UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

SOLUCIÓN MÓVIL PARA GESTIÓN DE DESPACHOS USANDO CÓDIGOS QR

Trabajo Especial de Grado presentado ante la ilustre

Universidad Central de Venezuela

Por el bachiller

Fernando Eliézer Escalona Mendoza C.I. 14952096

Para optar al título de:

Licenciado en Computación

Tutor:

Prof. Antonio Leal

CARACAS, FEBRERO 2012.

ACTA

Quienes suscriben, miembros del Jurado designado por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por el Bachiller Fernando Eliézer Escalona Mendoza, con el título: "Solución móvil para la gestión de despachos usando códigos QR**",** a los fines de optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del jurado, se fijó el día 02 de Marzo del 2012, para que su autor lo defendiera en forma pública, lo que hizo en el aula 1, de la Escuela de Computación, mediante una presentación oral de su contenido, luego de lo cual respondió las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

El Prof. Ronald Pietri sustituye a la Prof. Eleonora Acosta en la defensa del Trabajo Especial de Grado.

En fe de lo cual se levanta la presente Acta, en Caracas a los dos días del mes de marzo del año dos mil doce.

_____

Prof. Antonio Leal

Tutor

_____ _____

Prof. Ronald Pietri — Jurado

Prof. Eugenio Scalise — Jurado

DEDICATORIA

A Dios primeramente, Él con su infinita ayuda supo darme la paciencia y la sabiduría para sacar adelante todo este proyecto.

A mi Mamá por su apoyo y sacrificios durante toda mi vida. Por estar conmigo en todo momento, con su apoyo, sus consejos y sus bendiciones.

A mi abuela Isolina, por su comprensión, cariño incondicional y sus bendiciones.

A mi hermosa familia, especialmente mis tíos Leida, Eglé y Pedro, porque sin su ayuda, su apoyo, sus consejos y su cariño, esto no podría haber sido posible.

Fernando Eliézer Escalona Mendoza

AGRADECIMIENTOS

A Dios, por todas las bendiciones. Por todas las pruebas que me ha puesto a lo largo de mi vida y por la fuerza que me ha dado para superarlas. Por haberme permitido vivir cada momento, que bueno o malo me ha servido de aprendizaje y por demostrar cada día que está conmigo y no me abandona.

A la Universidad Central de Venezuela por ser mi casa de estudios, que me permitió superarme y ser lo que soy ahora. Gracias a todos los que la conforman y que hacen que me sienta orgulloso de ser ucevista.

A la Facultad de Ciencias, por ser el sitio donde estuve en mi carrera; donde conocí gente maravillosa, hice grandes amigos y pasé momentos que siempre recordaré con cariño y con nostalgia. Gracias a todos los profesores, amigos y compañeros que contribuyeron a mi formación como profesional y como persona, brindándome además de conocimientos, sus consejos y apoyo que siempre tendré presentes.

A mi mamá por traerme a este mundo, por ser el soporte fundamental de mi vida y razón principal de todo lo que hago, por haber dado lo mejor de sí para que saliera adelante sin importar cuánto eso le costara, por darme la fuerza y confianza para superar muchos de los obstáculos que se me presentaron, por bendecirme todos los días y confiar en mí a pesar de la distancia, aconsejándome siempre lo mejor. Te amo mamá.

A mi abuela Isolina, por ser mi segunda madre, por su apoyo, su ayuda en todos los sentidos, por haberme dado tanto incondicionalmente con un amor tal, que no me alcanzará la vida entera para retribuírselo.

A mis tías Leida y Eglé Mendoza, por también ser mis segundas madres, por sus consejos, sus regaños y por estar presente en todos los momentos, especialmente en los más difíciles. También ellas fueron parte fundamental para conseguir este logro.

A mi tío Pedro Mendoza por su inestimable apoyo y por abrirme las puertas de su casa en un momento importante. Gracias por ser un padre para mí.

A mi tío Pablo Mendoza y mi tía Catalina por también haberme dado cobijo en su humilde hogar, donde siempre me sentí en familia y pasé momentos muy gratos, nunca tendré como pagarles. Estén donde estén esto también es para ustedes.

A mi tía Carmen Suñé, por tratarme siempre como su sobrino, por recibirme como uno más en su hogar con gran cariño y respeto. A mi tío Alberto Ramunno, quien también me recibió en su hogar tanto a mí como a mi madre en un momento difícil y por ayudarme siempre que lo necesité, se que donde esté él también se alegra por este logro.

A mi madrina Aimara Godoy Escalona, por siempre ser consecuente conmigo y siempre estar pendiente en cada uno de los pasos de mi vida y mi carrera.

A mis hermanos David Moncada, Omar Aguilera y Gabriela Perdomo, por estar ahí siempre, tanto en las buenas como en las malas y sobre todo en las peores. Me siento afortunado de tener a los mejores amigos del mundo.

A mi otro hermano y compañero de luchas y proyectos universitarios Cesar Noriega, lastimosamente el destino no quiso que emprendiéramos este proyecto juntos, sin embargo agradezco a Dios que nuestra amistad y compañerismo se hayan mantenido intactos a pesar de todo.

A mi tutor Antonio Leal, por su gran ayuda durante el desarrollo de esta tesis, sus regaños, sus consejos, su paciencia y por haberme enseñado muchas cosas, además por haberme dado la oportunidad de compartir con él en el ámbito laboral.

A Lisbeth Mavárez, Francisco Páez y todo el personal de Kiberno por su gran ayuda, comprensión y de siempre estar pendiente de mí en el desarrollo de este proyecto.

A todas aquellas personas que de alguna u otra forma me ayudaron y pusieron un granito de arena tanto en el desarrollo de este trabajo como en el transcurso de toda mi carrera y mi vida.

Tabla de contenido

RESUMEN 9
INTRODUCCIÓN 10
OBJETIVOS GENERALES Y ESPECÍFICOS 12
OBJETIVO GENERAL 12
OBJETIVOS ESPECÍFICOS 12
Capítulo 1 Código QR 13
1.1 DEFINICIÓN Y ANTECEDENTES 13
1.2 CARACTERÍSTICAS DEL CÓDIGO QR 15
1.2.1 Capacidad de Almacenamiento 15
1.2.2 Capacidad de corrección de errores 16
1.2.3 Tamaño de impresión 17
1.2.4 Capacidad de codificación Kanji 18
1.2.5 Omnidireccionalidad 19
1.2.6 División y reconstrucción de Códigos QR 19
1.3 CONFIGURACIÓN DE CÓDIGOS QR 20
1.3.1 Tamaño del código QR 20
1.3.2 Versión del símbolo 20
1.3.3 Tamaño del módulo 21
1.3.4 Área del código QR 21
1.4 CREACIÓN DE UN CÓDIGO QR 22
1.5 ALGUNOS USOS DE LOS CÓDIGOS QR 29
Gestión de partes en la industria automotriz 29
Identificación de circuitos electrónicos 29
Control logístico de productos 30
Publicidad 30
Información de establecimientos y empresas 31
Capítulo 2 Edición Micro de Java (JME) 32
2.1 ARQUITECTURA JME 32
2.2 CONFIGURACIONES Y PERFILES EN JME 34
2.2.1 Configuración CLDC 35

2.2.2 Configuración CDC 36
2.2.3 Perfiles 37
2.2.3.1 Perfil MID 37
2.2.3.2 Perfil Foundation 38
2.2.3.3 Perfil Personal 39
2.3 MÁQUINA VIRTUAL DE JAVA EN JME 40
2.4 MIDLET 41
2.4.1 Ejecución de un MIDlet 42
Capítulo 3 Análisis y Diseño de la solución 44
3.1 ACTORES 44
EN ESTA SECCIÓN SE DESCRIBEN LOS ACTORES RELACIONADOS CON EL SISTEMA. 44
3.2 CASOS DE USO 46
EN ESTA SECCIÓN SE PRESENTAN LOS DIAGRAMAS Y DESCRIPCIONES DE LOS CASOS DE USO DEL SISTEMA. 46
3.2.1 Nivel 1 46
EL DIAGRAMA DEL NIVEL 1 PRESENTA UNA VISIÓN FUNCIONAL DE ALTO NIVEL DEL SISTEMA, CON LOS ACTORES QUE INTERACTÚAN CON EL SISTEMA Y LOS CASOS DE USO GENERALES DENTRO DEL SISTEMA, COMO SE MUESTRA EN LA FIGURA 43. 46
3.2.2 Nivel 2 47
EN EL NIVEL 2 SE PRESENTA LOS DIAGRAMAS REFINADOS DE LOS CASOS DE USO GENERALES Y LA DESCRIPCIÓN DE LOS CASOS DE USO. 47
3.2.2.1 Casos de uso Móvil 47
En esta sección se describen los casos de uso implementados en la aplicación móvil. 47
La figura 44 muestra el caso de uso móvil Crear Despacho. 47
La figura 45 muestra el caso de uso móvil Validar Despacho. 49
La figura muestra el caso de uso móvil Recibir Despacho 51
3.2.2.2 Casos de uso Web 53
En esta sección se describen los casos de uso implementados en la aplicación web. 53
La figura 47 describe el caso de uso Consultar Despachos. 53
3.3 INTERACCIONES MÓVIL WEB 54
Capítulo 4 Implementación 56
4.1 IMPLEMENTACIÓN WEB 56

4.1.1 Apache Wicket 56
4.1.2 JSON 59
4.1.3 Aplicación Web de Consulta de Despachos 62
4.2 IMPLEMENTACIÓN MÓVIL 70
4.2.1 Aplicación móvil Creación de Despachos 70
4.2.2 Aplicación móvil Manejo de Despachos 75
Capítulo 5 Conclusiones 86
Capítulo 6 Bibliografía y Referencias 89

RESUMEN

El presente Trabajo Especial de Grado, es el resultado de una propuesta de combinar tecnología de códigos de respuesta rápida (conocidos como QR, por *Quick Response*) y tecnologías móviles para alcanzar la automatización de un proceso de negocio conocido como el despacho de productos.

Se establecieron los casos de uso indispensables para una solución que incorpora códigos QR como artefactos que contienen información acerca de un despacho y los productos asociados.

Los códigos QR pueden ser leídos o capturados usando un dispositivo sencillo que contenga un lector de códigos QR. Para esta experiencia, empleamos un teléfono con una cámara fotográfica y capacidad de acceder a la red Internet, como dispositivo móvil que contiene una aplicación. La aplicación al capturar el código QR apropiado puede invocar a través de la red Internet una transacción en el *backend* administrativo de la solución desarrollada.

La transmisión de datos entre el dispositivo móvil y el *backend* administrativo se realiza empleando servicios web con el formato ligero para intercambio de datos conocido como JSON.

Las aplicaciones móviles están desarrolladas con la edición micro de Java y el *backend* administrativo con el *framework* para aplicaciones web, conocido como Apache Wiket, de la fundación de software Apache.

En el desarrollo del presente Trabajo Especial de Grado, se comprueban las ventajas de utilizar las tecnologías descritas al implementar la solución descrita.

INTRODUCCIÓN

La evolución de los teléfonos móviles y sus tecnologías relacionadas ha impactado notablemente la incorporación de los dispositivos móviles como herramientas esenciales en la vida cotidiana de los humanos. La posibilidad que los teléfonos móviles contengan aplicaciones que ejecuten lógica de alguna naturaleza, tomando en cuenta las capacidades limitadas de estos dispositivos abre nuevas oportunidades y nuevos paradigmas para el desarrollo de aplicaciones de software.

Actualmente, es común que un ciudadano convencional cuente con un dispositivo móvil como lo puede ser un teléfono con capacidad de acceder a su correo electrónico, realizar navegación web y acceder a una cantidad de aplicaciones móviles de diversa naturaleza.

En el área comercial, las organizaciones han adoptado el paradigma móvil como extensión de sus procesos de negocios, desde acceder al correo electrónico hasta sofisticadas aplicaciones para apoyar el negocio en un dispositivo móvil.

La adopción del paradigma móvil en el área comercial ha demostrado inmediatez y ubicuidad para la ejecución de transacciones desde los dispositivos móviles, así como la instantaneidad de acceso a la información. Por ello en la actualidad, el diseño de aplicaciones incorpora la visión móvil en algún punto.

La automatización del despacho de productos requiere que los actores involucrados puedan validar la información de un despacho y los productos asociados a ese despacho a lo largo del proceso que implica la creación, validación y recepción de despachos.

Los códigos QR son una tecnología en evolución que provee una serie de ventajas a la hora de codificar y leer información.

El presente Trabajo Especial de Grado, es el resultado de una propuesta de combinar tecnología de códigos de respuesta rápida (conocidos como QR, por *Quick Response*) y tecnologías móviles para apoyar la automatización de un proceso de negocio conocido como el Despacho de Productos.

Se establecieron los casos de uso indispensables para una solución móvil que incorpora códigos QR como artefactos que contienen información acerca de un despacho y sus productos asociados.

Para esta experiencia, empleamos un teléfono con una cámara fotográfica y capacidad de acceder a la red Internet, como dispositivo móvil que contiene una aplicación, que al capturar el código QR apropiado invoca a través de la red Internet una transacción en un *backend* administrativo de la solución desarrollada.

La transmisión de datos entre el dispositivo móvil y el *backend* administrativo se realiza empleando servicios web con el formato ligero para intercambio de datos conocido como JSON.

Las aplicaciones móviles están desarrolladas con la edición micro de Java y el *backend* administrativo con el *framework* para aplicaciones web, conocido como Apache Wicket, de la fundación de software Apache.

En este Trabajo Especial de Grado se establece el objetivo general y objetivos específicos, a continuación, el primer capítulo trata el tema de los códigos QR y su tecnología relacionada. El segundo capítulo aborda la edición micro de Java para el desarrollo de aplicaciones móviles. El tercer capítulo trata el análisis y diseño de la solución para este Trabajo Especial de Grado. El cuarto capítulo aborda la implementación de las partes web y móvil de la solución. El capítulo 5 contiene las conclusiones y por último la bibliografía empleada para el desarrollo del Trabajo Especial de Grado.

OBJETIVOS GENERALES Y ESPECÍFICOS

A continuación se establece el objetivo general y los objetivos especiales del Trabajo Especial de Grado.

OBJETIVO GENERAL

El objetivo general consiste en integrar un conjunto de tecnologías, en el análisis, diseño y desarrollo de una solución móvil y web que apoye la automatización del proceso de negocios conocido como Gestión de Despacho.

Entre las tecnologías a integrar están los códigos QR, dispositivos móviles inteligentes, *frameworks* para el desarrollo de aplicaciones móviles y web; y un formato ligero para la transferencia de datos.

Para lograr el objetivo general nos propusimos una serie de objetivos específicos, orientados al diseño y desarrollo de una solución móvil y web para la captura y procesamiento de códigos QR, como parte de la solución para la Gestión de Despacho.

OBJETIVOS ESPECÍFICOS

Para satisfacer el objetivo general, en este Trabajo Especial de Grado establecimos una serie de objetivos específicos:

Entre los objetivos específicos se encuentran:

- Uso de un dispositivo móvil con software para la captura del código QR.
- Persistir en el dispositivo móvil la captura del código QR y transmitir la información desde el móvil a un *backend*.
- Procesar la información en el *backend* y realizar una transacción con algún mecanismo de persistencia.
- Diseño y desarrollo de una aplicación móvil para la implementación de las funcionalidades móviles.
- Diseño y desarrollo de una aplicación web para la implementación de las funcionalidades web para el *backend*.
- Diseño y desarrollo de servicios web para integración móvil - *backend*.
- Comprobar la integración de las tecnologías propuestas para el buen desempeño de la solución.

Capítulo 1 Código QR

En este capítulo se abordará el código QR (Quick Response Barcode, Código de Barras de respuesta rápida), con una explicación de sus características, los parámetros usados para la creación de un código QR, así como los usos más resaltantes de esta tecnología.

En la sección 1.1 se presenta la definición y antecedentes del código QR, así como las diferencias en relación al código de barras convencional; en la sección 1.2 se describen algunas de las características más relevantes de los códigos QR y en la sección 1.3 se describe la configuración de los códigos QR. La sección 1.4 presenta una herramienta para crear diferentes tipos de códigos QR. La sección 1.5 describe algunos usos de los códigos QR en diferentes sectores.

1.1 Definición y antecedentes

El código QR es un código bidimensional con una matriz de propósito general diseñada para un escaneo rápido de información. Fue creado por la compañía japonesa Denso-Wave[1], la cual se encarga de manufacturar productos como escáneres de códigos de barra, códigos bidimensionales y robots industriales.

El código QR es un código omnidireccional, es decir que puede leerse en 360 grados, con forma cuadrada y puede ser fácilmente identificado por su patrón de cuadros oscuros y claros en tres de las esquinas del símbolo lo cual procura su alta velocidad de lectura. La figura 1 muestra un código QR convencional.

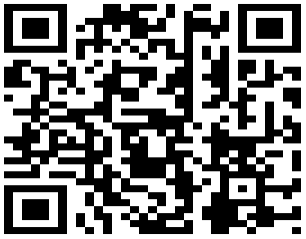


Figura 1 Código de barras QR.

El código QR tiene la capacidad de codificar todos los caracteres ASCII, además de información binaria. Se emplea un lector para obtener la información codificada en el código QR.

[1] Denso-Wave: http://www.denso-wave.com

El código QR es un código de barras para uso público y a diferencia de los códigos de barra habituales que se leen mediante un lector láser, los códigos QR pueden leerse mediante una cámara de video o cámara fotográfica.

Entre muchos usos, en Japón se utiliza comúnmente como medio para comunicar una dirección o dato de contacto y posteriormente, los usuarios pueden tomar nota de esos datos haciendo una fotografía al código. Por ejemplo, con un teléfono móvil o cualquier otro dispositivo que disponga de una cámara podemos almacenar en nuestra agenda los datos de contacto directamente de una hoja de papel o incluso desde la pantalla del computador.

En un principio el código QR se comenzó a utilizar como medio de inventario en la industria, pero su uso se ha extendido a otros usos gracias a la variedad de dispositivos móviles con cámaras, y a varias páginas web que permiten la creación del código QR vía web de manera gratuita e inmediata. La Figura 2 muestra la traducción de una serie de datos alfanuméricos a un Código QR.

Figura 2 Ejemplo de codificación de datos a un código QR.

El código QR es parte de una familia de códigos denominados bidimensionales o de dos dimensiones (2D). Estos deben su nombre a que, a diferencia de los códigos de barras convencionales, codifican información en dos dimensiones, siendo posible así almacenar más datos en un menor espacio.

El código de barra convencional es un código basado en la representación mediante un conjunto de líneas paralelas verticales de distinto grosor y espaciado que en su conjunto contienen una determinada información.

Con la proliferación de los códigos de barras convencionales, el mercado requería que los códigos de barras convencionales almacenaran más información, más tipos de caracteres, y que pudiera ser impreso en un espacio más pequeño.

Como resultado, se realizaron diversas gestiones para aumentar la cantidad de información almacenada por los códigos de barras convencionales, como el aumento del número de dígitos del código de barras o diseñar múltiples códigos de barras.

Sin embargo, estas mejoras también causaron problemas como la ampliación de la zona de código de barras convencional, lo que complica las operaciones de lectura, y encarece la impresión.

Los códigos bidimensionales surgieron en respuesta a estas necesidades. La Figura 3 muestra las diferencias gráficas de un código QR y un código de barras convencional.

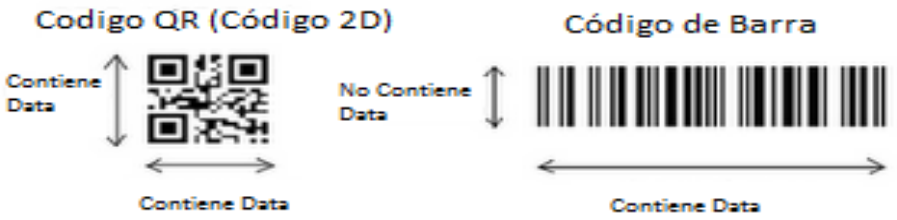


Figura 3 Diferencias visuales entre el código QR y el código de barras convencional.

La Figura 4 muestra la estructura del código QR y sus características principales: información de la versión, formato, corrección de errores, posición, entre otros; que serán explicados a continuación.

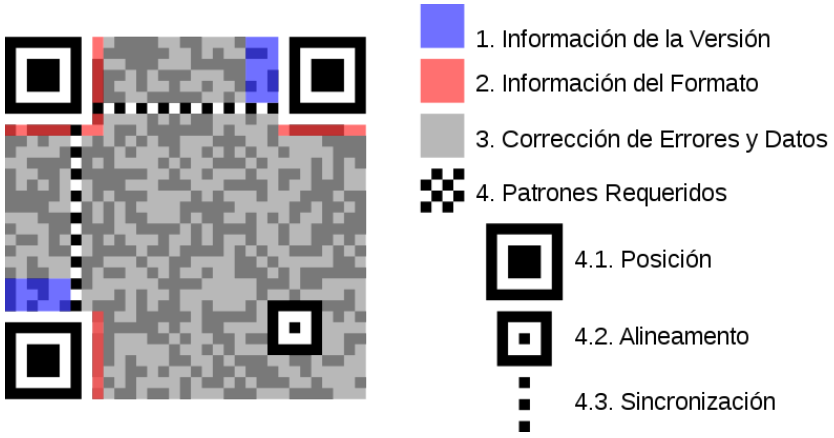


Figura 4 Estructura del código QR.

1.2 Características del código QR

Esta sección presenta algunas características relevantes de un código QR: capacidad de almacenamiento, capacidad de corrección de errores, tamaño de impresión y omnidireccionalidad entre otras.

1.2.1 Capacidad de Almacenamiento

Mientras los códigos de barras convencionales son capaces de almacenar un máximo de aproximadamente 20 dígitos, el código QR es capaz de almacenar desde varias decenas a varios cientos de veces más información.

El Código QR es capaz de manejar todos los tipos de datos, como los caracteres numéricos y alfabéticos, kanji, kana, Hiragana, símbolos binarios y códigos de control. Hasta 7.089 caracteres se pueden codificar en un solo símbolo.

Capacidad de datos del código QR:

- Solo numérico, máximo 7.089 caracteres
- Alfanumérico, máximo 4.296 caracteres
- Binario, máximo 2.953 bytes
- Kanji/Kana, máximo 1.817 caracteres

1.2.2 Capacidad de corrección de errores

Los códigos QR tienen capacidad de corrección de errores. Los datos pueden ser restaurados, incluso si el código QR está parcialmente sucio o dañado. Un máximo de 30% de datos dañados se pueden restaurar.

Cuatro niveles de corrección de errores están disponibles para que los usuarios elijan de acuerdo con el entorno operativo. El aumento de este nivel no sólo mejora la capacidad de corrección de errores sino que también aumenta la cantidad de datos codificados en el código QR.

A continuación los niveles disponibles de recuperación de data:

- Nivel L 7% de la data se puede restaurar
- Nivel M 15% de la data se puede restaurar
- Nivel Q 25% de las claves se pueden restaurar
- Nivel H 30% de las claves se pueden restaurar

Para seleccionar el nivel de corrección de errores, se toman en cuenta varios factores tales como el entorno operativo y el tamaño del código QR.

La función de corrección de errores del código QR se lleva a cabo mediante la adición de un código Reed-Solomon a los datos originales.

Los códigos Reed-Solomon es un método de corrección de errores matemáticos utilizados generalmente para los discos compactos de música, telefonía móvil, sondas espaciales, comunicaciones por satélite, en la transmisión digital de televisión, así como en los sistemas xDSL de comunicación por cable. La tecnología fue desarrollada originalmente como una medida contra el ruido de comunicación de los satélites artificiales y sondas planetarias. Es capaz de hacer una corrección a nivel de byte, y es conveniente para los errores de ráfaga concentrada.

La capacidad de corrección de errores depende de la cantidad de datos que debe corregirse. Por ejemplo, si hay 100 palabras clave de código QR a ser codificados, de las cuales 50 deben ser corregidos, se requieren 100 palabras clave de código Reed-Solomon, ya que estos códigos requieren el doble de la cantidad de palabras claves que deben corregirse.

En este caso, las palabras clave en total son 200, 50 de las cuales se pueden corregir. Por lo tanto, la tasa de corrección de errores para las palabras en clave total es de 25%. Esto corresponde al Nivel Q de corrección de errores del código QR.

Los Niveles Q o H pueden ser seleccionados para ambientes de fábrica en donde el código QR es más sensible a ensuciarse o dañarse, mientras que el nivel L puede ser seleccionado para ambientes limpios, con gran cantidad de datos. Por lo general, el nivel M (15%) es el más frecuentemente seleccionado.

La Figura 5 ilustra la forma en que el código QR maneja la distorsión. En estos casos se agregaron o eliminaron pixeles del código original para examinar el nivel de distorsión de los bordes. Las dos imágenes a las que se les alteraron los datos todavía son reconocibles y usan el nivel "L" de corrección de errores. Aún borrando o agregándole datos, el código puede ser reconocido.

Código QR fuente Datos borrados Datos agregados

Figura 5 Estructura de corrección de errores.

El código QR es resistente a la suciedad y a daños. Los datos pueden ser restaurados incluso si el símbolo fue parcialmente dañado o ensuciado. La Figura 6 muestra los casos en que un código QR puede ser dañado y/o ensuciado, y aún es posible leerlo.

Figura 6 Corrección de errores: el código a la izquierda está sucio y el código a la derecha está dañado.

1.2.3 Tamaño de impresión

El código QR contiene información tanto horizontal como verticalmente, lo cual permite codificar la misma cantidad de datos en aproximadamente una décima parte del espacio de un código de barras tradicionales. (Para un tamaño de impresión más pequeños, existe un micro código QR).

El micro código QR es de menor tamaño que el código QR convencional que se adapta a las aplicaciones que requieren un espacio más pequeño y el uso de pequeñas cantidades de datos, tales como identificación de placas de circuito impresos y componentes electrónicos, etc.

La Figura 7 muestra la comparación de un código de barras convencional y un micro QR.

Figura 7 Comparación visual del código de barras convencional y el micro QR.

La figura 8 muestra la comparación entre códigos QR convencional y micro.

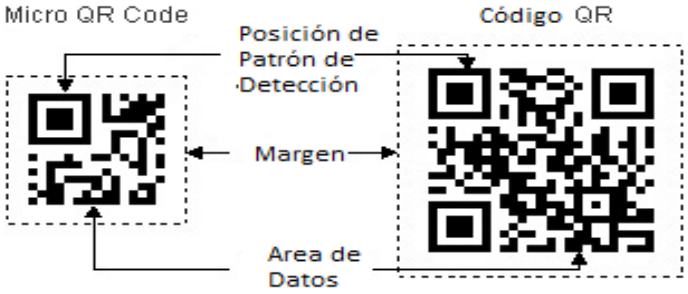


Figura 8 Comparación visual de un micro QR y un código QR.

1.2.4 Capacidad de codificación Kanji

El código QR fue desarrollado en Japón, y contiene capacidad de codificar JIS (Japanese Industrial Standards) nivel 1 y nivel 2 de caracteres Kanji. En el caso de los japoneses, un caracter Kanji es codificado en 13 bits, permitiendo al código QR almacenar 20% más comparado con otras simbologías 2D. La Figura 9 muestra la traducción de un texto en caracteres nipones a un código QR.

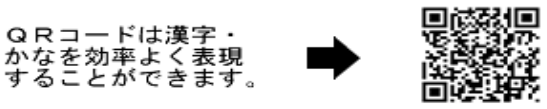


Figura 9 Traducción Kanji a código QR.

1.2.5 Omnidireccionalidad

El código QR tiene capacidad de lectura de alta velocidad desde cualquier posición, esto se denomina omnidireccionalidad. El código QR permite esto a través de patrones de detección de posición situados en las tres esquinas del símbolo. Estos patrones de detección de posición garantizan lectura estable de alta velocidad, evitando los efectos negativos de las interferencias. La Figura 10 muestra estos patrones en el código QR que le otorgan la capacidad omnidireccional.

Figura 10 Características omnidireccionales del código QR.

1.2.6 División y reconstrucción de Códigos QR

Un código QR se puede dividir en múltiples áreas de datos. Inversamente, la información almacenada en múltiples códigos QR puede ser reconstruida como símbolos de datos en un código único.

Un símbolo de datos puede ser dividido hasta en 16 sub símbolos, permitiendo la impresión en un área estrecha. La Figura 11 muestra como un código QR puede ser dividido en varios códigos QR conservando la misma información.

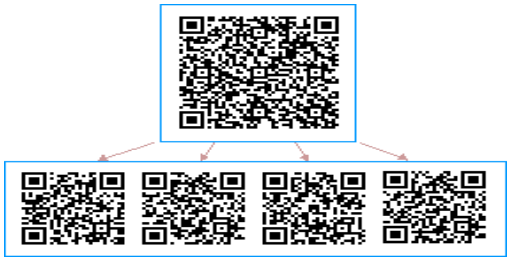


Figura 11 El código QR dividido en cuatro sub símbolos, los datos se pueden leer ya sea desde el símbolo origen o en los cuatro sub-símbolos.

1.3 Configuración de códigos QR

En esta sección se trata la configuración de un código QR. Se describe el proceso para determinar el tamaño y la cantidad de datos que se almacenarán en un código QR.

1.3.1 Tamaño del código QR

Para determinar el tamaño del código QR se emplea: la versión del símbolo, la base de capacidad de datos, el tipo de carácter y el nivel de corrección de errores. Además, el rendimiento de impresión o el escáner para la lectura, influyen en el proceso. La figura 12 muestra los parámetros para determinar el tamaño del código QR.

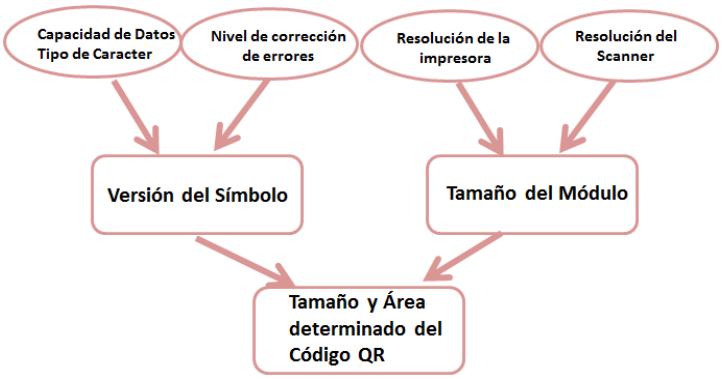


Figura 12 Parámetros empleados para determinar el tamaño y área del código QR.

1.3.2 Versión del símbolo

Las versiones de símbolo de código QR van desde la versión 1 a la versión 40. Cada versión provee un módulo único o un número de módulos. El módulo define la composición del código QR. La configuración del módulo, se refiere al número de módulos contenidos en un símbolo, a partir de la versión 1 (21 × 21 módulos) hasta la versión 40 (177 × 177 módulos). Cada número de versión mayor se compone de 4 módulos adicionales.

La Figura 13 muestra códigos QR con distintas versiones del símbolo.

Figura 13 Versiones de símbolo de un código QR.

Cada versión de símbolo de código QR tiene la capacidad de datos máxima de acuerdo a la cantidad de datos, tipo de carácter y el nivel de corrección de errores. A medida que la cantidad de datos aumenta, deben incluirse más módulos en el código QR, lo que resulta en símbolos más grandes del código QR.

1.3.3 Tamaño del módulo

Una vez determinada la versión del símbolo, se establece el tamaño de impresión del código QR. Cuánto más grande sea el módulo, será más fácil de capturar con un lector de códigos QR, y requerirá un área de impresión más grande.

Una vez considerado los factores pertinentes, es necesario determinar el tamaño del módulo de cada aplicación. Se recomienda que los símbolos del código QR se impriman lo más grande posible dentro del área de impresión disponible. La Figura 14 muestra dos códigos QR Versión 1 de 21 x 21 módulos de diferentes tamaños.

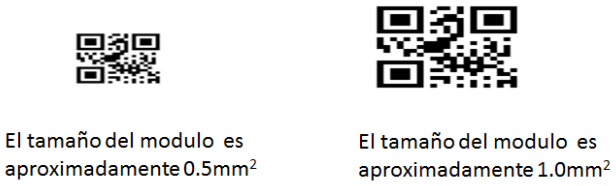


Figura 14 Versión 1 del código QR (21 × 21 módulos).

1.3.4 Área del código QR

Una vez determinada la versión de símbolo y el tamaño del módulo, se establece el tamaño del código QR. El área del código QR requiere un margen o *zona neutra* en torno al código QR. El margen es un área limpia alrededor del símbolo sin impresión. La Figura 15 muestra el margen de un código QR.

Figura 15 Margen del código QR.

1.4 Creación de un código QR

Los usos de códigos QR son múltiples y variados, pueden variar desde algo simple como un URL para un recurso web, un texto plano, hasta números telefónicos y envío de mensajes de texto.

En esta sección se describen algunos ejemplos de cómo crear códigos QR. Se utilizó el generador de códigos QR: http://qrcode.kaywa.com/.

Al acceder este URL se observa la interfaz en la figura 16, donde el usuario puede seleccionar el tipo de contenido con el cual desea crear el código QR.

El tipo de contenido puede ser de tipo: URL, texto simple, número telefónico ó SMS. Adicionalmente, el usuario puede seleccionar el tamaño del código QR, ya sea pequeño, medio, grande y extra grande.

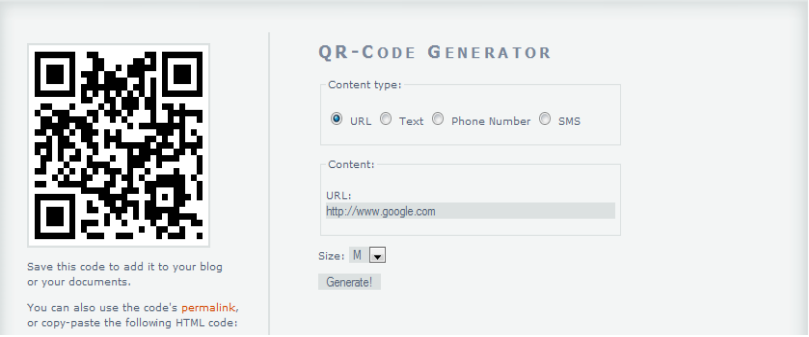


Figura 16 Generador de códigos QR para codificar el URL http://www.google.com.

Al hacer clic en el botón *Generate* se genera la imagen del código QR solicitado en la parte izquierda de la interfaz web, donde podrá abrirse y/o descargarse. La figura 17 muestra el código QR generado para el tipo de contenido URL, el cual contiene el URL http://www.google.com.

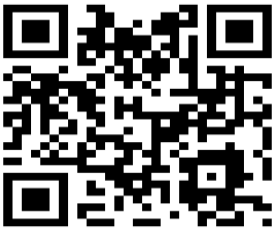


Figura 17 Código QR que contiene el URL http://www.google.com.

Al escanear el código generado, usando el lector del dispositivo móvil como se muestra en la figura 18, se visualiza el URL contenido en el código QR (http://www.google.com**)** en el navegador web del dispositivo móvil como se muestra abajo en la figura 19.

Figura 18 Lector de códigos QR **UpCode** en un dispositivo móvil Nokia**.**

Figura 19 Navegador web en el dispositivo móvil una vez capturado el código QR con el URL codificado.

Si desea generar un código QR que contenga contenido de tipo texto simple, se selecciona la opción *Text*, y se ingresa el texto deseado y se genera el código, como muestra la figura 20.

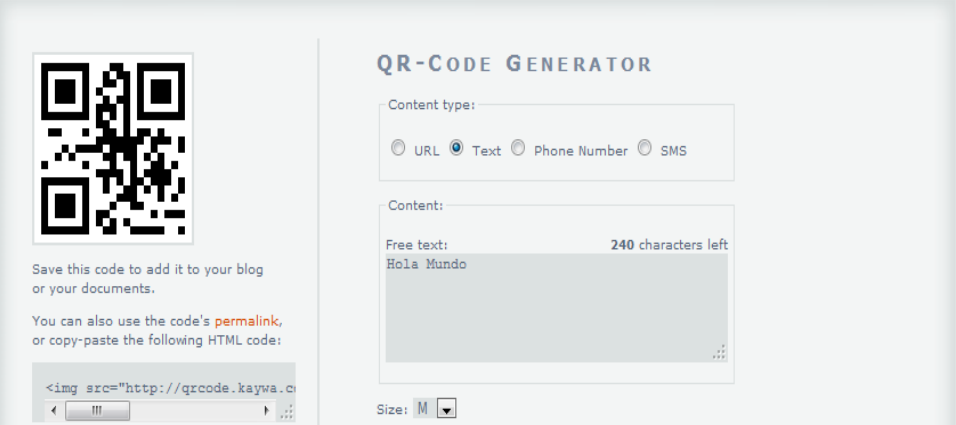


Figura 20 Generador de códigos QR para codificar el texto "Hola Mundo".

El código generado se muestra en la figura 21.

Figura 21 Código QR del texto "Hola Mundo".

Al escanearse el código generado, usando el lector en el dispositivo móvil como se muestra en la figura 22, se visualiza en el móvil el texto contenido en el código QR como se muestra en la figura 23.

Figura 22 Lector de códigos QR **UpCode** en un dispositivo móvil Nokia leyendo el código anterior.

Figura 23 Vista en el dispositivo móvil una vez capturado el código QR con texto codificado.

Si se desea generar un código QR con contenido de tipo número telefónico, seleccione la opción *Phone Number* se ingresa el número telefónico deseado y se genera el código, como muestra la figura 24. El código QR generado se muestra en la figura 25.

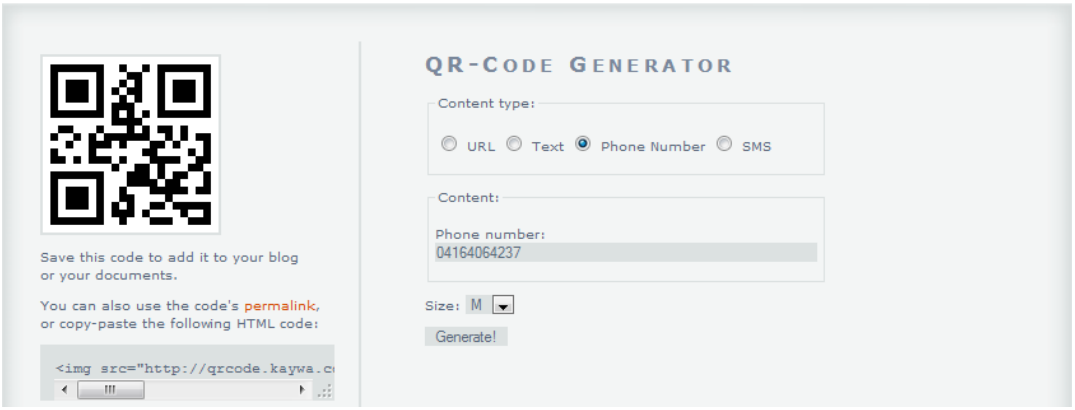


Figura 24 Generador de códigos QR para codificar un número telefónico.

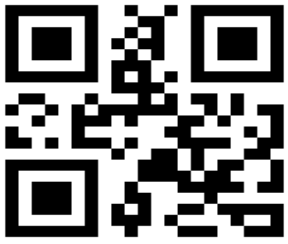


Figura 25 Código QR con un número telefónico codificado.

Al escanearse el código generado, a través del software lector instalado en el dispositivo móvil, el dispositivo le indica al usuario si desea hacer una llamada al número telefónico contenido en el código QR, como se muestra en la figura 26.

Figura 26 Vista en el dispositivo móvil luego de capturar el código QR con el número telefónico codificado, indicando si desea hacer la llamada al número indicado.

Si se desea generar un código QR que contenga un mensaje de texto (SMS) que requiera enviar a un número telefónico específico, se selecciona la opción SMS se ingresa el número telefónico de destino del SMS deseado, se ingresa el mensaje a enviar y se genera el código, como muestra la figura 27. El código generado se muestra en la figura 28.

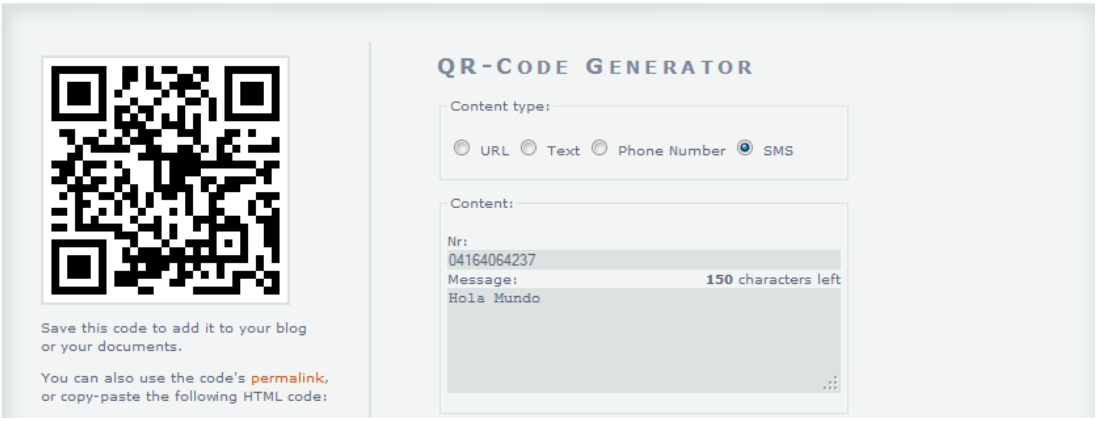


Figura 27 Generador de códigos QR con un SMS al mismo número telefónico, con el texto "Hola Mundo".

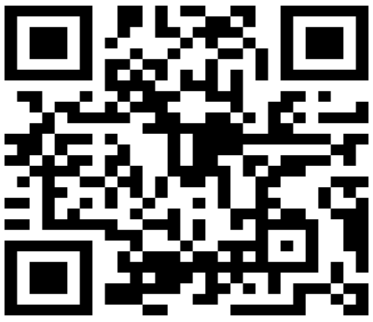


Figura 28 Código QR de un SMS al mismo número de la figura anterior con el mensaje "Hola Mundo".

Al capturar el código generado en el dispositivo móvil, se abre la interfaz de envío de SMS del dispositivo, con el número del receptor del mensaje y el texto del mensaje ya establecidos, tal como se muestra en la figura 29.

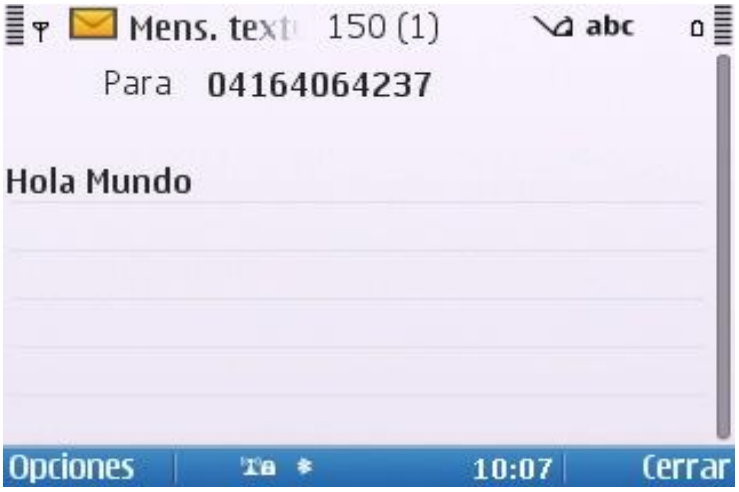


Figura 29 Vista en el dispositivo móvil una vez capturado el código QR, se abre la interfaz de envío de SMS con el número y el texto especificado.

1.5 Algunos usos de los códigos QR

Esta sección describe algunos usos de los códigos QR en distintos sectores.

Gestión de partes en la industria automotriz

En la industria automotriz, el transporte y recepción de productos están codificados con código QR que contienen los datos del cliente, los datos de remitente, el número de producto, cantidad, y otros datos. Los datos se utilizan para ordenar y analizar los productos. La figura 30 muestra el uso de códigos QR para la gestión de productos en la industria automotriz.

Figura 30 Uso de códigos QR para gestión de partes en la industria automotriz.

Identificación de circuitos electrónicos

En un fabricante de componentes electrónicos, las tarjetas de circuitos electrónicos se codifican con códigos QR que contienen la fecha de fabricación, cadena de producción, número de serie y otros datos relevantes a la tarjeta. Los datos se utilizan para el control del proceso y la configuración automática. La figura 31 muestra el uso de códigos QR para identificar tarjetas electrónicas.

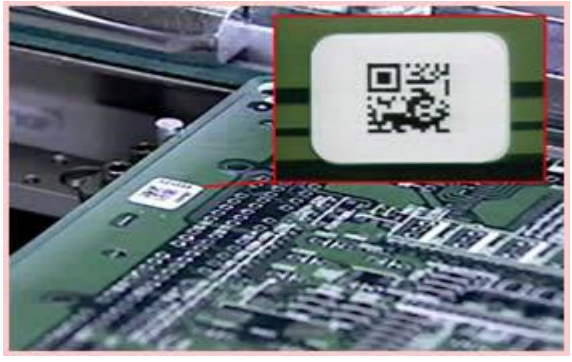


Figura 31 Uso de códigos QR para identificar tarjetas con circuitos electrónicos.

Control logístico de productos

El código del producto, fecha de vencimiento, bitácora de fabricación, y otros datos se codifican en un código QR. Los datos se utilizan para la gestión de logística de los productos. La figura 32 muestra productos en cajas que poseen un código QR asociado con su información.

Figura 32 Uso de códigos QR en un sistema de logística de productos alimenticios.

Publicidad

El portal web argentino infobae.com ha sacado una serie de publicidades en distintas revistas, promocionando su aplicación para dispositivos móviles Blackberry®. A fin de facilitar la descarga, se utiliza un código QR que contiene el URL para descargar la aplicación. La figura 33 muestra un código QR que permite descargar la aplicación móvil del portal infobae.com.

Figura 33 Uso de códigos QR en publicidad.

Información de establecimientos y empresas

La empresa Google ha integrado los códigos QR dentro de su servicio de lugares favoritos en su aplicación de geocodificación *Google Maps*, para proporcionar al usuario información detallada sobre un establecimiento en particular. La figura 34 muestra la integración de los códigos QR con lugares favoritos de *Google Maps.*

Figura 34 Integración de códigos QR en los lugares favoritos de Google Maps.

Lugares favoritos (*Favorite Places*) es un servicio basado en Google Maps que permite al usuario obtener información detallada sobre un establecimiento en particular, a partir de la lectura del código QR que éste tiene expuesto a pie de calle mediante una adhesivo que les entrega Google, de manera que desde nuestro dispositivo móvil podemos obtener vía internet información sobre la tienda o negocio simplemente a partir de la lectura del código QR. La figura 35 muestra el código QR en la puerta de establecimiento, que al ser leído, mostrará la información de dicho establecimiento.

Figura 35 Código QR en un establecimiento.

Capítulo 2 Edición Micro de Java (JME)

El siguiente capítulo aborda el lenguaje Java para desarrollar aplicaciones para un amplio rango de dispositivos diferentes a una computadora personal. Esta versión del lenguaje es denominada la Edición Micro de Java, por su acrónimo en inglés JME (Java Micro Edition). JME combina una máquina virtual de Java con un conjunto de APIs de Java para desarrollar aplicaciones para dispositivos móviles. La versión JME fue diseñada y provista por Sun Microsystems®.

La sección 2.1 describe la arquitectura JME, la sección 2.2 describe las configuraciones y perfiles en la arquitectura JME. La sección 2.3 trata la máquina virtual de Java en JME y la sección 2.4 describe los Midlets en la arquitectura JME.

2.1 Arquitectura JME

La versión de Java para desarrollar aplicación para dispositivos móviles se denomina JME, el acrónimo de Java Micro Edition (Java Edición Micro). Esta versión del lenguaje de desarrollo Java está orientada a un amplio rango de dispositivos diferentes a una computadora personal o servidor. Java ME fue concebido originalmente con el fin de hacer frente a las dificultades relacionadas con la creación de aplicaciones para dispositivos con capacidades limitadas de memoria, pantalla y energía.

La versión micro de Java no define un nuevo tipo de lenguaje de desarrollo, adapta el lenguaje Java para dispositivos.

El problema que enfrentaban los diseñadores del lenguaje Java, la empresa Sun Microsystems, y la industria móvil en general, es el amplio rango de dispositivos al que debía estar orientado la edición Java Micro, donde los dispositivos más limitados manejan desde 1 KB de memoria RAM y 24 KB de memoria ROM, con pequeñas pantallas y en algunos casos sin pantallas; mientras que los dispositivos más grandes y equipados, se aproximan a las capacidades de los laptops o computadoras personales.

Por lo que JME debía ser extremadamente flexible para ajustarse a las necesidades de los dispositivos móviles. Actualmente, el rango de dispositivos móviles en cuanto a características como memoria, velocidad y otras capacidades es bastante amplio.

Por lo tanto, la arquitectura JME pudo haber sido definida bajo los siguientes enfoques:

1. una arquitectura que contendría cada API disponible y/o posible, pero esto implicaría que los dispositivos de menor capacidad no responderían adecuadamente debido a las limitaciones.
2. una arquitectura común de menor denominación para los dispositivos, lo cual implicaría retirar la mayor parte del API de JME.

Durante la fase de diseño de JME, se consideró que ninguno de los enfoques era el indicado y la decisión fue que JME debía ser extremadamente modular.

Procurando la modularidad en la arquitectura JME, se definieron Configuraciones, Perfiles y Paquetes.

Las configuraciones están orientadas al dispositivo, mientras que los perfiles son orientados a aplicaciones.

Cada configuración provee un conjunto mínimo de características que cada dispositivo en la configuración debe proveer.

Un perfil agrega clases específicas a una configuración JME en particular o para usos específicos de dispositivos, un ejemplo, son clases para interfaz de usuario, mecanismos de persistencia e infraestructura de mensajería, entre otras.

Los paquetes son opcionales, y su objetivo es proveer un conjunto adicional de APIs de propósito particular.

Una combinación de configuraciones, perfiles y paquetes, es creada para la capacidad de procesamiento y entrada/salida de una categoría específica de dispositivos.

Sun Microsystems generó una referencia de implementación para las configuraciones y los perfiles, de manera que los fabricantes de dispositivos puedan implementarla.

Los perfiles y configuraciones son provistos por los fabricantes de dispositivos y son embebidos en los dispositivos. Los paquetes opcionales no son provistos por los fabricantes de dispositivos, están diseñados para que los desarrolladores los distribuyan con las aplicaciones móviles desarrolladas.

La figura 36 muestra la arquitectura JME.

Figura 36 Arquitectura JME.

2.2 Configuraciones y Perfiles en JME

Las configuraciones y perfiles permiten la modularidad y flexibilidad de JME.

Una configuración define una API de Java y los requerimientos mínimos de procesamiento y memoria en un dispositivo, para soportar la ejecución de aplicaciones móviles. Las configuraciones están orientadas a grupos de dispositivos con capacidades e interfases de usuario similares.

Actualmente, hay dos clases de configuraciones en la arquitectura JME:

1. CLDC, el acrónimo en inglés de Configuración de Dispositivo Limitado Conectado (Connected Limited Device Configuration), para dispositivos móviles y organizadores digitales personales.
2. CDC, el acrónimo en inglés de Configuración de Dispositivo Conectado (Connected Device Configuration), para dispositivos móviles inteligentes, organizadores digitales personales, localizadores, sistemas de navegación en automóviles, televisores con Internet, y decodificadores de televisión y dispositivos con Symbian[2].

Las configuraciones CLDC y CDC se combinan con uno o más perfiles para ofrecer a los desarrolladores una plataforma para crear aplicaciones en dispositivos con diferentes recursos.

La figura 37 muestra ambas configuraciones en la arquitectura JME.

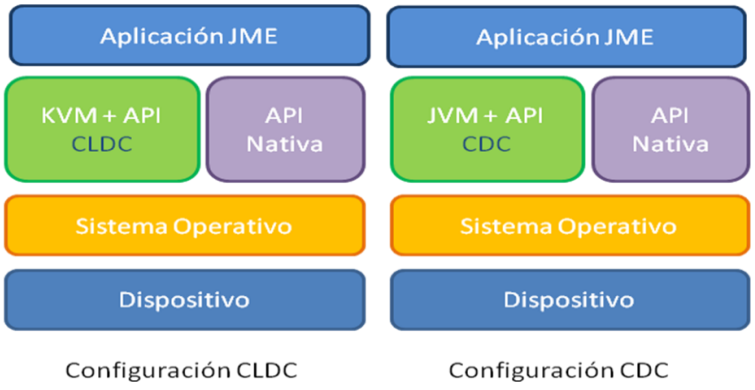


Figura 37 Configuraciones CLDC y CDC de la arquitectura JME.

Los perfiles trabajan sobre las configuraciones y están orientados a una familia de dispositivos. Un perfil permite especificar el subconjunto de APIs de Java, librerías de clases y características de la máquina virtual orientadas a una familia específica

[2] Symbian es un sistema operativo móvil http://licensing.symbian.org/.

de dispositivos. Los perfiles contienen las clases de Java que se enfocan en implementaciones específicas tales como interfaces de usuario y persistencia de datos. A continuación se describen ambas configuraciones.

2.2.1 Configuración CLDC

Una configuración define un API reducido de Java estándar, así como los requerimientos mínimos de procesamiento y memoria en un dispositivo móvil para soportar la ejecución de aplicaciones. La configuración CLDC está orientada a dispositivos tales como teléfonos móviles u organizadores digitales personales hasta cierto rango de características.

La especificación CLDC 1.0 fue liberada en el año 2000, y la especificación actual, CLDC 1.1, fue liberada en el año 2003. La especificación 1.1 es totalmente compatible con la versión anterior.

La especificación CLDC 1.1 requiere que el dispositivo tenga al menos 160 kilobytes de memoria para almacenar la implementación CLDC y un mínimo de 32 kilobytes disponibles para la pila de ejecución. La especificación CLDC no provee medios para persistir datos o crear interfaces gráficas para el usuario.

Con algunas excepciones, el API definido por CLDC es un subconjunto de los paquetes Java estándar java.lang, java.util y java.io. Adicionalmente, está definido un *framework* de conexión genérico en el paquete javax.microedition.io, la implementación del *framework* de conexión genérico no está incluida en la configuración.

La tabla 1 describe los paquetes incluidos en la configuración CLDC.

Paquete CLDC	Descripción
java.io	Clases e interfaces estándar de E/S.
java.lang	Clases básicas del lenguaje.
java.util	Clase de utilidades estándar
javax.microedition.io	Framework de conexión.

Tabla 1. Paquetes en la configuración CLDC.

La máquina virtual de Java para la configuración CDLC, denominada KVM, será descrita más adelante. Esta máquina virtual se ejecuta sobre el núcleo de la configuración CLDC.

La configuración CLDC utiliza la memoria no volátil del dispositivo para almacenar la máquina virtual de Java y las clases de Java en tiempo de ejecución. Cabe destacar que puede alojar otra máquina virtual creada para algún dispositivo en particular.

2.2.2 Configuración CDC

La configuración CDC tiene básicamente dos objetivos fundamentales, el primero es proveer soporte a dispositivos con recursos limitados; el segundo, garantizar compatibilidad con la tecnología que ofrece JSE, permitiendo a los desarrolladores aprovechar las librerías y herramientas que ofrece la tecnología Java.

La configuración CDC soporta las especificaciones de la máquina virtual de Java, incluyendo soporte de punto flotante y librerías para manejo de hilos y seguridad.

A nivel de librerías de clase, CDC utiliza librerías de JSE[3] cuyas implementaciones han sido optimizadas para entornos con memoria limitada, por lo que algunas de las librerías han modificado sus interfaces, mientras que otras se han eliminado por completo. El resultado es un entorno flexible para aplicaciones Java que se adapta a especificaciones limitadas de 2 MB de memoria RAM o superior. Las características de CDC están contenidas mayormente en el paquete javax.microedition.io el cual provee soporte para transferencias tipo http y comunicación basadas en datagramas tipo UDP.

Las librerías incluidas en esta configuración se describen en la Tabla 2.

Nombre de Paquete CDC	Descripción
java.io	Clases e interfaces estándar de E/S
java.lang	Clases básicas del lenguaje
java.lang.ref	Clases de referencias
java.lang.reflect	Clases e interfaces de reflexión
java.math	Clases de matemáticas
java.net	Clases e interfaces de red
java.security	Clases e interfaces de seguridad
java.security.cert	Clases de certificados de seguridad.
java.text	Clases de texto
java.util	Clase de utilidades estándar
java.util.jar	Clases y utilidades para archivos JAR
java.util.zip	Clases y utilidades para archivos ZIP
javax.microedition.io	Clases e interfaces para conexión

Tabla 2. Paquetes en la configuración CDC.

[3] JSE: acrónimo en inglés de Java Standard Edition, la edición estándar de Java.

La figura 38 muestra una relación entre las configuraciones CLDC y CDC y el API de JSE. La configuración CDC es un subconjunto del API JSE, a su vez la configuración CLDC es un subconjunto de la configuración CDC.

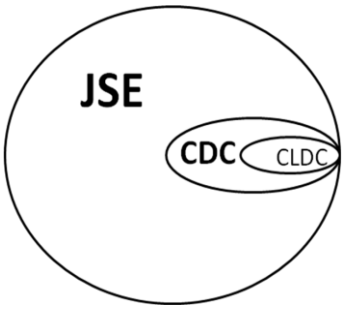


Figura 38 Relación entre configuraciones CLDC, CDC y la edición Java JSE.

2.2.3 Perfiles

Los perfiles en JME están definidos por una agrupación de APIs específicos a una familia de dispositivos. Los perfiles identifican a los diferentes grupos de dispositivos según la funcionalidad que proporcionan, como teléfonos móviles, electrodomésticos, y otra variedad dispositivos; y definen el tipo de aplicación que se ejecutará en los dispositivos. En muchos casos, los perfiles contienen librerías de interfaz gráfica y persistencia de datos, particulares a familias de dispositivos.

Actualmente, para la configuración CLDC se utilizan los perfiles MID Profile (MIDP) y PDA Profile. Para la configuración CDC se utilizan los perfiles Foundation Profile, Personal Profile y RMI Profile.

Una de las combinaciones más comunes en el mercado de dispositivos móviles, es la configuración CLDC y el perfil MIDP, acrónimo en inglés de Mobile Information Device Profile, en castellano, Dispositivo de Información Móvil.

Para describir algunos perfiles, se seleccionaron los perfiles MID, Foundation y Personal.

2.2.3.1 Perfil MID

El perfil MID es una combinación apropiada para la configuración CLDC, que procura minimizar el uso de memoria y energía requerida para dispositivos con recursos limitados. Provee un API básico para el desarrollo de aplicaciones; por ejemplo, provee el paquete javax.microedition.lcdui que permite crear elementos de interfaz gráfica que pueden ser desplegados en un dispositivo con la combinación MIDP y CLDC.

Los paquetes incluidos en este perfil se describen en la tabla 3.

Paquete perfil MID	Descripción
javax.microedition.lcdui	Clases e interfaces para GUIs
javax.microedition.rms	Clases para soporte de Record Management Storage
javax.microedition.midlet	Clases de definición de la aplicación
javax.microedition.io	Clases e interfaces de conexión genérica
java.io	Clases e interfaces de E/S esenciales
java.lang	Clases e interfaces del lenguaje Java
java.util	Clases e interfaces de utilidades estándar

Tabla 3. Paquetes Java incluidos en el perfil MIDP.

2.2.3.2 Perfil Foundation

Este perfil ofrece un API para la configuración CDC dirigida a dispositivos que carecen de interfaz gráfica, incluye algunos paquetes de JSE pero excluye totalmente los paquetes java.awt y java.swing que constituyen la interfaz gráfica del usuario. En caso que una aplicación requiera interfaz gráfica tendrá que implementar un perfil adicional que lo soporte.

Los paquetes incluidos en este perfil se describen en la Tabla 4.

Paquete perfil Foundation	Descripción
java.lang	Clases e interfaces del lenguaje Java.
java.util	Clases e interfaces de utilidades estándar.
java.net	Clases e interfaces de conectividad.
java. io	Clases e interfaces de E/S esenciales
java.text	Soporte para internalización.
java.security	Incluye códigos y certificados.

Tabla 4. Paquetes Java incluidos en el perfil Foundation.

La pila de software para la combinación CDC y perfil Foundation se muestra en la figura 39.

Figura 39 Configuración CDC y el perfil Foundation.

2.2.3.3 Perfil Personal

Este perfil viene dado por un subconjunto de la plataforma JSE, facilitando un entorno con un completo soporte gráfico AWT, la meta es proporcionar a la configuración CDC una interfaz gráfica completa, con capacidades web. Es necesaria una implementación del perfil Foundation para el perfil Personal.

Los paquetes incluidos en este perfil se describen en la Tabla 5.

Paquete perfil Personal	Descripción
java.applet	Clases para el manejo de applets.
java.awt	Clases para crear interfaces gráficas con AWT.
java.awt.datatransfer	Clases para transmisión de datos.
java.awt.event	Clases para manejo de eventos AWT.
java.awt.font	Clases para manejo de fuentes.
java.awt.image	Clases para manejo de imágenes.
java.microedition.xlet	Interfaces del perfil Personal para conexión.
java.beans	Clases que soportan Java Beans.

Tabla 5. Paquetes Java incluidos en el perfil Personal.

La pila de software para la combinación CDC y perfil Personal se muestra en la siguiente figura:

Figura 40 Configuración CDC y el perfil Personal.

2.3 Máquina Virtual de Java en JME

Al igual que la máquina virtual de Java estándar, la especificación JME requiere un intérprete del bytecode de aplicaciones Java para dispositivos móviles, que permita hacer las llamadas al sistema operativo y velar por la seguridad de ejecución en el dispositivo.

La máquina virtual KVM es una compacta y portable máquina virtual de Java destinada a dispositivos con pocos recursos como dispositivos móviles celulares, buscapersonas, organizadores personales, dispositivos móviles para Internet, electrodomésticos, entre otros; y está diseñada y especificada para soportar la implementación del API incluido en la arquitectura JME con la configuración CLDC.

La máquina virtual KVM tiene dos implementaciones de la máquina virtual de Java:

1. Una máquina virtual de Java ajustada para microprocesadores RISC/CISC de 16 ó 32 bits, con memoria disponible menor a 1 MB. Dispositivos típicos en este rango incluyen teléfonos celulares, organizadores personales digitales de rango bajo a medio.
2. Una máquina virtual de Java ajustada para microprocesadores RISC/CISC/DSP de 32 bits, con memoria disponible mayor a 1 MB. Dispositivos típicos en este rango incluyen teléfonos inteligentes, organizadores personales digitales de alto rango.

La máquina virtual de Java KVM, evolucionó de un proyecto de investigación en los laboratorios de Sun Microsystems. Es un derivado de la especificación de la máquina virtual de Java, escrita desde cero en el lenguaje de programación C, con especial énfasis en los siguientes aspectos de diseño:

- Optimizada para bajo consumo de memoria, la huella de memoria del núcleo de KVM es de 40 kilobytes a 80 kilobytes, dependiendo de la plataforma y opciones de compilación.
- Limpia, bien documentada y altamente portable.
- Modular y extensible.
- Tan rápida y completa como sea posible sin sacrificar los objetivos de diseño.

En cuanto a las limitaciones de KVM se pueden mencionar:

- No tiene soporte para tipo de datos como *double* y *float*.
- No soporta la Interfaz Nativa de Java (JNI).
- No se permiten hilos demonios.
- Provee manejo de excepciones limitado, ya que depende de la API del dispositivo, que controla las excepciones.

2.4 Midlet

Las aplicaciones desarrolladas bajo JME son denominadas Midlet. Los MIDlets son aplicaciones creadas usando la especificación MIDP (configuración CLDC con perfil MID).

En los dispositivos con la configuración CLDC no es posible gestionar la ejecución de MIDlets a través de una ventana de comando. En estos dispositivos reside un software que gestiona la ejecución de los MIDlets y los recursos que emplea.

El gestor de aplicaciones o AMS[4] es el software encargado de gestionar los diferentes estados de ejecución de los MIDlets.

El ciclo de vida de un MIDlet comprende cinco estados: Localización, Instalación, Ejecución, Actualización y Borrado, se describen a continuación:

- Localización: Es dónde se selecciona a través del gestor de aplicaciones la aplicación a descargar. Esta etapa es la etapa anterior a la instalación del MIDlet. Por tanto, el gestor de aplicaciones tiene que suministrar los mecanismos necesarios para realizar la elección del MIDlet a descargar. El

[4] AMS acrónimo en inglés del término Application Management System.

AMS es capaz de realizar la descarga de aplicaciones de diferentes maneras, dependiendo de las capacidades del dispositivo.

- Instalación: Todo el proceso es controlado por el gestor de aplicaciones, notificando la evolución de la instalación y de posibles problemas.
- Ejecución: El AMS gestiona los estados de los MIDlets en función de los eventos que ocurren durante su ejecución.
- Actualización: El AMS gestiona la actualización de los MIDlets y notifica si hay una aplicación presente en el dispositivo y que requiera una actualización.
- Borrado: El AMS es el encargado de eliminar el MIDlet del dispositivo.

La figura 41 muestra los estados del ciclo de vida de un MIDlet:

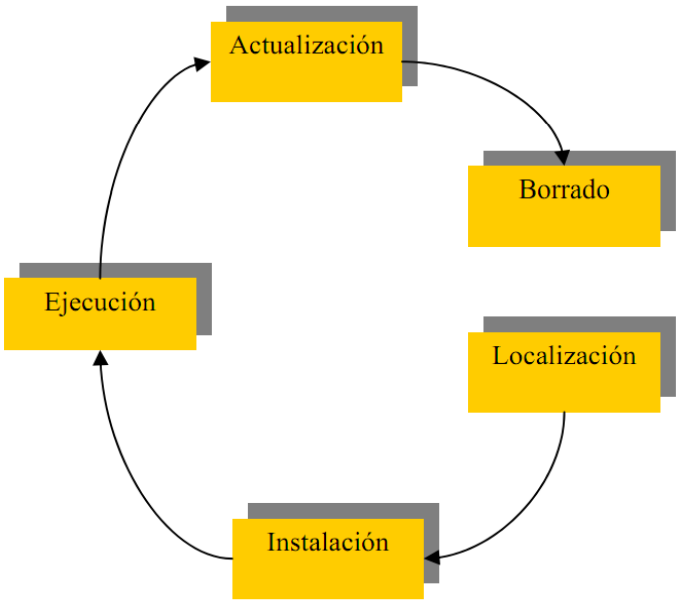


Figura 41 Ciclo de vida del MIDlet.

2.4.1 Ejecución de un MIDlet

Durante la ejecución de un MIDlet, éste puede pasar por los siguientes estados: Pausa, Activo o Destruido. Un MIDlet sólo puede estar en un estado en un momento dado.

A diferencia de un programa Java convencional, los Midlets no poseen un método main(). Al inicializar un Midlet, éste se encuentra en estado de Pausa hasta que se llama al método startApp() que inicia su ejecución. El método pauseApp() detiene la ejecución y método destroyApp() destruye el Midlet.

Estos métodos pueden ser invocados por el AMS o por el mismo MIDlet. La figura 42 muestra las posibles transiciones de estados:

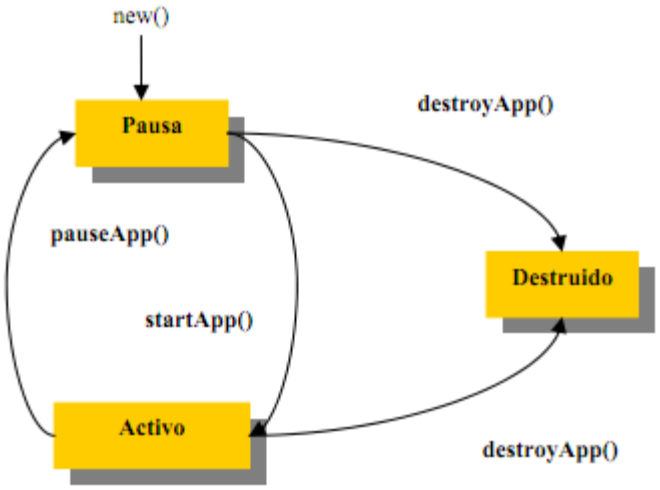


Figura 422 Estados de ejecución del MIDlet.

Capítulo 3 Análisis y Diseño de la solución

La realización del análisis de la solución está basada en las especificaciones provistas por el proceso unificado de desarrollo UP[5] (Unified Process), debido a que el mismo provee un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software y diferentes áreas de aplicación. UP propone actividades y artefactos para las diferentes etapas del proceso de análisis y diseño.

En el marco de trabajo genérico del proceso unificado, se desarrollaron las actividades de levantamiento de requerimientos y diseño de la solución. Para describir las funcionalidades de la solución como resultado de las actividades de análisis se emplearon los casos de uso y para describir el diseño de las interacciones de la solución se emplearon diagramas de secuencia. El diagrama de casos de uso y diagrama de secuencia están basados[6] en la especificación UML[7].

La sección 3.1 describe los actores relacionados con el sistema, además de su rol asociado. La sección 3.2 contiene la descripción de los casos de uso y la sección 3.3 contiene los diagramas de secuencia con las interacciones móviles y web.

3.1 Actores

En esta sección se describen los actores relacionados con el sistema.

ACTOR	ACT. #1 Creador de Despacho
DESCRIPCIÓN	Este actor es el responsable de crear el Despacho empleando una aplicación móvil.
RESPONSABILIDADES	• Crear el despacho y generar el código QR asociado al mismo.

ACTOR	ACT. #2 Despachador
DESCRIPCIÓN	Este actor es el responsable de validar el Despacho empleando una aplicación móvil.

[5] UP: http://www.methodsandtools.com/archive/archive.php?id=32

[6] UML y UP: http://www.jeckle.de/files/uniproc.pdf

[7] UML: acrónimo en inglés de Unified Modelling Language, lenguaje de modelado unificado.

RESPONSABILIDADES	• Realizar la validación del Despacho antes de ser despachado, y decidir enviarlo o cancelarlo, cambiando el estatus del despacho.

ACTOR	ACT. #3 Receptor de Despacho
DESCRIPCIÓN	Este actor es el responsable de recibir el Despacho empleando una aplicación móvil.
RESPONSABILIDADES	• Realizar la validación del Despacho, y decidir recibirlo o rechazarlo, cambiando el estatus del despacho.

ACTOR	ACT. #4 Administrador
DESCRIPCIÓN	Este actor es el responsable de consultar los Despachos, a través de una aplicación Web.
RESPONSABILIDADES	• Consultar el estado de los despachos, realizando filtros de los estatus existentes.

3.2 Casos de Uso

En esta sección se presentan los diagramas y descripciones de los casos de uso del sistema.

3.2.1 Nivel 1

El diagrama del nivel 1 presenta una visión funcional de alto nivel del sistema, con los actores que interactúan con el sistema y los casos de uso generales dentro del sistema, como se muestra en la figura 43.

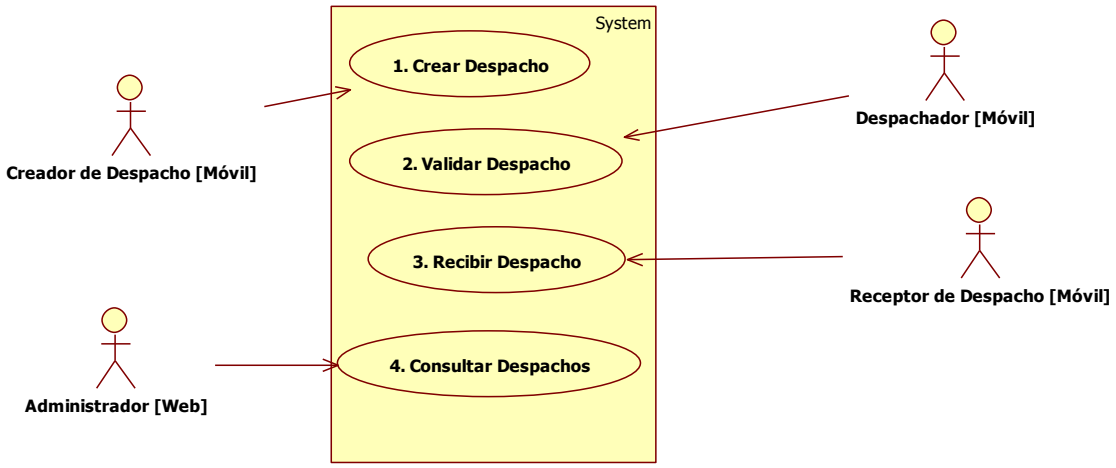


Figura 43 Visión general del sistema.

3.2.2 Nivel 2

En el nivel 2 se presenta los diagramas refinados de los casos de uso generales y la descripción de los casos de uso.

3.2.2.1 Casos de uso Móvil

En esta sección se describen los casos de uso implementados en la aplicación móvil.

La figura 44 muestra el caso de uso móvil Crear Despacho.

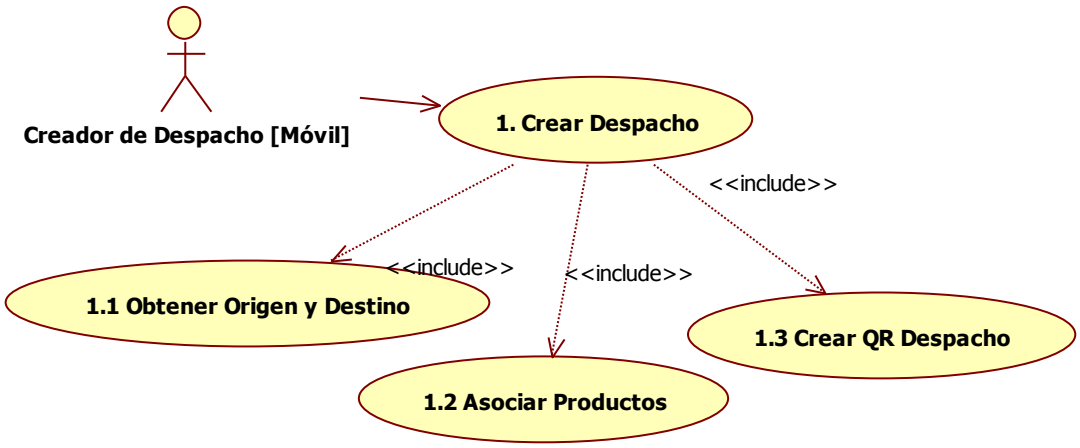


Figura 44 Caso de uso Crear Despacho.

CASO DE USO MÓVIL	CU #1. Crear Despacho
ACTOR	Creador de Despacho
DESCRIPCIÓN	El actor ingresa a la aplicación móvil instalada en el teléfono y crea el despacho.

FLUJO BÁSICO	1. El caso de uso inicia cuando el actor creador de despacho, ingresa a la aplicación móvil instalada en el dispositivo móvil, llena los campos asociados al despacho, asocia productos y solicita la creación del Despacho. 2. El servidor recibe la solicitud de creación de despacho y realiza la transacción. 3. El servidor envía un correo electrónico al actor creador de despacho, con un URL donde podrá descargar el código QR que contiene la información asociada al despacho. Este QR enviado será adjuntado al documento del despacho, el cual será recibido por el despachador y el receptor del despacho. 4. El servidor envía una respuesta positiva a la aplicación móvil, confirmando la creación del despacho y el número asociado al mismo.
FLUJO ALTERNO 1	1. En caso que se intente crear un despacho sin productos, el usuario recibe un mensaje de error que indica que debe tener al menos un producto asociado.
FLUJO ALTERNO 2	1. En caso de no haber una conexión exitosa al servidor al momento de crear el despacho, se generará el siguiente mensaje de error en el dispositivo: “Error en la comunicación con el host remoto”.
PRE CONDICIONES	El actor debe ser un usuario autorizado.
POST CONDICIONES	Se crea en el servidor el despacho con estatus *Creado*, se genera el código QR asociado al mismo, el cual es enviado por correo electrónico al actor.
NOTAS	El despacho al menos debe tener un producto asociado.

La figura 45 muestra el caso de uso móvil Validar Despacho.

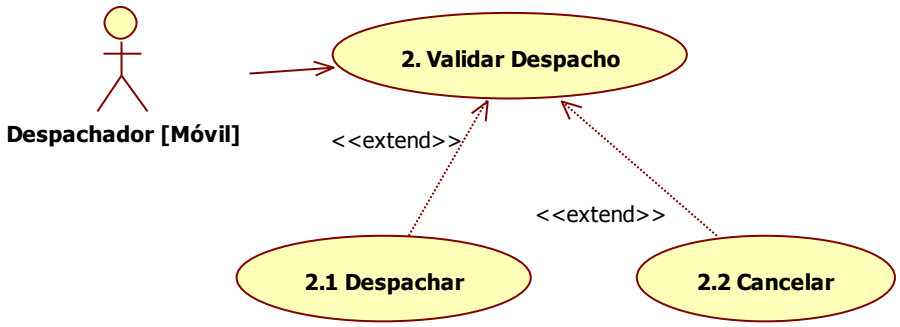


Figura 45 Caso de uso Validar Despacho.

CASO DE USO MÓVIL	CU #2 Validar Despacho
ACTOR	Despachador
DESCRIPCIÓN	El actor revisa y valida el Despacho, cambia el estatus del mismo a *Despachado* o Cancela el Despacho.
FLUJO BÁSICO	1. El caso de uso inicia cuando el actor Despachador ejecuta la aplicación móvil. 2. Escanea el código QR, automáticamente se genera una petición al servidor recuperando los datos asociados al Despacho. 2. El actor Despachador, revisa y valida el despacho, si es correcto selecciona la opción *Despachar*, lo cual cambia el estatus del despacho a *Despachado* en el servidor, y establece la fecha y hora de la validación del despacho y el usuario que realizó el cambio de estatus. 3. El actor Despachador, recibe un mensaje en la aplicación móvil indicando que el estatus ha sido modificado a *Despachado*.

FLUJO ALTERNO 1	1. En caso que el actor Despachador decida que el despacho escaneado no es válido, seleccionará la opción *Cancelar*, lo cual ocasiona que se cancele el despacho, cambiando su estatus a *Cancelado* en el servidor. En este caso, es obligatorio llenar el campo Observaciones, donde se describen las razones de la cancelación del despacho. En caso de no llenar este campo, recibirá un alerta indicando que debe hacerlo. 2. Para cancelar el despacho, el actor presiona el botón (o selección) Cancelar y recibe un mensaje de confirmación de la transacción. En caso de ser positiva se genera una petición al servidor, donde se cambia el estatus del despacho a C*ancelado,* se establece la fecha y hora de cancelación de la validación del despacho y el usuario que realizó el cambio de estatus. 3. El actor recibe en el móvil una notificación que el estatus del despacho ha sido cambiado a *Cancelado*.
FLUJO ALTERNO 2	1. En caso de escanear un despacho con estatus *Cancelado*, es posible consultar las observaciones ingresadas al momento de cancelarlo y consultar los productos asociados al despacho.
FLUJO ALTERNO 3	1. En caso de realizar un cambio de estatus y no hay una conexión exitosa al servidor, se generará el siguiente mensaje de error en el dispositivo: “Error en la comunicación con el host remoto”.
PRE CONDICIONES	El actor debe ser un usuario autorizado
POST CONDICIONES	Se cambia el status del despacho a *Despachado*.
NOTAS	

La figura muestra el caso de uso móvil Recibir Despacho.

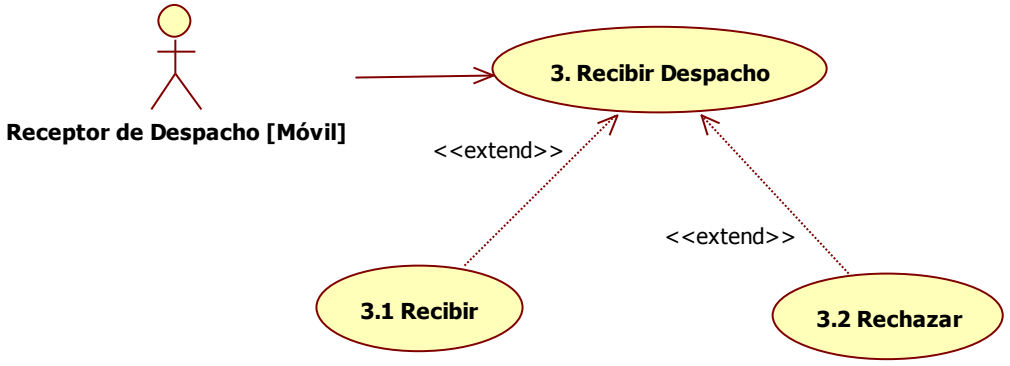


Figura 46 Caso de uso Recibir Despacho

CASO DE USO MÓVIL	CU #3 Recibir Despacho
ACTOR	Receptor de Despacho
DESCRIPCIÓN	En este caso de uso el actor Receptor de Despacho consulta la información asociada al despacho y decide si recibe el despacho, cambiando el status del despacho a *Recibido* o en caso de rechazarlo el estatus es modificado a *Rechazado*.
FLUJO BÁSICO	1. El caso de uso inicia cuando el actor Receptor de Despacho ejecuta la aplicación móvil, escanea el código QR del despacho recibido, generando automáticamente una petición al servidor recuperando los datos asociados al Despacho. 2. El actor Receptor de Despacho, revisa y valida el despacho recibido, si es correcto selecciona la opción *Recibir*, lo cual cambia el estatus del despacho a *Recibido* en el servidor, le establece la fecha y hora de la recepción del despacho y el usuario que realizó el cambio de estatus. 3. El actor Receptor de Despacho, recibe un mensaje en la aplicación móvil indicando que el estatus ha sido modificado a *Recibido*. 4. En caso de escanear un despacho con estatus *Recibido*, es posible consultar las observaciones ingresadas al momento de recibirlo y consultar los productos asociados al despacho.

FLUJO ALTERNO 1	1. En caso que el actor Receptor de Despacho decida que el despacho escaneado no es válido, seleccionará la opción *Rechazar*, cambiando su estatus a *Rechazado* en el servidor. Es obligatorio llenar el campo observaciones, donde se describen las razones del rechazo del despacho. En caso de no llenar este campo, recibirá un alerta indicando que debe hacerlo. 2. Para rechazar el despacho, el actor presiona el botón (o selección) *Rechazar* y recibe un mensaje de confirmación de la transacción. En caso de ser positiva se genera una petición al servidor, donde se cambia el estatus del despacho a *Rechazado,* se establece la fecha y hora del rechazo del despacho y el usuario que realizó el cambio de estatus. 3. El actor recibe en el móvil una notificación que el estatus del despacho ha sido cambiado a *Rechazado*.
FLUJO ALTERNO 2	1. En caso de escanear un despacho con estatus *Rechazado*, es posible consultar las observaciones ingresadas al momento de rechazarlo y consultar los productos asociados al despacho.
FLUJO ALTERNO 3	1._En caso de realizar un cambio de estatus y no hay una conexión exitosa al servidor, se generará el siguiente mensaje de error en el dispositivo: “Error en la comunicación con el host remoto”.
PRE CONDICIONES	El actor debe ser un usuario autorizado
POST CONDICIONES	Se cambia el status del despacho a “Recibido”.
NOTAS	

3.2.2.2 Casos de uso Web

En esta sección se describen los casos de uso implementados en la aplicación web.

La figura 47 describe el caso de uso Consultar Despachos.

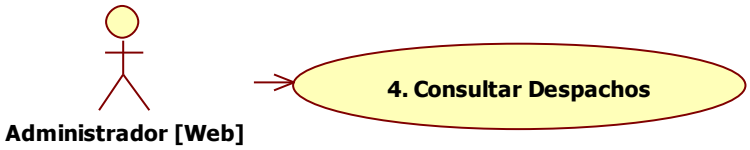


Figura 47 Caso de uso Consultar Despachos.

CASO DE USO WEB	CU #4 Consultar Despachos
ACTOR	Administrador
DESCRIPCIÓN	En este caso de uso el actor Administrador puede consultar los despachos realizados.
FLUJO BÁSICO	1. El caso de uso inicia cuando el administrador ingresa a la aplicación web de administración de despachos, se autentica y consulta los diferentes despachos realizados. 2. El actor puede consultar los despachos, filtrando por estatus (Creado, Recibido, Cancelado o Rechazado). 3. Una vez selecciona un despacho, el usuario puede consultar los detalles del despacho en ese momento: origen, destino, estatus, observaciones, productos asociados y el historial del despacho.
FLUJO ALTERNO	1. El administrador ingresa un usuario y/o password inválido, lo cual no le permite ingresar a la aplicación.
PRE CONDICIONES	El actor debe ser un usuario autorizado
POST CONDICIONES	

3.3 Interacciones Móvil Web

En esta sección se describe el diseño de las interacciones de las aplicaciones móviles con el módulo web para llevar a cabo los casos de uso.

La figura 48 muestra el diagrama de secuencia de la interacción de la aplicación móvil de Creación de Despachos con el módulo web.

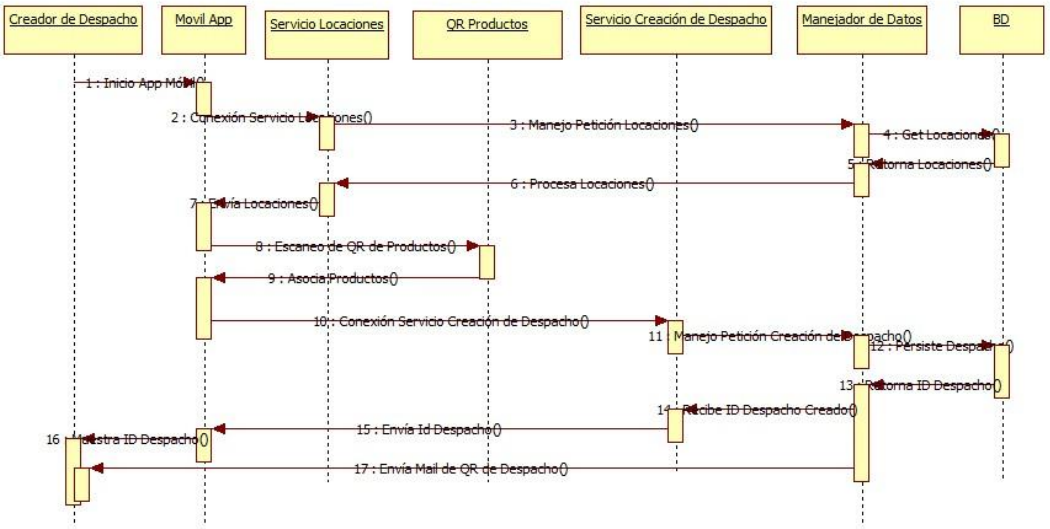


Figura 48 Diagrama de secuencia de la funcionalidad móvil Creación de Despachos.

La figura 49 contiene el diagrama de secuencia de la interacción de la aplicación móvil de Manejo de Despachos con el módulo web.

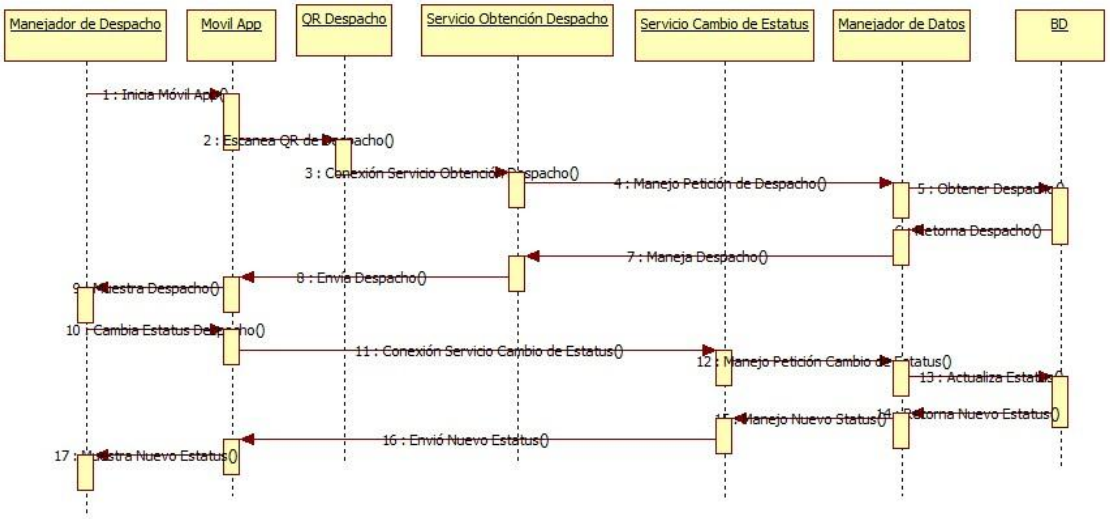


Figura 49 Diagrama de secuencia de la funcionalidad móvil de Manejo de Despachos.

La figura 50 muestra el diagrama de secuencia de la aplicación web de Consulta de Despachos.

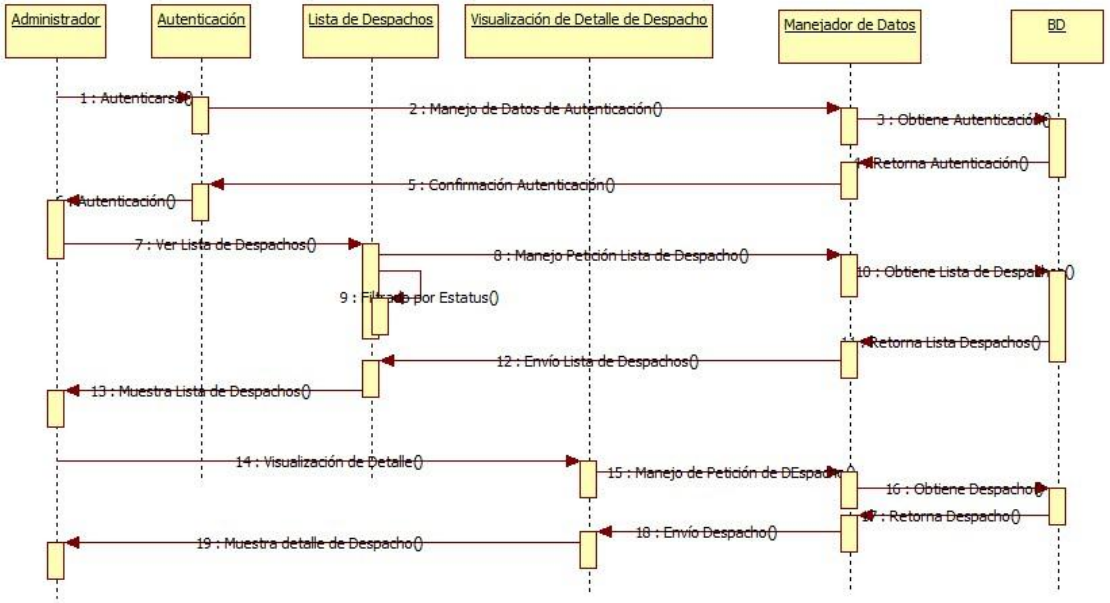


Figura 50 Diagrama de secuencia de la funcionalidad web Consulta de Despachos.

Capítulo 4 Implementación

En este capítulo se describirá la implementación de la solución, así como las tecnologías usadas para cumplir exitosamente los objetivos de dicha implementación.

En la sección 4.1 se describe la parte web de la solución, donde se describe el framework Apache Wicket empleando para desarrollar la parte web, la especificación JSON empleada para la transferencia de datos, así como el funcionamiento de la aplicación web de la solución. La sección 4.2 se describe el funcionamiento de la solución móvil implementada para el manejo de despachos.

4.1 Implementación Web

En esta sección se describen dos tecnologías empleadas para el desarrollo de la aplicación web que utiliza el actor Administrador para la consulta de despachos, así como las funcionalidades de la aplicación web.

4.1.1 Apache Wicket

Es un framework de desarrollo de aplicaciones web para la plataforma Java EE. Fue originalmente escrito por Jonathan Locke en 2004. Wicket es un proyecto de código abierto, disponible bajo los términos de la licencia Apache, versión 2.0.

Wicket es un framework basado en componentes, lo que lo pone en contraste con algunas de las soluciones anteriores en la tarea de programación web. Al igual que otros frameworks, Wicket se basa en la API de Java Servlets de Sun, sin embargo, a diferencia de los frameworks como Struts o Spring MVC, el desarrollador utiliza Wicket en su mayoría retirado del paradigma de petición/respuesta que es inherente al Web y los componentes web tipo Servlets.

Con el *framework* Wicket, la programación se basa en sólo Java y HTML utilizando sólo abstracciones significativas.

El *framework* Wicket no añade ninguna sintaxis especial a sus archivos html (como las tecnologías JSF[8]. Los elementos HTML y las clases Java son relacionados mediante Wicket IDs, que son atributos en las etiquetas HTML y las propiedades que se asignan a los componentes Java y permiten enlazar unívocamente entre el elemento HTML y el componente Java asociado. Al usar el *framework* Wicket, los programadores y los diseñadores de la interfaz web pueden trabajar de forma independiente y paralela.

[8] JSF: acrónimo en inglés de Java Server Faces.

Programación Java

Wicket permite programar sus componentes y las páginas usando constructores regulares Java.

Se crean componentes con la palabra clave *new*, se crean jerarquías mediante la adición de componentes hijos a los padres, y el uso de la palabra clave *extends* para heredar la funcionalidad de otros componentes.

El *framework* Wicket procura sacar el máximo provecho de las fortalezas del lenguaje y los numerosos entornos de desarrollo disponibles para ello.

El hecho que se pueda decidir cómo se crean los componentes le da un nivel importante de flexibilidad. Por ejemplo, se pueden codificar las páginas y los componentes de tal manera que requieren ciertos parámetros, a continuación un ejemplo:

```
public class EditPersonLink extends Link {
	private final Person person;
	public EditPersonLink(String id, Person person) {
		super(id);
		this.person = person;
	}
	public void onClick() {
		setResponsePage(new EditPersonPage(person));
	}

}
```

Este fragmento de código define un componente *EditPersonLink* el cual extiende del componente Wicket *Link* que obliga a sus usuarios a crear una instancia de *Person* y se la pasa al constructor de *EditPersonLink*.

El método *onClick*, el cual será llamado en una solicitud diferente al constructor del enlace, utiliza el mismo objeto *Person* provisto en el constructor del link y redirecciona a otra página pasando como parámetro el objeto *Person* definido. Wicket hace que este tipo de programación sea lo más sencilla posible y el estado entre las peticiones es manejado por el *framework*. Simplemente funciona.

A continuación describiremos cómo el *framework* Wicket utiliza HTML simple y llano para mantener el código de presentación.

Programación HTML

Usando Wicket, la parte de presentación de la aplicación web se define en plantillas HTML. Esta es una característica que diferencia a Wicket de la mayoría de los *frameworks*: obliga a sus usuarios a utilizar las plantillas limpias. Wicket exige el requisito que las plantillas HTML usadas, contienen únicamente lenguaje de marcado y algunos marcadores donde se conectan los componentes Wicket. No permite introducir lógica en la presentación de las plantillas HTML.

A diferencia del siguiente fragmento de código en un componente JSP:

```
<table>
	<tr>
	<c:forEach var="item" items="${sessionScope.list}">
		<td>
			<c:out value="item.name" />
		</td>
	</c:forEach>
	</tr>
</table>
```

A diferencia del siguiente fragmento usando Apache Velocity:

```
<table>
	<tr>
	#foreach ($item in $sessionScope.list)
		<td>
			${item.name}
		</td>
	#end
	</tr>
</table>
```

Usando Wicket, el código sería el siguiente:

```
<table>
	<tr>
		<td wicket:id="list">
			<span wicket:id="name" />
		</td>
	</tr>
</table>
```

Usando JSP, es necesario asegurarse que el contexto (páginas, request y atributos de sesión) se establezcan con los objetos que se necesita en la página. Se pueden añadir iteraciones, condicionales, y así sucesivamente al JSP.

Usando Wicket se debe conocer la estructura de la página de antemano. En el ejemplo de html anterior, un componente *list view* habría que ser añadido a la página con un *id: list*; y para cada fila del *listview*, debe tener un componente hijo con un *id: name*.

La forma en que se codifica JSP al parecer es más fácil, entonces habría que preguntarse por qué elegir Wicket donde la separación entre la presentación y la lógica es rígida. En gran medida debido a los problemas que conlleva mezclar la lógica y la presentación en las plantillas, entre los cuales se pueden mencionar:

- Codificar las plantillas puede resultar en código espagueti (estructura de control de flujo compleja e incomprensible). El código se puede tornar muy detallado, y puede ser difícil distinguir entre las partes de la lógica y los trozos de HTML normal, la plantilla completa puede llegar a ser difícil de leer, y por lo tanto difícil de mantener.
- Si se codifica lógica en scripts, el compilador no ayudará con la refactorización, ni ayudará a evitar errores de sintaxis.

- Es más difícil trabajar con diseñadores. Si se trabaja con diseñadores de páginas web por separado, les será difícil descifrar plantillas del tipo JSP. En lugar de mantenerse enfocado en su trabajo (la presentación y apariencia de las aplicaciones) tienen que comprender al menos los conceptos básicos del lenguaje de script de la plantilla.

Abstracciones Significativas

El *framework* Wicket permite escribir interfaces de usuario que se ejecutan en navegadores web. Por tanto, tiene abstracciones para todos los widgets que se pueden ver en una página web típica, como enlaces, listas desplegables, campos de texto y botones. El *framework* también proporciona abstracciones que no son directamente visibles, pero que tienen sentido en el contexto de aplicaciones web: aplicaciones, sesiones, páginas, validadores, convertidores, etc.

El *framework* Wicket apunta a reducir la brecha entre la programación orientada a objetos y el hecho que el web se basa en el protocolo http, que no posee estado. El *framework* Wicket administra el estado de forma transparente para que se pueda utilizar programación Java regular para la aplicación de la lógica entre las páginas y sus componentes.

4.1.2 JSON

JSON, es el acrónimo de *JavaScript Object Notation* (Notación de Objetos basada en JavaScript), una especificación de un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del lenguaje de programación JavaScript. JSON especifica un formato de texto que es completamente independiente del lenguaje, y utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes, esto es conocido como un *objeto*, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, un *objeto* es un conjunto desordenado de pares nombre/valor. Un objeto comienza con { (llave de apertura) y termine con } (llave de cierre). Cada nombre es seguido por : (dos puntos) y los pares nombre/valor están separados por , (coma). La figura muestra el formato básico de un objeto JSON.

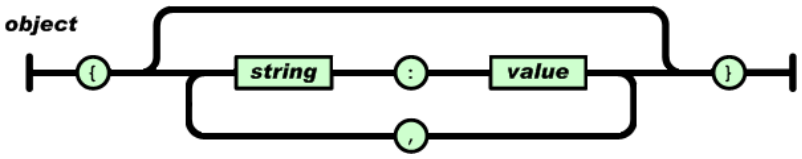


Figura 51 Formato de un objeto JSON.

Ejemplo de un objeto JSON:

{"balance":1000.21,"num":100,"nickname":null,"is_vip":true,"name":"foo"}

Un *arreglo* es una colección de valores. Un arreglo comienza con [(corchete izquierdo) y termina con] (corchete derecho). Los valores se separan por , (coma). La figura muestra el formato básico de un array JSON.

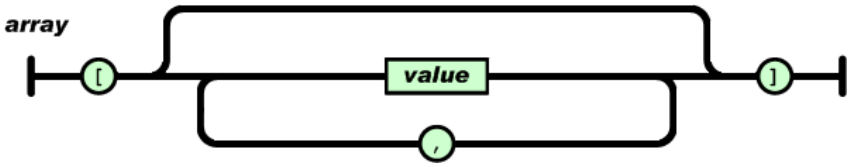


Figura 52 Formato de un array JSON.

Ejemplo de un arreglo JSON: Result: ["foo",100,1000.21,true,null]

Ejemplo de una combinación entre un objeto JSON y un arreglo JSON:

{"balance":1000.21,"list2":true,null,"num":100,"list1":["foo",100,1000.21],"nickna
me":null,"is_vip":true,"name":"foo"}

En este ejemplo, se crea un objeto que contiene un único miembro "bindings", que contiene un arreglo que a su vez contiene tres objetos, cada objeto tiene un "ircEvent", un "method" y un "regex".

```
{"bindings": [
        {"ircEvent": "PRIVMSG", "method": "newURI", "regex": "^http://.*"},
        {"ircEvent": "PRIVMSG", "method": "deleteURI", "regex": "^delete.*"},
        {"ircEvent": "PRIVMSG", "method": "randomURI", "regex": "^random.*"}
    ]
};
```

Un *valor* puede ser una *cadena de caracteres* con comillas dobles, o un *número*, o `true` o `false` o `null`, o un *objeto* o un *arreglo*. Estas estructuras pueden anidarse. La figura muestra los valores permitidos en un objeto JSON.

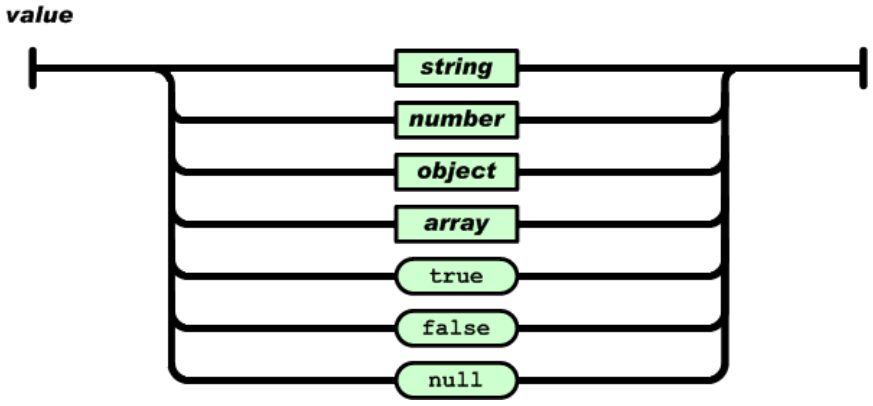


Figura 53 Valores permitidos en un objeto JSON.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en transferencias de datos, como es el caso con AJAX.

Comparación entre JSON y XML

Sencillez: JSON es más simple que XML. JSON tiene una gramática más pequeña y más directamente sobre los mapas de estructuras de datos utilizadas en lenguajes de programación modernos.

Extensibilidad: JSON no es extensible porque no es necesario. JSON no es un lenguaje de marcado, por lo que no es necesario definir nuevas etiquetas o atributos para representar los datos en ella.

Interoperabilidad: JSON tiene el mismo potencial de interoperabilidad que XML. XML es orientado a documento. JSON es orientado a datos.

Abierto: JSON es tan abierto como XML, quizás más porque no tiene problemas de estandarización.

Legibilidad: JSON es mucho más fácil de leer y escribir para los humanos que XML.

Intercambio de datos: Al igual que XML, se puede utilizar como un formato de intercambio para permitir a los usuarios mover los datos entre aplicaciones similares.

Análisis sintáctico: JSON, al ser una notación más simple, no requiere un parser o software especializado. En lenguajes como JavaScript y Python, la notación JSON se construye en el lenguaje de programación, sin necesidad de software adicional en absoluto. En otros lenguajes, sólo una pequeña cantidad de código JSON específico es necesario.

A continuación un ejemplo comparativo entre JSON y XML.

JSON:

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

XML:

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

4.1.3 Aplicación Web de Consulta de Despachos

En esta sección se describirá la aplicación web de Consulta de Despachos, utilizada por el actor Administrador.

La figura 54 muestra la pantalla inicial de la aplicación web de consulta de despachos. Se puede visualizar en la parte superior la fecha y hora en curso y un formulario básico de inicio de sesión.

Figura 54 Vista de inicio de sesión.

En caso de dejar en blanco el campo *login* del formulario, se mostrará un mensaje de error que indica que debe colocar el login, como se muestra en la figura 55.

Figura 55 Vista en caso de no colocar el login.

En caso de dejar en blanco el campo *password* del formulario, se mostrará un mensaje de error que indica que debe colocar el password, como se muestra en la figura 56.

Figura 56 Vista en caso de no colocar el password.

En caso de colocar el *Login* y/o el *Password* inválido, se mostrará un mensaje de error que indica que debe colocar el login y/o el password correctos, como se muestra en la figura 57.

Figura 57 En caso de colocar el login y/o password inválido.

Si el login y el password son correctos, ingresará a la aplicación de Consulta de Despachos, donde podrá visualizar en principio todos los despachos que se han creado hasta el momento, el estatus en el que se encuentran actualmente, su fecha y hora de creación y un enlace *Ver* (que será explicado más adelante) que permitirá visualizar el detalle del un despacho en particular. La figura 58 muestra la lista de todos los despachos creados.

Figura 58 Lista de todos los despachos creados con estatus y fecha de creación.

Sobre la lista de los despachos se puede visualizar una lista desplegable con los diferentes estatus en el que puede estar un despacho, al seleccionar un estatus en específico, se mostrara en la lista de despachos, sólo los que se encuentran en el

estatus seleccionado. La figura 59 muestra la lista desplegable con todos los estatus seleccionables.

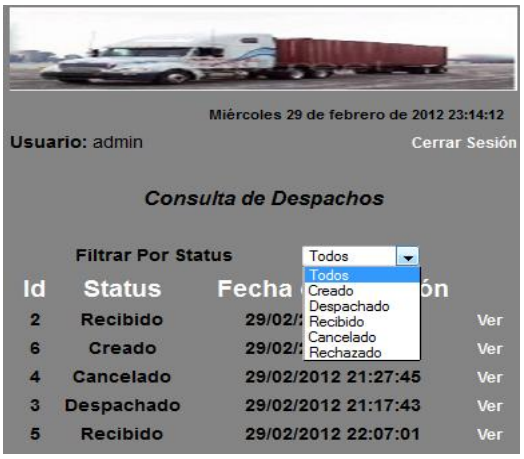


Figura 59 Lista desplegable de estatus a seleccionar.

Si se selecciona el estatus *Creado* se mostraran sólo los despachos en ese estatus, como se muestra en la figura 60.

Figura 60 Despachos en estatus creado.

Si se selecciona el estatus *Despachado* se desplegarán únicamente los despachos en ese estatus, como se muestra en la figura 61.

Figura 61 Despachos en estatus despachado.

Si se selecciona el estatus *Recibido* se mostraran sólo los despachos en ese estatus, como se muestra en la figura 62.

Figura 62 Despachos en estatus recibido.

Si se selecciona el estatus *Cancelado* se mostraran sólo los despachos en ese estatus, como se muestra en la figura 63.

Figura 63 Despachos en estatus cancelado.

Si se selecciona el estatus *Rechazado* se mostraran sólo los despachos en ese estatus, como se muestra en la figura 64.

Figura 64 Despachos en estatus rechazado.

Si se selecciona nuevamente la opción *Todos,* se desplegarán todos los despachos existentes, como se muestra en la figura 65.

Figura 65 Lista con todos los despachos.

La opción *Ver* que tienen todos los despachos es un enlace que abrirá una ventana modal con el detalle del despacho. Allí se podrá visualizar los datos básicos del despacho como el id, el origen, el destino, el estatus actual, los productos asociados (con código, nombre y cantidad) y el historial de estatus (estatus, ejecutor de la transacción y fecha de realización). En caso que el estatus del despacho sea cancelado o rechazado, también se podrá visualizar la observación del despacho. La figura 66 muestra la ventana modal de consulta de detalle de despachos.

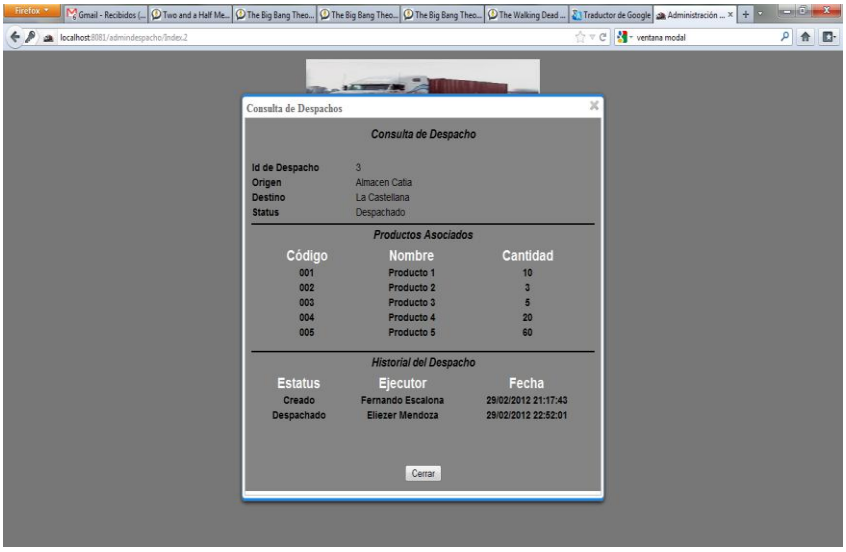


Figura 66 Ventana modal con el detalle de despacho.

En la figura 67 se muestra el detalle de un despacho en estatus *Creado.*

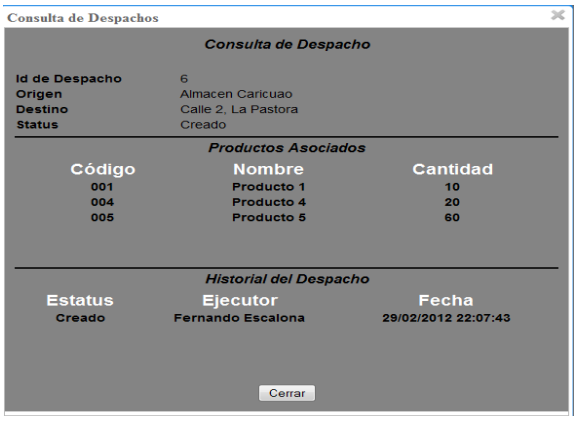


Figura 67 Detalle de despacho en estatus creado.

En la figura 68 se muestra el detalle de un despacho en estatus *Despachado.*

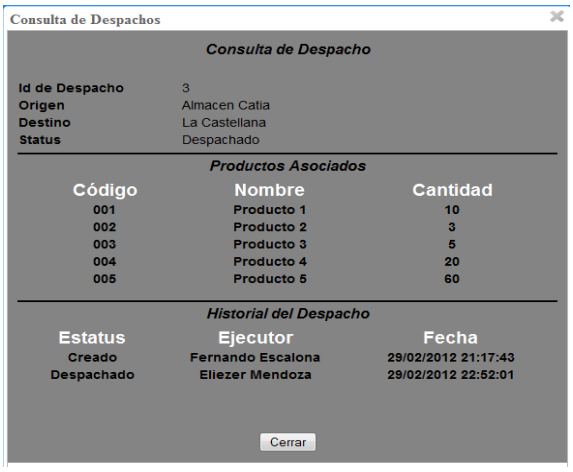


Figura 68 Detalle de despacho en estatus despachado.

En la figura 69 se muestra el detalle de un despacho en estatus *Recibido.*

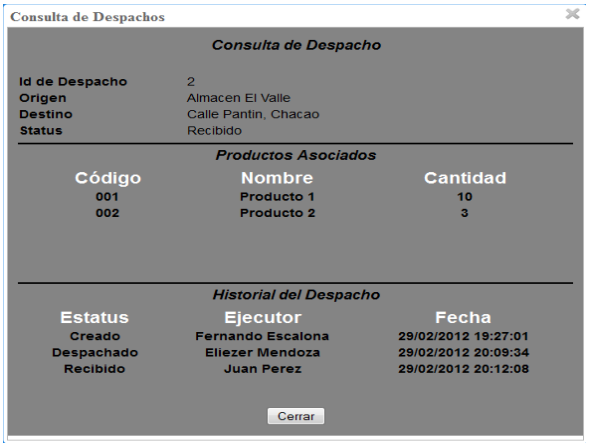


Figura 69 Detalle de despacho en estatus recibido.

En la figura 70 se muestra el detalle de un despacho en estatus *Cancelado.*

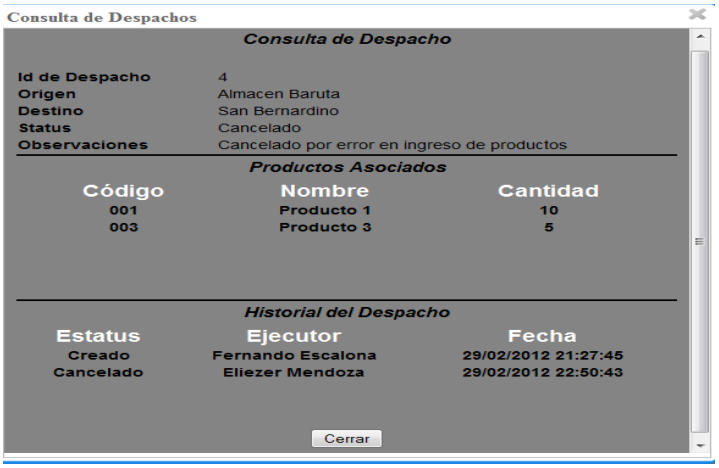


Figura 70 Detalle de despacho en estatus cancelado

En la figura 71 se muestra el detalle de un despacho en estatus *Rechazado.*

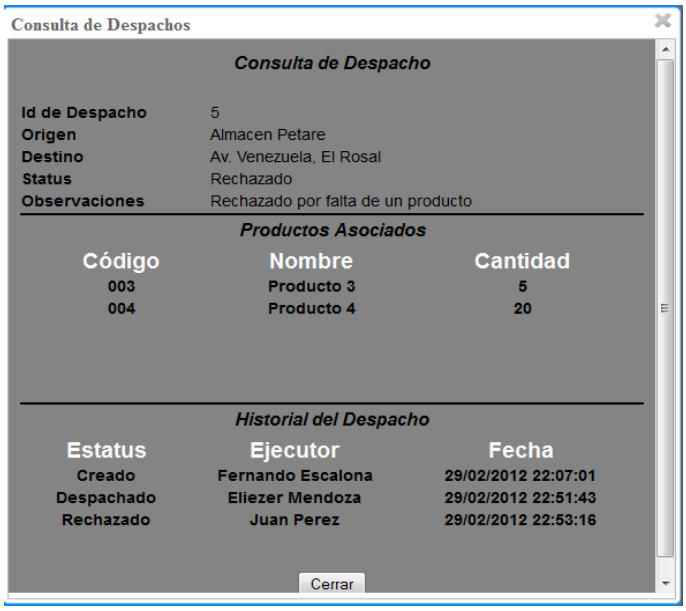


Figura 71 Detalle de despacho en estatus rechazado.

4.2 Implementación Móvil

Se describen las aplicaciones móviles desarrolladas para la creación y el manejo de despachos y los procesos involucrados en el flujo de estas operaciones. Las aplicaciones descritas son utilizadas por los actores Creador del Despacho, Despachador y Receptor del Despacho.

4.2.1 Aplicación móvil Creación de Despachos

A continuación se describe el flujo de la aplicación móvil de creación de despachos.

La figura 72 muestra el ícono de la aplicación *Crear Despacho* instalada en el dispositivo móvil.

Figura 72 Aplicación de Creación de Despachos instalada en el móvil.

Al ejecutar la aplicación *Crear Despacho*, se invoca al servicio web que permite obtener las locaciones tanto de origen, como de destino del despacho y son cargadas en la aplicación. La pantalla principal posee un botón para escanear el código QR de los productos y así asociarlos al despacho. La figura 73 muestra la pantalla principal de Creación del Despacho, con las listas desplegables de origen y destino, el botón de escaneo de productos, la lista de productos asociados y los ítems de menú para enviar o cancelar el despacho.

Figura 73 Pantalla principal de la aplicación móvil Crear Despacho.

La figura 74 muestra la acción que se ejecuta al pulsar el botón *Escanear Producto,* se abrirá el lector de códigos QR donde se leerá el QR asociado al producto y así asociarlo al Despacho.

Figura 74 Interfaz de lectura de códigos QR en la aplicación móvil.

Una vez leído el QR del producto, el mismo se asocia al Despacho y se inserta en la lista de productos asociados. En caso de querer asociar otro producto, se repite

la operación de Escanear Producto, esta operación puede repetirse cuantas veces se desee. La figura 75 muestra el estatus de la pantalla de Creación de Despachos al tener asociado un producto al despacho.

Figura 75 Estatus de la interfaz al asociar un producto.

En caso que se desee cancelar la creación del Despacho, se selecciona la opción del menú *Cancelar*, aparecerá un mensaje de confirmación para asegurar que realmente desea cancelar la creación del despacho. En caso de pulsar afirmativamente, se desasociarán todos los productos agregados y se reasignarán los valores por defecto de las listas de Origen y Destino. La figura 76 muestra el mensaje de confirmación que se despliega al momento de seleccionar la opción de cancelación de creación del despacho.

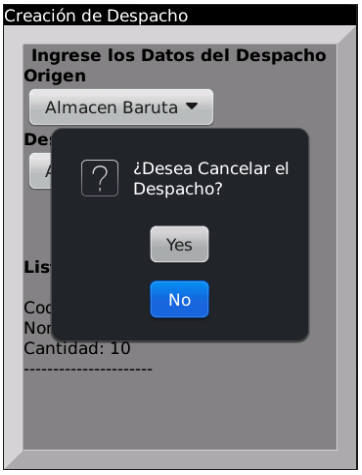


Figura 76 Mensaje de confirmación de cancelación de creación de despacho.

En caso que se desee realizar la creación del despacho, se selecciona la opción del menú *Enviar*, y se despliega un mensaje de confirmación de creación del despacho. En caso de pulsar afirmativamente, se invocará el servicio web de creación de despacho, al cual se le enviarán los datos de Origen, Destino y productos asociados. La figura 77 muestra el mensaje de confirmación que se visualiza al momento de seleccionar la opción de envío de creación del despacho.

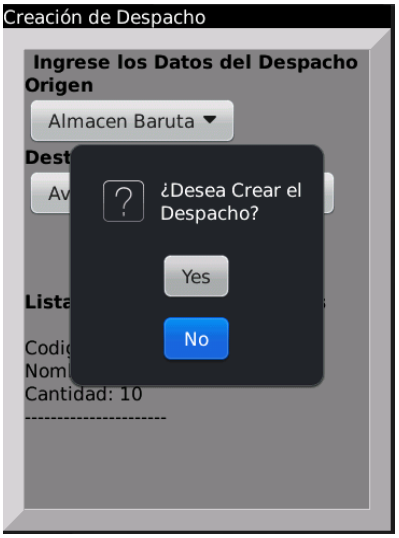


Figura 77 Mensaje de confirmación de envío de creación del despacho

El servicio en el *backend* recibirá los datos asociados a la creación del despacho, procesará la data enviada, la persistirá y generará un número de despacho, el cual se enviará de respuesta al móvil, indicando la creación correcta y exitosa del despacho.

En caso de crearse correctamente el despacho, se desplegará otra vista donde se indicará la correcta creación del despacho, el número del despacho generado y el código QR asociado al despacho, el cual contendrá el número de despacho generado. Además, el código QR será enviado por correo electrónico al correo asociado al creador del despacho.

Es importante tomar en cuenta que la imagen del código QR generado en la pantalla de la aplicación y el enviado por correo electrónico podrían no ser iguales al visualizarlos, sin embargo contienen la misma información.

La figura 78 muestra la vista al momento de recibirse la respuesta exitosa del servidor de la creación del despacho, se muestra el número de despacho generado y el QR asociado al mismo.

Figura 78 Pantalla de confirmación de creación del despacho.

Una vez creado el despacho, puede retornarse a la vista de creación de despachos para crear otro despacho si se desea. La figura 79 muestra nuevamente la pantalla de creación de despachos.

Figura 79 Pantalla de creación de despachos.

En caso de querer salir de la aplicación, se desplegará un mensaje de confirmación. Se debe pulsar afirmativo si desea salir de la aplicación. La figura 80 muestra el mensaje de confirmación de cierre de la aplicación móvil.

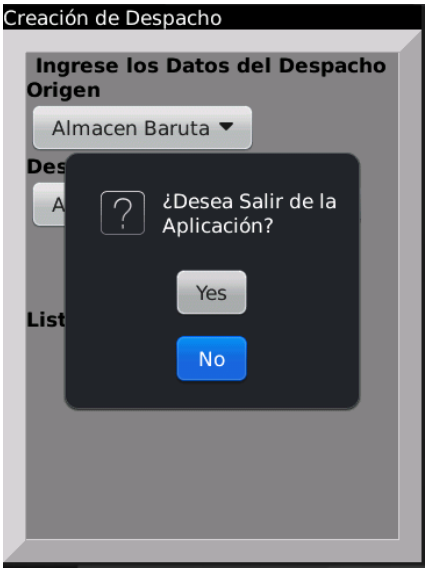


Figura 80 Mensaje de confirmación de cierre de la aplicación

4.2.2 Aplicación móvil Manejo de Despachos

A continuación se describe el funcionamiento de la aplicación móvil de manejo de despachos. La figura 81 muestra el ícono de la aplicación Manejo de Despacho en el dispositivo móvil.

Figura 81 Aplicación de Creación de Despachos instalada en el móvil

Al ejecutar la aplicación Manejo de Despacho, se desplegará una vista con el botón *Escanear Despacho*. Al pulsar el botón se abrirá el lector de códigos QR donde se podrá leer el QR asociado al despacho.

Una vez capturados los datos del despacho contenidos en el código QR (en este caso el número del despacho), se invoca al servicio web de manejo de despacho para obtener los datos asociados al despacho. La figura 82 muestra la vista inicial de la aplicación de manejo de despachos con el botón *Escanear Despacho*

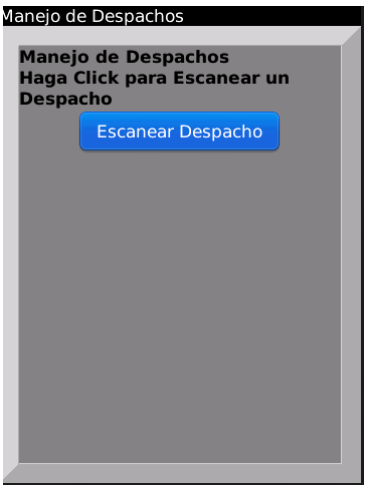


Figura 82 Pantalla inicial de la aplicación de manejo de despachos

En caso que el número de despacho contenido en el QR leído no exista, se abrirá una pantalla con un mensaje indicando que no existe el despacho solicitado como lo muestra la figura 83. Pulsar el botón *Volver* para regresar a la pantalla inicial de manejo de despachos.

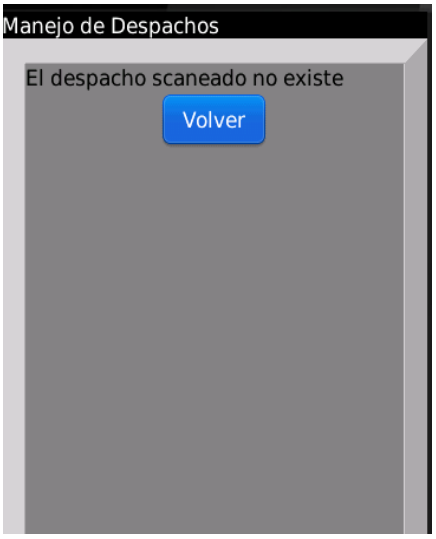


Figura 83 Vista indicando que no existe el despacho solicitado.

En caso que el despacho escaneado exista y su estatus sea *Creado*, se desplegará una vista con los datos básicos asociados al despacho (número de despacho, origen, destino, fecha de creación y estatus), un botón para visualizar los productos asociados al despacho, un campo observaciones (a ser llenado en caso de cancelar el despacho) y los botones de las acciones a tomar con el despacho (Estas acciones dependen del estatus actual del despacho, en este caso Despachar y Cancelar).

La figura 84 muestra la pantalla que visualiza los datos básicos del despacho escaneado en estatus creado y los estatus siguientes que se le puede asignar al despacho.

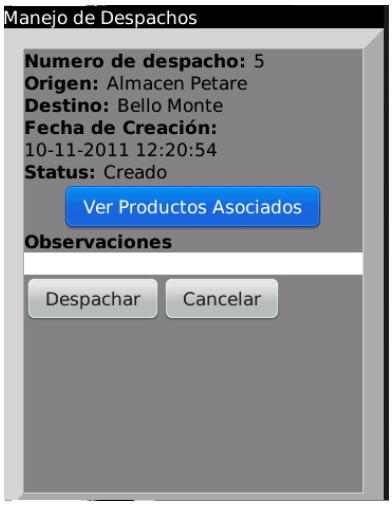


Figura 84 Vista que muestra los datos básicos de un despacho en estatus creado.

En caso de pulsar el botón *Ver Productos Asociados*, se desplegará una vista con la lista de productos asociados al despacho. Para volver a la pantalla anterior pulse el botón *Atrás*. La figura 85 muestra la pantalla con la lista de productos asociados al despacho.

Figura 85 Vista que muestra la lista de productos asociados al despacho.

En caso que se desee realizar el despacho, se debe pulsar el botón *Despachar*. Se mostrará un mensaje de confirmación para asegurar que desea realizar el despacho. En caso de pulsar afirmativamente se invocará el servicio web de manejo de despachos, al cual se le enviará el número del despacho, el próximo estatus a asignarle y el pin del dispositivo que realiza la operación (con el fin de llevar un control de quien está realizando el cambio de estatus).

La figura 86 muestra el mensaje de confirmación al momento de pulsar el botón despachar.

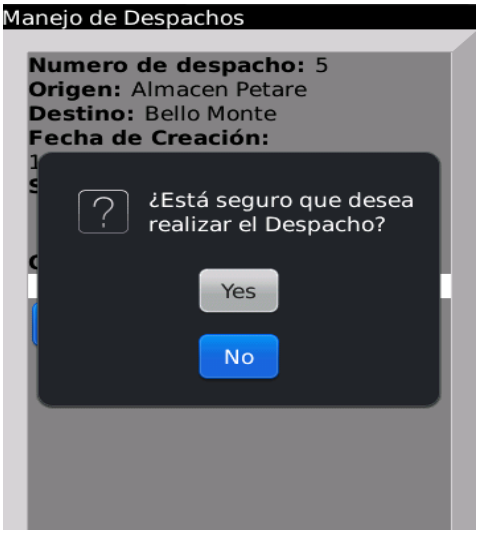


Figura 86 Confirmación de cambio de estatus a Despachado.

En caso que el cambio de estatus a *Despachado* haya sido exitoso, es decir que el servicio actualizó correctamente el estatus del despacho, el servicio enviará una respuesta al móvil indicando el éxito de la transacción y se mostrará un mensaje de confirmación del cambio de estatus del despacho. Se deshabilitan los botones de las acciones a tomar con el despacho como muestra la figura 87.

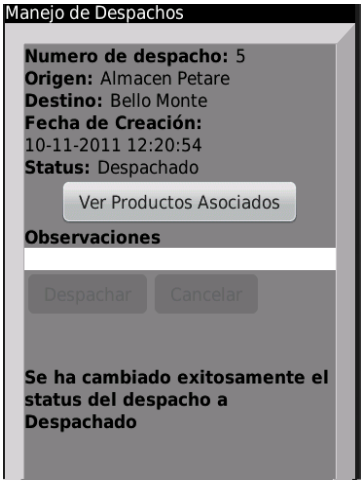


Figura 87 Pantalla de visualización de cambio de estatus a Despachado exitoso.

En caso que se desee cancelar el despacho, se debe pulsar el botón *Cancelar*. Se mostrará un mensaje de confirmación para asegurar que desea cancelar el despacho. En caso de pulsar afirmativamente se invocará el servicio web de manejo de despachos, al cual se le enviará el número del despacho, el próximo estatus a asignarle, el pin del dispositivo que realiza la operación (con el fin de llevar un control de quien está realizando el cambio de estatus) y la data insertada en el campo observación, lo cual es una explicación de la razón de la cancelación del despacho.

La figura 88 muestra el mensaje de confirmación al momento de pulsar el botón cancelar.

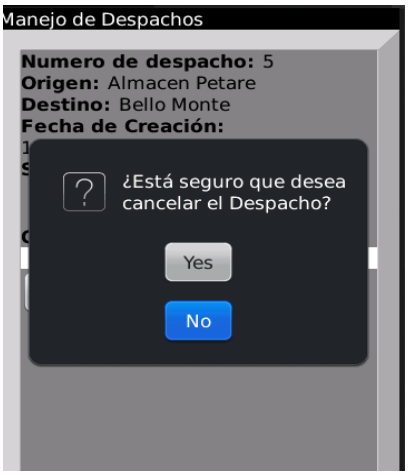


Figura 88 Confirmación de cambio de estatus a Cancelado.

En caso de pulsar afirmativamente a la cancelación del despacho y no haberse llenado el campo observación, aparecerá un mensaje de error indicando que debe llenarse dicho campo, tal como aparece en la figura 89.

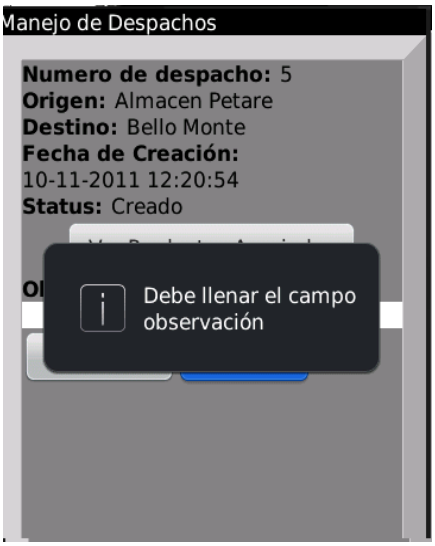


Figura 89 Mensaje de alerta que indica que debe llenarse el campo Observaciones.

La figura 90 muestra cómo debe llenarse el campo observaciones.

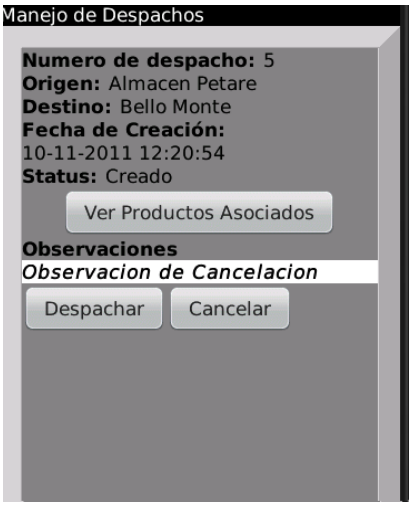


Figura 90 Vista del llenado del campo Observaciones.

En caso que el cambio de estatus a *Cancelado* haya sido exitoso, es decir que el servicio actualizó correctamente el estatus del despacho, el servicio enviará una respuesta al móvil indicando el éxito de la transacción y se mostrará un mensaje de confirmación del cambio de estatus del despacho. Se deshabilitan los botones de las acciones a tomar con el despacho, como muestra la figura 91.

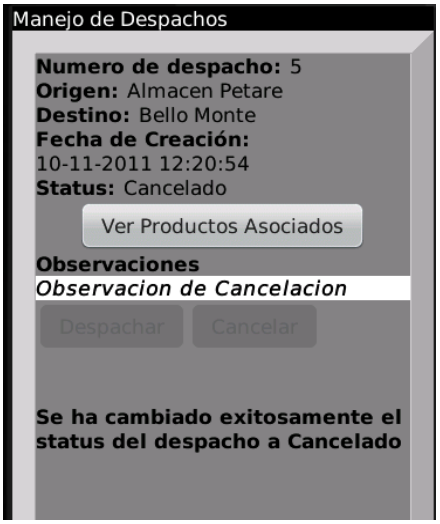


Figura 91 Pantalla de visualización de cambio de estatus a Despachado exitoso.

En caso de que el despacho escaneado exista y su estatus sea *Despachado*, se abrirá una vista con los datos básicos asociados al despacho (número de despacho, origen, destino, fecha de creación, fecha de despacho y estatus), un botón para visualizar los productos asociados al despacho, un campo observaciones (a ser llenado en caso de rechazar el despacho) y los botones de las acciones a tomar con el despacho (estas acciones dependen del estatus actual del

despacho, en este caso Recibir y Rechazar). La figura 92 muestra la vista que despliega los datos básicos del despacho escaneado en estatus despachado y los estatus siguientes que se le puede asignar al despacho.

Figura 92 Pantalla que muestra los datos básicos de un despacho en estatus despachado.

En caso de pulsar el botón *Ver Productos Asociados*, se abrirá una vista con la lista de productos asociados al despacho. Para volver a la pantalla anterior pulse el botón *Atrás*. La figura 93 muestra la vista con la lista de productos asociados al despacho.

Figura 93 Vista que muestra la lista de productos asociados al despacho.

En caso que se desee recibir el despacho, se debe pulsar el botón *Recibir*. Se mostrará un mensaje de confirmación para asegurar que desea recibir el despacho. En caso de pulsar afirmativamente, se invocará el servicio web de manejo de despachos, al cual se le enviará el número del despacho, el próximo estatus a asignarle y el pin del dispositivo que realiza la operación (con el fin de

llevar un control de quien está realizando el cambio de estatus). La figura 94 muestra el mensaje de confirmación al momento de pulsar el botón recibir.

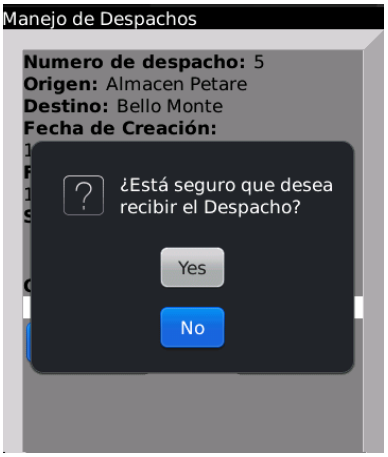


Figura 94 Confirmación de cambio de estatus a Recibido.

En caso que el cambio de estatus a *Recibido* haya sido exitoso, es decir que el servicio actualizó correctamente el estatus del despacho, el servicio enviará una respuesta al móvil indicando el éxito de la transacción y se mostrará un mensaje de confirmación del cambio de estatus del despacho. Se deshabilitan los botones de las acciones a tomar con el despacho, como lo muestra la figura 95.

Figura 95 Pantalla de visualización de cambio de estatus a Recibido exitoso

En caso que se desee rechazar el despacho, se debe pulsar el botón *Rechazar*. Se mostrará un mensaje de confirmación para asegurar que desea rechazar el despacho. En caso de pulsar afirmativamente se invocará el servicio web de manejo de despachos, al cual se le enviará el número del despacho, el próximo

estatus a asignarle, el pin del dispositivo que realiza la operación (con el fin de llevar un control de quien está realizando el cambio de estatus) y el texto insertado en el campo *Observación*, lo cual es una explicación de la razón del rechazo del despacho. La figura 96 muestra el mensaje de confirmación al momento de pulsar el botón rechazar.

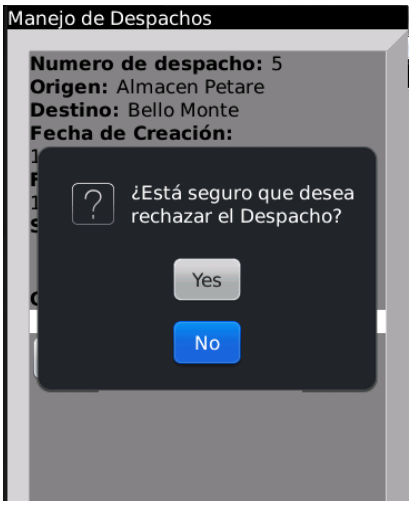


Figura 96 Confirmación de cambio de estatus a Rechazado.

En caso de pulsar afirmativamente al rechazo del despacho y no haberse llenado el campo *Observación*, aparecerá un mensaje de error indicando que debe llenarse dicho campo, tal como aparece en la figura 97.

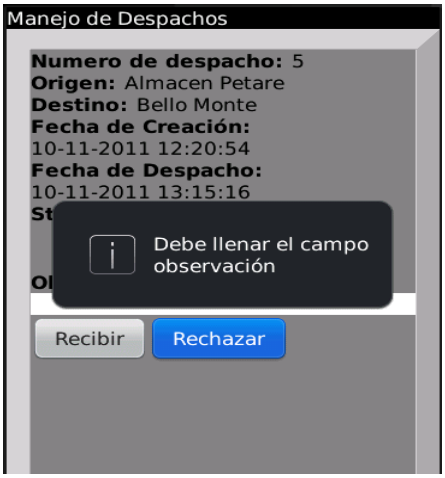


Figura 97 Mensaje de alerta que indica que debe llenarse el campo Observaciones.

La figura muestra cómo debe llenarse el campo observaciones.

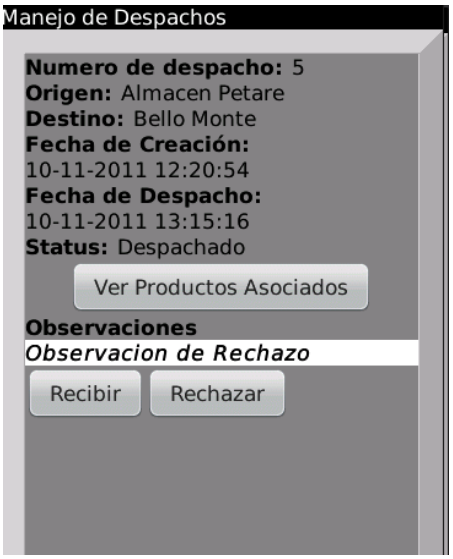


Figura 98 Vista con el llenado del campo Observaciones.

En caso que al escanear el QR del despacho, el mismo exista y su estatus sea *Recibido*, se abrirá la pantalla con los datos básicos asociados al despacho, además con un dato adicional como lo es la fecha de recepción y el botón para ver los productos asociados. Al ser un estatus final (es decir no tiene estatus siguientes) no salen botones de acciones a tomar, en este caso la información mostrada del despacho es sólo de consulta. La figura 99 muestra la información desplegada de un despacho en estatus Recibido.

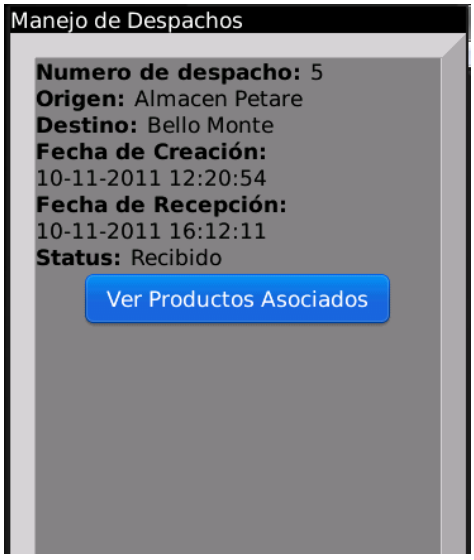


Figura 99 Información de despacho en estatus Recibido.

En caso que al escanear el QR del despacho, el mismo exista y su estatus sea *Cancelado*, se abrirá la pantalla con los datos básicos asociados al despacho, además con un dato adicional como lo es la fecha de cancelación y el botón para ver los productos asociados. Al ser un estatus final (es decir no tiene estatus siguientes) no salen botones de acciones a tomar, en este caso la información mostrada del despacho es sólo de consulta. La figura 100 muestra la información que se despliega de un despacho en estatus Cancelado.

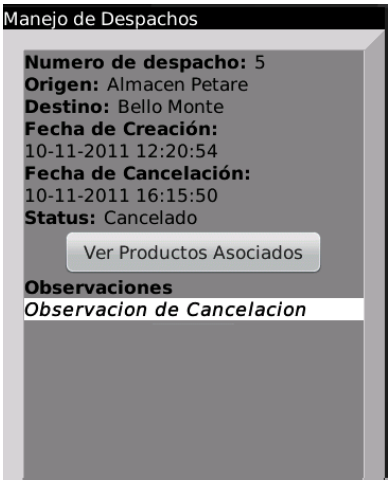


Figura 100 Información de despacho en estatus Cancelado.

En caso que al escanear el QR del despacho, el mismo exista y su estatus sea *Rechazado*, se abrirá la pantalla con los datos básicos asociados al despacho, además con un dato adicional como lo es la fecha de rechazo y el botón para ver los productos asociados. Al ser un estatus final (es decir no tiene estatus siguientes) no salen botones de acciones a tomar, en este caso la información mostrada del despacho es sólo de consulta. La figura muestra la información que se despliega de un despacho en estatus Rechazado.

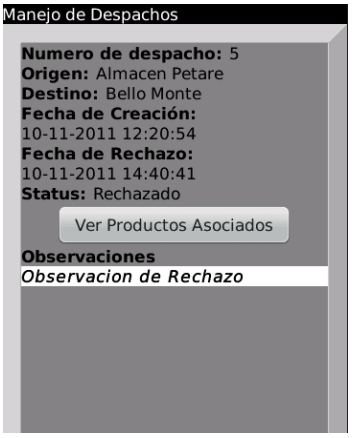


Figura 101 Información de despacho en estatus Rechazado.

Capítulo 5 Conclusiones

A continuación se presentan las conclusiones del Trabajo Especial de Grado.

Se cumplió el objetivo principal del Trabajo Especial de Grado al integrarse varias tecnologías en la captura y procesamiento de códigos QR, lo que nos permitió un manejo automatizado de la data capturada y apoyar el proceso de negocio conocido como Gestión de Despachos. Los objetivos específicos también fueron alcanzados.

Se aprovecharon las ventajas de los códigos QR como herramienta de almacenamiento de información de productos asociados a despachos, debido a su capacidad de contener información diversa (lo que permite adaptar la solución para cualquier tipo de producto), su fácil creación y codificación, su capacidad de lectura y su gran capacidad de corrección de errores. Esto último tiene vital importancia, debido a que, al ser productos que son transportados a grandes distancias, son más sensibles a borrarse o ensuciarse y por ende dañarse.

Las soluciones móviles construidas para el manejo de despachos permitieron la automatización del proceso de captura y procesamiento de los códigos QR asociados a los despachos, además nos dio la ventaja inherente de la movilidad, la comodidad de utilizar algo tan común en estos días como los móviles para la lectura de los códigos y no depender de lectores especializados. Al ser el móvil un dispositivo fácilmente identificable, se pudo obtener información de que dispositivos realizaban las transacciones dentro del proceso. Esto es de importancia prioritaria para las empresas, debido a que es de su interés por cuestiones de seguridad y logística, tener el registro de quienes realizan las transacciones.

Para esta solución se desarrollaron las aplicaciones móviles con lectores de códigos QR embebidos en los dispositivos, que permitieron integrar la solución QR al flujo del proceso de manejo de envío de despachos. Dichas aplicaciones móviles fueron desarrolladas para dispositivos Blackberry, usando Java Micro Edition (JME), que es una especificación de un subconjunto del API de Java para el desarrollo aplicaciones móviles. Específicamente se usó el Blackberry API Reference.

Hubo una curva de aprendizaje importante en el aprendizaje de dicha herramienta, específicamente en manejo de pantallas, elementos de la interfaz gráfica, integración de lectura de códigos QR, manejo de peticiones web, procesamiento de data enviada y recibida, entre otros.

La data contenida en el QR, fue capturada en el móvil, procesada y posteriormente transmitida desde el móvil a un backend.

En el proceso, no sólo se capturó la data de códigos QR existentes, sino que además se crearon códigos QR dinámicos en base a la información generada en la creación de despachos, lo que nos da un valor agregado al objetivo principal fijado. Esto es una gran ventaja, debido a que puede integrarse soluciones de datos generados dinámicamente con los códigos QR.

Para el procesamiento de datos a transmitir desde y hacia el móvil, se invirtió una importante cantidad de tiempo en buscar herramientas alternativas a las ya conocidas como XML, que proveyeran ventajas en cuanto a procesamiento de la data y en la velocidad de transmisión de la misma. De esta forma se utilizó un formato ligero y simple de datos llamado JSON, de fácil procesamiento y manejo, el cual resultó ser bastante sencillo de entender debido a su simplicidad, además de que posee una gran cantidad de documentación y numerosas APIs en distintos lenguajes de programación.

La versatilidad y la alta capacidad de almacenamiento de los códigos QR permitieron que el formato JSON no sólo fuera usado para la transmisión de datos, sino que además se usó para la información contenida dentro de los QR usados en la aplicación móvil. Esto permitió que al momento de leer el QR, el procesamiento de la data fuese igual de sencillo que procesar la data que se envía y se recibe en el móvil. De esta forma se integraron estas dos tecnologías donde una aprovechó las bondades de la otra.

La construcción de los servicios web fue realizada usando el framework web Apache Wicket. Para esto también hubo una curva de aprendizaje importante con el fin de poder realizar eficazmente la recepción de la data desde el móvil (request), el procesamiento, la persistencia y el envío posterior de la data hacía el móvil (response). Fueron desarrollados varios servicios que se encargaron de recibir la petición en formato JSON, procesarla y persistirla. El procesamiento de la petición JSON fue realizada por una API Java llamada JSON.simple, la cual contiene un conjunto de herramientas para el manejo del formato JSON.

La aplicación web de consulta de despachos también fue realizada usando el framework web Apache Wicket. Debido a su facilidad de aprendizaje y manejo, el desarrollo de esta funcionalidad fue bastante sencillo, también en parte gracias a la experiencia previa que tengo desarrollando bajo esta herramienta.

En cuanto a los códigos QR podemos concluir lo siguiente:

El código QR es capaz de contener información en ambas direcciones (verticalmente y horizontalmente) a diferencia de los tradicionales códigos de barra, que tan sólo son capaces de almacenar información en una dirección. Precisamente por este motivo, la capacidad de almacenamiento es mayor en el caso del código QR.

El código QR tiene la capacidad de corregir errores. Se pueden restaurar los datos si parte del código está dañado o manchado.

El código QR puede ser leído a alta velocidad desde todas las direcciones (en 360º).

Al ser un código cuyas especificaciones han sido publicadas, ha permitido la proliferación de lectores de código QR de muy bajo coste o incluso gratuitos. Además, se han desarrollado aplicaciones de software que permiten descifrar el código QR. Muchas de ellas son gratuitas e incluso de código libre.

La integración con dispositivos móviles (teléfonos y PDAs) ha permitido que la mayoría de los teléfonos puedan leer los códigos QR, puesto que sólo necesitan tener una cámara de fotos para la captura de los códigos y una aplicación (que en muchos casos es gratuita) para descifrar la información contenida en los mismos, esto permite que en cualquier lugar y en cualquier momento, este pueda ser leído, sin necesidad de contar con un lector especializado para leer este tipo de Código.

El código QR permite contener información diversa en sus imágenes de mosaico: urls, textos de hasta 250 caracteres, números telefónicos, etc. Un teléfono con cámara integrada que tenga instalado un lector de códigos QR, puede decodificar estos gráficos tomándole una fotografía y conectar así instantáneamente al usuario con, por ejemplo, un sitio en Internet, o enviarle un mensaje de texto, un video, o una foto.

Capítulo 6 Bibliografía y Referencias

Códigos QR
DENSO WAVE

Códigos QR, características, estandarización, creación, casos de estudio.

http://www.denso-wave.com/qrcode/index-e.html

Códigos QR
Blog Códigos QR

Información sobre Códigos QR, Lectores y Generadores QR Codes

http://www.codigos-qr.com/

Generador de Códigos QR

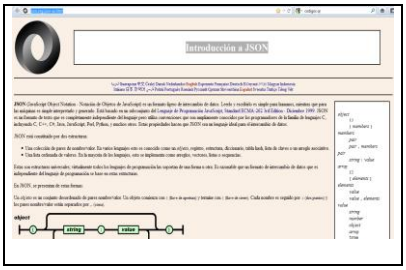
Generación de diferentes tipos de códigos QR.

http://qrcode.kaywa.com/

JSON

Introducción a JSON

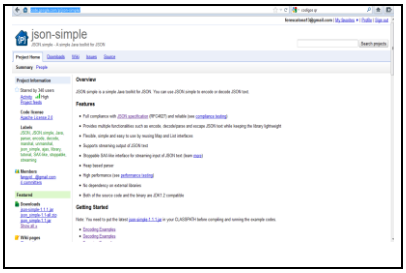
http://json.org/json-es.html



json-simple API

JSON.simple - A simple Java toolkit for JSON

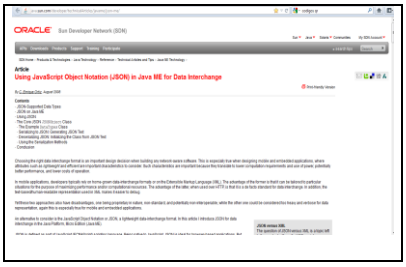
http://code.google.com/p/json-simple/



JSON

Using JavaScript Object Notation (JSON) in Java ME for Data Interchange

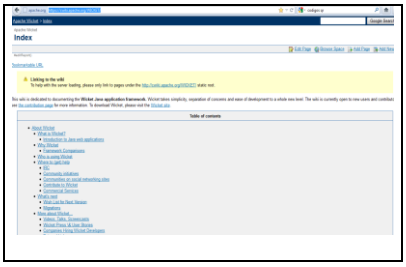
http://java.sun.com/developer/technicalArticles/javame/json-me/



Wicket Wiki

Wiki en inglés especializado en Wicket

https://cwiki.apache.org/WICKET/



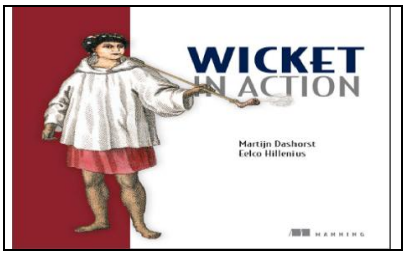
Apache Wicket

Web oficial de Wicket

http://wicket.apache.org/

Manning Wicket in Action
Martijn Dashorst, Eelco Hilenius

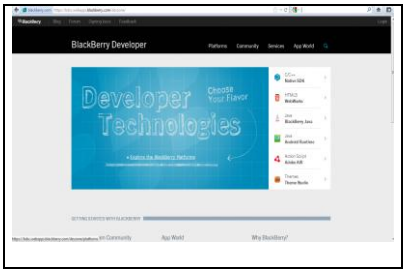
Introducción a Wicket, definición, arquitectura, ejemplos.
USA, 2008.



Blackberry Developers

Sitio web oficial para desarrolladores de Blackberry Applications

http://us.blackberry.com/developers



Blackberry JDE API – User Interface Field Reference

Sitio web de componentes de desarrollo para Blackberry Applications

http://devsushi.com/2007/12/02/blackberry-jde-api-user-interface-field-reference/