a los datos del negocio (correspondientes al rol asignado).
 - b. El trabajador tendrá el rol editado, perdiendo o ganando privilegios según el cambio.
 - c. El trabajador quedará desvinculado, perderá la opción de iniciar sesión hasta no quedar vinculado en otro negocio.

FA:

- FN3a. El correo no corresponde a un usuario registrado.
- FN3a. El correo introducido no tiene formato de correo.
- FN3a. El código de verificación introducido es erróneo y no corresponde con el mandado al trabajador.
- FN3b. El administrador no puede cambiarse de rol a él mismo.
- FN3b. No puedes eliminar tu propio perfil.

5.1.2.5 Añadir Gastos e Ingresos Manualmente

Cuando lo desee, un usuario con rol administrador podrá acceder a las estadísticas de Gastos e Ingresos. Estos se obtendrán de forma automática con la variación del stock, los salarios brutos o gastos e ingresos que se añadan automáticamente. Es por ello que le extiende el caso de uso Añadir Gastos e Ingresos Manualmente.

Requisitos Funcionales: 9, 15

i*: 5.1.1.3 Realizar Contabilidad

FN:

1. El Administrador accederá a la pantalla de Estadísticas.

2. Cargarán las estadísticas en forma de *pie chart* del mes actual de tipo:
 - a. Gastos
 - b. Ingresos
 - c. Gastos Extra
 - d. Ingresos Extra
 - e. Salarios
 - f. Balance (beneficios/pérdidas)
3. Podrá realizar las siguientes opciones:
 - a. Añadir gastos e ingresos extra.
 - b. Cambiar el mes.

FA:

FN2. No hay flujo de gastos e ingresos ese mes, no se mostrará nada.

FN3b. No hay gastos e ingresos ese mes, no se mostrará nada.

FN3c. El mes escogido no tiene gastos e ingresos, no se mostrará nada.

5.1.2.6 Realizar Instalaciones y Mantenimiento

Los operarios son los encargados de realizar las instalaciones y mantenimientos de productos. Estos cambios repercutirán en el stock y en las estadísticas del negocio forma automática. Lo extiende el caso de uso de Imprimir y Escanear etiquetas, ya que se incorporará conexión a impresoras térmicas bluetooth para poder imprimir el etiquetado de los productos instalados, facilitando su identificación en mantenimientos o incidencias posteriores por ejemplo.

Requisitos Funcionales: 9, 15, 16, 17, 18

i*:

- 5.1.1.2 Controlar el Stock
- 5.1.1.5 Mantenimientos Pendientes

FN:

1. El operario accederá a la pantalla de Clientes desde el menú inicial.
2. Seleccionará el cliente
3. Puede realizar las siguientes opciones:
 - a. Añadir nuevas sucursales, configurando sus propiedades, editarlas o eliminarlas.
 - b. Añadir nuevos productos dados de alta a la instalación, configurando sus parámetros como la fecha de fabricación, presión, etc. según el tipo de producto. Al añadir un producto a una instalación, se generarán las fechas de revisión automáticamente.
 - c. Editar instalaciones para reajustar los parámetros, revisar, retimbrar o imprimir su etiqueta.
 - d. Eliminar un producto dándolo de baja de la instalación.

4. Los cambios se guardarán y serán observables de forma instantánea.
5. El operario accederá a la pantalla de Mantenimiento desde el menú inicial.
6. Podrá realizar las siguientes opciones:
 - a. Seleccionar los clientes y sucursales que aparezcan, significando que tienen revisiones pendientes.
 - b. Buscar en la barra de búsqueda el número del equipo o su identificador para acceder al producto de forma fácil.
 - c. Escanear la etiqueta del producto para acceder a su ficha rápidamente.
7. Realizará la revisión del producto, reajustando sus parámetros o eliminándolo de la instalación si es necesario.
8. Los cambios se guardarán, desapareciendo los productos del panel de mantenimiento si ya han sido revisados.

FA:

- FN2. No hay clientes y sucursales dados de alta.
- FN3a. No hay productos dados de alta.
- FN3b. No hay productos en la instalación a editar.
- FN3b. No se ha podido conectar a la impresora térmica, imposibilitando imprimir el ticket.
- FN3c. No hay productos en la instalación a eliminar.
- FN6a. No hay mantenimientos pendientes en ningún cliente.
- FN6c. El código de barras no corresponde a ningún producto.

5.1.2.7 Controlar Stock

Los operarios podrán observar el stock disponible de los productos cuando lo deseen, este stock también dependerá de las instalaciones o mantenimientos realizados. Por ejemplo, si instalan dos extintores de un modelo concreto, su stock se verá alterado.

Requisitos Funcionales: 9, 15, 12

i*: 5.1.1.2 Controlar el Stock

FN:

1. El Operario accederá a la pantalla de Almacén.
2. Observará los productos dados de alta por un administrador y el stock de unidades disponibles.
3. Podrá modificar el stock mediante instalaciones y revisiones o mantenimientos de estas.

FA:

- FN2. No hay productos dados de alta.

5.2 PIM (DISEÑO DEL SISTEMA)

Una vez especificado el nivel superior de la MDA (CIM), serán necesarias aplicar ciertas transformaciones para bajar al siguiente nivel (PIM): el modelo conceptual y de diseño. En este nivel se suelen utilizar principalmente diagramas UML [14].

Este nivel representa la lógica del sistema y su interacción real sin detallar la tecnología que la implementará, al tratarse de un proyecto bastante práctico, utilizaremos tres representaciones, en este caso, Diagramas de Clases (DC en adelante), diagramas que especifiquen el comportamiento del sistema como el Diagrama de Actividad (DA) y después lo plasmaremos en una interfaz provisional con prototipado a partir de mockups.

5.2.1 DIAGRAMA DE CLASES

Es una notación UML los cuales nos permiten modelar la estructura de un sistema. Captura la estructura estática del sistema mostrando las clases y relaciones entre ellas mediante asociaciones, agregaciones o especializaciones [18]. Este diagrama lo obtendremos mediante transformaciones del MCU del CIM. El MCU nos permitía especificar las acciones del Sistema mientras que aquí se observan cómo son las clases de este, los atributos que las componen y las relaciones entre ellas.

No se incluirán CU que no añadan valor como el caso de la acción de Iniciar Sesión. Se asume que para todos los usuarios registrados, hay una precondición de iniciar sesión de forma satisfactoria.

T1: Los actores Administrador y Operario pasan a ser clases.

T2: Las clases Administrador y Operario heredarán de una clase Usuario, es decir, acceden a las características de esta. Administrador no hereda de Operario como en el MCU del CIM, debido a que aquí no se especifican las acciones que tiene el sistema, si no cómo es la estructura.

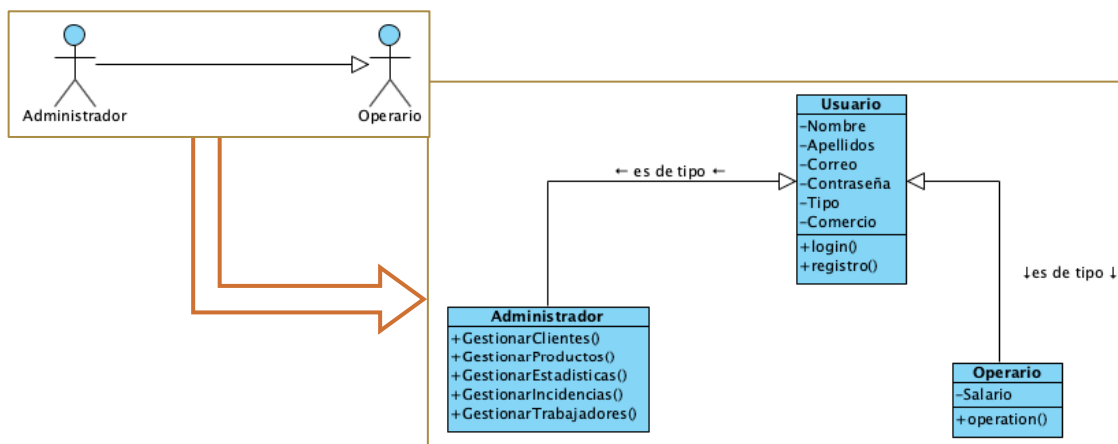


figura 14: T2 clases

T3: Algunas acciones en los CU pasan a ser Operaciones dentro de las clases, afectando a los atributos que las contienen o a otras clases que derivan de estas operaciones, por ejemplo:

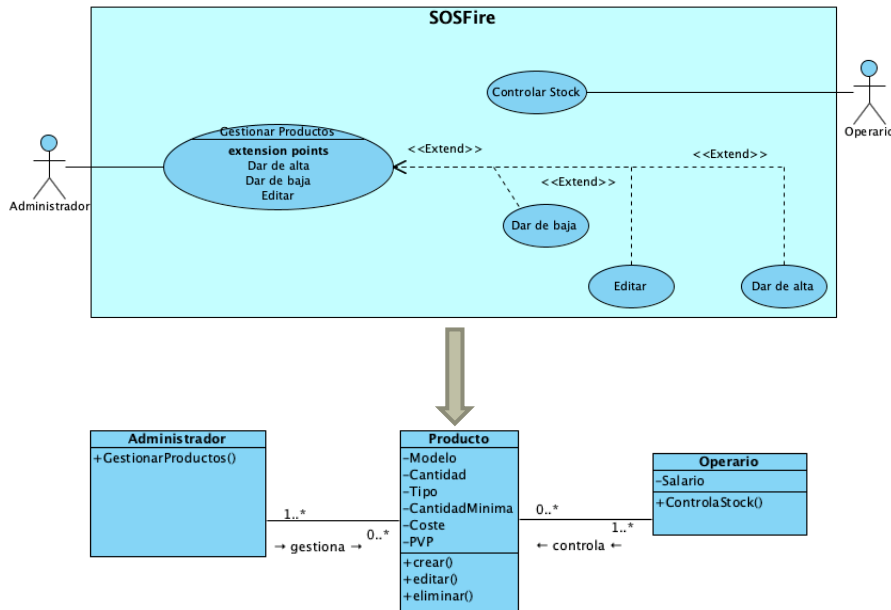


figura 15: T3 clases

T4: El Actor no Registrado no aparece en el Diagrama de Clases ya que no pertenece a la clase Usuarios ni tiene cabida en la estructura del sistema. Aparece la clase Usuarios que sería una vez el usuario no registrado ya lo haya hecho, por lo que no aparece como tal.

T5: Algunas de las Clases y sus atributos, se obtienen mediante los Flujos Normales especificados en los Casos de Uso anteriores.

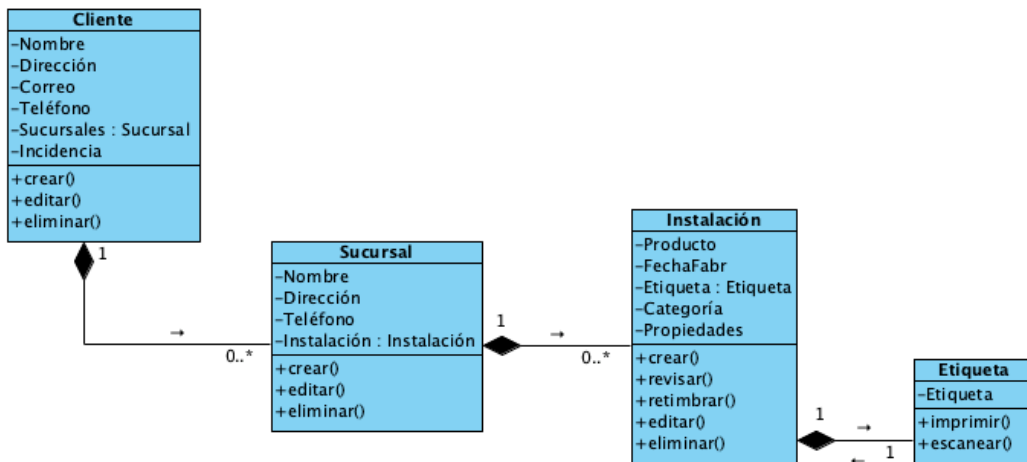


figura 16: T5 clases

T6: En el DC se señala la cardinalidad entre clases, señalando las instancias de una clase se asocian a las instancias de otra.

T7: Parte de los atributos han sido añadidos para ayudar en la implementación de código posterior y se han introducido de forma manual.

A continuación se adjunta una tabla con el resumen de las transformaciones aplicadas:

CIM	T	PIM
Aparecen los actores en el MCU	T1	Actores representados como clases
Heredan entre actores	T2	Heredan de una clase que los incluye
Acciones de los CU	T3	Operaciones de las clases
Actores sin cabida en la estructura	T4	No aparecen en el DC
Flujos Normales / Principales	T5	Clases y atributos de clases
No hay cardinalidad	T6	Se especifica cardinalidad entre clases
Falta de atributos	T7	Atributos añadidos de forma manual

Tras la aplicación de estas transformaciones, se obtiene el siguiente resultado:

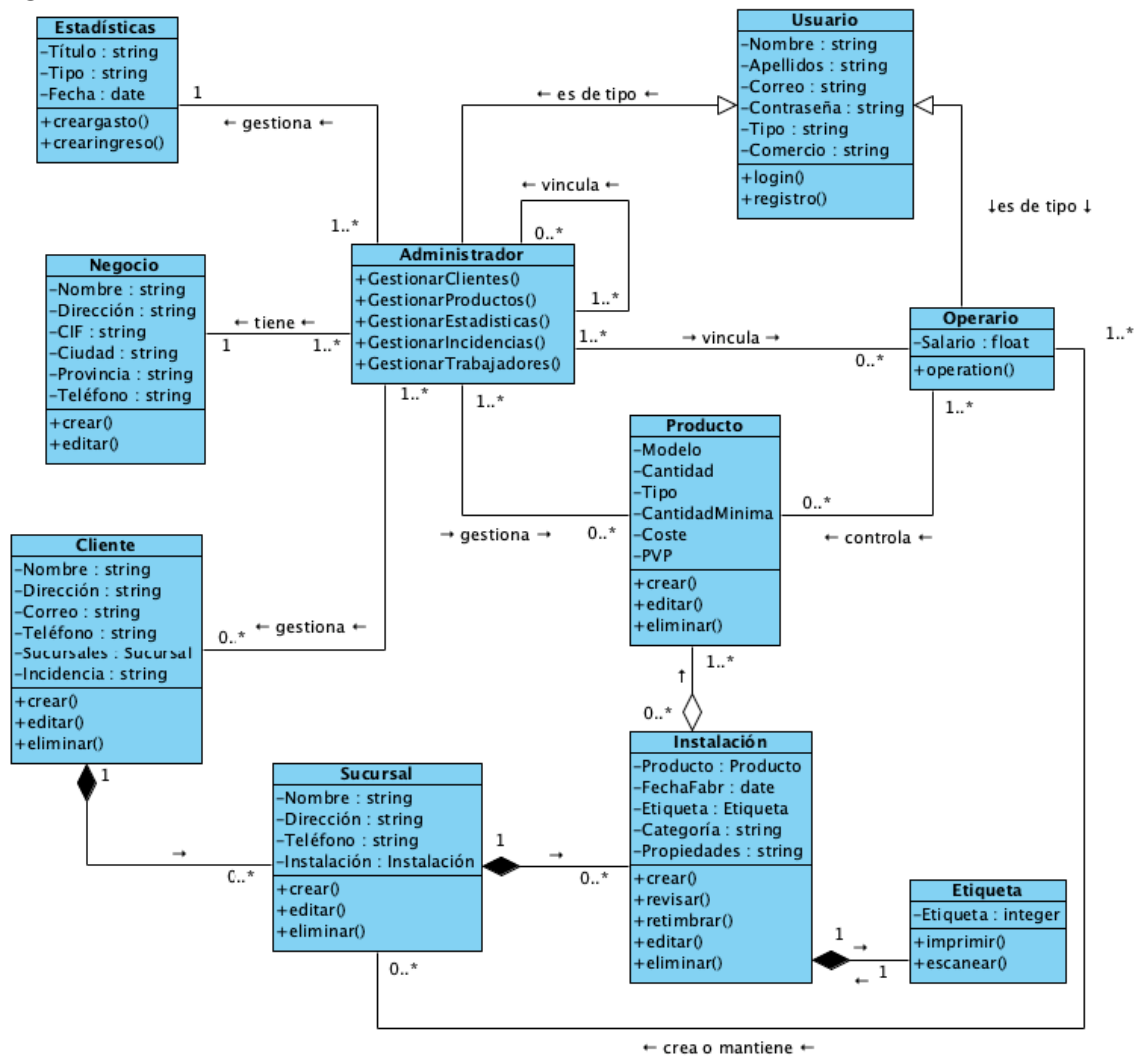


figura 17: diagrama de clases

5.2.2 DIAGRAMAS DE ACTIVIDAD

Tras especificar el esquema del sistema, se va a proceder a la creación de un modelo que describa el comportamiento de este como los Diagramas de Actividad (DA en adelante).

Los DA son utilizados para mostrar la secuencia de actividades y acciones, mostrando el flujo de trabajo desde un nodo inicial hasta uno final, indicando las rutas de decisión que pueden contener las actividades.

Sus componentes son:

- Nodo inicial: punto de partida en el flujo de acciones.
- Nodo final: punto donde terminan los flujos de acción.
- Actividades: conjunto de acciones, flujos y nodos que representan una actividad.
- Acciones: representan las tareas y acciones de una actividad.
- Flujo: ruta que muestra el orden de ejecución de las acciones.

Vamos a realizar una transformación vertical del CIM a partir del Modelo de Casos de Uso. Las reglas de transformación a aplicar son las siguientes:

T1: El DA se va a dividir en varias actividades, todas ellas juntas, representarán el sistema general, cada CU del MCU, representará una actividad del DA.

T2: Las acciones de las actividades, se determinan mediante el Flujo Normal o Principal del MCU. Por ejemplo:

5.1.2.1 Registrarse, Flujo Normal:

1. Se accederá al registro desde el menú principal.
2. Aparecerá un menú para escoger el tipo de usuario.
3. Una vez escogido, se rellenarán los datos del usuario.
 - a. Si escogió administrador, irá a otra pantalla para registrar los datos del negocio y finalizará el registro.
 - b. Si el usuario es operario, finalizará el registro.

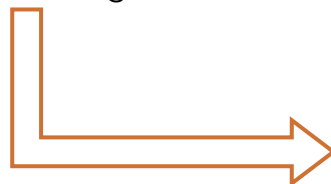
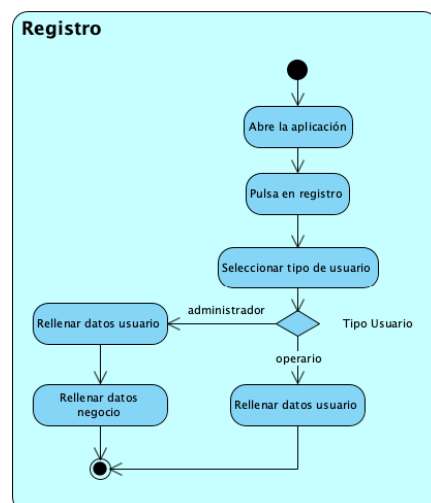


figura 18: T2 actividad



T3: Los CU marcados como Extend de otro CU, serán acciones que el usuario pueda realizar en la actividad del CU que contiene los extends.

T4: Al hablar de Pantallas en el FN del MCU, se representan en acciones en el DA.

El resumen de las transformaciones lo plasmamos en la siguiente tabla, y tras ello, los DA tras aplicar estas transformaciones.

CIM	T	PIM
CU del MCU	T1	Actividades del DA
FN del MCU	T2	Flujo de acciones del DA
CU como extend	T3	Acciones de la actividad
Pantallas en el FN	T4	Acciones del DA

Para mostrar mejor qué partes del CIM se transforman, se indicarán los CU del MCU que se han transformado.

Como precondiciones de los diagramas mostrados a continuación, salvo en la actividad de Registro, se asume que en el resto de actividades se ha iniciado sesión de forma satisfactoria y se encuentran en el Menú Inicial donde pueden acceder a las actividades que deseen realizar.

CU: 5.1.2.1 Registrarse

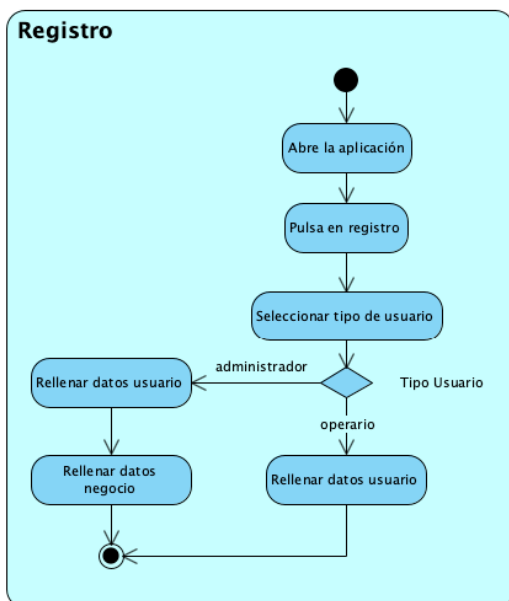


figura 19: DA registrarse

CU: 5.1.2.2 Gestionar Productos

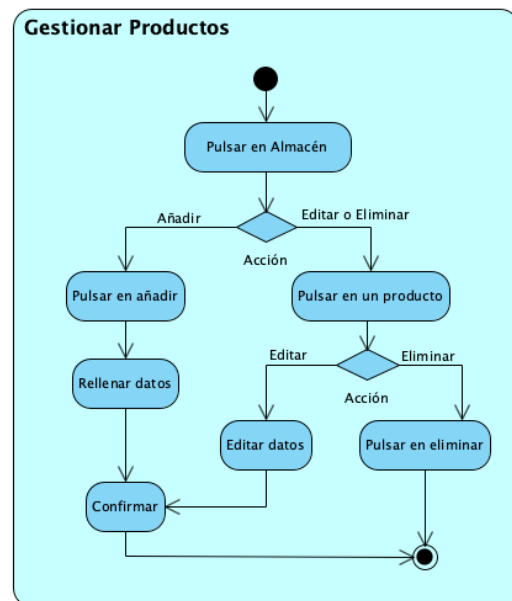


figura 20: DA gestionar productos

CU: 5.1.2.3 Gest. Fichas de Clientes

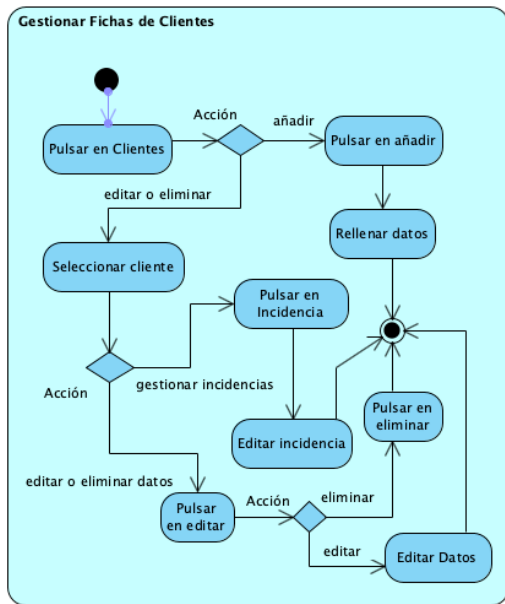


figura 21: DA gestionar fichas clientes

CU: 5.1.2.4 Gestionar Trabajadores

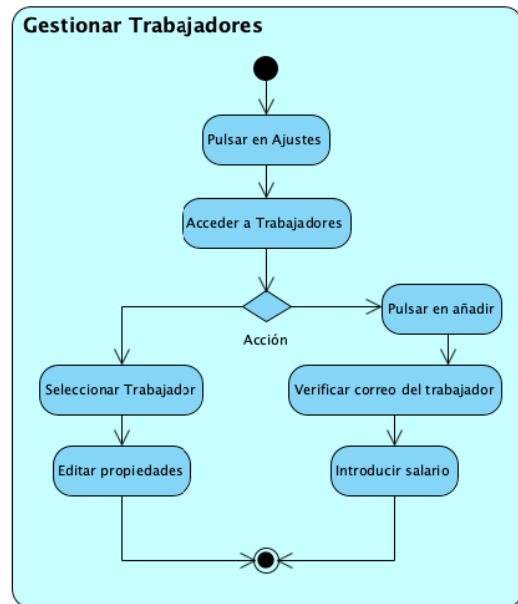


figura 22: DA gestionar trabajadores

CU: 5.1.2.5 Gastos e Ingresos

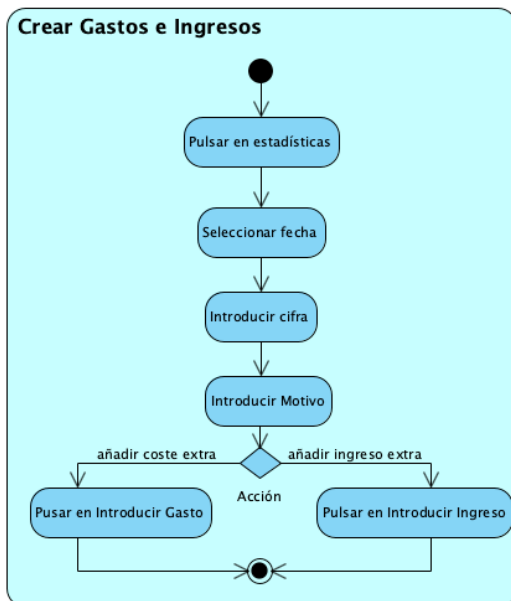


figura 23: DA gastos e ingresos

CU: 5.1.2.7 Controlar Stock

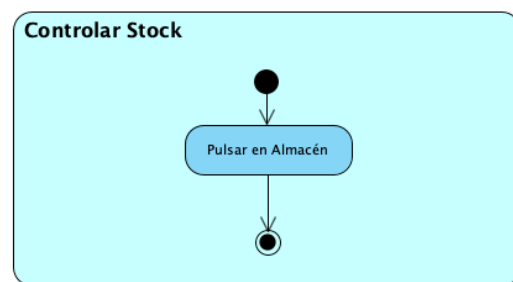


figura 24: DA controlar stock

No tiene CU

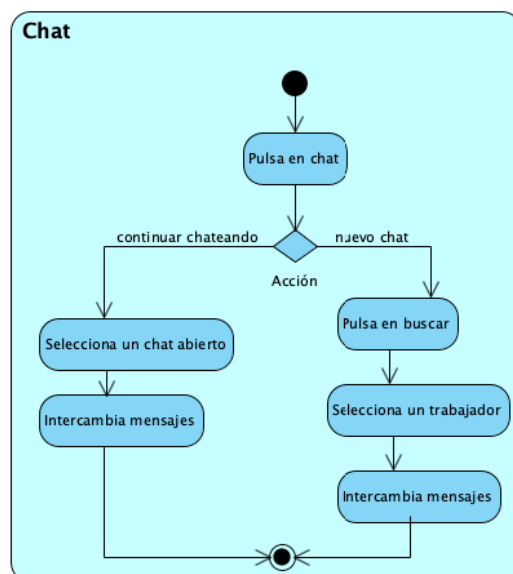


figura 25: DA chat

CU. 5.1.2.6 Instalaciones y Mantenimientos

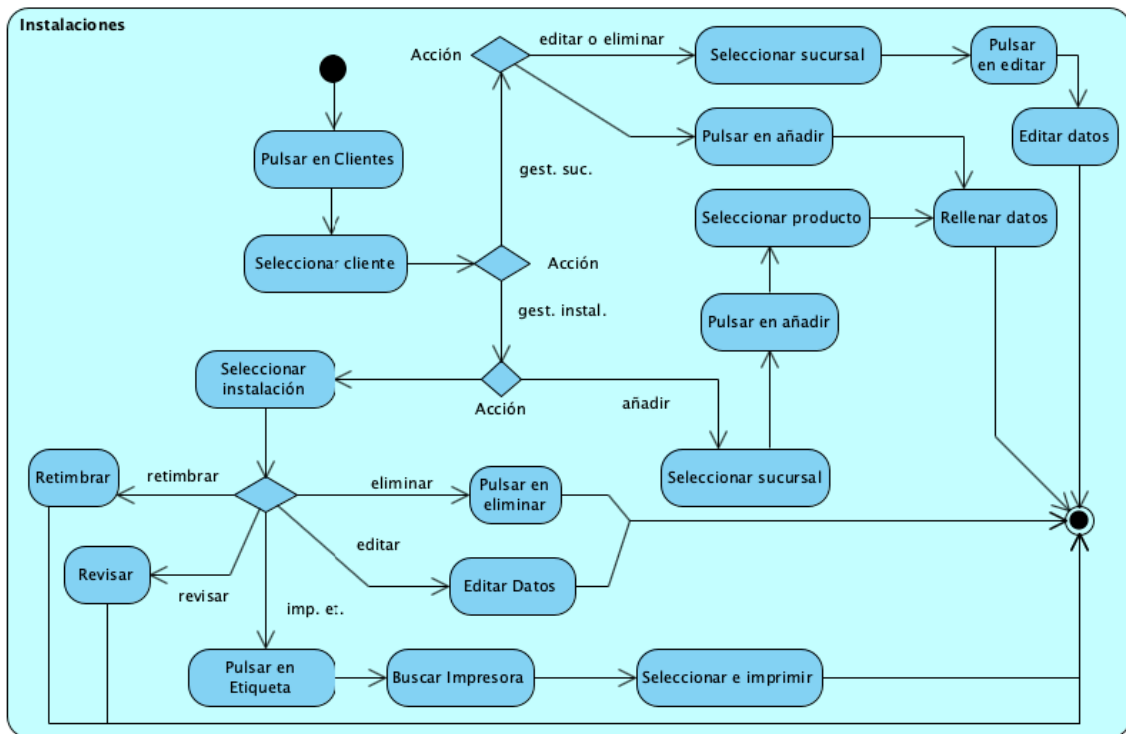


figura 26: DA instalaciones

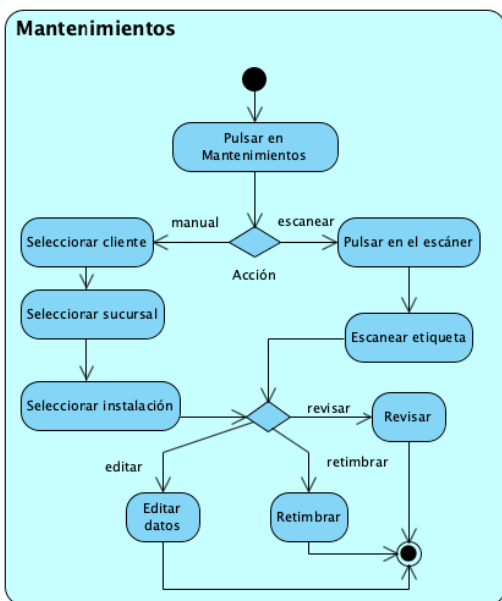


figura 27: DA mantenimientos

5.2.2 Prototipado y Mockups

Para terminar el modelado del PIM, vamos a realizar unas transformaciones horizontales entre los modelos del PIM para obtener un prototipado mediante mockups. Gracias a los Diagramas de Clases y de Actividad, podemos obtener una idea de la composición de las pantallas y la distribución de sus datos.

Los prototipos pueden elaborarse con herramientas distintas de dibujo vectorial, actualmente una de las mas utilizadas en el ámbito educativo y profesional es Balsamiq Cloud, que será la utilizada .

Cabe destacar que, como el propio nombre indica, se trata de un prototipado por lo que se verá sujeto a cambios (no drásticos, ya que tienen que verse reflejados en su versión final).

Las transformaciones a seguir son las siguientes:

T1: Cada atributo en las clases del DC, servirá para saber en mayor medida, los campos necesarios de las pantallas, su composición y las relaciones entre estas.

T2: El flujo en las actividades del DA, nos ayudan a entender la conexión entre las distintas pantallas, que junto el T1, nos ayudarán a realizar un prototipado consistente de las pantallas a implementar en el PSM.

T3: En los mockups se introducirán pantallas no especificadas en el modelo CIM ni PIM, ya que son de bajo valor para el sistema, como la pantalla de Chats, creada para añadir persistencia mediante el uso de FireBase pero que no supone un gran valor al sistema en cuanto a Requisitos Funcionales.

DC, DA	T	MOCKUPS
Atributos de clases DC	T1	Composición de pantallas
Flujo actividades DA	T2	Conexión entre pantallas
Actividades del sistema no especificadas	T3	Pantallas adicionales para cumplir un Requisito del Sistema

A continuación se adjuntan mockups de algunas pantallas y la explicación de su finalidad.

5.2.2.1 Registro



figura 28: mockups registro

Un usuario no registrado escoge su rol. Después, crea su perfil.

5.2.2.2 Registro de Negocio



Si el usuario selecciona el rol de usuario Administrador, pasará a registrar el negocio, indicando el nombre, dirección y demás parámetros necesarios. El administrador quedará vinculado al negocio de forma instantánea y ya podrá añadir a los operarios.

figura 29: mockups registro negocio

5.2.2.3 Menú Inicial

Desde aquí se podrá acceder a todas las opciones del sistema. Según el tipo de usuario que haya iniciado sesión, aparecerán distintas opciones.

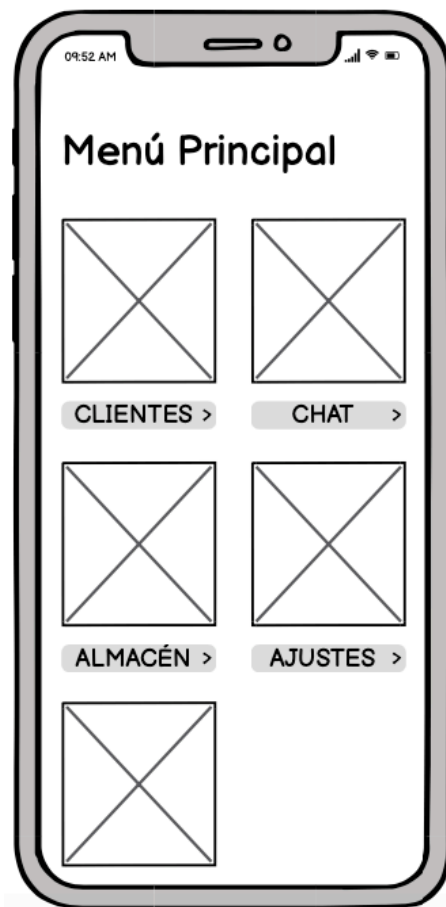


figura 30: mockups menú inicial

5.2.2.4 Almacén

En este apartado se podrá:

- Controlar el stock
- Dar de alta los productos*
- Editar productos *

*estas acciones solo por parte del administrador).



figura 31: mockups almacén

5.2.2.5 Clientes

Este apartado ofrece al usuario el listado de clientes dados de alta en el sistema. Se incluirá una barra de búsqueda y un orden alfabético para facilitar la selección del cliente. Habrán dos opciones:

- Añadir cliente: solo disponible para los administradores.
- Selección del cliente: mostrando las sucursales del cliente.

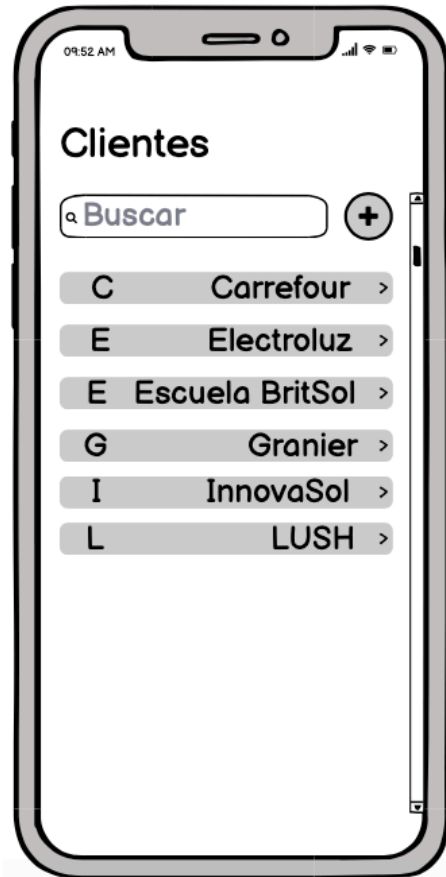


figura 32: mockups clientes



5.2.2.6 Sucursales del cliente

Esta pantalla permite seleccionar y crear las sucursales. Dentro de estas, aparecerá un listado de la instalación.

Se incluirá un botón de edición solo disponible para los administradores permitiendo editar o dar de baja el cliente seleccionado.

figura 33: mockups sucursales

5.2.2.7 Instalaciones del cliente

Al pulsar sobre una sucursal, se abrirá la ficha de la instalación realizada. Desde la ficha se permitirá editar la sucursal, notificar una incidencia o añadir productos a la instalación.

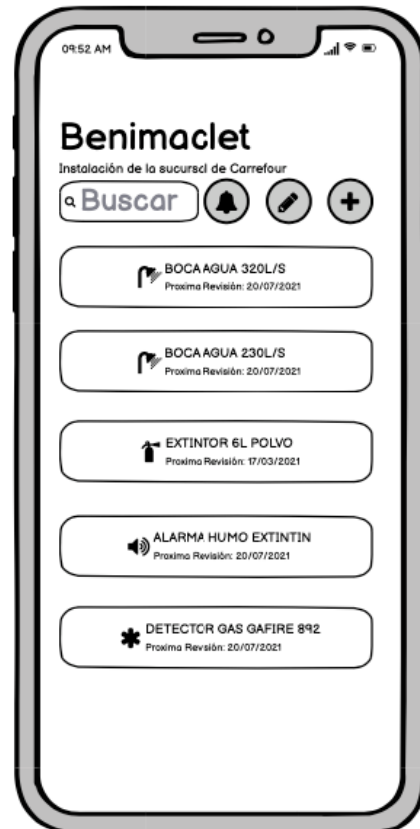


figura 34: mockups instalaciones

5.2.2.8 Incidencias del cliente



figura 35: mockups incidencias

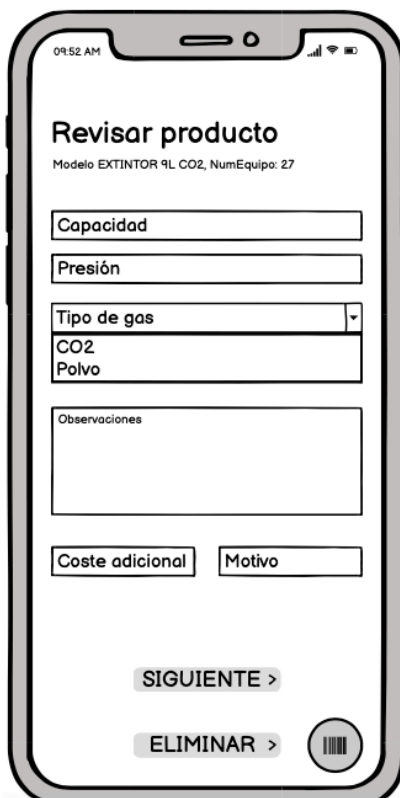
Cuando un cliente tenga una incidencia en una instalación, se señalará en el listado de clientes y sucursales distinguiéndola de los demás y cuando se acceda a la instalación, se notificará la incidencia creada por el administrador desde el botón indicado

5.2.2.9 Añadir productos en la Instalación



figura 36: mockups instalaciones (añadir)

El trabajador podrá seleccionar el producto (5.2.5.1) y anotar sus propiedades. Se añadirá una fecha de fabricación y la fecha de instalación (esta automáticamente), al guardar, se generarán las fechas de revisión, apareciendo la notificación de falta de revisión cuando llegue el momento en la pantalla de mantenimiento.



Los productos instalados podrán ser editados y dados de baja por cualquier trabajador. En la ficha de un producto instalado, se permitirá la impresión de una etiqueta mediante la conexión bluetooth con una impresora térmica.

figura 37: mockups instalaciones revisar

5.2.2.10 Mantenimientos



figura 38: mockups mantenimientos

Aparecerán las instalaciones con mantenimientos (revisiones o retimbres) pendientes ese mes o anteriores. Se podrá acceder a la ficha del producto buscándolo por su número de etiqueta, escaneándola o manualmente. Al pulsar sobre un producto con revisión pendiente, aparecerá la pantalla de revisar producto del punto anterior.

5.2.2.11 Vincular trabajadores

Tras registrarse un operario, tendrá que vincularse a un negocio. Para ello, el administrador introducirá su correo electrónico al cual le llegará un código de confirmación.

Ese código se le notifica al administrador, se introduce un salario bruto (puede ser 0) y si el código es correcto, se vinculará al negocio con el rol de Operario.

Se pueden editar los trabajadores (rol, salario) y dar de baja en cualquier momento.

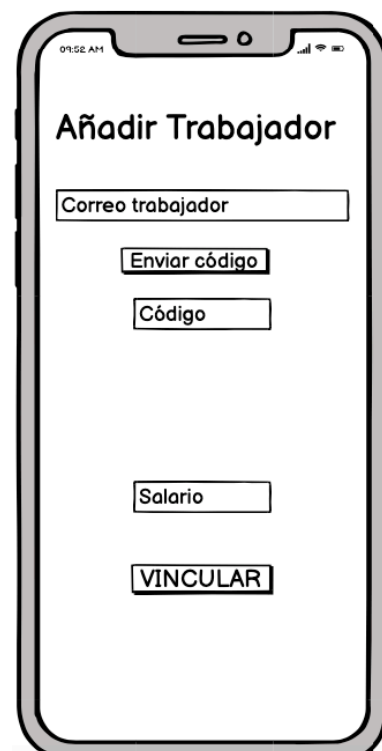


figura 39: mockups trabajadores

5.2.2.12 Estadísticas

El administrador podrá observar las estadísticas de los gastos e ingresos del negocio, pudiendo seleccionar la fecha. Estos datos se calculan automáticamente mediante el flujo de productos (ya que se apunta el coste y el PVP), los salarios o pudiendo agregar gastos e ingresos de forma manual.



figura 40: mockups estadísticas

5.2.2.13 Chat

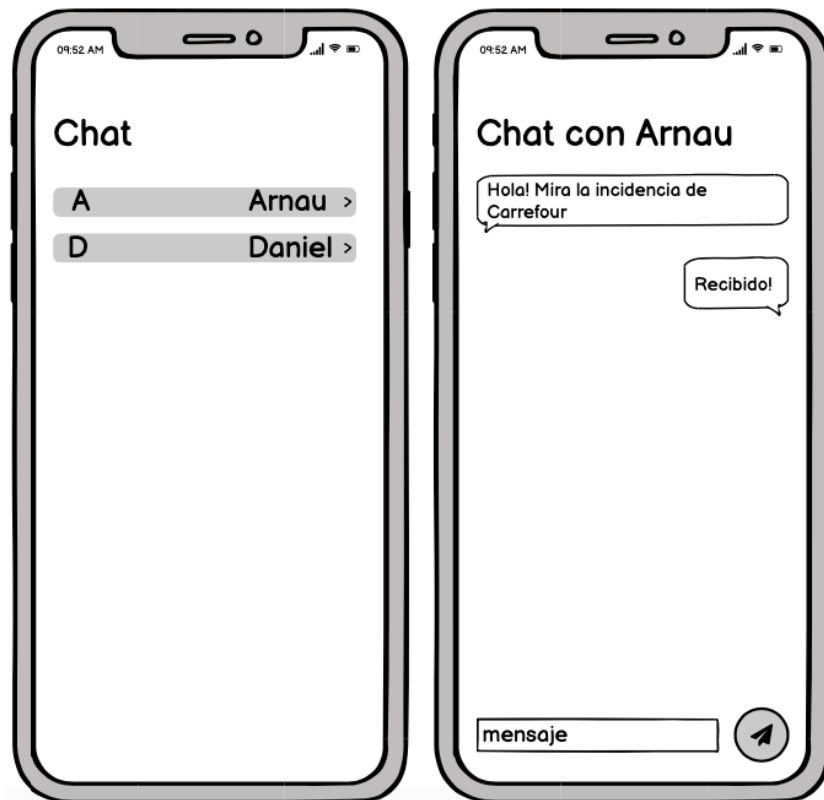


figura 41: mockups chat

Se habilitará el intercambio de mensajes entre usuarios vinculados a un mismo negocio.

5.3 PSM (CÓDIGO A IMPLEMENTAR)

Tras la especificación de los niveles superiores de la MDA, se aplicarán las transformaciones necesarias en el PIM para obtener las componentes del PSM.

En el PSM se encuentra la parte más difícil y laboriosa del proyecto: la implementación.

Para no aportar mucha información necesaria y de mas, se va a evitar en la mayor medida la copia y pega de mucho código. Se van a elaborar tres módulos a partir de las transformaciones:

- Diseño de la Base de Datos
- Código PHP y de funciones

5.3.1 Tecnología

Siguiendo lo anteriormente descrito, se utilizará el kit de desarrollo software de *Flutter* mediante el lenguaje *dart* y la aplicación Android Studio, proporcionándonos compatibilidad en dispositivos iOS y Android.

La conexión con la base de datos se realizará mediante PHP, estos ficheros se almacenarán en un *server* y se accederán a ellos mediante las funciones que *Flutter* proporciona con los permisos necesarios activados.

Los recursos gráficos utilizados son los proporcionados por librerías propias de *Flutter* (los iconos), creaciones propias y ficheros SVG gratuitos con referencias a los autores (adjuntadas en los anexos).

Se pasará a especificar la implementación de las pantallas más relevantes para el sistema, indicando qué bases de datos utiliza, qué funciones realiza y cómo lo hace. Al finalizar la implementación, se elaborará una tabla que indique la trazabilidad entre *i**, Casos de Uso, Requisitos y la implementación (con los PHP y partes de la implementación que los utilizan).

5.3.2 Bases de Datos

Las bases de datos son colecciones de datos estructurados. Para este proyecto, se han escogido las bases de datos relacionales y se elaborarán mediante el sistema de administración MySQL. Este sistema se almacena en un servidor *Hostinger* y se manipulará mediante consultas al servidor con ficheros PHP.

Su estructura se basa en múltiples tablas que almacenan y organizan la información. Estas tablas, se han elaborado siguiendo unas transformaciones verticales desde el nivel PIM. Se ha utilizado el modelo de Diagrama de Clases como base a la que aplicar las transformaciones, ya que se pueden relacionar fácilmente con las bases de datos. Las relaciones de transformación aplicadas son las siguientes:

T1: Las Clases pasarán a ser Tablas

T2: Los string en las clases pasan a ser VARCHAR(n) en las columnas.

T3: Los float en las clases pasan a ser DOUBLE en las columnas.

T4: Se introducen datos que ayuden a relacionar las tablas como las Primary Keys o información adicional requerida para el manejo de datos como en los gastosingresos (que hace referencia a la clase Estadísticas).

T5: La clase de instalaciones se separan en tipos de producto:

- Los que tienen retimbre como los extintores y bocas de incendio, comparten muchos elementos comunes.
- Los que no tienen retimbre, denominados centrales, son los detectores, sistemas de extinción y grupos de presión, comparten más elementos entre sí y carecen de retimbres.

T6: Se añade una tabla Trabajadores para facilitar la recuperación de los trabajadores vinculados a un negocio, así no se tiene que recorrer toda la lista de usuarios. Ahí se añadirá el Salario de los Operarios.

T7: Las operaciones de las clases pasarán a ser funciones que manejen las tuplas en la base de datos.

PIM	T	PSM
Clases en DC	T1	Tablas en BBDD
Atributos con string	T2	Columnas de tipo VARCHAR(n)
Atributos con float	T3	Columnas de tipo DOUBLE
Atributos limitados	T4	Columnas necesarias para las relaciones
Clase Instalaciones	T5	Tablas separadas de ExtintoresBocas y Centrales
Clases Administradores y Operarios	T6	Tabla Trabajadores
Operaciones de la clase	T7	Operaciones que editen las tablas

Todas las tablas (salvo la de 5.3.2.1 Usuarios) se generan cuando un usuario de tipo Administrador se registra. Los títulos de esta, corresponden a un identificador único almacenado también en la tabla usuarios, quedando de la siguiente forma ID+nombretabla.

Por ejemplo, la tabla clientes, tendría un nombre parecido a 7j129sakclientes. De esta forma, al iniciar sesión y comprobar el usuario en la tabla de usuarios, se obtiene el identificador del negocio y se sabe qué tablas corresponden al negocio. Estas son las tablas resultantes al aplicar las transformaciones:

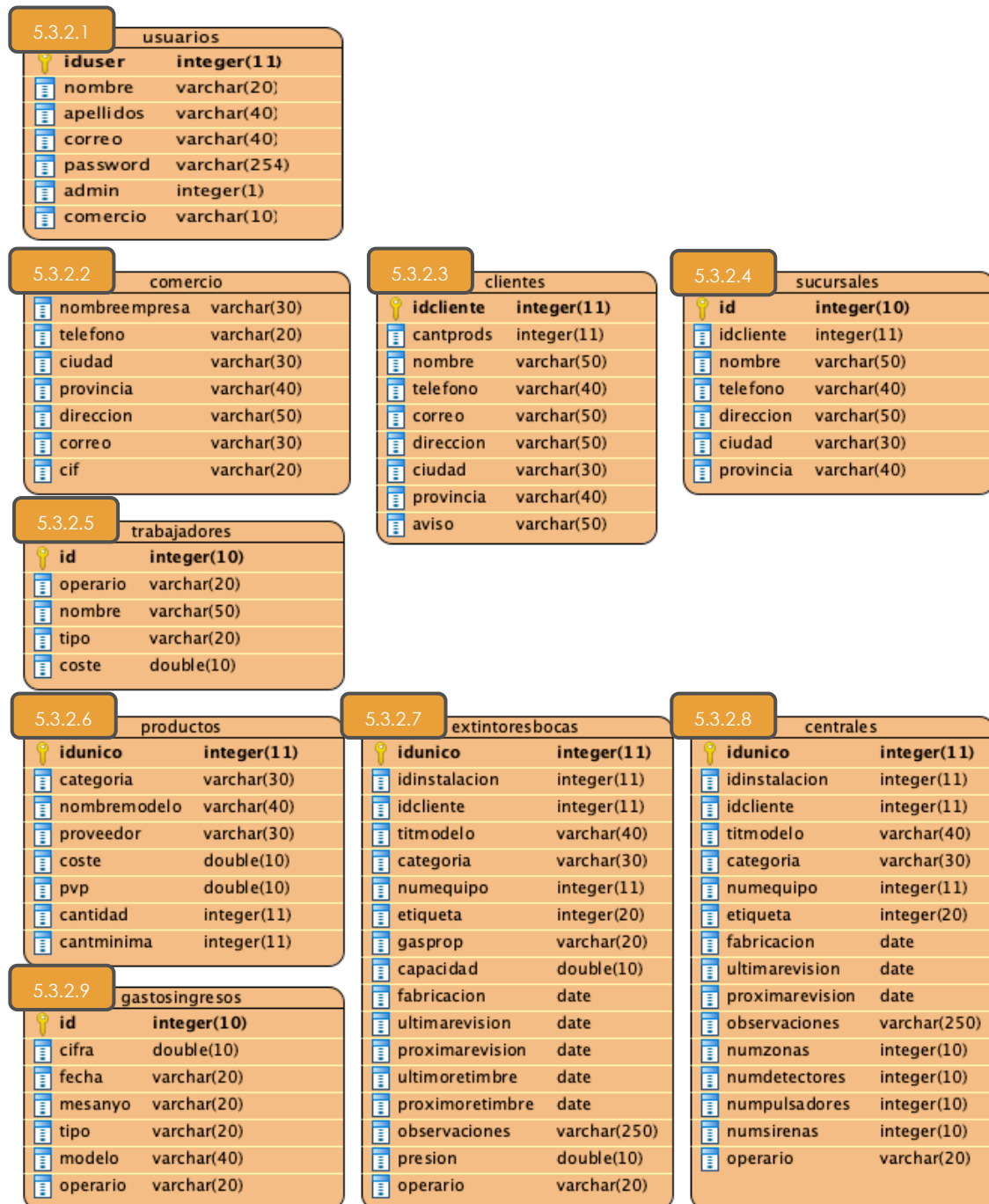


figura 42: tablas bases de datos

5.3.3 Código de Pantallas y Funciones

El código de las pantallas y funciones vendrán dados mediante la transformación del PIM en los modelos de DC, DA y basándonos en el prototipado de Mockups, que nos otorga una visión de cómo será la UI final. Este código se va a establecer mediante *Flutter*, un SDK que utiliza el lenguaje *dart* y el manejo de Bases de Datos MySQL o FireBase mediante PHP o funciones específicas de Flutter.

T1: Cada Actividad del DA será un conjunto de pantallas que comparta código entre ellas, pasando información de una a otra para facilitar el manejo de datos. Por ejemplo, al seleccionar un Cliente, la pantalla de sucursales tendrá los datos del cliente seleccionado en la pantalla anterior, para poder cargar justo esas sucursales y mostrar los datos.

T2: Las Clases del DC nos ayudarán en la composición de elementos en la pantalla, mostrando por ejemplo los datos a mostrar o los datos que se requieren rellenar.

Al especificar en la clase Negocio sus atributos, podremos saber que la pantalla que edite los atributos del negocio, mostrará los datos de esta clase.

T3: Se incluirán todas las implementaciones necesarias para hacer cumplir los requisitos del sistema, tanto funcionales como no funcionales, aunque estos no se hayan visto en modelados anteriores debido a su poco valor en el análisis o diseño del sistema (inicio de sesión, chat grupal, que el menú inicial varíe según el tipo de usuario logeado, etc.). Algunas de estas pantallas se han podido ver en el prototipado de Mockups.

T4: Los Mockups serán el punto de partida para diseñar la interfaz.

PIM	T	PSM
Actividad en DA	T1	Conjunto de pantallas y funciones
Clases en DC	T2	Composición de los elementos que aparecerán en la pantalla
Requisitos no abarcados por otros modelos	T3	Cumplir con todos los requisitos
Mockups	T4	Modelo base de interfaz

A continuación, se van a mostrar algunas de las transformaciones realizadas. Se especificará qué parte del DA representa, qué clase del DC utiliza y las tablas afectadas, junto un trozo de código que lo represente. No se adjuntará entero debido a la gran cantidad de código y se evitará repetir parte de este. Por ejemplo, las consultas para obtener datos de las diferentes pantallas son muy parecidas entre sí.

5.3.3.1 Registro y creación de tablas

Finalidad: El usuario no registrado seleccionará el tipo de usuario e introducirá sus datos. Si elige administrador, también introducirá los datos de su negocio.

Diagrama de Actividad: Registro

Clases DC: Usuario

Tablas afectadas: 5.3.2.1 Usuarios, 5.3.2.2 Comercio

Flutter Packages: UUID

Funcionamiento: Después de introducir los datos, estos se recogen y primero se comprueba que el usuario no esté registrado anteriormente mediante una consulta que recorre la tabla de usuarios, si es único, *hashea* la contraseña y tras ello, sus datos pasan a la tabla 5.3.2.1:

- Si es usuario de tipo Operario, en la casilla Admin va un 0 y en comercio un 0.
- Si es usuario de tipo Administrador, en la casilla Admin va un 1 y en la de Comercio se crea un ID único mediante el paquete UID.

Tras el registro del usuario, si es Administrador, se crearán todas las tablas del negocio con un prefijo el cual corresponde a la UID almacenada. En la tabla de comercio (5.3.2.2) se añadirán los datos del comercio creado.

PHP:

- Comprueba que el usuario no exista antes de registrarlo (si return true, llama a una consulta para añadirlo a la tabla):

```
function usernameAvailable() {
    global $connect, $correo;
    $statement = mysqli_prepare($connect, "SELECT * FROM usuarios WHERE correo = ?");
    mysqli_stmt_bind_param($statement, "s", $correo);
    mysqli_stmt_execute($statement);
    mysqli_stmt_store_result($statement);
    $count = mysqli_stmt_num_rows($statement);
    mysqli_stmt_close($statement);
    if ($count < 1){
        return true;
    }else {
        return false;
    }
}
```

- Crea las tablas del comercio:

```
$tablacomercio = mysqli_real_escape_string($connect, $_POST['tablacomercio']);
$tablaproductos = mysqli_real_escape_string($connect, $_POST['tablaproductos']);
$tablaextintoresbocas = mysqli_real_escape_string($connect, $_POST['tablaextintoresbocas']);
$tablacentrales = mysqli_real_escape_string($connect, $_POST['tablacentrales']);
$tablaclientesproveedores = mysqli_real_escape_string($connect, $_POST['tablaclientesproveedores']);
$tablasucursales = mysqli_real_escape_string($connect, $_POST['tablasucursales']);
$tablaoperarios = mysqli_real_escape_string($connect, $_POST['tablaoperarios']);
$tablagastosingresos = mysqli_real_escape_string($connect, $_POST['tablagastosingresos']);

$statement = mysqli_prepare($connect, "CREATE TABLE " . $tablacomercio . " (nombreempresa VARCHAR(30) NOT NULL, telefono VARCHAR(40) NOT NULL, ciudad VARCHAR(30) NOT NULL, provincia VARCHAR(40) NOT NULL, direccion VARCHAR(50) NOT NULL, correo VARCHAR(30) NOT NULL, cif VARCHAR(20) NOT NULL)");
mysqli_stmt_execute($statement);
mysqli_stmt_close($statement);

$statement = mysqli_prepare($connect, "CREATE TABLE " . $tablaproductos . " (idunico INT AUTO_INCREMENT PRIMARY KEY, categoria VARCHAR(30) NOT NULL, nombremodelo VARCHAR(40), proveedor VARCHAR(30), coste DOUBLE NOT NULL, pvp DOUBLE NOT NULL, cantidad INT NOT NULL, cantminima INT NOT NULL)");
mysqli_stmt_execute($statement);
```


Dart:

- Creación de nombres para las tablas:

```
var uuid = Uuid();
String uuidstring = uuid.v1();
String acertado = uuidstring.substring(0,8);
idtrabajofirebase = acertado;
String tablacomercio = acertado + "comercio";
String tablaproductos = acertado + "productos";
String tablaextintoresbocas = acertado + "extintoresbocas";
String tablacentrales = acertado + "centrales";
String tablaclientesproveedores = acertado + "clientesproveedores";
String tablasucursales = acertado + "sucursales";
String tablaoperarios = acertado + "operarios";
String tablagastosingresos = acertado + "gastosingresos";
```

- Llamadas a los PHP:

```
Future creausuario() async {
  final responseusuario = await http.post(
    "urlphpregistroadmin", body: {
      "nombre": widget.nombreadmin,
      "apellidos": widget.apellidosadmin,
      "correo" : widget.correoadmin,
      "password" : widget.passadmin,
      "admin" : "1",
      "comercio" : acertado,
    });
  if(responseusuario.statusCode == 200){
    Map<String, dynamic> data = jsonDecode(responseusuario.body);
    if(data["success"].toString() == "true"){
      final responsetablas = await http.post(
        "urlphpcreartablas", body: {
          "tablacomercio" : tablacomercio,
          "tablaproductos": tablaproductos,
          "tablaextintoresbocas" : tablaextintoresbocas,
          "tablacentrales" : tablacentrales,
          "tablaclientesproveedores" : tablaclientesproveedores,
          "tablasucursales" : tablasucursales,
          "tablaoperarios" : tablaoperarios,
          "tablagastosingresos" : tablagastosingresos,
        });
      if(responsetablas.statusCode == 200){
        Map<String, dynamic> data2 = jsonDecode(responsetablas.body);
        if(data2["success"].toString() == "true"){
          final responseempresa = await http.post(
            "http://www.editajob.com/EditaComercio.php", body: {
              "tablacomercio" : tablacomercio,
              "nombreempresa": nombreempresa.text,
              "telefonoempresa": telefonoempresa.text,
              "ciudadempresa": ciudadempresa.text,
              "provinciaempresa" :
                _provinciaEscogida.nombre,
              "direccionempresa" : direccionempresa.text,
              "correoempresa" : correoempresa.text,
              "cifempresa" : cifempresa.text,
            });
        }
      }
    }
  }
}
```

Los datos para las llamadas a los PHP, los obtiene de casillas de texto o de pantallas anteriores. En este caso, la anterior le manda los datos del usuario a registrar, por eso aparece el nombre en el título de la pantalla.

figura 43: captura registro

20:31

Bienvenidx, Arnau
Para terminar, rellena los datos del comercio.

Nombre de la empresa

Dirección de la empresa

Ciudad de la empresa

Provincia Álava

Teléfono de la empresa

Correo de la empresa

CIF de la empresa

FINALIZAR

1 2 3

5.3.3.2 Almacén

Finalidad: El Almacén permite crear los productos que van a utilizarse en las instalaciones. El Administrador selecciona qué tipo de producto se va a instalar, rellena unas características básicas y lo añade. Se permite introducir un aviso de falta de stock para cuando la cantidad de productos sea inferior, lo notifique cuando se accede a la pantalla.

Diagrama de Actividad: Gestión de Productos

Clases DC: Producto

Tablas afectadas: 5.3.2.6, 5.3.2.9

Funcionamiento: Se consultan los datos de los productos creados en la tabla productos (5.3.2.6) y se genera un ListView donde cada ListTile es un elemento encontrado en la tabla, mostrando los datos seleccionados (nombremodelo y cantidad). Cuando termina de cargar, se comprueba si la casilla cantidad es igual o inferior a la de cantminima, si es así, se muestra un AlertDialog informando de qué producto tiene poco stock (con nombremodelo).

Al crear o editar un producto, los gastos o ingresos que deriven de la acción, modifican la tabla de estadísticas (GastosIngresos, 5.3.2.9):

- Si se crea un producto de coste 20€ y tiene 5 de cantidad, habrá un gasto de 100€.
- Si se edita un producto y se añade mas cantidad, lo mismo.
- Si se edita un producto y se quita cantidad, habrá un ingreso ya que se entiende que se ha vendido.

PHP:

- Obtener productos:

```
$tablaproductos = mysqli_real_escape_string($connect, $_POST['tablaproductos']);
$respuesta = array();
$respuesta[$tablaproductos] = array();

$stmt = "SELECT * FROM " . $tablaproductos;
mysqli_stmt_execute($stmt);

$result = mysqli_query($connect, $stmt);

while($row = mysqli_fetch_array($result)){
    $tmp = array();
    $tmp["idunico"] = $row["idunico"];
    $tmp["categoria"] = $row["categoria"];
    $tmp["nombremodelo"] = $row["nombremodelo"];
    $tmp["proveedor"] = $row["proveedor"];
    $tmp["coste"] = $row["coste"];
    $tmp["pvp"] = $row["pvp"];
    $tmp["cantidad"] = $row["cantidad"];
}
```

- Añadir Productos:

```
$tablaproductos = mysqli_real_escape_string($connect, $_POST['tablaproductos']);
$categoria = mysqli_real_escape_string($connect, $_POST['categoria']);
$nombremodelo = mysqli_real_escape_string($connect, $_POST['nombremodelo']);
$proveedor = mysqli_real_escape_string($connect, $_POST['proveedor']);
$coste = mysqli_real_escape_string($connect, $_POST['coste']);
$pvp = mysqli_real_escape_string($connect, $_POST['pvp']);
$cantidad = mysqli_real_escape_string($connect, $_POST['cantidad']);
$cantminima = mysqli_real_escape_string($connect, $_POST['cantminima']);

$stmt = mysqli_prepare($connect, "INSERT INTO " . $tablaproductos . " (categoria, nombremodelo,
proveedor, coste, pvp, cantidad, cantminima) VALUES(?, ?, ?, ?, ?, ?, ?)");
mysqli_stmt_bind_param($stmt, "ssssss", $categoria, $nombremodelo, $proveedor, $coste, $pvp,
$cantidad, $cantminima);
```

Dart:

- Listar productos:

```
obtenerProductos() async {
  SharedPreferences prefs = await SharedPreferences.getInstance();
  String tablaproductos = prefs.getString('tablaproductos');
  idadmin = prefs.getString('tipousuario');

  final responseproductos =
    await http.post("urlphpobtenerproductos", body: {
      "tablaproductos": tablaproductos,
    });

  if (responseproductos.statusCode == 200) {
    Producto producto = Producto(
      jsonData[tablaproductos][i]["idunico"],
      jsonData[tablaproductos][i]["categoria"],
      jsonData[tablaproductos][i]["nombremodelo"],
      jsonData[tablaproductos][i]["proveedor"],
      jsonData[tablaproductos][i]["coste"],
      jsonData[tablaproductos][i]["pvp"],
      jsonData[tablaproductos][i]["cantidad"],
      jsonData[tablaproductos][i]["cantminima"]);

    productos.add(producto);
  }
}
```

```
ListView.builder(
  shrinkWrap: true,
  itemCount: productos.length,
  itemBuilder: (context, index) {
    Producto producto = productos[index];
    return ListTile(
      child: SvgPicture.asset(
        "assets/productos/" + producto.categoria + ".svg",
      ),
      title: Text(producto.nombremodelo, style: TextStyle(
        color: Colors.black54, fontWeight: FontWeight.bold),),
      subtitle: Text("Cantidad: " + producto.cantidad,),
      onTap: () {
        EditaProducto();
      },
    ),
  ),
);
```

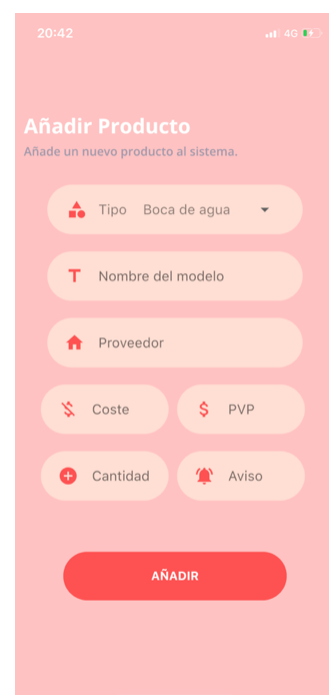


figura 44: captura almacén
figura 45: captura añadir producto

5.3.3.3 Clientes

Finalidad: El Administrador podrá crear, editar y dar de baja a los clientes. Para ello pulsará en el botón y rellenará sus datos. Si pulsa sobre él, se abrirá el listado de sucursales (5.3.9), desde ahí se permite la edición del cliente y añadir incidencias.

Diagrama de Actividad: Gestión Fichas de Clientes

Clases DC: Cliente

Tablas afectadas: 5.3.2.3, 5.3.2.4, 5.3.2.7, 5.3.2.8

Funcionamiento: Al acceder a la pantalla se obtienen los datos de la tabla de Clientes (5.3.2.3) y se muestran mediante un ListView (como en el punto anterior, con una clase cliente). Si la casilla Aviso contiene datos, el nombre del cliente pasa a rojo y se muestra el aviso al acceder a él. Este se puede crear y editar desde la ficha del cliente.

Para crear un cliente, se pulsa el botón y añaden los datos los cuales pasan a rellenar una tupla en la tabla Clientes, al hacerlo, el cliente aparecerá en el listado de forma automática. Al acceder a él se permite la edición de su ficha (modificar los datos) y creación o edición de incidencias.

Al eliminar/dar de baja un cliente, se eliminarán todas las sucursales e instalaciones vinculadas a él (gracias a su identificador en la tabla) en todas las tablas necesarias.

PHP:

- Editar clientes:

```
$tabla = mysqli_real_escape_string($connect, $_POST['tablaclientes']);
$idcliente = mysqli_real_escape_string($connect, $_POST['idcliente']);
$nombre = mysqli_real_escape_string($connect, $_POST['nombre']);
$telefono = mysqli_real_escape_string($connect, $_POST['telefono']);
$correo = mysqli_real_escape_string($connect, $_POST['correo']);
$direccion = mysqli_real_escape_string($connect, $_POST['direccion']);
$ciudad = mysqli_real_escape_string($connect, $_POST['ciudad']);
$provincia = mysqli_real_escape_string($connect, $_POST['provincia']);
$aviso = mysqli_real_escape_string($connect, $_POST['aviso']);

$stmtatement = mysqli_prepare($connect, "UPDATE " . $tabla . " SET nombre = '$nombre', telefono = '$telefono', correo = '$correo', direccion = '$direccion', ciudad = '$ciudad', provincia = '$provincia', aviso = '$aviso' WHERE idcliente = '$idcliente'");
mysqli_stmt_execute($statement);
mysqli_stmt_close($statement);
```

- Eliminar clientes:

```
$tabla = mysqli_real_escape_string($connect, $_POST['tabla']);
$idcliente = mysqli_real_escape_string($connect, $_POST['idcliente']);

$stmtatement = mysqli_prepare($connect, "DELETE FROM " . $tabla . " WHERE idcliente = '$idcliente'");
mysqli_stmt_execute($statement);
mysqli_stmt_close($statement);
```

Dart:

- Eliminar clientes:

```
SharedPreferences prefs = await SharedPreferences.getInstance();
String tablaclientes = prefs.getString('tablaclientesproveedores');
String tablasucursales = prefs.getString('tablasucursales');
String tablaextintoresbocas = prefs.getString('tablaextintoresbocas');
String tablacentrales = prefs.getString('tablacentrales');

final responselimnarclientes =
await http.post("urlphpclienteseliminar", body: {
  "tabla": tablaclientes,
  "idcliente": widget.idcliente,
});

final responseliminarsucursales =
await http.post("urlphpclienteseliminar", body: {
  "tabla": tablasucursales,
  "idcliente": widget.idcliente,
});

final responseliminarextintoresbocas =
await http.post("urlphpclienteseliminar", body: {
  "tabla": tablaextintoresbocas,
  "idcliente": widget.idcliente,
});

final responseliminarcentrales =
await http.post("urlphpclienteseliminar", body: {
  "tabla": tablacentrales,
  "idcliente": widget.idcliente,
});
```

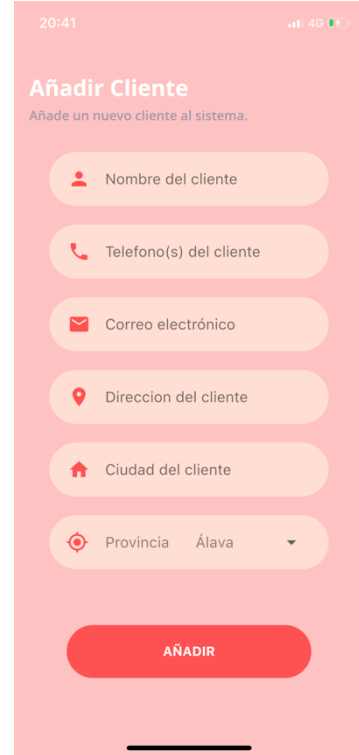
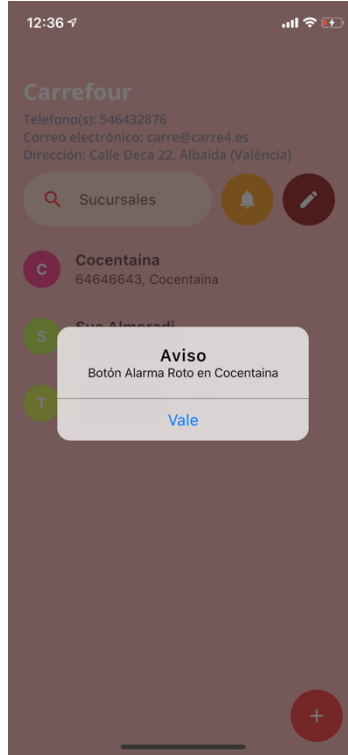
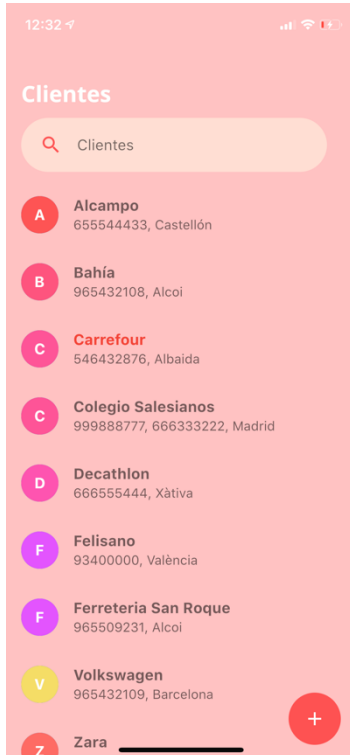


figura 46: captura clientes

figura 47: captura aviso incidencia

figura 48: captura añadir cliente

5.3.3.4 Trabajadores

Finalidad: El Administrador puede vincular, editar o dar de baja un trabajador. Tras vincularlo, este accederá a los datos del negocio (clientes, productos, etc) y es posible editar su rol y salario.

Diagrama de Actividad: Gestionar Trabajadores

Clases DC: Administrador, Operario, Usuario

Tablas afectadas: 5.3.2.1, 5.3.2.5

Flutter Packages: UUID, Mailer

Funcionamiento: Los datos de trabajadores se obtienen de la tabla trabajadores (5.3.2.5) con una consulta al servidor y mediante un ListView se genera una lista de los trabajadores que en ella aparecen. Al pulsar en ellos, se permite la edición de sus datos, y como antes, hay un PHP en el servidor que tras acceder a él enviándole los datos actualiza la tabla realizando los cambios.

Al querer vincular un trabajador, se enviará mediante el paquete Mailer y el servidor SMTP especificado en el código, un código generado con el paquete UUID. El trabajador lo comunica al Administrador y este lo introduce, si coincide (compara el valor introducido con el generado), quedará dado de alta y registrado en la tabla 5.3.2.5 y con la casilla Comercio de la 5.3.2.1 con el ID del comercio, de esta forma, cuando el usuario al inicie sesión, se comprobará la tabla de Usuarios y al comprobar la casilla, se sabrá que está vinculado a un negocio y cuál de todos es.

PHP:

- Añadir trabajador:

```
$statement = mysqli_prepare($connect, "INSERT INTO ". $tabla ." (operario, nombre, tipo, coste) VALUES (?, ?, ?, ?)");  
mysqli_stmt_bind_param($statement, "ssss", $operario, $nombre, $tipo, $coste);
```

- Editar trabajador:

```
$statement = mysqli_prepare($connect, "UPDATE ". $tablaoperarios ." SET tipo = '$tipo', coste = '$coste'  
WHERE operario = '$operario'");
```

Dart:

- Enviar correo verificación:

```
String username = "*****@*****.com";  
String password = "*****";  
  
final ssmmttppServer = hostinger(username, password);  
  
final message = Message()  
  ..from = Address(username)  
  ..recipients.add(correodestinatario)  
  ..subject = 'Código Verificación SOSFire'  
  ..text = 'Hola ' + nombredestinatario + ", están intentando registrarte en un comercio. El código de verificación es '" + codigoverif + "' (sin las comillas). Por favor, comunícalo al administrador. Si no lo has solicitado, ignora el mensaje."  
  try {  
    final sendReport = await send(message, ssmmttppServer);  
  }
```

figura 49: captura añadir trabajador



5.3.3.5 Estadísticas

Finalidad: La pantalla estadísticas nos permite observar una gráfica con los gastos e ingresos del negocio (derivados de la creación o edición de instalaciones y de los salarios de los trabajadores), se pueden listar, crear nuevos o elegir los de otro mes.

Diagrama de Actividad: Gastos e Ingresos

Clases DC: Estadísticas

Tablas afectadas: 5.3.2.9

Flutter Packages: flutter_cupertino_date_picker, pie_chart

Funcionamiento: Al acceder a esta pantalla, se calculan los gastos, ingresos, gastosextra, ingresosextra y salarios de las tablas de GastosEIngresos y Trabajadores (5.3.2.9 y 5.3.2.5).

- Las cifras de la primera tabla se separan gracias a la casilla Tipo, que distingue entre gastos, gastosextra, ingresos e ingresosextra, tras ello, se suman.
- Las de la segunda, se suman gracias a la casilla salario de la tabla trabajadores.

Una vez realizados los cálculos, se creará un mapeo de datos Clave-Valor, por ejemplo: gastos-3289.0 Estos mapeos los interpreta el paquete pie_chart, el cual elabora la gráfica separando cada mapeo con un color preestablecido. Se permite listar los gastos e ingresos. Para ello, se obtienen las tuplas de la tabla que en la casilla de mesanyo coincide con el seleccionado y se elabora un ListView.

PHP:

- Obtener gastos e ingresos:

```
$tablagastosingresos = mysqli_real_escape_string($connect, $_POST['tablagastosingresos']);
$mesanyo = mysqli_real_escape_string($connect, $_POST['mesanyo']);
$respuesta = array();
$respuesta[$tablagastosingresos] = array();

$stmt = "SELECT * FROM " . $tablagastosingresos . " WHERE mesanyo = '$mesanyo'";
mysqli_stmt_execute($stmt);

$result = mysqli_query($connect, $stmt);

while($row = mysqli_fetch_array($result)){
    $tmp = array();
    $tmp["cifra"] = $row["cifra"];
    $tmp["tipo"] = $row["tipo"];
    $tmp["modelo"] = $row["modelo"];
    $tmp["operario"] = $row["operario"];
    $tmp["fecha"] = $row["fecha"];
}
```

- Obtener salarios:

```
$tablaoperarios = mysqli_real_escape_string($connect, $_POST['tablaoperarios']);
$respuesta = array();
$respuesta[$tablaoperarios] = array();

$stmt = "SELECT * FROM " . $tablaoperarios;
mysqli_stmt_execute($stmt);

$result = mysqli_query($connect, $stmt);

while($row = mysqli_fetch_array($result)){
    $tmp = array();
    $tmp["operario"] = $row["operario"];
    $tmp["nombre"] = $row["nombre"];
    $tmp["tipo"] = $row["tipo"];
    $tmp["coste"] = $row["coste"];
}
```

Dart:

- Obtener gastos, ingresos y salarios:

```
final responseclientes = await http.post(
  "urlphpobtenergastoseingresos",
  body: {
    "tablagastosingresos": tablagastosingresos,
    "mesanyo": fecha,
  });

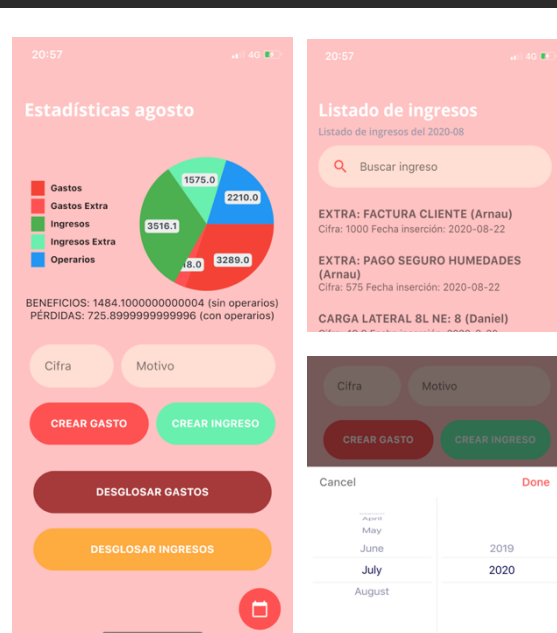
final responseworkers = await http.post(
  "urlphpobtenertrabajadores",
  body: {
    "tablaoperarios": tablaoperarios,
  });
if (responseworkers.statusCode == 200) {
  var DataWorkers = json.decode(responseworkers.body);
  for (var i = 0; i < DataWorkers[tablaoperarios].length; i++) {
    salarios = salarios +
double.parse(DataWorkers[tablaoperarios][i]["coste"]);}
}
if (responseclientes.statusCode == 200) {
  var jsonData = json.decode(responseclientes.body);

  for (var i = 0; i < jsonData[tablagastosingresos].length; i++) {
    GastoIngreso gastoingreso = GastoIngreso(
      jsonData[tablagastosingresos][i]["cifra"],
      jsonData[tablagastosingresos][i]["tipo"],
      jsonData[tablagastosingresos][i]["modelo"],
      jsonData[tablagastosingresos][i]["operario"],);
    gastoingresos.add(gastoingreso);
  }
}
```

- Pasar datos para la gráfica:

```
filterClients() {
  for(int i=0; i< gastoingresos.length; i++){
    if(gastoingresos[i].tipo == "ingreso"){
      ingresos = ingresos + double.parse(gastoingresos[i].cifra);
    }
    if(gastoingresos[i].tipo == "ingresoextra"){
      ingresoextra = ingresoextra + double.parse(gastoingresos[i].cifra);
    }
    if(gastoingresos[i].tipo == "gasto"){
      gastos = gastos + double.parse(gastoingresos[i].cifra);
    }
    if(gastoingresos[i].tipo ==
"gastoextra"){
      gastosextra = gastosextra +
double.parse(gastoingresos[i].cifra);
    }
  }
  dataMap.putIfAbsent("Gastos", () =>
gastos);
  dataMap.putIfAbsent("Gastos Extra",
() => gastosextra);
  dataMap.putIfAbsent("Ingresos", ()
=> ingresos);
  dataMap.putIfAbsent("Ingresos
Extra", () => ingresoextra);
  dataMap.putIfAbsent("Operarios", ()
=> salarios);}
}
```

figura 50: captura estadísticas
figura 51: captura listado ingresos
figura 52: captura escoger mes



5.3.3.6 Instalaciones e Impresión de Etiquetas

Finalidad: Cualquier usuario podrá crear o revisar/modificar la instalación de una sucursal seleccionada, pudiendo imprimir una etiqueta para facilitar la identificación.

Diagrama de Actividad: Instalación

Clases DC: Instalación

Tablas afectadas: 5.3.2.6, 5.3.2.7, 5.3.2.8, 5.3.2.9

Flutter Packages: esc_pos_utils, esc_pos_bluetooth

Funcionamiento: Tras seleccionar una sucursal, consultaremos las tablas de extintoresbocas (5.3.2.7) y centrales (5.3.2.8) discriminando por su id para obtener los productos instalados en esta. Cuando se da de alta una instalación, se consulta el producto escogido en la tabla 5.3.2.6 para generar los gastos o ingresos que derivan de él (añadiéndolo a la tabla 5.3.2.9) y coger sus características principales. Una vez se da de alta, automáticamente se generan las fechas de próximas revisiones.

Al pulsar en un producto instalado, se permitirá la edición de este. Desde este menú, se permite su revisión o retimbre (si le toca) o imprimir una etiqueta para identificarlo de forma mas sencilla. Al editar un producto, las fechas de revisión y retimbre pueden variar.

Si se desea imprimir una etiqueta, se buscarán los dispositivos bluetooth generando una lista para un ListView. Al pulsar en la impresora deseada, se imprimirá la etiqueta con los datos relevantes de esta.

Dart:

- Actualizar fechas (edición, revisión o retimbre):

```
if (tipoedit == "editar") {
  ultimarevision = widget.ultimarevision;
  ultimoretimbre = widget.ultimoretimbre;
  proximarevision = widget.proximarevision;
  proximoretimbre = widget.proximoretimbre;
}

if (tipoedit == "revisar") {
  ultimarevision = DateTime.now().year.toString() + "-" +
  DateTime.now().month.toString() + "-" + DateTime.now().day.toString();
  ultimoretimbre = widget.ultimoretimbre;
  proximarevision = (DateTime.now().year + 1).toString() + "-" +
  DateTime.now().month.toString() + "-" + DateTime.now().day.toString();
  proximoretimbre = widget.proximoretimbre;
  if (proximarevision == proximoretimbre) {
    proximarevision = (DateTime.now().year + 2).toString() + "-" +
    DateTime.now().month.toString() + "-" + DateTime.now().day.toString();
  }
}

if (tipoedit == "retimbrar") {
  ultimarevision = DateTime.now().year.toString() + "-" +
  DateTime.now().month.toString() + "-" + DateTime.now().day.toString();
  ultimoretimbre = DateTime.now().year.toString() + "-" +
  DateTime.now().month.toString() + "-" + DateTime.now().day.toString();
  proximarevision = widget.proximarevision;
  proximoretimbre = (DateTime.now().year + 5).toString() + "-" +
  DateTime.now().month.toString() + "-" + DateTime.now().day.toString();
}
```

- Guardar revisión:

```
final responseproducto = await http
  .post(urlphpextintoresybocas, body: {
    "tablaextintoresbocas": tablaextintoresbocas,
    "idunico": widget.idunico,
    "gasprop": isextintor ? _categoriaEscogida.tipo : "0",
    "capacidad": isextintor ? controlercapacidad.text : "0",
    "ultimarevision": ultimarevision,
    "proximarevision": proximarevision,
    "ultimoretimbre": ultimoretimbre,
    "proximoretimbre": proximoretimbre,
    "observaciones": controlerobservaciones.text,
    "presion": controlerpresion.text,
    "operario": operario,
  });
```

- Buscar dispositivos bluetooth:

```
printerManager.scanResults.listen((devices) async {
  setState() { _devices = devices; });
});
```

- Crear etiqueta:

```
String etiqueta = idinstalacion + idcliente + numequipo;
var rng = new Random();
while(etiqueta.length != 12){
  etiqueta = etiqueta + rng.nextInt(10).toString();
}
```

- Imprimir etiqueta:

```
Future<Ticket> Etiqueta(PaperSize paper) async {
  final Ticket ticket = Ticket(paper);
  ticket.hr();
  ticket.row([
    PosColumn(text: "NE: " + widget.numequipo, width: 4),
    PosColumn(text: widget.fabricacion, width: 4),
    PosColumn(text: widget.operario, width: 4, styles: PosStyles(align:
    PosAlign.right)),]);
  ticket.hr();
  ticket.barcode(Barcode.code128(barData), width: 2, height: 50);
  ticket.text(widget.nombrecliente,
    styles: PosStyles(bold: true, align: PosAlign.center), linesAfter: 2);
  return ticket;}
```



figura 53: captura instalaciones
figura 54: captura revisar instalación

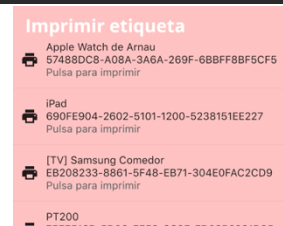


figura 55: captura lista bluetooth
figura 56: etiqueta impresa

5.3.3.7 Mantenimiento y Escaneo de Etiquetas

Finalidad: Proporciona un listado de los clientes, sucursales e instalaciones que requieren de una revisión o retimbre de forma sencilla, también permite escanear el código de barras de un producto para acceder a su ficha de forma rápida.

Diagrama de Actividad: Mantenimiento

Clases DC: Instalación

Tablas afectadas: 5.3.2.3, 5.3.2.4, 5.3.2.7, 5.3.2.8

Flutter Packages: barcode_scan

Funcionamiento: Cuando se accede a la pantalla de Mantenimientos, se realiza una comprobación de todas las instalaciones (tablas 5.3.2.7 y 5.3.2.8) obteniendo las que tienen como fecha de revisión o retimbre el año y mes actual o una fecha anterior.

Una vez obtenidas las instalaciones, se obtienen los ids de los clientes y de las sucursales, de esta forma podremos acceder a las instalaciones de dos maneras:

- Manualmente: aparecerán los clientes, las sucursales y las instalaciones que requieren de revisión.
- Escanear código de barras: con el paquete barcode_scan, se ha logrado implementar un lector que al enfocar la etiqueta, devuelve su número, por lo tanto se puede identificar el producto y acceder a la pantalla de revisión de ese en concreto (gracias a la casilla etiqueta de las tablas 5.3.2.7 y 5.3.2.8) de forma automática.

Dart:

- Escanear etiqueta:

```
var result = await BarcodeScanner.scan();
setState(() {
  this.barcode = result.rawContent;
  for(int i=0; i<extintorbocas.length; i++){
    if(result.rawContent == extintorbocas[i].etiqueta){
      for(int w=0; w<clienteseti.length; w++){
        print(clientes.length);
        if(extintorbocas[i].idcliente == clienteseti[w].id){
          Navigator.push(context, MaterialPageRoute(builder: (context) =>
            EditarExtintor(datosdelextintor)));
        }
      }
    }
  }
});
```

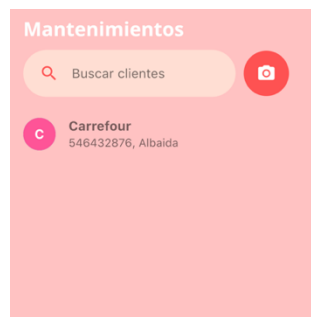


figura 57: captura mantenimientos

figura 58: captura escáner

figura 54: captura revisar instalación

5.3.3.8 Chat

Finalidad: Se incluye una pantalla para intercambiar mensajes entre los usuarios vinculados a un negocio.

Diagrama de Actividad: Chat

Flutter Packages: firebase_auth, cloud_firestore

Funcionamiento: Estas pantallas hacen uso de bases de datos en FireBase y Cloud Firestore, se ha tenido que configurar desde el portal web de FireBase en su modalidad gratuita y vincular al proyecto de FireBase con los métodos que Google proporciona. Al registrarse un administrador, se genera una colección de datos que identifica al administrador y su negocio (con la casilla de comercio de la tabla Usuarios. Cuando se vincula un trabajador a un negocio, esto también sucede.

Una vez se crean las colecciones de datos, al buscar un trabajador para empezar el chat, se selecciona a los usuarios que disponen el id del mismo comercio, mostrando gracias a ello a solo los usuarios de ese negocio. Cuando se pulsa en uno de ellos, se genera otra colección la cual incluye a ambos usuarios y un id único de la sala que los incluye, esa colección, incluye otra colección en su interior la cual almacena los mensajes (identificando el emisor y hora de envío, posicionando los mensajes en orden).

Al acceder a la pantalla de Chats, se hace una búsqueda de las colecciones de chats ya empezados.

Cloud Firestore:

- Estructura sala de chat:

```
idsala: " [redacted]_otmail.es [redacted]@hotmail.comArнау_Daniel"
```

```
usuarios
```

```
[0] "Daniel_[redacted].com"  
[1] "Arнау_[redacted]"
```

- Estructura mensaje dentro de sala:

```
message: "Revisa Carrefour, que tiene una incidencia :("
```

```
sendBy: "[redacted]@hotmail.es"
```

```
time: 1597619292940
```

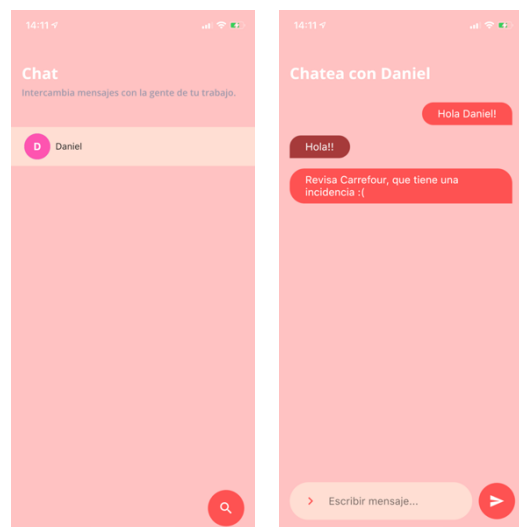


figura 59: captura seleccionar chat

figura 60: captura sala chat

5.4 DISCUSIÓN DEL MÉTODO

Una vez aplicado el método, se va a discutir la calidad de las transformaciones, la expresividad de los modelos y si hay pérdida o no de información en las transformaciones indicando si ha sido necesario añadir información faltante o no.

5.4.1 CIM

iEstrella (i):*

*i** nos devuelve un esquema en el que fácilmente se pueden observar las metas organizacionales junto sus tareas a realizar para obtenerlas y los requisitos especificados del sistema, mostrándonos la relación entre estos.

Se trata de un buen modelo para empezar un desarrollo, ya que nos permite de un solo vistazo, ver qué acciones del mundo real (las que realiza un actor) interviene en el sistema para conseguir lograr sus metas. El único pero es que a diferencia de otros modelos, las relaciones entre las tareas de un mismo actor pero de diferentes metas, han sido algo difíciles de especificar aunque tampoco tiene mucha relevancia especificarse en este tipo de modelos.

Transformación i-MCU:*

En los Casos de Uso en cambio solo aparecen las funciones finales del sistema. Se pierde la finalidad del sistema ya que al no hablar de metas y simplemente de funciones, no se sabe la finalidad para la cual se está implementando, simplemente aparecen los actores del sistema y qué funciones van a realizar.

No es necesario incluir funciones que no tengan un gran impacto en el sistema (como hacer *login* o un sistema de chat) en cuanto a requisitos funcionales. En esta transformación hemos perdido la figura del Cliente ya que no forma parte del sistema pero sí de la organización, ya que como hemos dicho, no se representa la finalidad del sistema (darle un servicio contra incendios al cliente).

Tampoco se observan los requisitos no funcionales ni las acciones automáticas que sí que están en el *i** como por ejemplo, que al realizar una instalación el cambio de productos en el stock se realice de forma automática. Simplemente se introduce a grandes rasgos que el stock se va a poder gestionar.

Es decir, MCU es como un modelo en el que se tienen que especificar las acciones en general que más peso tienen en el sistema. Para representar a un usuario no administrador o operario, se ha tenido que introducir de forma manual el CU de Registro, ya que en *i** no aparece tal opción como meta organizacional.

5.4.2 PIM

Transformación CIM-DC

El diagrama de clases nos permite observar de un vistazo, las clases que componen el sistema junto a sus atributos y las operaciones de estas.

Al aplicar las transformaciones, los actores quedan mas difusos ya que este modelo trata de especificar el esquema del sistema y no tanto las acciones de este.

En estas transformaciones, se ha tenido que recurrir bastante a los flujos principales de los casos de uso para poder obtener las clases junto a sus atributos y operaciones, pero aún así, se han ido añadiendo más de forma manual porque mediante transformaciones directas no podían obtenerse.

Nos permite una mejor visualización de las herencias entre las clases.

Transformación CIM-DA

Con el diagrama de actividad podemos recrear los flujos de acciones de las actividades que brinda el sistema a implementar.

Las transformaciones se han aplicado sobretodo desde los MCU y sus flujos principales o normales, ya que especifican bastante bien como van a ser las tareas que componen los casos de uso, los cuales se pueden extrapolar a actividades y tareas en los diagramas de actividad.

Las pérdidas en estas transformaciones, son la pérdida de atributos y tipos de datos que van a manejarse, aunque ganamos obtener de forma mas clara de las acciones que se van a realizar de forma ordenada para completar las actividades que componen el sistema.

En este caso, se ha decidido añadir manualmente la actividad de Chat, donde se intercambian mensajes entre trabajadores del mismo negocio. Al ser una tarea con poca repercusión en la organización del sistema, no se ha añadido en diagramas anteriores como el de CU o DC.

Transformación DC,DA-Mockups

A partir de las clases y actividades, se ha realizado un prototipado mediante mockups.

Gracias a las clases y atributos del DC, se pueden componer de una forma mas clara el contenido de las pantallas que compondrán el sistema. Para sus relaciones, se han seguido los flujos de acciones de los DA.

Se pierde el tipo de datos que manejan las pantallas y el flujo entre pantallas, ya que se muestran prototipos independientes entre sí y no en su totalidad ya que es una versión preliminar.

Es una muy buena aproximación para hacernos mejor a la idea de la implementación, tanto de la UI como del código fuente que compondrá el sistema.

5.4.3 PSM

Transformación PIM-PSM

En la fase final del PSM, se han realizado transformaciones del nivel superior de forma que se obtienen diferentes componentes de código para el sistema. Encontramos tres diferentes:

- **Bases de Datos:** gracias a los diagramas de clase, se ha podido realizar la versión preliminar de las bases de datos final. Estas se han ido adaptando conforme avanzaba el proyecto para una mejor obtención y relación de los datos, pero ha sido un muy buen punto de partida para organizarlos y observar sus relaciones y herencias.
- **Componentes de pantalla:** los *mockups* nos han servido como una gran base que, gracias a las múltiples funciones que *Flutter* proporciona para los diseños de interfaz (junto *Sketch*), nos ha permitido implementar una interfaz agradable, amigable y sencilla tal y como se pretendía.

Al tener creados los *mockups* de antemano, han servido para observar qué pantallas iban a tener componentes parecidos como los de listar: listar clientes, sucursales, instalaciones, productos. Por lo que se ha podido reutilizar código entre esas pantallas para que con unos pequeños ajustes, puedan mostrar las diferentes clases de objetos que se muestran en la lista.

- **Código del sistema:** gracias a los diagramas de actividad y clase, ha sido posible la reutilización de código. Por ejemplo, muchas pantallas que representan a clases, tenían las acciones de crear, editar y eliminar. Para todas ellas, se ha utilizado la misma base de código, tanto en *Flutter* como en PHP y adaptándolo a los atributos de las clases, nos ha permitido ahorrar tiempo y obtener menos errores.

CAP6. PRUEBAS Y RESULTADOS

En este capítulo nos centraremos en las pruebas de la herramienta. Se comprobará su solidez y cohesión de acuerdo a los requisitos solicitados, introduciendo y realizando acciones fuera de los rangos aceptados para comprobar el comportamiento del sistema.

6.1 PRUEBAS

Se van a realizar tres tipos de pruebas del sistema implementado:

- Primero se harán unas pruebas de usabilidad de los casos de uso, se comprobarán las pantallas que lo componen, indicando su funcionamiento normal y haciendo pruebas con datos en mal formato, ejecuciones diferentes y lo que se obtiene con estas.
- Después se les mandará el archivo de instalación para Android (Apple es mas restrictiva y necesitas hacer varios pasos para instalar una aplicación de desarrollador) a siete personas trabajadores en el sector, tras 48 horas aproximadamente, se les enviará un cuestionario para responder, con el cual se valorarán los resultados de los que extraeremos unas conclusiones.
- Finalmente, se comprobará cómo se adapta a diferentes pantallas de distintas pulgadas, sistemas operativos y resoluciones:
 - Dispositivo 1: 6.2", Android, 2340x1080
 - Dispositivo 2: 5.96", Android, 2560x1440
 - Dispositivo 3: 5.8", iOS, 2436x1125
 - Dispositivo 4: 9.7", iOS, 2048x1536

6.1.1 Pruebas de Usabilidad

6.1.1.1 Registrarse

Los usuarios que no estén dados de alta en el sistema podrán hacerlo seleccionando el tipo de usuario que son, tras ello, se configurará el comercio si se escoge administrador. La herramienta tiene que cerciorarse de que los datos son válidos y que la cuenta no existe previamente.

Pruebas:

- Introducir datos diferentes en las casillas de correo y contraseñas.
- Introducir un correo sin formato de correo electrónico.
- Intentar dar de alta usuarios con una cuenta creada.
- Dejarse datos por rellenar.

Resultados:

- No se han logrado obtener resultados fuera de los esperados, se ha implementado el sistema de forma que compruebe que las casillas no están vacías, que los correos y contraseñas coinciden y que no hay usuarios duplicados en la tabla.

6.1.1.2 Gestionar Productos

El administrador podrá dar de alta los productos que podrán utilizarse en las instalaciones. Estos tendrán datos como cantidad, cantidad mínima de aviso, coste, precio de venta al público, etc. Al crearse un producto y añadir su stock, se generará un gasto. Al editarse un producto, se podrán generar gastos o ingresos según disminuya o se incremente su stock de cantidad. También se permite editarlos y eliminarlos.

Pruebas:

- Dar de alta un producto siendo usuario Operario.
- Poner símbolos raros en las casillas de precios.
- Poner números negativos en el stock.
- Cambiar la categoría de un producto múltiples veces.

Resultados:

- Los usuarios que inician sesión como operarios no disponen del botón para añadir un producto o editar un producto existente por lo que directamente no se puede intentar la acción.
- No se permite la introducción de símbolos extraños en las casillas de precios o cantidad ya que los teclados virtuales se encuentran en modo teclado numérico, solo siendo posible introducir puntos y comas. Cuando se añade una cifra con una coma en vez de un punto (9,7 en vez de 9.7), estos decimales quedan suprimidos por lo que es un fallo. Cuando hay mas de una coma o punto, no se realiza ninguna acción de añadir.
- Se pueden introducir números negativos en el stock, no ha sido una restricción de los requisitos y podría haberse planteado limitar el stock a 0 como mínimo pero no se ha realizado ya que muchos comercios reciben los albaranes de los productos y los introducen a la larga, por lo que limitar a 0 el stock podría traer mas problemas que soluciones.
- La categoría se cambia sin problemas, puedes cambiar el tipo de producto sin fallos de ningún tipo.

6.1.1.3 Gestionar Fichas de Clientes

El administrador podrá dar de alta los clientes, editarlos, eliminarlos, añadir incidencias. Una vez dado de alta un cliente, cualquier usuario podrá administrar sus sucursales e instalaciones.

Pruebas:

- Gestionar los clientes siendo Operario.
- Dejar datos sin rellenar.
- Poner dos veces un mismo cliente.

Resultados:

- No se permite gestionar los clientes por parte de los operarios, no disponen de los botones y opciones necesarias ya que al acceder a la pantalla se comprueba qué tipo de usuario es y si es operario, se ocultan las opciones del administrador. Sí que puede editar las sucursales (crearlas, editarlas y darlas de baja).
- No se permite dejar datos sin rellenar.
- Se permite introducir el mismo cliente varias veces, puede que haya dos clientes con el mismo nombre y no se han introducido mecánicas que comprueben si el correo del cliente ya existe en otro. Cada cliente aunque sea con el mismo nombre tendrá su id único, por lo que no habrá un solape de datos en las sucursales o instalaciones.

6.1.1.4 Gestionar Trabajadores

El administrador puede vincular trabajadores a su negocio y editar sus datos como el salario o rol.

Pruebas:

- Gestionar los trabajadores siendo Operario.
- Dejar datos sin rellenar.
- Añadir mas de una vez a un trabajador.
- Introducir un correo electrónico de alguien no registrado.
- Introducir valores erróneos en la casilla de correo electrónico.
- Introducir un código de verificación erróneo.
- Dar de baja a si mismo o a todos los administradores.

Resultados:

- No se permite gestionar los trabajadores por parte de los operarios, no se les permite acceder a la pantalla de ajustes.
- Se puede dejar sin rellenar el salario (será 0), pero el código de verificación debe de existir y coincidir con el enviado, por lo que un código erróneo no es válido.
- No se permite añadir mas de un trabajador, se comprueba que no esté vinculado al mismo negocio antes.
- Al introducir un correo electrónico, este comprueba que exista en la tabla y también que tenga formato de correo electrónico, si no, devuelve error.
- No se permite dar de baja a si mismo, de forma que siempre como mínimo quedará un administrador vigente para evitar errores.

6.1.1.5 Estadísticas

El administrador puede visualizar los gastos e ingresos del negocio, separando estos por mes.

Pruebas:

- Escoger un mes sin gastos.
- Introducir gastos o ingresos extra con formato erróneo.
- Introducir cifras con decimales absurdos.

Resultados:

- Al escoger un mes sin gastos, la gráfica se oculta y aparece un texto que indica la falta de datos. Al listar los gastos o ingresos, no aparecerá ningún elemento.
- No se permite introducir gastos o ingresos extra con formato erróneo ya que se fuerza al uso de cifras.
- No se ha acertado el resultado de los gastos e ingresos, por lo que las cifras si que pueden aparecer con muchos decimales.

6.1.1.6 Instalaciones

Cualquier usuario vinculado al negocio puede crear, editar o dar de baja una instalación. También se permite imprimir una etiqueta mediante impresoras térmicas bluetooth.

Pruebas:

- Crear sucursales e instalaciones y editarlas siendo operario.
- Instalar mas de un producto igual en una sucursal.
- Seleccionar un dispositivo bluetooth que no es una impresora.

Resultados:

- Cualquier usuario puede crear y editar sucursales o instalaciones.
- Se permite instalar el mismo producto varias veces en una sucursal. Las instalaciones son únicas pero pueden apuntar al mismo producto dado de alta del administrador, cada instalación tiene sus características propias, aunque herede las de un tipo de producto.
- No sucede nada, no se muestra ningún error pero tampoco ocurre nada.

6.1.1.7 Mantenimientos

Cualquier usuario vinculado al negocio puede revisar una instalación con revisiones pendientes. También se permite escanear una etiqueta para acceder a la ficha de la instalación de forma rápida.

Pruebas:

- Acceder a instalaciones que no tengan revisión pendiente.
- Escanear una etiqueta que no es del negocio.

Resultados:

- Se permite acceder a las fichas de las instalaciones en cualquier momento desde la pantalla de Clientes, desde Mantenimientos solo aparecerán los clientes y las sucursales de este que contengan instalaciones con revisiones o retimbres pendientes, para facilitar el acceso.
- No accederá a ninguna ficha, volverá al listado de Mantenimientos sin mostrar ningún error en vez de acceder a la ficha de revisión de una instalación.

6.1.2 Cuestionario de Usabilidad

Para comprobar que el sistema cumple con lo solicitado, se ha realizado una prueba de usabilidad con usuarios reales trabajadores del tipo de negocio para el cual se ha realizado el desarrollo de la herramienta. Los encuestados son siete varones de edades comprendidas entre 25-55 años de edad. Han utilizado dispositivos Android gracias a que permite la instalación de aplicaciones de desarrollador de forma más sencilla que en iOS. Las preguntas han sido las siguientes:

1. De manera general, estoy satisfecho/a con lo fácil que es utilizar esta aplicación
2. Es sencillo utilizar la aplicación
3. Puedo completar mi trabajo eficazmente utilizando esta aplicación
4. Soy capaz de completar mi trabajo rápidamente utilizando esta aplicación
5. Soy capaz de completar mi trabajo efectivamente utilizando esta aplicación
6. Me siento cómodo utilizando esta aplicación
7. Fue fácil aprender a usar esta aplicación
8. Creo que me volví productivo/a rápidamente utilizando esta aplicación
9. La aplicación da mensajes de error que me dicen claramente cómo arreglar los problemas
10. Siempre que cometo un error utilizando la aplicación, me puedo recuperar fácil y rápidamente
11. La información (ayuda en línea, mensajes en pantalla y otra documentación) en la aplicación es clara

12. Es fácil encontrar la información que necesito
13. La información provista por la aplicación es fácil de entender
14. La información es efectiva al ayudarme a completar mi trabajo
15. La organización de la información en la pantalla de la aplicación es clara
16. La interfaz de la aplicación es agradable
17. Me gusta utilizar la interfaz de esta aplicación
18. La aplicación tiene todas las funciones y capacidades que yo espero que tenga
19. De manera general, estoy satisfecho/a con la aplicación
20. Escribe, si lo deseas, los tres aspectos mas negativos
21. Escribe, si lo deseas, los tres aspectos mas positivos
22. Comentarios o sugerencias

Cómo se puede ver, las preguntas nos permiten hacernos una idea de la eficiencia y eficacia al utilizar la aplicación, comprobando si la información que aparece pantalla es buena y si la interfaz es agradable y usable o se convierte en un lastre para su uso.

Las puntuaciones van del 1 al 7, siendo 1 el nivel mas bajo (fuertemente desacuerdo) y 7 el mas alto (fuertemente de acuerdo). Las respuestas han sido las siguientes:

- 1. De manera general, estoy satisfecho/a con lo fácil que es utilizar esta aplicación.**
 - 7: 3 personas
 - 6: 4 personas

- 2. Es sencillo utilizar la aplicación**
 - 7: 6 personas
 - 6: 1 persona

- 3. Puedo completar mi trabajo eficazmente utilizando esta aplicación**
 - 7: 1 persona
 - 6: 3 personas
 - 5: 3 personas

- 4. Soy capaz de completar mi trabajo rápidamente utilizando esta aplicación**
 - 7: 1 persona
 - 6: 3 personas
 - 5: 3 personas

- 5. Soy capaz de completar mi trabajo efectivamente utilizando esta aplicación**
7: 1 persona
6: 2 personas
5: 4 personas
- 6. Me siento cómodo utilizando esta aplicación**
7: 3 persona
6: 4 personas
- 7. Fue fácil aprender a usar esta aplicación**
7: 6 personas
6: 1 persona
- 8. Creo que me volví productivo/a rápidamente utilizando esta aplicación**
6: 4 personas
5: 3 personas
- 9. La aplicación da mensajes de error que me dicen claramente cómo arreglar los problemas**
7: 3 personas
6: 4 personas
- 10. Siempre que cometo un error utilizando la aplicación, me puedo recuperar fácil y rápidamente**
7: 4 personas
6: 2 personas
4: 1 persona
- 11. La información (ayuda en línea, mensajes en pantalla y otra documentación) en la aplicación es clara**
7: 4 personas
6: 2 personas
5: 1 persona
- 12. Es fácil encontrar la información que necesito**
7: 5 personas
6: 2 personas
- 13. La información provista por la aplicación es fácil de entender**
7: 5 personas
6: 2 personas

14. La información es efectiva al ayudarme a completar mi trabajo

7: 5 personas

6: 1 persona

5: 1 persona

15. La organización de la información en la pantalla de la aplicación es clara

7: 7 personas

16. La interfaz de la aplicación es agradable

7: 7 personas

17. Me gusta utilizar la interfaz de esta aplicación

7: 7 personas

18. La aplicación tiene todas las funciones y capacidades que yo espero que tenga

7: 3 personas

6: 2 personas

5: 2 personas

19. De manera general, estoy satisfecho/a con la aplicación

7: 2 personas

6: 5 personas

20. Escribe, si lo deseas, los tres aspectos mas negativos

- Faltan por pulir detalles
- La integración de artículos se hace de uno en uno
- Al salir de donde estas vuelve a la pantalla principal
- Al introducir un artículo toma de fecha de última revisión, la fecha de fabricación
- No deja introducir el número individual de artículo
- Falta poder crear ordenes de trabajo por meses
- No es posible editar gastos o ingresos
- Al cambiar cosas vuelve al menú principal
- No esconde el teclado en ocasiones
- Al crear una instalación se añaden los artículos de forma individual
- Hay campos que faltan por detallar mejor
- Vuelve al menú al realizar acciones

21. Escribe, si lo deseas, los tres aspectos mas positivos

- Ahora se realiza el mismo trabajo de forma más eficiente.
- Evitamos papeleo engorroso.
- Se puede acceder a la información necesaria sin tener que llamar a oficina.

- Digitaliza mi empresa lo que supone ahorrar en costes de traspaso de papel a ordenador.
- En todo momento puedo mirar en el móvil lo que antes necesitaba mirar en un ordenador.
- Es muy intuitiva y fácil de usar.
- Hace que la empresa realice el trabajo más eficientemente.
- En un momento puedes ver la rentabilidad de la empresa.
- Interfaz
- Rapidez código de barras
- El diseño
- Ahorra tener que depender de un ordenador
- Lo de escanear y imprimir etiquetas

22. Comentarios o sugerencias

- Implementar la integración de varios artículos a la vez
- Que se puedan añadir productos a las instalaciones en lotes y no uno a uno
- Que no vuelva al menú inicial cuando cambies o añades cosas

Se va a proceder a valorar los resultados. Se tiene que tener en cuenta el poco tiempo de uso ya que con 48 horas no pueden simular completamente cuál sería el uso del sistema en su día a día, pero permite hacer una valoración general temprana para poder corregir errores en un futuro. Los resultados han sido los siguientes:

- En general, la aplicación vuelve mas eficientes a los trabajadores que la utilizan, evitan depender de papeles y de la centralización del sistema a ordenadores que solo están en la oficina.
- La interfaz es del agrado para los trabajadores, permitiéndoles un control fácil de las opciones que el sistema presenta.
- La información que se presenta en la aplicación es adecuada y está correctamente mostrada.
- La herramienta tiene errores que en futuras versiones podrían subsanarse como que tras insertar un artículo o editarlo no vuelva al menú, que permita añadir mas de un artículo de una vez, editar gastos o ingresos, el teclado que permita ocultarse en todo momento...

En términos generales, el sistema cumple con todos los requisitos solicitados por la organización satisfaciendo las tareas que se necesitan llevar a cabo. Un uso mas prolongado, añadiendo clientes, productos, etc. a gran escala para su uso real, puliendo detalles y ampliando aún más opciones, mejoraría las valoraciones y podría hacerla aún mas usable. Para tratarse de una primera versión, se dan por cumplidos los objetivos de la aplicación del método.

6.1.4 Prueba de compatibilidad

Uno de los aspectos solicitados en los requisitos, es la compatibilidad con sistemas iOS y Android, por lo que se procede a comprobar la interfaz en distintos dispositivos de diferentes pantallas, resoluciones y sistemas operativos. Como se podrá observar, se adapta de forma general, perfectamente a todo tipo de pantallas y sistemas operativos.

- o **Dispositivo 1:** 6.2", Android, 2340x1080



figura 61: dispositivo 1-1

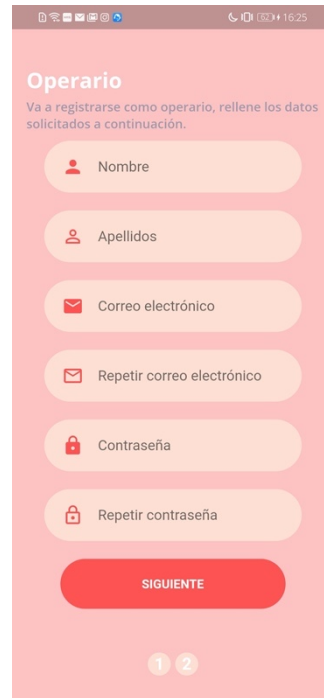


figura 62: dispositivo 1-2



figura 63: dispositivo 1-3



figura 64: dispositivo 1-4

- **Dispositivo 2:** 5.96", Android, 2560x1440

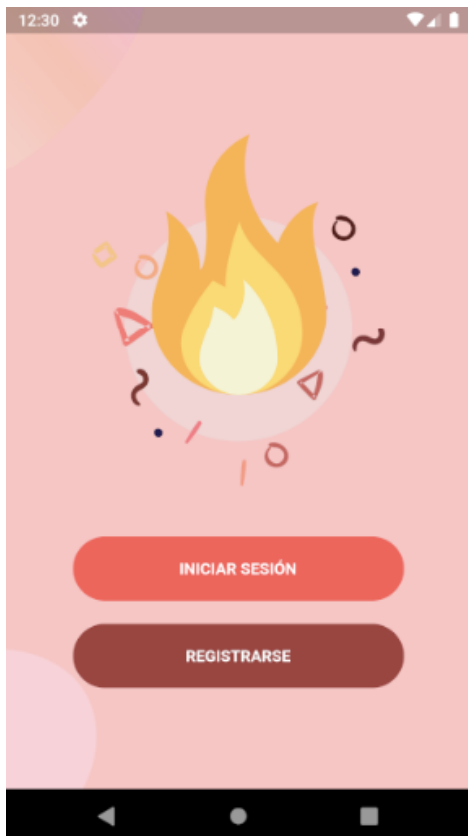


figura 65: dispositivo 2-1



figura 66: dispositivo 2-2



figura 67: dispositivo 2-3

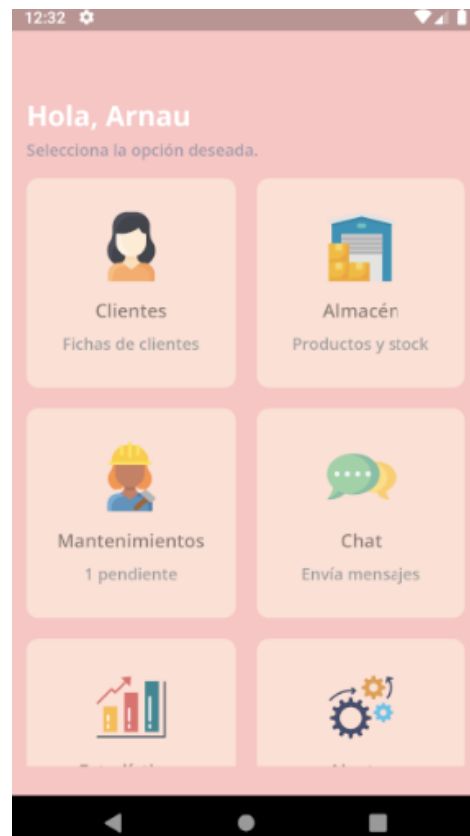


figura 68: dispositivo 2-4

○ **Dispositivo 3:** 5.8", iOS, 2436x1125



figura 69: dispositivo 3-1



figura 70: dispositivo 3-2



figura 71: dispositivo 3-3



figura 72: dispositivo 3-4

○ **Dispositivo 4:** 9.7", iOS, 2048x1536

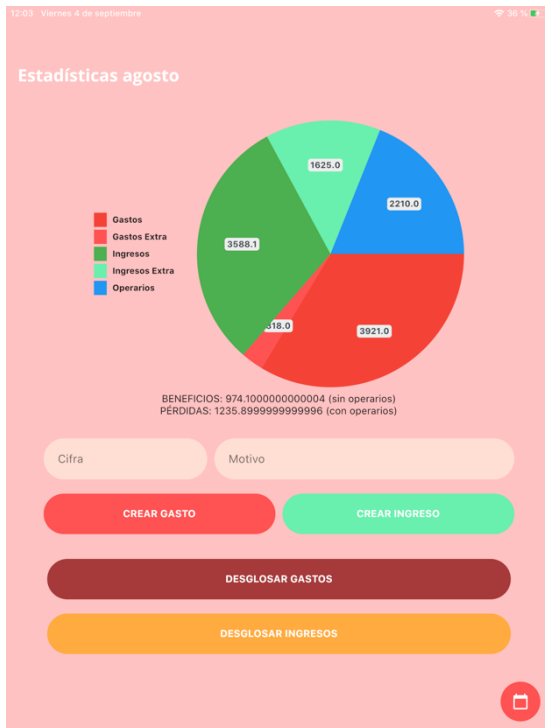


figura 73: dispositivo 4-1

Revisar Producto
 Revisar CARGA LATERAL 8L de la sucursal.

Tipo de gas: CO2

61

90

Últ. revisión: 2020-08-15 Próx: 2020-07-15
 Últ. retimbre: 2020-08-15 Próx: 2025-08-15

todo bienn

Operario: Arnau

REVISAR RETIMBRAR

EDITAR SIN REV/RET

ELIMINAR

figura 74: dispositivo 4-2

-
- Clientes**
- Cientes
- A** Alcampo
655544433, Castellón
 - B** Bahía
965432108, Alcoi
 - C** Carrefour
546432876, Albaida
 - C** Colegio Salesianos
999886777, 666333222, Madrid
 - D** Daniel Martinez
645865722, hola
 - D** Decathlon
66655444, Xàtiva
 - F** Felisano
93400000, València
 - F** Ferrería San Roque
965509231, Alcoi
 - V** Volkswagen
965432109, Barcelona
 - Z** Zara
98655444, Santiago de Compostela
 - Z** Zara Home
123456789, Galicia

figura 75: dispositivo 4-3

Chatea con Daniel

Hola Daniel!!

Hola!!

Revisa Carrefour, que tiene una incidencia :(

hola

Escribir mensaje...

figura 76: dispositivo 4-4

7. CONCLUSIONES

Tras los resultados obtenidos, podemos concluir con la validación del método de desarrollo software extendido objetivo de estudio.

Este método, nos permite abarcar todos los aspectos del desarrollo software, permitiendo desde un principio que se garantice el cumplimiento de requisitos organizacionales solicitados al llegar al nivel más bajo de la MDA. Al tratarse de transformación de modelos, podemos observar la trazabilidad de un modelo a otro, de forma descendente (CIM-PIM-PSM) o ascendente (PSM-PIM-CIM) gracias a las reglas de transformación, las cuales permiten observar los cambios entre modelos.

El modelo *i** nos ha permitido observar desde otros puntos de vista lo que se desea conseguir. Este tipo de modelado no lo había observado a lo largo de mis estudios, pero considero que tras aprender a utilizarlo, es muy útil a la hora de empezar a desarrollar el software. *i** nos muestra de una forma más esquematizada qué objetivos son los que se desean cumplir y cómo se cumplen actualmente, de forma que nos permite partiendo de este, elaborar otros modelos de una manera mas clara que directamente partiendo de una lista de requisitos funcionales y no funcionales, ya que estos no explican ni con qué finalidad se hace ni cómo se hace para lograr esas metas.

Se concluye que las reglas de transformaciones son una muy buena forma de desarrollar modelos a partir de otros modelos, ya que de un simple vistazo podemos ver qué cambios hay de uno a otro. Pese a que entre distintos modelos se encuentren pérdidas (debido a la naturaleza de cada modelo), se puede especificar mediante reglas que esos cambios se han realizado de forma manual, de forma que la trazabilidad entre modelos sigue cumpliéndose.

Las tecnologías que he utilizado para la implementación del código como *Flutter* y el lenguaje *dart*, pese a ser la primera vez que las he usado, me han hecho darme cuenta del avance de la tecnología en ámbitos de desarrollo software. Hasta hace no mucho, era impensable que un mismo código se pudiese utilizar en ambas plataformas móviles y encima utilizara recursos nativos de cada una de ellas. *React Native* era una aproximación, pero en varias partes del código se necesita especificar el tipo de plataforma al que se dirige, en *Flutter* no hace falta. El lenguaje *dart* me ha parecido bastante fácil de aprender debido a sus similitudes con JavaScript en la estructuración de código.

Al ingeniero software se le solicitarán desarrollos cada vez más rápido con más y más características según la tendencia hasta ahora. El MDD usando *i** como punto de partida, es una muy buena solución como se ha podido comprobar a lo largo del proyecto.

8. REFERENCIAS

1. *Model Driven Engineering Aplicado a Business Process Management*, Jose Manuel Pérez, Francisco Ruiz, Mario Piattini: https://www.researchgate.net/publication/242714903_Model_Driven_Engineering_Aplicado_a_Business_Process_Management
2. *Enterprise Resource Planning (ERP)*, Olivia Labarre: <https://www.investopedia.com/terms/e/erp.asp>
3. Flutter: <https://flutter-es.io>
4. PHP: <https://www.php.net>
5. MySQL: <https://www.mysql.com>
6. *ICTs in companies: evolution of technology and structural change in organizations*, Galo E. Cano Pita, Mariana J. García Mendoza: https://www.researchgate.net/publication/336001364_Las_TICs_en_las_empresas_evolucion_de_la_tecnologia_y_cambio_estructural_en_las_organizaciones#read
7. Digital 2019, HootSuite: <https://hootsuite.com/es/pages/digital-in-2019>
8. iStar/iEstrella/i*: http://istarwiki.org/tiki-view_articles.php
9. SQL Pyme: <https://www.distribok.com/aplicaciones/sectores/extintores/>
10. Hostinger: <https://www.hostinger.es>
11. FireBase: <https://firebase.google.com>
12. Sketch: <https://www.sketch.com>
13. Model-Driven Architecture, Oscar Pastor, José Ignacio Panach, Sergio España, Nathalie Aquino: <https://doi.org/10.1007/s00287-008-0275-8>
14. *Analytical Method of CIM to PIM Transformation in Model Driven Architecture (MDA)*, Martin Kardoš, Matilda Drozdová: https://www.researchgate.net/publication/282699811_Analytical_method_of_CIM_to_PIM_transformation_in_model_driven_architecture_MDA

15. *Model-driven architecture in practice - a software production environment based on conceptual modeling*, Oscar Pastor, Juan Carlos Molina: Springer 2007, ISBN 978-3-540-71867-3
16. *Models and Transformations in MDA*, Yashwant Singh, Manu Sood:
https://www.academia.edu/3497213/Models_and_Transformations_in_MDA
17. *Diagramas UML de Casos de Uso y Requisitos*, Javier Jesús Gutiérrez Rodríguez:
http://www.lsi.us.es/~javierj/cursos_ficheros/metricaUML/CasosUsoUML.pdf
18. *Modelado Conceptual – Diagrama de Clases*, M^a Carmen Penadés: <https://www.youtube.com/watch?v=JioEGJllg88>
19. *Model-Driven Development: Processes and practices*, Päivi Parviainen, Juha Takaio, Susanna Teppola & Maarit Tihinen:
<https://cris.vtt.fi/en/publications/model-driven-development-processes-and-practices>
20. [Uhl, 2008] Uhl, A., *Model-Driven Development in the Enterprise*, IEEE Software, January/ February 2008
21. [Hailpern & Tarr, 2006] Hailpern, B., Tarr, P., *Model-driven development: the good, the bad and the ugly*, IBM Systems Journal, Vol. 45, No. 3, pp. 451–461
22. *Model-Driven Challenges & Solutions*, Juha-Pekka Tolvanen and Steven Kelly:
https://www.metacase.com/papers/Tolvanen_Kelly_MODELSDWAR_D_2016.pdf
23. *Why you should implement Model Driven Development*, Johan Den Haan:
<http://www.theenterprisearchitect.eu/blog/2009/11/25/15-reasons-why-you-should-start-using-model-driven-development/>
24. StateFlow, MathWorks:
<https://es.mathworks.com/products/stateflow.html>
25. Visual Paradigm: <https://www.visual-paradigm.com>
26. *Towards a Holistic Conceptual Modelling-Based Software Development Process*, ER 2006: 437-450, Sergio España, Jose Ignacio Panach, Inés Pedriva, Oscar Pastor.









27. LabView: <https://www.ni.com/es-es/shop/labview.html>

28. WebRatio: <https://www.webratio.com/site/content/es/home>

9. ANEXO

9.1 RECURSOS SVG

Se adjunta una tabla con los recursos SVG utilizados para la implementación del código de las pantallas, el nombre del autor o de la autora y el enlace hasta el recurso. Los que no aparecen, son creación, edición propia o son libres de derechos:

Recurso	Autor	Enlace
	Becris	https://www.flaticon.com/authors/becris/
	Freepik	https://www.freepik.com
	Freepik	https://www.freepik.com
	Freepik	https://www.freepik.com
	Freepik	https://www.freepik.com
	Freepik	https://www.freepik.com
	Freepik	https://www.freepik.com
	Iconixar	https://www.flaticon.com/authors/iconixar/

	Nhor Pai	https://www.flaticon.com/authors/nhor_phai/
	smallikeart	https://www.flaticon.com/authors/smallikeart/
	Vitaly Gorvachev	https://www.flaticon.com/authors/vitaliy_gorvachev/
	Vitaly Gorvachev	https://www.flaticon.com/authors/vitaliy_gorvachev/