



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA
SUPERIOR
Grado en Ingeniería en Informática



**TFG 20.15 del Grado en
Ingeniería Informática**

**Estudio e implementación de
una herramienta LOW CODE
para una aplicación web**



Presentado por
Antoni Lluís García Expósito
en Universidad de Burgos – 20 de enero
de 2021

Tutores: José Manuel Galán Ordax y
Virginia Ahedo García

Información Indispensable

La aplicación está pensada para el uso en un solo terminal y por un solo usuario a la vez. Esto se debe a que está pensado para PYMES pequeñas, un sector olvidado por las empresas de software, ya que no le dan los mismos beneficios que a las grandes empresas.

Expongo esto porque a la hora de hacer pruebas por parte del tribunal, si se conectan dos usuarios a la vez puede haber inconsistencias en el tratamiento de datos.

Se añadirá en el apartado de futuros pasos, la implementación de una base de datos más completa y que la aplicación funcione en dos o más equipos a la vez. Pero no es el objetivo de este proyecto.

Índice de contenido

Apendice A (Planificación del proyecto software)	9
A.1- Introducción general	9
A.2- 1er Sprint "Inicio del proyecto" 14/10/2020-29/10/2020	10
A.3- 2do Sprint "Formación" 05/11/20-19/11/20	11
A.4- 3er Sprint "Inicio de la Aplicación" 19/11/20-3/12/20	12
A.5- 4rto Sprint "Dando forma a la Aplicación" 3/12/20-17/12/2020.....	13
A.6- 5nto Sprint "Añadiendo funcionalidades" 17/12/20-03/01/2021.....	14
A.7- 6to Sprint "Últimas funcionalidades" 03/01/21-11/01/21	15
A.8- 7mo Sprint "Optimización y Documentación" 03/01/21-10/01/21.....	17
A.8- Estudio de viabilidad	18
A.8- Viabilidad Económica	18
Costes actuales:	18
A.8- Viabilidad Legal	22
Apéndice B (Especificación de requisitos)	24
B.1- Introducción	24
B.2- Objetivos Generales	24
B.3- Catálogo de requisitos	24
Requisitos Funcionales.....	24
Requisitos NO Funcionales	27
B.4- Especificación de requisitos	28
Casos de uso:	28
Apendice C (Especificación de diseño)	39
C.1 Introducción	39
C.2- Diseño de datos	39
C.3- Campos de las Tablas	41
Pedido.....	41
TPV	42
Producto	43
ProductoPara (Pedido o TPV)	44
C.4- Diseño Arquitectónico	45

Modelo-Vista-Controlador	45
Modelo	45
Vista	45
Controlador	45
C.5- Diseño de Interfaz	46
Apendice D (Documentación técnica de programación)	48
D.1 Introducción.....	48
D.2 Estructura de directorios	48
Formes:	49
D.2 Manual del programador.....	50
OpenXava - Descarga e Instalación	50
Compilación y ejecución desde Openxava studio.....	54
Compilación de la aplicación para despliegue con Tomcat.....	56
Instalación TomCat	57
Despliegue mediante Tomcat.....	58
Conexión con un dominio	59
Apendice E (documentación de usuario).....	62
E.1 Introducción	62
E.2 Requisitos de usuario	62
E.3 Instalación	62
E.4 Manual de usuario.....	63
Inicio	63
Login.....	64
Una vez en la aplicación.....	65
Interactuar con las clases.....	65
Entidad TPV	68
Guardar venta	71
Pedido.....	71
E.5- Conclusiones manual de usuario.....	72
Apendice F (Bibliografía)	74

Índice de figuras

Ilustración 1 Tareas 1er Sprint	10
Ilustración 2 Tareas 2ndo Sprint	11
Ilustración 3 Tareas 3er Sprint	12
Ilustración 4 Tareas 4rto Sprint	13
Ilustración 5 Tareas 5nto Sprint.....	15
Ilustración 6 Tareas 6to Sprint	16
Ilustración 7 Tareas 7mo Sprint.....	17
Ilustración 8 Precios OpenXava Pro	20
Ilustración 9 Diagrama de Clases	39
Ilustración 10 Tabla Pedido.....	41
Ilustración 11 Tabla TPV.....	42
Ilustración 12 Producto.....	43
Ilustración 13 ProductoPara	44
Ilustración 14 Colores Para hoja de estilos.....	46
Ilustración 15 Hoja de Estilo.....	47
Ilustración 16 Página principal OpenXava	50
Ilustración 17 Formulario para la descarga	51
Ilustración 18 Contenido del Zip descargado.....	51
Ilustración 19 Apariencia.....	52
Ilustración 20 Nuevo Proyecto	52
Ilustración 21 Creación nuevo proyecto.....	53
Ilustración 22 Creación del proyecto Openxava.....	53
Ilustración 23 Ejecución de la aplicación	54
Ilustración 24 Apariencia del gestor de la base de datos.....	55
Ilustración 25 Crear archivo.war.....	56
Ilustración 26 Carpeta webapps.....	57
Ilustración 27 Ip y Redirección al puerto.....	59
Ilustración 28 DNS Ionos	60
Ilustración 29 Inicio de la aplicación	63
Ilustración 30 Login de usuario	64
Ilustración 31 Usuario no encontrado	64
Ilustración 32 Menu.....	65
Ilustración 33 Clientes Modo lista.....	65
Ilustración 34 Clientes Modo Modulos.....	65
Ilustración 35 Submenu comun entidades.....	66
Ilustración 36 Nuevo cliente	66
Ilustración 37 Error en los campos	67
Ilustración 38 Entidad TPV	68
Ilustración 39 Elección del empleado.....	68
Ilustración 40 Elección del cliente	69

Ilustración 41 Elección de los productos.....	69
Ilustración 42 Tpv Rellenado	70
Ilustración 43 Ticket	70
Ilustración 44 Pedido Vacío.....	71

Índice de Tablas

Tabla 1 - Total coste	21
Tabla 2 CU-01 Login Usuario	28
Tabla 3 CU-02 Añadir empleado	29
Tabla 4 CU-03 Borrar Empleado.....	29
Tabla 5 - CU-04 OrdenarEmpleado	30
Tabla 6- CU-05 Agrupar	31
Tabla 7 - CU-06 Añadir Marca/Categoría del articulo	32
Tabla 8 - CU-07 Borrar Marca/Categoría del producto.....	33
Tabla 9 - CU-08 Añadir punto pedido.....	34
Tabla 10 - CU-09 Imprimir Ticket.....	35
Tabla 11 - CU10-Abrir Pedido.....	36
Tabla 12 - CU11- Cerrar pedido	37

Apéndice A

PLANIFICACIÓN DEL PROYECTO SOFTWARE

A.1- Introducción general

Para la realización de esta TFG llegamos a un acuerdo con el tutor para trabajar mediante Metodologías Ágiles. Para ello quedamos en realizar entregas cada 10 o 15 días dependiendo de la dificultad del *sprint*.

La gestión de los *sprints* se realizará usando el repositorio de GitHub creado para el proyecto con un plugin propio "ZenHub" para su organización.

Cada Sprint cuenta con un nombre propio definido como *milestone* para así saber en qué fase del proyecto estamos de forma más natural.

Otro punto a valorar es que al tratarse de una aplicación con un plazo de entrega relativamente corto, se van a concentrar muchas tareas en cada uno de los *sprints* y no se realizarán tantos como me hubiese gustado.

A.2- 1er Sprint "Inicio del proyecto" 14/10/2020-29/10/2020

Como bien indica el nombre del *sprint*, estas primeras sesiones han servido tanto para la configuración del GitHub y ZenHub, así como para el estudio de la tecnología que se va a usar. En este caso se ha optado por una herramienta *Low Code* y sus posibles alternativas en el mercado.

Para documentar, correctamente, decidimos usar Zotero ya que es una alternativa fácil y completa a Mendeley. Además de ser una aplicación de escritorio que cuenta con extensiones para Mozilla cuenta con un plugin para Word que facilitará la citación correcta de nuestros documentos.

A cada tarea (*Issue*) le asignaremos un valor numérico que equivaldrá al tiempo en horas estimado. Cabe destacar que realizamos esta valoración en la revisión del primer *sprint*, es decir que, en el primer sprint, el Story Point hace referencia a la dificultad que se ha presupuesto antes de realizar dicha tarea. A partir del segundo ya sí que serán las horas que se estima que vaya a durar dicha tarea.

Al ser el primer *sprint*, podemos ver que casi todas las etiquetas hacen referencia a documentación, configuración, elección de software o diseño. A partir del segundo ya habrá más etiquetas relacionadas con el desarrollo o la formación.

Remaining Issues and Pull Requests			Story points
1	Instalación y aprendizaje ZenHub Configuration TFG_GII_20.15 #1 III In Progress Inicio del proyecto		3
1	Configuración del repositorio Configuration TFG_GII_20.15 #2 III Done Inicio del proyecto		2
1	Herramienta de prototipado Software choice TFG_GII_20.15 #3 III Done Inicio del proyecto		1
1	1er Sprint 22/10/20 a 29/10/20 documentation TFG_GII_20.15 #4 III Sprint Backlog Inicio del proyecto		Not estimated
1	Estudios relacionados documentation TFG_GII_20.15 #5 III Done Inicio del proyecto		5
1	Primer Prototipo Design TFG_GII_20.15 #6 III Product Backlog Inicio del proyecto		5
1	Documentar primer Sprint documentation TFG_GII_20.15 #7 III New Issues Inicio del proyecto		2

Ilustración 1 Tareas 1er Sprint

Al final no realicé todas las tareas ya que tuve problemas con el software seleccionado para el diseño del primer prototipo. Así pues, después de la entrega decidimos cambiar de Pencil a Adobe XD para poder así conseguir un prototipo más adecuado.

A.3- 2do Sprint "Formación" 05/11/20-19/11/20

En este *sprint* el foco estuvo en la formación, realizando el tutorial ofrecido desde la misma herramienta OpenXava. Es un tutorial que está orientado a los primeros pasos de la herramienta. Al acabarlo el resultado es una pequeña aplicación de facturación operativa.

Lo más interesante es que se pueden ir comprobando los cambios al instante, así si algo no resulta convincente se puede cambiar sin la necesidad de que se vea alterado todo el proyecto.

Durante la realización del tutorial surgieron dudas acerca del futuro de este proyecto ya que la idea inicial era la realización de una aplicación que combinase una herramienta ERP y una aplicación de venta online, pero al finalizar el tutorial fue evidente que la idea inicial no era posible con este lenguaje. Así pues, tras la última tutoría se acordó con los tutores dedicar todo el esfuerzo a realizar una herramienta ERP más completa. Tras este cambio el objetivo final ya estaba claro y era posible.

La formación ha sido separada en módulos para así facilitar el seguimiento de los pasos. Dichos módulos son cada una de las *issues* que han surgido durante esta etapa de formación. Como corregimos en el otro *sprint* ahora el peso se calcula con las horas previstas para la realización de dicha tarea. Como ya se ha dicho y se puede observar, en este *sprint* casi todas las tareas son de formación.

Completed Issues and Pull Requests			Story points
 OpenXava Configuration	TFG_GII_20.15 #8	III Closed  Etapa de formación	②
 Primer tutorial Formation	TFG_GII_20.15 #9	III Closed  Etapa de formación	①
 Leccion 2 Modelar con Java Formation	TFG_GII_20.15 #10	III Closed  Etapa de formación	①
 Tercera leccion, "Pruebas Automáticas" Formation	TFG_GII_20.15 #11	III Closed  Etapa de formación	②
 Estilo Gráfico de la aplicación Formation	TFG_GII_20.15 #12	III Closed  Etapa de formación	③

Ilustración 2 Tareas 2do Sprint

Las horas previstas son menores a las reales, ya que la agilidad esperaba no se dio y por tanto los tiempos fueron más extensos.

A.4- 3er Sprint "Inicio de la Aplicación" 19/11/20-3/12/20

Este ha sido un sprint accidentado, en la primera semana el ordenador sufrió un accidente y el proyecto se vio afectado a causa de haber estado una semana sin poder trabajar en condiciones. Durante este periodo escogí un programa para la realización de diagramas UML para la gestión correcta de mis clases y de mis paquetes.

La herramienta elegida es Draw.io que es un software gratuito que permite modelar de forma rápida y sencilla.

Remaining Issues and Pull Requests				Story points
①	Herramienta UML	Research Software choice	TFG_GII_20.15 #13 000New Issues	2
①	Creación del diagrama UML	Design	TFG_GII_20.15 #14 000New Issues	6
①	3er Sprint	documentation	TFG_GII_20.15 #15 000Sprint Backlog	Not estimated

Ilustración 3 Tareas 3er Sprint

A.5- 4rto Sprint "Dando forma a la Aplicación" 3/12/20-17/12/2020

Tras los inconvenientes del tercer Sprint, solucionado el problema, el ritmo ha vuelto a la normalidad. Estas han sido dos semanas muy intensas de trabajo, donde lo que a primera vista parecía un sprint muy mecánico se ha convertido en un sprint de conflictos con diferentes lenguajes (java, openxava, html, css).

Al final no he conseguido el objetivo de acabar lo que sería la aplicación ya que estos dos últimos días me han surgido los siguientes problemas:

- A) A la hora de personalizar la página inicial, para poner el logotipo de la empresa como imagen de bienvenida he observado que tengo dificultades trabajar con HTML y CSS.
- B) En la página de TPV, no ha habido forma de poder poner el precio final, multiplicando la cantidad del artículo por el precio de dicho. Este ha sido el mayor inconveniente.

Por ejemplo, si de un collar quiero 2 unidades, solo he sido capaz de poner el precio unitario. Lo he probado de todas las maneras que conozco en java, ya sea iterando la colección de productos, cogiendo desde una clase exterior los datos y multiplicándolos fuera, sin obtener el resultado buscado.

A pesar de la frustración, he descubierto un foro activo de este *framework*, y estoy a la espera de alguna respuesta a ver si la comunidad me puede guiar un poco.

Como la aplicación está casi acabada, se me están ocurriendo algunas clases más que en nuestro negocio no nos son útiles pero que para una pyme mayor sí que servirían.

Así que para el quinto sprint acabaré de implementar dichas clases por si algún día se expande la tienda y fuera necesarias.

Remaining Issues and Pull Requests				Story points
①	Creación del proyecto	Design		2
TFG_GII_20.15 #16 New Issues Dando forma a la Aplicación				
①	Aprendiendo CSS	Design	Research	3
TFG_GII_20.15 #17 New Issues Dando forma a la Aplicación				
①	Creación de todas las entidades	Design	Implementación	21
TFG_GII_20.15 #18 New Issues Dando forma a la Aplicación				

Ilustración 4 Tareas 4rto Sprint

A.6- 5to Sprint "Añadiendo funcionalidades" 17/12/20-03/01/2021

Este sin duda ha sido el *sprint* más productivo y en el que más he avanzado. Tras lograr arreglar el problema con la suma de los totales en la clase "tpv", ya tenía casi acabada del todo la aplicación respetando los requisitos iniciales.

No obstante, en la tutoría dedicada a dicho sprint ambos tutores me sugirieron dos funcionalidades muy interesantes y que me llevaron mucho más trabajo de lo que pensaba en un primer momento,

Añadir un MRP básico, tras una búsqueda profunda, llegamos a la conclusión de que sería expandir excesivamente el proyecto dado el tiempo restante y que solo una persona se encuentra realizando el proyecto. Además, solo implementaría un punto pedido básico: que en el momento que un producto llegase a una cantidad prefijada se añadiese directamente al pedido.

Esta funcionalidad se encuentra en un punto intermedio, es decir, al hacer una venta que haga que el producto llegue a una cantidad mínima, sí que se crea un pedido, pero se crea un pedido nuevo cada vez, hecho que no es el que estamos buscando.

Estamos buscando que el sistema detecte el pedido que este abierto y una vez encontrado que añada el producto allí. En caso de que ese producto ya contuviese el mismo producto, lo único que tendría que hacer es aumentar la cantidad que se pedirá al realizar el pedido. Y en caso de que no lo contenga simplemente añadir el producto a pedido.

La otra funcionalidad requerida, es aún más complicada, pero completamente necesaria. Se trata de la creación de un *ticket* para cada venta, algo básico en cualquier ERP. Aquí el problema radica en el diseño del ticket, ya que OpenXava funciona con JasperReport que es una librería para la creación de *reports* (*tickets*, informes, gráficos, etc.).

Dicho *report* ha de ser diseñado con un editor externo, como ireport o jaspersoft studio. En principio está pensado para la realización con el primero, lo que desafortunadamente dicho programa se dejó de actualizar en 2015. Actualmente es inutilizable por la cantidad de *bugs*. Por esta razón se ha optado por utilizar JasperSoft studio, un editor que funciona tanto poniendo campos de parámetro como conectándolo a una base de datos.

Para la realización del *ticket* se ha cogido el *ticket* que por defecto proporciona OpenXava hecho en ireport y que posteriormente ha sido modificado para satisfacer las necesidades del proyecto.

Este *sprint* también ha sido realmente útil ya que gracias a él he descubierto las Acciones de OpenXava y he podido observar cómo Cada acción se comporta por defecto (grabar, crear, modificar, borrar), pero Openxava nos permite modificarlas para que actúen como nosotros queramos.

De hecho, para la implementación del tpv se ha tenido que modificar la acción de grabar (guardar), para que actuase como se requería en ese momento.

Remaining Issues and Pull Requests				Story points
①	Añadir punto pedido	Implementación	TFG_GII_20.15 #19	21
	New Issues			
	Añadiendo funcionalidades			
①	Creación del ticket/factura	Implementación	TFG_GII_20.15 #21	13
	New Issues			
	Añadiendo funcionalidades			

Ilustración 5 Tareas 5to Sprint

A.7- 6to Sprint “Últimas funcionalidades” 03/01/21-11/01/21

Ya nos encontramos en uno de los últimos *sprints*, concretamente en el último de implementación. En este hemos conseguido acabar con las dos tareas pendientes: finalizar el punto pedido y la implementación de uno *report* que nos sirva de factura o *ticket* de venta.

En cuanto al punto de pedido, no resulta del todo satisfactorio el resultado, ya que al ser solo una persona no lo he podido realizar como me hubiese gustado. Al final he optado por un punto de pedido semi automático. Al realizar una venta que ocasione que el producto en cuestión esté por debajo del mínimo requerido, sí que se añade automáticamente al pedido abierto. Y es por lo siguiente por lo cual indico que se trata de un proceso semi automático: ya que es el usuario quien se tiene que encargar de cerrar el pedido cuando él considere que es oportuno, y antes de irse, tiene que crear un pedido vacío, para que los productos se vayan acumulando en este pedido nuevo.

Esta solución no es óptima para grandes empresas, pero al ser una aplicación pensada para una pyme de un solo terminal, es una solución que no supone un problema. En el apartado de próximos pasos, aparecerá la mejora de este punto, ya que es una funcionalidad muy interesante para la escalabilidad de la aplicación hacia un mercado no solo de pymes.

Ahora llegamos al punto más satisfactorio, que es la implementación de un ticket/factura personalizado para cuando se realiza una venta.

A pesar de que el diseño está pensado para una impresora (normal A4), solo faltaría adaptarlo para una impresora térmica de tickets que son las que normalmente se utilizan en los negocios.

Al final se tuvo que crear una acción nueva que modificase a la de imprimir para añadir dentro del modo lista de mi entidad, para que imprimiese lo que aparecía en pantalla en aquel momento antes de que se guardara el *ticket* en sí.

El mapeo al final tuvo su dificultad ya que se tienen que tratar diferente los parámetros que contiene la entidad y los campos que están dentro de la clase embebida que tiene la entidad, pero al final tras insistir organizando bien los datos se pudo crear el ticket que se estaba buscando.

La parte negativa del *sprint* viene dada por el despliegue con el tomcat de apache. Está resultando una tarea complicada la subida en local de la aplicación. Al ya agotar todas las opciones posibles conocidas se ha optado por conectar con el foro del framework para ver si algún otro usuario ha tenido el mismo problema y así pueda tener una solución que también me valga o al menos obtener ayuda sobre cómo afrontar el problema. En cuando tenga solucionado este problema no será difícil pasarlo a un host gratuito para así poder direccionarlo con el dominio que tengo en posesión para la facilidad del uso por parte del usuario final.

Remaining Issues and Pull Requests			Story points
1	Finalizar el punto pedido Implementación		8
TFG_GII_20.15 #22 New Issues Últimas funcionalidades			
1	Finalizar la implementación del ticket Implementación		6
TFG_GII_20.15 #23 New Issues Últimas funcionalidades			
1	Despliegue web Implementación		5
TFG_GII_20.15 #25 New Issues Últimas funcionalidades			

Ilustración 6 Tareas 6to Sprint

A.8- 7mo Sprint "Optimización y Documentación" 03/01/21-10/01/21

En este *sprint* se han realizado las tareas que se deberían hacer día a día pero que se van dejando y al final pasan factura. Entre ellas ir borrando líneas de código de prueba y documentar toda la práctica.

Además, al ser consciente de que dos clases hacían prácticamente lo mismo para optimizar un código se han juntado en una sola, así solo ha quedado una clase. Al hacer este cambio, se han tenido que retocar ligeramente las otras entidades que dependían de ellas. Echo que ha roto los cálculos de importes y la impresión por pantalla de algunos datos. Al desconocer cómo funciona el framework por debajo, no lo podía solucionar de manera rápida, así que la mejor opción fue reescribir las dos entidades afectadas de nuevo, aplicando los cambios desde el principio y resetear la base de datos. Una vez reescritas no ha habido ningún problema y se ha logrado ahorrar una clase.

A la hora de comentar toda la practica se ha usado la anotación básica del javadoc con sus anotaciones. En este punto ha surgido la duda de que al ser una herramienta lowcode, mucho del código es reciclado con pequeñas modificaciones y no sabía si podía poner mi autoría en la clase, así que he decidido solo ponerme como autor en las entidades o clases en las cuales la mayor parte del código fuese mío o modificado del de por defecto.

Otro punto para comentar es que al final por falta de tiempo y porque no afecta a la funcionalidad, la personalización de la aplicación en cuanto a imágenes y otros detalles al final se han descartado, quedando la aplicación eso sí, con los colores corporativos.

Remaining Issues and Pull Requests				Story points
❗	Eliminacion de clase repetida	Implementación	TFG_GII_20.15 #20	5
	New Issues Optimizar y Comentar código			
❗	Comentar toda la practica	documentation	TFG_GII_20.15 #24	2
	New Issues Optimizar y Comentar código			

Ilustración 7 Tareas 7mo Sprint

A.8- Estudio de viabilidad

En este apartado vamos a estudiar la viabilidad económica del proyecto en caso de que se lanzara al público.

A.8- Viabilidad Económica

La aplicación que se ha podido desarrollar es un ERP básico, con muchas posibilidades de crecimiento, así que a continuación se separarán los costes de lo que sería la aplicación tal y como está y los costes de la previsión de lo que sería la aplicación acabada al 100%.

Costes actuales:

Para empezar, vamos a detallar los costes de la aplicación,

- **Costes de Personal:**

Esta parte la calcularemos con los precios establecidos por la seguridad social en su página web: <http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537> .

La aplicación al ser realizada por un estudiante establecemos el precio de las horas a 15€/h. Durante estos tres meses de trabajo ha habido semanas con más carga de trabajo que otras. Así que en total se podría decir que se han trabajado una media de 30 horas a la semana durante los 3 meses que ha durado el proyecto aproximadamente (de finales de octubre a finales de enero).

$$30 \frac{\text{Horas}}{\text{Semana}} \times 15 \frac{\text{€}}{\text{Hora}} \times 4 \frac{\text{Semanas}}{\text{Mes}} = 1800\text{€ al mes}$$

Este sería el salario neto al mes para el estudiante. Para el cálculo del salario bruto se tendrían que aplicar los diferentes impuestos y las ratios que sugiere el Régimen general de la Seguridad Social, pero al ser un tema estrictamente económico resulta más conveniente dejar estos cálculos a gestorías u órganos especializados.

Pero para hacernos una idea, vamos a calcular de manera básica el salario bruto de lo que cuesta realmente el estudiante al aplicar los impuestos.

Los impuestos son: (sacados de la misma web de la SS)

- 23.6 % contingencias.
- 5.5 % desempleo.
- 0.20 % FOGASA.
- 0.60 % formación profesional.

Teniendo en cuenta estos impuestos calcularemos el gasto real:

$$\frac{1800 \frac{\text{€}}{\text{mes}}}{1 - (0,236 + 0,055 + 0,002 + 0,006)} = 2.567,76\text{€ al mes}$$

Como el proyecto ha durado 3 meses así que el gasto total de personal sube a un total de 7703,28€

- **Costes hardware**

Para la implementación de esta aplicación no es necesario ningún hardware específico, ya que se puede trabajar en un ordenador básico. En este caso se ha trabajado en un ordenador personal. Así que diría que gastos por hardware no hacen falta.

- **Costes por software:**

Para este TFG se ha optado por buscar siempre herramientas OpenSource gratuitas, así que para su implementación y despliegue el coste también es 0.

Es verdad que existe una versión de pago del framework de OpenXava con funcionalidades muy interesantes pero que para este TFG no resultaban necesarias.

El precio de la licencia de la versión de pago varía en cuanto a las funcionalidades que necesites.



Ilustración 8 Precios OpenXava Pro

En cuanto al dominio sí que ha de ser adquirido, después de realizar una búsqueda, y de comparar diversas opciones se ha optado por comprar el dominio mediante 1&1 o lo que actualmente es Ionos.

Cuyo precio es de 8€ el primer año y 10€ a partir de entonces.

(Precios obtenidos de: <https://www.ionos.es/dominios/com-dominio>)

- **Costes indirectos:**

Estos se componen de los gastos básicos como luz e internet.

La factura de la luz se ha visto incrementada unos 50€ del precio habitual a causa de tener el equipo en funcionamiento más tiempo de lo habitual. Otro factor que ha hecho que aumente la factura es el uso de estufas para mantener la oficina caliente.

La tarifa de internet es de 30€ mensuales, ya que no se necesita una gran conexión.

$$\left(50 \frac{\text{€}}{\text{Mes}} + 30 \frac{\text{€}}{\text{Mes}}\right) \times 3\text{Meses} = 240\text{€}$$

- **Total:**

Tipo de coste	Total
Personal	7703,28€
Hardware	0
Software	8€
Indirectos	240€
Total	7951,28€

Tabla 1 - Total coste

El coste total aproximado para la creación de esta aplicación sería de unos 7951,28€ lo que equivale a 2.650€ al mes.

- **Beneficios:**

Al tratarse de un ERP básico para su comercialización se deberían mejorar bastantes de los puntos. Así pues, no tendría cabida en el mundo comercial. De ese modo en los beneficios en estos momentos solo entrarían la finalización del grado, en caso de aprobar el TFG.

A.8- Viabilidad Legal

- OpenXava cuenta con una licencia tipo LGPL que permite realizar aplicaciones comerciales de forma gratuita.
- GitHub cuenta con una licencia GNU.
- HSQLBD es el gestor de bases de datos por defecto de Java y su licencia es de tipo BSD.
- Jaspersoft Studio es un software libre con licencia AGLP.

Como se puede observar, se han utilizado, en medida de lo posible, herramientas que sean de licencia libre y código abierto. Por este mismo motivo mi aplicación es de distribución libre sin licencia.

Apéndice B

ESPECIFICACIÓN DE REQUISITOS

B.1- Introducción

En este anexo se detallarán los diferentes objetivos y requisitos que debe cumplir nuestra aplicación. La mayoría son requisitos y objetivos propuestos al principio del proyecto, pero hay alguno que se ha ido añadiendo durante la etapa de desarrollo.

B.2- Objetivos Generales

Marcamos los objetivos de este proyecto con estos puntos:

- Analizar el mercado para saber cuál es la mejor opción para el desarrollo del proyecto.
- Desarrollar una aplicación que cumpla el cometido de una ERP, para un negocio de tipo PYME.
- Añadir un MRP básico, a modo de punto pedido.

B.3- Catálogo de requisitos

Vamos a utilizar la nomenclatura estándar para los requisitos, RFX para los requisitos funcionales y RNFX para los requisitos no funcionales.

Requisitos Funcionales

- **RF1- Login de usuarios:** El usuario debe poder acceder a la aplicación mediante unas credenciales propias y personalizadas.

- **RF2- Interactuar con empleado:** La aplicación debe ser capaz de permitir al usuario ejecutar las acciones por defecto de la entidad.
 - **RF2.1- Crear empleado:** El usuario debe poder añadir un nuevo empleado.
 - **RF2.2- Borrar empleado:** El usuario debe ser capaz de borrar a un empleado.
 - **RF2.3-Listar y ordenar:** El usuario debe poder listar a los empleados y ordenarlos dependiendo de sus características
 - **RF2.4-Agrupar:** El usuario debe poder agrupar a los empleados por un campo en concreto.

- **RF3- Interactuar con cliente:** La aplicación debe poder interactuar con la entidad cliente y dejar al usuario ejecutar las acciones por defecto de la entidad.
 - **RF3.1- Crear cliente:** El usuario debe poder añadir nuevos clientes en caso de que sea necesario para nuevas facturas.
 - **RF3-2- Borrar cliente:** El usuario debe ser capaz de borrar un cliente específico.
 - **RF3.3-Listar y ordenar:** El usuario debe poder listar a los clientes y ordenarlos dependiendo de sus características
 - **RF3.4-Agrupar:** El usuario debe poder agrupar a los empleados por un campo en concreto.

- **RF4-Interactuar con la entidad marca:** La aplicación debe ser capaz de dejar al usuario ejecutar las acciones por defecto de la entidad
 - **RF4.1- Crear marca/proveedor:** El usuario debe poder añadir una marca o un proveedor nuevo.
 - **RF4.2- Borrar mara/proveedor:** El usuario debe poder borrar una marca o un proveedor.
 - **RF4.3-Listar y ordenar:** El usuario debe poder listar a las marcas o proveedores y ordenarlos dependiendo de sus características

- **RF5-Interactuar con la entidad Categoría del artículo:** La aplicación debe dejar al usuario ejecutar las acciones por defecto de la entidad.
 - **RF5.1- Crear categoría del artículo:** El usuario debe poder añadir que tipo de artículo es.
 - **RF5.2-Borrar categoría del artículo:** El usuario debe ser capaz de borrar la categoría del artículo.
 - **RF5.3-Listar y ordenar:** El usuario debe poder listar la categoría del artículo y ordenarlos dependiendo de sus características

- **RF6-Interactuar con la entidad factura pendiente:** La aplicación debe dejar al usuario ejecutar las acciones por defecto de la entidad.
 - **RF6.1- Crear factura pendiente:** El usuario debe poder añadir facturas pendientes.
 - **RF6.2-Borrar facturas pendientes:** El usuario debe ser capaz de borrar una factura 11pendiente.
 - **RF6.3-Listar y ordenar:** El usuario debe poder listar y ordenar todas las facturas pendientes.
 - **RF6.4 Agrupar:** El usuario debe ser capaz de agrupar las facturas pendientes mediante sus características.

- **RF7-Interactuar con la entidad Producto:** La aplicación debe dejar al usuario ejecutar las acciones por defecto de la entidad.
 - **RF7.1-Crear producto:** El usuario debe ser capaz de añadir un nuevo producto.
 - **RF7.1.1- Definir punto pedido:** El usuario debe ser capaz de añadir un punto pedido.
 - **RF7.2-Borrar Producto:** El usuario debe ser capaz de borrar un producto.
 - **RF7.3-Listar y ordenar:** El usuario debe ser capaz de listar y ordenar los productos mediante sus campos.
 - **RF7.4-Agrupar:** La aplicación debe ser capaz de agrupar los productos dependiendo de los campos.

- **RF8- Interactuar con la entidad TPV:** La aplicación debe ser capaz de permitir al usuario la interacción con la entidad tpv.
 - **RF8.1 Crear una nueva venta:** El usuario debe ser capaz de realizar una nueva venta reflejada en un ticket.
 - **RF8.1.1-Creación del ticket:** La aplicación debe ser capaz de crear un *report* con los datos que aparecen en pantalla antes de que se guarden.
 - **RF8.1.2-Impresión del ticket:** El usuario debe ser capaz de la impresión del ticket en formato PDF.
 - **RF8.2- Borrar una venta:** El usuario debe ser capaz de borrar una venta.
 - **RF8.3-Listar y ordenar:** El usuario debe ser capaz de listar y ordenar las ventas mediante sus campos.
 - **RF8.4-Agrupar:** La aplicación debe ser capaz de agrupar las ventas dependiendo de los campos.
- **RF9-Interactuar con la entidad Pedido:** La aplicación debe permitir al usuario gestionar sus pedidos.
 - **RF9.1-Abrir pedido:** El usuario debe ser capaz de abrir un pedido.
 - **RF9.2-Cerrar pedido:** El usuario debe se capaz de cerrar un pedido

Requisitos NO Funcionales

- **RNF1-Usabilidad:** La aplicación debe ser fácil de usar para mejorar en todo lo posible la experiencia del usuario. Hacer una interfaz intuitiva y agradable ayuda a dicha experiencia.
- **RNF2-Eficiencia:** No debe haber mucho tiempo en el cambio de ventanas y procesamiento de datos.
- **RNF3-Escalabilidad:** La aplicación debe estar preparada para un aumento de prestaciones y procesamiento de más cantidades de trabajo.

B.4- Especificación de requisitos

Como casi todos los requisitos tienen los mismos puntos, en los casos de uso, lo que haré será especificar para que requisitos sirviera el mismo caso de uso.

Casos de uso:

CU-01	Login de usuario
Requisitos Relacionados	RF1
Descripción	Permitir al usuario entrar en su sesión.
Precondiciones	Que se le haya autorizado al usuario y que se haya dado su contraseña.
Acciones	<ol style="list-style-type: none">1. El usuario entra en la aplicación.2. El usuario introduce sus credenciales3. Pulsar el botón de entrar
Postcondiciones	El usuario debe coincidir con uno de los que estén habilitados
Excepciones	El usuario haya cometido una errata y no este bien escrito o el usuario o la contraseña.
Importancia	Alta

Tabla 2 CU-01 Login Usuario

CU-02	Añadir Empleado
Requisitos Relacionados	RF2 RF2.1
Descripción	Permitir al usuario añadir un nuevo empleado.
Precondiciones	El usuario debe estar loggeado y en la pestaña de empleado.
Acciones	<ol style="list-style-type: none">1. El usuario pulsa nuevo en la pestaña de empleado2. El usuario introduce los campos del empleado3. Pulsar el botón de grabar
Postcondiciones	-
Excepciones	-No haber escrito alguno de los campos obligatorios
Importancia	Media

Tabla 3 CU-02 Añadir empleado

CU-03	Borrar
Requisitos Relacionados	RF2.2
Descripción	Permitir al usuario borrar un nuevo empleado.
Precondiciones	-El empleado a borrar debe estar creado con anterioridad.
Acciones	<ol style="list-style-type: none"> 1. El usuario se sitúa encima del empleado que quiere borrar. 2. El usuario verá que aparece una papelera a la izquierda de su nombre. 3. El usuario debe pulsar el botón y se borrara el usuario.
Postcondiciones	-
Excepciones	-No haber escrito alguno de los campos obligatorios -El empleado no puede tener ningún ticket a su nombre.
Importancia	Media

Tabla 4 CU-03 Borrar Empleado

CU-04	Ordenar Empleado
Requisitos Relacionados	RF2 RF2.3
Descripción	Permitir al usuario listar y ordenar a los empleados por un campo en concreto
Precondiciones	Estar en modo lista.
Acciones	<ol style="list-style-type: none"> 1. El usuario se sitúa encima del campo de empleado con el que quiere realizar la ordenación. 2. Pulsa encima de las flechas que le salen por pantalla.
Postcondiciones	-
Excepciones	-
Importancia	Baja

Tabla 5 - CU-04 OrdenarEmpleado

Los últimos tres casos de uso también servirían para la creación del cliente (RF3) y para la creación de facturas pendientes (RF6)

CU-05	Agrupar
Requisitos Relacionados	RF2.4 RF3.4 RF6.4 RF7.4 RF8.4
Descripción	Permitir al usuario agrupar mediante un solo campo a los objetos.
Precondiciones	Estar en modo lista.
Acciones	<ol style="list-style-type: none"> 1. El usuario despliega la lista de arriba a la izquierda de la pantalla. 2. Pulsa encima del criterio que quiere ordenar.
Postcondiciones	-
Excepciones	-
Importancia	Baja

Tabla 6- CU-05 Agrupar

Este último caso de uso es uno de los que es común para todos los requisitos, y se procede ejecutando los mismos pasos para todas las clases.

CU-06	Añadir Marca/Categoría del artículo
Requisitos Relacionados	RF4.1 RF5.1
Descripción	Permitir al usuario añadir una nueva marca/ categoría del artículo
Precondiciones	El usuario debe estar loggeado y en la pestaña de correspondiente.
Acciones	<ol style="list-style-type: none"> 1. El usuario pulsa nuevo en el modo lista de la pestaña 2. El usuario introduce el nombre de la característica 3. Pulsar el botón de grabar
Postcondiciones	-
Excepciones	-No haber escrito alguno de los campos obligatorios
Importancia	Media

Tabla 7 - CU-06 Añadir Marca/Categoría del artículo

CU-07	Borrar Marca/Categoría del artículo
Requisitos Relacionados	RF4.2 RF5.2
Descripción	Permitir al usuario borrar una nueva marca/ categoría del artículo
Precondiciones	Que no haya ningún producto con la marca/categoría.
Acciones	<ol style="list-style-type: none"> 1. El usuario se sitúa encima de la marca/categoría que quiere borrar. 2. El usuario verá que aparece una papelera a la izquierda de su nombre. 3. El usuario debe pulsar el botón y se borrara el usuario.
Postcondiciones	-
Excepciones	Que no se haya eliminado el producto que tenga el campo que queremos borrar.
Importancia	Media

Tabla 8 - CU-07 Borrar Marca/Categoría del producto

Tanto el CU-06 y el CU-07 sirven para las entidades marca y categoría de artículo, ya que actúan de la misma manera.

CU-08	Añadir punto pedido
Requisitos Relacionados	RF7.1.1
Descripción	Permitir al usuario añadir un punto pedido, (stock mínimo que quiere tener)
Precondiciones	Estar creando el producto o modificando uno.
Acciones	<ol style="list-style-type: none"> 1. El usuario crea un producto o modifica uno. 2. Escribe la cantidad que quiere en punto pedido. 3. Al hacer una venta de dicho producto y que su stock sea menor que el punto pedido, se añadirá la cantidad necesaria para llegar al punto pedido al pedido.
Postcondiciones	-
Excepciones	-
Importancia	Alta

Tabla 9 - CU-08 Añadir punto pedido

CU-09	Imprimir ticket de compra
Requisitos Relacionados	RF8.1 RF8.2
Descripción	Permitir al usuario crear un ticket de una compra y que lo puede imprimir
Precondiciones	La entidad tpv debe estar rellena por completo. -Tener activadas las pestañas emergentes en tu navegador.
Acciones	<ol style="list-style-type: none"> 1. El usuario rellena todos los datos de tpv. 2. Pulsar el botón imprimir pdf. 3. Cambiará de pestaña automáticamente y se vera el ticket.
Postcondiciones	-
Excepciones	-No haber escrito alguno de los campos obligatorios
Importancia	Alta

Tabla 10 - CU-09 Imprimir Ticket

CU-10	Crear Pedido
Requisitos Relacionados	RF9.1
Descripción	Permite al usuario abrir un pedido.
Precondiciones	El usuario se debe asegurar que solo haya abierto un ticket a la vez.
Acciones	<ol style="list-style-type: none"> 1. El usuario crea un pedido nuevo. 2. Le pone el Id correspondiente. 3. Deja los demás campos en blanco. 4. El usuario pulsa el botón de grabar.
Postcondiciones	-Asegurarse de que solo hay un pedido abierto.
Excepciones	-No haber escrito la Id del pedido.
Importancia	Alta

Tabla 11 - CU10-Abrir Pedido

CU-11	Cerrar pedido
Requisitos Relacionados	RF9.2
Descripción	Permitir al usuario cerrar un pedido.
Precondiciones	-Que haya algún pedido abierto.
Acciones	<ol style="list-style-type: none"> 1. El usuario abre el pedido que quiere cerrar. 2. Activa el checkbox de está pedido. 3. Pulsa el botón grabar.
Postcondiciones	-Tiene que abrir manualmente otro pedido.
Excepciones	
Importancia	Alta

Tabla 12 - CU11- Cerrar pedido

Apéndice C

ESPECIFICACIÓN DE DISEÑO

C.1 Introducción

En este anexo vamos a explicar cómo se han organizado los datos de la aplicación y los diseños escogidos.

C.2- Diseño de datos

Nuestra base de datos está formada por diferentes tablas, que se relacionan entre sí de la siguiente manera:

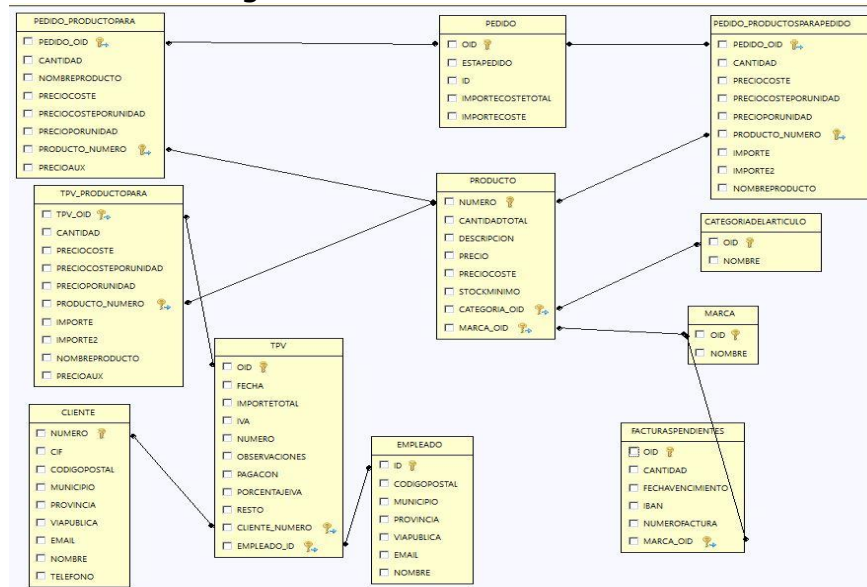


Ilustración 9 Diagrama de Clases

- **Pedido:** En esta tabla se almacenarán los pedidos que se vayan haciendo. El identificador se tiene que ir aumentando manualmente. Contiene un booleano para saber si el pedido está pedido o aún no.
 - Esta tabla se relaciona con Producto mediante Pedido_ProductoPara ya que el pedido contiene productos.

- **TPV:** En esta tabla se almacenarán las ventas que se harán. Como ocurre en la tabla Pedido, esta está relacionada con Producto mediante TPV_ProdcutoPara ya que la venta contendrá productos. También está relacionada con la tabla Empleado y Cliente para tener más información a la hora de realizar la venta.
- **Producto:** Esta es la tabla con más relaciones y, viéndolo desde el punto de vista de una tienda real, es normal que sea el epicentro del diagrama. Ya que casi todas las tablas se verán directa o indirectamente afectadas por los productos. En esta tabla se almacenarán los productos que tiene la tienda.
- **Cliente:** En esta tabla se almacenarán a los clientes que tendremos en la tienda.
- **Empleados:** En esta tabla se almacenarán a los empleados que trabajarán en la tienda.
- **Facturas Pendientes:** En esta tabla es donde llevaremos el control de las facturas que nos quedan por pagar.
- **Marca:** En esta tabla se almacenarán las marcas con las que trabajarán en la tienda.
- **Categoría del Producto:** En esta tabla se almacenarán la categoría del producto

C.3- Campos de las Tablas

Todas las tablas tienen un identificador que es OID en todas exceptuando en dos en las cuales el identificador es un entero que actúa de PK.

Ya que muchas clases son secundarias o embebidas, solo veremos los campos de las tablas más importantes.

Pedido



Ilustración 10 Tabla Pedido

- **EstaPedido:** Es un booleano que será el que se encargará de identificar si el pedido está abierto o no. En caso de estar marcado significará que el pedido está cerrado.
- **Id:** Es un Integer que actúa de "identificador" para saber qué número de pedido es por pantalla.
- **Importe Coste:** Es un BigDecimal que calculará la multiplicación de la cantidad de Producto por el precio de coste del producto.
- **Importe Coste Total:** Es un BigDecimal que calculará la suma de todos los ImporteCoste que tenga en pedido.

TPV

TPV	
<input type="checkbox"/>	OID
<input type="checkbox"/>	FECHA
<input type="checkbox"/>	IMPORTETOTAL
<input type="checkbox"/>	IVA
<input type="checkbox"/>	NUMERO
<input type="checkbox"/>	OBSERVACIONES
<input type="checkbox"/>	PAGACON
<input type="checkbox"/>	PORCENTAJEIVA
<input type="checkbox"/>	RESTO
<input type="checkbox"/>	CLIENTE_NUMERO
<input type="checkbox"/>	EMPLEADO_ID

Ilustración 11 Tabla TPV

- **Fecha:** Es un atributo de tipo Date, que se pondrá automáticamente en el día correspondiente.
- **Importe Total:** Es un BigDecimal que sumará todos los precios más el IVA, para obtener el total de la venta.
- **Observaciones:** Un campo de tipo texto para anotar alguna observación en caso de que se crea oportuno.
- **PagaCon:** Es un bigDecimal que es el dinero con el que nos paga el cliente.
- **Porcentaje IVA:** es un entero capado a dos cifras para definir el % que queremos sumarle de IVA.
- **Cliente:** Es la clave primaria de cliente y nos servirá para definir qué cliente ha hecho la compra.
- **Empleado:** Es la clave primaria del Empleado y nos servirá para saber quién ha realizado la venta.

Producto

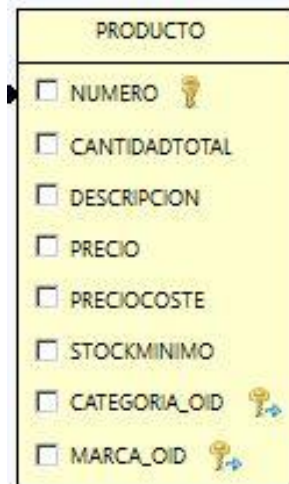


Ilustración 12 Producto

- **Numero:** Es un entero que será la PK de la clase producto y que será de generación manual.
- **Cantidad Total:** Es el stock que tiene la tienda en el momento.
- **Descripción:** Es un String para el nombre del producto
- **Precio:** Es un BigDecimal que se encarga de darnos el precio real del producto.
- **Precio coste:** Es un BigDecimal que se encarga de darnos el precio de coste del producto.
- **Stock Mínimo:** Es el punto pedido, que se encarga de mantener nuestro stock siempre con un mínimo, en caso contrario se añade a pedido.
- **Categoría:** Es la id de categoría para saber qué tipo de producto es.
- **Marca:** Es la id de la marca para saber de qué marca es el producto.

ProductoPara (Pedido o TPV)

Esta es la llamada común que tienen tanto pedido como tpv a producto y se encarga de la colección de Productos que tienen ambas entidades. Lo único que cambia es el TPV_OID por Pedido_OID



TPV_PRODUCTOPARA	
<input type="checkbox"/>	TPV_OID 
<input type="checkbox"/>	CANTIDAD
<input type="checkbox"/>	PRECIOCOSTE
<input type="checkbox"/>	PRECIOCOSTEPORUNIDAD
<input type="checkbox"/>	PRECIOPORUNIDAD
<input type="checkbox"/>	PRODUCTO_NUMERO 
<input type="checkbox"/>	IMPORTE
<input type="checkbox"/>	IMPORTE2
<input type="checkbox"/>	NOMBREPRODUCTO
<input type="checkbox"/>	PRECIOAUX

Ilustración 13 ProductoPara

- **Cantidad:** Es un entero que nos definirá la cantidad del producto que se venderá o en caso de pedido la que se añadirá.
- **Precio Coste:** Es el BigDecimal que nos informa de la suma de los precios de coste de los productos seleccionados.
- **Precio Coste por Unidad:** Es el precio de coste del artículo seleccionado.
- **Precio Aux:** Es el BigDecimal que nos informa de la suma de los precios de los productos seleccionados.
- **Precio por Unidad:** es el precio base del producto por unidad.
- **Importe:** Es la multiplicación del precio por unidad por la cantidad.
- **Importe2:** Es la multiplicación del precio de coste por unidad por la cantidad.
- **Nombre producto:** Es un String que contiene el nombre del producto.
- **Producto_Numero:** Es la id del producto seleccionado.

C.4- Diseño Arquitectónico

Para ayudar a tener un buen diseño de software y que la aplicación sea más fácil de mantener se usa el patrón de diseño arquitectónico MVC.

Modelo-Vista-Controlador

Este patrón se basa en tres partes por separado, que al juntarse hacen que la aplicación sea robusta. Cada parte se ocupa de un campo distinto, Modelo, esta parte se ocupa de los datos, de la lógica del negocio y de los mecanismos de persistencia. Vista, es la parte que se encarga de la interfaz de usuario y Controlador, es la parte que actúa de intermediario entre las otras dos partes gestionando su flujo de datos.(1)

Modelo

Esta compuesta por las clases y entidades del proyecto OpenXava, y la Base de datos, en mi caso HSQLDB que es la nativa de java. Las clases y entidades serán las que se encargarán de crear los objetos y gestionarlos para realizar las acciones correspondientes.

Vista

Es la interfaz de usuario que se encarga de la interacción con el usuario, esta debe ser agradable e intuitiva. Gracias a OpenXava se autogenera la interfaz, provocando que cumpla los requisitos de usabilidad.

Controlador

Es la parte que se encarga de la comunicación de la interfaz con el modelo, en nuestro caso se trata del Tomcat de Apache.

C.5- Diseño de Interfaz

Dado que la interfaz se autogenera con el propio OpenXava, lo único que he valorado hacer ha sido poner los colores corporativos de la tienda ya que modificar la interfaz supondría un aumento de trabajo muy significativo y al tratarse de una aplicación en la que pesa más la funcionalidad que la decoración se ha considerado que dicha modificación no era relevante.

Para cambiar los colores lo que hay que hacer es crear una hoja de estilos con los colores y la organización que se quiere conseguir.

Primero, para saber qué colores son los adecuados es necesario abrir un editor de fotos para obtener los colores en valor Hexadecimal:



Ilustración 14 Colores Para hoja de estilos

Para la hoja de estilos he usado 3 tonalidades del logotipo de la tienda, y al final ha quedado esta hoja de estilos:

```

1 @import 'base.css';
2
3 :root {
4   --formes: #C2D7D2;
5   --light-formes: #E2ECEB;
6   --dark-formes: #033649;
7   --transparent-formes: #E2ECEB;
8   --acqua: #036564;
9   --acqua-profonda: #A4C3BB;
10
11   /* Modules left menu */
12   --modules-list-color: var(--light-formes);
13   --modules-list-background: var(--acqua-profonda);
14   --modules-list-selected-color: white;
15   --modules-list-selected-border-right: red;
16
17   /* Module header */
18   --module-header-color: white;
19   --module-header-background: var(--acqua-profonda);
20
21   /* Button bar */
22   --button-bar-background: var(--formes);
23   --list-formats-color: var(--light-formes);
24   --list-formats-hover-color: white;
25
26   /* Backgrounds & labels */
27   --background: var(--light-formes);
28   --color: var(--dark-formes);
29
30   /* Frames */
31   --frame-background: var(--transparent-formes);
32
33   /* Actions */
34   --action-color: var(--acqua);
35   --action-hover-color: var(--acqua-profonda);
36   --action-hover-background: white;
37   --default-action-button-background: var(--acqua-profonda);
38
39   /* Editors */
40   --required-editor-border: var(--formes);
41 }
42 }
43

```

Ilustración 15 Hoja de Estilo

Apendice D

DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

D.1 Introducción

Al ser una herramienta low code, como usuario no tengo acceso a todas las clases o directorios que tiene el proyecto. Así pues, solo serán documentados aquellos en los cuales sí tenga acceso y aquellos en los cuales haya tenido que desarrollar la aplicación

D.2 Estructura de directorios

En mi repositorio de github hay cuatro directorios principales:

- **Formes:** es el directorio que contiene todo el proyecto, clases, librerías, archivos, etc. Este es el directorio en el que he trabajado, es donde he tenido que crear el proyecto, los paquetes y las clases necesarias para sacar la aplicación adelante.
- **OpenXava:** Es el directorio en el cual están los entresijos de la herramienta LowCode. En este directorio están las clases internas para la gestión de la interfaz, para las acciones por defecto y para todas esas utilidades que te tienes no gestionamos nosotros al trabajar con una herramienta de tipo low Code
- **Doc:** Contiene la documentación del proyecto, la memoria y los anexos.
- **Tomcat:** La aplicación se puede desplegar desde el propio OpenXava Studio, pero sería en local, así que en este directorio estará la configuración para el despliegue web.

Formes:

En este directorio como ya se ha mencionado, estarán las carpetas que son personalizables por el usuario.

- SRC: Dentro tendrá los cuatro paquetes que hemos creado y programado nosotros.
 - Actions: Será el paquete donde meteremos las acciones personalizadas.
 - Calculadores: En este paquete meteremos las clases que hacen de calculador para diferentes campos de las entidades.
 - Modelo: Es el paquete más importante. Este es el paquete que contiene todas nuestras clases.
 - Util: Este paquete es el menos utilizado y es el que contiene los campos que tienen que ser por defecto. En nuestro caso solo estará el I.V.A. por defecto.
 - _run: Este paquete es el que nos viene por defecto y es el que contienen el *main*, y permite la ejecución de la aplicación y el encendido de la base de datos.

En esta carpeta también encontramos la plantilla del ticket que hemos creado y la foto que necesita dicho ticket.

- I18n: Es la carpeta de internacionalización y cambio de las etiquetas mostradas por pantalla. En mi caso lo uso para el segundo cometido.

El resto de paquetes o archivos vienen dado por defecto. Aunque pueden ser modificados como he hecho por ejemplo con el estilo de la aplicación para que tenga los colores corporativos.

D.2 Manual del programador

OpenXava - Descarga e Instalación

Para empezar la descarga iremos a su página oficial (www.openxava.org) y pulsaremos el botón de descargar.



Ilustración 16 Página principal OpenXava

El botón se va actualizando constantemente ya que los desarrolladores de OpenXava aseguran un mínimo de 6 actualizaciones por año. La mayoría son actualizaciones de mantenimiento para corregir posibles bugs o añadir pequeñas funcionalidades propuestas por la comunidad. Una vez pulsado el botón de descarga nos llevará a un formulario que debemos rellenar para pedir a los desarrolladores que nos envíen el link para realizar la descarga de la última versión.

Descarga OpenXava

Rellena los datos de abajo y recibirás un correo electrónico con un vínculo para descargar OpenXava

Correo electrónico

Nombre

Apellidos

Permisos de contacto

El equipo de OpenXava usará la información que usted proporcione en este formulario para estar en contacto con usted y para enviarle actualizaciones sobre OpenXava y XavaPro. Por favor, háganos saber que tipo de comunicaciones le gustaría recibir:

☐ Noticias relacionadas con OpenXava

Puede cambiar de opinión en cualquier momento haciendo clic en el enlace desuscribir que hay en el pie de página de cualquier correo electrónico que reciba de nuestra parte, o poniéndose en contacto con nosotros al info@openxava.org. Trataremos su información con respeto. Para obtener más información acerca de nuestras prácticas de privacidad, visite nuestro sitio web. Al hacer clic a continuación, acepta que podamos procesar su información de acuerdo con estos términos.

 We use Mailchimp as our marketing platform. By clicking below to subscribe, you acknowledge that your information will be transferred to Mailchimp for processing. [Learn more about Mailchimp's privacy practices here.](#)

Suscribirse

Ilustración 17 Formulario para la descarga

Una vez recibido el correo de respuesta, que es casi automático, nos da la opción de descargar la última versión en el Sistema Operativo deseado. En mi caso, para este proyecto, será en Windows. La última versión del framework es openxava-6.4.2.

El entorno para programar con Openxava está basado en Eclipse como podemos ver en su logotipo. Y se trata de OpenXava Studio











Nombre	Estado	Fecha de modificación	Tipo	Tamaño
 doc		09/11/2020 13:54	Carpeta de archivos	
 studio		09/11/2020 13:33	Carpeta de archivos	
 workspace		09/11/2020 13:42	Carpeta de archivos	
 openxava		09/11/2020 13:28	Aplicación	417 KB
 openxava		09/11/2020 13:28	Opciones de confi...	1 KB

Ilustración 18 Contenido del Zip descargado

Y al abrirlo nos encontramos con la misma interfaz que Eclipse

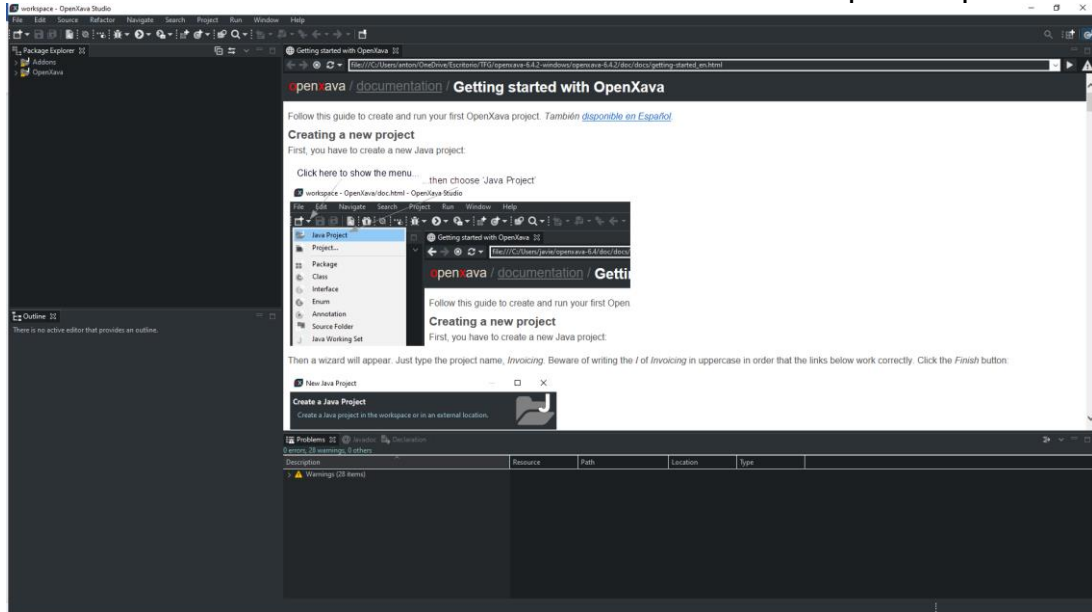


Ilustración 19 Apariencia

Para empezar, lo primero que he realizado, ha sido cambiar el entorno a un entorno más claro. Para cambiarlo *Window -> Preference -> DevStyle* y seleccionamos uno más claro.

Una vez configurado a nuestro gusto podemos empezar, para ello creamos un nuevo proyecto:

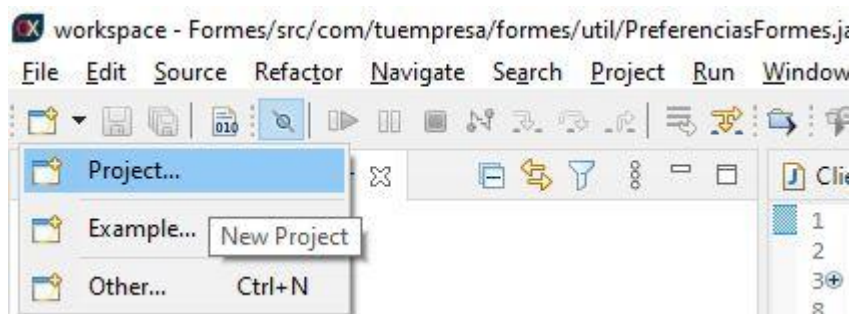


Ilustración 20 Nuevo Proyecto

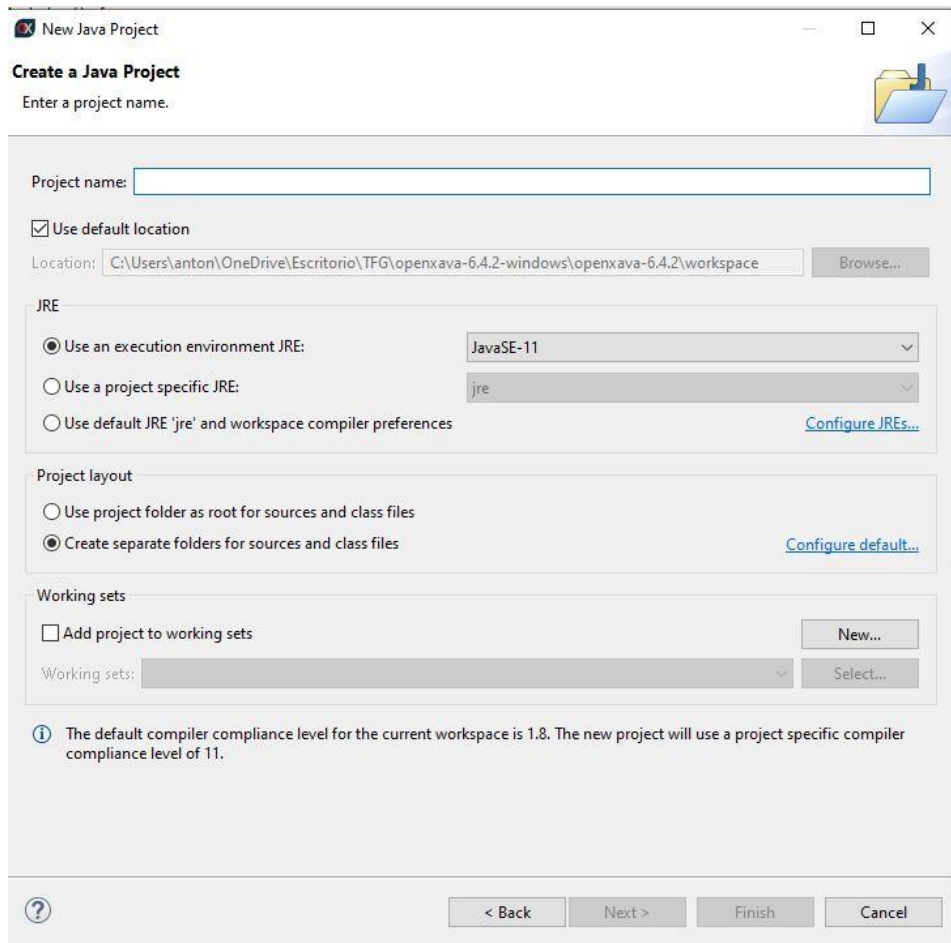


Ilustración 21 Creación nuevo proyecto

Le ponemos el nombre que deseamos y le damos a finalizar. Una vez tenemos el proyecto creado, vamos a crearlo, pero en versión openxava. Para ello, pulsamos y nos saldrá una ventana donde tendremos que poner el nombre del proyecto que hemos creado anteriormente para transformarlo en un proyecto OpenXava.

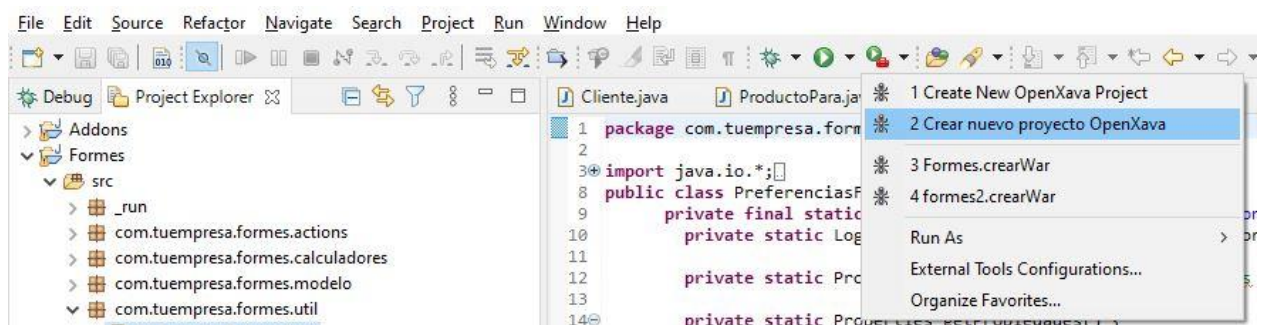


Ilustración 22 Creación del proyecto Openxava

Cuando ya tenemos el proyecto en modo openxava, ya se trata de un proyecto estándar de java y vamos añadiendo las clases que necesitamos para la realización de nuestra práctica.

Compilación y ejecución desde Openxava studio

Desde el mismo entorno, se puede ejecutar el proyecto y desplegarlo en modo local así como visualizar el manager de la base de datos. Para hacerlo hay que compilar el proyecto es como en eclipse, el botón que hay al lado de guardar, si hay algún error de sintaxis o de programación saldrá en la compilación. Si todo es correcto podemos proceder a la ejecución para ello vamos a la clase `_Run_formes` dentro del paquete `_run`, le damos *click* derecho y Run AS -> Java Application

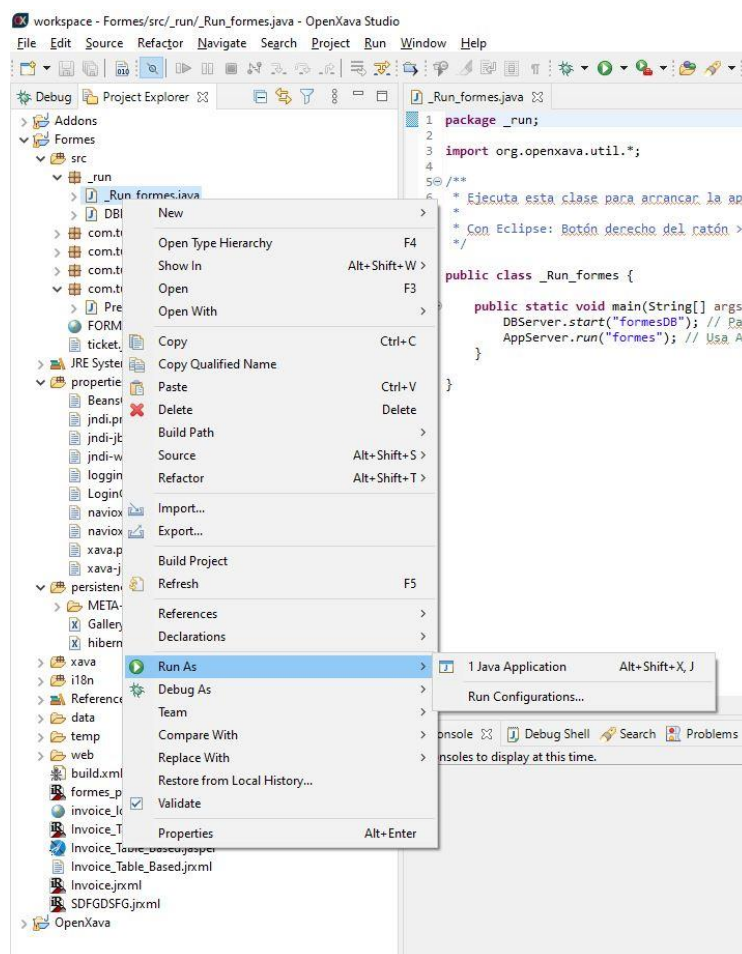


Ilustración 23 Ejecución de la aplicación

En la consola de eclipse se nos desplegará la traza del proyecto con diferente información útil. Y si no ha habido errores nos dará el siguiente mensaje:

“Aplicación iniciada. Ve a <http://localhost:8080/formes>”

Esto implicará que nuestra aplicación habrá sido lanzada en modo local, en caso de querer ver el manager de la base de datos ejecutamos los mismos pasos que en este caso, pero en la clase DB_Manager del paquete _run. Cabe destacar que el orden es importante. Primero tenemos que inicializar el proyecto para que abra la base de datos y después ejecutar el manager para así ver la base de datos.

La interfaz de la base de datos es la interfaz de HSQLDB que es el gestor de bases de datos nativo de java. Que luce así:

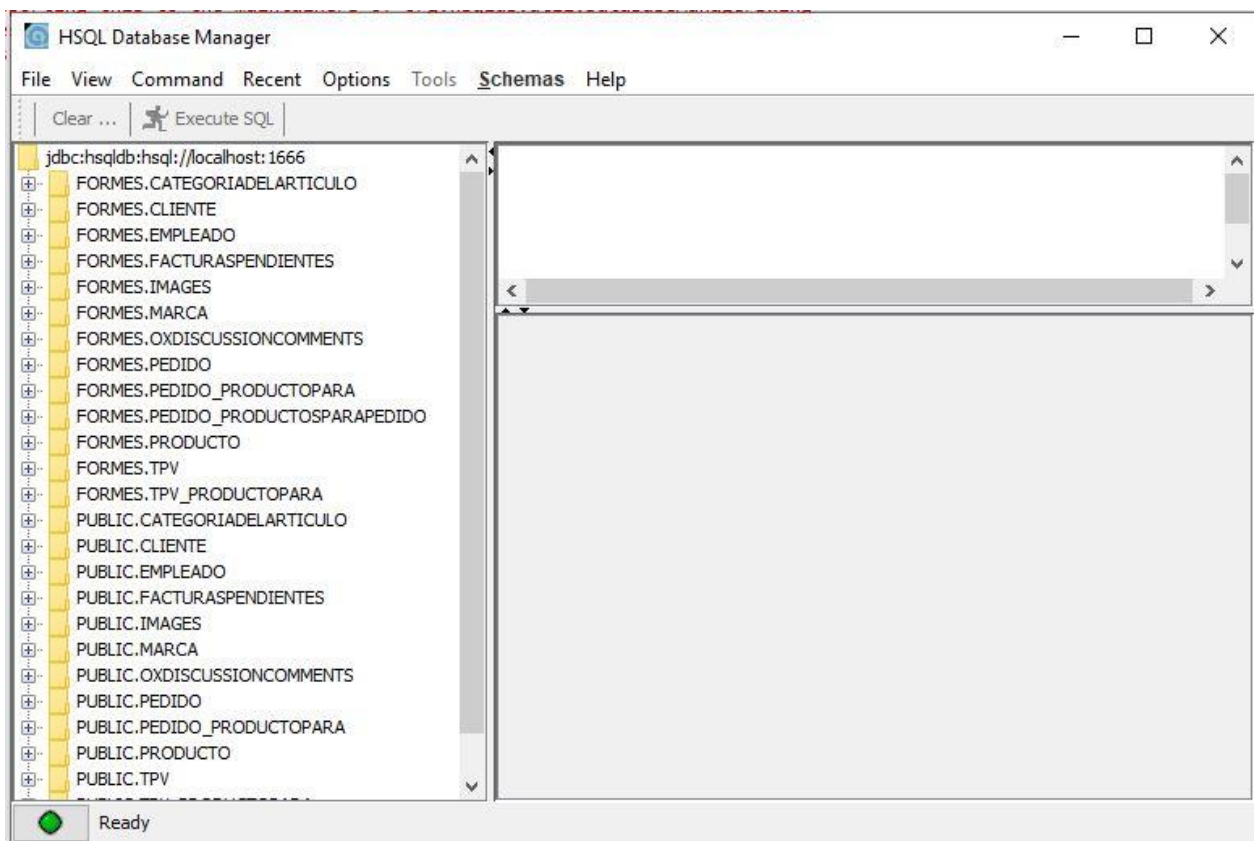


Ilustración 24 Apariencia del gestor de la base de datos

Compilación de la aplicación para despliegue con Tomcat

Para poder desplegar la aplicación fuera del entorno de programación el propio openxava nos da varias opciones, sobre todo con la elección de la base de datos. En nuestro caso hemos usado la de por defecto, pero tienes la opción de usar otras bases de datos como son MySQL, PostGress, MariaDB, Oracle entre otras. Los únicos archivos que necesitaras modificar son el context.xml y el persistence.xml de tu proyecto. Para darles la ruta a la base de datos a ejecutar.

Para empezar nuestro despliegue nos tenemos que situar en el proyecto y desde el entorno y en el archivo build.xml dar al *click* derecho y ejecutarlo como "ant build...". A continuación se nos abrirá una ventana:

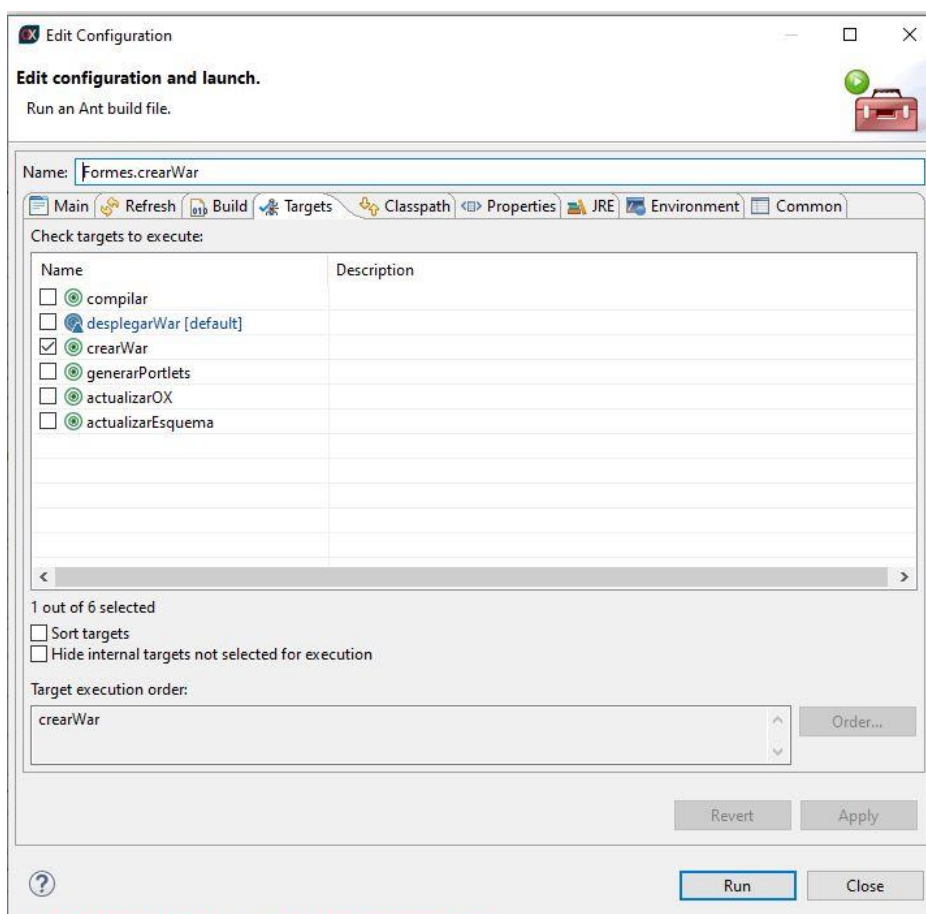


Ilustración 25 Crear archivo.war

Le ponemos el nombre de nuestro proyecto "formes.crearWar" y le damos a run. Se nos estará compilando en la consola.

Una vez la "construcción" sea correcta nos habrá creado un directorio en nuestra carpeta de openxava que será workspace.dist, donde tendremos nuestra aplicación compilada.

Instalación TomCat

Para la instalación del tomcat nos dirigiremos a su pagina oficial ya que es una herramienta gratuita.

(<https://tomcat.apache.org/download-90.cgi>)

Bajamos el .zip y lo descomprimos en el directorio que queramos. Una vez dentro lo que tenemos que hacer es dirigirnos a la carpeta webapps y borrar todas las que vienen por defecto.

Ahora añadiremos nuestro proyecto compilado que se encuentra en la carpeta workspace.dist y copiamos el archivo formes.war.

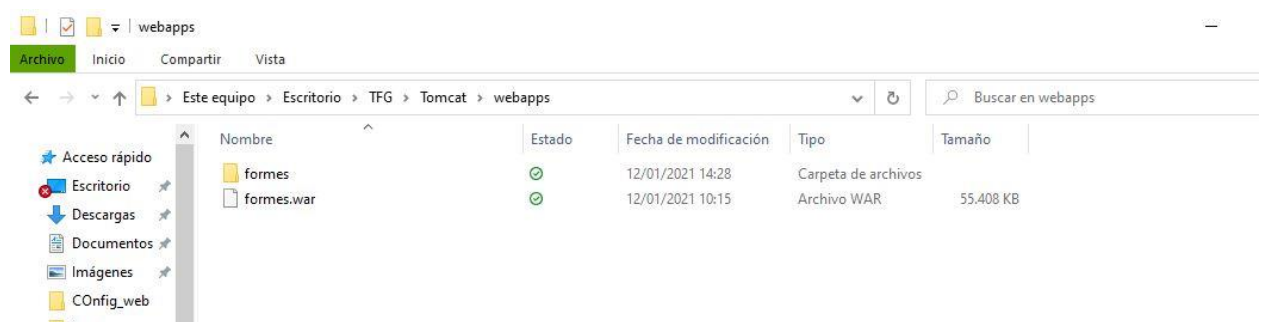


Ilustración 26 Carpeta webapps

El siguiente paso es copiar el archivo hsqldb.jar de la carpeta de openxava/lib a la librería del tomcat.

Una vez lo tenemos copiado ya podemos proceder al lanzamiento.

Despliegue mediante Tomcat

Para el despliegue lo primero de todo será inicializar nuestra base de datos, para ello abrimos una terminal (CMD) y navegamos hasta el directorio donde tenemos la carpeta lib.

En mi caso:

```
C:\Users\anton>cd OneDrive\Escritorio\TFG\Tomcat\lib
```

Una vez dentro ejecutamos la siguiente instrucción para iniciar la base de datos:

```
java -classpath hsqldb.jar org.hsqldb.Server -database mi-database-  
db -port 1666 -silent true -trace false
```

El puerto escogido es el de por defecto 1666. Ahora que tenemos la base de datos inicializada, procedemos a lanzar el despliegue del tomcat. Para ello abrimos otra terminal y nos dirigimos al directorio *bin* del tomcat:

```
C:\Users\anton\OneDrive\Escritorio\TFG\Tomcat\bin>
```

Definimos la variable JAVA_HOME a nuestro java instalado en el pc:

```
set JAVA_HOME=C:\Program Files\Java\jdk-11.0.8
```

Y para finalizar ejecutamos la siguiente instrucción:

```
C:\Users\anton\OneDrive\Escritorio\TFG\Tomcat\bin>startup
```

Se nos abrirá la consola del tomcat donde veremos la interacción de los usuarios con la aplicación. La aplicación queda abierta en el puerto por defecto 8080. Si nos dirigimos a él podemos ver la página inicial de la aplicación.

Conexión con un dominio

Para que la conexión no sea solo en local y se pueda usar la aplicación desde cualquier parte del mundo, he hecho el dominio apunte a la dirección de mi pc.

“ www.formestallesgrans.com/formes”












Para no tener que montar un servidor que esté las 24 horas abierto lo que he hecho ha sido abrir el puerto del *router* 8080 y redirigir el puerto 80 al que acabamos de abrir 8080. Como con los servicios tradicionales solo hay una ip pública y hay que escoger a qué ip se redirigirá a cada petición del exterior. En mi caso he puesto la ip de mi pc. Que con una configuración dhcp será siempre la misma ip.

Nombre <input type="text" value="http"/>	Protocolo <input type="text" value="TCP"/>	Puerto/Rango Externo <input type="text" value="80:80"/>	Puerto/Rango Interno <input type="text" value="8080:8080"/>	Dirección IP <input type="text" value="192.168.1.65"/>	Activar <input checked="checked" type="checkbox"/>
---	---	--	--	---	---

Ilustración 27 Ip y Redirección al puerto

Una vez configurado este paso, nos creamos una cuenta en NO-IP que es un servicio que ofrecen los routers convencionales, que nos proporciona un subdominio de los suyos en este caso (formestallesgrans.ddns.net). Nos permite que, dado que la naturaleza del servicio de telefonía se basa en ip's dinámicas, al cambiar de ip publica gracias al subdominio, no tengamos que ir a nuestro DNS a cambiar cada vez la ip.

Y finalmente iremos a nuestro proveedor de dominio, ionos en mi caso, y haremos que el dominio que hemos adquirido apunte al subdominio que nos ha proporcionado NO-IP.

Añadir registro					
<input type="checkbox"/>	TIPO	NOMBRE DE HOST	VALOR	SERVICIO ▲	ACCIONES
<input type="checkbox"/>	CNAME	_domainconnect	_domainconnect.1and1.com	Domain Connect	
<input type="checkbox"/>	MX	@	mx00.ionos.es	Mail	
<input type="checkbox"/>	MX	@	mx01.ionos.es	Mail	
<input type="checkbox"/>	CNAME	autodiscover	adsredir.ionos.info	Mail	 
<input type="checkbox"/>	A	@	217.160.0.138	Redireccionamiento...	
<input type="checkbox"/>	AAAA	@	2001:8d8:100f:f000:0:0:21b	Redireccionamiento...	
<input type="checkbox"/>	TXT	_dep_ws_mutex	"0b728b4a0d1fe99ff48d92f92eb735a99e8e8f25...	Redireccionamiento...	 
<input type="checkbox"/>	CNAME	www	formestallesgrans.ddns.net	-	 

Apéndice E

DOCUMENTACIÓN DE USUARIO

E.1 Introducción

En este anexo vamos a explicar cómo utilizar nuestra aplicación por parte de un usuario.

E.2 Requisitos de usuario

Al tratarse de una aplicación web, el usuario bastará que tenga un navegador actualizado con las ventanas emergentes desbloqueadas.

Para acceder a la aplicación, el usuario bastará que busque el siguiente dominio: www.formestallesgrans.com/formes que le redirigirá a la aplicación que esta “alojada” en mi pc a modo de servidor provisional

E.3 Instalación

Al ser una aplicación web no hace falta instalarla, solamente se requiere usar un navegador y una conexión a internet.

E.4 Manual de usuario

Todos los datos que saldrán en el siguiente manual son datos inventados para no violar la LOPD.

Inicio

Esta es la primera página que se verá al entrar en la aplicación. En ella podemos ver el nombre y los colores corporativos, a la vez que un usuario y una contraseña facilitadas para que el tribunal haga las pruebas pertinentes.

Cabe destacar que se trata de una aplicación pensada para el uso en un solo terminal y con solo un usuario a la vez. De ese modo, a la hora de realizar pruebas, si se realizan simultáneamente en más de un terminal puede haber problemas de persistencia en la base de datos.



Ilustración 29 Inicio de la aplicación

Login

La siguiente página es aquella destinada a iniciar sesión. Con OpenXava free, solo nos permite acceder a la aplicación si previamente estás introducido en su código fuente, con la versión de pago sí que nos permite hacer registros de usuarios y control de accesos.




Ilustración 30 Login de usuario

El usuario para pruebas y para ver la aplicación es: tribunal y su contraseña Tribunal.

En caso de que intente entrar algún usuario sin cuenta nos saltá el error de usuario no encontrado.



Ilustración 31 Usuario no encontrado

Una vez en la aplicación

El usuario podrá navegar por el menú de las diferentes opciones que tiene y podrá elegir en qué pestaña desea colocarse.

Como podemos observar cada usuario puede personalizar sus páginas favoritas en función de las que más vaya a usar.



Ilustración 32 Menu

La navegación entre pestañas es muy cómoda ya que el menú siempre está visible a la izquierda de la pantalla.

Interactuar con las clases

La interacción base es la misma en todas las pestañas, al principio entraremos en la vista modo lista, como podemos ver en la ilustración 33

En caso de quererlo ver de diferente manera la aplicación nos permite la visualización por módulos.



Ilustración 34 Clientes Modo Modulos

En cada pestaña contamos con un submenú con las acciones por defecto de la aplicación:

- Nuevo: es el botón que nos enviará a la página para crear un nuevo objeto de la clase, en caso de cliente nos enviará a la creación de un cliente nuevo.
- Generar PDF: Nos imprime un PDF con el listado de todos los clientes con los datos que queremos.
- Generar Excel: Nos genera una hoja de Excel con todos los campos del objeto.
- Importar datos: En caso de que tengamos una fuente externa de datos nos permite añadirlos.

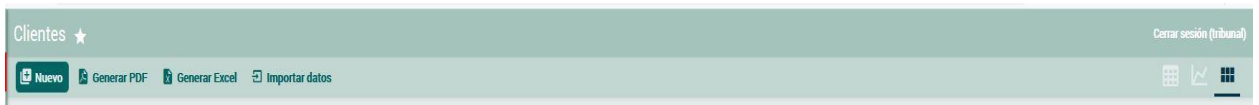


Ilustración 35 Submenu comun entidades

Nuevo Objeto “caso cliente”

Al acceder a la creación de un nuevo cliente nos aparece una pantalla con los campos a rellenar y con otro submenú con las acciones posibles.

The image shows a web form for creating a new client. On the left is a sidebar with a menu under 'Formes Moda' containing 'Clientes', 'Productos', 'Tpv', 'Pedidos', 'Categoria del Artículo', 'Empleado', 'Marca', and 'Facturas pendientes'. The 'Clientes' item is selected. The main area has a header with 'Clientes' and a star icon, and a sub-menu with 'Lista', 'Nuevo', 'Grabar', and 'Refrescar'. The form fields are grouped under 'Dirección' and include: 'Vía pública', 'Código postal', 'Municipio', 'Provincia', 'Id*', 'Teléfono*', 'Nombre*', 'CIF', and 'Email*'. A 'GRABAR' button is at the bottom.

Ilustración 36 Nuevo cliente

Una vez los campos estén rellenos correctamente procederemos al guardado en la base de datos. En caso de que algún campo no esté correcto nos saldrá un mensaje de error explicándonos qué es en lo que hemos fallado. Por ejemplo, en no poner ningún dato:

The screenshot shows a web application interface for managing clients. On the left is a sidebar with navigation links: 'Formes Moda', 'Clientes ★', 'Productos ★', 'Tpv ★', 'Pedidos ★', 'Categoría del Artículo', 'Empleado', 'Marca', and 'Facturas pendientes'. The main area is titled 'Clientes ★' and contains a form with the following fields: 'Dirección' (expanded), 'Via pública', 'Código postal', 'Municipio', 'Provincia', 'Id*', 'Teléfono*', 'Nombre*', 'CIF', and 'Email*'. A 'GRABAR' button is located below the 'Email*' field. On the right side of the form, there are five red error messages, each with a close button (X):

- Es obligado que CIF en Cliente tenga valor
- Es obligado que Número en Cliente tenga valor
- Es obligado que Teléfono en Cliente tenga valor
- Es obligado que Nombre en Cliente tenga valor
- Es obligado que Correo electrónico en Cliente tenga valor

Ilustración 37 Error en los campos

Nos han saltado los errores de que los campos obligatorios están vacíos.

Gracias a que la aplicación es muy intuitiva, casi todas las clases funcionan de la misma forma, así que solo faltaría explicar entidad principal que es la de TPV y la entidad pedido que es la única diferente.

Entidad TPV

Esta entidad es el corazón de la aplicación y en la que el usuario más tiempo pasará, ya a primera vista en modo lista es muy parecida a las demás, pero es al entrar a la creación de una nueva venta donde vemos que es una pestaña mucho más completa.

Ilustración 38 Entidad TPV

En ella podemos observar todos los campos y otras entidades que se mezclan en esta.

Para la elección del cliente, empleado y los productos podemos situarnos encima de cada una de las lupas para buscar en la base de datos.

Ilustración 39 Elección del empleado

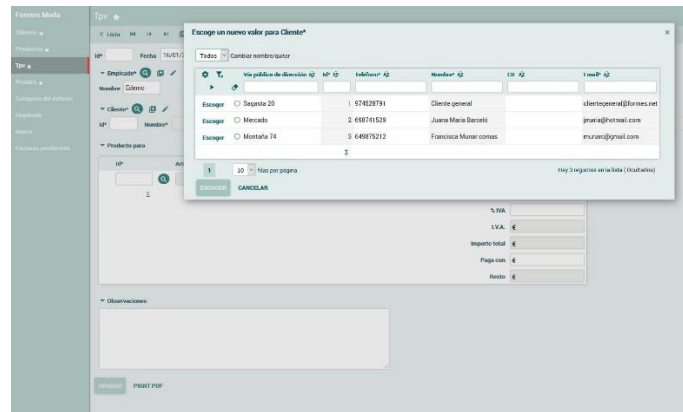


Ilustración 40 Elección del cliente

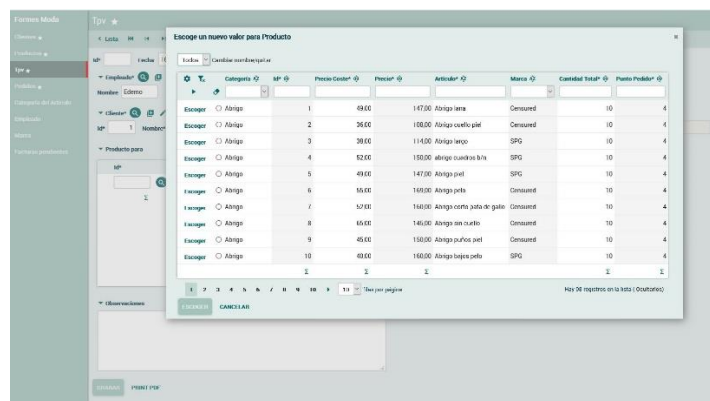


Ilustración 41 Elección de los productos

Una vez escogido cada producto, hay que definir la cantidad que se va a llevar el cliente y veremos cambiado el precio. Así mismo se tiene que definir el I.V.A. que se va a aplicar a la factura.

Otra funcionalidad es que se puede introducir el efectivo recibido y se calcula el resto a devolver automáticamente.

Ilustración 42 Tpv Rellenado

Impresión ticket

En caso de querer imprimir la factura/ticket bastaría pulsar el botón "Print pdf" que nos generará un documento en otra ventana.

Ilustración 43 Ticket

Guardar venta

La impresión del ticket no guarda la venta en la base de datos, para ello se tiene que volver a la aplicación y guardarla mediante el botón de grabar.

Al grabar una venta el stock de los productos se verá reducido en el numero que se haya vendido de manera automática. En caso de poner una cantidad mayor a la que se tiene en stock saltará el mensaje *Stock insuficiente*.

Otra funcionalidad de la acción de grabar es que, si un producto supera inferiormente a su punto pedido, este producto se añadirá al pedido que esté abierto para su posterior pedido al proveedor.

Pedido

La entidad “pedido” es otra de las entidades peculiares ya que solo puede haber un pedido abierto al mismo tiempo y es responsabilidad del usuario que así sea.

Para crear un pedido basta crearlo como las otras entidades, pero vacío, es decir, sin productos.

Ya que se irá rellenando automáticamente. En caso de que el usuario cierre un pedido, tiene que abrir uno nuevo para que se vayan añadiendo de manera correcta.

The screenshot shows a web application interface for creating a new order. At the top, there's a header bar with the title 'Pedidos' and a star icon. Below it, a navigation bar contains links: '< Lista', '«', '»', 'Nuevo' (with a plus icon), 'Grabar' (with a save icon), 'Refrescar' (with a refresh icon), and 'Borrar' (with a delete icon). The main form area has a section for 'Id' with a text input and a checkbox labeled 'Está pedido'. Below this is a section titled 'Producto para' with a dropdown arrow. Underneath, there's a table with columns: 'Id*', 'Artículo*', 'Cantidad', 'Precio coste por unidad', and 'Importe'. The table is currently empty, showing only summary rows with 'Σ' symbols and values of '0,00'. At the bottom right of the table, there's a row for 'Importe coste total' with a value of '0,00'. A green 'GRABAR' button is located at the bottom left of the form.

Ilustración 44 Pedido Vacío

E.5- Conclusiones manual de usuario

He buscado una herramienta que permitiese tener una interfaz muy fácil de usar y aprender para que no hubiese problemas a la hora de su uso.

Gracias a OpenXava que nos permite crear muchas entidades con comportamientos similares y una interfaz muy limpia.

Apendice F

BIBLIOGRAFÍA

1. Modelo vista controlador (MVC) [Internet]. [citado 16 de enero de 2021]. Disponible en:
<https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html>