Gestión de inventario, stock y almacenes, una alternativa moderna

Nombre Estudiante: Julian Gernun

Master Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Nombre Consultor/a: Roger Montserrat Ribes

Profesor/a responsable de la asignatura: Carles Garrigues Olivella

7 de Junio de 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Gestión de inventario, stock y almacenes, una alternativa moderna*
Nombre del autor:	*Julian Gernun*
Nombre del consultor/a:	*Roger Montserrat Ribes*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	07/2017
Titulación::	Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Idioma del trabajo:	*Castellano*
Palabras clave	*Stock, inventario, artículos*

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Debido a la alta competitividad en el ecosistema empresarial, hoy en día es importante ser ágil y eficiente. Una de las maneras de obtener estas cualidades es la renovación tecnológica. El continuo avance en el sector IT hace posible que se realicen las mismas tareas de una manera muchísimo más rápida que hace unos años. Uno de esos avances, han sido las aplicaciones híbridas, las cuales permiten a los desarrolladores de software crear aplicaciones para todos los dispositivos mediante tecnologías Web.

Este trabajo pretende cubrir la necesidad de gestionar el stock de forma moderna y gratuita en pequeñas y medianas empresas. El usuario será capaz, mediante una interfaz intuitiva y simple, de poder gestionar de manera precisa las entradas y salidas de todos los artículos de su/s almacén/es desde cualquiera de sus dispositivos. La mayoría de las aplicaciones de este tipo no tienen todas las funcionalidades de las que dispondrá esta, solamente están en inglés o son de pago, haciendo que la aplicación híbrida sobre la que trata este Trabajo sea atractiva para el mercado actual.

1.2 Objetivos del Trabajo

Se pretende mostrar lo aprendido durante el Master, particularmente mejorar en la parte más débil, las aplicaciones híbridas. El objetivo principal de este Trabajo es lograr organizar bien el tiempo, logrando entregar todo lo previsto sin que la calidad final del código se vea afectada.

Se implementarán las siguientes funcionalidades:

- Autenticación usuarios: Para que pueda obtener/gestionar sus datos desde cualquier dispositivo.

- Añadir/editar artículos: Una vez añadidos, se podrán editar en cualquier momento.

- Escáner EAN (código de barras): Se usará la API http://barcodefinder.com/api para identificar los artículos en primera instancia.

- Componente Web para listados: Debe ser único y configurable mediante una configuración lo más simple posible.

- Añadir/editar proveedores

- Añadir/editar almacenes

- Crear inventario: Artículos en un almacén concreto.

- Crear pedidos: Artículos a pedir a un proveedor concreto.

- Crear recepción: Artículos recibidos por un proveedor.

- Crear recepción a partir de pedido: Comparar artículos recibidos por un proveedor a partir de un pedido creado.

- Crear mermas: Artículos por proveedor y el tipo de merma (caducado, estropeado,…)

- Exportación de un listado a formato CSV

*Dentro de cada funcionalidad se incluye la parte back-end, de ahí que no exista una tarea específica para la creación de la base de datos.

1.3 Enfoque y método seguido

Aunque la naturaleza continua del Trabajo lo dirija hacía la metodología de desarrollo en cascada, se ha preferido utilizar otro tipo de proceso. Al ser un único desarrollador y dada su experiencia con metodologías ágiles, se ha preferido utilizar Kanban. Se irán creando tareas a medida que aparezca su necesidad y se irán solucionando hasta que todos los requisitos estén implementados. Además, en caso de encontrar problemas de diseño a posteriori, se podrá adaptar dependiendo de las necesidades en el momento que ocurran.

Se entregarán las tareas necesarias para cada una de las PECs, viendo cada una como un proyecto diferente dentro del Trabajo. Es decir, cada PEC tendrá su listado de tareas que se gestionarán como está definido por los principios del método Kanban.

Se desarrollará un producto nuevo, aunque se incluirán frameworks que se hayan utilizado durante el Master y librerías ya existentes para facilitar las tareas.

1.4 Planificación del Trabajo

Para el Trabajo se necesita de un servidor con:

- NodeJS para la API Rest y el servidor HTTP
- MySQL para los datos usados por la API Rest.

Figura 1

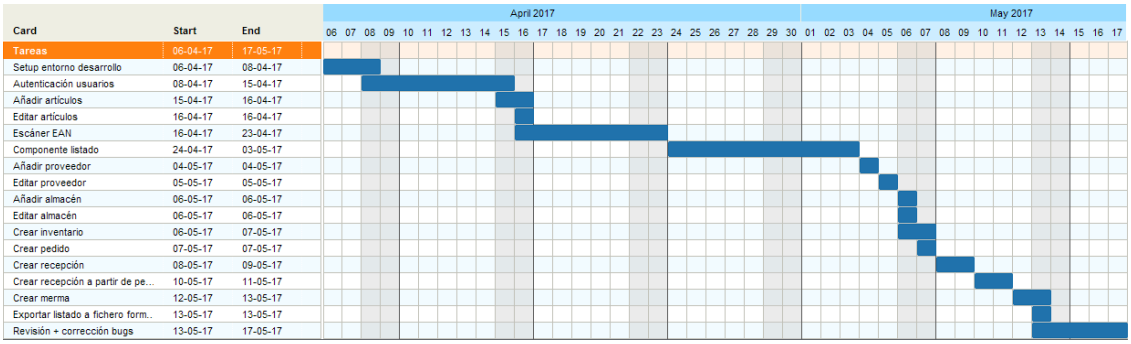


Figura 2

1.5 Breve sumario de productos obtenidos

- Memoria
- Video presentación del Trabajo
- Código fuente del proyecto
- Manuales de instalación y compilación (Anexo)

1.6 Breve descripción de los otros capítulos de la memoria

- Usuarios y contexto de uso: Descripción de actores y sus necesidades dentro de su entorno.
- Diseño conceptual: Creación de los escenarios de uso a partir de los datos obtenidos en el capítulo anterior.
- Prototipado: Prototipo horizontal de la aplicación.
- Definición de los casos de uso: Listado de casos de uso obtenidos.
- Diseño de la arquitectura: Diagramas de los diferentes sistemas del proyecto.
- Consideraciones y decisiones de arquitectura y diseño.
- Revisión de la planificación.
- Revisión del diseño.
- Trabajo futuro: Posibles aspectos a modificar en la aplicación en el futuro.
- Bibliografía
- Anexos:

2. Usuarios y contexto de uso

Dada la naturaleza de la aplicación, solamente existen dos tipos de usuarios, en las tablas a continuación se desglosan en su perfil, sus posibles condiciones de trabajo y sus posibles necesidades a la hora de realizar sus tareas:

Tipo	Encargado de los artículos en almacén/es
Perfil	- Alto conocimiento de su entorno de trabajo - Contacto directo con los proveedores
Condiciones	- Puede no tener acceso a internet en su puesto
Necesidades	- Conocimiento de cantidades restantes de un artículo en almacén - Gestión de inventarios - Gestión de pedidos - Gestión de recepciones (entradas de artículos) - Gestión de mermas - Gestión de almacenes

Tipo	Encargado de cuentas
Perfil	- Interés por los gastos - Interés por los ingresos - Rutina trabajando con números
Condiciones	- No tiene por qué usar dispositivo móvil para su trabajo
Necesidades	- Números en cuanto a compra/venta de artículos

3. Diseño conceptual

Se da por entendido que el usuario está logado en todo momento para poder utilizar la aplicación en su dispositivo móvil.

Usuario	Encargado de artículos de almacén/tienda
Contexto	Hay que hacer un inventario de un almacén
Necesidad	Requiere de un recuento de todos los artículos que hay en el almacén

Usuario	Encargado de artículos de almacén/tienda
Contexto	Un producto ha caducado
Necesidad	Registrar una merma

Usuario	Encargado de artículos de almacén/tienda
Contexto	Un artículo se está agotando
Necesidad	Realizar un pedido Dividir pedidos por proveedor

Usuario	Encargado de artículos de almacén/tienda
Contexto	Llega un pedido

Necesidad	Registrar la entrada de ese pedido

Usuario	Encargado de cuentas
Contexto	El almacén/tienda está abierto durante un evento que solamente dura unas horas
Necesidad	Comprobar que la caja cuadra

Usuario	Encargado de cuentas
Contexto	Va a hacer un pedido
Necesidad	Poder ver el número de teléfono del proveedor

Usuario	Encargado de artículos
Contexto	Realiza un inventario en otro almacén
Necesidad	Poder gestionar los almacenes por separado

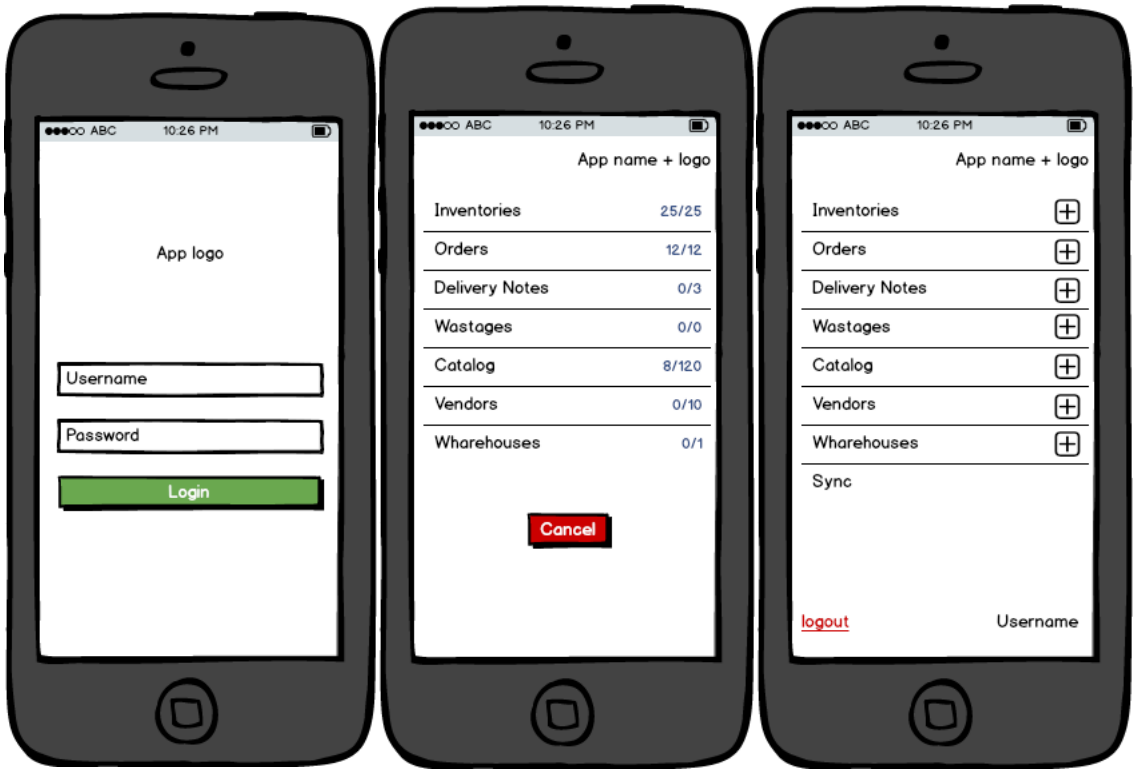
Usuario	Encargado de artículos
Contexto	Otro departamento necesita de una hoja de cálculo con los datos.
Necesidad	Poder exportar a un fichero que se pueda abrir con un programa de gestión de hojas de cálculo

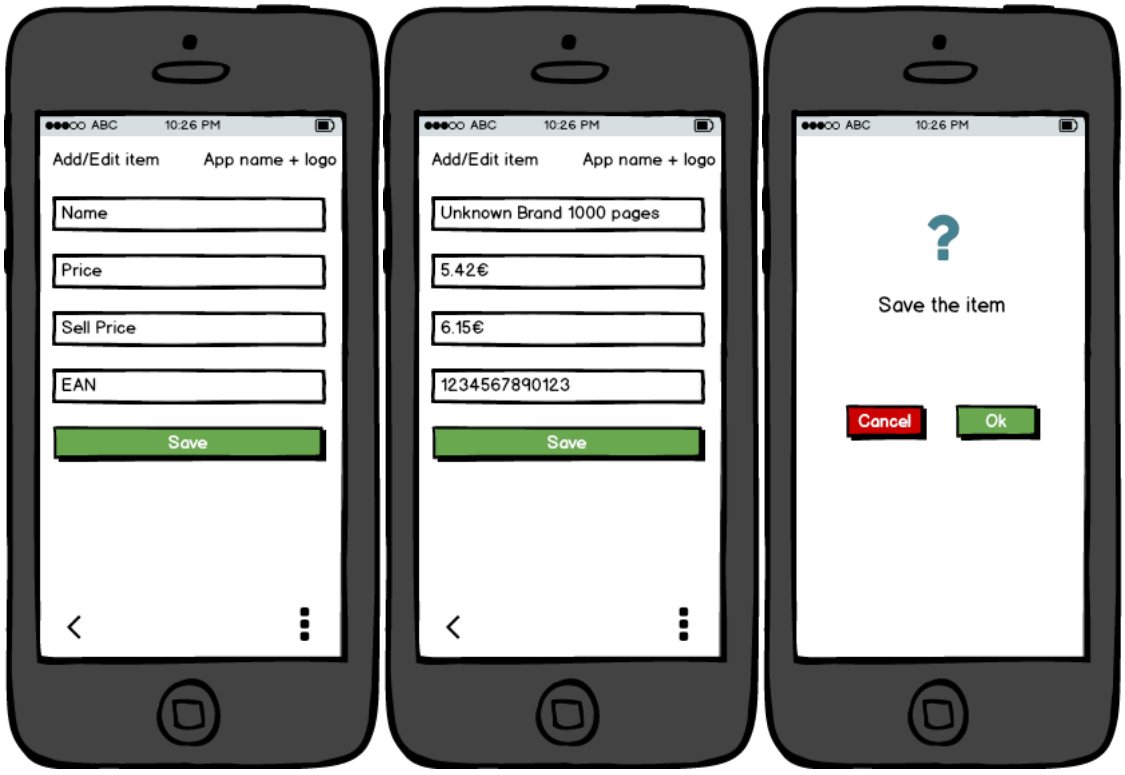
Usuario	Encargado de artículos y/o encargado de cuentas
Contexto	Un compañero ha realizado una acción con su dispositivo
Necesidad	Poder actualizar y ver los cambios realizados

Usuario	Encargado de artículos
Contexto	Procede a registrar un producto
Necesidad	Poder escanear el código de barras Obtener todos los datos posibles a través de la API online

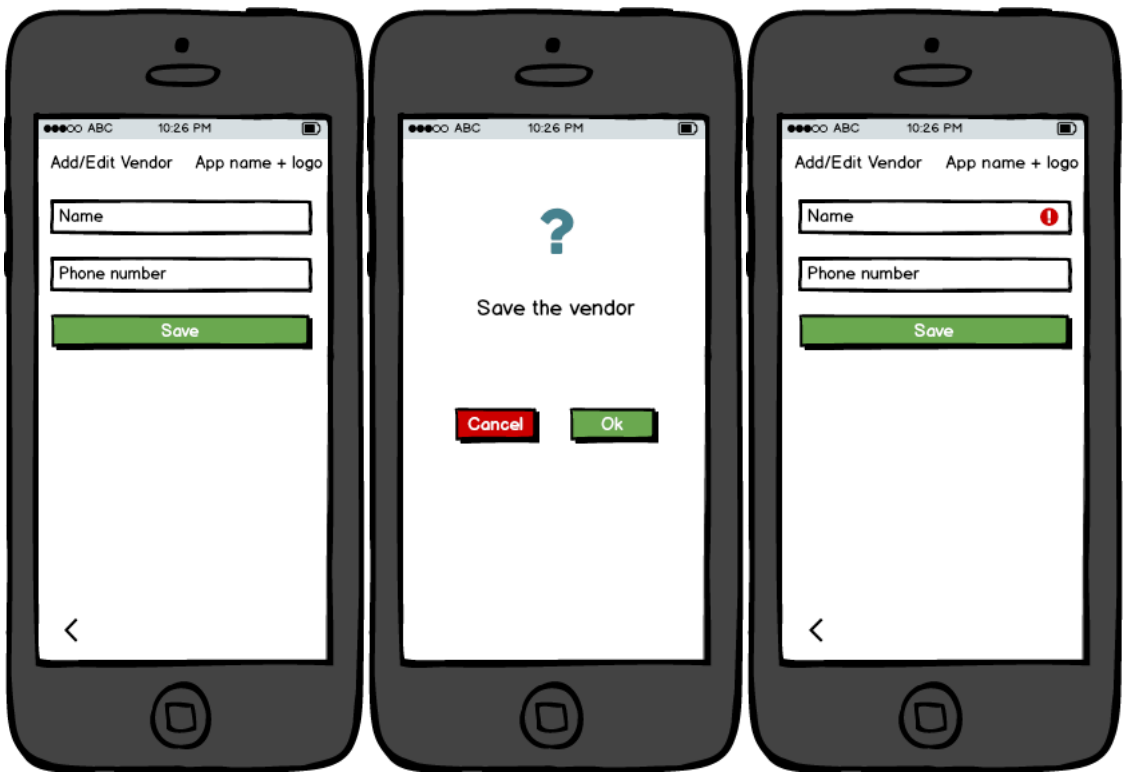
Usuario	Encargado de artículos y/o encargado de cuentas
Contexto	Un compañero ha realizado una acción con su dispositivo
Necesidad	Poder actualizar y ver los cambios realizados

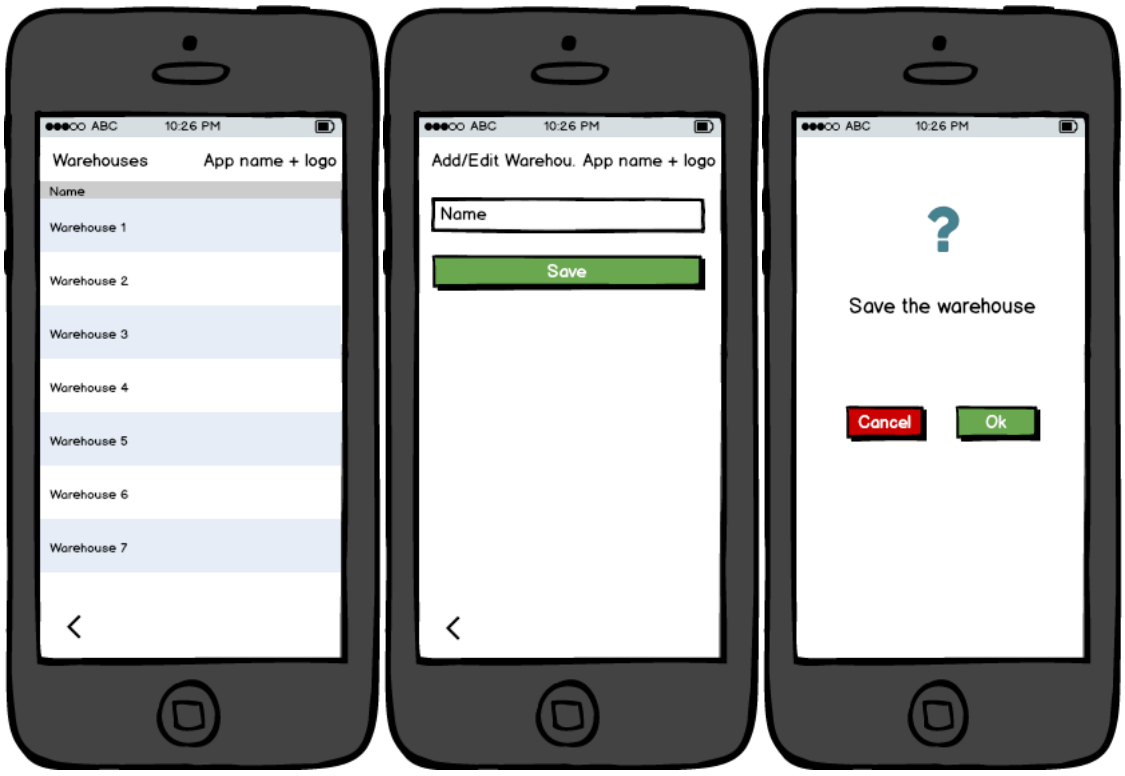
4. Prototipado



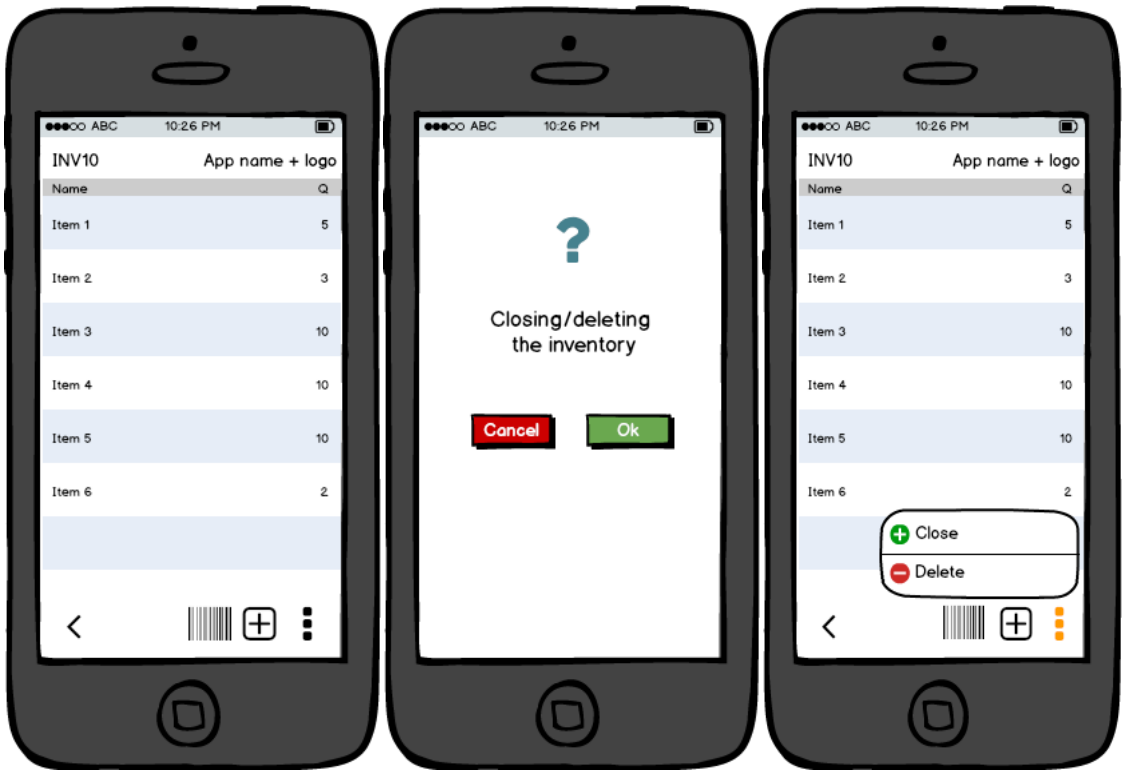

ABC 10:26 PM
Add/Edit item
App name + logo
Name
Price
Sell Price
EAN
Save
ABC 10:26 PM
Add/Edit item
App name + logo
Unknown Brand 1000 pages
5.42€
6.15€
1234567890123
Save
ABC 10:26 PM
?
Save the item
Cancel
Ok

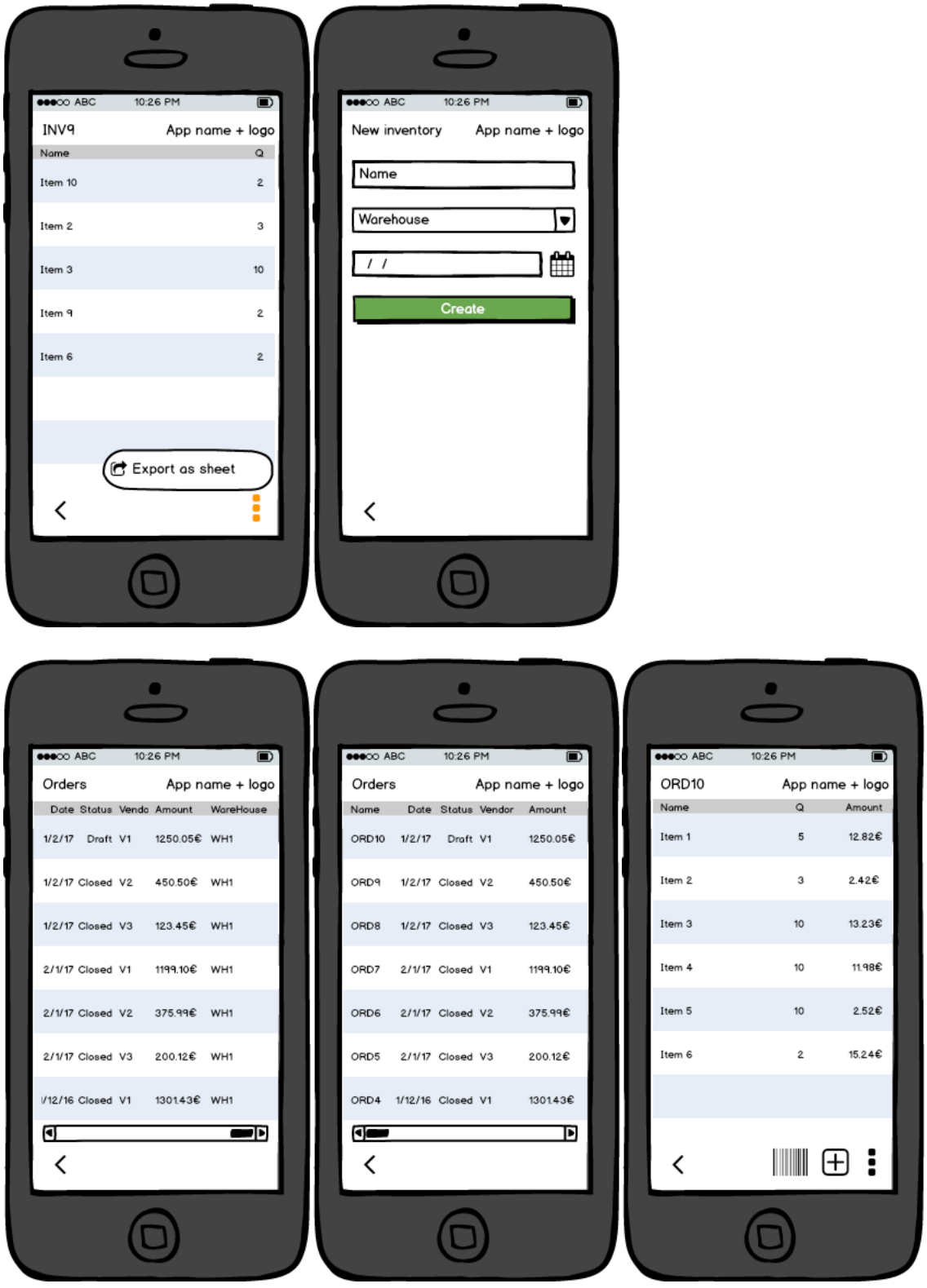

ABC 10:26 PM
Add/Edit item
App name + logo
Name
5.42€
6.15€
1234567890123
Save
Scan item
Fetch data online
ABC 10:26 PM
Vendors
App name + logo
Name
Phone Nr
Vendor 1
XXX-XXX-XXX
Vendor 2
XXX-XXX-XXX
Vendor 3
XXX-XXX-XXX
Vendor 4
XXX-XXX-XXX
Vendor 5
XXX-XXX-XXX
Vendor 6
XXX-XXX-XXX
Vendor 7
XXX-XXX-XXX

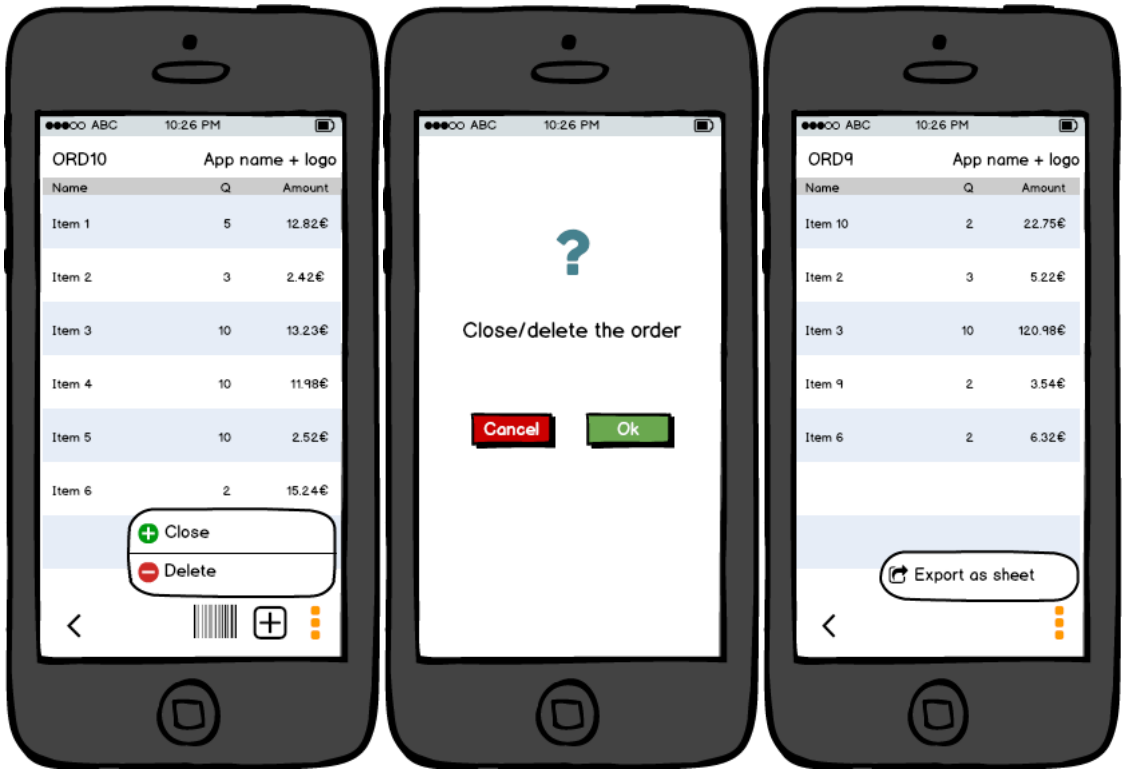

ABC 10:26 PM
Add/Edit Vendor App name + logo
Name
Phone number
Save
ABC 10:26 PM
Save the vendor
Cancel
Ok
ABC 10:26 PM
Add/Edit Vendor App name + logo
Name
Phone number
Save

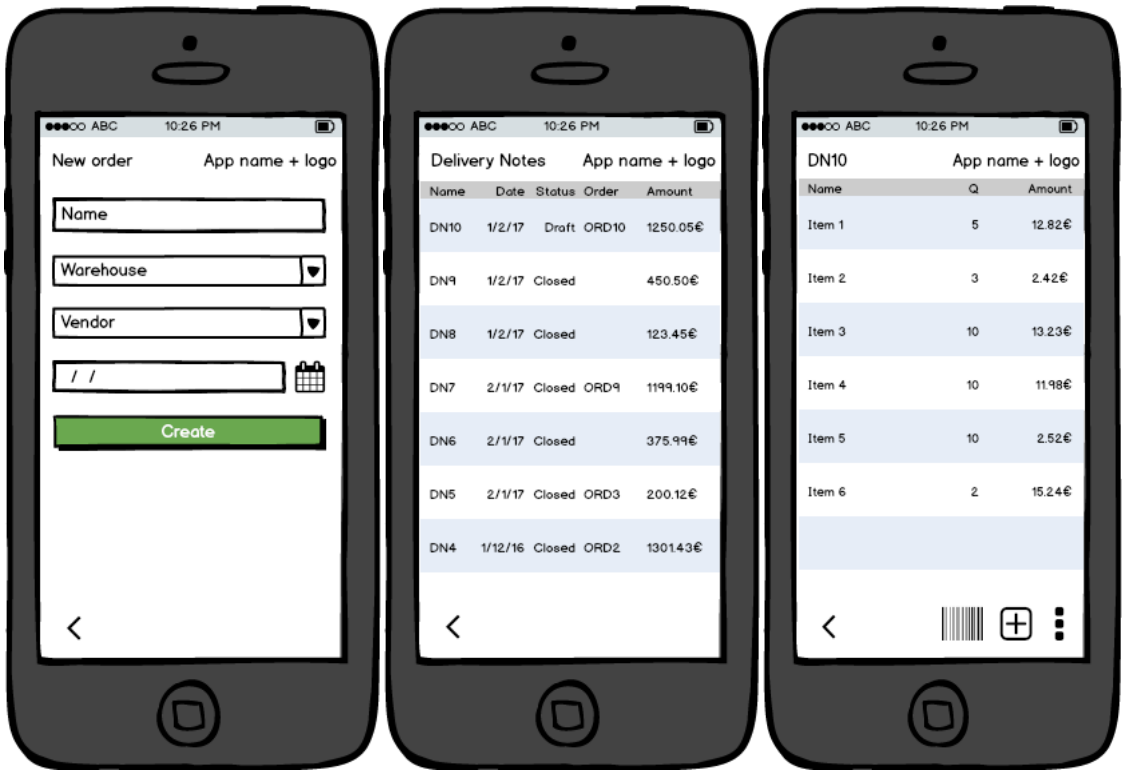

ABC 10:26 PM
Warehouses App name + logo
Name
Warehouse 1
Warehouse 2
Warehouse 3
Warehouse 4
Warehouse 5
Warehouse 6
Warehouse 7
ABC 10:26 PM
Add/Edit Warehou. App name + logo
Name
Save
ABC 10:26 PM
Save the warehouse
Cancel
Ok

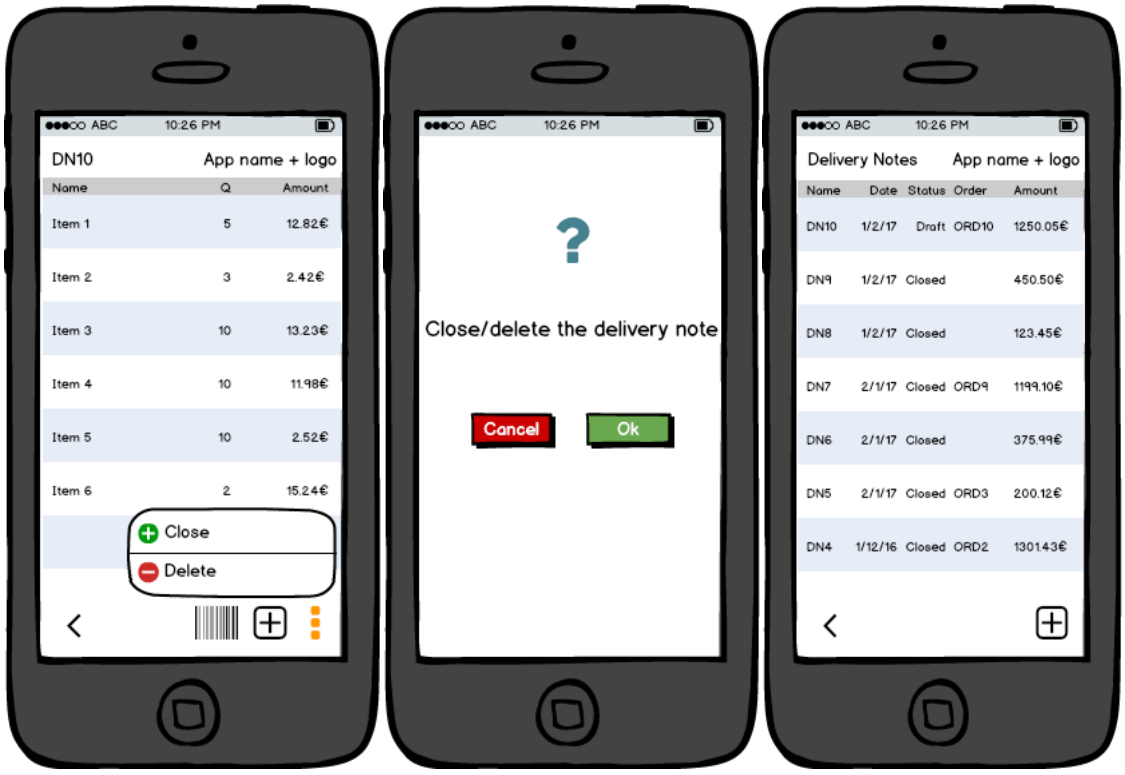

ABC 10:26 PM
Add/Edit Warehou. App name + logo
Name
Save
ABC 10:26 PM
Inventories App name + logo
Name Date Status Warehouse
INV10 1/4/17 Draft warehouse1
INV9 1/3/17 Closed warehouse1
INV8 1/2/17 Closed warehouse1
INV7 2/1/17 Closed warehouse1
INV6 1/12/16 Closed warehouse1
INV5 1/11/16 Closed warehouse1
INV4 1/10/16 Closed warehouse1

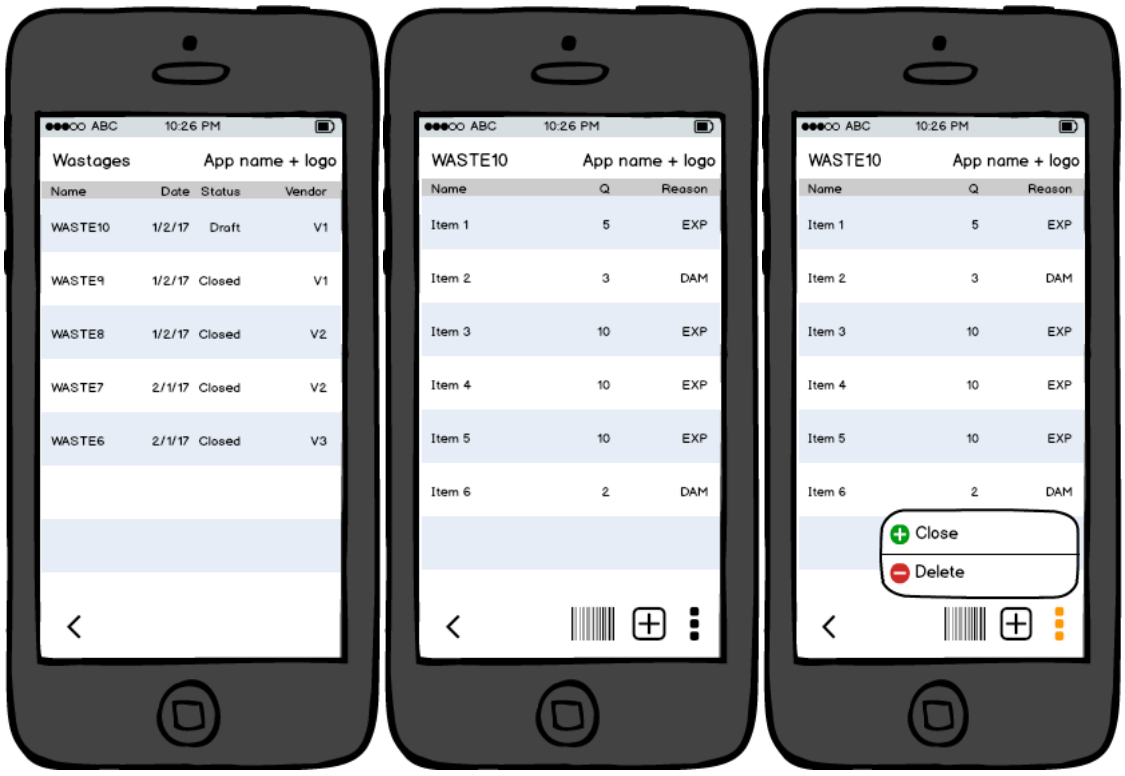

ABC 10:26 PM
INV10 App name + logo
Name Q
Item 1 5
Item 2 3
Item 3 10
Item 4 10
Item 5 10
Item 6 2
ABC 10:26 PM
?
Closing/deleting the inventory
Cancel
Ok
ABC 10:26 PM
INV10 App name + logo
Name Q
Item 1 5
Item 2 3
Item 3 10
Item 4 10
Item 5 10
Item 6 2
Close
Delete

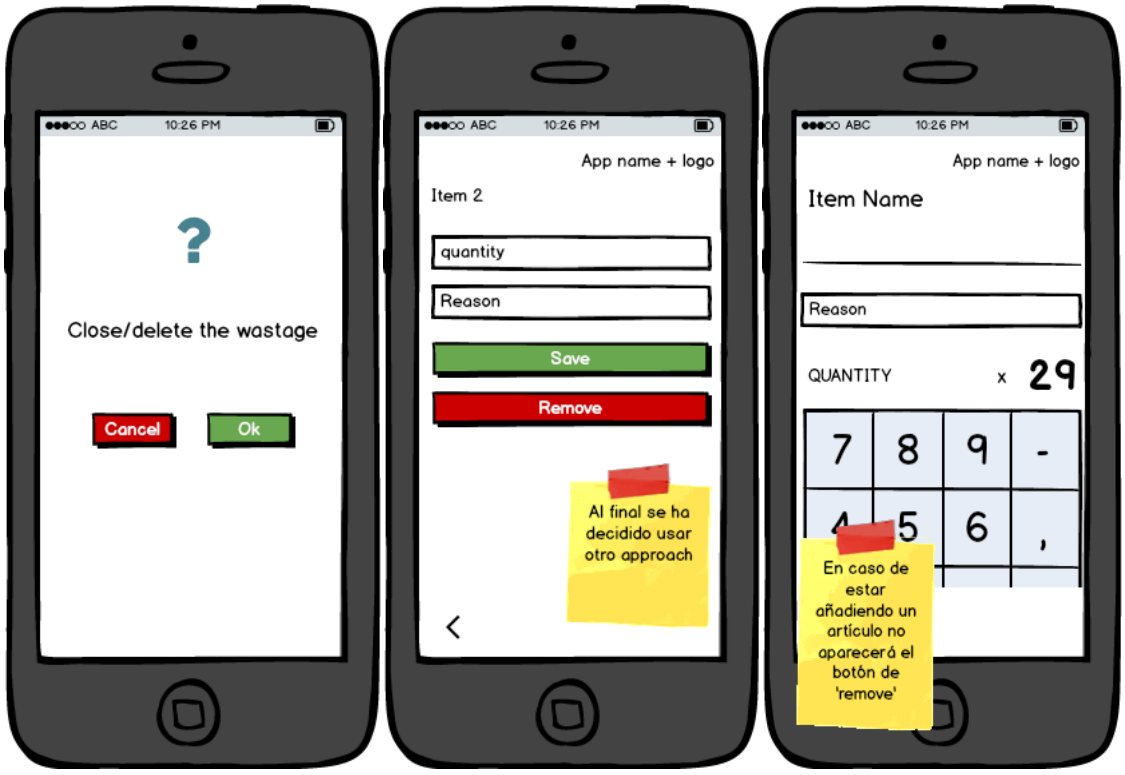

ABC 10:26 PM
INV9
App name + logo
Name Q
Item 10 2
Item 2 3
Item 3 10
Item 9 2
Item 6 2
Export as sheet
ABC 10:26 PM
New inventory
App name + logo
Name
Warehouse
/ /
Create
ABC 10:26 PM
Orders
App name + logo
Date Status Vendo Amount WareHouse
1/2/17 Draft V1 1250.05€ WH1
1/2/17 Closed V2 450.50€ WH1
1/2/17 Closed V3 123.45€ WH1
2/1/17 Closed V1 1199.10€ WH1
2/1/17 Closed V2 375.99€ WH1
2/1/17 Closed V3 200.12€ WH1
1/12/16 Closed V1 1301.43€ WH1
ABC 10:26 PM
Orders
App name + logo
Name Date Status Vendor Amount
ORD10 1/2/17 Draft V1 1250.05€
ORD9 1/2/17 Closed V2 450.50€
ORD8 1/2/17 Closed V3 123.45€
ORD7 2/1/17 Closed V1 1199.10€
ORD6 2/1/17 Closed V2 375.99€
ORD5 2/1/17 Closed V3 200.12€
ORD4 1/12/16 Closed V1 1301.43€
ABC 10:26 PM
ORD10
App name + logo
Name Q Amount
Item 1 5 12.82€
Item 2 3 2.42€
Item 3 10 13.23€
Item 4 10 11.98€
Item 5 10 2.52€
Item 6 2 15.24€


ABC 10:26 PM
ORD10
App name + logo
Name Q Amount
Item 1 5 12.82€
Item 2 3 2.42€
Item 3 10 13.23€
Item 4 10 11.98€
Item 5 10 2.52€
Item 6 2 15.24€
Close
Delete
ABC 10:26 PM
?
Close/delete the order
Cancel
Ok
ABC 10:26 PM
ORD9
App name + logo
Name Q Amount
Item 10 2 22.75€
Item 2 3 5.22€
Item 3 10 120.98€
Item 9 2 3.54€
Item 6 2 6.32€
Export as sheet


ABC 10:26 PM
New order
App name + logo
Name
Warehouse
Vendor
/ /
Create
ABC 10:26 PM
Delivery Notes
App name + logo
Name Date Status Order Amount
DN10 1/2/17 Draft ORD10 1250.05€
DN9 1/2/17 Closed 450.50€
DN8 1/2/17 Closed 123.45€
DN7 2/1/17 Closed ORD9 1199.10€
DN6 2/1/17 Closed 375.99€
DN5 2/1/17 Closed ORD3 200.12€
DN4 1/12/16 Closed ORD2 1301.43€
ABC 10:26 PM
DN10
App name + logo
Name Q Amount
Item 1 5 12.82€
Item 2 3 2.42€
Item 3 10 13.23€
Item 4 10 11.98€
Item 5 10 2.52€
Item 6 2 15.24€


ABC 10:26 PM
DN10
App name + logo
Name Q Amount
Item 1 5 12.82€
Item 2 3 2.42€
Item 3 10 13.23€
Item 4 10 11.98€
Item 5 10 2.52€
Item 6 2 15.24€
Close
Delete
ABC 10:26 PM
Close/delete the delivery note
Cancel
Ok
ABC 10:26 PM
Delivery Notes
App name + logo
Name Date Status Order Amount
DN10 1/2/17 Draft ORD10 1250.05€
DN9 1/2/17 Closed 450.50€
DN8 1/2/17 Closed 123.45€
DN7 2/1/17 Closed ORD9 1199.10€
DN6 2/1/17 Closed 375.99€
DN5 2/1/17 Closed ORD3 200.12€
DN4 1/12/16 Closed ORD2 1301.43€


ABC 10:26 PM
Wastages
App name + logo
Name Date Status Vendor
WASTE10 1/2/17 Draft V1
WASTE9 1/2/17 Closed V1
WASTE8 1/2/17 Closed V2
WASTE7 2/1/17 Closed V2
WASTE6 2/1/17 Closed V3
ABC 10:26 PM
WASTE10
App name + logo
Name Q Reason
Item 1 5 EXP
Item 2 3 DAM
Item 3 10 EXP
Item 4 10 EXP
Item 5 10 EXP
Item 6 2 DAM
ABC 10:26 PM
WASTE10
App name + logo
Name Q Reason
Item 1 5 EXP
Item 2 3 DAM
Item 3 10 EXP
Item 4 10 EXP
Item 5 10 EXP
Item 6 2 DAM
Close
Delete


ABC 10:26 PM
Close/delete the wastage
Cancel
Ok
ABC 10:26 PM
App name + logo
Item 2
quantity
Reason
Save
Remove
Al final se ha decidido usar otro approach
ABC 10:26 PM
App name + logo
Item Name
Reason
QUANTITY x 29
7 8 9 -
6 ,
En caso de estar añadiendo un artículo no aparecerá el botón de 'remove'


ABC 10:26 PM
Remove item from list
Cancel
Ok
ABC 10:26 PM
App name + logo
Item Name
QUANTITY x 29
7 8 9 -
4 5 6 ,
1 2 3 0
Save
Remove


ABC 10:26 PM
ABC 10:26 PM
ABC 10:26 PM
Fetching data online
this might take a couple of seconds
ABC 10:26 PM
No data found for this item
Ok

Figura 3

5. Definición de los casos de uso:

Acción	Logarse
Actores	Usuario de la aplicación
Precondiciones	No estar logado
Flujo	Abrir la aplicación Rellenar usuario/contraseña Enviar el formulario
Postcondiciones	El usuario estará logado (en caso de ser correctos los datos) Si no hay base de datos creada, la aplicación tratará de sincronizarse Si hay base de datos, abrirá el menú inicial

Acción	Sincronizar
Actores	Usuario de la aplicación
Precondiciones	Estar logado
Flujo	Pulsar sobre el módulo de sincronización
Postcondiciones	La aplicación empezará a sincronizarse, subiendo al servidor todo lo que no haya podido subir al cerrarse un inventario, pedido, merma, recepción o al añadirse un almacén, proveedor o artículo al catálogo y después descargando todos los datos de las diferentes tablas.

Acción	Deslogarse

Actores	Usuario de la aplicación
Precondiciones	Estar logado
Flujo	Ir al menú inicial Pulsar el link de deslogarse
Postcondiciones	Será redirigido a la pantalla de login

Acción	Cancelar sincronización
Actores	Usuario de la aplicación
Precondiciones	Estar logado
Flujo	Pulsar cancelar una vez haya comenzado la sincronización
Postcondiciones	En caso de no haber habido base de datos, el usuario será deslogado En caso de haber habido base de datos, todo quedará como estaba

Acción	Ver artículos añadidos al catálogo
Actores	Usuario de la aplicación
Precondiciones	Estar logado
Flujo	Pulsar sobre el módulo de catálogo
Postcondiciones	Aparecerá un listado con todos los artículos añadidos al catálogo

Acción	Añadir artículos manualmente al catálogo
Actores	Usuario de la aplicación
Precondiciones	Estar logado Ir al menú principal

Flujo	Pulsar sobre el botón + a la derecha del módulo de catálogo Rellenar formulario
Postcondiciones	Se habrá añadido un artículo al catálogo

Acción	Añadir código de barras en el formulario de inserción manual de artículos del catálogo
Actores	Usuario de la aplicación
Precondiciones	Estar logado Ir al listado de artículos de catálogo Pulsar el botón +
Flujo	Pulsar en el icono de menú secundario Pulsar sobre el icono de escáner (se mostrará la cámara) Escanear código de barras
Postcondiciones	Se auto-rellenará el campo de código de barras

Acción	Auto-rellenar datos de un artículo
Actores	Usuario de la aplicación
Precondiciones	Estar logado Añadir un código de barras en el formulario de inserción manual de artículo del catálogo
Flujo	Pulsar sobre el botón de menú secundario Pulsar sobre el botón de relleno online Esperar a que cargue los datos

Postcondiciones	En caso de haberse conseguido datos correspondientes al código de barras insertado, se autorellenará el nombre En caso contrario, se mostrará un mensaje de error y posteriormente no se autorellenará nada

Acción	Ver listado de proveedor
Actores	Usuario de la aplicación
Precondiciones	Estar logado Ir al menú inicial
Flujo	Pulsar sobre el módulo de proveedores
Postcondiciones	Aparecerá una vista con todos los proveedores

Acción	Añadir proveedor
Actores	Usuario de la aplicación
Precondiciones	Estar logado Ir al menú inicial
Flujo	Pulsar sobre el + a la derecha del módulo de proveedores Rellenar formulario
Postcondiciones	Se habrá insertado un proveedor y se habrá subido al servidor

Acción	Añadir almacén
Actores	Usuario de la aplicación
Precondiciones	Estar logado Ir al menú inicial

Flujo	Pulsar sobre el + a la derecha del módulo de almacenes Rellenar formulario
Postcondiciones	Se habrá insertado un almacén y se habrá subido al servidor

Casos de uso en común de los módulos de inventario, pedido, recepción de mercancía o merma (a partir de ahora “documento”)

Acción	Crear un nuevo documento
Actores	Usuario de la aplicación
Precondiciones	Estar logado Haber sincronizado la app
Flujo	Pulsar en el icono + al lado de nombre del documento a crear. Rellenar los campos que se piden en el formulario Enviar el formulario
Postcondiciones	Se habrá creado un documento en estado borrador

Acción	Ver listado de documentos creados
Actores	Usuario de la aplicación
Precondiciones	Estar logado Haber sincronizado la app
Flujo	Pulsar en el botón correspondiendo al tipo de documento que se quiera ver
Postcondiciones	Aparecerán todos los documentos creados

Acción	Ver los artículos (líneas) del documento
Actores	Usuario de la aplicación
Precondiciones	Estar logado Haber sincronizado la app
Flujo	Entrar en el listado de documentos creados Pulsar sobre uno de ellos
Postcondiciones	Aparecerán todas las líneas del documento

Acción	Cerrar un borrador
Actores	Usuario de la aplicación
Precondiciones	Estar logado Haber sincronizado la app Haber creado un documento en estado borrador
Flujo	Ver las líneas de un documento en estado borrador Pulsar sobre el icono de menú secundario Pulsar sobre finalizar Aceptar la modal
Postcondiciones	Se subirá al servidor el documento creado Se cambiará el estado del documento, pasando a estar cerrado

Acción	Borrar un documento en estado borrador
Actores	Usuario de la aplicación
Precondiciones	Estar logado Haber sincronizado la app

	Haber creado un documento en estado borrador
Flujo	Ver las líneas de un documento en estado borrador Pulsar sobre el icono de menú secundario Pulsar sobre borrar Aceptar la modal
Postcondiciones	Se borrará el documento en local

Acción	Añadir artículo manualmente a un documento en estado borrador
Actores	Usuario de la aplicación
Precondiciones	Estar logado Haber sincronizado la app Haber creado un documento en estado borrador
Flujo	Ver las líneas de un documento en estado borrador Pulsar en el + del menú (aparecerá el listado de catálogo de artículos) Elegir uno (aparecerá la vista de edición de cantidad) Elegir cantidad Pulsar en guardar
Postcondiciones	Se habrá añadido un artículo como línea de un documento en estado borrador

Acción	Editar una línea de un documento en estado borrador
Actores	Usuario de la aplicación
Precondiciones	Estar logado Haber sincronizado la app

	Haber creado un documento en estado borrador Haber añadido una línea
Flujo	Ver las líneas de un documento en estado borrador Pulsar en una línea Cambiar la cantidad o la razón (en caso de mermas) Darle a guardar
Postcondiciones	Se habrá editado la/s propiedad/es de la línea seleccionada de un documento en estado borrador

Acción	Borrar una línea de un documento en estado borrador
Actores	Usuario de la aplicación
Precondiciones	Estar logado Haber sincronizado la app Haber creado un documento en estado borrador Haber añadido una línea
Flujo	Ver las líneas de un documento en estado borrador Pulsar en una línea Darle a borrar
Postcondiciones	Se habrá borrado la línea seleccionada de un documento en estado borrador

Acción	Editar una línea de un documento en estado borrador
Actores	Usuario de la aplicación
Precondiciones	Estar logado

	Haber sincronizado la app Haber creado un documento en estado borrador Haber añadido una línea
Flujo	Ver las líneas de un documento en estado borrador Pulsar en una línea Cambiar la cantidad o la razón (en caso de mermas) Darle a guardar
Postcondiciones	Se habrá editado la/s propiedad/es de la línea seleccionada de un documento en estado borrador

Acción	Añadir artículo mediante escáner a un documento en estado borrador
Actores	Usuario de la aplicación
Precondiciones	Estar logado Haber sincronizado la app Haber creado un documento en estado borrador
Flujo	Ver las líneas de un documento en estado borrador Pulsar en el icono de escáner (aparecerá la cámara) Escanear un código de barras (aparecerá la vista de edición de cantidad) Elegir cantidad Pulsar en guardar
Postcondiciones	Se habrá añadido un artículo como línea de un documento en estado borrador

Acción	Exportar líneas de un documento cerrado
Actores	Usuario de la aplicación
Precondiciones	Estar logado Haber sincronizado la app Tener documentos en estado cerrado
Flujo	Ir a las líneas de un documento cerrado Pulsar sobre el icono de menú secundario Pulsar sobre ‘exportar’
Postcondiciones	Se creará un fichero .csv en el dispositivo

6. Diseño de la arquitectura

6.1. Base de datos

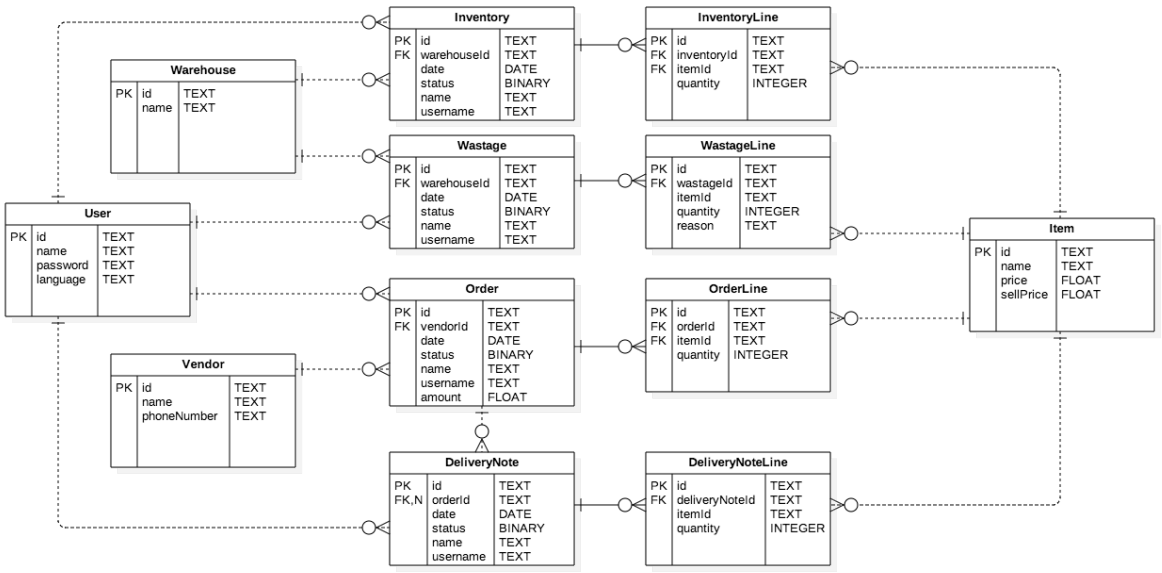


Figura 4

6.2. API (Servidor)

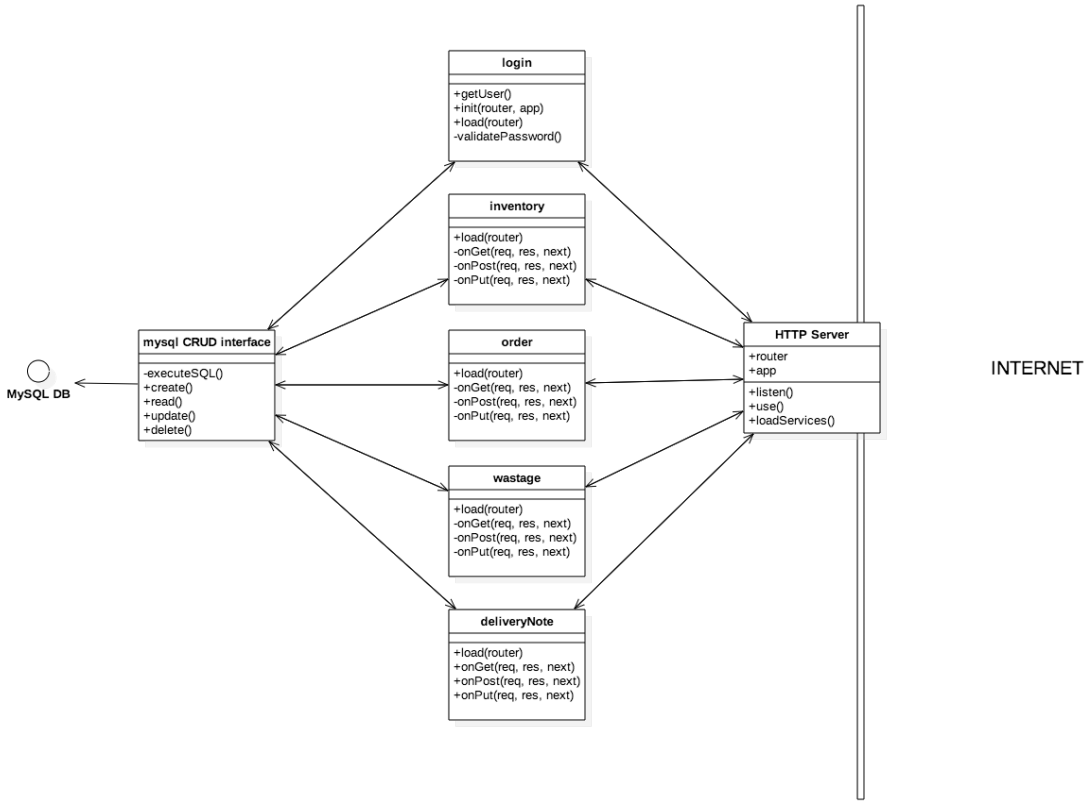


Figura 5

6.3. Cliente

Las entidades y clase se dividen en tres estructuras claramente definidas:

6.3.1. Sincronización de datos:

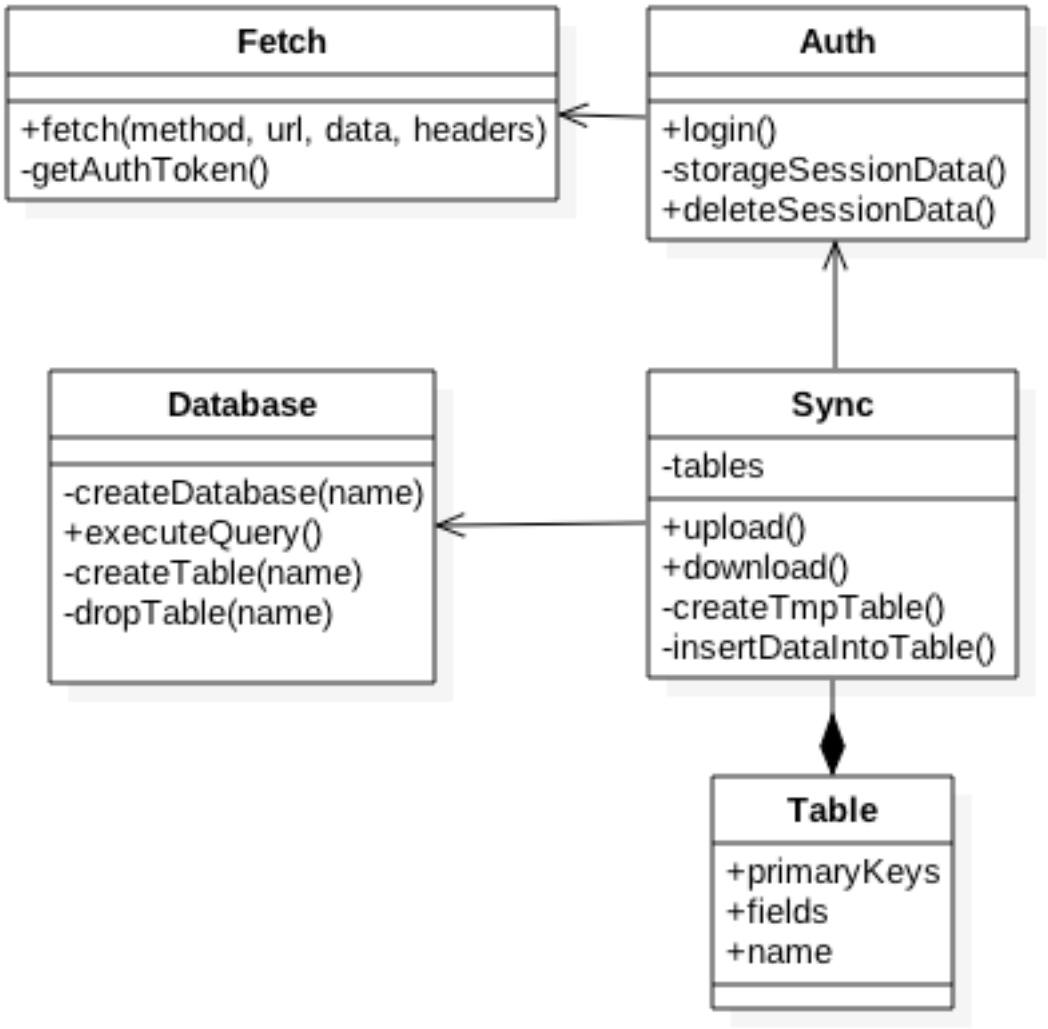


Figura 6: Arquitectura sincronización de datos.

6.3.2. Visualización de datos e interacción:

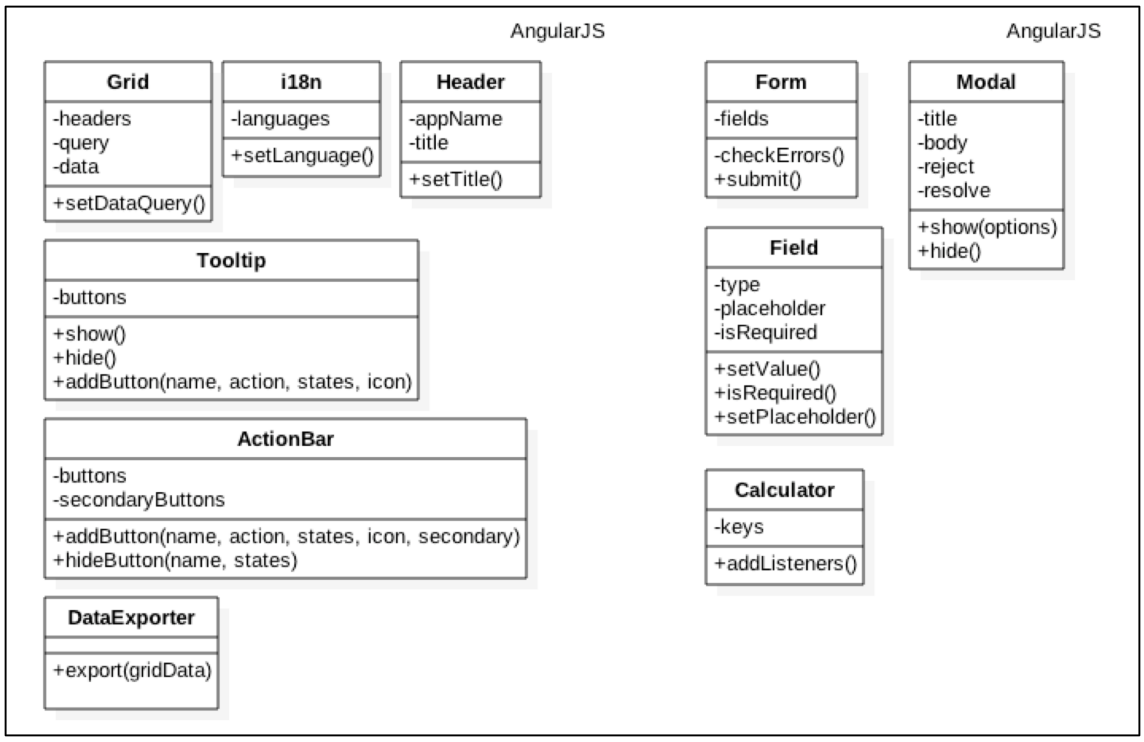


Figura 7: Estructura de clases en cliente.

6.3.1. Estructura

Cliente:

Este Trabajo de final de Master utiliza como base el framework *AngularJS* y su sistema de routing para mostrar las diferentes vistas de la aplicación. Además, se integra la librería *Ionic* para una mayor flexibilidad entre diferentes dispositivos en las vistas. Se diferenciará cada "componente" de la aplicación, creando una carpeta por cada una de las vistas. En estas carpetas estarán los controladores, plantillas y librerías necesarias únicamente para una vista. Además habrá una carpeta que incluirá todo el sistema de sincronización y base de datos del sistema. Se utilizará *SQLite* para la persistencia de

datos en el sistema. Gracias a *browserify* se podrán utilizar módulos npm en el proyecto, al ser estos concatenados y minificados posteriormente con tareas creadas con *gulp*. También se incluirán librerías para una facilitar el desarrollo, como *lodash o squel*. La sesión se almacenará en LocalStorage.

Servidor:

Creado con *NodeJS*, utilizará el framework *express* para gestionar todas las llamadas HTTP. Al iniciar el servidor, se cargarán los diferentes servicios, los cuales se encargarán de gestionar exclusivamente su interfaz. Se utilizará la librería *passport* para la comprobación de credenciales. En caso de ser correctos, se generará un JWT(JSON Web Token) que debería ser enviado por el cliente en la cabecera para poder identificar a usuarios válidos. La base de datos elegida ha sido MySQL. Por último, dispondrá de un script para poder generar contraseñas.

7. Consideraciones y decisiones de arquitectura y diseño:

1. Al no tener por qué ser usuarios con un trasfondo tecnológico, se ha intentado mantener un diseño muy sencillo de entender y seguir, por ello se intenta crear una experiencia homogénea entre los diferentes módulos/funcionalidades.

2. Se creará un almacén por defecto para hacer más fácil la primera interacción con la aplicación.

3. Podría haber vistas con múltiples columnas, de manera que no lleguen a visualizarse bien los datos en la pantalla de un dispositivo móvil, por ello se hará que se pueda hacer tanto scroll horizontal como vertical.

4. Al poder no haber cobertura allí dónde esté el almacén, es importante que los datos se puedan almacenar en local hasta que se sincronice con el servidor.

5. Habrá tablas con las mismas columnas, pero al ser funcionalidades diferentes y con el fin de que la estructura de la base de datos siga siendo lo más escalable posible, se decide repetir schemas para estos casos. Se ha de dejar al usuario poner nombres a sus inventarios, pedidos, recepciones y mermas para un reconocimiento más rápido de sus datos.

6. Hay tablas en la base de datos cuya clave primaria se ha creado deliberadamente a partir de una id nueva y no a partir de una combinación de columnas (por ejemplo itemId + vendorId) ya que queremos que el usuario pueda insertar más de una vez el mismo artículo en un listado ya que podría contarlo en varias tandas, dependiendo de cómo esté repartido el almacén.

7. El idioma a mostrar en la app se definirá por el campo ‘language en la tabla User de la base de datos.

8. Al ser una aplicación más general y dirigida a pequeñas y medianas empresas en las que estos dos tipos de usuarios sean la misma persona, se decide que no habrá cuentas diferentes para roles diferentes en la empresa, es decir, el encargado de almacén tendrá el acceso a los mismos datos que el encargado de cuentas de la empresa.

9. Este Trabajo Final de Master no creará una aplicación servidor que gestione todos los artículos, inventarios, almacenes, etc... de todas las empresas que usen la aplicación. Cada empresa deberá configurar su servidor y crear sus usuarios. Por ello, tampoco se implementará un sistema de alta de usuarios.

8. Revisión de la planificación

Por desgracia, no se ha podido seguir la planificación tal y como se había planteado en un principio ya que la primera tarea (Setup entorno de desarrollo) tardó muchísimo más de lo esperado. Fue muy complicado lograr un entorno que compilase y generase un código ejecutable por un dispositivo móvil. Además, a medida que se iban implementando nuevas funcionalidades, iban surgiendo pequeños problemas que fueron alargando todo el proceso, llegando a no tener margen para la mejora o resolución de bugs.

A pesar de las incidencias, se logró terminar todas las tareas técnicamente posibles ya que se decidió terminar los módulos de Proveedor, Almacén y Catálogo como primeros, por ser de los que dependen directamente todos los demás. Técnicamente no posibles fueron dos puntos; no se pudo añadir el campo “cantidad” al listado de pedidos ya que sqlite no permite columnas calculadas, al igual que no permite un “right join” de dos tablas,

teniendo que eliminar la posibilidad de que una recepción no sea correspondiente a un pedido concreto. Siendo el coste de la resolución de estos problemas, se decidió prescindir de ellos pero retomarlos al final, pero no ha dado tiempo.

9. Revisión del diseño

Se han seguido las pautas de los estilos marcados, pero ha habido cambios y retoques finales con la intención de ayudar a la interacción con el usuario. Cabe por destacar, la implementación de pantallas modales en vez de vistas completas para la confirmación o cancelación de una acción o la petición de una decisión para una acción. También se ha optado por cambiar algunos botones que en principio eran secundarios y se han añadido como primarios al ser los únicos disponibles y así darle al usuario más claridad en cuanto a las acciones de las que dispone en cada momento.

Un rediseño completo de la vista solamente tuvo lugar en la vista de sincronización de documentos, que pasó de ser un listado de las diferentes entidades para acabar siendo constando solamente de una imagen “spinner” y un botón de cancelar. El problema residía en que al haber pocos datos, la carga se realizaba casi instantáneamente, haciendo que al usuario no le diera tiempo de visualizar y entender lo que estaba ocurriendo.

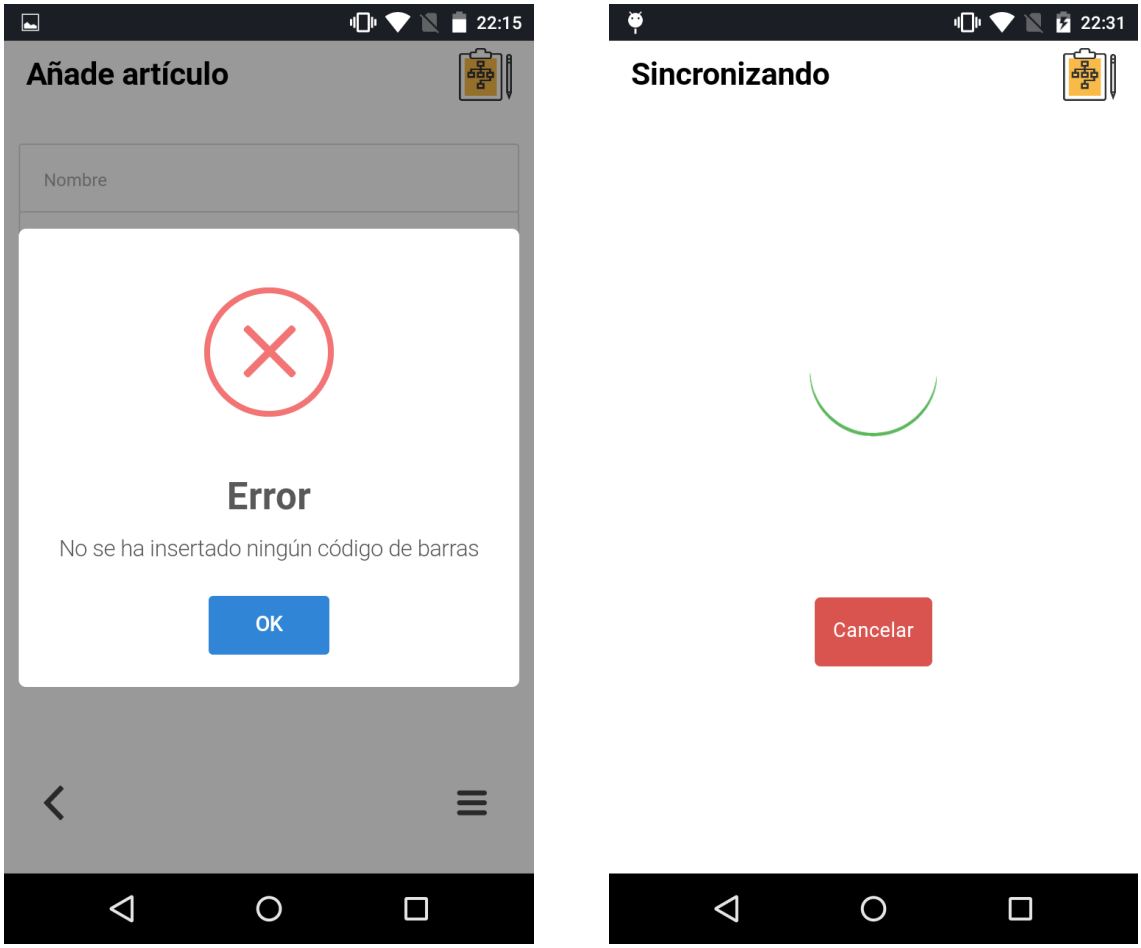


Figura 8: Cambios diseño final.

10. Trabajo futuro

Los siguientes pasos a implementar, dependiendo desde el punto de vista, pueden ser, "técnicos" o "de producto":

- Técnicos:
  - Refactor de código, se puede derivar el trabajo de la mayoría de los controladores a clases más pequeñas.
  - Tests unitarios.
  - Sincronización de entidades con "web workers" para cuando haya muchos datos.

- De producto:
    - Añadir columna cantidad al listado de pedidos.
    - Añadir la posibilidad de crear una recepción sin un pedido asociado.
    - Retoque de los estilos.

11. Bibliografía

1. Node.js and MySQL tutorial | Codeforgeek [Internet]. [cited 2017 April]. [Source code] Available from: https://codeforgeek.com/2015/01/nodejs-mysql-tutorial/

2. JSON Web Tokens - jwt.io [Internet]. [cited 2017 May 24]. [Source code] Available from: https://jwt.io/#debugger

3. Use JavaScript to Export Your Data as CSV | appendTo [Internet]. [cited 2017 April]. [Source code] Available from: https://appendto.com/2017/04/use-javascript-to-export-your-data-as-csv/

4. Plunker [Internet]. [cited 2017 April]. [Source code] Available from: http://plnkr.co/edit/BXQKYm6HzoagbdAFkqbd?p=preview

5. Ionicons: The premium icon font for Ionic Framework [Internet]. [cited 2017 May 24]. [Source code] Available from: http://ionicons.com/

6. Use SQLite - Mozilla | MDN [Internet]. [cited 2017 April]. [Source code] Available from: https://developer.mozilla.org/en-US/Add-

ons/Thunderbird/HowTos/Common_Thunderbird_Extension_Techniques/Use_SQLite

7. SQLite Documentation [Internet]. [cited 2017 April]. [Source code] Available from: https://www.sqlite.org/docs.html

8. Shotts K. Mastering PhoneGap mobile application development : take your PhoneGap experience to the next level and create engaging real-world applications [Internet]. [cited 2017 April]. [Source code] Available from: https://books.google.es/books?id=ZFhLDAAAQBAJ&pg=PA222&lpg=PA222&dq=websql+float+numbers&source=bl&ots=_fgTJGqCY9&sig=N9FbmQ05y0T_ia2CGV9ouuHQWb4&hl=de&sa=X&ved=0ahUKEwjLzYe2urjTAhUGbRQKHaOZDMUQ6AEIUzAF#v=onepage&q=websql float numbers&f=false

9. MySQL :: MySQL 5.7 Reference Manual :: 12.21.2 DECIMAL Data Type Characteristics [Internet]. [cited 2017 April]. [Source code] Available from: https://dev.mysql.com/doc/refman/5.7/en/precision-math-decimal-characteristics.html

10. MySQL :: MySQL 5.7 Reference Manual :: 13.2.5 INSERT Syntax [Internet]. [cited 2017 April]. [Source code] Available from: https://dev.mysql.com/doc/refman/5.7/en/insert.html

11. Full-height app layouts: A CSS trick to make it easier [Internet]. [cited 2017 April]. [Source code] Available from: http://blog.stevensanderson.com/2011/10/05/full-height-app-layouts-a-css-trick-to-make-it-easier/

12. Grid - Ionic API Documentation - Ionic Framework [Internet]. [cited 2017 April]. [Source code] Available from: https://ionicframework.com/docs/api/components/grid/Grid/

13. Icon Buttons - Ionic Components [Internet]. [cited 2017 April]. [Source code] Available from: https://ionicframework.com/docs/v1/components/#icon-buttons

14. angular translate - i18n for your Angular apps, made easy. [Internet]. [cited 2017 April]. [Source code] Available from: https://angular-translate.github.io/

15. MySQL :: MySQL 5.7 Reference Manual :: 13.1.18.6 Using FOREIGN KEY Constraints [Internet]. [cited 2017 April]. [Source code] Available from: https://dev.mysql.com/doc/refman/5.7/en/create-table-foreign-keys.html

16. AngularJS: API: select [Internet]. [cited 2017 May]. [Source code] Available from: https://docs.angularjs.org/api/ng/directive/select

17. Open Sans - Google Fonts [Internet]. [cited 2017 May]. [Source code] Available from: https://fonts.google.com/specimen/Open+Sans

18. 83 Flat Line UX And E-Commerce Icons For Free – Smashing Magazine [Internet]. [cited 2017 May]. [Source code] Available from: https://www.smashingmagazine.com/2016/09/freebie-flat-line-ux-and-e-commerce-icon-sets-83-icon-sets-ai-eps-png-svg/

19. loading.io - Your SVG + GIF Ajax Loading Icons [Internet]. [cited 2017 May]. [Source code] Available from: https://loading.io/

20. File - Apache Cordova [Internet]. [cited 2017 May 24]. [Source code] Available from: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-file/

21. Scroll - Ionic API Documentation - Ionic Framework [Internet]. [cited 2017 May 24]. [Source code] Available from: https://ionicframework.com/docs/api/components/scroll/Scroll/

12. Anexos

6.1. Instalación y configuración de servidor MySQL (Linux)

- Antes de poder usar el servidor, hay que tener instalado un servidor mysql. Para ello, desde la terminal, hay que ejecutar los siguientes comandos:
    - sudo apt-get update
    - sudo apt-get install mysql-server
    - sudo mysql_secure_installation
- Una vez instalado, pasamos a conectarnos al servidor MySql para poder crear el esquema de tablas inicial, para ello ejecutamos:
    - mysql –u root –h localhost –p
- En el siguiente paso, creamos una base de datos, para ello es necesario ejecutar el siguiente código desde la línea de comandos sql:
    - CREATE DATABASE [nombre base de datos];
- Una vez creada la base de datos, es necesario crear las tablas requeridas por el servidor de este Trabajo de fin de Máster, para ello:
    - USE [nombre base de datos creada]
    - source ./db/mysql/queries.sql

6.2 Instalación servidor de este Trabajo de fin de Máster (Linux):

- Descomprimir servidor
- Desde la terminal, entrar en la carpeta y ejecutar los siguientes comandos:
    - npm install
    - npm start

6.3 Instalación cliente de este Trabajo de fin de Máster (Linux):

- Con una terminal abierta en la ruta del proyecto cliente, se han de ejecutar el siguiente comando para la instalación:
    - npm install
- Una vez instalado, podemos generar el código final mediante el comando:
    - gulp build --production
- Para poder ejecutar el proyecto en un dispositivo móvil, se ha de ejecutar lo siguiente:
    - ionic platform add android@6.2.2
    - ionic run Android
- Si se quisiera ejecutar el código en un dispositivo iOS, es necesario ejecutar las siguientes instrucciones en vez de las anteriores:
    - ionic platform ad dios
    - ionic run ios