Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación para la venta de tickets en autocares

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: [Silvia Sahuquillo Falaguera]

Tutor: [María José Vicent López]

[Curso 2015-2016]

Tabla de contenidos

1. Introducción

Este trabajo de fin de Grado se ha realizado en el contexto de trabajo de la empresa Softour Sistemas S.L. a la que se le encargó hacer un sistema de venta de tickets a bordo de autobuses con su posterior gestión de los mismos, control de recaudación, gestión de cajas de conductores, estadística y otros muchos elementos.

Como ya se expuso en el documento de Solicitud de TFG Externo, es imposible llegar a exponer el proyecto completo en el ámbito de exigencia de este Trabajo Final de Grado, por lo que se describirá con la mayor exactitud posible la parte móvil y la sincronización con el servidor, sin entrar en las aplicaciones para el posterior tratamiento de datos y otras aplicaciones para la venta.

Como decíamos este proyecto trata de una aplicación móvil para la venta de tickets abordo de autobuses. Concretamente, está enfocado a explotaciones privadas de transporte de pasajeros de líneas regulares.

Nuestro cliente, nos informó desde el primer momento, de la necesidad de que estos datos se sincronizasen automáticamente con una plataforma online a la que poder acceder desde cualquier sitio, sin necesidad de descargar los datos de los terminales, conectándolos físicamente con un ordenador.

Se buscaron unos terminales con unas características muy específicas, deberían ser móviles, puesto que la venta podría realizarse tanto en el propio autocar como a pie del mismo. Rugerizados, por trabajar en condiciones extremas en las que podrían caer al suelo o golpearse, debían soportar altas temperaturas, puesto que situados en el salpicadero el impacto solar a través del cristal es mayor. Para la sincronización con el servidor sin conexión física, era imprescindible la conexión a la red de datos. Debían tener integrada una impresora térmica para imprimir los tickets o comunicarse con ella a través de bluetooth o cable. Para simplificar el trabajo del conductor, facilitar el aprendizaje y la adaptación se propusieron terminales con pantallas táctiles.

Todos estos requisitos los encontramos en varios modelos de terminales que funcionan con el sistema operativo Windows mobile. Por ello la aplicación se desarrolla en Windows forms con lenguaje de programación c# .net.

En este documento se desarrollan algunas partes del ciclo de vida de este proyecto. En cuanto al Análisis se refiere, contaremos con el documento de declaración de alcance donde se describen los requisitos para cubrir las necesidades propuestas por el cliente. En el diagrama de casos de uso y las fichas de tareas se presentan cada uno de los procesos que se deben realizar. El diagrama conceptual, da una primera impresión de cómo fluirá la información a través del sistema, aunque no es un reflejo fiel de la estructura de formularios final. Y el diagrama de clases, que contiene los objetos del dominio.

Otro de los aspectos del ciclo de vida del proyecto que se tratarán en esta memoria es el Diseño, tanto arquitectónico donde se ha implementado utilizando un modelo 3 capas, tratando de separar la lógica de negocio de la lógica de diseño y la capa de negocio de la de datos. Como a nivel de interfaces, donde se han tenido en cuenta los principios de Gestalt, para favorecer la usabilidad de la aplicación.

Por último de adjunta un manual de usuario y las conclusiones.

2. Declaración del alcance

2.1 Misión del proyecto

El cometido de este proyecto es desarrollar un sistema que permita la venta de tickets para autobuses.

El software que se desarrollará debe descargar información sobre conductores, empresas, explotaciones, líneas, paradas, tarifas y bonos vendidos en otros puntos de venta. Por otro lado, debe registrar los tickets vendidos a bordo del autobús, ofrecer la posibilidad de anularlos y guardar información de la jornada, expedición y parada en que se ha vendido.

2.2 Objetivos

2.2.1 Generales

Uno de los objetivos principales de la aplicación es que el conductor agilice las ventas y minimice el tiempo que para en cada parada.

El conductor debe tener un control de caja y número de pasajeros a bordo.

Con fines estadísticos, el sistema debe tener constancia de la hora y parada en que se vende cada uno de los tickets.

Facilitar las tareas de administración y control de cajas de todos los conductores de una misma explotación.

2.2.2 Específicos

Cada ticket se deberá imprimir en un periodo inferior a 5 segundos. Para que en cada parada no permanezca parado más de 6 minutos.

En administración deben recibir los datos de las ventas en un plazo de 24 horas.

2.3 Definición general (Requisitos)

2.3.1 Funcionales

Conductor:

Podrá configurar la ruta.

Podrá vender y anular tickets.

Debería registrar parada actual.

Podrá comprobar validez del bono y canjearlos.

Debería poder cerrar expedición y jornada.

2.3.2 No funcionales

El acceso al sistema se realizará mediante una clave única para cada conductor.

Los bonos de los colaboradores tendrán un código único, el importe total del bono, la fecha de compra, una marca de si está anulado o no, la fecha de uso, además contendrá las tarifas de cada uno de los tickets que contiene así como las unidades de tickets de cada tarifa.

Por último por cada ticket se almacenará la fecha de emisión, la parada, la tarifa y la expedición, un código, el terminal que lo ha expedido, marcas de anulación y de sincronización y fecha de anulación.

Los terminales elegidos funcionan con Windows Mobile 6, por lo que el programa debe estar optimizado para este sistema operativo.

El sistema debe de poder vender tickets y tener toda la funcionalidad aunque no tenga conexión a de datos. 3G/4G.

El sistema debe de imprimir los tickets en un periodo de tiempo lo más corto posible, como máximo de 5 segundos por ticket, para facilitar que la recogida de los pasajeros se haga en el menor tiempo posible.

La interfaz debe de estar adaptada al dispositivo táctil. La mayor parte del trabajo debe hacerse mediante selectores y evitar la utilización del teclado.

Cada **empresa** deberá tener un cif y un nombre comercial.

Cada **conductor** deberá tener un nombre, nif, código de acceso a la aplicación y un identificador de la empresa a la que pertenece.

Cada **vehículo** deberá tener matrícula, número y empresa a la que pertenece.

El sistema deberá diferenciar entre las diferentes **líneas** que puede tener una empresa se almacenará nombre de línea, el periodo de anulación de los tickets que se vendan en esa línea y el periodo de validez.

Cada línea tendrá asociadas un conjunto de **paradas**. Para cada parada deberá tener un nombre, número de orden dentro de la línea.

A su vez, cada línea tiene unas **tarifas** asociadas, cada tarifa tendrá una descripción, precio, número de copias que se imprimirán cuando se venda un ticket de esa tarifa y un periodo de validez en lo que se refiere a que esa tarifa estará a la venda.

Deberemos almacenar los **horarios** de paso por la parada inicial de cada línea, es decir el inicio de expedición, para ello deberemos almacenar el identificador de la línea, la hora de paso y el periodo de validez.

Cada **terminal** tendrá un código único.

En lo que se refiere a la **jornada**, se almacenará la fecha y hora en que se inició, fecha y hora de cierre, el terminal, el conductor y recaudación total.

Sobre las **expediciones,** se guardará la hora de salida, el conductor, la parada inicial, la línea a la que pertenecen, hora de salida programada, vehículo, recaudación, la jornada y hora de cierre.

3. Especificación conceptual del sistema (UML)

3.1 Descripción de actores y stakeholders.

Actores primarios:

Conductor, que es la persona que va a utilizar la aplicación.

Actores secundarios:

Los clientes que utilizan el servicio de transporte.

Otros stakeholders:

Colaboradores, Webs que venden bonos

Empresa que contrata la aplicación.

Personal de administración que hará la gestión posterior a las ventas.

3.2 Modelo de dominio

3.2.1 Definición de objetos

Conductor: Persona que utiliza la aplicación.

Cliente: Persona que utiliza el servicio de transporte.

Ticket: Justificante que el conductor entrega al cliente y le da acceso al servicio.

Vehículo: Medio de transporte que realiza el servicio

Línea: Ruta de la empresa

Parada: Cada una de las paradas que componen una línea

Tarifa: Tipos de entrada que oferta una línea

Colaboradores: webs de venta online de bonos

Bono: justificante de compra en una web de colaborador

LineaBono: cada una de las líneas de un bono. Contiene información de la tarifa y las unidades para imprimir los tickets correspondientes.

3.2.2 Diagrama de objetos del dominio

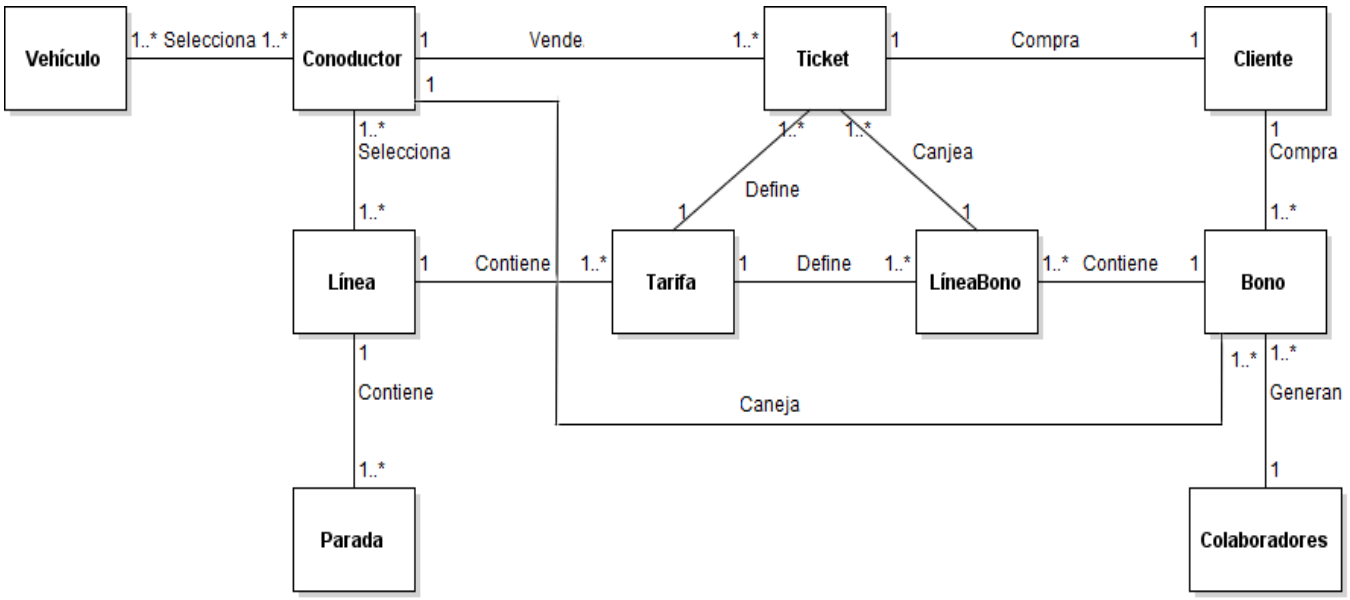

Imagen 3.1 Diagrama de Objetos del dominio

3.3 Casos de uso

3.3.1 Diagrama de casos de uso

Estos son los casos de uso que se representan en el diagrama de la imagen 3.2.

Identificarse en la aplicación

El conductor se identifica en el sistema introduciendo su usuario y contraseña.

Abrir nueva expedición

El conductor abre una nueva expedición antes de comenzar a vender tickets. De esta forma, los tickets se organizan en las diferentes expediciones en que se venden y el conductor puede cuadrar la caja al finalizar cada expedición.

Vender Tickets

El conductor vende los tickets seleccionando la tarifa a la que pertenece cada pasajero.

Validar Bono

Cuando el cliente entrega al conductor un bono comprado en una web de colaborador, el conductor lo lee con el terminal para validarlo y se imprimen los tickets correspondientes a las líneas del bono.

Anular tickets

El cliente puede comprar un ticket y anularlo posteriormente durante un determinado periodo de tiempo. O puede que haya un problema en el correcto funcionamiento del servicio y la empresa apruebe la anulación y devolución del importe de los tickets. El cliente tiene que entregar el ticket, el conductor lo lee y el terminal imprime una copia con una marca y hora de anulación.

Comunicar la parada actual

Cada vez que arranca el vehículo el conductor debe de marcar el paso de parada actual, y dejar el sistema apuntando a la próxima parada de la línea.

Cerrar expedición

Cuando el conductor llega a la parada inicial debe hacer un cierre de expedición para tener un control de la caja. Se emitirá un justificante de ventas y se podrá abrir la siguiente expedición.

Reenviar datos

Este es un mecanismo de recuperación de errores. El conductor puede mandar datos de jornadas anteriores que no se sincronizaron con el servidor por problemas de conexión a la red de datos.

Cierre de jornada

Al terminar la jornada de trabajo, el conductor cierra la jornada y se imprime un justificante para cuadrar la caja del día.

Comprar bonos

El cliente puede comprar bonos en web de colaboradores que posteriormente canjeará, al subir al vehículo, por tickets.

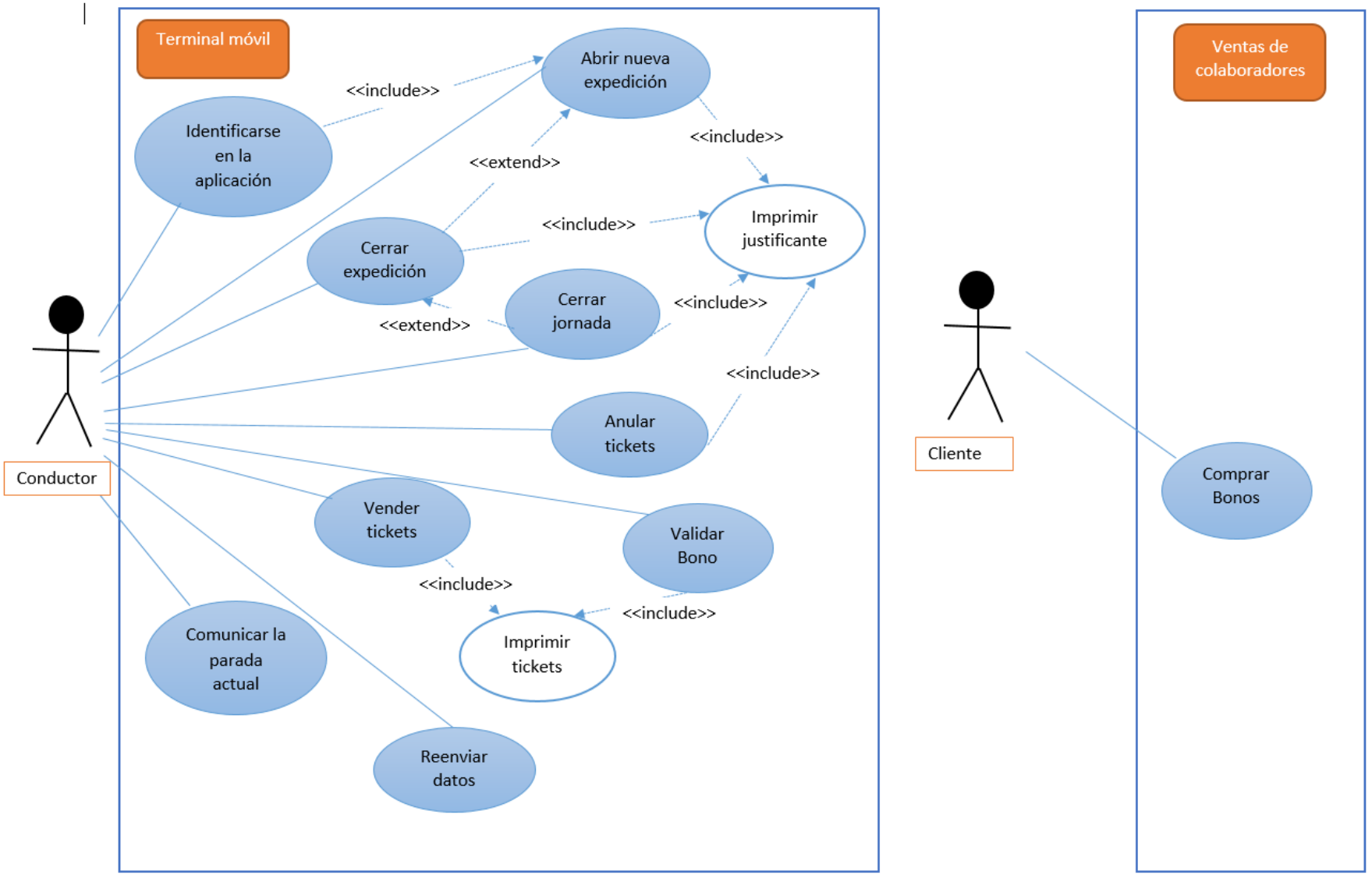

Imagen 3.2 Diagrama de Casos de uso

3.3.2 Ficha de tareas

Acción del usuario	Respuesta del sistema
Identificarse en la aplicación	El conductor comienza su jornada laboral, para ello tiene que abrir una nueva expedición. Antes que nada se identifica en el sistema introduciendo su usuario y contraseña. El sistema valida que las credenciales con las que ingresa son correctas. A continuación le muestra una pantalla para la apertura de expedición. El sistema le ofrecerá las líneas, paradas y horarios asociados a su empresa.
Abrir nueva expedición	El conductor selecciona la línea parada y horario en que va a trabajar El sistema le pide una validación de los datos seleccionados. Si el conductor acepta los datos de apertura de expedición, el sistema registra la apertura de expedición y le imprime un justificante. Si no acepta los datos de apertura, el sistema vuelve a la pantalla anterior y le ofrece de nuevo las líneas, paradas y horarios para que rectifique.
Vender Tickets	El conductor tiene que vender unos tickets, para ello selecciona la opción del menú: venta de tickets, introduce el número de unidades de cada ticket y pulsa imprimir. El sistema le muestra un resumen de la venta, le facilita el importe total a cobrar. A continuación espera a que el conductor acepte la venta. El conductor acepta la venta y pulsa imprimir. Mientras, cobra a los clientes. El sistema imprime los tickets, actualiza el número de pasajeros de la expedición y muestra la pantalla de venta de tickets.
Validar Bono	Un cliente da al conductor un bono de colaboradores. El conductor lee el código de barras del bono. El sistema busca el bono vendido por un colaborador. Cuando lo encuentra, muestra un resumen de los tickets que contiene el bono para que el conductor pueda validar que el bono no ha sido manipulado. Si el conductor acepta se imprimen los tickets correspondientes.

Anular tickets	Un cliente pide la devolución del importe del ticket por una avería en el servicio. El conductor le pide el ticket físico, selecciona en el menú la opción de Anular y lee el código de barras del ticket. El sistema alerta al conductor en caso de que haya excedido el tiempo de anulación. Imprime una copia del ticket con la marca de anulado y actualiza el estado del ticket.
Comunicar la parada actual	El conductor arranca de nuevo el autocar, pero antes pulsa el botón de Avance de parada. El sistema refleja que se ha pasado de parada y sincroniza los tickets con el servidor.
Cerrar expedición	El conductor selecciona Cerrar expedición. El sistema imprime un justificante con el resumen de tickets vendidos. A continuación ofrece los horarios pendientes en la jornada y ofrece el más cercano por defecto. El sistema manda la información de la expedición al servidor.
Reenviar datos	El conductor selecciona la opción de reenvío de datos, selecciona el periodo de tiempo que desea enviar y lo envía. El sistema busca la información entre las copias que almacena y las envía al servidor.
Cierre de jornada	El sistema imprime un justificante con el resumen de todas las expediciones de la jornada con sus tickets, la recaudación y el número de pax. Mientras manda la información al servidor y cierra la aplicación automáticamente.

3.4 Diseño conceptual

3.4.1 Descripción de contenedores

En este apartado describiré los contenidos que aparecen en el diagrama de contenido de la imagen 3.3.

En el diagrama de contenidos, aunque es muy similar a la estructura de la interfaz, no la sigue fielmente y como veremos algunos de los contenedores aparecen fusionados en un mismo formulario de la interfaz, y por el contrario, otros de un contenedor sacaremos dos o tres formularios.

Identificación Identifica al conductor en el sistema, a partir de un código. **Funciones** • Identificar al usuario • Mostrar las opciones de acceso **Enlaces** ▸ Abrir expedición ▸ Reenviar datos **Objetos** Conductor, Jornada **Restricciones**	Abrir expedición Permite crear una expedición a partir de unos valores relacionados con el conductor identificado. **Funciones** • Crear una nueva expedición • Mostrar opciones • Imprimir justificante ▪ Mostrar resumen **Enlaces** ▸ Principal **Objetos** Expedición, Conductor, Linea, Parada, Horario **Restricciones** -La expedición se debe crear en base a las líneas relacionadas con el usuario identificado. -Debe haber papel en la impresora de la máquina.	Principal (Menú) Contiene el menú con todas las posibles opciones **Funciones** • Cerrar la jornada ▪ Mostrar el número de tickets vendidos en la expedición ▪ Mostrar información de la expedición, Línea y Hora **Enlaces** ▸ Venta de tickets ▸ Anular tickets ▸ Cierre de expedición **Objetos** Parada, Expedición, Linea,Horario **Restricciones**
Reenviar datos Envía datos de jornadas anteriores. **Funciones** • Envía los datos de las jornadas de un periodo especificado **Enlaces** ▸ Principal ▸ Login **Objetos** Jornada, Expedicion, Tickets **Restricciones** -Conexión 3G	Cerrar expedición Cierra la expedición actual. **Funciones** ▪ Imprimir justificante • Crear una nueva expedición • Sincronizar información **Enlaces** ▸ Abrir expedición ▸ Principal **Objetos** Expedición **Restricciones** -Debe haber papel en la impresora de la máquina.	Cerrar jornada Cierra la jornada actual. **Funciones** ▪ Imprimir justificante • Cerrar la aplicación • Sincronizar información **Enlaces** **Objetos** Jornada **Restricciones** -Debe haber papel en la impresora de la máquina.

<table>
<tr>
<td>Venta
Muestra las tarifas con sus importes disponibles para la venta.
Funciones
▪ Mostrar tarifas
• Seleccionar los tickets para vender.
Enlaces
▸ Principal
▸ Bonos
Objetos
Tarifas
Restricciones</td>
<td>Tickets
Permite generar los tickets
Funciones
• Imprimir ticket
▪ Mostrar resumen e importe de los tickets que se van a vender
Enlaces
▸ Venta
Objetos
Ticket, Tarifa, Parada
Restricciones
-Debe haber papel en la impresora de la máquina.</td>
<td>Bonos
Permite generar los tickets a partir de un bono impreso.
Funciones
• Imprimir tickets
▪ Mostrar resumen de los tickets que contiene el bono
Enlaces
▸ Venta
Objetos
Bono, Ticket, tarifa
Restricciones
-Debe haber papel en la impresora de la máquina.</td>
</tr>
<tr>
<td>Anular tickets
Permite anular un ticket leyendo su código de barras
Funciones
▪ Mostrar el resumen del ticket leído
• Imprimir justificante de anulación
Enlaces
▸ Principal
Objetos
Ticket
Restricciones
-Debe haber papel en la impresora de la máquina.</td>
<td>Paso de parada
Permite situar al sistema en la parada actual
Funciones
▪ Mostrar la parada actual
• Seleccionar la parada
• Sincronizar datos
Enlaces
▸ Principal
Objetos
Parada
Restricciones</td>
<td></td>
</tr>
</table>

3.4.2 Diagrama de contenidos

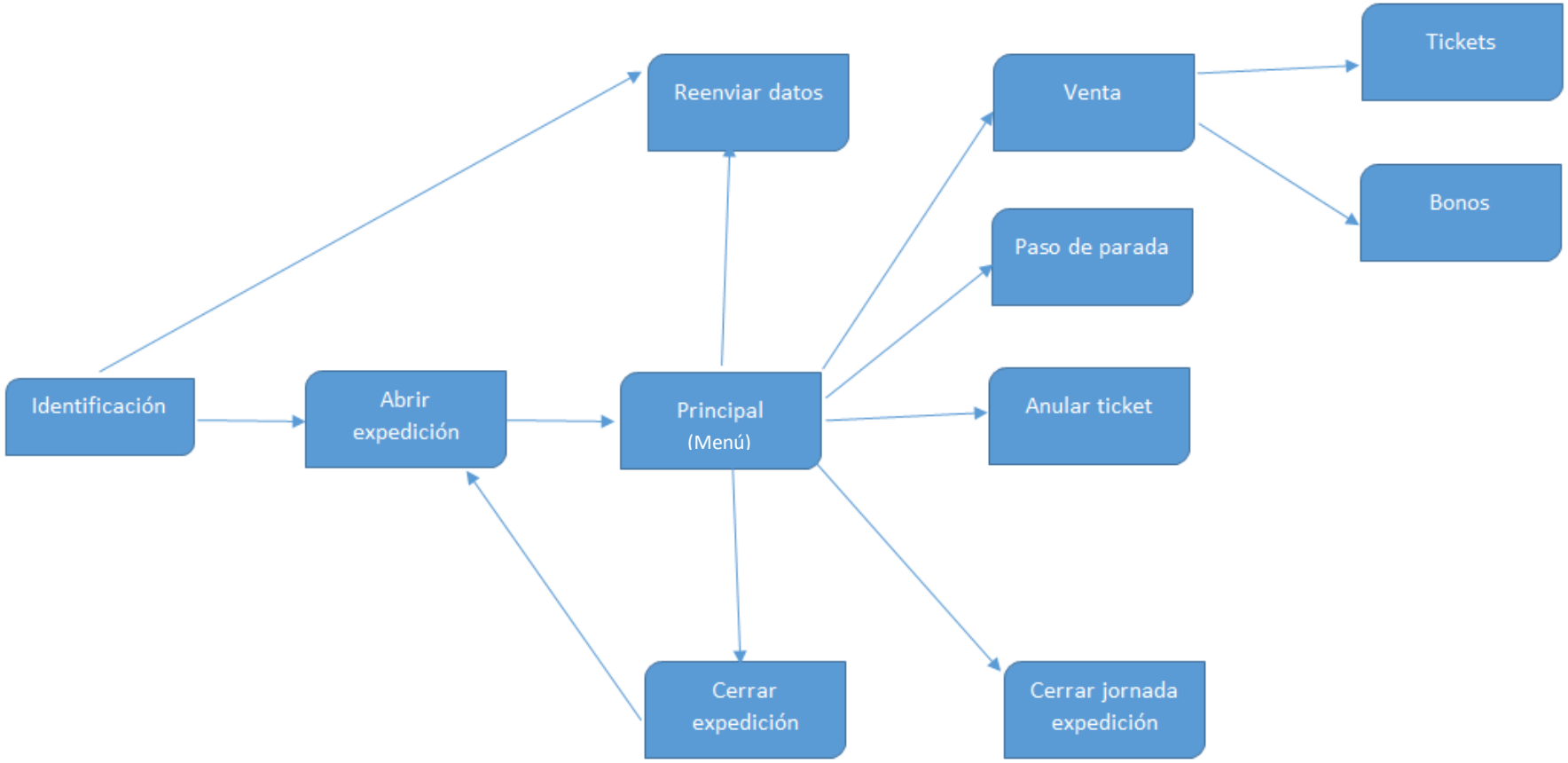

Imagen 3.3 Diagrama de contenidos

4. Diagrama de clases

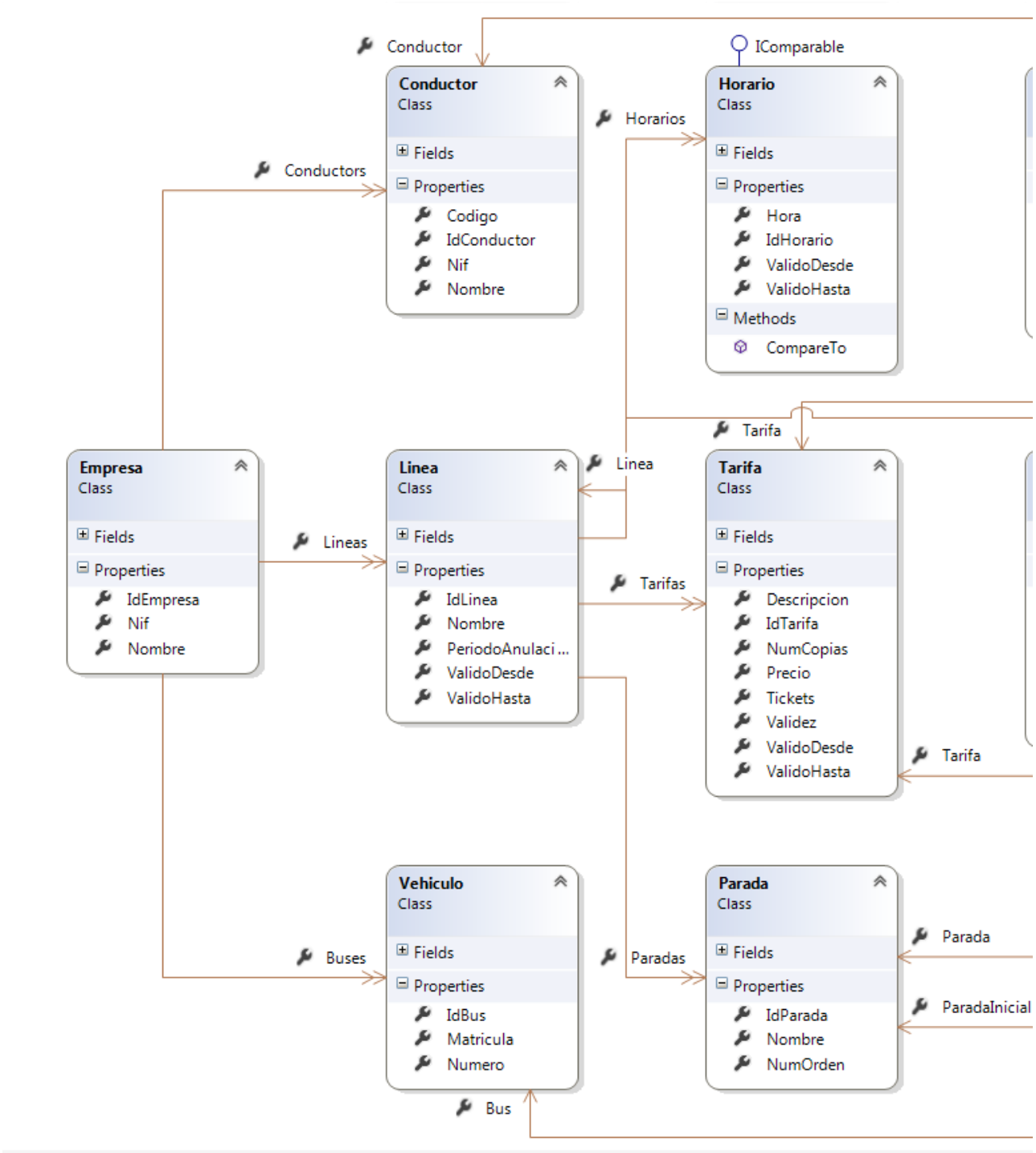

Imagen 4.1 Diagrama de clases 1/2

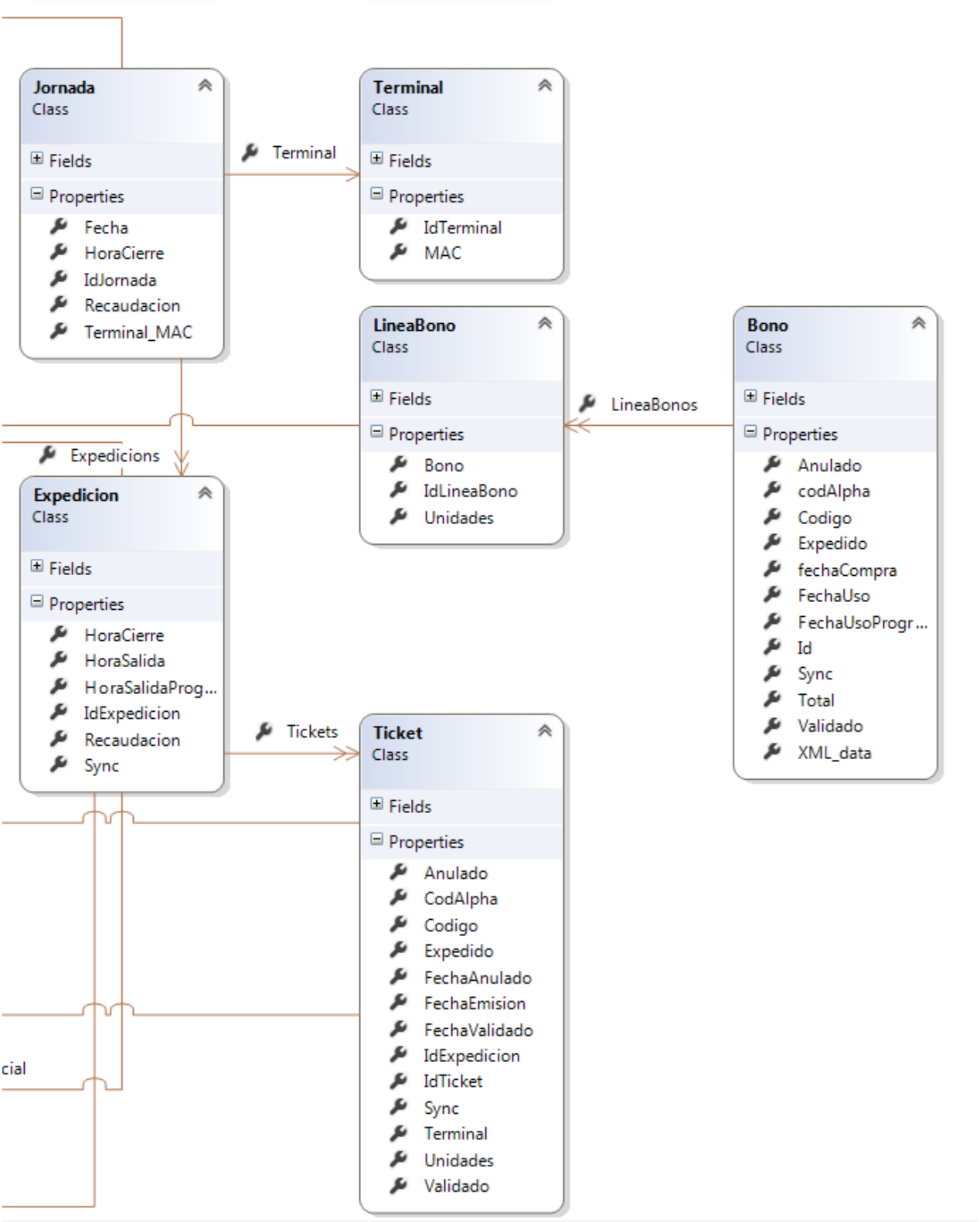

Imagen 4.2 Diagrama de clases 2/2

4.1 Clases básicas de la lógica del negocio

En este apartado voy a relacionar los objetos con las partes del programa en que se han utilizado, justificando así sus propiedades y relaciones que se muestran en el diagrama anterior. (Imagen 4.1 y 4.2)

Clase empresa

Contiene la información básica de una empresa, el sistema está diseñado para poder trabajar con varias empresas y operará con aquella a la que pertenezca el conductor que se identifica.

El objeto consta de IdEmpresa, un identificador único, su nif y su nombre. Además, tiene una lista con los conductores que trabajan en ella, una lista de líneas y una de los vehículos de los que dispone que ofrecerá en la apertura de la expedición.

Clase Conductor

Contiene la información básica del conductor.

El objeto Conductor consta de un código que utiliza para identificarse en la aplicación. Un identificador único en todo el sistema, su nif y su nombre.

Clase Línea

Es la clase que representa las líneas de una explotación. Antes de ofrecerla al conductor para abrir su expedición, el programa deberá validar que la línea se encuentra en vigor, comprobando los parámetros de validez (ValidoDesde y ValidoHasta). Cuando se abre la expedición, después de seleccionar la línea, el sistema ofrece los horarios y las paradas que tiene asociados.

En otra de las partes en que comprobamos la línea de la expedición abierta, es en la parte que da sentido al programa, en la venta de tickets, en esta caso, se cargarán las tarifas de la línea seleccionada.

A la hora de anular un ticket, el programa también deberá tener en cuenta el periodo de anulación de la línea (PeriodoAnulacion) ya que puede variar y si se encuentra fuera del periodo debe mostrar una alerta al conductor.

Además, contiene un identificador único, un nombre.

Clase Vehículo

Este es un objeto sencillo, se selecciona al abrir la expedición y con tiene un identificador único, la matrícula y un número que es el valor por el que los conductores identifican al vehículo.

Clase Horario

El horario también se selecciona en la apertura de la expedición, como vemos implementa la Interfaz Comparable. Esto es así, porque nos interesa comparar estos objetos por un atributo en concreto: Hora. Se emplea esta comparación para ordenar las horas que se ofrecen al conductor en la apertura de expedición y para ofrecerle la siguiente a la que eligió en la apertura anterior.

Además contiene un identificador único y un periodo de validez igual que la línea.

Clase Tarifa

La tarifa contiene una descripción del tipo de ticket que se va a vender, además del precio de dicho ticket, el número de copias de cada ticket que se han de imprimir, aunque inicialmente sería una, puede haber casos en que se tenga que emitir por duplicado.

Da información del tiempo de validez de los tickets y de la misma forma que las líneas y los horarios tiene un periodo de validez en que una tarifa estará operativa.

Clase Parada

La parada es un objeto simple, que tiene un nombre, un identificador único y un número de orden, para que cuando el conductor efectúe el paso de parada se sitúe en la siguiente.

Clase Jornada

La jornada se crea automáticamente en el sistema, en el momento en que el conductor se identifica en la aplicación, refleja la fecha y hora de apertura y almacena la mac del terminal. Al final del día, cuando se cierra, añade la información de la hora de cierre y la recaudación total.

Clase Expedición

Se crea cuando el conductor se identifica en el sistema la primera vez y selecciona la línea en la que va a trabajar, parada inicial, horario y el conductor valida la información que ha

seleccionado previamente, este objeto, también se crea cada vez que se cierra una expedición y se abre una nueva.

Almacena los identificadores de todos estos objetos antes mencionados: línea, parada y horario y además añade la hora de salida dejando la hora de cierre y la recaudación pendientes hasta que llegue el momento.

Tiene la propiedad: Sync, que es el que indica si se ha sincronizado con el servidor.

Clase Terminal

Este objeto es muy sencillo, simplemente almacena un identificador del terminal y su mac, de esta forma.

Clase Ticket

Se crea cuando el conductor lo imprime después de seleccionar las tarifas y unidades adecuadas a la venta, por lo que contiene el identificador de la tarifa así como el identificador de la expedición. Contiene la fecha y hora en que se ha vendido y el código que aparece en el código de barras. Además con fines exclusivamente estadísticos, también se almacena la parada en que se ha vendido.

Cuando un conductor anula un ticket, el sistema lo busca y lo marca como anulado, incorpora la fecha de anulación y el terminal que lo ha anulado.

El objeto ticket, tiene la propiedad: Expedido, en este se almacena en un formato codificado quien ha vendido ese ticket y su origen, si es del propio terminal o de un punto de venta online y ha entrado en el sistema a partir de un bono.

Por último vemos que tiene una propiedad: Sync que del mismo modo que en las expediciones marca si se ha sincronizado con el servidor o no.

Clase Bono

Los bonos se generan en otras aplicaciones al margen de esta que es la que nos ocupa en este TFG. Aun así, hemos de tenerlos en cuenta a la hora de comentar una parte de la aplicación móvil que nos ocupa.

El objeto bono es como el resumen de una venta, una cabecera de la cual cuelgan las líneas que contienen información de los tickets.

Los bonos llegan al terminal descargados del servidor y quedan almacenados en todos los terminales que están operando en el sistema para simplificar la búsqueda y agilizar el proceso de canje, en caso de que el cliente se presente con el bono para canjearlo. Los bonos “físicos”

tienen un código de barras que corresponde a la propiedad: codigo en el modelo. Cuando el conductor lee el código de barras del bono físico, y lo valida, en la pantalla le aparece la información del bono descargado(fechaCompra e Importe), una vez canjeado añade la fecha de uso.

Podemos ver que el objeto Bono tiene una propiedad FechaUsoProg esta propiedad sirve para que el servidor sea selectivo y descargue a los terminales sólo aquellos bonos que estén en vigor siendo la FechaUsoProg igual o posterior a la de hoy, que no estén anulados comprobando la propiedad Anulado y que la FechaUso no tenga valor.

Igual que los tickets tienen una propiedad Expedido que da información sobre el origen del bono y Sync que indica si se ha sincronizado con el servidor.

Clase LineaBono

Como hemos visto el objeto Bono, no tiene información de los tickets que se han de imprimir, esta información la encontramos en las líneas las líneas tienen información de la tarifa y las unidades que deberá imprimir.

Por tanto, cuando el conductor lea el bono “físico”, el sistema recuperará el objeto Bono y sus líneas asociadas y cuando el conductor lo valide imprimirá tantos tickets de las tarifas adecuadas como unidades haya encontrado en sus líneas.

5. Diseño del sistema

5.1 Estilo arquitectónico

5.1.1 Introducción

Como decía al principio del documento, este proyecto además de los terminales móviles se compone de un servidor alojado en internet, necesario para la sincronización de la información.

La comunicación entre los terminales y el servidor se compone de una capa intermedia de acceso a datos, middleware. Esta capa es un Web Api programado en c# .net. Diseñarlo de esta forma, tiene una ventaja, y es que como se trata de un Web Api genérico y trabaja con objetos serializados, esta parte se podría mantener sea cual sea el sistema operativo del dispositivo que lo consuma. (Img 5.1)

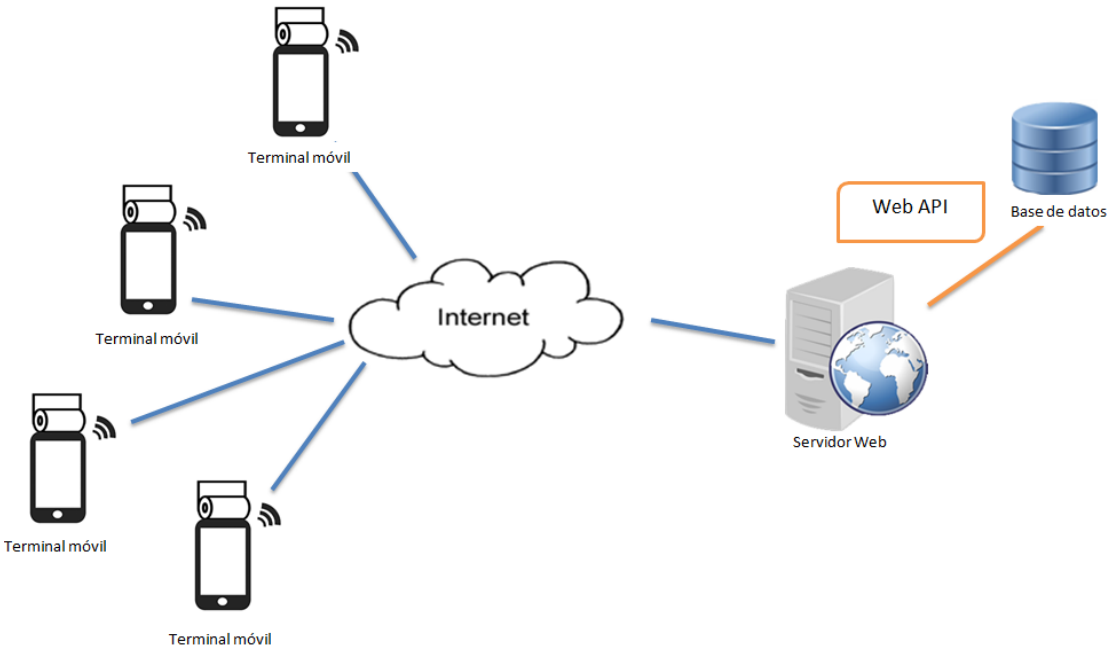

Imagen 5.1: Estructura de red

Cuando el conductor inicia una nueva jornada, se hace una conexión con el servidor, para comprobar que las líneas, paradas, horarios y tarifas están actualizadas en el terminal.

Otros puntos en que la aplicación se comunica con el servidor son: en la apertura de expedición donde indica la apertura de expedición, en cada paso de parada, donde sincronizan los tickets pendientes de sincronizar, en el cierre de expedición donde se envían un resumen de la expedición y en el de jornada donde se envía un resumen de la jornada.

Cuando se cierra la jornada la aplicación almacena un fichero xml con el objeto jornada serializado, en el que se incluyen las expediciones y tarifas. Estos ficheros se utilizan porque cuando se envían los datos de forma manual porque por la razón que sea no han llegado a sincronizarse con el servidor, la aplicación busca los correspondientes a los días seleccionados y los manda de nuevo al Web Api, asegurándose esta vez de que el terminal tiene conexión a la red de datos.

Todas estas comunicaciones se realizan en hilos en background para que en el hilo principal se pueda seguir trabajando normalmente.

Se ha diseñado utilizando una arquitectura 3 capas, intentando separar la lógica de negocio con la lógica de diseño y la capa de negocio de la de datos.

A continuación se expone cada una de las capas.

5.1.2 Capa de Presentación

En esta sección explicaremos el diseño de los formularios.

El diseño está optimizado para pantallas de 3,5” con una resolución de 240 x 320px. Utilizando Windows Forms para Windows Mobile no es posible hacer interfaces adaptativas y por fortuna, los terminales elegidos, siempre han tenido este tamaño de pantalla.

Para realizar el diseño, se han tenido en cuenta algunas de las **leyes de Gestalt**, teniendo en cuenta que es fundamental la organización de los componentes en la pantalla. Una de estas leyes o principios, es el **principio de consistencia**. Toda la interfaz tiene el mismo esquema de color, orden de los botones, nombres de los conceptos y la misma estética. Hay contraste entre el fondo y los componentes utilizados para que de un golpe de vista se pueda identificar el contenido.

A conciencia de que el conductor, cuando vende los tickets, mantiene una comunicación con el cliente que puede distraerlo, olvidándose de lo que estaba haciendo, las interfaces están orientadas a recordarles el siguiente paso. Aplicando el **principio de la organización** conceptual, las cosas que tienen relación aparecen agrupadas, como en el caso de las tarifas, importes y unidades. *(Img 5.2)*

Imagen 5.2: Pantallas de tarifas

Teniendo en cuenta el **Principio de aprovechamiento del conocimiento previo**, a la hora de seleccionar el número de tickets *(Img 5.2)*, usa las flechitas de arriba y abajo para incrementar o decrementar el número de tickets, esto es un concepto que seguro que le resulta familiar. Además este componente obliga al conductor a introducir únicamente valores numéricos, lo que simplifica la tarea y previene de errores.

Para centrar la atención del conductor, las cosas más importantes aparecen en la parte superior de la pantalla, como la hora de la expedición, el botón de imprimir, o el número de pasajeros de la expedición actual o en un tamaño de letra más grande, como el importe a cobrar. *(Img. 5.3)* Por otra parte, los avisos aparecen en el centro de la pantalla, como cuando se acaba el rollo de papel o hay un atasco en la impresora. (Img. 5.4)

Imagen 5.3: Impresión de tickets

Imagen 5.4: Aviso impresora

Para favorecer la rapidez en el aprendizaje se ha seleccionado un menú con estructura de árbol con 3 niveles como máximo. Para ello, se ha diseñado una pantalla con todas las posibles opciones del menú desde la que se abren todas las otras pantallas. El menú está compuesto por una serie de botones grandes que contienen un texto autoexplicativo con una breve descripción del proceso que se va a realizar. El texto usa términos familiares y consistentes, empleando palabras clave que definen la tarea y se sitúa centrado en el botón.

Además está ordenado por orden de importancia y frecuencia de uso, de tal forma que las tareas que sólo se realizarán una vez al día, como el cierre de Jornada, o puede que no se realicen, como el proceso de envío de datos a petición, aparecen las últimas y hace falta desplazarse con la scrollbar vertical para acceder a ellas (Img. 5.6). Por el contrario aquellas tareas a las que se recurre constantemente como la venta de tickets o el paso de paradas o la anulación de tickets aparecen las primeras. (Img. 5.5)

Para navegar entre las distintas opciones y luego volver al menú, a lo largo de todas las pantallas, contamos con un botón de retroceso, Atrás. Este botón se encuentra siempre fijo en la parte inferior derecha de la pantalla y aparece deshabilitado cuando no procede su uso, como por ejemplo en la pantalla del menú, ya que al tener una estructura en forma de árbol, éste es el origen.

Imagen 5.5: Menú 1/2 Primeras posiciones

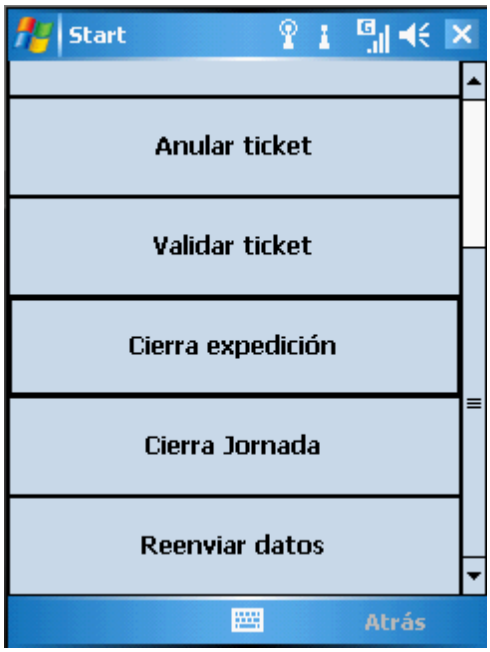

Imagen 5.6: Menú 2/2 Últimas posiciones

5.1.3 Capa de Negocio

Aplicación móvil

En la aplicación, esta parte se encuentra separada en dos niveles. Por un lado tenemos la lógica de la aplicación donde se encuentra la funcionalidad de cada formulario en el proyecto de Windows Forms.

Por otro, en un proyecto de librería, donde se encuentran definidos los objetos del dominio que constituyen la Lógica de Negocio y los métodos que alojan la lógica general de la aplicación. Es decir, validaciones, búsquedas, modificación de objetos del dominio, labores de sincronización con el servidor, registro de tickets…

Los formularios utilizan estos métodos, y como están situados en el proyecto de librería se reutilizan.

Web API

Los objetos con los que trabaja la aplicación móvil que se han descrito en el apartado 4 Diagrama de clases y los que trabaja el servidor no son exactamente iguales. Para simplificar y aligerar la información que se transmite por la red, la aplicación móvil trabaja sólo con la información básica y es el Web Api el que trata estos objetos y los convierte según el sistema al que los comunica.

Para la programación de la capa de negocio se han tenido en cuenta algunas **buenas prácticas de codificación**. Entre ellas, la utilización tipos específicos en lugar de los tipos definidos en el namespace System.

Para iniciar variables de tipo String se usa String.Empty en lugar de “” y para la concatenación de Strings en lugar de utilizar ‘+’ se usa la clase StringBuilder.
Todas las conexiones a la base de datos o al web Api, siempre se cierran en bloques finally para asegurar que aunque haya errores no se queden conexiones en stand by.
La declaración de variables locales se realiza al principio del método y se inicializan para limpiar su valor predeterminado.
Se utiliza una nomenclatura Pascal Casing(primer carácter de todas las palabras se expresa en mayúscula y el resto de los caracteres en minúscula) para los nombres de las clases y para el nombre de los métodos.
El nombre del archivo de la clase coincide con el nombre de la clase.
Se utiliza una nomenclatura Camel casing(primer carácter de todas las palabras, excepto la primera palabra se expresa en mayúscula y el resto de los caracteres en minúscula) para declaración de variables y parámetros de los métodos.
Se utiliza el prefijo “I” con Pascal Casing para Interfaces.
Se utiliza de una línea en blanco para separar grupos lógicos de código y se utiliza #region para agrupar piezas de código relacionadas.

5.1.4 Capa de Persistencia

En esta capa está programado el Acceso a datos y la persistencia de la información.

Aplicación móvil

Como el sistema operativo que van a llevar las máquinas en Windows Mobile, en este proyecto se ha elegido el SQL Server Compack Edition de Microsoft como Sistema de Gestión de Bases de Datos, y la tecnología ADO.Net para acceder a los datos.

Los métodos de lectura y escritura en la base de datos y la conversión de la información de la base de datos a los objetos del dominio se encuentran alojados en otro proyecto.

Otro aspecto de la persistencia en la aplicación móvil, es que por cada cierre de jornada, se almacena un fichero en la tarjea SD del terminal. Este fichero de texto, contiene el objeto Jornada, con sus expediciones y tickets serializado en formato xml por si fuese necesario recuperarlos o cargarlos en el sistema de forma manual.

6. Manual de usuario

6.1 Login

Al acceder a la aplicación se encuentra con la siguiente pantalla.

Por un lado se puede identificar en la aplicación, para ello hay que introducir su código de usuario y seleccionar: “Abrir Jornada” en caso de que sea la primera vez que accede, o “Continuar Jornada” en caso de que tuviese una abierta previamente. A continuación hay que pulsar Entrar para que el sistema compruebe las credenciales y de acceso a la aplicación o no.

Por otro lado podemos ver al pie de la pantalla un botón para Enviar datos. Este botón, como indica su nombre, sirve para enviar los datos que en circunstancias sin conexión hayan quedado pendientes de sincronizarse con el servidor.

Imagen 6.1: Login

Imagen 6.2: Login - Desplegable

6.2 Apertura de expedición

Al abrir la expedición aparece seleccionada la empresa a la que pertenece el usuario que se ha identificado en la aplicación.

A continuación, seleccionaremos el vehículo y línea con los que se vaya a trabajar. Estos desplegables aparece precargado con los vehículos y líneas que pertenecen a la empresa del usuario identificado. (Img 6.3)

Una vez seleccionada la línea. Se selecciona la parada inicial y el horario en que va a empezar el servicio. Estos desplegables se cargarán con los datos de la línea seleccionada previamente. (Img. 6.4 y 6.5)

Una vez seleccionados todos los elementos necesarios para la apertura de la expedición, se confirma pulsando Aceptar.

Imagen 6.3: Apertura de expedición - Vehículo

Imagen 6.4: Apertura de expedición – Parada

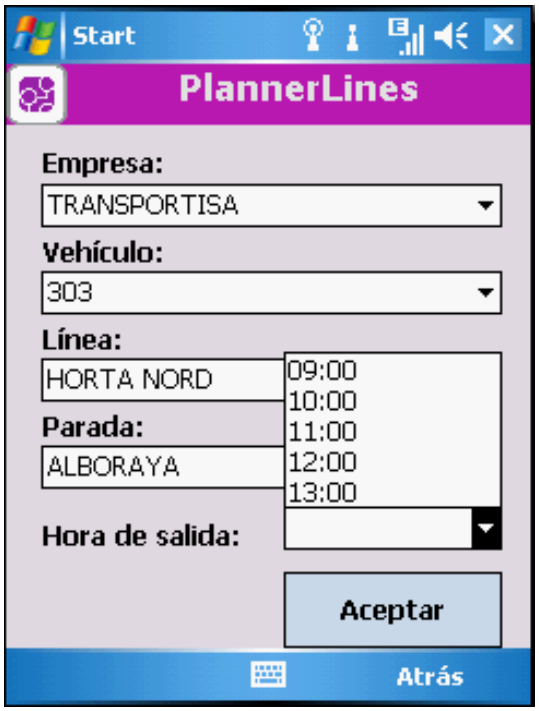

Imagen 6.5: Apertura de expedición - Horario

Imagen 6.6: Apertura de expedición – Fin de la selección

A continuación, se presenta una pantalla para confirmar los datos. (Img 6.7)

En caso de que haya algún dato incorrecto, hay que pulsar Cancelar o Atrás para regresar a la pantalla anterior (Img 6.6) y cambiar los datos.

Si todo está correcto pulsando Aceptar se abrirá la expedición y obtendremos el justificante. (Img 6.8) y se abrirá el menú principal (Img 6.9)

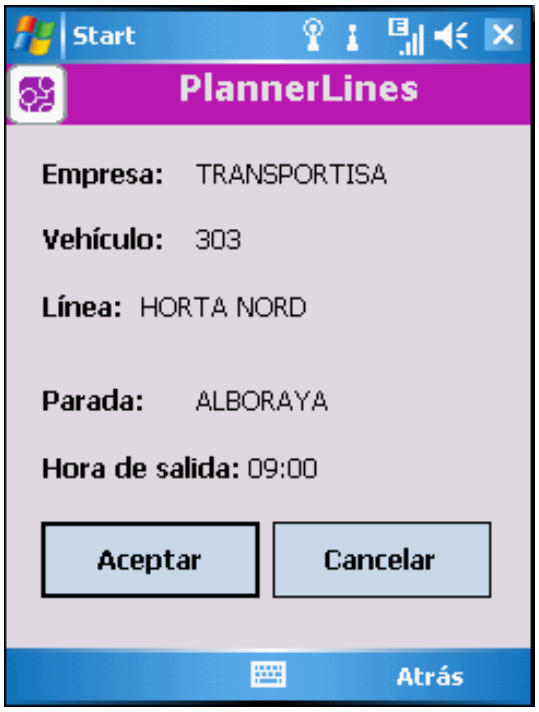

Imagen 6.7: Apertura de expedición - Confirmación

Imagen 6.8: Apertura de expedición - Justificante

6.3 Menú Principal

En la parte superior se muestra la información sobre el estado del sistema, la hora y la línea de la expedición actual y debajo se muestra la parada actual y el número de pasajeros que han subido al vehículo en esa expedición, lo tickets que se han vendido. (Img 6.9)

A continuación, se muestran todas las opciones disponibles ordenadas por uso y prioridad, de forma que las que tienen un uso más frecuente aparecen arriba del todo y por el contrario aquellas que se utilizan menos a lo largo de la jornada aparecen al desplazarse verticalmente. (Img 6.10)

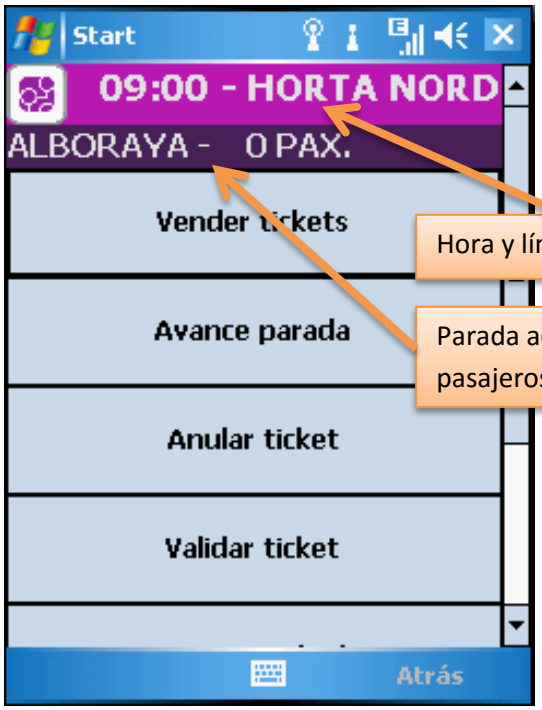

Imagen 6.9: Menú principal (1/2)

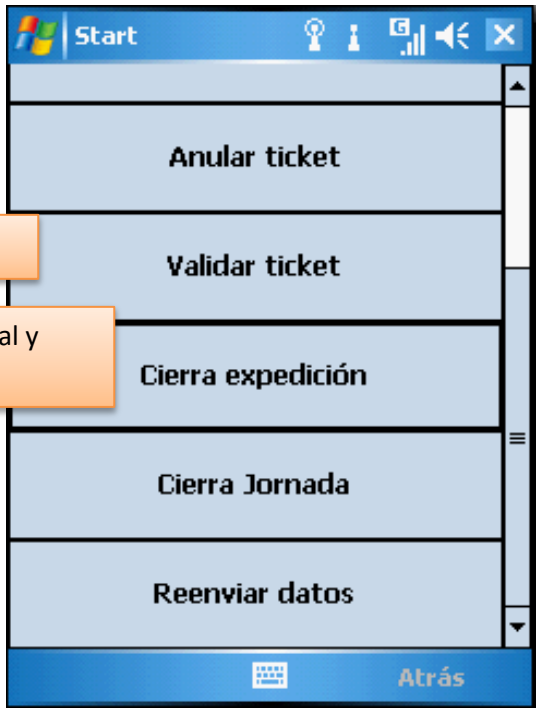

Imagen 6.10: Menú principal (2/2)

6.4 Vender tickets

Pulsando la primera opción del menú se abre la pantalla (Img 6.11). Se muestran las tarifas y los importes y como vemos en la Imagen 6.12 a la derecha de cada fila, hay un campo para seleccionar las unidades de cada tarifa. Una vez seleccionado se pulsa Imprimir.

Imagen 6.11 Venta de tickets

Imagen 6.12: Venta de tickets - Selección

Imagen 6.13: Venta de tickets - Resumen

Imagen 6.14: Ticket impreso

A continuación se abre una pantalla con el resumen de la venta.(Img 6.13) Para hacer algún cambio habría que pulsar atrás. Para confirmar la venta hay que pulsar Imprimir, se imprimirán todos los tickets de la selección y el sistema se posicionará en la pantalla anterior (Img 6.11) y actualizará la información del número de pasajeros en la cabecera de la aplicación.

En la Imagen 6.14 se muestra un ticket.

6.5 Avance de parada

La segunda opción de menú posiciona el sistema en la siguiente parada al pulsar.

Como se puede ver en la Imagen 6.15 el sistema ha cambiado de parada.

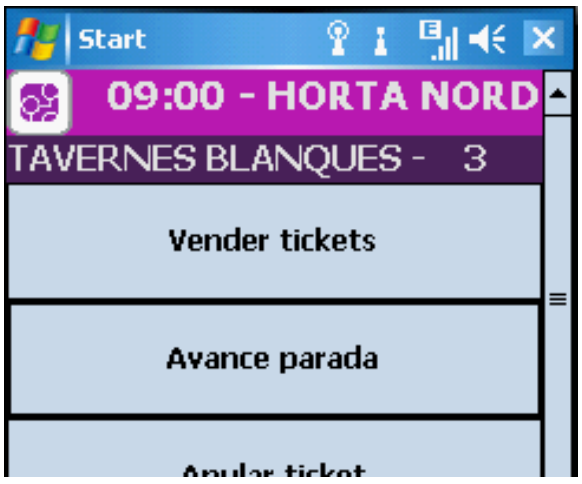

Imagen 6.15: Avance de parada

6.6 Anular Ticket

Pulsando la opción Anular ticket en el menú principal, se abre una pantalla para leer el código de barras del ticket que se va a anular.

Una vez leído, valida que esté dentro del periodo de anulación y si es así muestra la fecha de emisión y el importe (Img 6.16) y al pulsar Anular imprime una copia del ticket con la fecha de anulación.

Si el ticket no se encuentra en el periodo de anulación se muestra un aviso (Img 6.17), aunque permite generar el justificante de anulación por si la primera vez ha ocurrido un problema como que la impresora no tuviese papel.

Imagen 6.16: Anular ticket

Imagen 6.17: Anular ticket - Aviso

6.7 Cierre de expedición

Pulsando en la opción Cierra expedición, se genera el justificante de cierre (Img 6.18). En él se pueden ver los datos relativos a la expedición: Línea, empresa, matrícula del vehículo, hora de cierre, conductor, número de vehículo, parada origen y hora de salida. Y se presenta el número de tickets vendidos agrupados por tipología de tarifa y al pie el número total de tickets vendidos y la recaudación de la expedición.

A continuación, el programa solicita una nueva apertura de expedición (Img 6.19) en el que la empresa y el vehículo aparecen deshabilitados y sólo se puede seleccionar la línea parada y hora que salida. Estos selectores aparecen precargados con la información de la anterior expedición y en cuanto al horario se selecciona el siguiente.

En caso de que no haya que abrir una nueva expedición, pulsando Atrás se regresará al menú principal, en el sólo se podrá cerrar la jornada o enviar datos.

HORTA NORD
TRANSPORTISA
B-1234

CIERRE DE EXPEDICION
26/08/2016 19:24

Conductor: SILVIA
Vehiculo: 303
Linea: HORTA NORD
Parada origen: ALBORAYA
Hora de salida: 09:00
Hora de cierre: 26/08/2016 19:24

TICKETS:----------
Adulto : 2 x 1.20 = 2.40
Nio (5-16) : 1 x 0.50 = 0.50
BONOS AGENCIA:----------
BONOS HOTELES:----------
Unidades : 0

Pasajeros: 3 PAX.
Importe: 2.90 EUR.

Imagen 6.18: Justificante de cierre de expedición

Imagen 6.19: Apertura de nueva expedición

6.8 Cierre de jornada

Pulsando en la opción del menú Cierra jornada, el programa comprueba si tiene alguna expedición abierta, en caso afirmativo cierra la expedición e imprime el justificante (Img. 6.18)

A continuación imprimirá el justificante de cierre de jornada. (Img. 6.20)

Éste es muy parecido al justificante de cierre de expedición, aparecen los datos de cabecera línea, empresa, matrícula del vehículo, fecha y hora de cierre, conductor y número de vehículo. A continuación un resumen de tickets agrupados por tipología de tarifa por cada expedición con su hora de salida y su recaudación. Después también agrupados por tipología aparecen los tickets anulados a lo largo de la jornada y el importe total en tickets anulados y por último, al pie, el número de pasajeros, el número de bonos canjeados, la recaudación total y un código de barras utilizado para la posterior gestión.

Una vez impresos estos justificantes, se envían los datos al servidor y cuando termina, el programa se cierra automáticamente.


HORTA NORD
TRANSPORTISA
B-1234

CIERRE DE JORNADA
26/08/2016 19:25

Conductor: SILVIA
Vehiculo: 303
Linea: HORTA NORD

Expedicion 1: 09:00 2.90
TICKETS:---
Adulto : 2 x 1.20 = 2.40
Nio (5-16) : 1 x 0.50 = 0.50
BONOS AGENCIA:---
BONOS HOTELES:---
Unidades : 0
Pasajeros: 3 PAX.

ANULADOS---
Adulto : 1 x 1.20 = 1.20
Total anulados: 1.20 EUR.

Pasajeros: 2 PAX.
Bonos hotel: 0
Recaudacion:1.70 EUR

00300010260820160000170

Imagen 6.20: Justificante de cierre de jornada

6.9 Envío de datos

Aunque la aplicación sincroniza la información en cada paso de parada, cierre de expedición y cierre de jornada, hay ocasiones en que el cierre de jornada no llega al servidor por problemas de conexión a la red de datos.

Para subir los datos de forma manual, hay que pulsar Reenviar datos en el menú principal (Img 6.10) o desde la pantalla de identificación (Img 6.1).

Una vez pulsado, se abre la pantalla (Img 6.21) donde se puede seleccionar el periodo que se desea enviar.

Al pulsar sobre los campos de fecha se despliega un calendario mensual donde seleccionar el día.

Pulsando Enviar, se enviarán las jornadas de los días seleccionados y se volverá a la pantalla anterior, la de menú principal o la de identificación según el caso desde donde se haya pulsado.

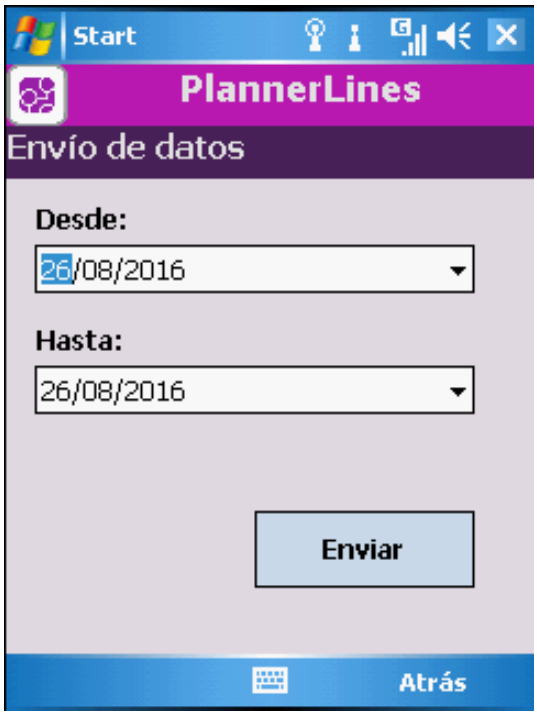

Imagen 6.21: Envío de datos

7. Conclusiones

A la hora de diseñar las interfaces de una aplicación, es muy importante tener en cuenta al usuario. Esta aplicación se ha diseñado para que se pueda utilizar la pantalla táctil a lo largo de toda la aplicación, en el único caso en que se utiliza el teclado es en la pantalla de identificación para ingresar con el código de conductor. Opcionalmente, se puede usar el teclado para seleccionar el número de tickets en la pantalla de Ventas, aunque hemos visto que hay unos botones con flechas arriba y abajo para incrementar o decrementar el número de tickets, en ocasiones en que sean muchos es más sencillo introducir el número por teclado.

Como hemos visto en el apartado de la capa de presentación, para el diseño de las interfaces se han tenido en cuenta algunos de los principios de Gestalt aplicados a las aplicaciones informáticas. La aplicación de estos principios sirven para dirigir la atención del usuario, favorecen el fácil aprendizaje del usuario, que en todo momento y pese a las distracciones del entorno de trabajo sepa donde se encuentra en la aplicación y que se reduzcan los errores ya que los usuarios ven lo que esperan ver.

Otro aspecto importante del proyecto es la capacidad de reponerse de errores no forzados, que pueda seguir trabajando sin conexión a la red de datos ya que en un entorno en que constantemente se encuentra cambiando de ubicación, la calidad de la señal puede variar afectando así a la sincronización en tiempo real, pero no al funcionamiento fundamental del programa en cuanto a la venta se refiere.

Pese a las restricciones en cuanto al sistema operativo de los terminales, ya los únicos que se encontraron rugerizados al inicio del proyecto soportaban Windows mobile. La capa de sincronización con el servidor, el Web Api, podríamos decir que es genérico, se comporta igual sin tener en cuenta el sistema operativo del dispositivo que establece conexión con él. Por ello no se descarta que en un futuro, se reprograme esta aplicación móvil para otros sistemas operativos como Android si se encuentran terminales que encajen en el perfil que se requiere.

8. Bibliografía

- D. Stone, C. Jarrett, M. Woodroffe. User Interface Design and Evaluation. Morgan Kaufmann, 2005.
- B. Shneiderman y C. Plaisant. Designing the User Interface. Pearson 5th ed., 2010
- A. Dix, J. Finlay, G. Abowd, R. Beale. Human-Computer Interaction. Prentice Hall, 2004.
- Lauesen, S. User Interface Design. A Software Engineering Perspective. 2005
- Norman, D. A. The Design of Everyday Things. Basic Books. 2013
- https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/gestalt-principles-of-form-perception
- https://www.packtpub.com/books/content/connecting-microsoft-sql-server-compact-35-visual-studio
- https://msdn.microsoft.com/en-us/library/dd630621.aspx
- https://msdn.microsoft.com/en-us/library/dd721907.aspx
- https://www.ingenuityworking.com/knowledge/w/knowledgebase/1348.net-compact-framework-applications-for-varying-device-screen-sizes.aspx
- https://www.pluralsight.com/courses/aspnetwebapi