TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Desarrollo de una app híbrida para la gestión de partes de verificación para vehículos

Mención:

Ingeniería del Software/Tecnologías de la Información

Curso 2017/2018

Alumno/a:

Juan Antonio Rodríguez Baeza

Director/es:

Rafael Guirado Clavijo

Mariano González Galdeano

Dedicado a mi madre.

Mirando al mar soñé
que estabas junto a mí;
mirando al mar, yo no sé qué sentí,
que, acordándome de ti, lloré.

La dicha que perdí,
yo sé que ha de tornar
y sé que ha de volver a mí,
cuando yo esté mirando al mar.

Agradecimientos

Me gustaría hacer una reflexión previa a esta carta para darle aún más solemnidad a las palabras que siguen a continuación.

Era 2011, estaba rodeado de trabajos precarios en periodos estivales, y una visión de futuro bastante desmotivada, con 28 años, sin estudios, ni trabajo y viviendo en casa de mis padres, y en ese invierno larguísimo a la vista de los huecos entre trabajo de verano y trabajo de verano me planteé terminar mi bachillerato con tal de tener la cabeza ocupada durante el invierno, así comienza mi personal historia de superación, con el incondicional apoyo de mi madre, a la que esté donde esté, nunca tendré tiempo para agradecerle el empeño que puso en mí, y a la que dedico este TFG cuya importancia radica en la propia sorpresa de llegar a la oportunidad misma de hacerlo.

Siguiendo con mi historia, pero avanzando un poco en el tiempo, tras un muy fructífero FP, en 2014 se me plantea la oportunidad irrepetible de empezar una carrera a 600 kilómetros de mi casa, cruzando de cabo a rabo la tierra de las inoportunidades, me vine a Almería. Este primer año resultó más una prueba de fuerza de voluntad que un filtro de conocimientos por parte del centro educativo, tras la pérdida de mi madre en mitad del primer cuatrimestre, recogió el testigo mi compañera, amiga, amor de mi vida, Inma. Si me faltan palabras para describir el apoyo de mi madre, el apoyo y empujón emocional que me dio Inma en el peor momento de mi vida solo puedo compensarlo con un profundo agradecimiento y dedicación durante el resto de mi vida.

En ese punto de la historia me encontraba como digo en el peor momento de mi vida, comenzando el proyecto más titánico al que me había enfrentado, no tardé en conocer a los que hoy considero mis compañeros no de clase, sino de camino, Jorge, Diego, Juan y Aitor. Sin ellos en ese oscuro momento, la realización de este trabajo no hubiera sido posible hoy, por eso les doy igualmente las gracias. Sin olvidar a compañeros con los que he ido sorteando los obstáculos/asignaturas por el camino, Nikita, Jesús, Mateo, y a mi compañero becario Juan Alberto Llopis por supuesto, a todos ellos también gracias.

En este tiempo en Almería no solo he encontrado buenas personas en la universidad, también tengo que acordarme de Gema y Manolo y por supuesto de Emma a la que tarde o temprano le comeré un pie. Y de Cristóbal que me ha echado una buena mano.

Y como no agradecer también a personas con una dedicación por su trabajo que me resulta inspiradora, quienes lean estas líneas y conozcan a Miguel Ángel Navarro Pascual entenderán de que hablo. Los mismos motivos me llevan a nombrar a Antonio Corral, Antonio Becerra, Paco Gil, Manolo Torres, Pilar, Gloria, José Antonio Bermejo, Juan Fco Sanjuán, Rosa Mª Ayala, Isabel Mª del Águila, Rafael Guirado mi director de TFG. Profesores con los que he compartido buena parte de este camino, muchas gracias a todos ellos.

Por último, he de agradecer a mis jefes del departamento de informática de Asetir, S.L.U. en el grupo JCarrion, por darme la oportunidad de cursar la Beca D-UAL con ellos, y de hacerme partícipe, tanto a mi como a mi compañero, del día a día del departamento desde el minuto uno, además de darme la oportunidad y la confianza de realizar este proyecto con ellos.

Índice general

Índice de ilustraciones

Índice de tablas

Al encontrarme ejerciendo una Beca D-UAL[1] con una temporización bastante larga, se me ofreció realizar el Trabajo Fin de Grado como parte de esta, desarrollando un proyecto interno de la empresa en la cual efectúo dicha beca.

> *"…La estancia en la empresa se divide en dos periodos:*
>
> *El primero abarca la enseñanza oficial, que se reconocen en términos de créditos de las asignaturas oficiales, en las prácticas curriculares y en la posibilidad de realización del trabajo fin estudios;*
>
> *Y un segundo periodo. en el que se realiza una práctica extracurricular (o contrato en prácticas), que la dota de mayor flexibilidad en la integración en la empresa o entidad. Ambas acciones se realizan en una empresa o entidad bajo la supervisión de tutores académicos, que designan los centros responsables de los títulos, como por profesionales de las organizaciones participantes…"*

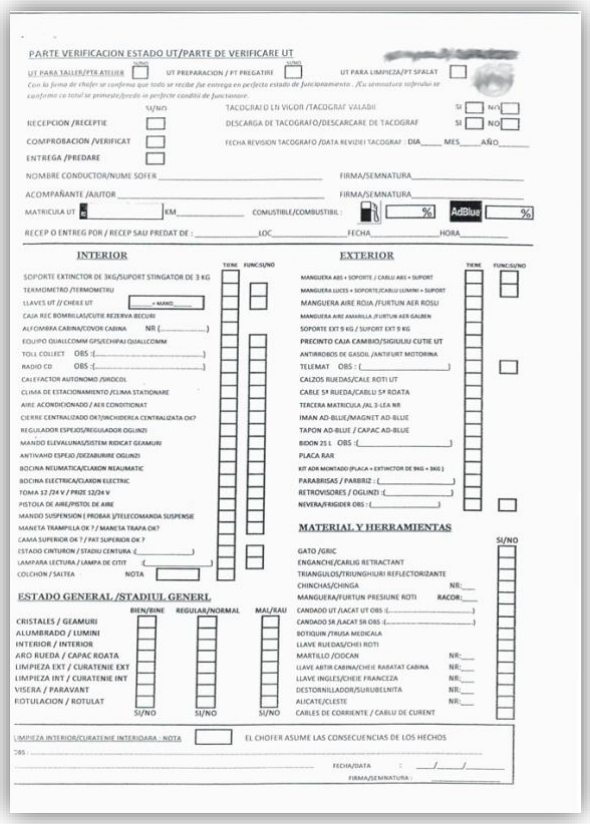
A lo largo de este apartado se describe de una manera introductoria el planteamiento teórico del proyecto. Además de hacer un escueto razonamiento sobre el estado del desarrollo de aplicaciones móviles, y el "porque" del llamado Desarrollo Híbrido.

1. Introducción

La empresa es realmente un grupo de empresas, la actividad principal es el transporte de mercancías por carretera, en ese contexto se han constituido otras empresas agregadas al grupo para cubrir otras actividades que se hicieron necesarias. En mi caso, me encuentro, dentro del grupo, en la empresa que contiene, entre otros, el departamento de informática, entre cuyas tareas está el desarrollo de nuevo software que venga a facilitar el día a día realizado por el resto de los trabajadores del grupo empresarial.

Los vehículos de la flota realizan largos viajes internacionales, por lo que sufren un severo desgaste, por ejemplo, en neumáticos, además de estar expuestos a cualquier desperfecto, rotura, golpe, etc. El control del estado de los vehículos se hacía antiguamente a través de unos partes de verificación en papel que rellenaba el operario ($ver\ ilustración\ 1$), uno a la salida (viaje) de un vehículo y otro a la entrada de este. Estos partes contenían una gran cantidad de información acerca del estado de los neumáticos, frenos, niveles de combustible, aceite, estado de la carrocería, número de palés y barras de sujeción de la mercancía, etc. Cuando se rellenaban los partes correspondientes a la salida y llegada de un vehículo en un viaje, se unían para tener una visión global y comparativa del antes y el después del viaje.

[1] http://cms.ual.es/UAL/universidad/organosgobierno/vformacion/pagina/BECASDUAL



PARTE VERIFICACION ESTADO UT/PARTE DE VERIFICARE UT

INTERIOR

EXTERIOR

MATERIAL Y HERRAMIENTAS

ESTADO GENERAL /STADIUL GENERL

Ilustración 1 - Ejemplo de parte de verificación en papel

Con el tiempo y las nuevas tecnologías este mecanismo se informatizó, de manera que estos partes se transformaron en el módulo “Partes de Verificación” (en adelante PdV) parte de la aplicación de escritorio “Tráfico” (*ver ilustración 2*).

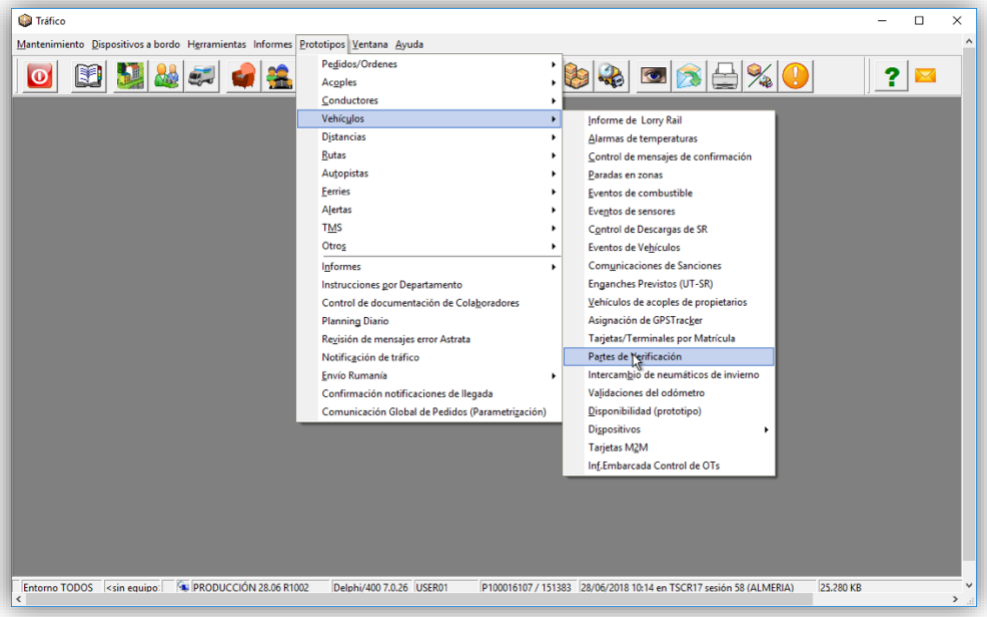


Ilustración 2 - Acceso Partes de Verificación de la aplicación Tráfico.

El objetivo de PdV es el control del estado y equipamiento de cada tractora y semirremolque controlada/os en el cambio de tripulación (*ver ilustración 3*) (registrándose un Parte de Verificación de Recepción cuando un conductor entrega el vehículo al personal de campa y otro Parte de Verificación de Entrega cuando el personal de campa entrega el vehículo al conductor asignado).

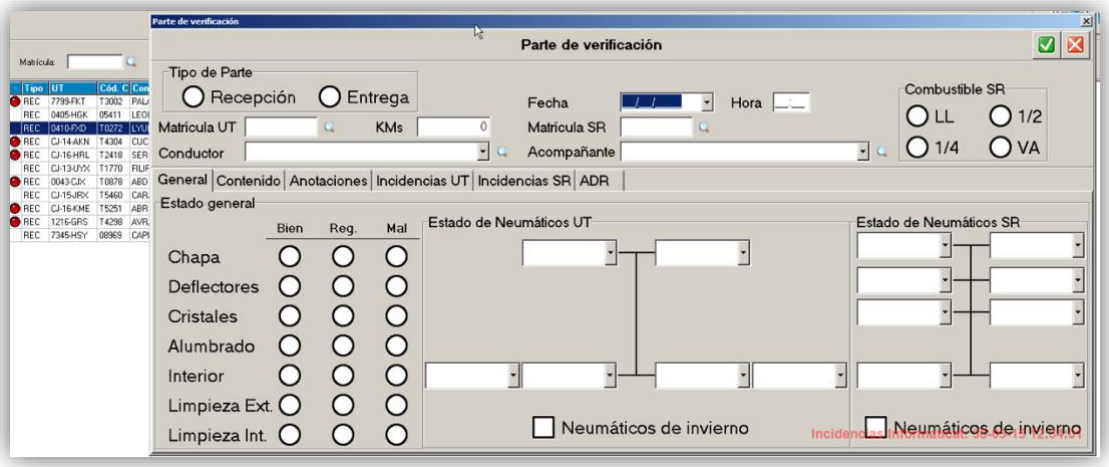


Ilustración 3 - Ejemplo de ventana para rellenar un Parte de Verificación.

El Parte de Verificación de Recepción de una UT / SR estará ligado al formulario de Parte de Verificación de Entrega último realizado a esa misma UT / SR, con el objeto de detectar las diferencias en el estado y Equipamiento del vehículo entre su última entrega y su recepción.

El uso del citado módulo PdV facilita el almacenamiento de la información de una manera mucho más estructurada, disminuyendo el riesgo de pérdida por la replicación en los sistemas informáticos propiedad de la empresa, aun así, la recogida de información continúa siendo a papel en los antiguos partes, puesto que el operario no puede llevar consigo la maquina durante la correspondiente inspección de los vehículos. Esta información debe ser volcada más tarde en PdV, lo que supone un trabajo extra, al fin y al cabo, apuntar la información en papel para luego volcarla al sistema informático.

Por lo que se me propone desarrollar una aplicación móvil destinada a ofrecer la interfaz propia de PdV directamente sobre el terreno, de manera que eliminemos el trasbordo de información del papel a la aplicación informática, de lo que se encargaría directamente la aplicación móvil al conectarla con los servicios de back-end asociados a la aplicación de escritorio. De esta forma la información será volcada directamente, en función de la conectividad del dispositivo móvil, en las bases de datos correspondientes.

1.1. Desarrollo Híbrido.

La continua y rápida evolución de las tecnologías contemporáneas hace que sea mucho más cómodo hacer uso de PdV a través de algún dispositivo móvil, como es el caso de una tableta.

Si queremos desarrollar una aplicación para dispositivos móviles tendremos que decidir para que tipo de sistema informático queremos hacerlo (a grandes rasgos iOS[2] o Android[3]).

Para desarrollar en iOS tendremos que hacer uso bien del IDE de desarrollo XCode[4] disponible para macOS a través de su App Store, o a través de su plataforma de descargas para desarrolladores a la que podemos acceder si disponemos de una ID de Apple. O bien usando algún otro IDE o herramienta que nos permita incorporar las librerías correspondientes de Apple para este propósito. En cualquier caso, el lenguaje de programación nativo para escribir programas que se ejecuten en iOS es Swift[5].

Si por el contrario decidimos desarrollar en Android, Google pone a nuestra disposición el IDE Android Studio[6] disponible para Windows, Mac y Linux a través de su plataforma de descarga abierta, sin necesidad de registro. También para esta situación existen alternativas concretas que nos permiten el desarrollo directamente de manera nativa, que en el caso de Android se trata de Java[7].

En el caso concreto de Android, como decimos el lenguaje de programación es Java, por lo que es necesario disponer de las herramientas para desarrolladores JDK[8].

Por mencionar algún ejemplo de alternativa de IDE para el desarrollo nativo de aplicaciones móviles en sendos sistemas descritos, podemos hablar de las herramientas de JetBrains[9]. Se trata de una compañía que ofrece hoy en día 22 productos diferentes, entre los que encontramos extensiones para alguna otra aplicación nativa como es el caso de Visual Studio de Microsoft, además de un total de 10 IDEs para el desarrollo especifico de uno o varios lenguajes de programación, entre los que encontramos algunos que nos interesarían para el caso que nos ocupa en este trabajo.

- AppCode10, se trata de un IDE destinado al desarrollo en Swift y Objective-C que podemos usar para realizar trabajos en iOS o en macOS.

[2] https://www.apple.com/es/ios
[3] https://www.android.com
[4] https://developer.apple.com/support/xcode/
[5] https://developer.apple.com/swift/
[6] https://developer.android.com/studio/
[7] https://www.java.com/es/
[8] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[9] http://www.jetbrains.com/company/
[10] http://www.jetbrains.com/objc/

- IntelliJ IDEA11, IDE con el que podremos desarrollar bajo el entorno JVM en Java que podemos usar para realizar trabajos en Android, o en cualquier otro entorno que se apoye en JVM.

Puede darse el caso de que no esté tan claro para qué entorno queremos desarrollar algún producto y nuestra intención es llegar al máximo número de usuarios posible, podemos buscar las estadísticas de uso[12] y decantarnos por la plataforma que cuente con más dispositivos finales, o abstraer el desarrollo del producto, a un nivel que conocemos como desarrollo de aplicaciones multidispositivo o híbridas.

El desarrollo de aplicaciones híbridas combina la estabilidad de la programación nativa con alguna otra tecnología como puede ser el desarrollo web, confeccionando un producto final con la característica de ser multidispositivo; conseguiremos de esta forma llegar a un mayor número de usuarios al ser capaces de lanzar nuestro producto indiferentemente de la plataforma de destino. Existen varias herramientas destinadas a este propósito, como es el caso de Xamarin[13] o Titanium[14].

En el contexto del desarrollo de aplicaciones híbridas, la empresa ya ha realizado algún trabajo, para lo que ha usado la plataforma XOne[15]. Esta plataforma combina el código nativo con XML, CSS, JS o VBS. No se pretende maximizar el número de usuarios finales de la aplicación que se desarrolla en este caso, lo que realmente se intenta es desarrollar sin pensar en el sistema operativo de destino, puesto que, si esta aplicación se implanta, podría ser usada en algún caso por trabajadores externos a la empresa como puede ser un taller colaborador, con lo que facilitamos su uso.

Según la documentación oficial de la compañía propietaria de la plataforma XOne, la solución es ampliamente escalable y modularizada, por lo que permite iniciar desde un producto piloto muy básico, y a través de estas características, añadir la funcionalidad requerida para posteriores versiones o evoluciones del producto, por lo que es perfecto para el desarrollo de este proyecto en su conjunto, primero como TFG y más tarde como su Complemento.

XOne pone a nuestra disposición el IDE web XoneCloud, que usaremos para el desarrollo, además de la documentación oficial a través de una wiki (Wiki XOne, 2018) y de un foro (Foro XOne, 2018). El entorno de desarrollo está fuertemente abstraído a través de XML, lo que nos permite ir dibujando la interfaz de un modo relativamente cómodo usando la siguiente estructura $(ver\ ilustración\ 4)$.

[11] http://www.jetbrains.com/idea/
[12] https://www.digital55.com/ios-vs-android-tendencias-y-cuota-de-mercado/
[13] https://visualstudio.microsoft.com/es/xamarin/
[14] https://www.appcelerator.com/Titanium/
[15] http://xoneisp.com/web/

```
<?xml version="1.0" encoding="iso-8859-15" standalone="yes"?>
<xml>
  <app prefix="" >
        <coll name="" >
              <group name="" >
                    <frame name="" >
                          <prop name="" />
                    </frame >
              </group >
        </coll >
  </app>
```

Ilustración 4 - Estructura de un fichero de vista en XOneCloud.

Para la interacción con la interfaz usamos JavaScript, y para dar estilo a los componentes CSS. Por lo que el desarrollo de software se hace muy parecido al desarrollo web.

1.2. Especificación previa

En resumen, la empresa necesita una aplicación móvil que conecte con los servicios de base de datos internos, y sea capaz de realizar las siguientes tareas. El operario debe ser capaz, a través de una interfaz de usuario similar a la de la aplicación de escritorio, introducir un nuevo parte de verificación en cualquier momento, en el cual podrá incorporar los datos específicos del vehículo o conjunto de vehículos (UT/SR[16]), se debe poder marcar si se trata de un parte de entrega o recepción, fecha, hora y estado del tanque de combustible.

Debe de haber una pantalla de inicio se sesión, que recoja datos de usuario y contraseña y permita entrar a la aplicación en caso de ser correcta la información.

La interfaz tendrá una serie de pestañas o un menú de navegación que nos permitirá movernos a través de las subinterfaces *General*, *EquipamientoUT*, *IncidenciasUT, EquipamientoSR*, *IncidenciasSR.*

1. Dentro de la pestaña *General*, se podrán introducir los datos acerca de Chapa, Deflectores, Cristales, Alumbrado, Interior, Limpieza Ext., Limpieza Int.; además del estado de los neumáticos de UT y SR.

2. Desde la pestaña *EquipamientoUT* se debe introducir información acerca de si la UT dispone de:

    a. ¿Tiene? si/no.

Tabla 1 - Elementos de la pestaña EquipamientoUT sección ¿Tiene?

✓ soporte extintor 3kg	✓ mang. aire roja ✓ mang. aire amarilla	✓ tapón ADBLUE ✓ bidón 25 litros	✓ enganche ✓ triangulo

[16] UT unidad tractora. SR semirremolque

✓ soporte extintor 9kg ✓ rollos papel tac ✓ caja rec. bombillas ✓ alfombra cabina ✓ cama superior ✓ colchón	✓ precinto caja cambio ✓ calzos ruedas ✓ cable 5ª rueda ✓ tercera matricula ✓ imán ADBLUE	✓ placa RAR ✓ kit ADR montado ✓ parabrisas ✓ retrovisores ✓ gato	✓ Mangueras ✓ Candado UT ✓ candado SR ✓ botiquín ✓ llave ruedas

b. ¿Tiene? si/no + cantidad.

Tabla 2 - Elementos de la pestaña EquipamientoUT sección ¿Tiene? + cantidad

✓ Cinchas ✓ Martillo	✓ llave abt cabina ✓ llave inglesa	✓ destornillador ✓ alicate

c. ¿Tiene? si/no + ¿funciona? s/no.

Tabla 3 - Elementos de la pestaña EquipamientoUT sección ¿Tiene? + funciona

✓ Termómetro ✓ segunda llave ✓ equipo qualcomm ✓ calefactor autónomo ✓ clima estacionamiento ✓ aire acondicionado ✓ cierre centralizado ✓ regulador espejos	✓ mando elevalunas ✓ bocina neumática ✓ bocina eléctrica ✓ maneta intermitente ✓ pistola aire ✓ mando suspensión ✓ maneta trampilla ✓ estado cinturón	✓ lámpara lectura ✓ mang. abs. + sop ✓ mang. luces. + sop ✓ antirrobos gasoil ✓ telemat ✓ nevera

3. Desde la pestaña de *IncidenciasUT* se podrán cargar fotografías que el operario considere oportunas.

4. Desde la pestaña *EquipamientoSR*, de un modo parecido al de EquipamientoUT.

    a. ¿Tiene? si/no + cantidad.

Tabla 4 - Elementos de la pestaña EquipamientoSR sección ¿Tiene? + cantidad

✓ Papel termógrafo ✓ Candado propio ✓ Con llave (candado)	✓ Rotulado ✓ Barras de seguridad	✓ Europallet ✓ Candado portapalet

    b. ¿Preguntas? si/no.

Tabla 5 - Elementos de la pestaña EquipamientoUT sección ¿Tiene?

✓ ¿se realiza pre-trip?	✓ ¿se encuentran fallas? ✓ ¿necesita interv en taller?

    c. Descripción fallas.

    d. Combustible SR.

5. Desde la pestaña de *IncidenciasSR* se podrán cargar fotografías que el operario considere oportunas.

2. Fases y cronograma

En este apartado se describen las tareas que se han llevado a cabo a lo largo del proyecto, cuantificando el tiempo dedicado que, para la Normativa específica del Trabajo Fin de Grado de la Escuela Superior de Ingeniería de la UAL[17], según el Artículo 2. *Naturaleza del TFG* punto 2º:

> *"El TFG a desarrollar por el estudiante se ajustará en tiempo y forma a 12 créditos ECTS que corresponden a 300 horas de trabajo, tal y como refleja la memoria de verificación del grado."*

Aclaración. Se describen a continuación las tareas pertenecientes al TFG (consultar ilustración 5).

El primer punto importante en el buen desarrollo y termino de un proyecto es la planificación y estimación (Brice, 2015).

2.1. Diagrama de Gantt

Para la temporización del proyecto se ha realizado un diagrama de Gantt $(ver\ ilustración\ 5)$. Este tipo de diagrama usado para planificación de proyectos ofrece una visión total de la temporización, donde se aprecian los tiempos parciales que corresponden a cada una de las tareas y la dependencia entre ellas. Para hacernos una idea clara de cómo se van a invertir las horas que se estipulan en la normativa, se detallan a continuación las que corresponden al TFG:

- Análisis. - Se pretende realizar un análisis preliminar de los requisitos tanto funcionales como no funcionales para la aplicación. 40 horas.
- Diseño. - Para el diseño de la interfaz de usuario se pretende seguir la línea de la aplicación de escritorio existente, aun así, se pretende realizar algún mock-up en líneas básicas. 30 horas.
- Formación XOne. - La empresa pretende poner a mi disposición un periodo de formación en el entorno de XOne, posiblemente una semana variable. 40 horas.
- Implementación. - Codificación de la aplicación a través del entorno de XOne. 140 horas.
- Redacción de la memoria del TFG. - 50 horas.

Se ha decidido seguir una metodología tradicional en cascada (Brice, 2015), ya que las tareas que se necesitan terminar están bien definidas y acotadas en el tiempo, aunque hoy en día están ampliamente extendidas las metodologías ágiles, que son más favorables desde un punto de vista multidisciplinar en un equipo de trabajo.

[17] normativa_tfg_informatica1617.pdf

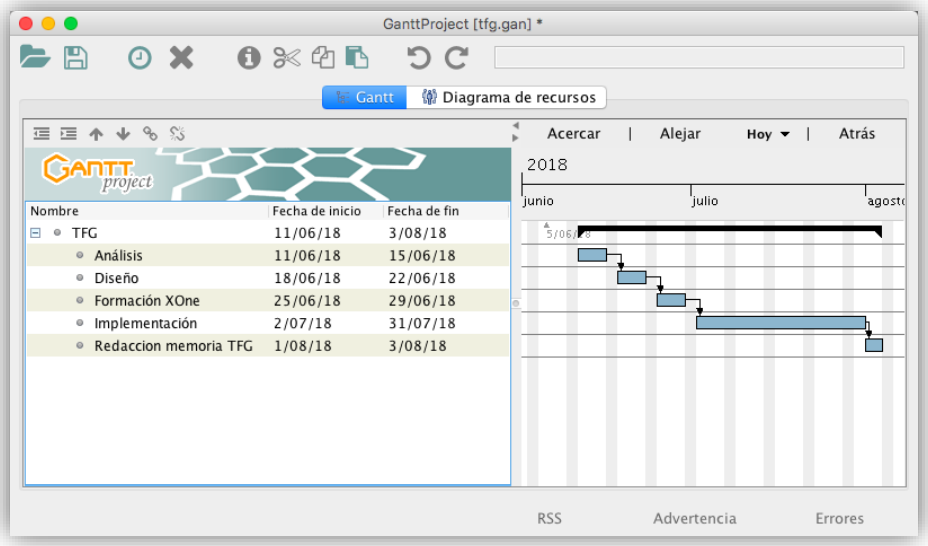


Ilustración 5 - Diagrama de Gantt del TFG

2.2. Estimación de esfuerzos

Para hacernos una idea de lo que va a significar este proyecto en personas/tiempo, vamos a realizar un análisis teórico de puntos de función y seguido de estudio del resultado por modelo COCOMO.

2.2.1. Análisis de puntos de función

Para hacer una estimación del número de líneas de código que puede suponer este proyecto vamos a realizar un análisis de puntos de función a través de las categorías de transacciones ficheros lógicos. Las transacciones serán el número de entradas, salidas y consultas de datos que debe realizar la aplicación, mientras que los ficheros son los distintos almacenes que soportan esos datos, vamos a verlo más detalladamente.

1. Entradas. - Informaciones que llegan a la aplicación desde el exterior.

2. Salidas. - Informaciones elaboradas por la aplicación que salen de los límites de la aplicación. Incluye informes y mensajes al usuario, a otras aplicaciones y a sus usuarios.

3. Consultas. - Entradas on-line que producen inmediatamente una salida on-line

4. Ficheros lógicos internos. - Almacenamientos de datos permanentes mantenidos por la aplicación.

5. Interfaces. - Ficheros a los que accede la aplicación con el único objetivo de obtener información. Son mantenidos por otras aplicaciones.

Según se detalla en la especificación previa (punto 1.2), la aplicación va a tener unas 140 entradas, 0 salidas, 1 consulta, 1 fichero interno y 3 interfaces

(*consultar tabla* 6). Por lo que podemos asumir que todas las categorías son de dificultad media, distribuyéndolos en la siguiente tabla:

Tabla 6 - Puntos de función sin ajustar

	Simple		Media		Compleja		Total
	Cantidad	Peso	Cantidad	Peso	Cantidad	Peso	
Entradas		*3	140	*4		*6	560
Salidas		*4	0	*5		*7	0
Consultas		*3	1	*4		*6	4
Fic. lógicos		*7	1	*10		*15	10
Fic. Interfaces		*5	3	*7		*10	21
Total, puntos de función sin ajustar (PFSA)							595

Para terminar, debemos tener en cuenta los factores de complejidad del software, se trata de 14 factores que completan la visión externa de la aplicación pero que no están recogidos en la funcionalidad. Toman un valor entre 0 y 5.

Tabla 7 - Factores de complejidad del software

i	Valor de complejidad	F
1	¿Se requiere comunicación de datos?	3
2	¿Existen funciones de procesamiento distribuido?	0
3	¿Es crítico el rendimiento?	1
4	¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado?	1
5	¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?	0
6	¿Requiere el sistema entrada de datos interactiva?	0
7	¿Se ha diseñado la aplicación para ser fácilmente utilizada por el usuario?	4
8	¿Se actualizan los archivos maestros de forma interactiva?	1
9	¿Es complejo el procesamiento interno?	0
10	¿Se ha diseñado el código para ser reutilizable?	3
11	¿Están incluidas en el diseño la conversión y la instalación?	0
12	¿Requiere el sistema copias de seguridad y de recuperación fiables?	4
13	¿Se ha diseñado el sistema para soportar instalaciones en difern organizaciones?	0
14	¿Se ha diseñado la aplicación para facilitar los cambios?	0
Factor de ajuste de la complejidad (FAC): $0.65 + 0.01 * \sum(Fi)$		

$$PF = PFSA * (0.65 + 0.01 * 17) = 595 * 0.82 = 487$$

Si ahora tenemos en cuenta que trabajando con la plataforma de XOne, el lenguaje de programación está totalmente abstraído a través de XML, asumiremos un multiplicador de puntos de función de lenguajes de 5º generación igual a 5, luego.

$$487 * 5 = 2439\ lineas\ de\ código$$

2.2.2. COCOMO

El Modelo Constructivo de Costos conocido por sus siglas COCOMO (COnstructive COst MOdel), es un sistema basado en cálculos matemáticos que permite hacer una estimación de costes para proyectos de desarrollo de software, ya que se basa en las líneas de código escritas.

Existen tres modelos que ajustan el nivel de detalle a la hora de hacer estimaciones:

- COCOMO básico. Calcula el esfuerzo y el coste del desarrollo del software en función del tamaño del programa (LDC estimadas).

- COCOMO intermedio. Calcula el esfuerzo y el coste del desarrollo del software en función del tamaño del programa y de un conjunto de atributos, los cuales evalúan de forma subjetiva el producto, el hardware, el personal y los atributos del proyecto.

- COCOMO avanzado. Incorpora todas las características del modelo intermedio y lleva a cabo una evaluación del impacto de los conductores del coste en cada fase del proceso de ingeniería del software.

Y estos modelos se pueden aplicar a tres tipos de proyectos escalados en complejidad:

- Modo orgánico. Proyectos de software relativamente pequeños y sencillos, realizados por pequeños equipos con buena experiencia en la aplicación, y requisitos poco rígidos. Ej.: pequeño almacén de existencias

- Modo semi-acoplado. Proyectos de software intermedios en tamaño y complejidad, realizados por equipos con variados niveles de experiencia, y con requisitos poco o menos rígidos. Ej.: sistema de ventas en grandes almacenes.

- Modo empotrado. Proyectos de software que deben ser desarrollados en un conjunto de hardware, software y restricciones operativas muy restringidas. Proyectos únicos, con poca experiencia. Ej.: Control de vuelo de una nave.

En nuestro caso se trata de un proyecto orgánico, y vamos a usar el modelo COCOMO básico:

Tabla 8 - Tabla de referentes para calcular COCOMO básico.

Proyecto Software	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semi-acoplado	3.0	1.12	2.5	0.35
Empotrado	3.6	1.20	2.5	0.32

(Analysis, 2009)

Para la realización de este proyecto se estiman unas 2.4K líneas de código.

Calculo de estimación:

1) Esfuerzo. Para calcular el esfuerzo en personas-mes.

$$E = a * KLDC^b$$

Donde E es el esfuerzo y KLDC el número de líneas de código en miles.

$$E = 2.4 * 2.4^{1.05} = 6.01\ personas/mes$$

2) Duración.

$$D = c * E^d$$

Donde D es la duración en meses.

$$D = 2.5 * 6.01^{0.38} = 4.9\ meses$$

3) Número de personas.

$$N = \frac{E}{D}$$

Donde N es el número de personas

$$N = \frac{6.01\ \frac{p}{m}}{4.9\ m} = 1.21\ p$$

Los resultados teóricos obtenidos indican que para realizar este proyecto son necesarias 1.21 personas trabajando durante 4.9 meses.

2.3. Tareas

Como ya hemos comentado, este proyecto abarca el desarrollo de una aplicación híbrida móvil, destinada a la gestión de llegada y salida de unidades tractoras y semirremolques (desde el punto de vista de taller) para una empresa de transportes.

2.3.1. Análisis

Una buena base para la correcta elaboración de un producto software es conocer y entender muy bien que es lo que se desea desarrollar.

> *El Área de conocimiento de análisis de requisitos describe las tareas y técnicas utilizadas por un analista de negocios para analizar los requisitos establecidos con el fin de definir las capacidades requeridas de una posible solución que satisfaga las necesidades de las partes interesadas. Cubre la definición de requisitos de partes interesadas, que describen qué debe ser capaz de hacer una solución para satisfacer las necesidades de uno o más grupos de partes interesadas, y los requisitos de solución, que describen el comportamiento de los componentes de la solución con suficiente detalle para permitir su construcción. Las tareas en esta área de conocimiento se aplican a los requisitos de las partes interesadas y de la solución. (Analysis, 2009)*

Para todo el modelado del proyecto se usa UML como estándar, desde los requisitos hasta los diagramas entidad relación, por la claridad de esta metodología (Laurent & Fien, 2016)

La comunidad de estandarización internacional, incluida la IEEE ha reflejado el análisis de requisitos manuales de buenas prácticas como el *IEEE Recommended Practice for Software Requirements Specifications* (The Institute of Electrical and Electronics Engineers, Inc, 1998). Según la guía SWEBOK, "…*En su aspecto más básico, un requisito de software es una propiedad que debe exhibirse para resolver algún problema en el mundo real…"* (IEEE Computer Society, 2014)

Requisitos funcionales del sistema

- Requisitos de información.

Tabla 9 - IRQ-01 Información sobre los vehículos

IRQ - 01	Información sobre los vehículos
Versión	1.0
Dependencias	
Descripción	El sistema deberá almacenar la información correspondiente a los vehículos en relación al estado, procedente de la base de datos de la empresa. En concreto:
Datos específicos	• Datos generales (matricula, kilómetros, conductor, etc.) • Estado general • Estado de los neumáticos • Equipamiento • incidencias
Importancia	P.D.

Prioridad	P.D.
Estado	-
Comentarios	Ninguno

Tabla 10 - IRQ-02 Información sobre los partes

IRQ - 02	Información sobre los partes
Versión	1.0
Dependencias	
Descripción	El sistema deberá almacenar localmente la información correspondiente a los partes que se generen en relación con el vehículo correspondiente en concreto:
Datos específicos	• Datos generales (matricula, kilómetros, conductor, etc.) • Estado general • Estado de los neumáticos • Equipamiento • incidencias
Importancia	P.D.
Prioridad	P.D.
Estado	-
Comentarios	Ninguno

Requisitos no funcionales del sistema

Tabla 11 - RNF - 01 Disponibilidad

RNF - 01	Disponibilidad
Versión	1.0
Descripción	El sistema depende de la información contenida en las bases de datos de la compañía, de las cuales recoge la información siempre que disponga de conexión, por lo demás, al disponer de su propia base de datos, tiene un nivel de independencia alto.
Importancia	P.D.
Prioridad	P.D.
Estado	-
Comentarios	Ninguno

Tabla 12 - RNF - 02 Integridad

RNF - 02	Integridad
Versión	1.0
Descripción	Los datos introducidos serán primero almacenados de manera local en el dispositivo para sincronizarlos con el servidor de la empresa más tarde, lo que quiere decir que los datos deberán comprobarse adecuadamente antes de realizar esta sincronización.
Importancia	P.D.
Prioridad	P.D.
Estado	-
Comentarios	Ninguno

Tabla 13 - RNF - 03 Fiabilidad

RNF - 03	Fiabilidad
Versión	1.0
Descripción	Se debe minimizar el número de fallos para que el sistema trabaje con fluidez, manejar excepciones si fuera necesario.
Importancia	P.D.
Prioridad	P.D.
Estado	-
Comentarios	Ninguno

Tabla 14 - RNF - 04 Facilidad de uso

RNF - 04	Facilidad de uso
Versión	1.0
Descripción	La aplicación está destinada a operarios del aparcamiento de la compañía, no se trata de un público desconocido, además se intentará diseñar la interfaz lo más parecida posible a la versión de escritorio, si fuera necesario se puede realizar una formación previa al uso.
Importancia	P.D.
Prioridad	P.D.
Estado	-
Comentarios	Ninguno

Tabla 15 - RNF - 05 Robustez

RNF - 05	Robustez
Versión	1.0
Descripción	Los datos introducidos se comprueban antes de guardarlos si no son correctos o íntegros el sistema responderá adecuadamente.
Importancia	P.D.
Prioridad	P.D.
Estado	-
Comentarios	Ninguno

Tabla 16 - RNF - 06 Rendimiento

RNF - 06	Rendimiento
Versión	1.0
Descripción	No debe ser muy pesado manejando los datos, se trata de una app de tránsito entre el almacén real de los datos, no debería tardar más de 5 segundos en realizar una operación.
Importancia	P.D.
Prioridad	P.D.
Estado	-
Comentarios	Ninguno

Tabla 17 - RNF - 07 Legislación

RNF - 07	Legislación
Versión	1.0
Descripción	Se cumplirán los estándares y la legislación vigente, una de las opciones de la aplicación consiste en tomar fotografías con la cámara del dispositivo, se atenderá en este caso celosamente a la ley de protección de datos de carácter personal.
Importancia	P.D.
Prioridad	P.D.
Estado	-
Comentarios	Ninguno

Tabla 18 - RNF - 08 Portabilidad

RNF - 08	Portabilidad
Versión	1.0
Descripción	La aplicación se ejecuta en dispositivos móviles propiedad de la empresa, se atenderá al tamaño y resolución de estos dispositivos únicamente, puesto que se va a escribir con el único fin de uso para la compañía.
Importancia	No tiene.
Prioridad	P.D.
Estado	-
Comentarios	Ninguno

Tabla 19 - RNF - 09 Seguridad

RNF - 09	Seguridad
Versión	1.0
Descripción	Los datos se tratan a través de una pasarela externa a la compañía dependiente de la corporación que suministra el entorno de desarrollo, esto deberá realizarse de forma segura.
Importancia	No tiene.
Prioridad	P.D.
Estado	-
Comentarios	Ninguno

2.3.2. Diseño: Front-End y Back-End

Hoy en día cuando desarrollamos una aplicación se tienen en cuenta dos grandes partes del mismo producto, el "*front-end*" y el "*back-end*". Ambas partes se interrelacionan de manera que, tal y como se entiende hoy la ingeniería del software, van de la mano. El back-end se encarga de almacenar y manejar los datos desde el punto de vista del servidor, mientras que el Frontend hace lo propio desde el punto de vista del cliente; el front-end muestra los datos de manera intuitiva y ordenada, el back-end los guarda de manera robusta y persistente.

Por una cuestión de paralelismo en la manera de hablar, además de que los inicios etimológicos de muchas de los términos que giran en torno a estos temas son nuevos anglicismos, cuando se habla de front-end y back-end, también solemos referirnos a la parte del desarrollo que corresponde a cada uno de los mismos, es decir, el front-end es el desarrollo del propio front-end, de un modo similar al acrónimo recursivo GNU.

Cuando hablamos de front-end, nos referimos al desarrollo de la estructura visual de la aplicación, tamaños, formas, colores, animaciones, etc. Además de la capacidad de interacción que se desea ofrecer al usuario, lo que llamamos UX, siglas en ingles de "*user experience"*. En esta parte del desarrollo normalmente intervienen lenguajes como HTML, XML, CSS, JavaScript, etc.

Por otro lado, el back-end abarca el desarrollo de la estructura interna, tipos de datos, almacenes, lógica del negocio, etc. No esta tan enfocado su desarrollo a la interacción con el usuario, puesto que precisamente la interacción entre el usuario y el back-end esta abstraída por el front-end. Cuando el usuario toca en un botón para filtrar unos resultados por fecha, lo que realmente está sucediendo es que el front-end le está pidiendo los resultados a la base de datos o al back-end, y cuando los tiene los ordena o filtra según la petición que haya realizado el usuario.

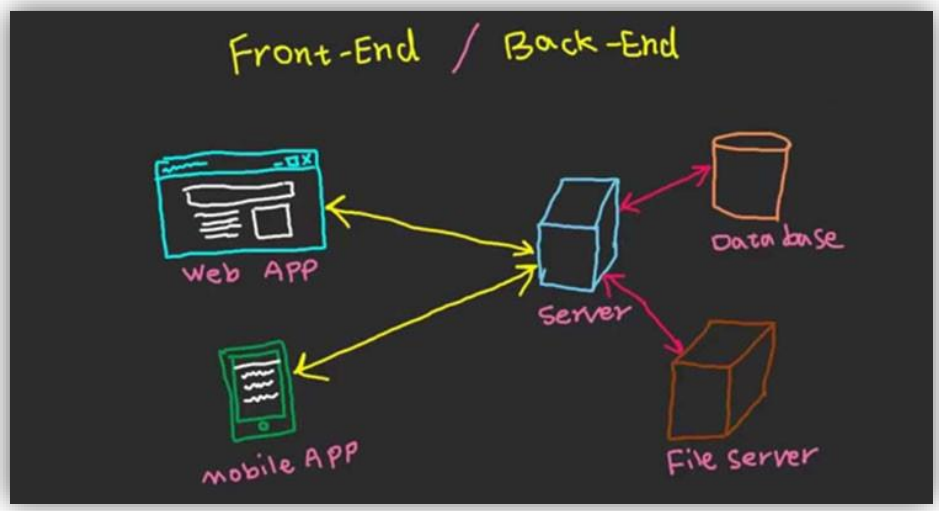


Ilustración 6 - Esquema Front-end & Back-end

Este modelo de abstracción ha permitido derivar en patrones de diseño (Debrauwer, 2012). Se trata de un modelo de programación basado en patrones, un Patrón describe un problema recurrente y una solución. Cada patrón nombra, explica, evalúa un diseño recurrente en sistemas Orientados a Objetos. Uno de los más conocidos y usados es el MVC “Model View Controller” (Center, s.f.) *(modelo vista controlador)*. Según el cual:

> *“...una aplicación consta de un modelo de datos, de información de presentación y de información de control. El patrón requiere que cada uno de estos elementos esté separado en distintos objetos…”*

2.3.2.1. Front-End

Antes de empezar a dibujar la interfaz de usuario, se ha realizado un diagrama de casos de usos (*ver ilustración* 7) para aclarar mejor la funcionalidad que se desea conseguir con cada una de las partes de la aplicación.

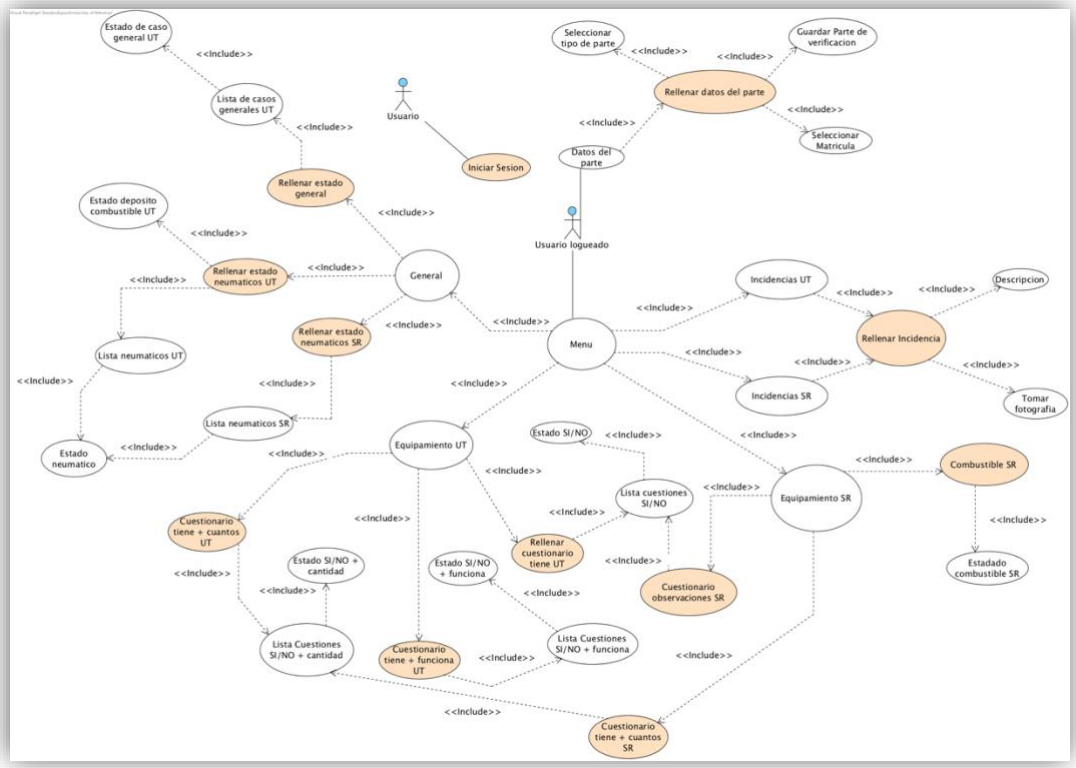


Ilustración 7 - Diagrama de casos de usos

Especificación de Actores del Sistema

Tabla 20 - CDU-ACT 01 Operario de campa

CDU-ACT 01	Usuario
Versión	1.0
Dependencias	Ninguna
Descripción	Este actor representa al grupo de usuarios cuya función es iniciar sesión.
Comentarios	

Tabla 21 - CDU-ACT 02 Usuario logueado

CDU-ACT 02	Usuario logueado
Versión	1.0
Dependencias	Ninguna
Descripción	Este actor representa al grupo de usuarios cuya función es rellenar los partes de verificación de los vehículos al llegar o salir de viaje.
Comentarios	

Especificación de Casos de Uso del Sistema

Tabla 22 - UC 01 Iniciar sesión

UC 01	Iniciar sesión
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 01 desee iniciar sesión.
Precondición	Ninguna
Secuencia normal	• El actor usuario introduce su login de “usuario” y “contraseña”. • El sistema le dará paso tras comprobar que la información existe y es correcta.
Post condición	-
Excepciones	-
Importancia	Alta
Prioridad	P.D.
Estado	-

Comentarios	ninguno

Tabla 23 - UC 02 Rellenar datos de parte

UC 02	Rellenar datos parte
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar los datos del parte “tipo” y “matricula”.
Precondición	Ninguna
Secuencia normal	• Se selecciona el check-box correspondiente para el tipo de parte • Se selecciona la matrícula de un campo desplegable
Post condición	-
Excepciones	-
Importancia	Alta
Prioridad	P.D.
Estado	-
Comentarios	ninguno

Tabla 24 - UC 03 rellenar estado general

UC 03	Rellenar estado general
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar la información general del cuestionario.
Precondición	Ninguna
Secuencia normal	• Se selecciona un estado “bien”, “regular” o “mal” para los datos correspondientes. Se deben rellenar todos los campos.
Post condición	-
Excepciones	-
Importancia	Alta

Prioridad	P.D.
Estado	-
Comentarios	ninguno

Tabla 25 - UC 04 Rellenar estado neumáticos UT

UC 04	Rellenar estado neumáticos UT
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar el estado de los neumáticos de la cabeza tractora.
Precondición	Ninguna
Secuencia normal	• El sistema carga los datos correspondientes a marca y número de serie de los neumáticos del vehículo, el usuario deberá seleccionar si el estado de la cubierta de cada neumático es correcto a través de unos check-box (si /no) y establecer el porcentaje de inflado de los mismos desde un campo desplegable. (25%, 50%, 75%, 100%). Se deben rellenar todos los campos.
Post condición	-
Excepciones	-
Importancia	Alta
Prioridad	P.D.
Estado	-
Comentarios	ninguno

Tabla 26 - UC 05 Rellenar estado neumáticos SR

UC 05	Rellenar estado neumáticos SR
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar el estado de los neumáticos del semirremolque.
Precondición	Ninguna

Secuencia normal	• Este caso de uso es idéntico a UC 04 en su secuencia, con la aclaración de que el semirremolque dispone de 8 ruedas en lugar de 6. Se deben rellenar todos los campos.
Post condición	-
Excepciones	-
Importancia	Alta
Prioridad	P.D.
Estado	-
Comentarios	ninguno

Tabla 27 - UC 06 Rellenar cuestionario tiene UT

UC 06	Rellenar cuestionario tiene UT
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar las cuestiones ¿tiene? (si / no) para la cabeza tractora
Precondición	Ninguna
Secuencia normal	• El usuario deberá seleccionar un valor para cada uno de los check-box del apartado correspondiente. Se deben rellenar todos los campos.
Post condición	-
Excepciones	-
Importancia	Alta
Prioridad	P.D.
Estado	-
Comentarios	ninguno

Tabla 28 - UC 07 Rellenar cuestionario tiene + funciona UT

UC 07	Rellenar cuestionario tiene + funciona UT
Versión	1.0
Dependencias	Ninguna

Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar las cuestiones ¿tiene? (si / no) y ¿funciona? (si / no) para la cabeza tractora
Precondición	Ninguna
Secuencia normal	• El usuario deberá seleccionar un valor para cada uno de los check-box del apartado correspondiente. Se deben rellenar todos los campos.
Post condición	-
Excepciones	-
Importancia	Alta
Prioridad	P.D.
Estado	-
Comentarios	ninguno

Tabla 29 - UC 08 Rellenar cuestionario tiene + cuento UT

UC 08	Rellenar cuestionario tiene + cuento UT
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar las cuestiones ¿tiene? (si / no) y ¿cuántos? (campo número) para la cabeza tractora
Precondición	Ninguna
Secuencia normal	• El usuario deberá seleccionar un valor para cada uno de los check-box e introducir un número en el campo correspondiente. Se deben rellenar todos los campos.
Post condición	-
Excepciones	-
Importancia	Alta
Prioridad	P.D.
Estado	-
Comentarios	ninguno

Tabla 30 - UC 09 Rellenar cuestionario tiene + cuento SR

UC 09	Rellenar cuestionario tiene + cuento SR
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar las cuestiones ¿tiene? (si / no) y ¿cuántos? (campo número) para el semirremolque.
Precondición	Ninguna
Secuencia normal	• El usuario deberá seleccionar un valor para cada uno de los check-box e introducir un número en el campo correspondiente. Se deben rellenar todos los campos.
Post condición	-
Excepciones	-
Importancia	Alta
Prioridad	P.D.
Estado	-
Comentarios	ninguno

Tabla 31 - UC 10 Rellenar cuestionario observaciones SR

UC 10	Rellenar cuestionario observaciones SR
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar las cuestiones “observaciones” para el semirremolque
Precondición	Ninguna
Secuencia normal	• El usuario deberá seleccionar un valor para cada uno de los check-box e introducir una descripción del problema si fuese necesario. Se deben rellenar todos los campos
Post condición	-
Excepciones	-
Importancia	Alta
Prioridad	P.D.
Estado	-

Comentarios	ninguno

Tabla 32 - UC 11 Rellenar cuestionario combustible SR

UC 11	Rellenar cuestionario combustible SR
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar el apartado de combustible para el semirremolque
Precondición	Ninguna
Secuencia normal	• El usuario deberá seleccionar un valor para el estado del tanque de combustible del semirremolque: vacío VA, un cuarto de tanque ¼, medio tanque ½, lleno LL.
Post condición	-
Excepciones	-
Importancia	Alta
Prioridad	P.D.
Estado	-
Comentarios	ninguno

Tabla 33 - UC 12 Rellenar incidencia

UC 12	Rellenar incidencia
Versión	1.0
Dependencias	Ninguna
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario CDU-ACT 02 desee rellenar una incidencia.
Precondición	Ninguna
Secuencia normal	• El usuario deberá escribir una descripción de la incidencia y podrá tomar una fotografía si lo considera conveniente.
Post condición	-
Excepciones	-
Importancia	Alta

Prioridad	P.D.
Estado	-
Comentarios	ninguno

El usuario, tras iniciar sesión, podrá seleccionar el tipo de parte y matrícula del vehículo a través del panel superior. Y a través de un menú de pestañas, podrá moverse por las diferentes opciones:

“General”, podrá rellenar el estado general del vehículo, así como el estado de los neumáticos del conjunto a través de unas listas de check-box, también se puede introducir el estado del tanque de combustible.

“Equipamiento UT”, podrá rellenar tres cuestionarios cada uno con un tipo de lista de preguntas, una lista para cuestiones “Tiene + funciona” propia de esta pestaña, una lista para cuestiones “Tiene + cantidad” y otra lista para cuestiones “Tiene”, las dos últimas comparte funcionalidad con la pestaña de “Equipamiento SR”

“Equipamiento SR” tiene las dos listas mencionadas y además un apartado para el estado del tanque de combustible del semirremolque.

“Incidencia UT” e “Incidencia SR” comparte la función de añadir una incidencia al parte, en sendas pestañas se desplegará una ventana emergente que nos permite introducir una descripción y una fotografía a través de la cámara integrada del dispositivo.

A continuación, se expone el mock-up de la interfaz de usuario (*consultar ilustraciones de la 8 a la 19*).

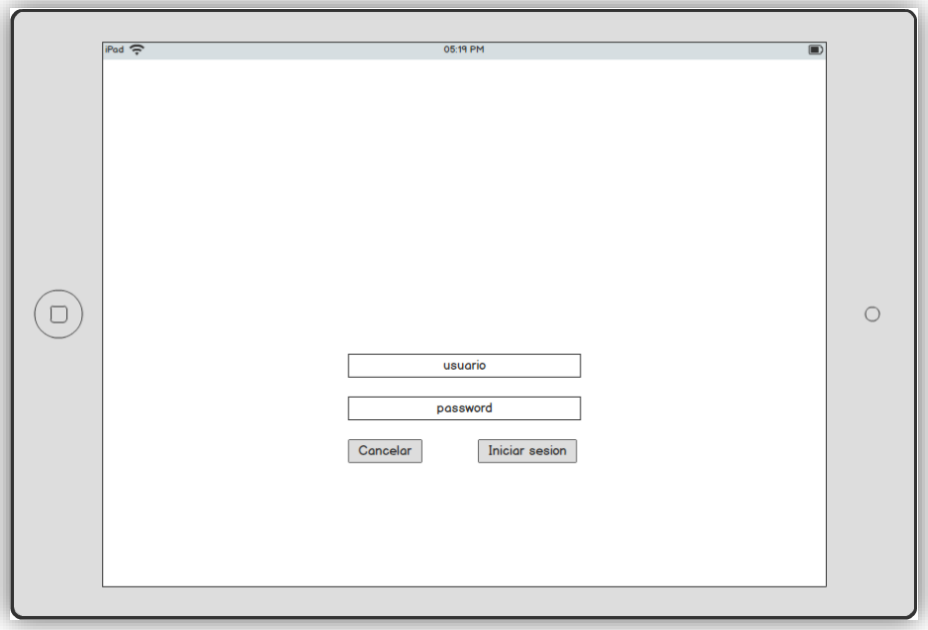


Ilustración 8 - Inicio sesión

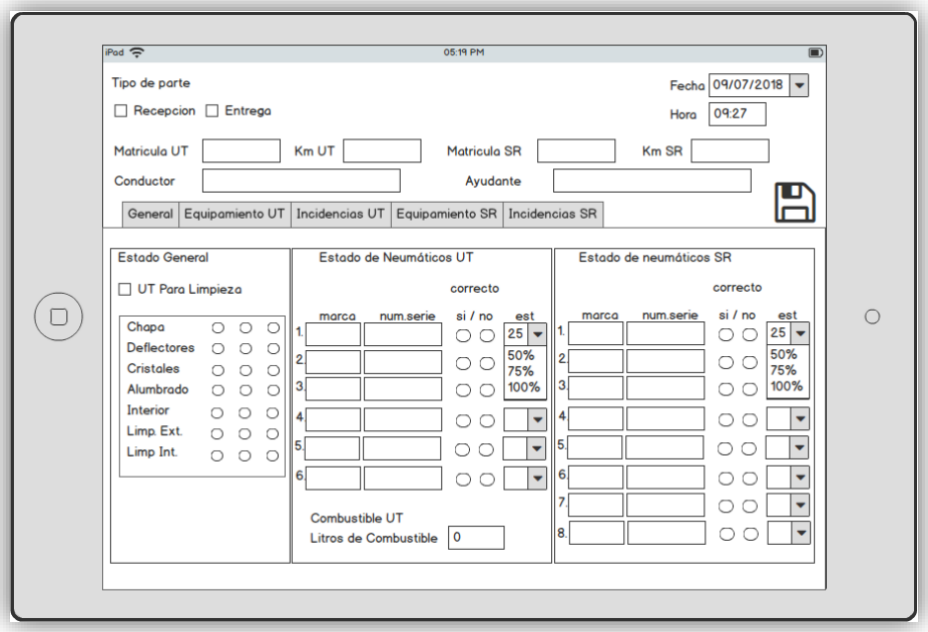


Ilustración 9 – General

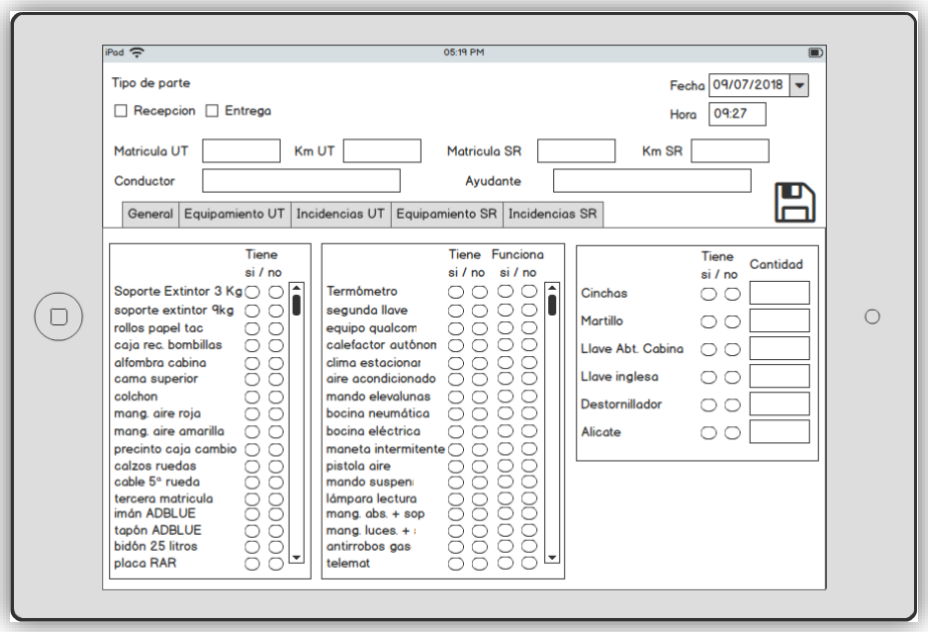


Ilustración 10 - Equipamiento UT

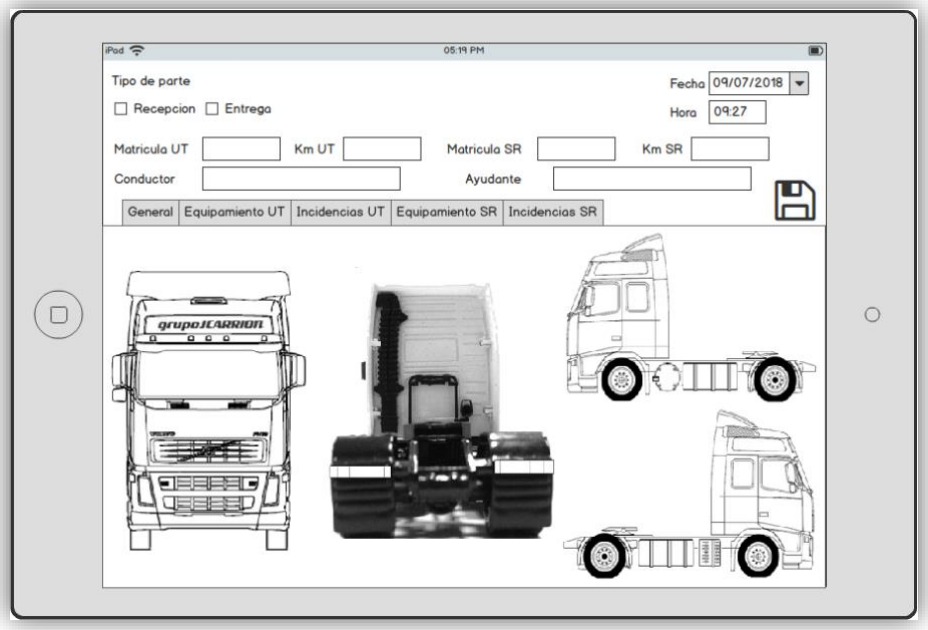


Ilustración 11 - Incidencias UT

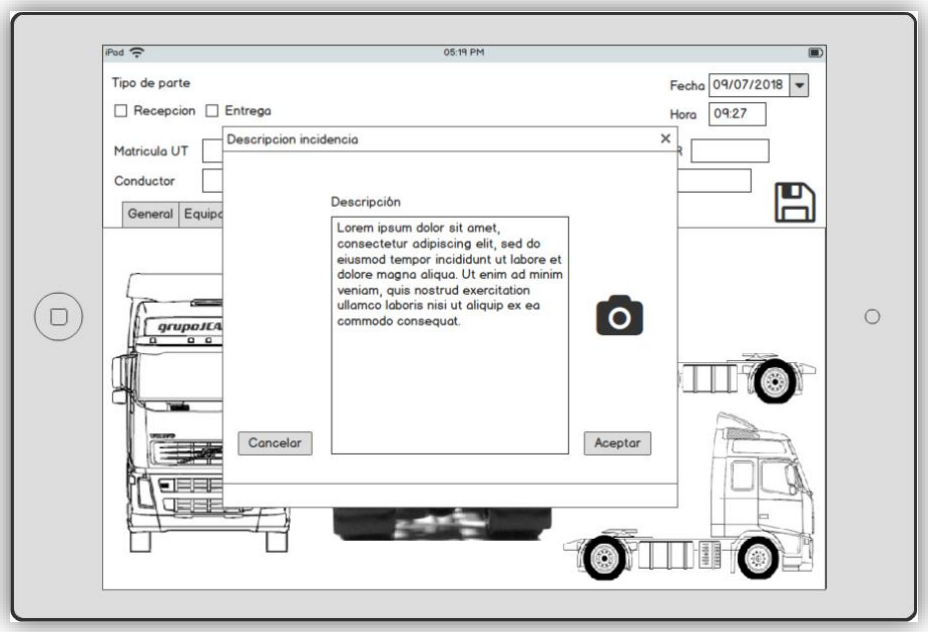


Ilustración 12 - Incidencias UT descripción

Ilustración 13 - Incidencias UT foto

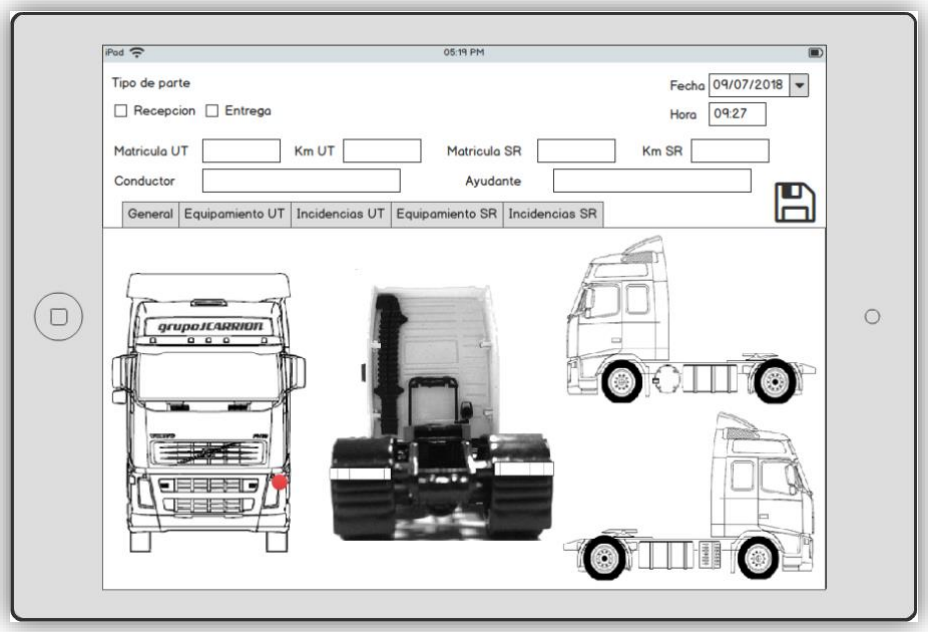


Ilustración 14 - Incidencias UT marca

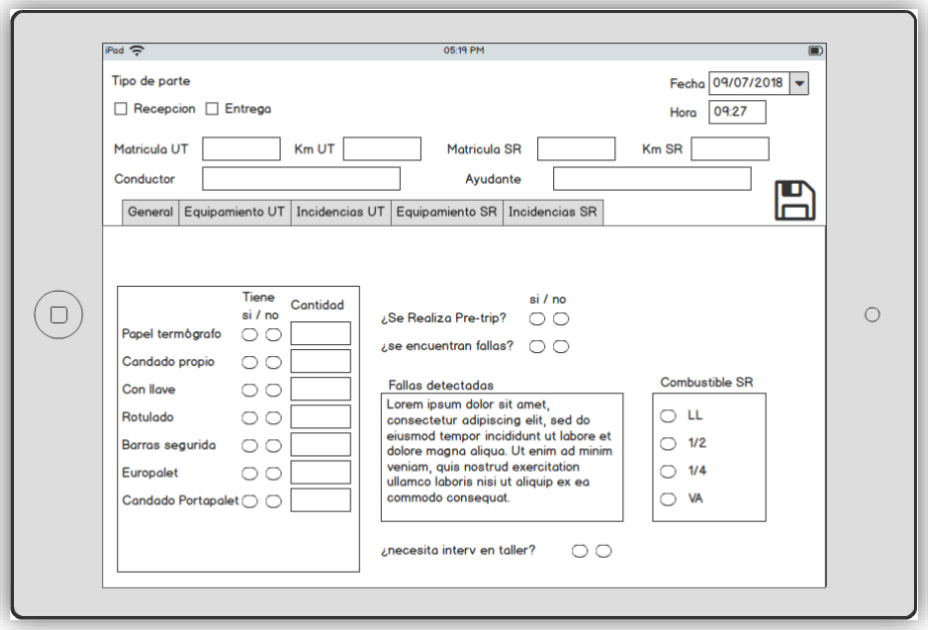


Ilustración 15 - Equipamiento SR

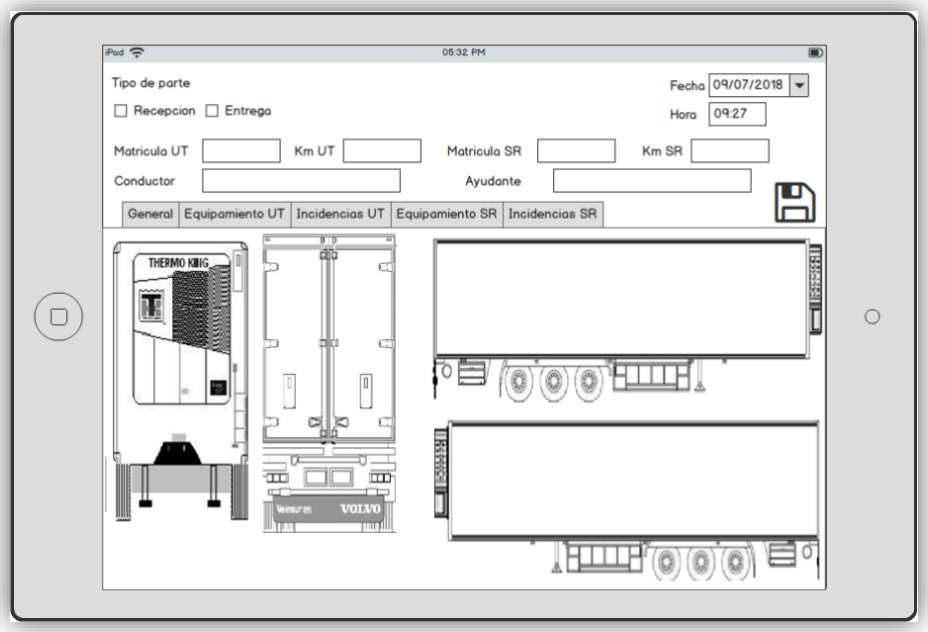


Ilustración 16 - Incidencias SR

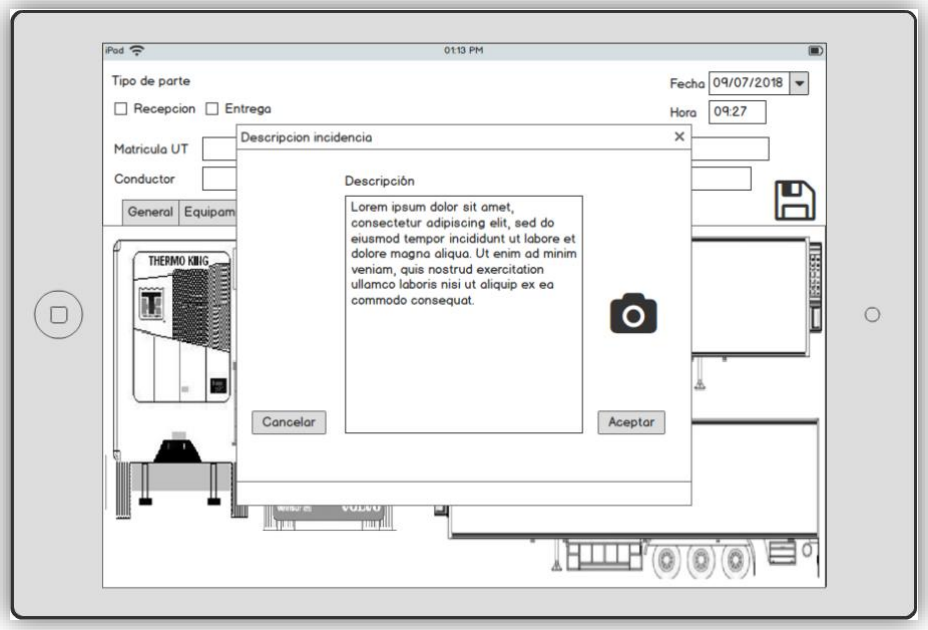


Ilustración 17 - Incidencias SR. Descripción.

Ilustración 18 - Incidencias SR. foto.

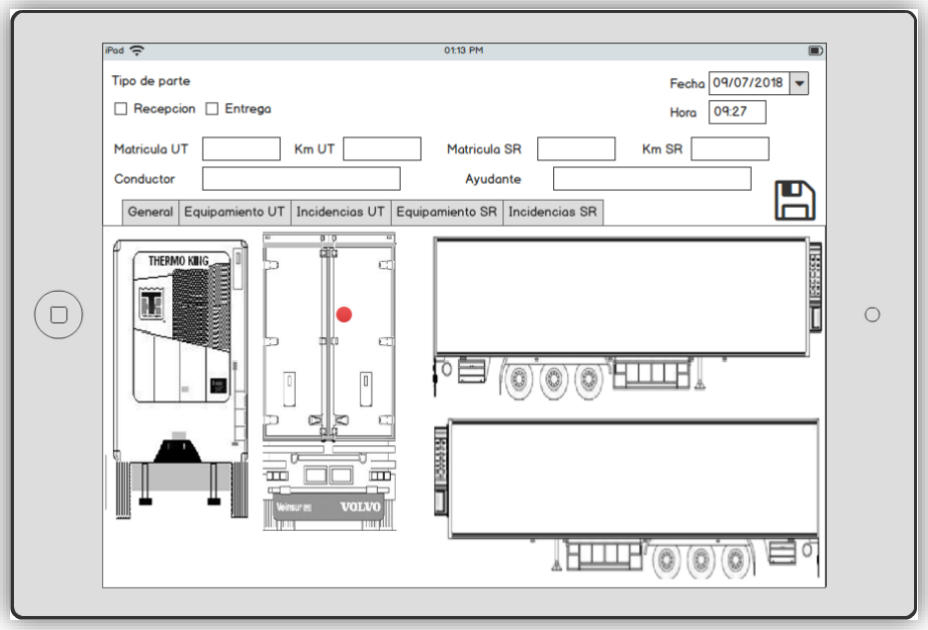


Ilustración 19 - Incidencias SR. Marca.

2.3.2.2. Back-End

El siguiente paso es empezar a definir la base de datos. El motor local de base de datos de la aplicación esta en SQLite, que es el motor usado por aplicaciones complejas en dispositivos móviles (Android e iOS). El diagrama ER (entidad relación) a continuación ha sido diseñado desde MySQL Workbench.

El proceso de diseño de la base de datos local para la aplicación ha sido condicionado por la existencia de la base de datos general de la empresa de donde nuestra app, descargará la información que necesita, que será una parte del total. De manera que el primer paso que hemos llevado a cabo ha sido imaginar cómo sería el esquema local que gira en torno al "parte", el resultado a continuación (*ver ilustración* 20).

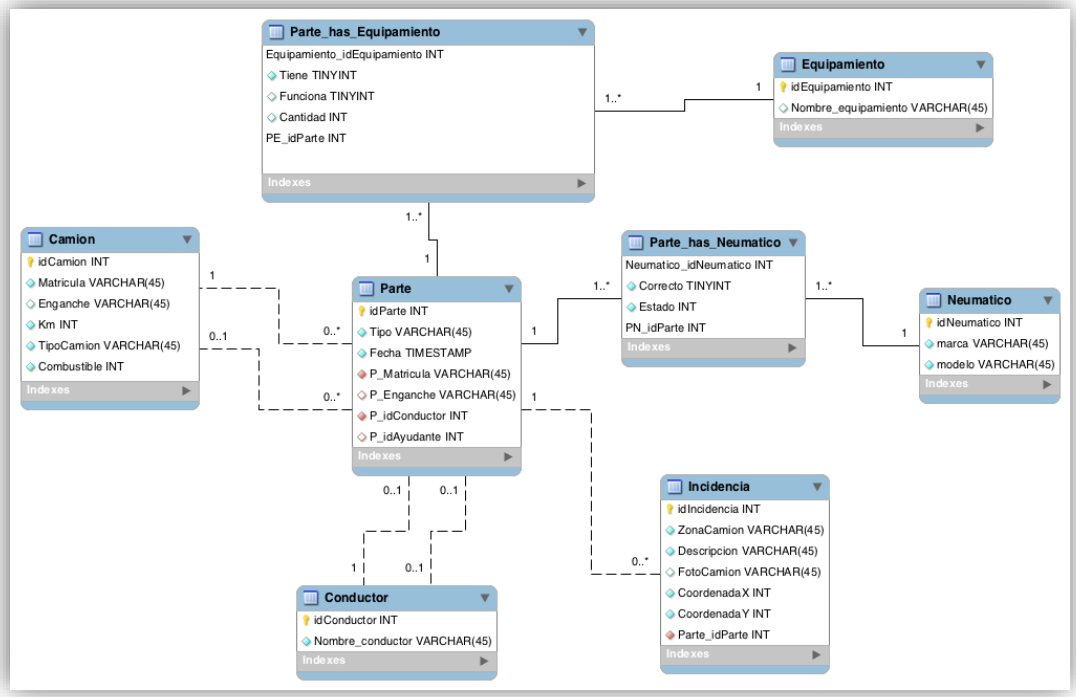


Ilustración 20 - 1er diseño de base de datos

Seguidamente hemos tratado de hacer una adaptación de la parte que corresponde a nuestra aplicación de la información de que dispone la compañía, con lo que hemos conseguido un esquema centrado en el camión, como se puede observar el resultado es paralelo al 1er diseño (*ver ilustración* 21).

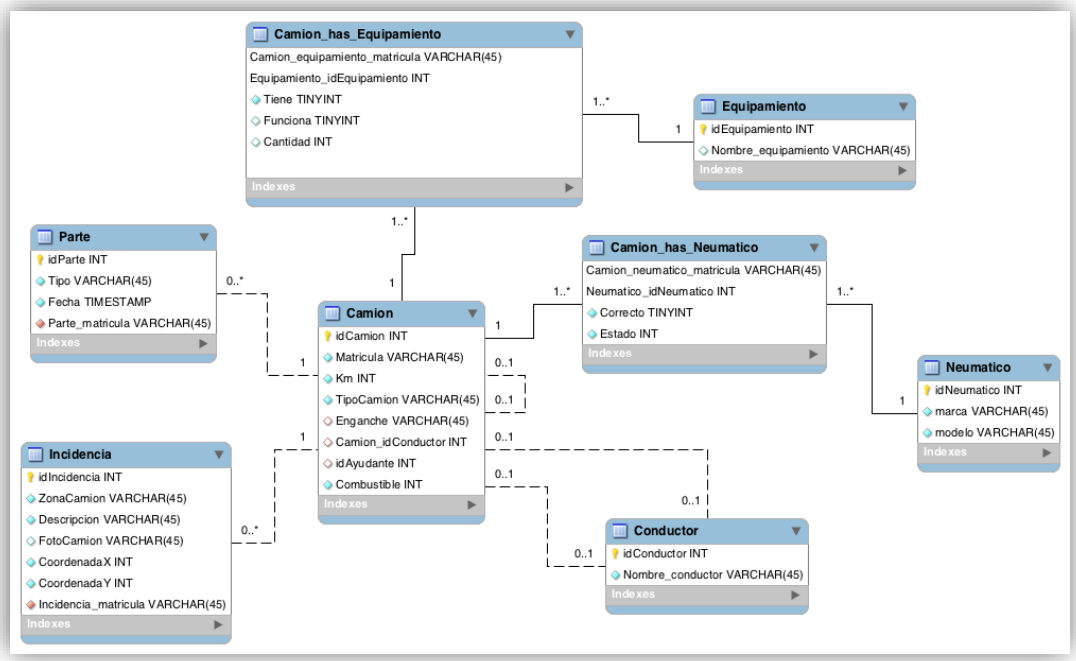


Ilustración 21 - 2º diseño de base de datos

Por último, se ha tratado de mezclar ambos diseños con la idea de conseguir un esquema unificado, de esta forma se tratamos de almacenar la información necesaria descargada del servidor por un lado y generar la nueva información relativa a los partes por otro. En resultado es un esquema fusión de los dos diseños anteriores (*ver ilustración* 22).

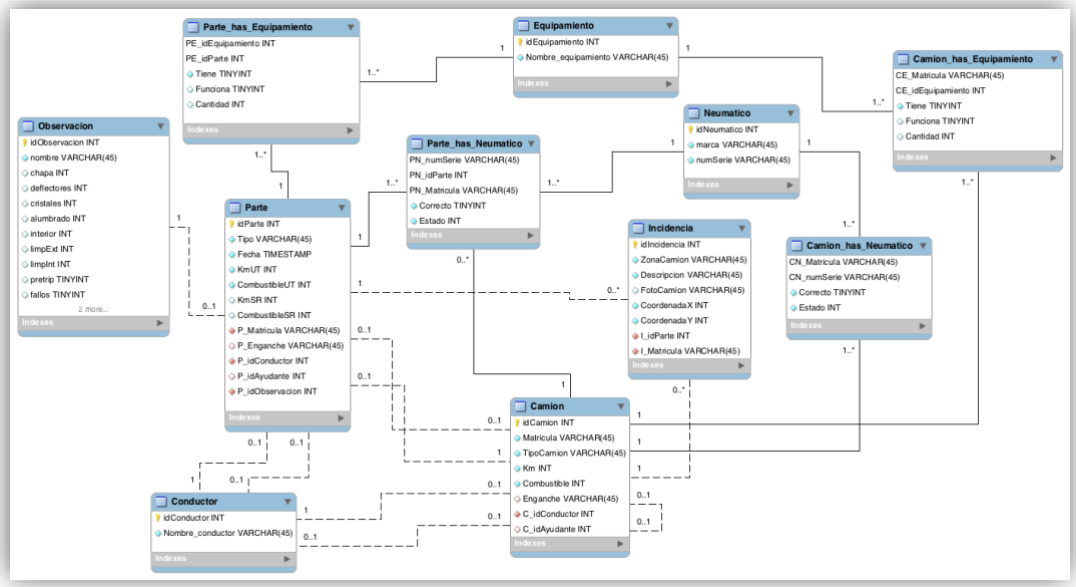


Ilustración 22 - 3er Diseño de la base de datos

2.3.3. Formación XOne

La plataforma XOne es de carácter privado, y ofrece formación a las empresas que contratan, esta abarca una toma de contacto con el IDE de desarrollo a utilizar, las tecnologías que lo componen, etc.

La herramienta XOneCloud es un entorno de desarrollo integrado web, es necesaria una cuenta de usuario para entrar a la aplicación (*ver ilustración* 23).

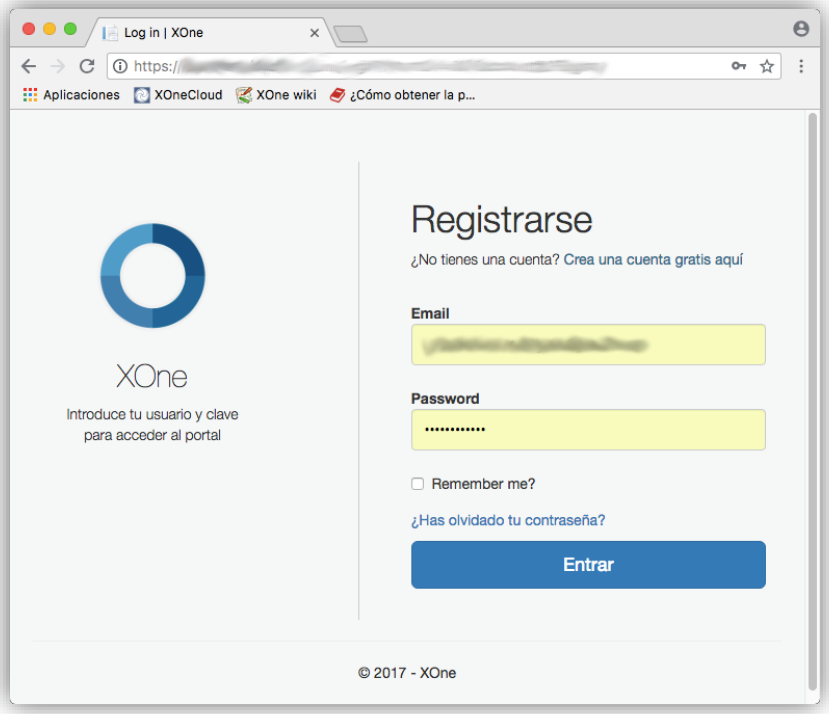


Ilustración 23 - Inicio de sesión de XoneCloud

Una vez dentro podemos crear un nuevo proyecto a partir de alguna de las plantillas que nos proporcionan (*consultar ilustraciones* 24, 25 *y* 26).

Ilustración 24 - Pantalla principal XoneCloud

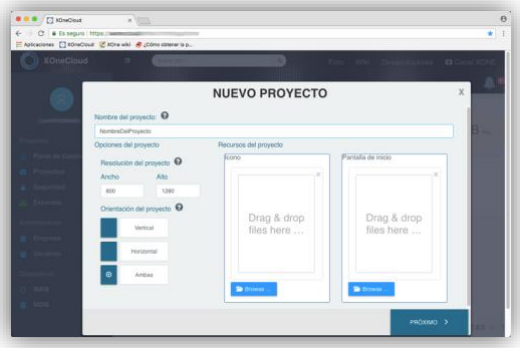


Ilustración 25 - Creando nuevo proyecto

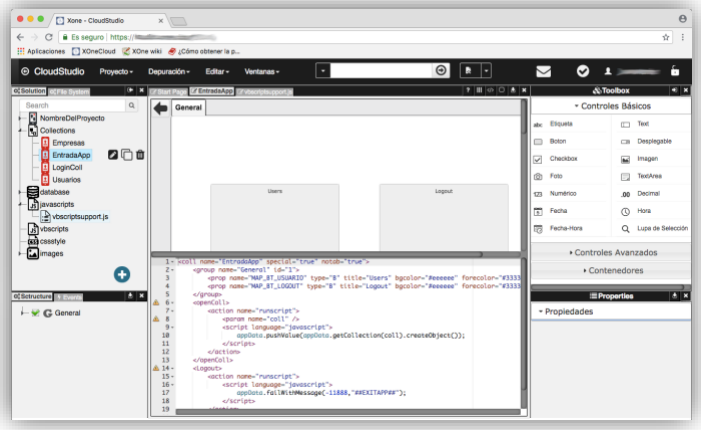


Ilustración 26 - Pantalla principal del IDE.

Como podemos ver es un entorno de desarrollo simplificado, a las necesidades de la programación que con él se desarrolla, nos permite escribir el código XML que representa (abstrae) prácticamente todos los aspectos del desarrollo: construcción de las ventanas con el contenido inherente, interacción con la interfaz, manejo dinámico de los datos, además de permitirnos escribir en JavaScript o VBScript, según nuestras preferencias, para controlar cualquier detalle que no esté implementado ya.

2.3.4. Implementación

Tras haber llevado a cabo todas las tareas previas de análisis y diseño, la acometida del proyecto es muy intuitiva gracias al entorno de desarrollo de XOne, a la izquierda tenemos dos ventanas que tocaremos constantemente: *Solution* y *File System* (*consultar ilustraciones* 27 *y* 28). Desde *Solution* llevamos a cabo el desarrollo tanto visual como lógico de la aplicación.

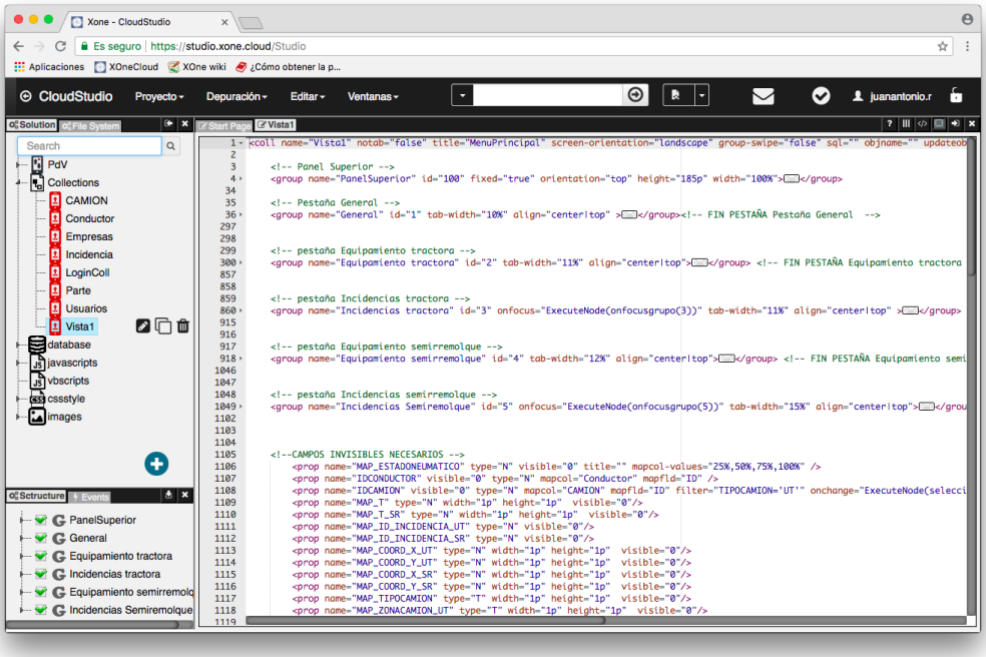


Ilustración 27 - Vista de la ventana Solution

Mientras que desde *File System* tenemos acceso a los ficheros reales del proyecto, pudiendo cargar nuevas imágenes que necesitemos, por ejemplo.

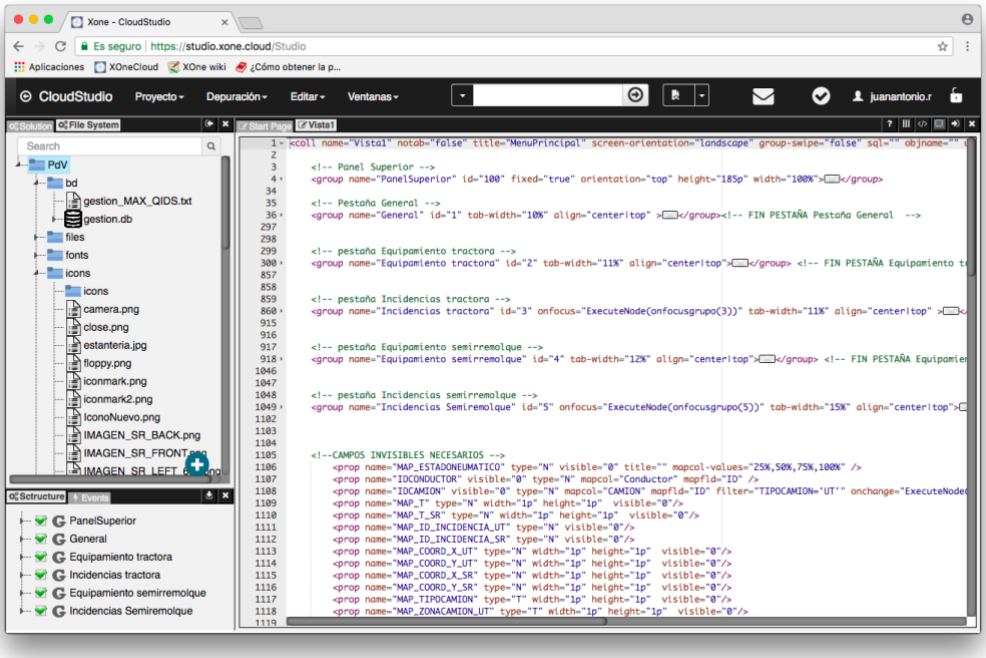


Ilustración 28 - Vista de la ventana File System

En las imágenes anteriores vemos un extracto de la vista principal que hemos diseñado para todo el proyecto, se trata de una estructura en forma de menú de pestañas (como se aprecia en los mock ups). En cada pestaña se establecen los cuestionarios pertinentes.

El entorno de desarrollo tiene un hándicap en el control de versiones, aspecto del cual no dispone, para salvar esta situación, hemos hecho uso de las opciones de exportar e importar del entorno. Podemos descargarnos el proyecto en un fichero *.Zip*, como se muestra en la siguiente ilustración (*ver ilustración* 29).

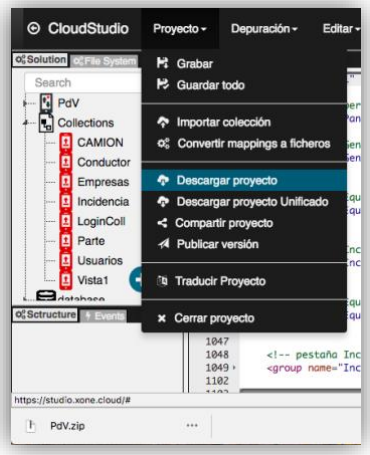


*Ilustración 29 - Exportar/Descargar proyecto *.Zip*

Con el proyecto descargado hemos gestionado el control de versiones creando un repositorio Git local (*ver ilustración* 30), el cual hemos aprovechado para gestionar también el resto de material asociado a este TFG, como es este mismo documento, imágenes de la aplicación, etc.

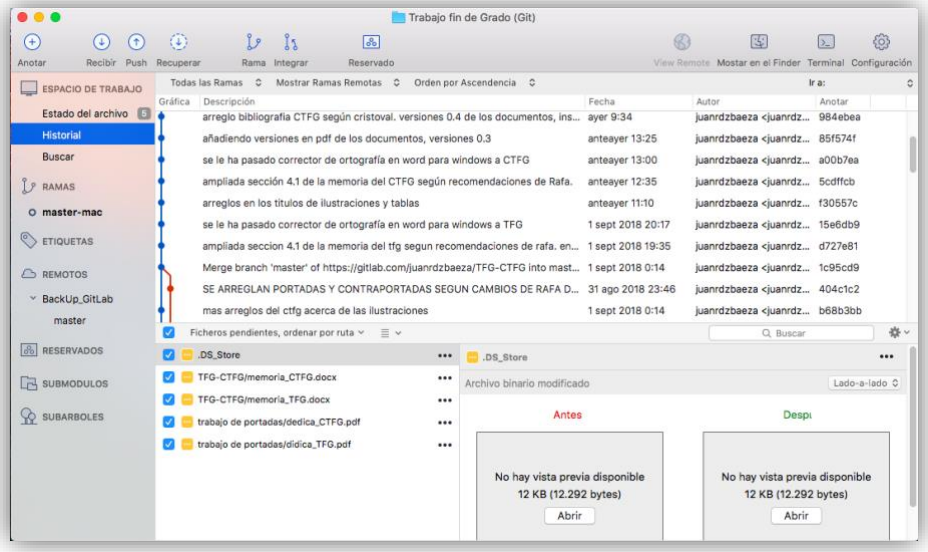


Ilustración 30 - Vista del repositorio Git local

3. Herramientas y tecnologías.

En este apartado se hace una breve descripción de las tecnologías, herramientas y programas que se usan a lo largo del proyecto.

3.1. Informes y documentación.

En cualquier proyecto se hace necesaria la documentación, para este que nos atañe se han usado las siguientes herramientas, para la redacción y arreglo de esta memoria, así como del resto de material que lo acompaña.

3.1.1. Balsamiq Mockups 3

Balsamiq Mockups18 es un software para maquetación de interfaces de usuario a través de un editor WYSIWYG ("what you see is what you get") que nos permite arrastrar y soltar componentes para ir dibujando la interfaz. Los componentes pueden ser enlazados para permitirnos "navegar" por los diseños como si se tratase de la interfaz real (Balsamiq, 2018).

Justificación de uso

Para realizar los dibujos de la interfaz en lugar de hacerlo a mano alzada, por ejemplo, es mucho más cómodo dibujarlas directamente con una herramienta gráfica destinada a este propósito, ya que eliminamos el traslado de información a través de escáner o algún otro periférico de entrada de imagen. Y en cuanto a porque Balsamiq, ya es conocida del curso de algunas asignaturas de la carrera, es muy intuitiva y fácil de manejar, y nos permite exportar las plantillas tanto en PDF, como en algún formato de imagen (como es PNG) con lo que nos facilita muchísimo manejarlas.

3.1.2. Documentación

Para el traspaso de cualquier información documentada a modo de informe la elección es sin duda el formato de documento portátil PDF19. Considerado por la comunidad, y más aún por la ISO, prácticamente como un estándar.

Justificación de uso

Al estar considerado como un estándar, es muy fácil de leer, cualquier navegador tiene integración con este formato, por lo que hace casi innecesario realizar cualquier cambio en la máquina donde se intente leer un documento en este formato, centrando la atención en lo que verdaderamente interesa, acceder a la información contenida en él. Además, este formato es capaz de portar fácilmente texto, imágenes, tablas, etc.

[18] https://balsamiq.com
[19] https://acrobat.adobe.com/es/es/acrobat/about-adobe-pdf.html

3.1.3. GanttProject

GanttProject[20] es una aplicación OpenSource y Multiplataforma escrita en Java que permite la planificación de proyectos de una forma muy simple e intuitiva a través de diagramas de Gantt (Ganttproject, 2018).

Justificación de uso

Este proyecto no tiene una gran planificación con mucha carga de tareas distintas, ni tampoco con recursos que gestionar para cada tarea, por lo que MS Project, por ejemplo, es una aplicación demasiado poderosa para aplicarla, en cambio GanttProject, que también posee características para manejar tareas y recursos, es bastante más simple, y nos ayuda en este caso de un proyecto no muy complejo, a planificarlo de una manera más concreta.

3.1.4. Gimp

GIMP[21] es un editor de imágenes multiplataforma de código abierto. Proporciona herramientas sofisticadas para la edición de imágenes.

Justificación de uso

El famoso programa de manipulación de imágenes de código abierto se ha usado para la edición de algunas de las imágenes usadas en la interfaz de la aplicación, o incluso a lo largo de este documento, así como en la documentación que acompaña a este proyecto. Es intuitivo y fácil de usar si se tiene un mínimo de conocimientos, y muy conocido por los miembros del proyecto, por lo que se ha elegido esta herramienta en lugar de otros editores y manipuladores de imágenes.

3.1.5. MySQL Workbench

MySQL Workbench[22] es una completa suite que integra desde el diseño hasta el mantenimiento de bases de datos de MySQL, pasando por las diferentes etapas que giran en torno a la ingeniería del software, sobre todo en lo que respecta al back-end. Actualmente es propiedad de la multinacional ORACLE.

Justificación de uso

Uno de los aspectos destacados de esta herramienta es la del diseño de bases de datos desde un visual diagrama entidad relación, a través de ingeniería directa podemos desplegar un esquema de base de datos, e incluso podríamos hacer ingeniería inversa para extraer el diagrama entidad relación desde un esquema.

[20] https://www.ganttproject.biz
[21] https://www.gimp.org
[22] https://www.mysql.com/products/workbench/

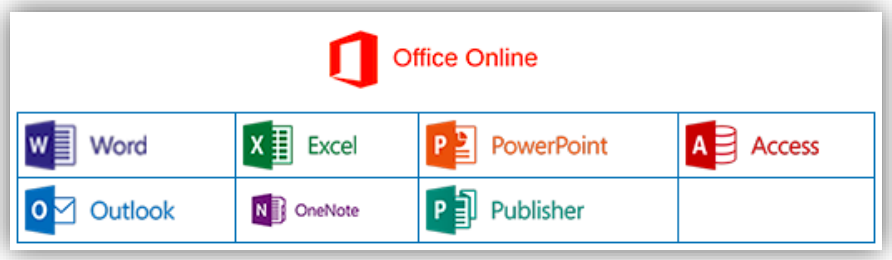
Nuestra elección de MySQL Workbench es debida a este módulo de diseño de diagramas, que nos permite generar fácilmente el correspondiente a la aplicación de este TFG.

3.1.6. Paquete Microsoft Office

Gracias al Acuerdo Microsoft Office 365 ProPlus - UAL[23]

> *"La Universidad de Almería, mediante convenio con Microsoft, proporciona a todos los miembros de la comunidad universitaria el paquete Office365 ProPlus de forma gratuita. Se permitirá, a cada usuario, instalarlo hasta en 5 dispositivos personales: PC, MAC, tabletas, iPad, iPhone y móviles Android."*

La comunidad universitaria puede hacer uso del Paquete Office[24].



Para la elaboración de este TFG se han usado principalmente:

3.1.6.1. Outlook

Microsoft Outlook es un cliente de correo y gestor de calendarios y tareas desarrollado por Microsoft, disponible como parte de la suite Microsoft Office.

Justificación de uso

La comunicación con las personas relacionadas con el proyecto; directores, tutores, colaboradores, etc. Habría sido muy complicada si dependiera en exclusiva de reuniones en físicas. El correo electrónico es de gran ayuda en estas situaciones ya que aporta una comunicación asíncrona con histórico de conversaciones.

Toda la potencia de la herramienta ofimática por excelencia es más que suficiente como justificación de uso, aun así, existen otros clientes de correo muy famosos como son Mozilla Thunderbird o Mail de macOS, ¿Por qué Outlook? Porque ya se usaba el paquete Office para sus otras herramientas y Outlook estaba ahí.

[23] http://cms.ual.es/UAL/universidad/serviciosgenerales/stic/servicios/servicio/OFFICE365PROPLUS
[24] https://products.office.com/es-es/products

3.1.6.2. PowerPoint

Microsoft PowerPoint es un programa de presentación desarrollado por Microsoft, disponible como parte de la suite Microsoft Office.

Justificación de uso

Existen alternativas, por ejemplo, MS Sway[25] (también de Microsoft), Prezy[26] que en su sitio oficial venden como mejor que PowerPoint. Dado su amplio uso durante años, PowerPoint es muy conocida, y se hace muy fácil e intuitivo su uso para realizar cualquier presentación.

3.1.6.3. Word

Microsoft Word es un programa informático orientado al procesamiento de textos desarrollado por Microsoft, disponible como parte de la suite Microsoft Office.

Justificación de uso

Cuando ha sido necesaria la redacción de algún documento aclarativo o descriptivo acerca del proyecto, documentación de la aplicación, además de la redacción de este informe. El procesador de textos Word es en mi humilde opinión insuperable.

3.1.7. Trello

Trello[27] es un software de gestión de proyectos o tareas de tipo kanban, nos permite crear y gestionar tableros con listas de tareas, podemos ir arrastrando estas tareas de una lista a otra fácilmente para tener en todo momento una visión perfectamente controlada de que se está haciendo y que está terminado.

Justificación de uso

Para gestionar todas las tareas que han surgido alrededor de la realización de este proyecto, ha sido muy útil el uso de esta aplicación que dispone de cliente de escritorio, su elección se debe a su simpleza, lo intuitiva que resulta y sobre todo a que su uso es gratuito a pesar de no ser open source, existen alternativas muy potentes como es el caso de Jira, pero que son de pago.

[25] https://sway.com
[26] https://prezi.com/es/
[27] https://trello.com

3.2. Lenguajes

Breve repaso a las tecnologías y lenguajes usados para realizar este proyecto.

3.2.1. XML

El Lenguaje de Marcado Extensible XML[28] se usa muy habitualmente en los frameworks de traducción para las interfaces de usuario tanto de XCode como de Android Studio. (w3.org, 2018)

> *"Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere."*

Justificación de uso

En nuestro caso, la elección misma del entorno de desarrollo XOne Cloud Studio exige el uso de este lenguaje de marcado, ya que es el que usa el IDE. Su uso está muy extendido en el contexto del diseño de interfaces de aplicaciones móviles, escribiéndose en ficheros XML los atributos de los diferentes componentes que las conforman, desde los colores a los bordes, pasando por el posicionamiento (Hebuterne, 2018).

3.2.2. UML

El Lenguaje de Modelado Unificado UML[29]

Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el Object Management Group (OMG®, 2018).

Justificación de uso

Para el diseño de los distintos diagramas que han sido necesarios hemos hecho uso de la aplicación Visual Paradigm, cuya lógica de diagramas se basa precisamente en este intuitivo lenguaje.

[28] https://www.w3.org/XML/
[29] http://www.uml.org

3.2.3. JavaScript

JavaScript[30] (abreviado comúnmente JS) (Vigouroux, 2018) es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico (JavaScript.com, 2018).

Justificación de uso

De la misma manera que a XML, JS es usado por el IDE XOne Cloud Studio para las actividades de complejas que se deseen realizar a través de las aplicaciones desarrolladas, tratamiento de estructuras de datos, algoritmos que se ejecuten en caliente al interactuar con la interfaz, llamadas a módulos del sistema operativo (por ejemplo, la cámara del dispositivo), etc.

3.3. Desarrollo

Se describen a continuación las herramientas y tecnologías que han sido necesarias en las diferentes etapas del desarrollo de la aplicación.

3.3.1. Git

Git[31] es un software para control de versiones de repositorios distribuidos (Dauzon, 2018), es de código abierto, muy sencillo de implementar y de usar, pero muy útil y potente en un entorno de desarrollo de software, aunque *permite gestionar versiones de cualquier tipo de fichero en un computador* (Chacon & Straub, 2018) según la documentación oficial:

> *Git es un sistema de control de versiones distribuidas de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia.*

> *Git es fácil de aprender y tiene una huella pequeña con un rendimiento increíblemente rápido. Supera a las herramientas de SCM como Subversion, CVS, Perforce y ClearCase con funciones como ramificación local barata, áreas de preparación conveniente y flujos de trabajo múltiples.*

Justificación de uso

La elección de Git como gestión del control de versiones es además de por el conocimiento de la tecnología, como dice su documentación, Git es muy eficiente desde el punto de vista de los recursos de la máquina, puesto que no necesitamos desplegar ningún servicio si no es necesario, y muy intuitivo desde el punto de vista

[30] https://www.javascript.com
[31] https://git-scm.com

de la experiencia de usuario. Si a su uso agregamos algún cliente con interfaz gráfica su capacidad de información visual se incrementa notablemente.

3.3.2. GitLab

GitLab[32] es una aplicación web para la gestión integral de proyectos software desde el punto de vista de la filosofía Git.

> *GitLab is the first single application for all stages of the DevOps lifecycle. Only GitLab enables Concurrent DevOps, unlocking organizations from the constraints of the toolchain.*

> *GitLab provides unmatched visibility, higher levels of efficiency, and comprehensive governance. This makes the software lifecycle 3 times faster, radically improving the speed of business.*

Justificación de uso

La primera razón de usar la plataforma GitLab es que permite que los repositorios sean de carácter privado, por lo demás, como en el caso de otras herramientas usadas, la preferencia de GitLab se debe al conocimiento del portal y su uso durante los años de la carrera.

3.3.3. Google Chrome

El famoso navegador Google Chrome[33] del gigante californiano es realmente un derivado del proyecto Chromium[34]. Se trata de un proyecto de Open Source

> *The Chromium projects include Chromium and Chromium OS, the open-source projects behind the Google Chrome browser and Google Chrome OS, respectively. This site houses the documentation and code related to the Chromium projects and is intended for developers interested in learning about and contributing to the open-source projects.*

Justificación de uso

Existen una multitud de navegadores web en el mercado, desde los IExplorer de Microsoft y Safari de Apple, pasando por Firefox y derivados, Tor, Opera, etc. La elección de Google Chrome viene en parte supeditada al uso del IDE XOneCloud, que, tras probarlo en varios de los navegadores citados, es en este último en el que daba menos problemas de uso.

[32] https://about.gitlab.com/product/
[33] https://www.google.com/intl/es_ALL/chrome/
[34] https://www.chromium.org

3.3.4. Microsoft SQL Server Management Studio 17

Microsoft SQL Server (Gabillaud, 2017) es una herramienta profesional con toda la potencia del *Structured Query Language SQL*[35] (Lenguaje de consulta estructurado), de la cual podemos elegir entre varias versiones según nuestras necesidades entre las que tenemos una versión de prueba con algunas limitaciones técnicas, pero más que suficiente para desarrollo (Microsoft, 2018).

> *Express Edition*
>
> *SQL Server 2017 Express es una edición gratuita de SQL Server, ideal para el desarrollo y la producción de aplicaciones de escritorio y web, y pequeñas aplicaciones de servidor.*

Justificación de uso

La empresa cuenta con servidores de bases de datos Microsoft SQL Server[36] montados sobre Windows server 2012. Disponen de una versión Standard para trabajos de desarrollo avanzado y producción, y de una versión más pequeña Express para algunas pruebas y desarrollos más embrionarios.

3.3.5. SourceTree

SourceTree[37] es un cliente de la compañía Atlassian, para Git que dispone de interfaz gráfica.

> *A free Git client for Windows and Mac*
>
> *SourceTree simplifies how you interact with your Git repositories so you can focus on coding. Visualize and manage your repositories through SourceTree's simple Git GUI.*

Justificación de uso

También existen múltiples productos para visualizar la gestión de repositorios Git, es muy común que los propios IDEs dispongan ya de una herramienta integrada para el control de versiones, que permita, entre otros métodos, Git. Por defecto, Git puede manejarse por comandos de consola, en Windows nos ofrece un cliente en modo texto que recuerda a una Shell de Unix, en Linux y macOS hace uso de la propia Shell de estos sistemas operativos. La elección del cliente SourceTree se debe al extenso conocimiento de su funcionamiento tras usarlo durante toda la carrera.

[35] https://docs.microsoft.com/es-es/sql/odbc/reference/structured-query-language-sql?view=sql-server-2017
[36] https://www.microsoft.com/es-es/sql-server/sql-server-downloads
[37] https://www.sourcetreeapp.com

3.3.6. SQLite

SQLite[38] es un motor autónomo de bases de datos relacionales, está escrito en C, y hoy en día está ampliamente extendido ya que su versatilidad y ligereza lo hacen perfecto para el desarrollo de aplicaciones móviles (SQLite, 2018).

> *SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. SQLite is the most used database engine in the world.*

Justificación de uso

Al desarrollar aplicaciones móviles, tanto en iOS como en Android existen tres alternativas muy similares para el almacenamiento de los datos: ficheros de texto plano, las funciones "*User Defaults*" y "*SharedPreferences*" en iOS y Android respectivamente, y *Core Data* para iOS, que es Apple define como un framework para gestionar las bases de datos de las aplicaciones, que se apoya en SQLite, y en Android se trabaja directamente con ficheros de base de datos de SQLite. Nuestra elección es clara en este caso, el uso de SQLite nos permite la robustez de las bases de datos, la robustez del lenguaje SQL y la ligereza de este sistema autónomo.

3.3.7. Sublime Text

Sublime Text[39] es un moderno editor de texto con características especiales destinadas al resaltado de código e indentación[40] del mismo, además de ofrecer herramientas de autocompletado que lo acercan a un IDE, sin dejar de ser un editor de texto.

> *"Sublime Text is a sophisticated text editor for code, markup and prose. You'll love the slick user interface, extraordinary features and amazing performance."*

Justificación de uso

Como pasa con otras herramientas usadas en este proyecto, existen alternativas más simples como el bloc de notas de Windows, TexEdit de Mac, o incluso teniendo en cuenta que estamos hablando de texto plano, editores de texto por consola como Nano, estas opciones las descartamos por la magnitud de los documentos que estamos editando. Del otro lado están los editores modernos como Visual Studio Code[41], Atom[42], Brackets[43], de entre los cuales elegimos Sublime Text primero porque después de años utilizándolo ya es nuestro editor de texto para

[38] https://www.sqlite.org/index.html
[39] https://www.sublimetext.com
[40] Indentación es un anglicismo de indentation, no reconocido por la RAE, es muy usado en el desarrollo de código para referirse al sangrado de este.
[41] https://code.visualstudio.com
[42] https://atom.io
[43] http://brackets.io/index.html

absolutamente todo, y porque no deja de ser simple de usar y a la vez potente para cualquier lenguaje.

3.3.8. Visual Paradigm

Este es un software de modelado UML que nos permite analizar, diseñar, codificar, probar y desplegar. Dibuja todo tipo de diagramas UML, genera código fuente a partir de dichos diagramas y también posibilita la elaboración de documentos (Visual Paradigm, 2018).

Justificación de uso

Existen alternativas a Visual Paradigm[44], gratuitas y de pago, incluso online, el lenguaje UML es ampliamente usado por la comunidad de desarrolladores de software, hemos elegido VP por haberlo usado a lo largo de la carrera en multitud de ocasiones en distintas asignaturas, lo que hace que sea bastante conocido, quizás para este proyecto no se ha utilizado toda la potencia de esta gigantesca herramienta. Además, el acuerdo que mantiene la UAL para la Academic Partner Program de Visual Paradigm nos permite tener una licencia académica del producto.

3.3.9. XOne Cloud Studio

XOne Cloud Studio[45] es una herramienta colaborativa de desarrollo web pensada para facilitar el trabajo a los programadores en XOne, se trata de un Entorno de Desarrollo Integrado, más conocido por sus siglas en ingles IDE (integrated development environment). Su principal característica es que se trata de una herramienta web que permite la edición colaborativa (XOne, 2018).

Justificación de uso

La empresa dispone de un contrato de uso con la compañía XOne, con lo que disponemos de cuentas Gold, existen alternativas enfocadas al desarrollo de aplicaciones hibridas, como ya comentamos en el punto 1.1, en el caso de XOne, se trata de una compañía nacional, que ofrece un buen soporte a través de correo electrónico, videoconferencias, foros de ayuda propios, etc. Además de existir una relación comercial previa al comienzo de este proyecto.

[44] https://www.visual-paradigm.com
[45] http://xoneisp.com/web/desarrolla-tu-app-con-xone-cloud-studio/

4. Ejecución del proyecto

Nuestro "ecosistema"[46] de desarrollo, ya cargado de herramientas, se completa con el framework de carga de aplicaciones que pone a nuestra disposición la compañía XOne, este se encarga de conectar el IDE con la tableta de desarrollo vía wifi, el requisito es que tanto tableta como computador estén dentro de la misma estructura de red.

Desde el IDE web, cuando queremos lanzar la aplicación simplemente seleccionamos el modo depuración, anclamos el dispositivo que estemos usando e iniciamos la depuración, lógicamente el dispositivo debe llevar instalado el framework correspondiente que se puede obtener en la misma pantalla del modo depuración a través de un código QR. En la siguiente ilustración se aprecia el detalle $(ver\ ilustración\ 31)$.

Ilustración 31 - Pantalla de depuración en dispositivo.

Podemos observar que es posible asignar un dispositivo manualmente añadiendo su dirección IP o escaneando el código QR de la izquierda de la imagen, si aún no tuviéramos instalado el framework en la Tablet podríamos buscarlo a través del correspondiente código de la parte derecha, una vez este la tableta enlazada, podemos copiar todos los ficheros de la aplicación si se trata de un primer despliegue por ejemplo, o solo algún cambio si estamos

[46] Nos referimos con ecosistema de trabajo al conjunto del IDE y de las herramientas necesarias para la consecución de este TFG.

haciendo labores de depuración, en la siguiente imagen podemos ver una captura de pantalla de una Tablet recibiendo la aplicación tras iniciarla depuración $(ver\ ilustración\ 32)$.

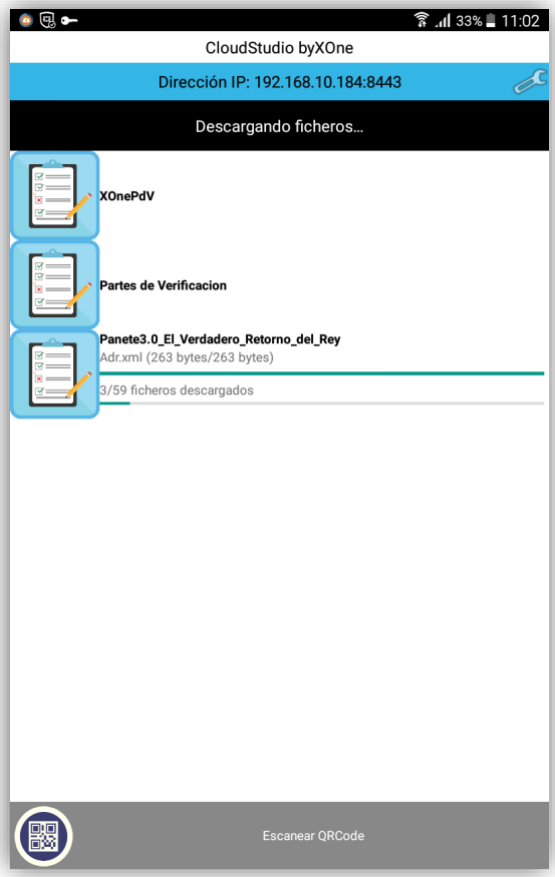


Ilustración 32 - Tablet recibiendo información.

Se puede observar como el proyecto se ha ido desplegando con diferentes nombres en el proceso de desarrollo.

Este modo de depuración tiene sus pros y contras, por un lado, se hace muy cómodo ejecutar la aplicación en un dispositivo real para testearla manualmente cada vez que queramos, pero el modo *debug* en tiempo de ejecución, al que tan acostumbrados estamos, no existe, por lo que la forma de depurar algunas veces pasa por mensajes de alertas en ventanas emergentes, haciéndose algo pesado.

Estamos trabajando con dispositivos Samsung rugerizados cuyo hardware podemos ver en la ilustración 33 $(ver\ ilustración\ 33)$, se trata de dispositivos preparados a conciencia para uso industrial, según la tienda online telefonostodoterreno.es, se trata de *dispositivos que poseen características especiales, siendo tablets resistentes a impactos, resistentes a partículas de polvo, tablets resistentes al agua y sumergibles*. Y según la página oficial de Samsung[47]:

[47] https://www.samsung.com/es/business/tablets/galaxy-tab-active-8-0-t365/sm-t365nngaphe/

Protección anti golpes

Gracias a la funda protectora rugerizada de Galaxy Tab Active, ya no tendrá que preocuparse de golpes o caídas ocasionales. El tablet está diseñado para soportar caídas de hasta 1,2 metros cuando lleva su funda puesta.*

Resistencia al agua y al polvo (IP67)

Rinda al máximo en cualquier entorno con su Galaxy Tab Active. La certificación IP67 garantiza una resistencia de su tablet al polvo y al agua.

La certificación IP[XY], o grado de protección IP, hace referencia al estándar internacional IEC 60529[48] Degrees of Protection, se usa frecuentemente en para catalogar equipamiento electrónico en el ámbito industrial, como pueden ser medidores o sensores.

El estándar cuantifica de una manera numérica dos factores, la entrada de sustancias en suspensión medida por su tamaño en mm (sería la X), y la resistencia a la humedad y/o a líquidos (representada por la Y) (*consultar tabla* 34).

Tabla 34 - Tabla de referencia para IP e IK

IPX[ ]	Tamaño del objeto entrante	IP[ ]Y	Protección frente a	IK XX	Energía de impacto en Julios
0	Sin protección	0	Sin protección	00	Sin protección
1	> 50 mm	1	Goteo de agua	01	0,15
2	> 12 mm	2	Goteo de agua fuerte	02	0,2
3	> 2,5 mm	3	Agua nebulizada (spray)	03	0,35
4	> 1 mm	4	Chorro de agua ligero	04	0,5
5	Protección contra polvo	5	Chorro de agua medio	05	0,7
6	Protección fuerte contra polvo	6	Chorro de agua potente	06	1
		7	Inmersión completa en agua	07	2
		8	Inmersión completa y continua en agua	08	5
				09	10
				10	20

[48] IEC 60529:1989+AMD1:1999+AMD2:2013

Como vemos en la tabla anterior también existe el estándar IK, para cuantificar la resistencia al impacto, este hace referencia a la norma IEC EN 62262[49]. En este caso no se especifica en la página oficial que nivel IK posee el dispositivo.

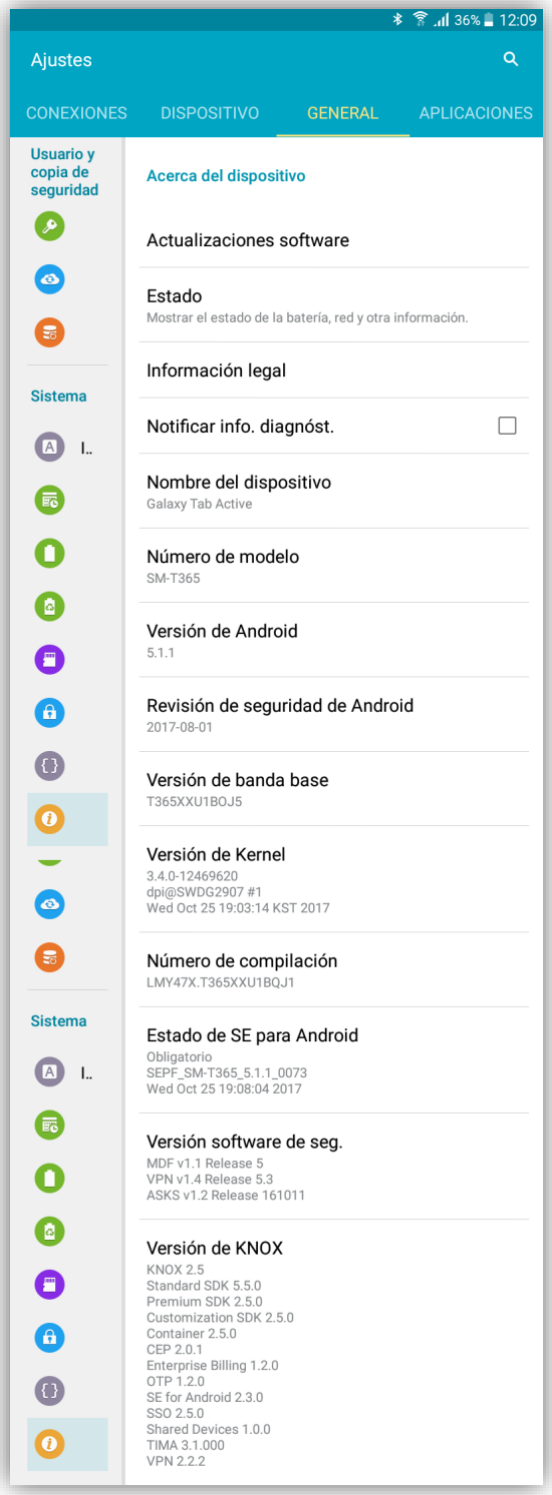


Ilustración 33 - Hardware del proyecto.

[49] IEC 62262:2002

4.1. Fase de desarrollo

A partir de los mock-up de la fase de diseño, se ha realizado la interfaz de usuario, se trata de una vista completa en un fichero .XML que contiene una cabecera o parte fija, con un menú horizontal a modo de pestañas, que nos permite navegar por las distintas acciones de la aplicación, al existir ya un producto de escritorio, se ha tratado de ser lo más fiel posible al diseño original

A continuación, podemos ver un extracto del fichero que contiene la vista de la aplicación (*ver ilustración* 34).

Ilustración 34 - Extracto general del código de la vista

Podemos observar que las pestañas se corresponden a las líneas 48, 312, 872, 930 y 1061, mientras que el panel superior marcado como fijo en la zona superior actúa a modo de cabecera, a continuación, una captura de pantalla de la aplicación en ejecución (*ver ilustración* 35).

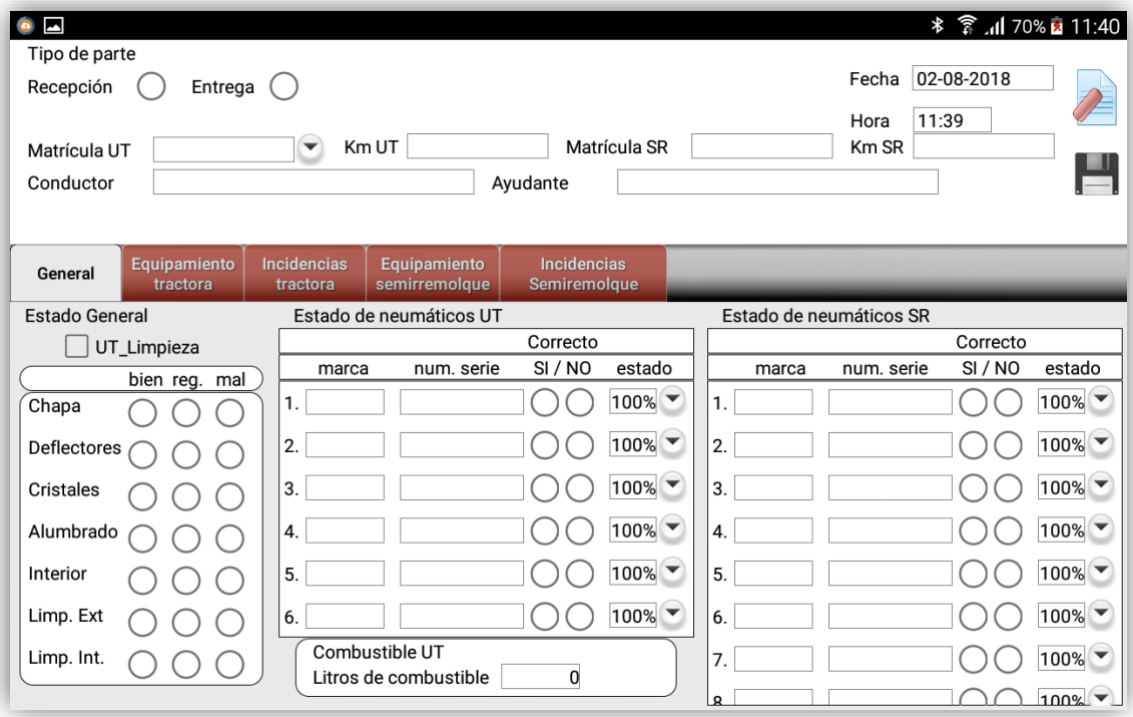


Ilustración 35 - Captura de pantalla de la aplicación en ejecución (modo desarrollo)

En la imagen podemos ver la zona del panel superior y la zona del menú donde se aprecian las pestañas.

Además, se ha diseñado una pantalla de login que será la forma de iniciar sesión en la aplicación, la estructura del proyecto nos permite desactivar esta vista para hacer más ágil el desarrollo, en la siguiente imagen podemos ver un extracto del código del fichero de configuración global donde se declaran las distintas vistas (*ver ilustración* 36).

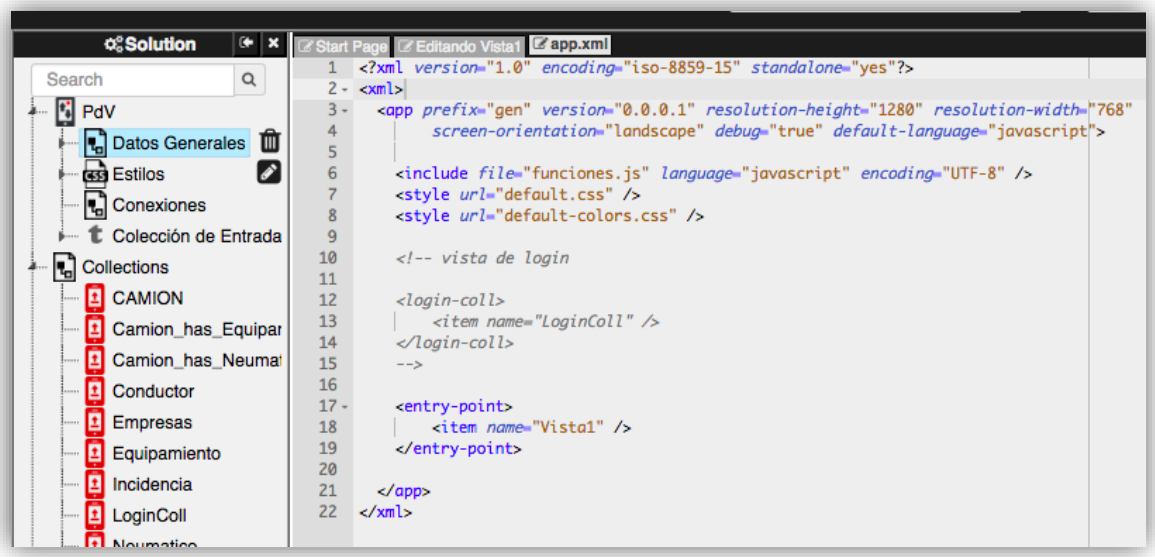


Ilustración 36 - Fichero de configuración global.

Vemos que la vista de login esta comentada, si la des comentamos y volvemos a lanzar la aplicación nos pedirá que iniciemos sesión (*ver ilustración* 37).

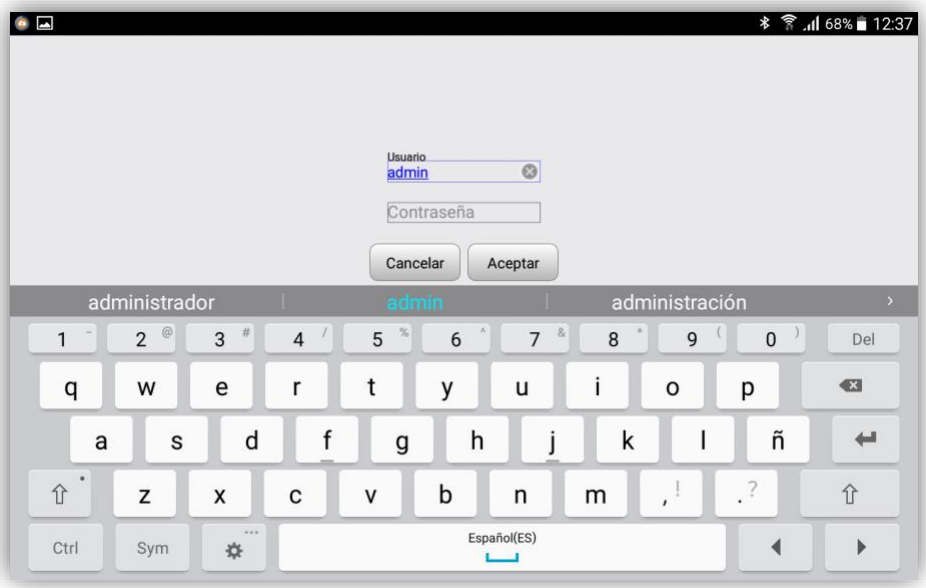


Ilustración 37 - Captura de pantalla de la pantalla de login.

Con esta explicación preliminar podemos entender de una manera superficial el modo de trabajo que se ha de seguir en el desarrollo de una aplicación a través de XOneCloud.

A continuación, mostramos imágenes de la aplicación en ejecución (*consultar ilustraciones de la 38 a la 45*).

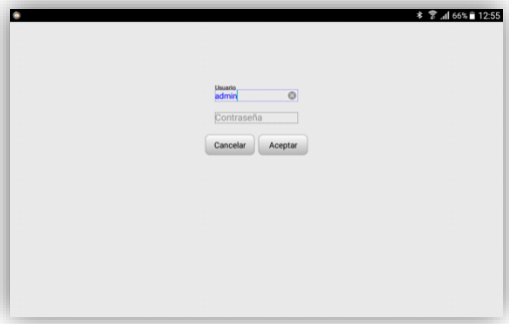


Ilustración 38 - Vista inicio sesión

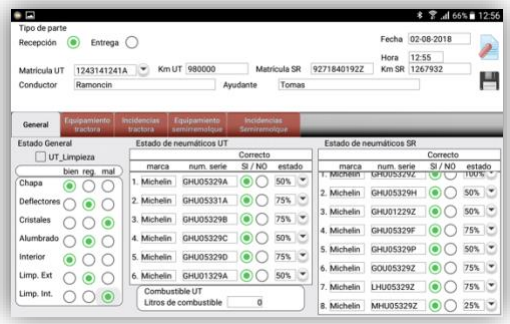


Ilustración 39 - Vista pestaña general

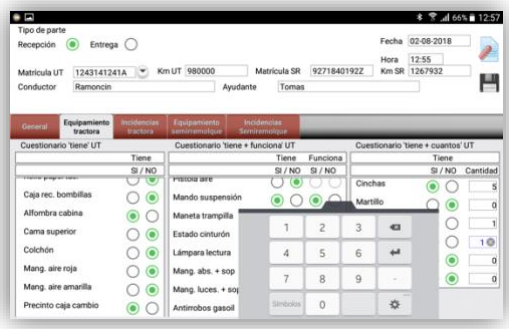


Ilustración 40 - Vista pestaña equipamiento tractora

Ilustración 41 - Vista pestaña incidencias tractora

Ilustración 42 - Vista ventana emergente para agregar incidencia (posibilidad de adjuntar foto)

Ilustración 43 - Vista pestaña incidencia tractora con marca de incidencia existente

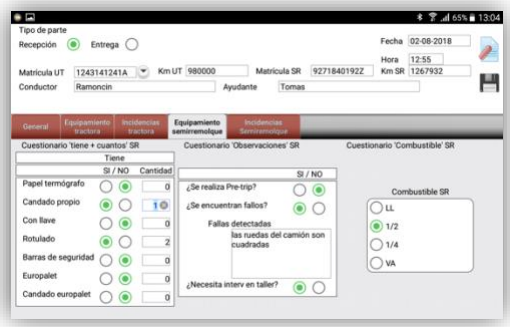


Ilustración 44 - Vista pestaña equipamiento semirremolque

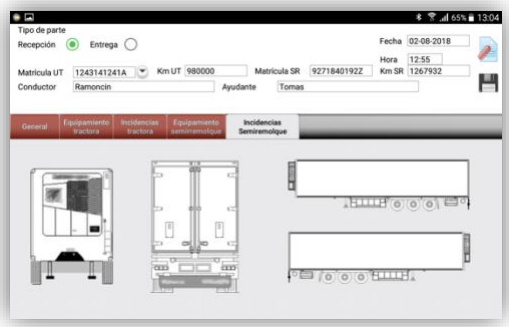


Ilustración 45 - Vista pestaña incidencias semirremolque

Para las incidencias del semirremolque la acción de añadir incidencia es idéntica a la de la cabeza tractora, por lo que no se ha añadido imagen de ilustración del proceso.

Vamos a intentar explicar un poco mejor la integración de la interfaz de usuario con el funcionamiento interno de la aplicación. Como ya hemos visto, la interfaz está escrita en XML, y dividida en bloques dentro de la vista, esos bloques representan cada pestaña de la aplicación, y contienen los campos necesarios para manejar la información, tanto la que se ve como la que no, esto lo explicaremos un poco más adelante.

Veamos por ejemplo el bloque de código correspondiente al panel superior (*consultar ilustración* 46)

```
<coll   name="Vista1" notab="false" title="MenuPrincipal"   screen-orientation="landscape" group-swipe="false" sql="" objname="" updateobj=""
        progid="" filter="" sort="" special="true" editmask="0" mask="0" cell-selected-bgcolor="#FFC000" selected-item-start-index="-1">

    <!-- Panel Superior -->
    <group name="PanelSuperior" id="100" fixed="true" orientation="top" height="185p" width="100%">
        <frame name="frameTop" scrollbar="false" height="185p" width="100%" align="left|top">
                <prop name="MAP_PARTE" type="TL" visible="1" title="Tipo de parte" width="36%" align="left|top" lmargin="20p"/>
                <prop name="MAP_PARTE_RECEPCION" type="TL" visible="1" title="Recepción" labelwidth="6" lmargin="20p" tmargin="10p"/>
                <prop name="MAP_RECEPCION" type="NC" title=" " width="50p" visible="1" radio-group="1" linkedto="CHECK_PARTE"
                        check-type="radio" lmargin="10p" bit="0" newline="false" onchange="ExecuteNode(actualizarTipoParte);" />
                <prop name="MAP_PARTE_ENTREGA" type="TL" visible="1" title="Entrega" labelwidth="4" lmargin="20p" tmargin="10p" newline="false" />
                <prop name="MAP_ENTREGA" type="NC" title=" " width="50p" visible="1" radio-group="1" linkedto="CHECK_PARTE"
                        check-type="radio" lmargin="10p" bit="1" newline="false" onchange="ExecuteNode(actualizarTipoParte);"  />
                <prop name="CHECK_PARTE" type="N"  visible="0" />
                <prop name="FECHA" type="T" title="Fecha" lmargin="622p" visible="1" fieldsize="9" lpadding="5p" labelwidth="4" locked="true" bgcolo
                <prop name="HORA" type="T" title="Hora" lmargin="963p"  visible="1" fieldsize="5" lpadding="5p" labelwidth="4" locked="true" bgcolor
                <frame name="div-divisor" width="100%" height="2p">
                    <prop name="divisor" type="TL" title=" " width="100%" height="100%" visible="1"/>
                </frame>
                <prop name="MATRICULA_UT" type="T" title="Matricula UT" linkedto="IDCAMION" linkedfield="MATRICULA" showinline="true" showinline-key
                <prop name="KM_UT" type="T" title="Km UT" visible="1" newline="false" lmargin="20p" lpadding="5p" labelwidth="4" fieldsize="9" locke
                <prop name="MATRICULA_SR" type="T" title="Matricula SR" visible="1" newline="false" lmargin="20p" lpadding="5p" labelwidth="8" field
                <prop name="KM_SR" type="T" title="Km SR" visible="1" newline="false" lmargin="20p" lpadding="5p" labelwidth="4" fieldsize="9" locke

                <prop name="CONDUCTOR" type="T" title="Conductor" width="40%" visible="1" lmargin="20p" lpadding="5p" labelwidth="8" fieldsize="10"
                <prop name="AYUDANTE" type="T" title="Ayudante" width="40%" visible="1" newline="false" lmargin="20p" lpadding="5p" labelwidth="8" f
                <frame name="flotnew" top="35p" left="1218p" width="50p" height="65p" floating="true" >
                    <prop name="BTNEW" type="B" visible="1" labelwidth="0" width="100%" height="100%" img="IconoNuevo.png" imgsel="IconoNuevo.png"
                </frame>
                <frame name="floatadd" top="130p" left="1220p" width="51p" height="50p" floating="true" >
                    <prop name="BTADD" type="B" visible="1" labelwidth="0" width="100%" height="100%" img="floppy.png" imgsel="floppy.png"   onclick
                </frame>
        </frame>
        <!-- posible botonera del menu futura
        <frame name="framebotones" scrollbar="false" height="50p" width="100%">
            <prop name="BTGRUPO1" type="B" visible="1" labelwidth="1" width="20%" height="100%" title="General" bgcolor="#000000" forecolor="#00ffff
            <prop name="BTGRUPO2" type="B" visible="1" newline="false" labelwidth="1" width="20%" height="100%" title="Equipamento" bgcolor="#00ffff
            <prop name="BTGRUPO3" type="B" visible="1" newline="false" labelwidth="1" width="20%" height="100%" title="Indidencia" bgcolor="#00ffff
            <prop name="BTGRUPO4" type="B" visible="1" newline="false" labelwidth="1" width="20%" height="100%" title="Equipamento2" bgcolor="#00fff
            <prop name="BTGRUPO5" type="B" visible="1" newline="false" labelwidth="1" width="20%" height="100%" title="Semiremolque" bgcolor="#00fff
        </frame>
        -->
    </group>
```

Ilustración 46 - Fragmento de código correspondiente al panel superior.

En la ilustración anterior podemos ver que el panel superior es un grupo de la vista, que contiene un frame, que a su vez contiene todos los campos y propiedades que se muestran en el panel. Fijémonos en las líneas 8 a la 14, estas corresponden a los campos seleccionables de tipo check-box para marcar el parte como de recepción o entrega del vehículo, lo que hacen estas líneas es mapear la información relevante, para luego poder ser gestionadas a través de JavaScript. Por ejemplo, en este caso, el valor del check-box seleccionado será asignado al campo de la línea 14 CHECK_PARTE, esto que comentamos podemos observarlo en las líneas 9 y 12, con la propiedad *linkedto* estamos haciendo referencia al campo de la línea 14, y el valor que se almacenará es el de la propiedad bit que podemos ver en las líneas 10 y 13, en el caso de la recepción bit vale 0 y para la entrega vale 1.

Ahora bien, supongamos que vamos a comenzar un parte nuevo, el valor de CHECK_PARTE estaría vacío, cuando se seleccione un valor, controlaremos los campos MAP_RECEPCION y MAP_ENTREGA, como vemos en la imagen 46 tienen otra propiedad llamada *onchange* que a través de un ExecuteNode() llama a la función actualizarTipoParte *(consultar ilustración 47)*

```
1495        <actualizarTipoParte show-wait-dialog="false" refresh="false">
1496          <action name="runscript">
1497              <param name="nombre"/>
1498              <script language="javascript">
1499                  actualizarTipoParte();
1500              </script>
1501          </action>
1502      </actualizarTipoParte>
```

Ilustración 47 - Fragmento de código correspondiente a actualizarTipoParte (XML).

Esta función que se encuentra incrustada dentro del propio XML llama a su homóloga que se encuentra en el fichero JavaScript *(consultar ilustración 48)*

```
439 //AL MODIFICAR EL TIPO DE PARTE GUARDA ESA MODIFICACIÓN EN LA BD LOCAL
440 function actualizarTipoParte()
441 {
442     var coll = appData.getCollection("Parte");
443     var obj = coll.findObject("ID = " +self.MAP_IDPARTE);
444
445     if(obj)
446     {
447         obj.TIPOPARTE = self.CHECK_PARTE;
448
449         obj.save();
450     }
451
452     obj=null;
453     coll.clear();
454     coll=null;
455 }
```

Ilustración 48 - Fragmento de código correspondiente a actualizarTipoParte (JavaScript).

Como podemos ver la función rescata la información de la colección Parte, y genera un objeto a partir del parte (en el caso de que exista) cuyo ID en la base de datos coincida con el campo mapeado MAP_IDPARTE, una vez este generado correctamente el objeto parte, se asignará el valor del campo CHECK_PARTE al atributo TIPOPARTE del objeto parte, y se guarda el objeto. Debemos aclarar que cuando llamamos a la función *save()* lo que está sucediendo es una actualización en la base de datos SQLite local. A continuación, mostramos la colección (así es como lo llaman en el entorno de XOne) o lo que en POO llamaríamos la clase Parte (*consultar ilustración* 49)

```
<coll name="Parte" title="" sql="SELECT * FROM ##PREF##Parte" objname="Parte" updateobj="Parte" progid="ASData.CASBasicDataObj" filter="" sort="">
  <group name="General" id="1">
    <prop name="TIPOPARTE" type="N"/>
    <prop name="FECHA" type="DT"/>
    <prop name="KMUT" type="N"/>
    <prop name="COMBUSTIBLEUT" type="N"/>
    <prop name="KMSR" type="N"/>
    <prop name="COMBUSTIBLESR" type="N"/>
    <prop name="TACOGRAFOCORRECCION" type="D"/>
    <prop name="TACOGRAFOREVISION" type="D"/>
    <prop name="KMMANTENIMIENTOUT" type="N"/>
    <prop name="KMMANTENIMIENTOSR" type="N"/>
    <prop name="P_MATRICULA" type="T" linkedto="MAP_IDCAMION" linkedfield="MATRICULA"/>
    <prop name="P_ENGANCHE" type="T" linkedto="MAP_IDENGANCHE" linkedfield="MATRICULA"/>
    <prop name="P_IDCONDUCTOR" type="N" linkedto="MAP_IDCONDUCTOR" linkedfield="ID"/>
    <prop name="P_IDAYUDANTE" type="T" linkedto="MAP_IDAYUDANTE" linkedfield="ID"/>
    <prop name="P_IDOBSERVACION" type="T" linkedto="MAP_IDOBSERVACION" linkedfield="ID"/>

    <!--CAMPOS INVISIBLES NECESARIOS -->
    <prop name="MAP_IDCAMION" visible="0" type="N" mapcol="CAMION" mapfld="ID"/>
    <prop name="MAP_IDENGANCHE" visible="0" type="N" mapcol="CAMION" mapfld="ID"/>
    <prop name="MAP_IDCONDUCTOR" visible="0" type="N" mapcol="Conductor" mapfld="ID"/>
    <prop name="MAP_IDAYUDANTE" visible="0" type="N" mapcol="Conductor" mapfld="ID"/>
    <prop name="MAP_CONDUCTOR" type="T"/>
    <prop name="MAP_AYUDANTE" type="T"/>
    <prop name="MAP_IDOBSERVACION" visible="0" type="N" mapcol="Observacion" mapfld="ID"/>

  </group>
```

Ilustración 49 -Fragmento de código correspondiente a la colección o clase Parte.

Por último, vamos a tratar de explicar un poco el funcionamiento de la aplicación. Para empezar, el operario introducirá sus datos de login (*ver ilustración* 50)

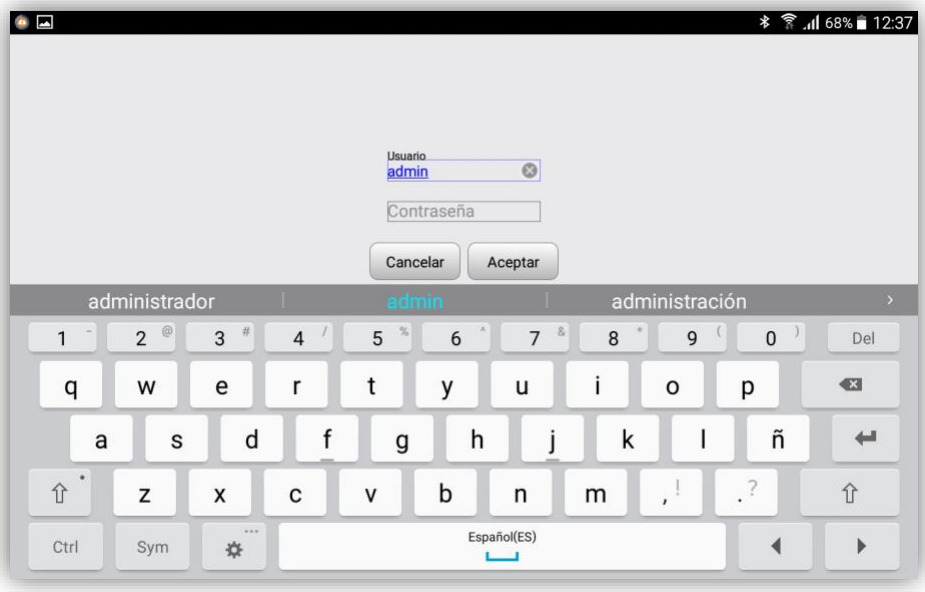


Ilustración 50 - Inicio de sesión.

Una vez dentro seleccionará una matricula de unidad tractora del desplegable correspondiente (*ver ilustración* 51).

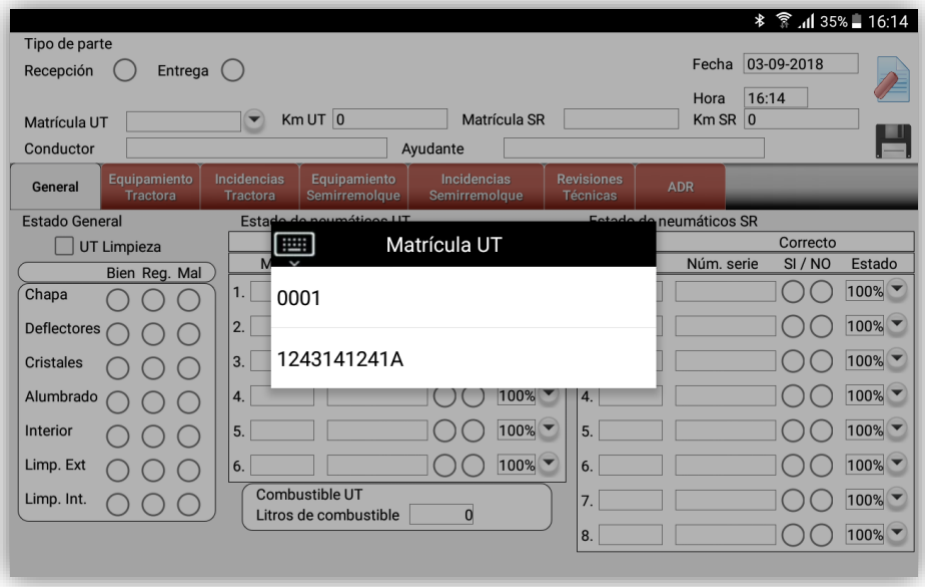


Ilustración 51 - Selección de matrícula.

Debe seleccionar si se trata de un parte de recepción o de entrega, y deberá ir seleccionando un valor para cada check-box de todas las pestañas, en el caso de los neumáticos, la aplicación carga los que esta usando el vehículo y el operario deberá asignar un valor al estado de los mismos (porcentaje), el volumen en litros del tanque de combustible, y cuando corresponda asignar una cantidad al campo correspondiente, consideramos que el uso de la aplicación es muy intuitivo en términos generales (*ver ilustraciones* 52 y 53).

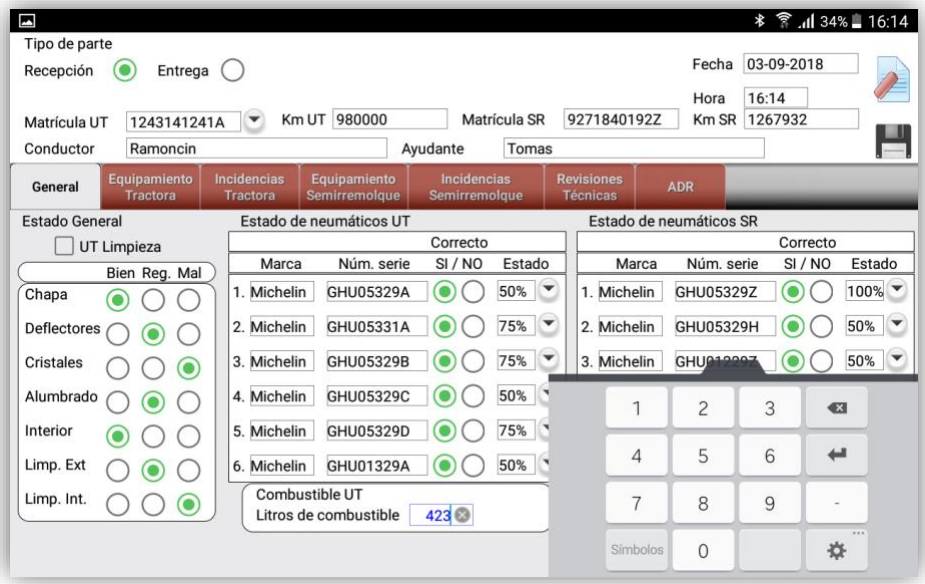


Ilustración 52 - Asignando cvalores check-box.

Ilustración 53 - Asignando más valores y cantidades

Únicamente consideramos que merece una explicación mas exhaustiva el uso de las pestañas *Incidencias Tractora/Semirremolque*, y que, al ser idéntico, bastará con explicar una de ellas, hemos escogido *Incidencias Tractora* por simple orden de las pestañas en el menú.

Para empezar, el operario no podrá agregar una incidencia si no ha seleccionado una matricula, por lo que consideramos que ya se ha seleccionado una de la lista desplegable. Una vez dentro de la pestaña, tendremos a la vista un croquis o esquema de una cabeza tractora por delante, por detrás y por ambos lados (*ver ilustración* 54).

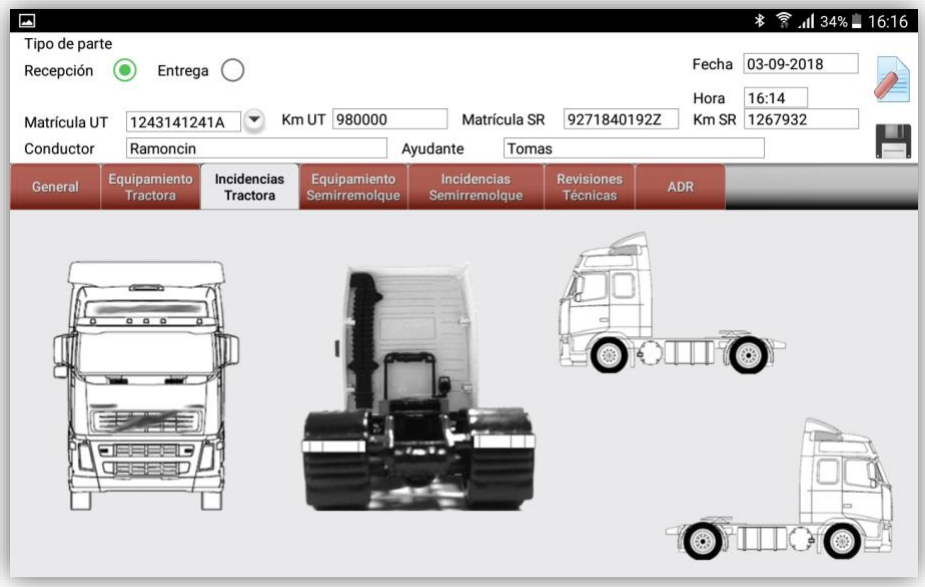


Ilustración 54 - Pestaña de incidencias

Podremos seleccionar el punto de la pantalla que deseemos (dentro del espacio correspondiente al contenido de la propia pestaña) el programa es capaz de detectar con una cierta precisión el lugar seleccionado, y despliega una ventana emergente muy simple (*ver ilustración* 55).

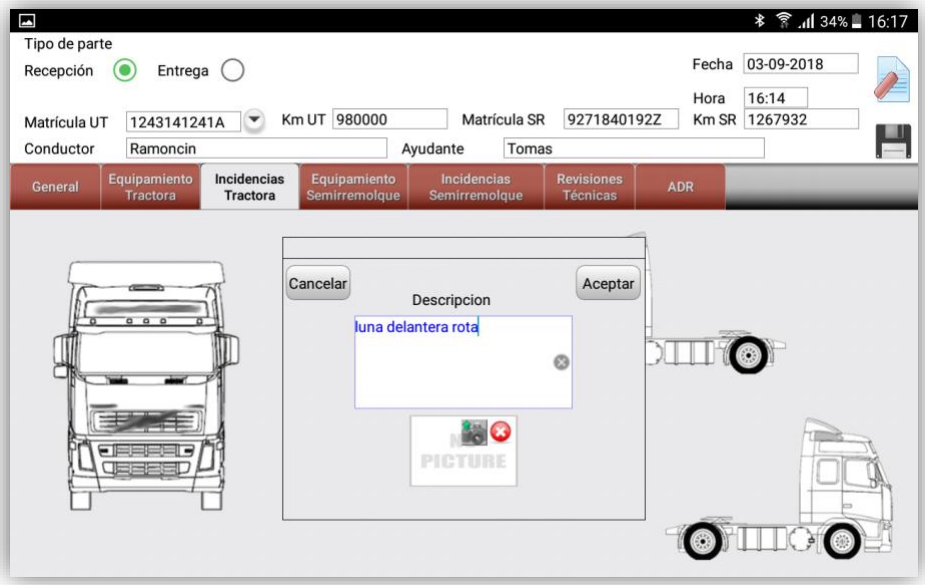


Ilustración 55 - Generando incidencia

Podremos pulsar cancelar arriba a la izquierda, lo que cancela el proceso y hace desaparecer la ventana emergente. Tenemos un campo de texto extendido, para introducir una breve descripción de la incidencia, un botón papelera para eliminar cualquier dato ya introducido anteriormente, un espacio de carga de imágenes, que invoca a la función cámara del dispositivo y que nos permite hacer una foto si procede para completar la información de la incidencia, y un botón aceptar arriba a la derecha para hacer persistente los datos que hayamos introducido (*ver ilustración* 56 y 57).

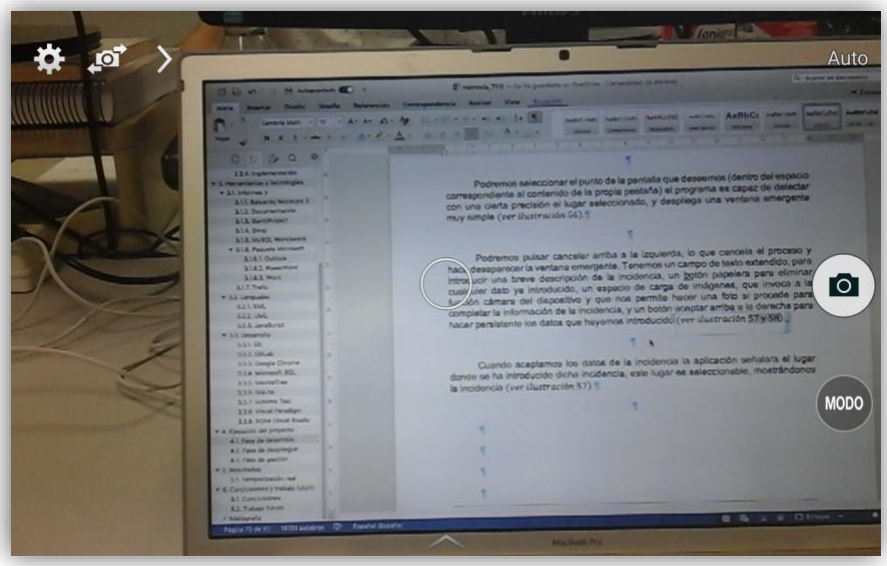


Ilustración 56 - Tomando fotografía para una incidencia.

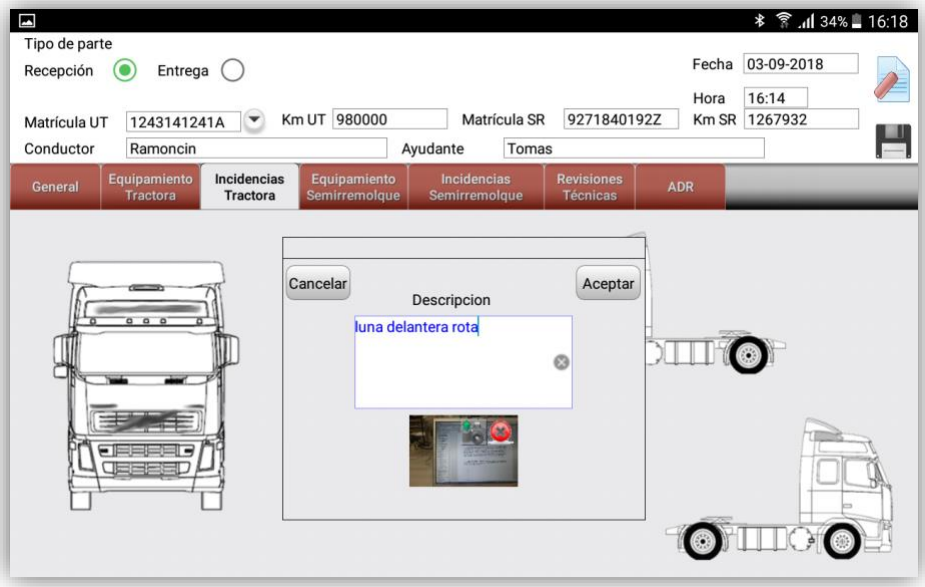


Ilustración 57 - Completando la incidencia

Cuando aceptemos los datos de la incidencia la aplicación señalara el lugar donde se ha introducido dicha incidencia, este lugar es seleccionable, mostrándonos la incidencia (*ver ilustración* 58).

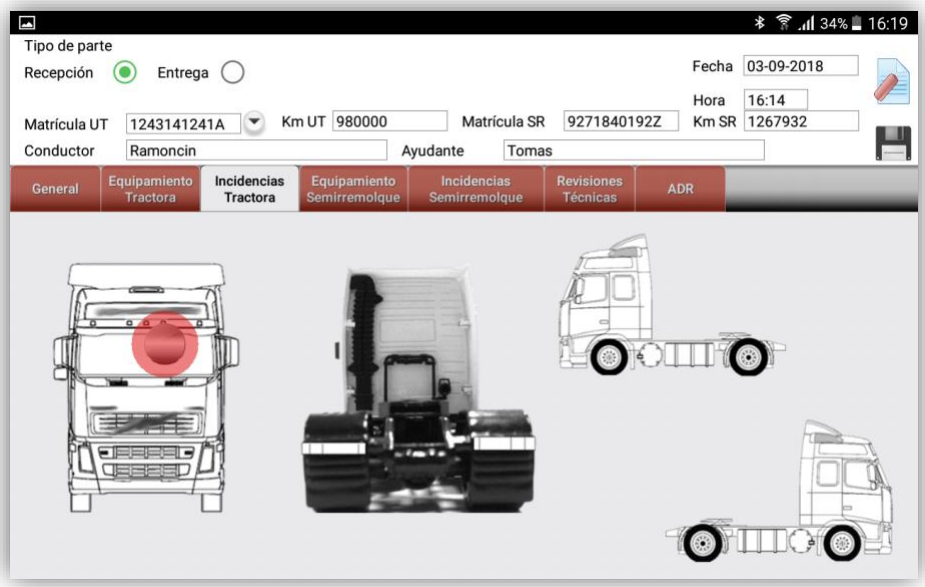


Ilustración 58 - La incidencia ha sido creada

4.2. Fase de despliegue

Una vez concluido el desarrollo de la aplicación, pasamos a la fase de despliegue en producción pre-alfa. Empezando por el despliegue de la base de datos en el servidor Microsoft SQL Server. Para empezar, es necesario contar con unas bases de datos que acompañan los servicios de XOne, para el modo administración y réplica, como ya la compañía dispone de trabajos anteriores realizados de forma externa, ya encontramos estas bases de datos el servidor, a continuación, una captura de pantalla del SQL Server Management Studio donde podemos observar como existen pareadas las bases de datos Manager y Replicator, para los modos desarrollo y producción (*ver ilustración* 59).

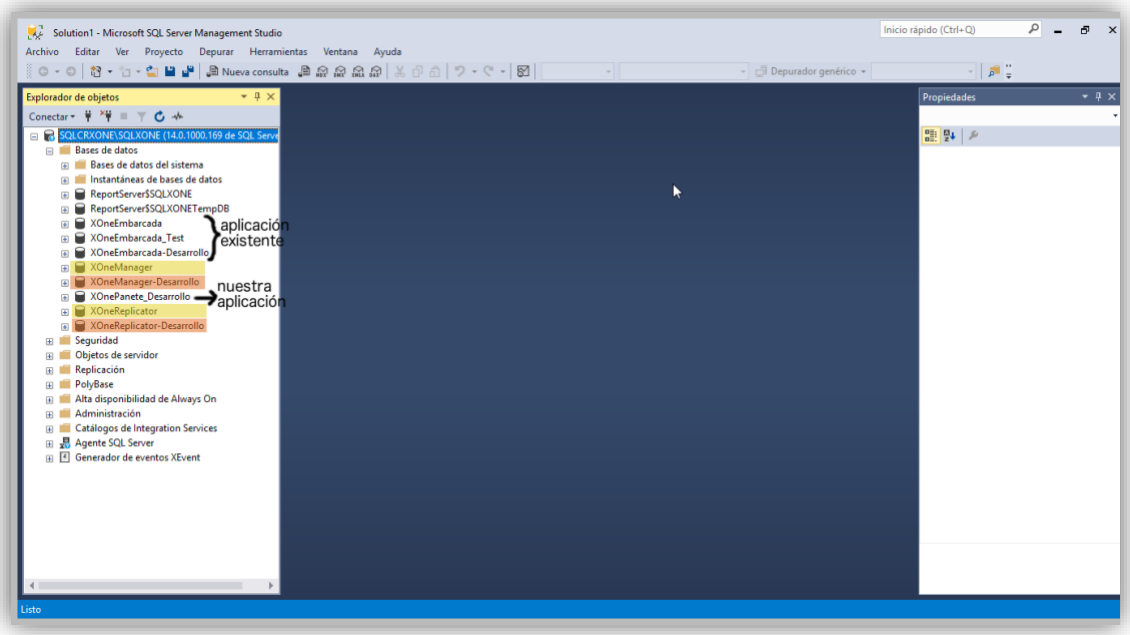


Ilustración 59 - Captura de pantalla de SSMS.

Hemos resaltado en naranja las bases de datos correspondientes a la administración en desarrollo y en amarillo las de producción, estas bases de datos contienen parámetros de configuración correspondientes a XOne para que el servicio funcione correctamente, obviamente nosotros usaremos las resaltadas en naranja XOneManager-Desarrollo y XOneReplicator-Desarrollo.

También podemos ver la base de datos para nuestra app con el nombre en clave del proyecto XOnePanete_Desarrollo. La estructura de este esquema igualmente necesita de una buena cantidad de tablas destinadas a la gestión y administración del servicio, donde se incluyen parámetros como pueden ser la conexión con la propia réplica de la base de datos, una vez generado el esquema "vacío" (con las mencionadas tablas de gestión) pasamos a añadir nuestras tablas extraídas del IDE XOne Cloud, este nos permite exportar la base de datos a un script SQL preparado para el gestor de Microsoft, para MySQL, para Oracle o para SQLite (*ver ilustración* 60).

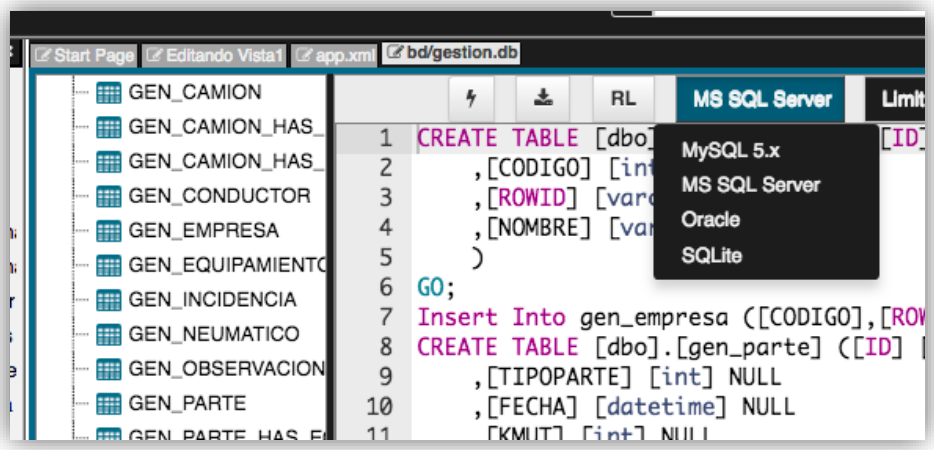


Ilustración 60 - Exportando BD desde XOneCloud.

Quedando la estructura de la base de datos en el servidor como sigue, se expone la captura de pantalla cortada por ser bastante larga, incluso se ha omitido alguna tabla del final para no alargarlo y centrarnos en la explicación (*ver ilustraciones* 61 *y* 62).

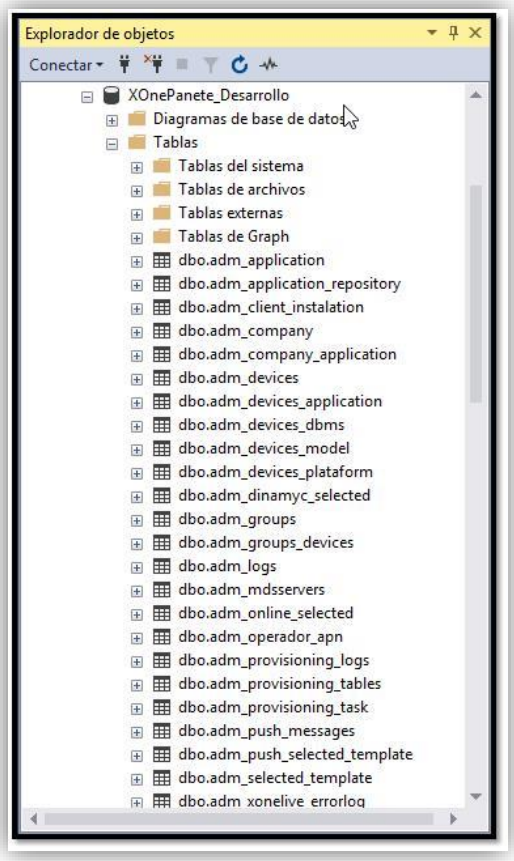


Ilustración 61 - Estructura BD para la app

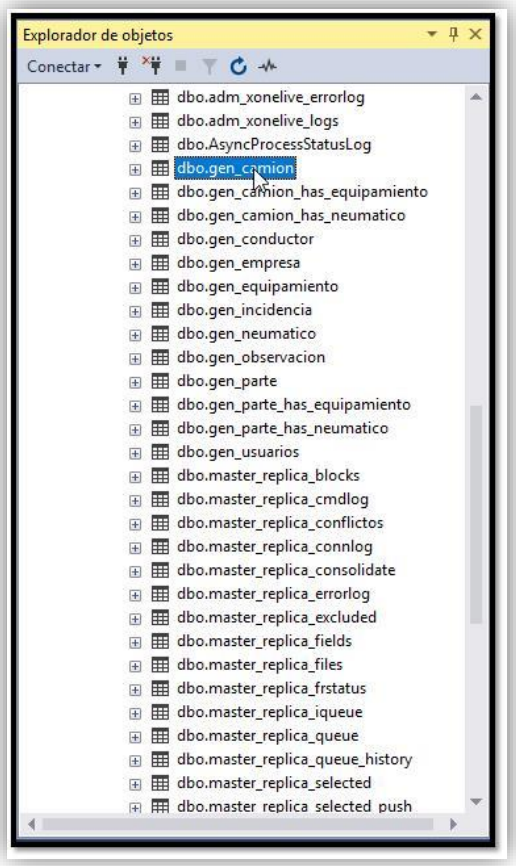


Ilustración 62 - Estructura BD para la app bis

Como podemos observar, es en la ilustración 49 donde aparecen ya las tablas correspondientes a nuestra base de datos, el resto son tablas que contienen información de gestión para el funcionamiento del entorno.

Para que toda esta conectividad tenga sentido, la estructura de red de la empresa para la plataforma de XOne es algo compleja, lógicamente se establecen cortafuegos y medidas de seguridad para impedir en la mayor medida posible los accesos no autorizados, así como la extracción de datos sensibles. A continuación, podemos ver el esquema de comunicaciones que se sigue (*ver ilustración* 63).

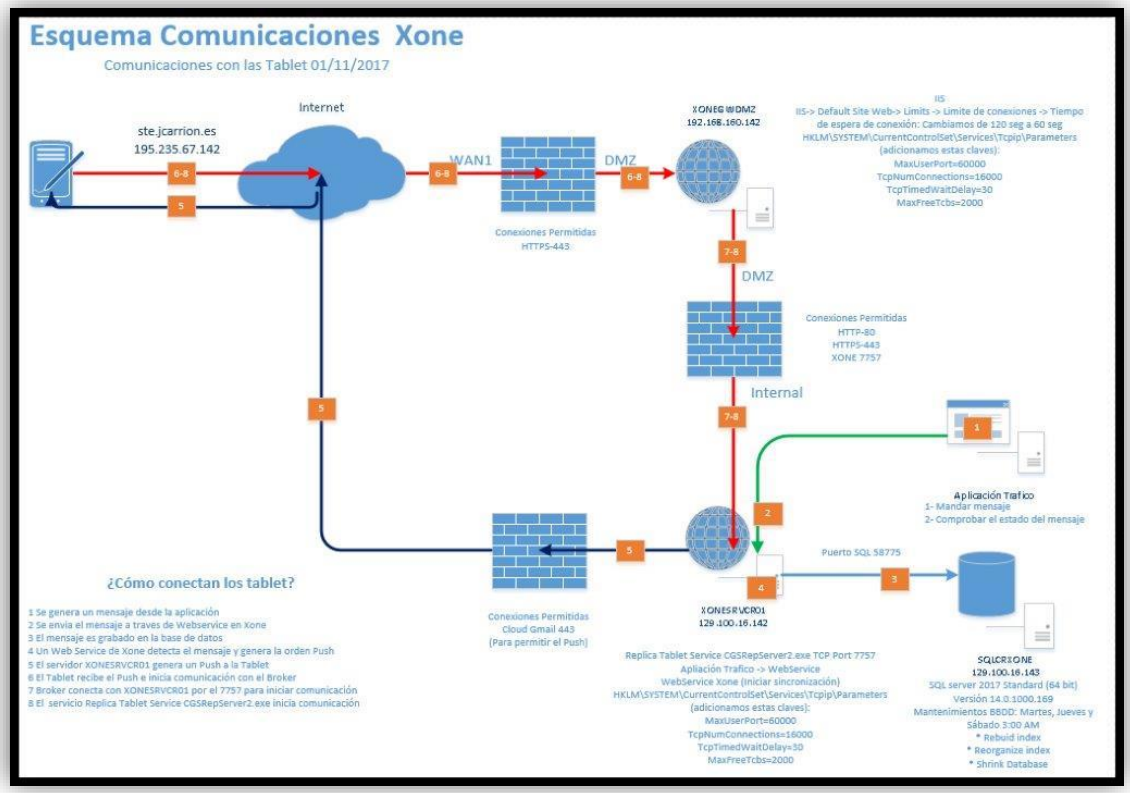


Ilustración 63 - Esquema Comunicaciones XOne (proporcionado por la empresa)

4.3. Fase de gestión

Para administrar tanto el software como el hardware asociado a nuestro proyecto, disponemos de una interfaz web desde la que podemos gestionar las aplicaciones que están disponibles en la plataforma de XOne, además de una administración básica de los dispositivos dados de alta a través de su número IMEI, podemos gestionar las aplicaciones de las que se pueden hacer uso, por ejemplo.

Nota. Por razones obvias no podremos mostrar demasiada información sobre el funcionamiento interno de la plataforma al ser de carácter privado.

Hemos dado de alta nuestra aplicación en la plataforma, a través de las tablas correspondientes en la base de datos *XOneManager-Desarrollo* y *XOneReplicator-*

Desarrollo. Hecho lo cual podremos gestionar todo a través de la interfaz web, a continuación, una captura de pantalla (*ver ilustración* 64).

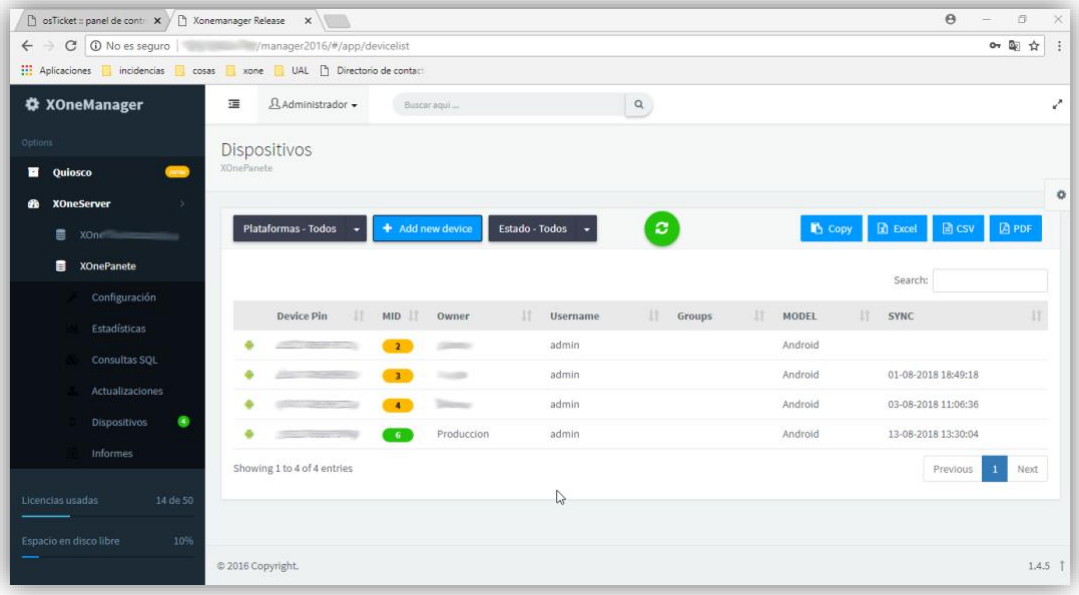


Ilustración 64 - XOneManager para gestión de aplicaciones y dispositivos

En el servidor hemos colocado el proyecto descargado del IDE XOneCloud, en el directorio correspondiente, para que el XOneManager disponga de estos ficheros y sea capaz de lanzarlos al dispositivo que los solicite, tanto ficheros fuente de la app como tablas de la base de datos correspondientes únicamente a la aplicación, generando con ellas un fichero de SQLite que podrá ser igualmente lanzado al dispositivo que lo solicite.

Una vez este todo este proceso correctamente configurado y funcionando, el dispositivo móvil, tal como hemos diseñado la base de datos, tendrá unas tablas de lectura, de donde extraerá la información que necesite, y otra zona de tablas de escritura, donde la aplicación realiza la persistencia de los datos (*ver ilustración* 65), cuando esto ocurre, el sistema de XOne reconoce los cambios y los gestiona a través de dos tablas pertenecientes al esquema de la aplicación, *master_replica_iqueue* y *master_replica_queue*. La primera recibe las notificaciones de cambios, estos se incorporan a la base de datos a través del *insert*, *update* o *delete* correspondiente, y una vez realizado el cambio la notificación pasa a la segunda tabla, existen además una serie de tablas que rodean a este proceso que se encargan de almacenar *logs*, *errores*, *conflictos*, etc.

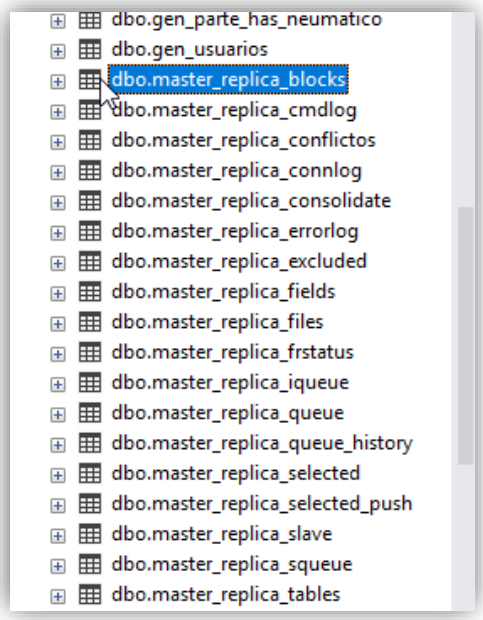


Ilustración 65 - Tablas de gestión de cambios y sincronización entre tablet y SQL Server

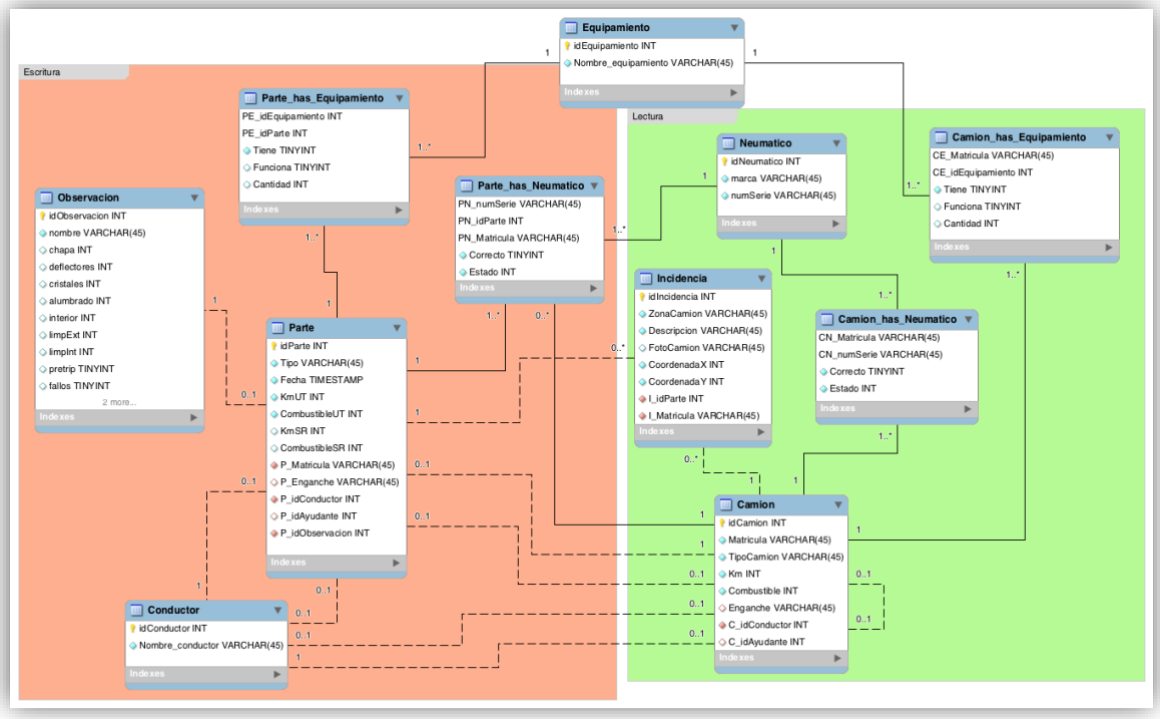


Ilustración 66 - Modelo de la base de datos, zona de lectura en verde. Zona de escritura en rojo

En este punto del proyecto estamos en la siguiente situación.

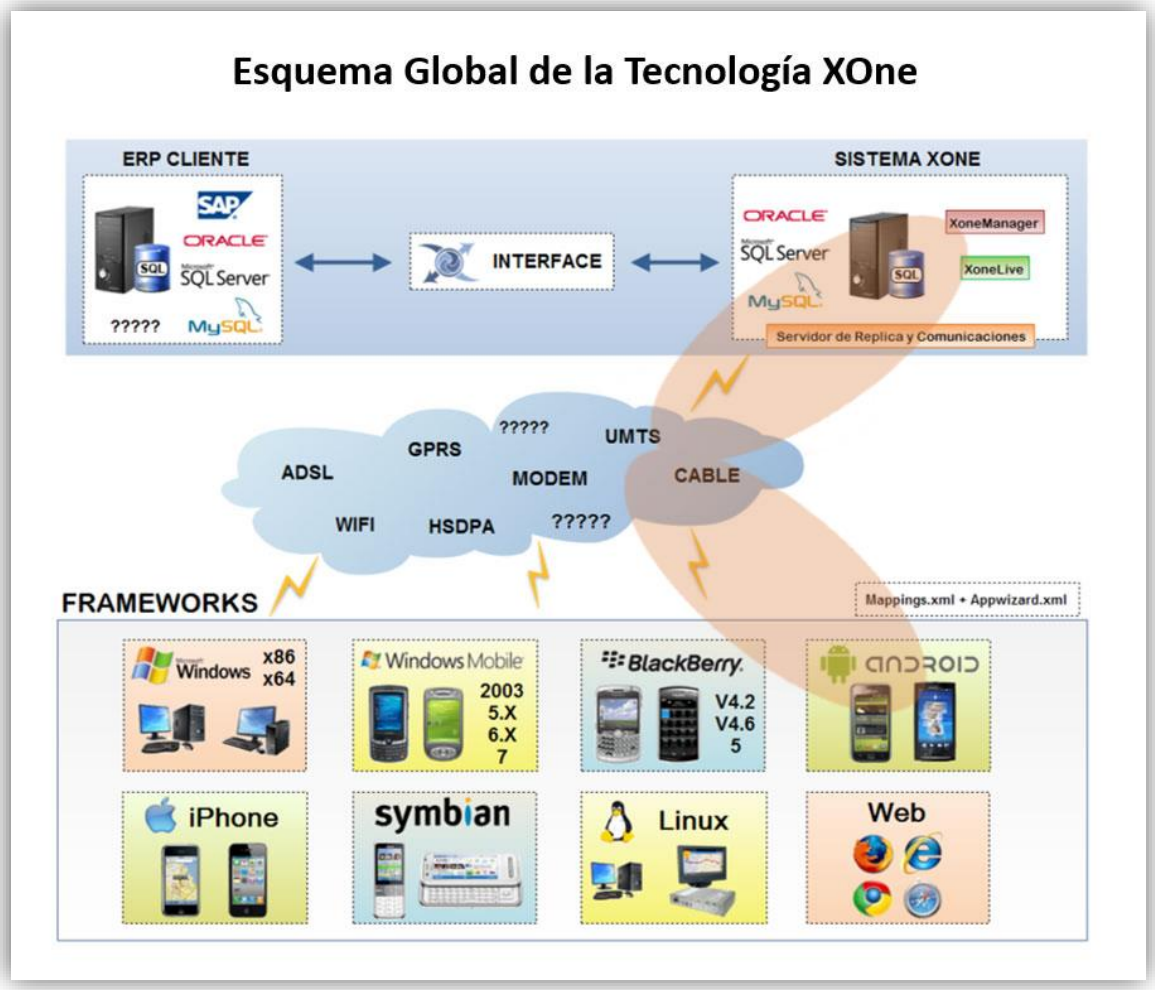


Ilustración 67 - Esquema global de XOne. En naranja estado-hito del proyecto[50].

Como se aprecia en la ilustración anterior (*ver ilustración 67*), hemos pasado por el diseño y desarrollo de la aplicación, y la interconexión y sincronización con el sistema XOne. A partir de aquí queda la unión con el ERP, para lo que se utilizan interfaces de datos, primero interpretaremos correctamente lo que se está introduciendo en el SQL Server en el sistema XOne, para después, a través de interfaces, hacer corresponder los datos con los que se encuentren en el ERP de la empresa, para lo que usaremos, en un principio, algún web service, con la intención de trasladarlo más adelante a otro tipo de *interfaseado* propio de la empresa y del funcionamiento del nuevo ERP que se encuentra en fase de despliegue temprano.

[50] Tecnología XOne Server

5. Resultados

A continuación, vamos a hacer una comparativa entre la temporalización teórica resultado del análisis realizado en el punto 2.1 de este documento que recordamos era como sigue según el diagrama de Gantt:

- Análisis. - 40 horas.
- Diseño. - 30 horas.
- Formación XOne. - 40 horas.
- Implementación. - 140 horas.
- Redacción de la memoria del TFG. - 50 horas.

Temporización teórica total: 300 horas

5.1. Temporización real

En el capítulo 2.1 realizamos una estimación teórica de la temporiza del proyecto usando una planificación clásica a partir de puntos de función y modelo COCOMO, en la que se estimaban 4,9 meses, mientras que el tiempo real invertido en el proyecto han sido unos 3,5 meses.

Los tiempos en semanas por tareas para compararlos con los que se estimaron en el punto 2.1 se desglosan a continuación:

- *Análisis. - Se pretende realizar un análisis preliminar de los requisitos tanto funcionales como no funcionales para la aplicación.* 46 horas.
- *Diseño. - Para el diseño de la interfaz de usuario se pretende seguir la línea de la aplicación de escritorio existente, aun así, se pretende realizar algún mock-up en líneas básicas.* 27 horas.
- *Formación XOne. - La empresa pretende poner a mi disposición un periodo de formación en el entorno de XOne, posiblemente una semana variable.* 12 horas.
- *Implementación. - Codificación de la aplicación a través del entorno de XOne.* 183 horas.
- *Redacción de la memoria del TFG.* - 69 horas.

Temporización real total: 337 horas

Tras comparar los tiempos, vemos que donde se ha reducido más es en la formación, aun así, la implementación no se ha visto drásticamente incrementada.

6. Conclusiones y trabajo futuro

Como capítulo final de este trabajo de fin de grado vamos a hacer un poco de retrospectiva en las formas y el contenido de este. ¿Merece la pena la realización de una aplicación de forma hibrida?, ¿hemos escogido el "*ecosistema*" de trabajo correcto?, ¿y la metodología?, etc.

6.1. Conclusiones

Lo primero que queremos resaltar en este apartado es que ha sido relativamente cómodo trabajar desde un punto de vista tan abstraído, el simple hecho de seleccionar los campos deseados a través de etiquetas XML predefinidas por la compañía XOne, y decidir los atributos necesarios tanto para el aspecto visual como para el funcionamiento interno. Esto hace que desarrollar la interfaz de usuario sea, de algún modo, parecido al desarrollo de una interfaz web, de hecho, es muy parecido el desarrollo de la aplicación completa al de una aplicación web.

Desde el punto de vista de la compañía, es igualmente muy cómodo porque se pueden crear aplicaciones concretas, sin necesidad de diseños sobrecargados, nuestro caso por ejemplo es muy sencillo visualmente.

Si hablamos del back-end pasa algo parecido, la capacidad de XML de asimilar llamadas de funciones en JavaScript, o incluso de aceptar pequeños fragmentos de código es muy similar a como se hace en HTML.

Por todo esto podríamos decir que sí puede merecer la pena escoger un modelo hibrido de desarrollo de aplicaciones móviles, siempre que se pretenda llegar al máximo número de dispositivos. En nuestro particular ejemplo se puede echar en falta la ejecución de la aplicación sobre hardware de Apple, pero, al tratarse de un proyecto real donde el hardware es el que es, además del más que evidente hándicap del precio de los dispositivos de Apple, no ha sido posible realizar estas pruebas.

Por lo que respecta al "*ecosistema"*, el entorno de desarrollo de XOneCloud, aunque cumple bien con su cometido, se echan en falta herramientas tan indispensables como es la opción de depuración en tiempo real, así como herramientas para la gestión del control de versiones. Opciones que realizamos de una manera más manual en este proyecto y que ya han sido comentadas a lo largo de este TFG.

En cuanto a la metodología que hemos seguido, no es una metodología ágil, sino más bien clásica, puesto que se ha realizado una planificación del proyecto completo desde el principio. Es cierto que se trataba de un proyecto de pequeño tamaño, y que la finalización de este no es ni mucho menos la aplicación funcionando en el entorno de la empresa, por lo que se hace manejable desde el punto de vista de una planificación clásica, ya que no hay que estar iterando sobre los requisitos, estos son los que son para este proyecto embrionario.

6.2. Trabajo futuro

Como bien decimos, este proyecto ha servido de ejemplo para ver cómo funcionan los entornos destinados a la creación de aplicaciones hibridas para distintos sistemas operativos, pero se trata un proyecto con un enfoque real proporcionado por la empresa, que lógicamente espera resultados.

Un primer avance de trabajo futuro es el propio Complemento del TFG, que añade alguna funcionalidad más, esto será visto y explicado en la memoria correspondiente a dicho complemento.

La aplicación está pendiente de ser probada en entornos alfa y beta, con la consecuente posibilidad de cambios correctivos, además, este proyecto se enmarca en uno mayor que pretende llevar algunas funcionalidades más a dispositivos móviles tipo tabletas.

En estos momentos estamos en situación de interfaces, interpretando lo que existe en el SQL Server, para hacer la correspondiente correlación a lo que corresponda en los volúmenes de datos de la empresa.

7. Bibliografía

Analysis, I. I. (2009). *A Guide to the Business Analysis Body of Knowledge® (BABOK® Guide)*. Obtenido de https://cs.anu.edu.au/courses/comp3120/public_docs/BOKV1_6.pdf

Balsamiq. (2018). *balsamiq.com*. Obtenido de balsamiq.com: https://balsamiq.com

Brice, A. G. (2015). Gestión de proyectos informáticos - Desarrollo, análisis y control. Editions ENI. Recuperado de http://www.eni-training.com/client_net/mediabook.aspx?idR=135229

Center, I. K. (s.f.). *Patrón de diseño de modelo-vista-controlador*. Recuperado el 24 de Julio de 2018, de https://www.ibm.com/support/knowledgecenter/es/SSZLC2_8.0.0/com.ibm.commerce.developer.doc/concepts/csdmvcdespat.htm

Chacon, S., y Straub, B. (2018). About Version Control. En *Pro Git.* Apress. Recuperado de https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones

Dauzon, S. (2018). Git y la gestión de versiones. En *Controle la gestión de sus versiones (conceptos, utilización y casos prácticos).* Editions ENI. Recuperado de http://www.eni-training.com/client_net/mediabook.aspx?idR=246213

Debrauwer, L. (2012). Introduccion a los patrones de diseño. En *Patrones de diseño para C# - Los 23 modelos de diseño.* Editions ENI. Recuperado de http://www.enitraining.com/client_net/mediabook.aspx?idR=65573

Foro XOne. (2018). *Foro XOne*. Obtenido de http://xoneisp.com/foro/

Gabillaud, J. (2017). Presentación. En *SQL Server 2016 - Administración.* Editions ENI. Recuperado de http://www.eni-training.com/client_net/mediabook.aspx?idR=211406

Ganttproject. (2018). *www.ganttproject.biz*. Obtenido de www.ganttproject.biz: https://www.ganttproject.biz

Hebuterne, S. (2018). Diseño avanzado. Dibujar en XML. En *Desarrolle una aplicación Android - Programación en Java con Android Studio.* Editions ENI. Recuperado de http://www.eni-training.com/client_net/mediabook.aspx?idR=249102

IEEE Computer Society. (2014). *Guide to the Software Engineering Body of Knowledge. SWEBOK V3.0*. Obtenido de https://www.computer.org/web/swebok/v3

JavaScript.com. (2018). *Ready to Try JavasCript?* Obtenido de www.javascript.com: https://www.javascript.com

Laurent, D., y Fien, V. D. (2016). Modelado de los requisitos, Modelado de la dinámica y Modelado de los objetos. En *UML 2.5 - Iniciación, ejemplos y ejercicios corregidos.* Editions ENI. Recuperado de http://www.eni-training.com/client_net/mediabook.aspx?idR=184766

Microsoft. (2018). *SQL Server*. Obtenido de https://www.microsoft.com/es-es/sql-server

OMG®. (2018). *Welcome To UML Web Site!* Obtenido de www.uml.org: http://www.uml.org

SQLite. (2018). *Small. Fast. Reliable. Choose any three.* Obtenido de sqlite.org: https://sqlite.org/index.html

The Institute of Electrical and Electronics Engineers, Inc. (1998). *IEEE Recommended Practice for Software Requirements Specifications.* New York. Recuperado de http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

Vigouroux, C. (2018). Presentación del lenguaje JavaScript. En *Aprender a desarrollar con JavaScript.* Editions ENI. Recuperado de http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

Visual Paradigm. (2018). *visual-paradigm*. Obtenido de www.visual-paradigm.com: https://www.visual-paradigm.com

w3.org. (2018). *w3.org/XML*. Obtenido de https://www.w3.org/XML/

Wiki XOne. (2018). *Plataforma XOne wiki documentación ByXOne*. Obtenido de http://www.xoneisp.com/xonewiki/doku.php
XOne. (2018). *XOne*. Obtenido de XOne: https://xone.cloud/

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

El desarrollo de aplicaciones móviles hoy en día esta enfocado en dos mundos paralelos, iOS y Android, lo que implica que crear una aplicación, si se desea llegar al máximo número de dispositivos – usuarios posibles, signifique desarrollar nativamente para sendas plataformas. La ingeniería de software permite abstraer las formas y contenidos a través de UML, suavizando las fases de desarrollo de código, pero existe otra tecnología que, en asociación con el modelado, nos permite idear el producto una sola vez utilizando lenguajes y herramientas que nos ofrecen un entorno abstraído, y que son capaces de compilar en las diferentes plataformas.

Estamos hablando del desarrollo hibrido de aplicaciones. A través de lenguajes como puede ser XML, podremos escribir los metadatos necesarios para que los compiladores pertinentes puedan fabricar los ejecutables para la plataforma o sistema operativo que deseemos, ejemplos como Xamarin o Titanium.

The development of mobile applications today is focused on two parallel worlds, iOS and Android, which means that creating an application, if you want to reach the maximum number of possible devices - users, means to develop natively for both platforms. Software engineering allows us to abstract the forms and content through UML, softening the phases of code development, but there is another technology that, in association with modelling, allows us to devise the product only once using languages and tools that offer us an abstracted environment, and that are able to compile on different platforms.

We are talking about hybrid development of applications. Through languages such as XML, we will be able to write the necessary metadata so that the pertinent compilers can manufacture the executables for the platform or operating system that we want, examples such as Xamarin or Titanium.