

**ESCUELA POLITÉCNICA DEL EJÉRCITO**

**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**

**“ANÁLISIS, DISEÑO Y DESARROLLO DE UN SISTEMA  
DE CONSTATAción FÍSICA DE BIENES PARA LA  
DIRECCIÓN FINANCIERA DE LA ESPE USANDO  
DISPOSITIVOS MÓVILES Y LA PLATAFORMA .NET”**

**Previa a la obtención del Título de :**

**INGENIERO EN SISTEMAS E INFORMÁTICA**

**POR:**

**JUAN FRANCISCO TAXI TOPON**

**SANGOLQUI, 22 de Diciembre de 2008**

## **CERTIFICACION**

Certifico que el presente trabajo fue realizado en su totalidad por el Sr. JUAN FRANCISCO TIXI TOPON como requerimiento parcial a la obtención del título de INGENIERO EN SISTEMAS E INFORMÁTICA.

Sangolquí, 22 de Diciembre de 2008

---

INGENIERA LORENA DUQUE  
DIRECTOR

## DEDICATORIA

El presente trabajo lo dedico a mi madre Anita a mis hermanos Ximena y Gustavo a mi padre Francisco, quienes me han ayudado durante el transcurso mi vida universitaria, a mi novia Mery que gracias a su empeño y motivación me ha incentivado el finalizar mi tesis.

Francisco Tixi

## **AGRADECIMIENTOS**

Agradezco a Dios, a mi madre Anita quien ha luchando y a sufrido por ver este trabajo finalizado, a Ximena y Gustavo quienes en todo momento me supieron apoyar, a Mery que permanentemente a estado con migo en todo momento.

Al Ing. Ramiro Delgado a la Ing. Lorena Duque, Codirector y Director respectivamente, quienes me han brindado grandes enseñanzas.

Al Ing. Byron Amores, quien durante todo este tiempo me ha estado motivando y ayudando para finalizar mi tesis.

A todos quienes conforman la Sección Bienes de la Espe

Francisco Tixi

## INDICE DE CONTENIDOS

CERTIFICACION.....	ii
DEDICATORIA.....	iii
AGRADECIMIENTOS.....	iv
INDICE DE CONTENIDOS.....	v
INDICE DE FIGURAS.....	viii
INDICE DE TARJETAS.....	xi
INDICE DE HISTORIAS DE USUARIO.....	xii
INDICE DE TABLAS.....	xiii
INDICE DE DIAGRAMAS.....	xiv

**RESUMEN .....**¡Error! Marcador no definido.

### **CAPITULO I..... 2**

1. GENERALIDADES.....	2
1.1. INTRODUCCION.....	2
1.2. ANTECEDENTES.....	3
1.3. JUSTIFICACION.....	4
1.4. OBJETIVOS.....	5
1.4.1. Objetivo General.....	5
1.4.2. Objetivos Específicos.....	6
1.5. ALCANCE.....	6
1.6. FACTIBILIDAD.....	8
1.6.1. TECNICA.....	8
1.6.2. HUMANA.....	8

### **CAPITULO II..... 9**

2. MARCO TEORICO.....	9
2.1. ADMINISTRACION Y CONTROL DE ACTIVOS FIJOS.....	9
2.1.1. CONCEPTOS GENERALES.....	9
2.1.1.1. Activo fijo.....	9
2.1.1.2. Bienes considerados activos fijos.....	9
2.1.1.3. Valor y tiempo de vida útil para ser considerado A.F.....	9
2.1.1.4. Bienes que no se consideran activos fijos.....	10
2.1.2. CODIFICACION DE ACTIVOS FIJOS.....	10
2.1.3. ADMINISTRACION Y CONTROL DE ACTIVOS FIJOS.....	12
2.1.3.1. Controles varios mediante formularios.....	12
2.1.3.2. Diseño de formularios.....	13
2.1.3.3. Normas de Administración y Control de Bienes.....	13
2.1.4. RUTINA DESCRIPTIVA DE ALTA DE BIENES.....	14
2.1.4.1. Encargado Administrativo.....	14
2.1.4.2. Conservadores/custodios de bienes.....	14
2.1.4.3. Contabilidad.....	15
2.1.4.4. Procesamiento automatizado de datos – PAD.....	16
2.1.5. RUTINA DESCRIPTIVA DE BAJA DE BIENES.....	16
2.1.6. NORMAS PARA ADMINISTRACION Y CONTROL DE BIENES	
16	
2.1.6.1. Ámbito de aplicación.....	16
2.2. METODOLOGÍA DE PROGRAMACIÓN EXTREMA (XP).....	18
2.2.1. Fases de la Metodología de Programación Extrema.....	19
2.2.1.1. Planificación.....	20

2.2.1.2.	Diseño.....	22
2.2.1.3.	Desarrollo .....	25
2.2.1.4.	Pruebas .....	28
2.3.	Plataforma .Net.....	29
2.3.1.	Plataforma .Net .....	29
2.3.1.1.	.Net Framework .....	30
2.3.1.2.	.Net Compact Framework .....	36
2.3.2.	Capa de Interfaz .....	38
2.3.2.1.	Visual C#.net .....	38
2.3.3.	Capa Lógica.....	39
2.3.3.1.	Web Services .....	39
2.3.3.2.	Protocolo Soap.....	40
2.3.3.3.	Multifuncionalidad entre sistemas operativos .....	45
2.3.4.	Capa de datos.....	46
2.3.4.1.	Base de datos SQL Server .....	46
2.3.4.2.	Base de datos SQL CE .....	47
2.4.	Dispositivos Móviles .....	48
2.4.1.	Pocket PC .....	49
2.4.2.	Windows CE.....	50
2.5.	Nunit testeador de pruebas unitarias .....	50
<b>CAPITULO III.....</b>		<b>52</b>
3.	DISEÑO DEL SISTEMA DE CONSTATAcion FISICA.....	52
3.1.	INTRODUCCION .....	52
3.2.	PLANIFICACIÓN.....	53
3.2.1.	HISTORIAS DE USUARIO .....	53
3.2.2.	PLAN DE ENTREGAS .....	57
3.2.3.	VELOCIDAD DEL PROYECTO .....	58
3.2.4.	PLAN DE ITERACIONES .....	58
3.2.5.	ROTACIONES .....	59
3.2.6.	REUNIONES.....	60
3.3.	DISEÑO DE LA SOLUCION .....	60
3.3.1.	METAFORA DEL SISTEMA .....	60
3.3.2.	TARJETAS CRC .....	60
3.3.3.	SOLUCIONES PUNTUALES .....	65
3.3.4.	FUNCIONALIDAD MINIMA.....	65
3.3.5.	RECICLAJE .....	66
<b>CAPITULO IV .....</b>		<b>67</b>
4.	DESARROLLO DEL SISTEMA .....	67
4.1.	INTRODUCCION .....	67
4.2.	DISPONIBILIDAD DEL CLIENTE.....	67
4.3.	UNIDADES DE PRUEBA .....	68
4.4.	PROGRAMACION POR PAREJAS .....	69
4.5.	INTEGRACION.....	70
4.6.	ESTANDARES .....	70
4.6.1.	CODIFICACION.....	70
4.6.2.	INTERFAZ DE USUARIO.....	72
4.6.2.1.	POCKET PC.....	72
4.6.2.2.	WINDOWS.....	74
<b>CAPITULO V.....</b>		<b>79</b>
5.	PRUEBAS .....	79

5.1.	DE IMPLANTACION.....	82
5.2.	DE ACEPTACION.....	83
<b>CAPITULO VI.....</b>		<b>85</b>
6.	CONCLUSIONES, RECOMENDACIONES Y ANEXOS.....	85
6.1.	CONCLUSIONES.....	85
6.2.	RECOMENDACIONES.....	86
6.3.	ANEXO A: MANUAL DE INSTALACION DE WEB SERVICE Y APLICACIONES.....	87
6.4.	ANEXO B: MANUAL DEL PROGRAMADOR.....	87
6.5.	ANEXO C: MANUAL DE USUARIO.....	87
	BIBLIOGRAFIA.....	178
	GLOSARIO.....	179

## INDICE DE FIGURAS

Figura 2.1 Fases de la Programación Extrema.....	19
Figura 2.2 Common Language Runtime (CLR).....	31
Figura 2.3.: Biblioteca de Clases de .Net Famework.....	34
Figura 2.4: Biblioteca de Clases.....	38
Figura 2.5: Esquema de Web Services.....	40
Figura 2.6: SQL CE.....	47
Figura 2.6: PDA – Falcon 4210.....	49
Figura 3.1: Plan de entregas de historias de usuario.....	58
Figura 3.2: Plan de Iteraciones.....	59
Figura 4.1: Ventana test Creación de username.....	69
Figura 4.2: Caja de Texto.....	72
Figura 4.3: Etiqueta (label).....	72
Figura 4.4: Etiquetas (creadas por código).....	73
Figura 4.5: Radio botones.....	73
Figura 4.6: Datagrids.....	73
Figura 4.7: Mensajes.....	74
Figura 4.8: Ventana acceso al sistema de los POCKET PC.....	74
Figura 4.9: Caja de texto.....	75
Figura 4.10: Hiper link.....	75
Figura 4.11: XpLabels.....	75
Figura 4.12: Check box.....	75
Figura 4.13: Datagrid.....	75
Figura 4.14: Control colapsable.....	76
Figura 4.15: Ventana de acceso al sistema de administración.....	76
Figura 5.1: Ventana test Creación de username.....	83
Figura A.1: Pagina de prueba de IIS.....	91
Figura A.2: Inicio de Windows.....	94
Figura A.3: Acceso a Oracle.....	95
Figura A.4: Opciones instaladas en Oracle.....	95
Figura A.5: Configuración de Tnsname.....	95
Figura A.6: Net Configuration Assistant.....	96
Figura A.7: Selección de Configuración de Nombre del Servicio de Red Local.....	96
Figura A.8: Selección de la opción Agregar.....	97
Figura A.9: Selección de la base de datos.....	97
Figura A.10: Selección del nombre del servicio.....	98
Figura A.11: Selección del protocolo de red.....	98
Figura A.12: Ingreso del nombre del host.....	99
Figura A.13: Probar conexión.....	99
Figura A.14: Pagina de prueba del Web Services Athentication.aspx.....	100



<b>Figura A.15: Creación de Mobile Device .....</b>	<b>101</b>
<b>Figura A.16: Pocket PC .....</b>	<b>103</b>
<b>Figura B.1: Inicio de Windows .....</b>	<b>107</b>
<b>Figura B.2: Acceso a Oracle .....</b>	<b>107</b>
<b>Figura B.3: Opciones instaladas en Oracle .....</b>	<b>108</b>
<b>Figura B.4: Configuración de Tnsname .....</b>	<b>108</b>
<b>Figura B.5: Net Configuration Assistant .....</b>	<b>108</b>
<b>Figura B.6: Selección de Configuración de Nombre del Servicio de Red Local.....</b>	<b>109</b>
<b>Figura B.7: Selección de la opción Agregar .....</b>	<b>109</b>
<b>Figura B.8: Selección de la base de datos.....</b>	<b>110</b>
<b>Figura B.9: Selección del nombre del servicio .....</b>	<b>110</b>
<b>Figura B.10: Selección del protocolo de red.....</b>	<b>111</b>
<b>Figura B.11: Ingreso del nombre del host.....</b>	<b>111</b>
<b>Figura B.12: Probar conexión .....</b>	<b>112</b>
<b>Figura B.13: Servicios de Internet Infomation Server .....</b>	<b>113</b>
<b>Figura C.1: Inicio del Sistema parte Móvil.....</b>	<b>150</b>
<b>Figura C.2: Ventana de Logon .....</b>	<b>150</b>
<b>Figura C.3: Conexión con el Web Service .....</b>	<b>151</b>
<b>Figura C.4: Mensaje de Error: Username o Password incorrectos... </b>	<b>151</b>
<b>Figura C.5: Mensaje de Error: Usuario sin autorización .....</b>	<b>152</b>
<b>Figura C.6: Ventana de Procesos .....</b>	<b>152</b>
<b>Figura C.7: Salida de Sesión.....</b>	<b>153</b>
<b>Figura C.8: Ventana de logeo fuera de línea.....</b>	<b>154</b>
<b>Figura C.9: Ventana de Descarga de Información.....</b>	<b>155</b>
<b>Figura C.10: Búsqueda de una determinada dependencia .....</b>	<b>156</b>
<b>Figura C.11: Mensaje de Error: No existe ningún texto a buscar ....</b>	<b>156</b>
<b>Figura C.12: Mensaje de Error: Seleccione el nombre de una dependencia .....</b>	<b>157</b>
<b>Figura C.13: Mensaje de no existencia de bienes .....</b>	<b>157</b>
<b>Figura C.14: Ventana de constatación de bienes .....</b>	<b>158</b>
<b>Figura C.15: Ventana de Datos Generales.....</b>	<b>158</b>
<b>Figura C.16: Ventana de Responsables actuales del bien .....</b>	<b>159</b>
<b>Figura C.17: Confirmación de eliminación de un responsable .....</b>	<b>159</b>
<b>Figura C.18: Búsqueda de un nuevo responsable.....</b>	<b>160</b>
<b>Figura C.19: Confirmación de asignación de un nuevo responsable. </b>	<b>160</b>
<b>Figura C.20: Ventana del estado actual del bien .....</b>	<b>161</b>
<b>Figura C.21: Ventana de la dependencia actual del bien.....</b>	<b>161</b>
<b>Figura C.22: Búsqueda de una nueva dependencia.....</b>	<b>162</b>
<b>Figura C.23. Mensaje de actualización de la información del bien ...</b>	<b>163</b>
<b>Figura C.24. Mensaje de Sincronización.....</b>	<b>164</b>
<b>Figura C.25. Mensaje de no existencia de bienes a actualizar .....</b>	<b>164</b>
<b>Figura C.26. Ventana de acceso al módulo de Windows.....</b>	<b>165</b>

<b>Figura C.27: Mensaje de Error PC: Username o Password incorrectos</b>	<b>166</b>
<b>Figura C.28: Ventana Cambiar Password</b>	<b>166</b>
<b>Figura C.29: Mensaje: La clave ha sido actualizada</b>	<b>167</b>
<b>Figura C.30: Mensaje de Error PC: La nueva clave y su confirmación no son iguales</b>	<b>167</b>
<b>Figura C.31: Ventana de Administración de Seguridad</b>	<b>168</b>
<b>Figura C.32: Ventana: Crear Usuarios</b>	<b>169</b>
<b>Figura C.33: Mensaje: Generación de UserName</b>	<b>169</b>
<b>Figura C.34: Elección de Usuario a Modificar o Eliminar</b>	<b>170</b>
<b>Figura C.35: Mensaje: La Información fue actualizada</b>	<b>170</b>
<b>Figura C.36: Mensaje: La Información fue eliminada</b>	<b>171</b>
<b>Figura C.37: Ventana: Crear Perfiles y Asignación de módulos</b>	<b>171</b>
<b>Figura C.38: Mensaje: Perfil Creado</b>	<b>172</b>
<b>Figura C.39: Ventana: Elección de Perfil a Modificar o Eliminar</b>	<b>172</b>
<b>Figura C.40: Mensaje: La Información fue actualizada</b>	<b>173</b>
<b>Figura C.41: Mensaje: La Información fue eliminada</b>	<b>173</b>
<b>Figura C.42: Ventana: Asignación de módulos</b>	<b>174</b>
<b>Figura C.43: Mensaje: La asignación fue actualizada</b>	<b>174</b>
<b>Figura C.44: Ventana: Asignación de perfiles</b>	<b>175</b>
<b>Figura C.45: Mensaje: La asignación fue realizada</b>	<b>175</b>
<b>Figura C.46: Ventana de Reportes</b>	<b>176</b>
<b>Figura C.47: Reportes de Usuarios del Sistema</b>	<b>176</b>

## INDICE DE TARJETAS

<b>Tarjeta 2.1 Tarjeta CRC.....</b>	<b>24</b>
<b>Tarjeta 3.1 CRC USUARIO .....</b>	<b>61</b>
<b>Tarjeta 3.2 CRC SEGURIDAD.....</b>	<b>61</b>
<b>Tarjeta 3.3 CRC BIENES.....</b>	<b>62</b>
<b>Tarjeta 3.4 CRC DESCARGAR.....</b>	<b>62</b>
<b>Tarjeta 3.5 CRC SINCRONIZAR .....</b>	<b>63</b>
<b>Tarjeta 3.6 CRC BIENESLOG .....</b>	<b>63</b>
<b>Tarjeta 3.7 CRC WEBSERVICES.....</b>	<b>64</b>

## INDICE DE HISTORIAS DE USUARIO

<b>Historia 3.1 Historia de Usuario 1 .....</b>	<b>54</b>
<b>Historia 3.2 Historia de Usuario 2 .....</b>	<b>55</b>
<b>Historia 3.3 Historia de Usuario 3 .....</b>	<b>55</b>
<b>Historia 3.4 Historia de Usuario 4 .....</b>	<b>55</b>
<b>Historia 3.5 Historia de Usuario 5 .....</b>	<b>56</b>
<b>Historia 3.6 Historia de Usuario 6 .....</b>	<b>56</b>
<b>Historia 3.7 Historia de Usuario 7 .....</b>	<b>56</b>
<b>Historia 3.8 Historia de Usuario 8 .....</b>	<b>57</b>
<b>Historia 3.9 Historia de Usuario 9 .....</b>	<b>57</b>

## INDICE DE TABLAS

<b>Tabla 4.1: Estándares de codificación de clases .....</b>	<b>70</b>
--	-----------

## INDICE DE DIAGRAMAS

<b>Diagrama 3.1 Diagrama de Clases .....</b>	<b>64</b>
--	-----------

## RESUMEN

En el presente trabajo el lector encontrara una breve descripción de una de las principales metodologías extremas, que es la base sobre la cual se han ido desarrollando nuevas metodologías de desarrollo de software, la metodología que se describe es programación Extrema, la principal característica de este tipo de metodología es el enfoque que da al usuario, así como la manera de capturar los requerimientos, la realización del diseño del software y el manejo de las pruebas unitarias.

Además encontrara una descripción del Framework de .Net, sus diferentes componentes, una introducción al uso de base de datos empotradas, la descripción de Servicios Web, introducción a uno de los nuevos lenguajes de programación creado por Microsoft, Visual C#.

El lector podrá ver como basado en el proceso metodológico que establece Programación Extrema, se ha creado una solución que funciona en dispositivos móviles usando Visual C#, encontrara los diferentes componentes lógicos como; cadenas de conexión, capa de datos, capa de negocios, web services, que pueden servir de base para la creación de nuevos proyectos.

Encontrara además la manera de implementar y automatizar una de las características más importantes de Programación Extrema como son el uso de las Pruebas Unitarias.

## CAPITULO I

# 1 GENERALIDADES

## 1.1 INTRODUCCION

Con el avance de la tecnología en cuanto a dispositivos móviles, la disminución del costo de estos equipos, y la gran versatilidad y facilidad de llevar y manejar datos e información ha llevado a que varias de las organizaciones cuenten con equipos de este tipo.

Para el manejo de datos e información de manera rápida y oportuna sin tener que contar con un gran equipo como es una laptop o una PC de escritorio, en la actualidad contamos con sofisticados dispositivos móviles. Dispositivos como son Pocket PC, phone, Table PC los mismos que por su rapidez en el encendido, gran capacidad de procesamiento, grandes tiempos de respuesta y gran capacidad de almacenamiento mejoran el manejo de la información.

Poder en cualquier momento transferir, manipular, compartir información han llevado a un rápido crecimiento dentro del mercado de tecnología y soluciones a los conocidos dispositivos móviles. La manera de intercambiar datos e información por parte de los dispositivos móviles es una más de las bondades que nos ofrecen, así como el uso de base de datos empujadas que permiten la independencia de un canal dedicado hacia el servidor de datos, y para la integración de la información contamos con sistemas de Replicación.



Existen en el mercado una gran cantidad de herramientas de desarrollo propietarias y de uso libre que permiten construir software para los dispositivos móviles, para el desarrollo de la solución del presente trabajo, se utilizara Visual Studio .Net 2003, una plataforma desarrollada por Microsoft, orienta a objetos en su totalidad, y usando el nuevo lenguaje de programación Visual C#.

Dentro de las organizaciones tanto publicas como privadas una de las actividades que demanda gran cantidad de recursos humanos y tecnológicos es el manejo y control de sus activos fijos, hablar de la ubicación actual del bien, de la persona que se encuentra responsable en ese momento, del costo real, del tiempo de vida útil, de su valor depreciado y de su estado actual, es una tarea sumamente complicada si no se cuenta con un adecuado sistema que gestione el manejo de los activos y que permita su constatación física.

En el proceso de toma de decisiones con respecto a los activos fijos, es donde se muestra el gran problema que se tiene en su control, ya que mantener la información actualizada en todo momento tiene un costo muy alto por el número de personas que debe realizarlo.

## **1.2 ANTECEDENTES**

La Sección Bienes del Departamento Financiero de la Escuela Politécnica del Ejército es la responsable directa del manejo y control de todos los activos fijos que son propiedad de la organización.

La Sección Bienes lleva el control de los activos fijos y los detalles de cada uno de ellos en hojas Excel para luego cargar la información al sistema de Activos Fijos que se encuentra actualmente, manteniendo de esta manera la información actualizada de acuerdo a sus posibilidades, debido a que el proceso de actualización de información al sistema lo hace la persona encargada del área de sistema que labora en la entidad.

Además la presentación de reportes de bienes es un trabajo que demanda, por parte de los usuarios del sistema gran cantidad de tiempo en su elaboración, la impresión de etiquetas de códigos de barras con las cuales se identifica un bien dentro de la organización es un tarea bastante compleja que por lo general lleva a que los usuarios soliciten a menudo soporte técnico para cumplir con las impresiones.

### **1.3 JUSTIFICACION**

Actualmente se exige que las organizaciones sean más eficientes y competitivas por lo que, cada vez son más necesarias las soluciones móviles para las organizaciones de cualquier tipo y tamaño. Las soluciones móviles aprovechan la seguridad, velocidad y fiabilidad de la tecnología del lado del servidor y cliente, facilitando de esta manera la obtención de software, construido y producido por menos dinero, con gran capacidad en el tiempo de respuesta, procesamiento y seguridad en las transacciones y sobre todo con costos no muy elevados.

Con la finalidad de satisfacer las exigencias de la Sección Bienes, se propone el desarrollo de un sistema de constatación de activos fijos adecuado a sus necesidades.

El sistema a desarrollar permitirá que el proceso de constatación de activos fijos se lleve a cabo en menos tiempo y con un número reducido de personas.

Los principales beneficios que brindará el sistema a desarrollar son los siguientes:

- Permitir la actualización, luego de la constatación del activo fijo
- Brindar a la ESPE una solución baja en costos con un alto rendimiento en el manejo de base de datos y gran facilidad de conexión a su servidor principal.
- Generar reportes de acuerdo a lo que los usuarios necesiten para facilitar la obtención de información y la toma de decisiones a nivel gerencial

## **1.4 OBJETIVOS**

### **1.4.1 Objetivo General**

Analizar, Diseñar y Desarrollar un sistema de constatación física de bienes (activos fijos) para la Escuela Politécnica del Ejército usando dispositivos móviles y .Net como plataforma de desarrollo.

#### **1.4.2 Objetivos Específicos**

- Conocer la potencialidad y alcance de la tecnología .Net, analizando aplicaciones y soluciones existentes.
- Conocer el manejo de los activos fijos que realiza la Sección Bienes, mediante el análisis de los procesos actuales.
- Utilizar como Metodología de desarrollo Programación Extrema para la obtención del producto final.
- Implantar el sistema generando la documentación necesaria, y realizando las pruebas así como la capacitación a los usuarios finales.
- Instalar el software en los equipos móviles (Falcon 420), disponibles por parte de la Escuela Politécnica del Ejército y realizar la prueba piloto.
- Definir un estándar de programación que contribuya y facilite el mantenimiento del código de la solución.

#### **1.5 ALCANCE**

- Desarrollar el marco teórico de referencia
- Se desarrollara una solución que permita realizar la constatación física de bienes (activos fijos) de la Escuela Politécnica del Ejército y genere los reportes que brinden un apoyo tanto al nivel operativo como directivo.
- La solución a desarrollar consumirá los datos que se encuentran residentes en la actual base de datos, con la cual el sistema de activos fijos de la Sección Bines de la Escuela Politécnica del Ejército se encuentra trabajando.

- La solución presentara dos partes que son:

- Cliente Inteligente

Aquí se concentra toda la parte de constatación física del bien y constara de tres procesos que son:

- Cargar datos de la base principal.- almacenará la información de los bienes de acuerdo a una dependencia seleccionada en una base de datos local por Pocket.
- Constatación física del bien.- permitirá verificar la información del bien luego de ser leído su respectivo código de barras.
- Sincronización.- actualizará la base de datos principal con los cambios realizados, y maneja la auditoria de la constatación física de los bienes

- Cliente Windows:

Maneja la parte de seguridades del sistema y reportes, los procesos principales se describen a continuación:

- SEGURIDADES.- Crea nuevos usuarios y permitirá el mantenimiento
- REPORTES.- Genera los reportes diseñados con anterioridad

## **1.6 *FACTIBILIDAD***

### **1.6.1 *TECNICA***

- Un Computador
- Un Pocket

### **1.7 *HUMANA***

- Un Programador

## CAPITULO II

### **2 MARCO TEORICO**

#### **2.1 ADMINISTRACION Y CONTROL DE ACTIVOS FIJOS**

##### **2.1.1 CONCEPTOS GENERALES**

###### **2.1.1.1 Activo fijo**

Es todo bien tangible sujeto a depreciación de acuerdo a su vida útil, que en algunos casos es susceptible de ser trasladado de un lugar a otro, como en el caso de los bienes muebles.

###### **2.1.1.2 Bienes considerados activos fijos**

Son aquellos bienes que reúnan las siguientes condiciones o características:

- Que sean de prioridad de la Institución;
- Que sean de prioridad de la Institución;
- Que sirvan para producir bienes y/o servicios;
- Que tengan un valor razonablemente significativo y sea susceptible de depreciación; y,
- Que por lo menos duren un (1) año en operación.

###### **2.1.1.3 Valor y tiempo de vida útil para ser considerado A.F.**

Para que un bien sea considerado como activo fijo, deberá superar el valor de US\$. 150,00, y su vida útil no será menor de un año calendario.

#### **2.1.1.4 Bienes que no se consideran activos fijos**

Los bienes que no reúnan las condiciones antes indicadas, no se clasificarán como activos fijos; sin embargo, serán controlados y administrados en registros independientes a la clasificación codificada, bajo la responsabilidad del respectivo Custodio que para cada área administrativa o en general, designe la Administración de la Institución.

#### **2.1.2 CODIFICACION DE ACTIVOS FIJOS**

Será identificada y controlada bajo la estructura organizacional de la Organización.

Por lo indicado, se tomará en cuenta, los siguientes aspectos:

1. Institución

Este código es asignado para identificar a la institución.

2. Ubicación geográfica de los bienes

Se le asigna un código a los diferentes lugares donde estén ubicados los activos fijos.

3. Organigrama estructural de la Institución



Independientemente a la codificación contable, y dependiendo de la institución y de el número de departamentos o áreas administrativas que se posea en el organigrama estructural, para identificar con claridad a los conservadores/custodios de inventarios (responsables de la administración y control de activos fijos), cuya codificación se lista en anexo independiente.

#### 4. Cuenta de libro mayor

En esta parte se procede a clasificar de acuerdo al tipo de bien como son: Muebles de oficina, equipos de oficina, equipos de computación, vehículos, línea, redes e instalaciones, edificios, etc.

#### 5. Subcuenta

En este concepto se clasifica el tipo de bien que sea inventariado, ejemplo: mesas, sillas, escritorios, teléfonos, calculadoras, etc.

#### 6. Auxiliar de la subcuenta

En este campo se utiliza el código de acuerdo a la especificación del bien, ejemplo: escritorio de madera, escritorio de metal, silla secretaria, silla fija, anaquel de madera, anaquel de metal, etc.

#### 7. Secuencial de cada bien

Cada tipo de bien tiene una numeración secuencial, la misma que nos permitirá conocer, anaqueles, teléfonos, computadores, etc. existentes en la Institución y que son propiedad de la misma.

#### 8. Estructura tipo de la codificación

NNN	Cuenta Contable Presupuestaria
N	Servicio
NN	Conjunto
NN	Subconjunto
NN	Artículo

#### 9. Grados de responsabilidad

Cada custodio/conservador de bienes es responsable de los mismos, tanto para su uso así como para la conservación; para lo cual se debe proceder a legalizar la correspondiente acta de entrega – recepción.

### **2.1.3 ADMINISTRACION Y CONTROL DE ACTIVOS FIJOS**

#### **2.1.3.1 Controles varios mediante formularios**

Nos permite mediante formularios poder llevar un control mediante formularios como por ejemplo:

1. Altas;
2. Bajas;

3. Traspasos: temporales y/o definitivos; y,
4. Devolución de bienes traspasados temporalmente.

### **2.1.3.2 Diseño de formularios**

Los formularios deben tener un formato especial, según las necesidades de la organización para poder elaborar los formularios, estos pueden tomar un formato donde consten:

1. Altas;
2. Bajas;
3. Traspasos;
4. Devolución de bienes traspasados temporalmente;
5. Controles individuales;
6. Reporte detallado de inventarios;
7. Reporte resumido de inventarios; y,
8. Registro para ser digitado.

### **2.1.3.3 Normas de Administración y Control de Bienes**

La utilización adecuada de los códigos asignados a cada bien será la base importante para el control de activos fijos.

La persona encargada de administrar, controlar y mantenerlos actualizados se maneja bajo los estatutos internos de la institución, y los custodios/conservadores de los bienes deberán responder ante esta persona.

#### **2.1.4 RUTINA DESCRIPTIVA DE ALTA DE BIENES**

Este proceso se lo realizará mediante la elaboración de una acta de entrega – recepción, con el fin de asignar y responsabilizar a cada custodio/conservador de los bienes que utilicen individualmente.

##### **2.1.4.1 Encargado Administrativo**

El encargado de administrar, controlar y mantener los activos actualizados debe realizar:

1. Control de los bienes entregados a cada una de las unidades administrativas u operativas;
2. Remite a Contabilidad un ejemplar de las actas individuales entregadas a los custodios/conservadores de bienes, debidamente legalizadas.
3. Conserva para su control, una copia de las actas de entrega – recepción clasificadas por custodio/conservador de bienes.

##### **2.1.4.2 Conservadores/custodios de bienes**

Tienen como deber cumplir con rutinas como:

1. Recibe del Tesorero una copia de la acta de entrega – recepción de los bienes que estarán bajo su control y responsabilidad;
2. Revisa el contenido de la acta;
3. Verifica físicamente la existencia de los bienes entregados a su cargo;
4. Firma la respectiva acta en tres ejemplares;
5. Archiva el duplicado de la acta, para control y administración de los bienes.

### **2.1.4.3 Contabilidad**

Tiene como deber cumplir:

1. Recibe del tesorero copia de la acta legalizada por éste y el conservador/custodio de los bienes;
2. Registra contablemente en el sistema informático el alta de los bienes;
3. Archiva copia del acta;
4. Periódicamente realizará monitoreo por muestreo sobre la existencia física de los bienes a cargo de los custodios (conservadores de inventarios); y,
5. Antes de proceder a la entrega de la respectiva liquidación de haberes de un custodio/conservador de bienes, cuando el mismo se

separe de la Institución, procederá a la verificación física de los bienes y en caso de faltantes o destrucción, se descontará el valor de los mismos, previa autorización por escrito del Encargado principal.

#### **2.1.4.4 Procesamiento automatizado de datos – PAD**

1. Mantiene actualizadas las formas para digitar los inventarios bajo las modalidades de altas, traspasos temporales, traspasos definitivos y bajas; y,
2. Sobre los listados que prepare esta unidad, mantendrá los respectivos archivos.

#### **2.1.5 RUTINA DESCRIPTIVA DE BAJA DE BIENES**

Se lo realizará utilizando el formulario de baja de bienes, con el fin que cada custodio notifique al Encargado General, los bienes en mal estado y bajo las instrucciones por escrito de éste, se procederá a dar de baja.

#### **2.1.6 NORMAS PARA ADMINISTRACION Y CONTROL DE BIENES**

##### **2.1.6.1 Ámbito de aplicación**

1. Las presentes Normas se aplicarán al uso, operación, administración y control de los bienes que constituyen Activos Fijos.
2. Para efectos de su aplicación se definen a los siguientes términos:

*INSTITUCION:* ESCUELA POLITECNICA DEL EJERCITO

*DEPENDENCIA:* Se refiere a una Unidad Administrativa en general, sea ésta área o departamento.

*ADMINISTRACION Y CONTROL DE BIENES:* Dependencia encargada de la administración y control de los bienes inventariables o Activos Fijos.

*CONSERVADOR/CUSTODIO DE INVENTARIOS:* Persona o funcionario, responsable de la existencia y conservación de los bienes pertenecientes a un área o departamento, cuyo nivel o alcance está definido en el organigrama estructural de la Institución.

*ALTA:* Incorporación a los inventarios de la Institución por primera ocasión a través de una acta de entrega – recepción.

*BAJA:* Exclusión de los inventarios y registros contables de la Institución, de los bienes que no se encuentran en condiciones de seguir prestando sus servicios por mal estado de conservación, obsolescencia, pérdida o destrucción total o parcial, cuya constancia figurará en el formulario respectivo de baja, previa autorización del Directorio.

*TRASPASO:* Movilización de un bien de una dependencia a otra en forma temporal o definitiva. En el primer caso se controlará a través de un formulario de traspaso temporal. En el segundo, deberá realizarse un movimiento en los registros contables y en el sistema informático de los inventarios de bienes.

Para ser considerado temporal, el traspaso de los bienes no excederá de treinta días; transcurridos éstos, se considerará definitivo.

*DONACION:* Transferencia gratuita de dominio de un bien de la Institución, a otra persona jurídica o natural totalmente diferente de aquella, con autorización de las autoridades.

*BIEN(ES):* Concepto que denota existencia de propiedad de la Institución, destinado al servicio de la misma; y, que desde el punto de vista contable es tratado como activo fijo, sujeto en su administración y control bajo las disposiciones de estas normas que sean establecidas.

## **2.2 METODOLOGÍA DE PROGRAMACIÓN EXTREMA (XP)**

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo



en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

### 2.2.1 Fases de la Metodología de Programación Extrema

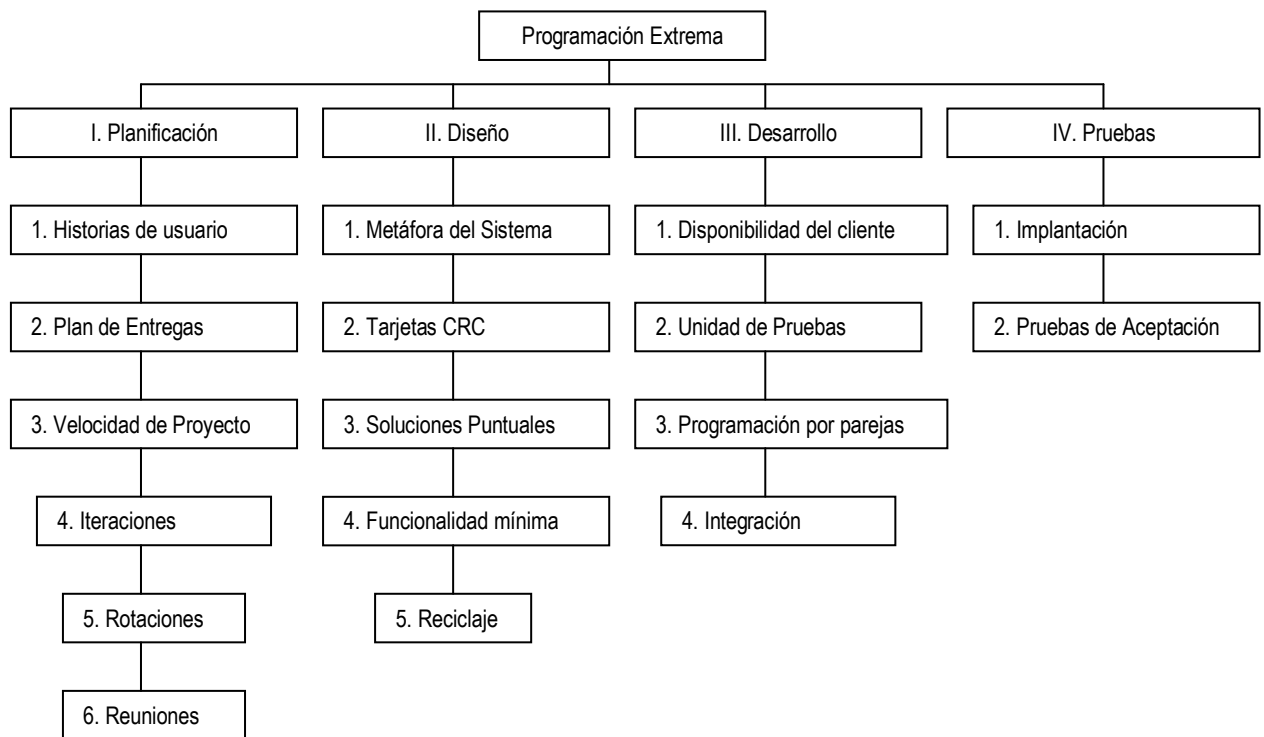


Figura 2.1 Fases de la Programación Extrema

### 2.2.1.1 Planificación

XP plantea la planificación como un permanente dialogo entre las partes, la empresarial (deseable) y la técnica (posible). Las personas del negocio necesitan determinar:

Ámbito: ¿Qué es lo que el software debe de resolver para que este genere valor?

Prioridad: ¿Qué debe ser hecho en primer lugar?

Composición de versiones: ¿Cuánto es necesario hacer para saber si el negocio va mejor con software que sin el?.

Fechas de versiones: ¿Cuáles son las fechas en las cuales se presentaran cada una de las iteraciones?

Estimaciones: ¿Cuanto tiempo lleva implementar una característica?

Consecuencias: Informar sobre las consecuencias de la toma de decisiones por parte del negocio.

Procesos: ¿Cómo se organiza el trabajo y el equipo?

Programación detallada: Dentro de una versión ¿Qué problemas se resolverán primero?

- Historia de usuarios

Es la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo

suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas

- Plan de Entregas

Para ello existen una serie de reglas que hay que seguir para que las tres partes implicadas en este proceso (equipo de gestión, equipo de desarrollo y cliente) tengan voz y se sientan parte de la decisión tomada, que al fin y al cabo debe contentar a todos.

En la planificación iterativa se especifican las historias de los usuarios cuya implementación se considera primordial y se añaden aquéllas que no han pasado las pruebas de aceptación de anteriores iteraciones. La planificación de una iteración también hace uso de tarjetas en las que se escribirán tareas, que durarán entre uno y tres días (la duración la deben decidir los propios desarrolladores).

- Velocidad de Proyecto

Establece la manera en la que el proyecto va avanzando, de tal forma que no se debe implementar más funcionalidad que la requerida en cada historia de usuario.

- Iteraciones

Permite establecer que las historias de usuario pueden ser desarrolladas de una a tres semanas de tiempo de programación (para no superar el tamaño de una iteración). Las historias de usuario son descompuestas

en tareas de programación (task card) y asignadas a los programadores para ser implementadas durante una iteración.

- Rotaciones

El código con propiedad compartida es una de las características de esta metodología. Nadie es el propietario de nada, todos son el propietario de todo. Este método difiere en mucho a los métodos tradicionales en los que un simple programador posee un conjunto de código. Los defensores de XP argumentan que mientras haya más gente trabajando en una pieza, menos errores aparecerán.

- Reuniones

Deben ser una constante las reuniones periódicas durante esta fase de planificación. Estas pueden ser a diario, con todo el equipo de desarrollo para identificar problemas, proponer soluciones y señalar aquellos puntos a los que se les ha de dar más importancia por su dificultad o por su punto crítico.

### **2.2.1.2 Diseño**

- Metáfora

El principal objetivo de la metáfora es mejorar la comunicación entre todos los integrantes del equipo al crear una visión global y común del sistema que se pretende desarrollar. La metáfora debe estar expresada en términos conocidos para los integrantes del grupo, por ejemplo

comparando lo que se va a desarrollar con algo que se puede encontrar en la vida real. Aunque también se incluye información sobre las principales clases y patrones que se usarán en el sistema.

Un apoyo a la metáfora a lo largo del proyecto es una correcta elección y comunicación de los nombres que se escojan durante el proyecto para los módulos, sistemas, clases, métodos, etc. Nombres bien puestos implican claridad, reusabilidad y simplicidad tres conceptos a los que XP otorga una gran importancia.

- Tarjetas CRC (Clases, Responsabilidades y Colaboración)

Aunque en general el diseño es realizado por los propios desarrolladores en ocasiones se reúnen aquellos con más experiencia o incluso se involucra al cliente para diseñar las partes más complejas. En estas reuniones se emplean un tipo de tarjetas denominadas CRC (Class, Responsibilities and Collaboration - Clases, Responsabilidades y Colaboración) cuyo objetivo es facilitar la comunicación y documentar los resultados. Para cada clase identificada se rellenará una tarjeta de este tipo y se especificará su finalidad así como otras clases con las que interaccione. Las tarjetas CRC son una buena forma de cambiar de la programación estructurada a una filosofía orientada a objetos. Aunque los grandes gurús de la programación extrema sostienen que bien hechas suelen hacer el diseño obvio, recomiendan hacer sesiones CRC en caso de que el sistema que se pretenda crear tenga un grado de complejidad grande. Este tipo de sesiones es una simulación, tarjetas

CRC en mano, de las interacciones entre los diferentes objetos que puede realizar el equipo de desarrollo.

Clase	
Responsabilidad	Colaboración

Tarjeta 2.1 Tarjeta CRC

- Soluciones Puntuales

Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más 3 meses.

- Refactorización

Permite a los equipos de programadores XP mejorar el diseño del sistema a través de todo el proceso de desarrollo. Los programadores evalúan continuamente el diseño y recodifican lo necesario. La finalidad es mantener un sistema enfocado a proveer el valor de negocio mediante la minimización del código duplicado y/o ineficiente.

- Reciclaje

Reaprovechar cuando sea posible. Cuando se elimina redundancia, se elimina funcionalidad inútil, y fortalecemos antiguos diseños, estamos reciclando código. El reciclaje, dentro del ciclo de vida de un proyecto, ahorra tiempo e incrementa la calidad. El reciclaje implicará mantener el código limpio y fácil de comprender, modificar y ampliar. Esto puede resultar un poco costoso al principio, pero resulta fundamental a la hora de realizar diseños futuros.

### **2.2.1.3 Desarrollo**

Esta etapa debe reunir las siguientes características y cualidades:

- Cliente In-situ

Se le dará poder para determinar los requerimientos, definir la funcionalidad, señalar las prioridades y responder las preguntas de los programadores. Esta fuerte interacción cara a cara con el programador disminuye el tiempo de comunicación y la cantidad de documentación, junto con los altos costes de su creación y mantenimiento. Este representante del cliente estará con el equipo de trabajo durante toda la realización del proyecto.

- Estándares de codificación

Define la propiedad del código compartido así como las reglas para escribir y documentar el código y la comunicación entre diferentes piezas de código desarrolladas por diferentes equipos. Los programadores las han de seguir de tal manera que el código en el sistema se vea como si hubiera estado escrito por una sola persona.

- Pruebas Unitarias

Las pruebas unitarias se implementan a la vez que el código de producción. De hecho cada vez que se va a implementar una pequeña parte se escribe una prueba sencilla y luego el código suficiente para que la pase. Cuando la haya pasado se repite el proceso con la siguiente parte. Aunque intuitivamente esto parezca contraproducente, a la larga hará que la generación de código se acelere. Los creadores de la programación extrema argumentan que encontrar un error puede llegar a ser cien veces más caro que realizar las pruebas unitarias. La idea, en definitiva, se resumen en la siguiente frase: "Todo código que pueda fallar debe tener una prueba". Además, hay que tener en cuenta que se hacen una vez y luego se pueden reutilizar multitud de veces, incluso por otros desarrolladores que desconocen los entresijos de esa parte o de todo el sistema.



- Programación por parejas

El principio más radical y en el que la mayoría de gerentes de desarrollo pone sus dudas. Requiere que todos los programadores XP escriban su código en parejas, compartiendo una sola máquina. De acuerdo con los experimentos, este principio puede producir aplicaciones más buenas, de manera consistente, a iguales o menores costes. Aunque el *pair-programming* puede no ser para todo el mundo

- Integración continua

El código se debe integrar como mínimo una vez al día, y realizar las pruebas sobre la totalidad del sistema. Una pareja de programadores se encargara de integrar todo el código en una maquina y realizar todas las pruebas hasta que estas funcionen al 100%.

- Propiedad colectiva

Cualquiera que crea que puede aportar valor al código en cualquier parcela puede hacerlo, ningún miembro del equipo es propietario del código. Si alguien quiere hacer cambios en el código puede hacerlo. Si hacemos el código propietario, y necesitamos de su autor para que lo cambie entonces estaremos alejándonos cada vez mas de la comprensión del problema, si necesitamos un cambio sobre una parte del código lo hacemos y punto. XP propone un propiedad colectiva sobre

el código nadie conoce cada parte igual de bien pero todos conoce algo sobre cada parte, esto nos preparará para la sustitución no traumática de cada miembro del equipo.

#### **2.2.1.4 Pruebas**

Las pruebas unitarias se deberían implementar con mayor frecuencia. Aunque se puede considerar que es una técnica ya existente en los años 70, la programación extrema puede hacer que se incremente su uso en los proyectos de software libre y, probablemente, que sea su mayor aportación a la misma. Las pruebas unitarias no tienen por qué hacerse estrictamente antes de codificar; al fin y al cabo es mejor tener pruebas, aunque sea con posterioridad, que no tenerlas.

Cuando un programador tiene que modificar código que ha sido desarrollado por otra persona se corre un alto riesgo de que introduzca algún error (bug). Las pruebas unitarias reducen este riesgo dado que avisarán al instante si algo deja de funcionar. Naturalmente para ello debe adquirirse el hábito de ejecutar todas las pruebas del sistema tras hacer un cambio. En los proyectos de software libre, la reducción de este factor de riesgo es muy útil, dado que es muy habitual modificar código escrito por otros.

Es probable que la introducción de las pruebas unitarias conlleve un cambio de filosofía en el mundo del software libre. De la búsqueda heroica de errores en la que se hace tanto énfasis, se debe pasar a la implantación de buenas maneras que redunden en código menos proclive a tenerlos. En definitiva, se debe tener a desarrolladores con talento creando código (que es

probablemente lo que mejor hagan y más les estimule) en vez de repasando código erróneo de otros.

En un futuro, esperemos que este tipo de pruebas puedan estar integradas en sistemas de control de versiones como el CVS. Quizás haya que esperar a que Subversion, la herramienta de siguiente generación de control de versiones que todavía se encuentra en fase de desarrollo, las implemente. De esta forma, las pruebas unitarias formarían parte del sistema de control de versiones.

No nos debemos olvidar por otro lado de las pruebas de aceptación. La mayoría de los proyectos de software libre carecen en sus páginas web de información de hacia dónde va el proyecto: las funcionalidades que serán añadidas en un futuro próximo, los cambios que se van a hacer, etc. Una buena forma de resolver esto y abrir el desarrollo a más desarrolladores podría ser la creación de pruebas de aceptación previas a la implementación de la siguiente operación que podrán ser comprobadas al final de la misma, tal y como ocurren en los ciclos iterativos de la programación extrema. Esto también puede utilizarse para saber las funcionalidades que son prioritarias para los usuarios.

## ***2.3 Plataforma .Net***

### **2.3.1 Plataforma .Net**

.NET es la plataforma de Microsoft para servicios Web XML, la siguiente generación de software que conecta nuestro mundo de información, dispositivos y personas de una manera unificada y personalizada.

La Plataforma .NET permite la creación y uso como servicios de aplicaciones, procesos y sitios Web basados en XML, que compartan y combinen información y funcionalidad entre ellos por diseño, en cualquier plataforma o dispositivo inteligente, para proveer soluciones a la medida para empresas e individuos.

Por esto, .Net es la estrategia de Microsoft para conectar sistemas, información y dispositivos a través de los servicios Web (Web Services) con el fin de que las personas puedan colaborar y comunicarse más eficazmente, proporcionando una capacidad de construcción más rápida, manejo y uso de la conexión, obteniendo una solución segura y reforzada a través del uso de los servicios Web.

La plataforma está orientada a Internet y los entornos distribuidos. Por eso hace especial hincapié en la distribución de código en diferentes ubicaciones y a la comunicación de estas partes de código mediante objetos remotos o servicios Web entre otras tecnologías.

### **2.3.1.1 .Net Framework**

Es un conjunto de servicios de programación diseñados para simplificar el desarrollo de aplicaciones en el entorno altamente distribuido de Internet. El **.NET Framework** se instala como un componente aparte en Windows 2000, mientras que Windows XP y las futuras versiones de Windows lo incorporan

directamente al sistema operativo. Como por ejemplo Windows Server 2003 o Windows .NET CE.

Los componentes del .NET Framework proveen los "ladrillos" necesarios para construir las aplicaciones Web, los servicios Web y cualquier otra aplicación dentro de Visual Studio .NET.

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables.

Los principales componentes de este entorno son:

- Lenguajes de compilación
- Biblioteca de clases de .Net
- CLR (Common Language Runtime)

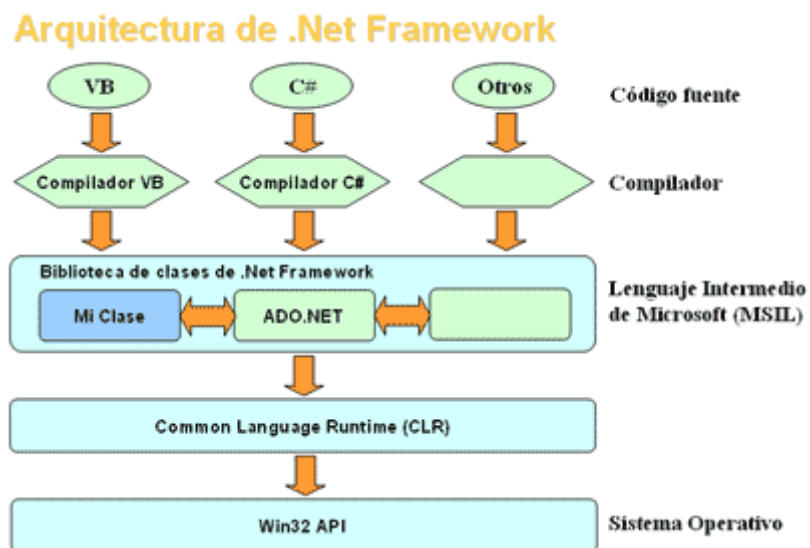


Figura 2.2 Common Language Runtime (CLR)

El CLR es el verdadero núcleo del Framework de .Net, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios que ofrece el sistema operativo estándar Win32.

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un mismo código, denominado código intermedio (MSIL, Microsoft Intermediate Lenguaje). Para generar dicho código el compilador se basa en el Common Language Specification (CLS) que determina las reglas necesarias para crear código MSIL compatible con el CLR.

De esta forma, indistintamente de la herramienta de desarrollo utilizada y del lenguaje elegido, el código generado es siempre el mismo, ya que el MSIL es el único lenguaje que entiende directamente el CLR. Este código es transparente al desarrollo de la aplicación ya que lo genera automáticamente el compilador.

Sin embargo, el código generado en MSIL no es código máquina y por tanto no puede ejecutarse directamente. Se necesita un segundo paso en el que una herramienta denominada compilador JIT (Just-In-Time) genera el código máquina real que se ejecuta en la plataforma que tenga la computadora.

De esta forma se consigue con .Net cierta independencia de la plataforma, ya que cada plataforma puede tener su compilador JIT y crear su propio código máquina a partir del código MSIL.

La compilación JIT la realiza el CLR a medida que se invocan los métodos en el programa y, el código ejecutable obtenido, se almacena en la memoria caché de la computadora, siendo recompilado sólo cuando se produce algún cambio en el código fuente.

- Biblioteca de clases de .Net

Cuando se está programando una aplicación muchas veces se necesitan realizar acciones como manipulación de archivos, acceso a datos, conocer el estado del sistema, implementar seguridad, etc. El Framework organiza toda la funcionalidad del sistema operativo en un espacio de nombres jerárquico de forma que a la hora de programar resulta bastante sencillo encontrar lo que se necesita.

Para ello, el Framework posee un sistema de tipos universal, denominado Common Type System (CTS). Este sistema permite que el programador pueda interactuar los tipos que se incluyen en el propio Framework (biblioteca de clases de .Net) con los creados por él mismo (clases). De esta forma se aprovechan las ventajas propias de la programación orientada a objetos, como la herencia de clases predefinidas para crear nuevas clases, o el polimorfismo de clases para modificar o ampliar funcionalidades de clases ya existentes.

## Biblioteca de clases de .NET Framework

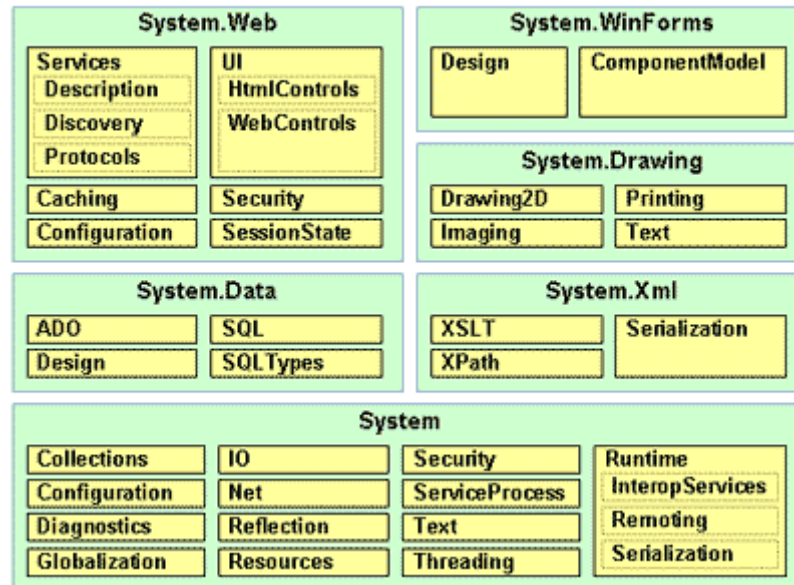


Figura 2.3.: Biblioteca de Clases de .Net Framework

La biblioteca de clases de .Net Framework incluye, entre otros, tres componentes clave:

- **ASP.NET:** para construir aplicaciones y servicios Web.
- **WINDOWS FORMS:** para desarrollar interfaces de usuario.
- **ADO.NET:** para conectar las aplicaciones a bases de datos.

La forma de organizar la biblioteca de clases de .Net dentro del código es a través de los espacios de nombres (namespaces), donde cada clase está organizada en espacios de nombres según su funcionalidad. Por ejemplo, para manejar ficheros se utiliza el espacio de nombres System.IO y si lo que se quiere es obtener información de una fuente de datos se utilizará el espacio de nombres System.Data.



La principal ventaja de los espacios de nombres de .Net es que de esta forma se tiene toda la biblioteca de clases de .Net centralizada bajo el mismo espacio de nombres (System). Además, desde cualquier lenguaje se usa la misma sintaxis de invocación, ya que a todos los lenguajes se aplica la misma biblioteca de clases.

- Ensamblados

Uno de los mayores problemas de las aplicaciones actuales es que en muchos casos tienen que tratar con diferentes archivos binarios (DLL's), elementos de registro, conectividad abierta a bases de datos (ODBC), etc.

Para solucionarlo el Framework de .Net maneja un nuevo concepto denominado ensamblado. Los ensamblados son ficheros con forma de EXE o DLL que contienen toda la funcionalidad de la aplicación de forma encapsulada. Por tanto la solución al problema puede ser tan fácil como copiar todos los ensamblados en el directorio de la aplicación.

Con los ensamblados ya no es necesario registrar los componentes de la aplicación. Esto se debe a que los ensamblados almacenan dentro de si mismos toda la información necesaria en lo que se denomina el manifiesto del ensamblado. El manifiesto recoge todos los métodos y propiedades en forma de meta-datos junto con otra información descriptiva, como permisos, dependencias, etc.

Para gestionar el uso que hacen las aplicaciones de los ensamblados .Net utiliza la llamada caché global de ensamblados (GAC, Global Assembly Cache). Así, .Net Framework puede albergar en el GAC los ensamblados que puedan ser usados por varias aplicaciones e incluso distintas versiones de un mismo ensamblado, algo que no era posible con el anterior modelo COM.

### **2.3.1.2 .Net Compact Framework**

Microsoft ha desarrollado .NET Compact Framework con un claro objetivo: la creación de aplicaciones. Nos referimos a aplicaciones capaces de mostrar, recopilar, procesar y enviar datos; el tipo de aplicación que justifica que los usuarios decidan llevar encima un dispositivo. Aunque normalmente estas aplicaciones tienen una interfaz, no siempre es necesario. Los datos con los que estas aplicaciones trabajan pueden ser locales, remotos o tal vez una combinación de ambos.

.NET Compact Framework simplifica el desarrollo de aplicaciones para dispositivos inteligentes. Actualmente, esto incluye a los dispositivos Pocket PC, Pocket PC 2002, Pocket PC Phone Edition y otros dispositivos que ejecuten Windows CE.NET 4.1 o posterior.

Necesitará Visual Studio .NET 2003 para crear aplicaciones destinadas a .NET Compact Framework. Puede crear aplicaciones utilizando Visual C# .NET, Visual Basic .NET o ambos.

.NET Compact Framework tiene dos componentes principales: el tiempo de ejecución en lenguaje común y la biblioteca de clases de .NET Compact Framework.

- El tiempo de ejecución es la base de .NET Compact Framework, ya que se encarga de administrar el código en el momento de la ejecución, proporcionando servicios esenciales como la administración de la memoria y de los subprocesos, al mismo tiempo que garantiza la seguridad y la precisión. Si el código está destinado al tiempo de ejecución se denomina código administrado, si no lo está, como ocurre con eMbedded Visual C++, se denomina código no administrado o nativo.
- La biblioteca de clases de .NET Compact Framework es una colección de clases reutilizables que se pueden utilizar para desarrollar aplicaciones de manera fácil y rápida. Este marco se ha diseñado pensando en la portabilidad, tanto para plataformas Microsoft como de otros fabricantes. ¿Qué significa esto? Sencillamente que las técnicas de codificación y las aplicaciones creadas hoy en un Pocket PC se pueden ejecutar en otras plataformas, como un teléfono móvil o un PDA de otro fabricante, si se ha creado una versión de .NET Compact Framework para dicha plataforma.

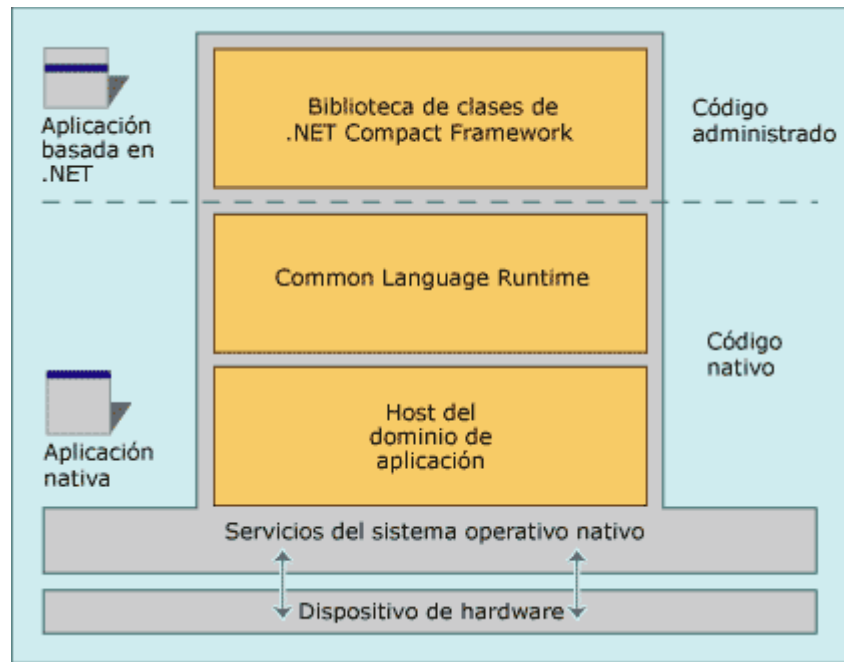


Figura 2.4: Biblioteca de Clases

## 2.3.2 Capa de Interfaz

### 2.3.2.1 Visual C#.net

Visual C# .NET proporciona a los programadores un lenguaje moderno orientado a componentes con el que pueden crear con rapidez soluciones interesantes controladas por datos. Gracias a la rapidez de diseño, programación e implementación de servicios Web XML, las soluciones controladas por datos de alto rendimiento y los diseñadores visuales de servidor, Visual C# .NET ofrece funcionalidad superior para optimizar procesos empresariales. Permite a los programadores generar soluciones para una gama amplísima de clientes, incluidas aplicaciones Web, aplicaciones basadas en Windows Forms y dispositivos cliente ligeros e inteligentes. Además, con un lenguaje de programación elegante y moderno, los programadores de C#

pueden aprovechar sus conocimientos de C++ y Java para disfrutar de una experiencia satisfactoria con la plataforma Microsoft .NET.

La herramienta es utilizada de la misma forma tanto para el desarrollo en PC de escritorio como para desarrollo móvil.

C# es la versión de C y C++ para .Net es una recopilación de lo mas relevante de las dos herramientas es muy parecida en su sintaxis a java la que le hace muy robusta en tiempo de desarrollo

### **2.3.3 Capa Lógica**

#### **2.3.3.1 Web Services**

Son aplicaciones pequeñas, reusables, que permiten a las maquinas que poseen muchos sistemas operativos diferentes, trabajar juntas intercambiando mensajes. Los Web Services son basados en protocolos que incluyen el lenguaje XML (Extensible Markup Language), SOAP (Simple Object Access Protocol) y WSDL (Web Services Description Language). Estos protocolos ayudan a las computadoras a trabajar juntas de una plataforma a otra.

Web Services no solo sirve para conectar sistemas, sino que además ayuda a los usuarios a conectarse y obtener la información que necesita.

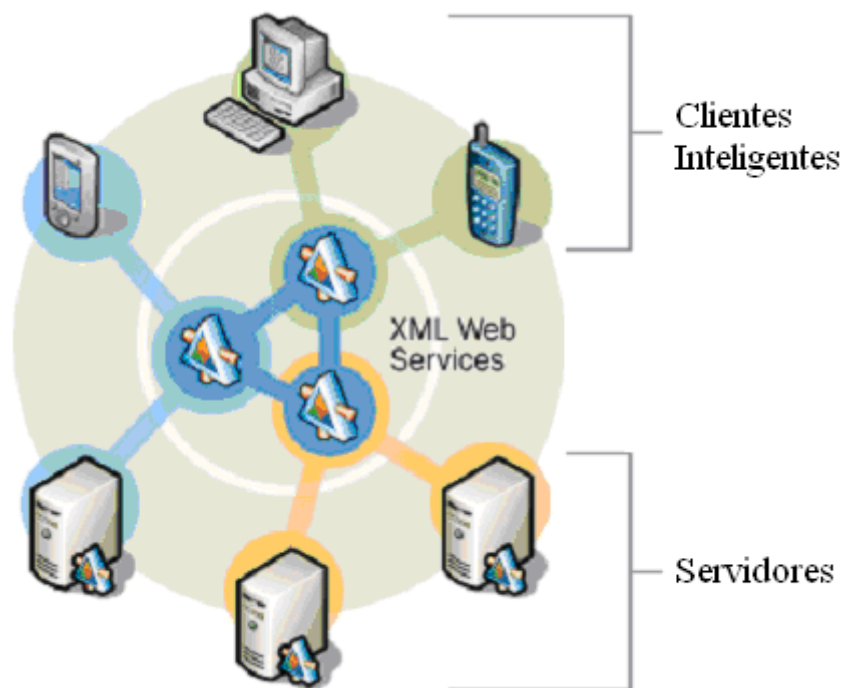


Figura 2.5: Esquema de Web Services

Web Services nos permite conectarnos como clientes inteligentes de diferentes dispositivos con los equipos servidores a través de la Web por otros sistemas, los cuales pueden interactuar con el Web Services utilizando mensajes XML y transportados por protocolos de Internet como el SOAP.

### 2.3.3.2 Protocolo Soap

SOAP (Simple Object Access Protocol) es un protocolo ligero basado en XML, para el intercambio de información en un ambiente descentralizado y distribuido. Este protocolo fue originalmente desarrollado por Microsoft y luego por IBM basándose en protocolos de formato de serialización. El SOAP permite la intercomunicación entre objetos de cualquier tipo - sobre cualquier plataforma, en cualquier lenguaje.

Dado que el XML se basa en textos y es legible por personas, es ideal como marco de referencia de transporte para otros servicios de Web débilmente acoplados. El fondo de todo esto es que las transacciones automatizadas incrementan la productividad, reducen costos y mejoran los servicios. La presencia de estándares en la red hace que sean posibles las transacciones automatizadas.

SOAP es una tecnología derivada de estándares anteriores basados en XML (XML-RPC), y de algún modo, apunta hacia un estándar emergente ebXML. El establecimiento de ebXML se encuentra en proceso de desarrollo, intentando proveer una definición amplia de mensajes compartidos de negocios entre socios comerciales. SOAP es más modesto en alcance y menos complejo de implementar.

En principio una petición/respuesta de SOAP puede usar cualquier protocolo de transporte (HTTP, SMTP, etc.). Las cabeceras difieren de protocolo a protocolo, pero el contenido de SOAP es el mismo. La petición de SOAP en XML consiste de tres partes:

*Envelope* - El sobre de SOAP define un marco de referencia general para expresar que es lo que contiene el mensaje, quién debe recibirlo, y si es opcional u obligatorio.

*Codificación* - Las reglas de codificación de SOAP definen un mecanismo de estandarización que puede ser utilizado para intercambiar instancias de tipos de datos definidos por la aplicación.

*Estándar* - La representación RPC de SOAP define una convención que puede ser utilizada para representar los llamados y respuestas de procedimientos remotos.

Dado que SOAP utiliza XML para codificar mensajes, es relativamente sencillo procesar los mensajes en cada paso del proceso. Además, la facilidad de depuración de mensajes SOAP permite la convergencia rápida de las diversas implementaciones de SOAP, la cual es importante en la interoperabilidad a gran escala.

Un mensaje SOAP es un documento XML que consta de los siguientes elementos:

*Envelope* (sobre) que define el contenido del mensaje

*Header* (cabecera) que es opcional y que contiene meta información referente al mensaje

*Body* (cuerpo) que contiene la información de la llamada y de la respuesta



Ejemplo de comunicación (sin cabeceras HTTP):

Supongamos que tenemos una sencilla aplicación en la que hay una clase CheckServer que tiene un método llamado pingServer() que recibe como parámetro una cadena de caracteres que será el nombre del servidor a quien enviaremos la petición de PING.

El mensaje generado por el cliente sería:

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <pingServer>
      <server>activalink.net</server>
    </pingServer>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

El mensaje que devolvería el servidor sería:

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <pingServerResponse>
      <return>PONG</return>
    </pingServerResponse>
  </SOAP-ENV:Body>
```

</SOAP-ENV:Envelope>

Como vemos, la principal diferencia entre ambos mensajes (XML) es que la respuesta se encuentra en una etiqueta que se llama como el método invocado más la palabra Response (indicando que es la respuesta).

Ejemplo de comunicación completa (con cabeceras HTTP):

El mensaje generado por el cliente sería:

POST /InStock HTTP/1.1

Host: www.activalink.net

Content-Type: appliation/soap; charset=utf-8

<?xml version="1.0"?>

<soap:Envelope

xmlns:soap="http://www.w3.org/2001/12/soap-envelope"

soap:encondingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.activalink.net/stock">

<m:GetStockPrice>

<m:StockName>Tornillos</m:StockName>

</m:GetStockPrice>

</soap:Body>

</soap:Envelope>

El mensaje que devolvería el servidor sería:

HTTP/1.1 200 OK

Connection: close

Content-Type: application/soap; charset=utf-8

Date: Wed, 22 Oct 2003 08:09:04 GMT

```
<?xml version="1.0"?>
```

```
  <soap:Envelope
```

```
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
```

```
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
```

```
      <soap:Body
```

```
        xmlns:m="http://www.acticalink.net/stock">
```

```
          <m:GetStockPriceResponse>
```

```
            <m:Price>0.5</m:Price>
```

```
          </m:GetStockPriceResponse>
```

```
        </soap:Body>
```

```
      </soap:Envelope>
```

Cuando ocurre una excepción o error en una comunicación SOAP, este error se transmite en forma de elemento de fallo (Fault Element). Sólo puede aparecer un elemento de fallo y siempre en el cuerpo del mensaje.

### **2.3.3.3 Multifuncionalidad entre sistemas operativos**

Gracias a los protocolos que permiten las transacciones con el Web Services, pueden conectarse a los datos aplicaciones que pueden estar corriendo en diferentes sistemas operativos, conectados a la Web pueden realizar las conexiones respectivas con la base de datos deseada por medio del Web Services el cual es manejada por el lenguaje XML.

### **2.3.4 Capa de datos**

#### **2.3.4.1 Base de datos SQL Server**

SQL Server 2000 es la base de datos totalmente habilitada para Web. Además, se centra en lo que es escalabilidad y confiabilidad, que son críticas para el éxito de una base de datos empresarial. Una de sus principales características es la velocidad del procesamiento de transacciones.

Entre sus principales características tenemos:

- Es totalmente habilitado para la Web,
- Proporciona las capacidades necesarias de programación de base de datos basadas en estándares Web.
- Permite un fácil acceso a los datos a través de la Web.
- Proporciona las seguridades necesarias de cifrado para mantener la integridad de los datos almacenados en la base de datos.
- Permite la realización de respaldos automáticamente lo cual asegura la información de la empresa almacenada en la base de datos.

- Permite un rápido desarrollo y depuración, ya que permite optimizar y depurar consultas de manera interactiva, mover y transformar rápidamente datos provenientes de cualquier origen.
- Puede diseñar y codificar visualmente aplicaciones de base de datos con cualquier herramienta de Visual Studio.
- Permite una administración y optimización simplificadas, ya que permite realizar copias y respaldo de las bases de datos permaneciendo en línea hacia otros equipos.

#### 2.3.4.2 Base de datos SQL CE



Figura 2.6: SQL CE

SQL Server CE es la base de datos compacta que amplían capacidades de la administración de los datos de la empresa a los dispositivos móviles, ideal para los ambientes móviles y wireless. Está apoyado en el lenguaje estructurado de consulta SQL y proporcionando un modelo del desarrollo y un API constante con SQL Server.

El SQL CE está diseñado para integrarse con el .NET Compact Framework y Visual Studio .Net.

## **2.4 Dispositivos Móviles**

Los dispositivos móviles o también llamados computadoras de bolsillo (por su tamaño), ofrecen diferentes utilidades; algunas de ellas vienen directamente de fábrica, como agenda, calculadora, libreta de direcciones y anotador de ideas; y otras pueden ser bajadas de Internet, donde hay más de 10.000 aplicaciones orientadas a diferentes áreas de interés.

Existen algunas características que permiten identificar a un dispositivo móvil, como:

- son aparatos pequeños,
- con algunas capacidades de procesamiento,
- con conexión permanente o intermitente a una red,
- con memoria limitada,
- diseñados específicamente para una función, pero que pueden llevar a cabo otras más generales.
- Normalmente se asocian al uso individual de una persona, tanto en posesión como en operación, el cual puede adaptarlos a su gusto.
- La mayoría de estos aparatos pueden ser transportados en el bolsillo del propietario

A continuación vamos a conocer algunos de estos dispositivos:

### 2.4.1 Pocket PC

El PocketPC es el primer PDA (Personal Desktop Assistant) que ha llegado a un público masivo, consiguiendo su atractivo gracias a su reducido tamaño y por ende su portabilidad.

Las PDA's son de las herramientas más sofisticadas de uso personal y se dividen en dos categorías principalmente de tipo "hand-held" y tipo "PocketPC" y la diferencia de ambas es el tamaño, siendo el PocketPC el más pequeño.

La Figura muestra un dispositivo PocketPC



Figura 2.6: PDA – Falcon 4210

Sus principales funciones son:

- Organizador de tareas, permite organizar y recordar tareas pendientes
- Bloc de notas, se puede almacenar cualquier tipo de nota.
- Hoja de cálculo y editor de texto, da la facilidad de realizar cualquier tipo de informe desde este dispositivo.
- Permite reproducir archivos de audio y video

- Acceso a Internet, se puede almacenar correo electrónico y correr aplicaciones móviles desde este dispositivo.

### **2.4.2 Windows CE**

Es el sistema operativo para la nueva generación altamente conectada de dispositivos de 32bits. Esta plataforma altamente modular permite a los desarrolladores construir aplicaciones de manera flexible y confiable para la nueva generación de dispositivos que se integran con Windows e Internet.

### **2.5 Nunit testeador de pruebas unitarias**

Nunit 2.2 es un framework de software libre que fue creado bajo C# y sirve para realizar pruebas unitarias para .Net.

Una prueba unitaria es un procedimiento usado para validar que un módulo de código fuente funciona apropiadamente, el beneficio de las pruebas unitarias es que permite aislar segmentos del programa que pueden ser probados para verificar que funcionan correctamente.

Para poder utilizar Nunit 2.2 debe ser referenciada la dll, nunit.framework.dll dentro del proyecto que contiene el código a probar.

Los métodos que constan dentro de una clase que forma parte de las clases de pruebas no deben llevar parámetros.



Cada método que va a ser llevado a prueba debe tener la etiqueta [Test]

## CAPITULO III

### 3 DISEÑO DEL SISTEMA DE CONSTATAACION FISICA

#### 3.1 INTRODUCCION

El Diseño del sistema de constatación física se basa sobre dos aspectos fundamentales; esto es, el sistema móvil que constituye el marco general de la solución y el sistema para PC de escritorio que constituye el complemento de la solución global; estos dos sistemas siendo diferentes a la vez son coadyuvantes, porque tanto el uno como el otro se necesitan y se apoyan mutuamente para poder accionar y obtener el producto total; así por ejemplo:

Para el diseño de clientes inteligentes (smart clients) consideré como fundamental e importante los siguientes requisitos:

- Baja conectividad a servidores centrales de base de datos
- Cambio en la información relativamente bajo
- Experiencia del usuario muy limitada

Solo ellos me permitieron el desarrollo de aplicaciones para smart clients (POCKET PC), y garantizaron un gran porcentaje de éxito, dando soluciones que apoyan al sistema informático principal.

Es importante indicar que el diseño de soluciones smart clients ha hecho mas frecuente el uso de los WEB SERVICE, situación que ha impactado con gran

importancia por las facilidades que éstos componentes prestan al desarrollo de la solución.

Así mismo para el diseño de aplicaciones Windows (PC normales), se consideró básicamente la experiencia del usuario a fin de que la aplicación tenga el éxito deseado, con productos mejores y de gran versatilidad.

## **3.2 PLANIFICACIÓN**

Alcanzar el presente software ha sido el producto de un recorrido lógico y sistemático; el mismo que se cumplió a través de los siguientes aspectos:

- Historias de usuarios
- Plan de entregas
- Velocidad del proyecto
- Plan de iteraciones
- Rotación de personal
- Reuniones
- Diseño
- Desarrollo; y,
- Pruebas,

### **3.2.1 HISTORIAS DE USUARIO**

La metodología XP de la solución alcanzada otorga al usuario facilidad para que éste pueda escribir sus requerimientos, y una vez que lo escribió el

programador transforma en código, el mismo que procesa la petición y entrega la respuesta.

Esta metodología proporciona gran utilidad al usuario en razón de que ella responde a sus necesidades así por ejemplo: la extracción de información por dependencias; situación que evita un camino largo y engorroso si se utilizaría otra metodología.

La ventaja de esta metodología es que cada usuario puede escribir más de una historia, y no tiene importancia escribir en un orden, ya que el plan de entregas ordena sus requerimientos

Las historias que son procesadas en la solución actual fueron entregadas por el personal que labora en la ESPE

HISTORIA DE USUARIO N. 01							
Fecha	05/04/2005	Tipode Actividad	<input checked="" type="radio"/> Nueva <input type="radio"/> Corrección <input type="radio"/> Mejora			Prioridad Técnica	<input checked="" type="radio"/> Normal <input type="radio"/> Media <input type="radio"/> Alta
Referencia		Riesgo	<input type="radio"/> Ninguno <input type="radio"/> Medio <input type="radio"/> Alto			Prioridad Usuario	<input checked="" type="radio"/> Normal <input type="radio"/> Alta
Usuario	Ing. Sosa	Cargo	Responsable Bienes	Clasificación	DESCARGAR INFORMACION		
Descripción							
PARA FACILIDAD DEL PROCESO DE CONSTATACION FISICA DE BIENES SE DEBE PERMITIR GUARDAR LA INFORMACION DE LOS BIENES POR CADA UNA DE LAS DEPENDENCIAS							
SELECCIONAMOS UNA DE LAS DEPENDENCIAS Y PROCEDEMOS A GUARDAR EN LOS POCKET PC							
Notas							
Seguimiento							
Estado:	<input checked="" type="radio"/> Atendido <input type="radio"/> En Proceso <input type="radio"/> Finalizado	Comentarios:				Ultimo Seguimiento:	

Historia 3.1 Historia de Usuario 1

HISTORIA DE USUARIO N. 02						
Fecha	05/04/2005	Tipo de Actividad	<input checked="" type="radio"/> Nueva <input type="radio"/> Corrección <input type="radio"/> Mejora	Prioridad Técnica	<input checked="" type="radio"/> Normal <input type="radio"/> Media <input type="radio"/> Alta	
Referencia		Riesgo	<input checked="" type="radio"/> Ninguno <input type="radio"/> Medio <input type="radio"/> Alto	Prioridad Usuario	<input checked="" type="radio"/> Normal <input type="radio"/> Alta	
Usuario	Ing. Sosa	Cargo	Responsable Bienes	Clasificación	DESCARGAR INFORMACION	
Descripción						
LA INFORMACIÓN QUE DEBE GUARDARCE EN LOS POCKET PC POR CADA UNA DE LOS BIENES ES LA SIGUIENTE: DESCRIPCIÓN, CODIGO, MARCA, SERIE, MODELO, ESTADO, DEPENDENCIA, OBSERVACIÓN, FECHA DE ALTA, TIPO DE ACTIVO DE LA INSTITUCIÓN, RESPONSABLES DE LOS BIENES						
ESTA INFORMACION DEBE MOSTRARSE EL MOMENTO DE LA CONSTATAION FISICA DE BIENES						
Notas						
Seguimiento						
Estado:	<input checked="" type="radio"/> Atendido <input type="radio"/> En Proceso <input type="radio"/> Finalizado	Comentarios:			Ultimo Seguimiento:	

Historia 3.2 Historia de Usuario 2

HISTORIA DE USUARIO N. 03						
Fecha	05/04/2005	Tipo de Actividad	<input checked="" type="radio"/> Nueva <input type="radio"/> Corrección <input type="radio"/> Mejora	Prioridad Técnica	<input checked="" type="radio"/> Normal <input type="radio"/> Media <input type="radio"/> Alta	
Referencia		Riesgo	<input checked="" type="radio"/> Ninguno <input type="radio"/> Medio <input type="radio"/> Alto	Prioridad Usuario	<input checked="" type="radio"/> Normal <input type="radio"/> Alta	
Usuario	Ing. Sosa	Cargo	Responsable Bienes	Clasificación	CONSTATAION FISICA	
Descripción						
DEBEMOS TENER UNA OPCION LA CUAL NOS DEBE PERMITIR ALMACENAR LOS CAMBIOS EN LA INFORMACION DE CADA UNA DE LOS BIENES QUE VAYAMOS CONSTATANDO.						
Notas						
Seguimiento						
Estado:	<input checked="" type="radio"/> Atendido <input type="radio"/> En Proceso <input type="radio"/> Finalizado	Comentarios:			Ultimo Seguimiento:	

Historia 3.3 Historia de Usuario 3

HISTORIA DE USUARIO N. 04						
Fecha	05/04/2005	Tipo de Actividad	<input checked="" type="radio"/> Nueva <input type="radio"/> Corrección <input type="radio"/> Mejora	Prioridad Técnica	<input checked="" type="radio"/> Normal <input type="radio"/> Media <input type="radio"/> Alta	
Referencia		Riesgo	<input checked="" type="radio"/> Ninguno <input type="radio"/> Medio <input type="radio"/> Alto	Prioridad Usuario	<input checked="" type="radio"/> Normal <input type="radio"/> Alta	
Usuario	Ing. Sosa	Cargo	Responsable Bienes	Clasificación	SINCRONIZACION	
Descripción						
DEBE EXISTIR UNA OPCION QUE NOS PERMITA ACTUALIZAR LA INFORMACION EN NUESTRA BASE DE DATOS DE ACTIVOS FIJOS DE TODOS LOS BIENES QUE HAN SIDO MODIFICADOS DURANTE LA CONSTATAION FISICA.						
Notas						
Seguimiento						
Estado:	<input checked="" type="radio"/> Atendido <input type="radio"/> En Proceso <input type="radio"/> Finalizado	Comentarios:			Ultimo Seguimiento:	

Historia 3.4 Historia de Usuario 4

HISTORIA DE USUARIO N. 05									
Fecha	05/04/2005	Tipo de Actividad	<input checked="" type="radio"/> Nueva	<input type="radio"/> Corrección	<input type="radio"/> Mejora	Prioridad Técnica	<input checked="" type="radio"/> Normal	<input type="radio"/> Media	<input type="radio"/> Alta
Referencia		Riesgo	<input checked="" type="radio"/> Ninguno	<input type="radio"/> Medio	<input type="radio"/> Alto	Prioridad Usuario	<input checked="" type="radio"/> Normal	<input type="radio"/> Alta	
Usuario	Ing. Sosa	Cargo		Responsable Bienes		Clasificación	REPORTES		
Descripción									
ES IMPORTANTE PODER VISUALIZAR LA INFORMACIÓN DE LOS BIENES Y NECESITAMOS QUE LOS REPORTES SE NOS MUESTREN POR:									
USUARIOS									
Notas									
Seguimiento									
Estado:	<input checked="" type="radio"/> Atendido	Comentarios:				Ultimo Seguimiento:			
	<input type="radio"/> En Proceso								
	<input type="radio"/> Finalizado								

Historia 3.5 Historia de Usuario 5

HISTORIA DE USUARIO N. 06									
Fecha	05/04/2005	Tipo de Actividad	<input checked="" type="radio"/> Nueva	<input type="radio"/> Corrección	<input type="radio"/> Mejora	Prioridad Técnica	<input checked="" type="radio"/> Normal	<input type="radio"/> Media	<input type="radio"/> Alta
Referencia		Riesgo	<input checked="" type="radio"/> Ninguno	<input type="radio"/> Medio	<input type="radio"/> Alto	Prioridad Usuario	<input checked="" type="radio"/> Normal	<input type="radio"/> Alta	
Usuario	Ing. Sosa	Cargo		Responsable Bienes		Clasificación	SEGURIDADES		
Descripción									
DEBE EXISTIR LAS OPCIONES DE CREAR USUARIOS, PERFILES Y PODER ASIGNAR UN PERFIL DETERMINADO A CADA UNO DE LOS USUARIOS DE LA APLICACIÓN.									
Notas									
Seguimiento									
Estado:	<input checked="" type="radio"/> Atendido	Comentarios:				Ultimo Seguimiento:			
	<input type="radio"/> En Proceso								
	<input type="radio"/> Finalizado								

Historia 3.6 Historia de Usuario 6

HISTORIA DE USUARIO N. 07									
Fecha	05/04/2005	Tipo de Actividad	<input checked="" type="radio"/> Nueva	<input type="radio"/> Corrección	<input type="radio"/> Mejora	Prioridad Técnica	<input checked="" type="radio"/> Normal	<input type="radio"/> Media	<input type="radio"/> Alta
Referencia		Riesgo	<input checked="" type="radio"/> Ninguno	<input type="radio"/> Medio	<input type="radio"/> Alto	Prioridad Usuario	<input checked="" type="radio"/> Normal	<input type="radio"/> Alta	
Usuario	Ing. Sosa	Cargo		Responsable Bienes		Clasificación	SEGURIDADES		
Descripción									
DEBEMOS MANEJAR UNA AUDITORIA DE LO QUE SE REALIZA SOBRE LA INFORMACIÓN DE LOS BIENES ASI COMO DE QUIEN REALIZA ACCIONES SOBRE LA INFORMACIÓN DE LOS BIENES.									
Notas									
Seguimiento									
Estado:	<input checked="" type="radio"/> Atendido	Comentarios:				Ultimo Seguimiento:			
	<input type="radio"/> En Proceso								
	<input type="radio"/> Finalizado								

Historia 3.7 Historia de Usuario 7

HISTORIA DE USUARIO N. 08						
Fecha	05/04/2005	Tipode Actividad	<input checked="" type="radio"/> Nueva <input type="radio"/> Corrección <input type="radio"/> Mejora	Prioridad Técnica	<input checked="" type="radio"/> Normal <input type="radio"/> Media <input type="radio"/> Alta	
Referencia		Riesgo	<input checked="" type="radio"/> Ninguno <input type="radio"/> Medio <input type="radio"/> Alto	Prioridad Usuario	<input checked="" type="radio"/> Normal <input type="radio"/> Alta	
Usuario	Ing. Sosa	Cargo	Responsable Bienes	Clasificación	CONSTATACION FÍSICA	
Descripción						
DEBE INDICARSE SI UN BIEN YA FUE CONSTATADO, NO ES UNA LIMITANTE PARA VOLVER HA ACTUALIZAR						
DE SER EL CASO, NO PERMITIR QUE UN BIEN CONSULTADO SE CONSIDERE EN OTRA DEPENDENCIA						
Notas						
Seguimiento						
Estado:	<input checked="" type="radio"/> Atendido	Comentarios:			Ultimo Seguimiento:	
	<input type="radio"/> En Proceso					
	<input type="radio"/> Finalizado					

Historia 3.8 Historia de Usuario 8

HISTORIA DE USUARIO N. 09						
Fecha	05/04/2005	Tipode Actividad	<input checked="" type="radio"/> Nueva <input type="radio"/> Corrección <input type="radio"/> Mejora	Prioridad Técnica	<input checked="" type="radio"/> Normal <input type="radio"/> Media <input type="radio"/> Alta	
Referencia		Riesgo	<input checked="" type="radio"/> Ninguno <input type="radio"/> Medio <input type="radio"/> Alto	Prioridad Usuario	<input checked="" type="radio"/> Normal <input type="radio"/> Alta	
Usuario	Ing. Sosa	Cargo	Responsable Bienes	Clasificación	CONSTATACIÓN FÍSICA	
Descripción						
SE DEBE DESCARGAR LAS SIGUIENTES TABLAS QUE CONTIENEN PARAMETROS:						
ESTADOS, EMPLEADOS Y DEPENDENCIAS						
Notas						
Seguimiento						
Estado:	<input checked="" type="radio"/> Atendido	Comentarios:			Ultimo Seguimiento:	
	<input type="radio"/> En Proceso					
	<input type="radio"/> Finalizado					

Historia 3.9 Historia de Usuario 9

### 3.2.2 PLAN DE ENTREGAS

Una vez obtenido las historias, y determinado la necesidad en el orden de realización, como desarrollador ubique los tiempos de las entregas, a fin de estructura el plan de iteraciones.

El tiempo que tomó realizar todas las historias de usuario de la solución fue de 50 días laborables, conforme se demuestra en la figura 3.1





Las iteraciones sistematizadas fueron codificadas respetando la necesidad del usuario, aspecto fundamental para llegar al producto final, es decir la solución.

Con el propósito de ser objetivo en lo antes especificado me permito sistematizar el plan de iteraciones en la figura 3.2

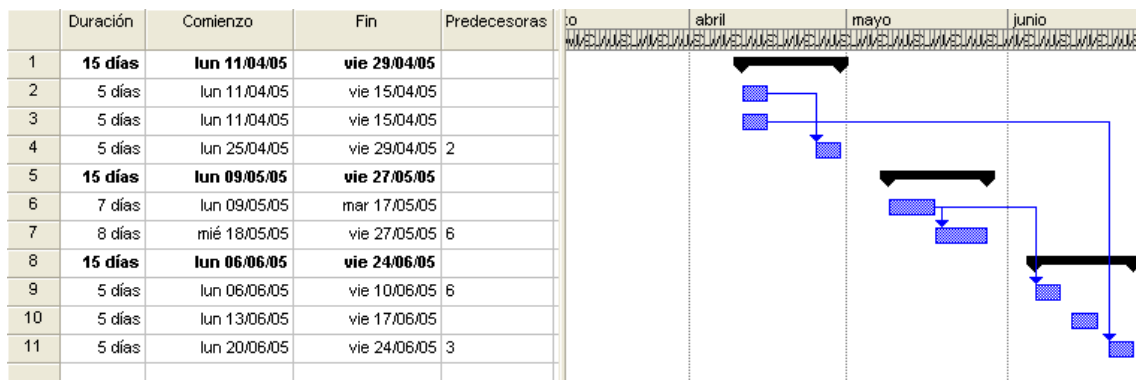


Figura 3.2: Plan de Iteraciones

### 3.2.5 ROTACIONES

Dentro del proceso metodológico de XP se contempla la realización de rotaciones, ya que ella permite al personal que integra el equipo de desarrollo rotar para cumplir las diferentes actividades del desarrollo de software.

En razón de que la solución alcanzada obedece a la autoría de una sola persona no ha sido posible cumplir con este paso

### **3.2.6 REUNIONES**

Alcanzar la solución o software fue producto de un sin número de sesiones de trabajo con los usuarios de la institución. Ellos en calidad de beneficiarios, y con mi asesoramiento técnico identificamos las necesidades y por tanto apoyaron para construir la solución respectiva.

## **3.3 DISEÑO DE LA SOLUCION**

### **3.3.1 METAFORA DEL SISTEMA**

La solución creada permitirá constatar físicamente los activos fijos de todas las dependencias de la Escuela Politécnica del Ejército. Esta información se extraerá de la base de datos central, los mismos que son enviados a través de Web Service al dispositivo móvil y éste almacena en una base de datos local (SQL CE), para finalmente proceder a la actualización.

### **3.3.2 TARJETAS CRC**

La metodología XP, contempla la necesidad de contar con tarjetas CRC en razón de que ellas reflejan la interacción de las clases que tiene el sistema, permitiendo la obtención del diagrama de clases.

En la solución alcanzada se cuenta con las siguientes tarjetas:

### TARJETA 3.1: CRC USUARIO

<b>USUARIO</b>	
<b>Responsabilidad</b>	<b>Colaboración</b>
1. Registrar los usuarios del sistema	
2. Crear username	
3. Generar contraseña	
4. Modificar contraseña	

Tarjeta 3.1 CRC USUARIO

La tarjeta CRC USUARIO, tiene como función principal el crear, eliminar y actualizar a los usuarios

### TARJETA 3.2: SEGURIDAD

<b>SEGURIDAD</b>	
<b>Responsabilidad</b>	<b>Colaboración</b>
1. Registrar los Perfiles	
2. Registrar los módulos del sistema	
3. Asignar los módulos a los perfiles	
4. Asignar a los usuarios el perfil	Usuarios

Tarjeta 3.2 CRC SEGURIDAD

La tarjeta CRC SEGURIDAD, facilita la creación de las seguridades de la aplicación

### TARJETA 3.3: CRC BIEN

<b>BIEN</b>	
<b>Responsabilidad</b>	<b>Colaboración</b>
1. Buscar la información de un bien	Estados
	ResponsablesBien
	Dependencia
2. Mantenimiento de la información	
3. Verificación si fue o no ya revisada	

Tarjeta 3.3 CRC BIENES

La tarjeta CRC BIENES, tiene como función el buscar y actualizar los bienes.

### TARJETA 3.4: CRC DESCARGAR

<b>DESCARGAR</b>	
<b>Responsabilidad</b>	<b>Colaboración</b>
1. Mostrar información de los bienes	
2. Guardar la información	Estado
	ResponsablesBien
	Dependencia
	Bien
	Usuario

Tarjeta 3.4 CRC DESCARGAR

La tarjeta CRC DESCARGAR, tiene como función el guardar la información de los bienes en los dispositivos móviles

### TARJETA 3.5: CRC SINCRONIZAR

<b>SINCRONIZAR</b>	
<b>Responsabilidad</b>	<b>Colaboración</b>
1. Publicar la información a sincronizar	Bien
2. Enviar lista de acciones realizadas	BienLog
3. Mostrar fechas	

Tarjeta 3.5 CRC SINCRONIZAR

La tarjeta CRC SINCRONIZAR, tiene como función el de actualizar la información modificada de los bienes en la base de datos central

### TARJETA 3.6: CRC BIENLOG

<b>BIENLOG</b>	
<b>Responsabilidad</b>	<b>Colaboración</b>
1. Guarda la acción realizada	
2. Guarda el usuario que realizo	Usuario
3. Guarda la información del bien	Bien

Tarjeta 3.6 CRC BIENESLOG

La tarjeta CRC BIENLOG, tiene como función el registrar la fecha y usuario que modificó la información del bien

### TARJETA 3.7: CRC WEBSERVICES

WEBSERVICES	
Responsabilidad	Colaboración
1. Envía la información a descargar	Descargar
2. Recibe la información a actualizar	Sincronizar
3. Login	Usuario

Tarjeta 3.7 CRC WEBSERVICES

La tarjeta CRC WEBSERVICES, tiene como función el trasladar la información al Pocket Pc y desde él, hacia la base de datos central.

### DIAGRAMA DE CLASES

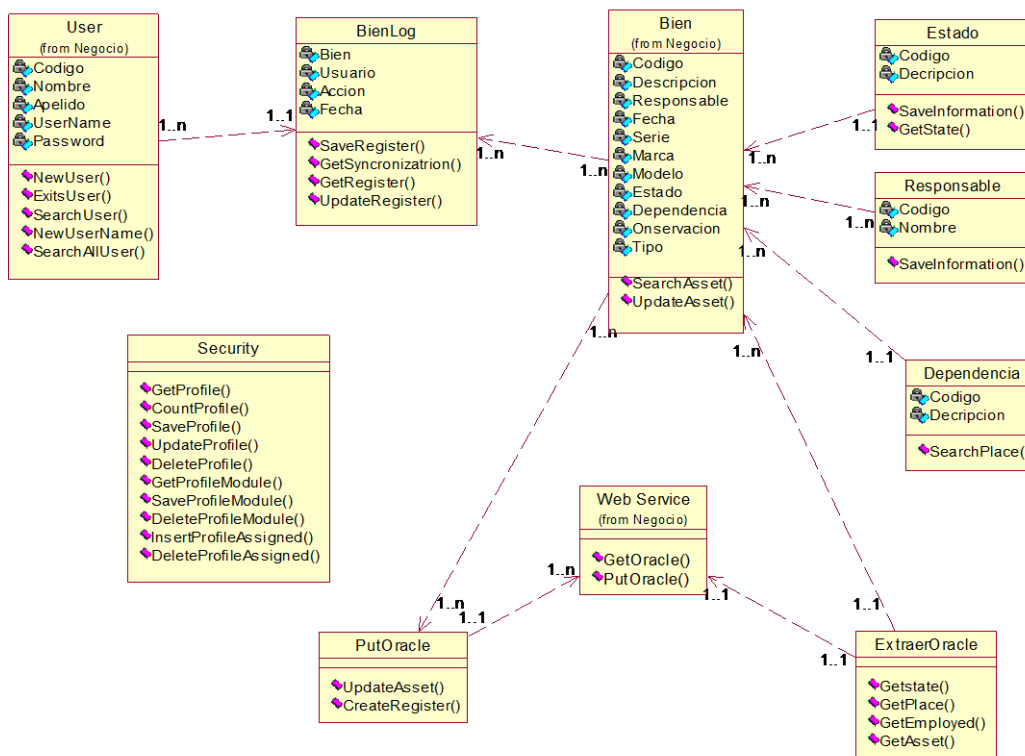


Diagrama 3.1 Diagrama de Clases

El diagrama representa la iteración entre las clases producidas por las tarjetas CRC de la solución

### **3.3.3 SOLUCIONES PUNTUALES**

Se desarrolló para dos entornos diferentes, que unidos dentro de cada uno, permitirá ver la funcionalidad completa de toda la solución.

La solución móvil

- Descarga de información
- Constatación
- Sincronización

La solución Windows

- Seguridades
- Reportes

### **3.3.4 FUNCIONALIDAD MINIMA**

El sistema diseñado tiene una funcionalidad mínima que le permite al usuario obtener lo que él solicita en las historias de usuario, esta respuesta tiene un aspecto operativo fundamental el de entregar solo la petición y no cosas adicionales que el desarrollador desea.

### **3.3.5 RECICLAJE**

La metodología XP establece que en lo posible se mantenga la menor cantidad de código comentado, esto facilita que el código sea de fácil entendimiento para cualquier integrante del equipo de desarrollo.

El código que ha permitido obtener la solución en el presente trabajo se encuentra en su totalidad libre de código comentado, situación que facilita al desarrollador entender y evitar complicaciones.



# 4 DESARROLLO DEL SISTEMA

## 4.1 INTRODUCCION

En el capítulo anterior se especificó que el sistema tiene dos partes, la primera los dispositivos móviles y la segunda las PC de escritorio, partes que integran la solución.

El desarrollo del programa para dispositivos móviles, está sobre la base de los siguientes procesos:

- Cargar la información al dispositivo móvil
- Constatar físicamente un bien
- Sincronizar la información al servidor de base de datos Oracle

Así mismo el desarrollo del programa para PC de escritorio, está sobre la base de los siguientes procesos:

- Creación de usuarios y generación de permisos
- Generación de reportes

## 4.2 DISPONIBILIDAD DEL CLIENTE

Durante el desarrollo del software, el cliente fue privilegiado y como tal estuvo presente en todas las etapas del proceso, situación que facilitó la realización de

pruebas de funcionamiento del sistema y a la vez se aclararon inquietudes manifiestas en la etapa de planificación.

La circunstancia descrita permitió la consolidación y fortificación del funcionamiento del software, lo que quiere decir que el cliente tuvo un nivel excepcional en el proceso.

### **4.3 UNIDADES DE PRUEBA**

La eficiencia y eficacia del software se dio en función a un código, el mismo que permitió identificar errores de funcionamiento del sistema, para lo que se utilizó las siguientes pruebas:

- Creación de username

La prueba consistió en conocer si el username o usuario del sistema que se generaría, se encuentra o no registrado en la base de datos.

El resultado de la prueba unitaria aplicada a la creación del UserName generó datos satisfactorios.

A continuación se muestra el test de la prueba unitaria bajo nunit:

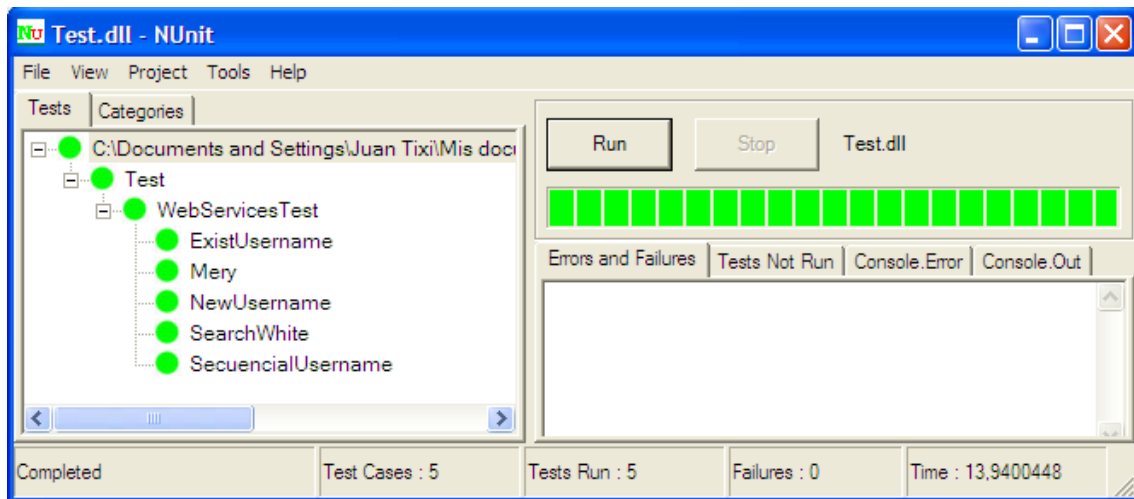


Figura 4.1: Ventana test Creación de username

- Descarga de información a los Dispositivos Móviles

La prueba permitió identificar caracteres que generan errores al momento de almacenar la información en la base de datos SQLCE, situación que controló la posible presencia de errores.

- Sincronización a la base de datos central desde los Dispositivos Móviles

La prueba optimizó la transferencia de los datos de los bienes desde los dispositivos móviles hacia Oracle, para lo que la prueba buscó únicamente los bienes actualizados o consultados.

#### **4.4 PROGRAMACION POR PAREJAS**

En razón del proceso metodológico este principio no fue aplicado, es decir porque la autoría de la tesis es de una sola persona.

## 4.5 INTEGRACION

El código fue elaborado y unificado por una sola persona, en razón de que el trabajo es individual.

## 4.6 ESTANDARES

El desarrollo del software obedece a los siguientes estándares:

### 4.6.1 CODIFICACION

Gran parte del código es reutilizable, la capa de datos es genérica.

Para poder entender el código de mejor manera se siguió los siguientes estándares en cuanto a escritura de código:

TIPO	DETALLE	NOMENCLATURA
CLASE	Nombre descriptivo de lo que realiza la clase	LocalStore
METODO	Nombre descriptivo de lo que realiza el método	GetState()
ATRIBUTO	Nombre descriptivo de lo que contiene el atributo	codigoEmpleado

Tabla 4.1: Estándares de codificación de clases

Permite caracterizar con propiedad la manera de nombrar a CLASE, METODO y ATRIBUTO.

**Clase:** Es la representación de un objeto del mundo real.

```
public class Service
{
}
}
```

**Método:** Es la acción que es ejecutada sobre una clase.

```
public string Encrypt(string text)
{
    try
    {
        Byte [ ] bytes = Encoding.ASCII.GetBytes(text);
        return Convert.ToBase64String(bytes,0, bytes.Length);
    }
    catch(Exception ex)
    {
        System.Diagnostics.Debug.Write(ex.Message);
        return String.Empty;
    }
}
```

**Atributo:** Consiste en las características de cada clase.

```
string username = "ATOPON";
```

Dentro del código fuente, no se debe escribir ningún tipo de comentario, con relación a lo que esa porción de código realiza.

Para mantener un código fuente lo mas claro ya que este tipo de comentarios del código llevan a confusiones.

Para poder conocer que realiza cada una de los métodos que contienen cada una de las clases, refiérase al ANEXO B.

## 4.6.2 INTERFAZ DE USUARIO

El sistema está conformado por POCKET PC y Windows, ellos tienen la siguiente presentación:

### 4.6.2.1 POCKET PC

El software para los dispositivos móviles se diseñó considerando su utilidad y presentación, así por ejemplo en la interfaz de usuario se utiliza:

- Cajas de texto: se utiliza cajas de texto para el ingreso y despliegue de información

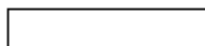


Figura 4.2: Caja de Texto

- Etiquetas (labels): Las etiquetas se las usa para identificar el contenido de las cajas de texto.

Usuario

Figura 4.3: Etiqueta (label)

- Etiquetas (creadas por código): las etiquetas creadas por código se las utiliza para la ejecución de determinadas tareas: Sign In, Salir, Sign Out, Descargar base, Constatación y Sincronizar.

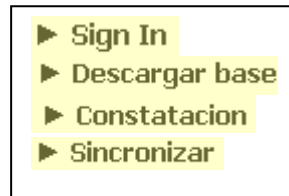


Figura 4.4: Etiquetas (creadas por código)

- Radio botones: Los radios botones son utilizados para las opciones de selección

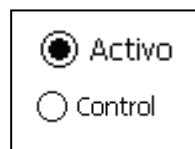


Figura 4.5: Radio botones

- Datagrids: Los datagrids son utilizados para realizar acciones, desconectados de la fuente de datos.

	DEPENDE	CODIGO	ID
▶	BIBLIOTE	1175	200

Figura 4.6: Datagrids

- Tipo de letra: Times New Roman 10
- Colores: Amarillo, verde y azul

- Cuadros de Mensajes: Los cuadros de mensajes son utilizados para dar alertas al usuario

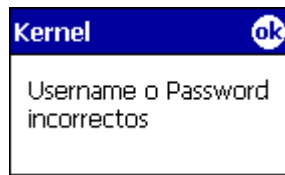


Figura 4.7: Mensajes

A continuación se muestran algunas de las interfaces



Figura 4.8: Ventana acceso al sistema de los POCKET PC

#### 4.6.2.2 WINDOWS

El software para las PC de escritorio se diseñó considerando su utilidad y presentación, así por ejemplo en la interfaz de usuario se utilizó:



- Caja de texto: La caja de texto se utilizan para el ingreso y despliegue de información



Figura 4.9: Caja de texto

- Hiper links: Los hiper links permiten llamar ventanas, son utilizados para la ejecución de los siguientes procesos: Cambiar password, Sign In, Sign Out, Salir, Acerca de



Figura 4.10: Hiper link

- XpLabels: Los XpLabels son botones que permiten invocar métodos y o abrir ventanas, se utilizan para el ingreso a Seguridad y Reportes.

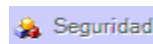


Figura 4.11: XpLabels

- Check box: Los Check box sirven para presentar opciones de verdadero o falso, utilizados para las opciones de selección



Figura 4.12: Check box

- Datagrids: Los datagrids son utilizados para realizar acciones, desconectados de la fuente de datos.

PERFILES	
	DESCRIPCION
1	ADMINISTRADOR
3	PRUEBA
2	POCKET
*	

Figura 4.13: Datagrid

- Control Colapsable: Esos controles están representados por flechas y permite minimizar secciones



Figura 4.14: Control colapsable

A continuación se muestran algunas de las interfaces

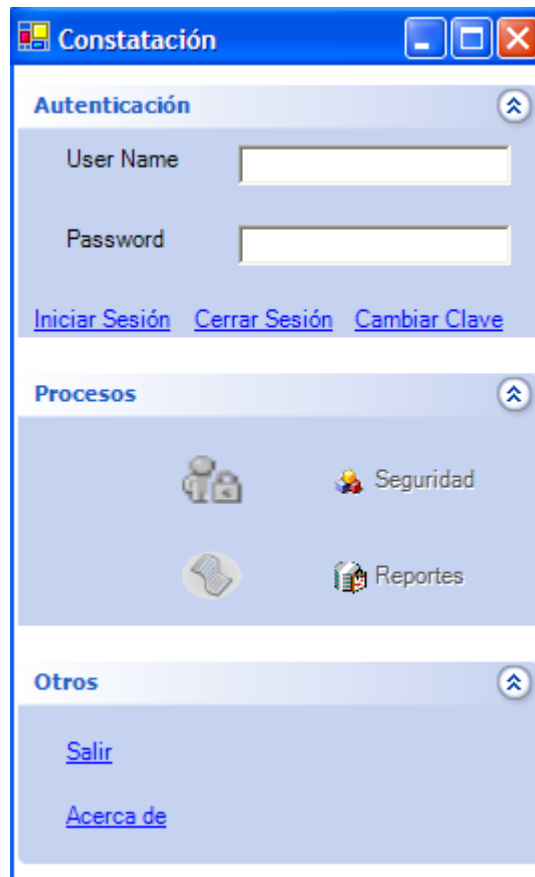


Figura 4.15: Ventana de acceso al sistema de administración

### 4.6.3 BASE DE DATOS

Definir un estándar para la creación, manejo y manipulación de las estructuras físicas de las bases de datos, siempre ha sido un inconveniente en vista de que cada equipo de desarrollo lo puede estandarizar de acuerdo a sus necesidades.

Al no existir ninguna normativa de cómo se debe nombrar a cada uno de los objetos de la base de datos, se ha estandarizado de la siguiente manera.

### **Entidades**

Para poder identificar de manera adecuada, y representar un ente de una realidad, los nombres poseen la siguiente estructura.

Primer carácter:	Letra inicial del sistema
Segundo carácter:	Letra inicial del identificador
Tercero al quinto carácter:	Tres primeras letras del nombre
Sexto carácter:	Guión bajo
Séptimo carácter en adelante:	Nombre descriptivo de la entidad

Ejemplo:

CERES\_RESPONSABLE

### **Atributos**

Quienes representan las principales particularidades de una entidad, deben tener un nombre descriptivo y su tipo de dato debe ir acorde al contenido que información que ellos poseerán.

Los nombres poseen la siguiente estructura.

Primer al quinto carácter:	Los cinco iniciales de la entidad
Sexto carácter:	Guión bajo
Séptimo carácter en adelante:	Nombre descriptivo del atributo

Ejemplo:

CERES\_CEDULA

## **Relaciones**

Son quienes nos permiten conocer como actúan las entidades entre si, además nos ayudan a establecer la relación de dependencia.

Los nombres poseen la siguiente estructura

Palabra relación

Nombre de la tabla padre

Nombre de la tabla dependiente

Entre cada palabra debe existir un guión bajo

Ejemplo:

Relacion\_Empleado\_Bien

## CAPITULO V

### 5 PRUEBAS

Las pruebas de Implantación y de Aceptación, facilitaron obtener un producto final eficiente y un buen desempeño de la solución desarrollada.

Con la ayuda de Nunit se pudo probar cada una de las pruebas unitarias realizadas, obteniendo una solución bastante estable y con errores controlados.

El siguiente código muestra cada uno de los requisitos para poder realizar una prueba unitaria.

Cada uno de los métodos contiene la palabra Test, no poseen parámetros

El método SearchWhite, busca un espacio en blanco en una cadena

```
[Test]
public void SearchWhite()
{
    string text = "ANITA MARIA";
    int result = 0;
    int i;
    for (i = 0; i < text.Trim().Length; i++)
    {
        if(text.Substring(i, 1) == " ")
        {
            result = i;
            i = text.Trim().Length;
        }
    }
}
```

El método `SecuencialUsername` busca el siguiente número del usuario

```
[Test]
public void SecuencialUsername()
{
    string username = "ATOPON";
    string query;
    query = string.Format("Select count(*) from AEUSM_USUMO where
        AEUSM_USERNAME like '{0}%', username + "%");
    Convert.ToInt16(Kernel.Instance.ExecuteScalar(query));
}
```

El método `ExistUsername` busca si existe un usuario con ese nombre

```
[Test]
public void ExistUsername()
{
    string username = "ATOPON";
    string query;
    query = string.Format("Select count(*) from AEUSM_USUMO where
        AEUSM_USERNAME = '{0}'", username);
    Convert.ToInt16(Kernel.Instance.ExecuteScalar(query));
}
```

El método `NewUsername` crea un nuevo usuario

```
[Test]
public void NewUsername()
{
    string nombre = "ANITA MARIA";
    string apellido = "TOPON ROMERO";
    string cedula = "1700800145";
    string nombreI;
    string nombreF = String.Empty ;
    string apellidoParteno;
    int posicionEspacion;
    string username;
    string apellidoMenor = String.Empty;
    string parteApellido = String.Empty;
    string respuesta = "";
    try
    {
        posicionEspacion = 7;
        if (posicionEspacion > 0)
        {
```

```

        nombreI = nombre.Substring(0,
        posicionEspacion);
        nombreF = nombre.Substring(posicionEspacion +
        1, nombre.Length - posicionEspacion - 1);
    }
    else
    {
        nombreI = nombre;
    }

    posicionEspacion = 7;

    if (posicionEspacion > 0)
    {
        apellidoParteno = apellido.Substring(0,
        posicionEspacion);
    }
    else
    {
        apellidoParteno = apellido;
    }
    if (apellidoParteno.Length > 7)
    {
        apellidoParteno =
        apellidoParteno.Substring(0,7);
    }

    username = nombreI.Substring(0, 1) +
    apellidoParteno;

    if (1 == 1)
    {
        if (apellidoParteno.Length > 6)
            parteApellido =
            apellidoParteno.Substring(0,6);

        if (apellidoParteno.Length < 7)
            parteApellido = apellidoParteno;

        if (apellidoParteno.Length == 7)
            parteApellido = apellidoParteno;

        if (nombreF == string.Empty)
            username = nombreI.Substring(0, 1) +
            parteApellido;
        else
        {
            if (parteApellido.Length == 7)
            {
                parteApellido =
                parteApellido.Substring(0, 6);
            }

            username = nombreI.Substring(0,1) +
            nombreF.Substring(0,1) + parteApellido;
        }
        if (1 == 1)
        {
            if (username.Length < 8)
            {

```

```

        username = username +
            Convert.ToString("ATOPON");
    }

    if (username.Length == 8)
    {
        username = username.Substring(0,
            7) + Convert.ToString("ATOPON");
    }
}
respuesta = username;
}
catch
{
    respuesta = "Error";
}
}

```

El método Encrypt cambia una frase a texto seguro

```

[Test]
public string Encrypt()
{
    string text = "1700800";
    try
    {
        byte[] bytes = Encoding.ASCII.GetBytes(text);
        return Convert.ToBase64String(bytes, 0 , bytes.Length);
    }
    catch(Exception ex)
    {
        System.Diagnostics.Debug.Write(ex.Message);
        return String.Empty;
    }
}

```

## 5.1 DE IMPLANTACION

Facilitan la estabilidad de la solución, y fueron realizadas usando el código que se creo durante la etapa de pruebas unitarias, y utilizando el software Nunit 2.2 para medir la eficacia de la prueba.



La siguiente figura indica el resultado de haber ejecutado la prueba unitaria para la creación del UserName:

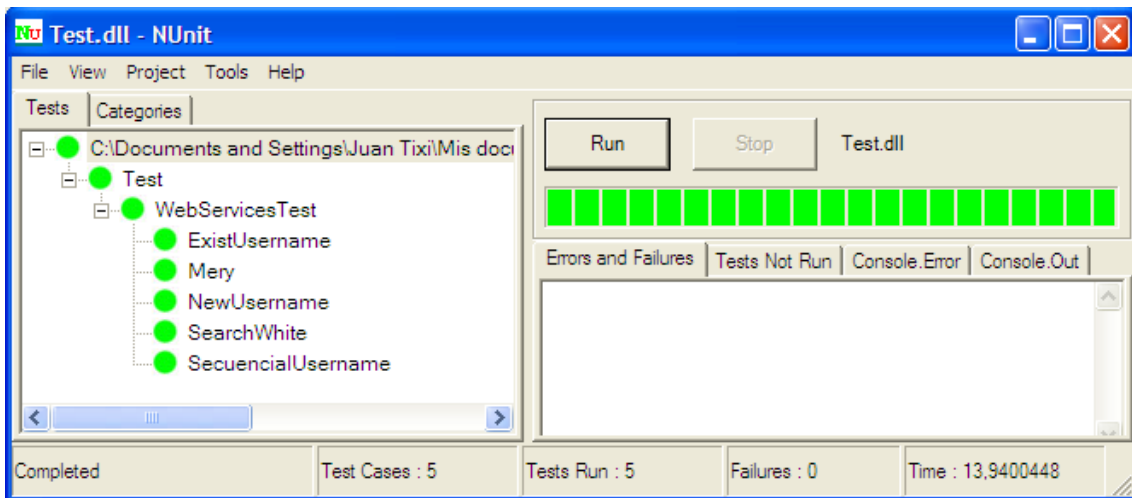


Figura 5.1: Ventana test Creación de username

El círculo color verde al lado izquierdo de cada uno de los métodos de la clase indica que la prueba fue exitosa.

Si existiese un círculo de color amarillo al lado izquierdo de un método de la clase indica que la prueba no fue realizada

Si existiese un círculo de color rojo al lado izquierdo de un método de la clase indica que la prueba no fue exitosa.

## 5.2 DE ACEPTACION

El actor principal en este tipo de pruebas es el usuario final, en vista de que es quien indica que desea probar y esta pendiente de los resultados que la solución le presente.

Las pruebas se realizaron en la parte de la solución móvil y específicamente en:

***Descarga de los datos***

Resultado de la prueba: Satisfactorio para el usuario.

***Constatación Física***

Resultado de la prueba: Satisfactorio para el usuario.

Los resultados obtenidos de las pruebas realizadas a la descarga de los datos y a la constatación física, dejaron al usuario final seguro de que el software desarrollado cumplió con todos los requerimientos.

## CAPITULO VI

# 6 CONCLUSIONES, RECOMENDACIONES Y ANEXOS

## 6.1 CONCLUSIONES

- La utilización de la herramienta de desarrollo Visual Studio .Net permite la creación de todo tipo de soluciones, ellas están orientadas a satisfacer la necesidad del cliente.
- La elaboración de las historias de usuario permite la materialización óptima y la concreción para abstraer y representar los requerimientos del mismo, ya que el usuario es el sujeto que sistematiza su propia solución al requerimiento.
- El uso simultáneo del plan de entregas e iteraciones dentro de la Metodología Xp facilita al usuario el conocimiento del tiempo y de cuándo se finalizará toda o parte de la solución.
- La utilización de las tarjetas clase, relación , colaboración (CRC) facilita la obtención del diagrama de clases y además permite conocer la interrelación entre cada una de éstas, permitiendo llegar al diseño de la solución final.

- La integración del usuario final dentro del equipo de desarrollo optimiza la productividad de los programadores o desarrolladores y despeja inquietudes que en el proceso se presenta.
- La utilización de herramientas de pruebas unitarias, permite al programador mayor concentración para escribir el código que da solución al negocio.
- El estándar XML permite la integración de sistemas operativos e inter comunica a las soluciones realizadas en diferentes plataformas.

## **6.2 RECOMENDACIONES**

- La Metodología utilizada en el presenta trabajo, si es factible de ejecutar en proyectos medianos y grandes, siempre y cuando se cuente con el compromiso decidido del cliente.
- Para la realización de soluciones a dispositivos móviles se recomienda cuando exista poca conectividad con la base de datos
- Es importante la correcta utilización del documento de instalación web services y aplicaciones para garantizar el funcionamiento óptimo del sistema.

- Es fundamental e importante para el programador la integración del usuario en el proceso a fin de que verifique su necesidad y garantice la solución.

### ***6.3 ANEXO A: MANUAL DE INSTALACION DE WEB SERVICE Y APLICACIONES***

### ***6.4 ANEXO B: MANUAL DEL PROGRAMADOR***

### ***6.5 ANEXO C: MANUAL DE USUARIO***

# **ANEXOS**

**ANEXO A**

**MANUAL DE INSTALACION**

**WEB SERVICES**

**Y**

**APLICACIONES**

## **MANUAL DE INSTALACION DE WEB SERVICE Y APLICACIONES**

El óptimo funcionamiento del sistema, estará sujeto a la instalación por un profesional técnico en la materia.

Instale el Framework 1.1 en las PC donde funcionará las aplicaciones, para ello:

- Inserte el CD, cuya etiqueta es C.A.F. CD INSTALADORES
- De doble clic en la carpeta FRAMEWORK
- De doble clic en dotnetfx.exe
- Siga las instrucciones que indica el instalador.

Realizado ordenadamente este proceso, está en capacidad de instalar cada uno de los siguientes componentes del sistema:

- **A.** Web Services (Servidor)
- **B.** Constatación física (Dispositivos móviles)
- **C.** Administración (Windows)

### **A. WEB SERVICES (SERVIDOR)**

Contiene la lógica de la solución, además establece la comunicación entre la base de datos y la aplicación de Administración y Constatación física, no se necesita instalar ninguna base de datos ni correr ningún script de SQL, ya que se trabajara contra una base de datos y esquema existente.



Para una adecuada instalación, considere los siguientes aspectos:

✓ **Verifique que el IIS esté instalado en el servidor:**

Digite la palabra “localhost” en la barra de direcciones del Internet Explorer y presionar enter, si se encuentra instalado e iniciado el servicio se presentará la siguiente pantalla



Figura A.1: Pagina de prueba de IIS

De presentarse la pantalla de error, opte por la opción A o B:

A.- Instale el servicio de IIS, siga los siguientes pasos:

1. Haga clic en Inicio de Windows
2. Haga clic en Panel de Control

3. Haga clic en Agregar o quitar programas
4. Haga clic en Agregar o quitar componentes de Windows
5. Haga clic en Servicios de Internet Information Server(IIS),  
Su servidor está instalado

B.- Inicie el servicio de IIS, siga los siguientes pasos:

1. Haga clic en Inicio de Windows
2. Haga clic en Panel de Control
3. Haga clic en Herramientas Administrativas
4. Haga clic en Servicios de Internet IIS
5. Seleccione el Servidor Web
6. Seleccione el Sitio Web,
7. De clic derecho
8. Seleccione Iniciar,  
Su servidor está iniciado.

✓ **Copie los archivos a publicar**

Cree una carpeta en el disco duro del servidor con el nombre “Bienen.Business.WebServices”, en ella guarde los siguientes archivos y carpetas:

1. Inserte el CD, cuya etiqueta es C.A.F. CD INSTALACION
2. Busque la carpeta SERVIDOR
3. En la carpeta creada, copie los siguientes archivos y carpetas

Archivos:

Authentication.asmx

GetOracle.asmx

PutOracle.asmx

Rpt.asmx

Sec.asmx

User.asmx

Web.config

Carpeta:

Bin

✓ **Cree un directorio virtual, para ello siga los siguientes pasos:**

1. Hacer clic en Inicio de Windows
2. Elegir la opción "Panel de Control" del menú Inicio
3. Dar un clic en el icono de Herramientas Administrativas
4. Hacer clic en el icono de Servicios de Internet Information Server (IIS)
5. Elegir el Sitio Web predeterminado y dar clic derecho
6. Hacer clic en la opción Nuevo
7. Elegir la opción Directorio Virtual, aparecerá Alias
8. A continuación aparece un Asistente para la creación de un Directorio Virtual
9. Dar un clic en el botón Siguiente
10. En Alias, escribir el nombre Bienen.Business.WebServices
11. Hacer clic en el botón Siguiente

12. En Directorio, haga clic en el botón Examinar, aparecerá una ventana similar al explorador de Windows

13. Elegir la carpeta que contiene los archivos a publicar y dar presionar el botón Aceptar.

14 Cuando ya se haya elegido la carpeta deseada presionar el botón Siguiente y se desplegarán los Permisos de acceso

15. En permisos de acceso, presionar el botón Siguiente.

16. Pulsar el botón Finalizar

Con los anteriores pasos se ha creado el Directorio virtual.

#### ✓ **Configuración Cliente - Servidor**

1. Hacer clic en Inicio de Windows

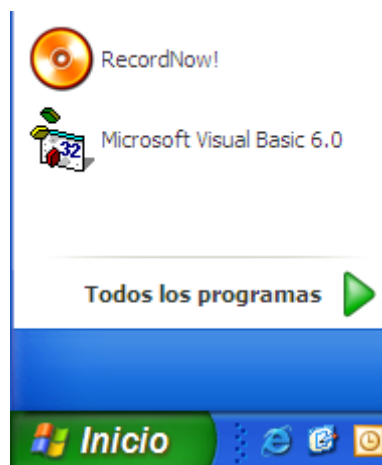


Figura A.2: Inicio de Windows

## 2. Elegir la opción “Todos los programas” del menú Inicio

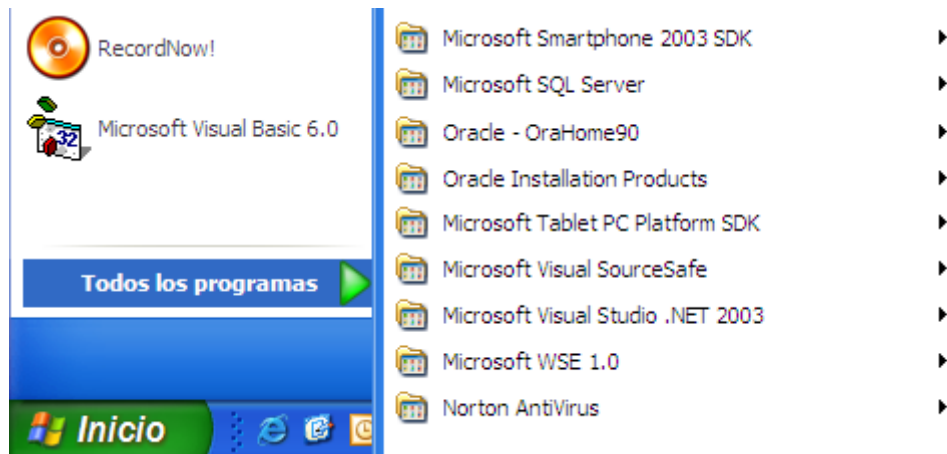


Figura A.3: Acceso a Oracle

## 3. Dar un clic en “Oracle – OraHome80”

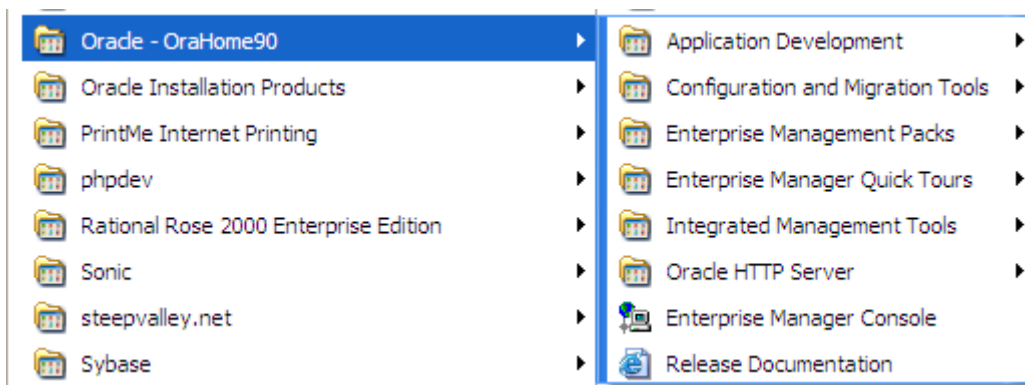


Figura A.4: Opciones instaladas en Oracle

## 4. Seleccionar la opción “Configuration and Migration Tools”

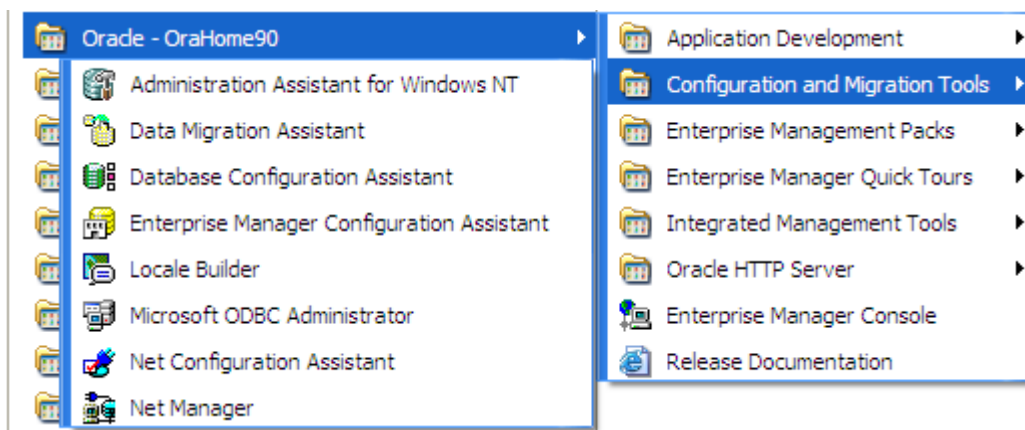


Figura A.5: Configuración de Tnsname

5. Elegir “Net Configuration Assistant”, así:

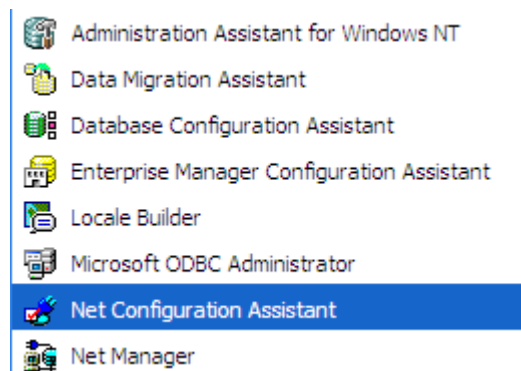


Figura A.6: Net Configuration Assistant

6. Seleccionar “Configuración de Nombre del Servicio de Red Local”, y luego presionar el botón “Siguiente”, así:



Figura A.7: Selección de Configuración de Nombre del Servicio de Red Local

7. Seleccionar “Agregar”, y luego presionar el botón “Siguiente”, así:

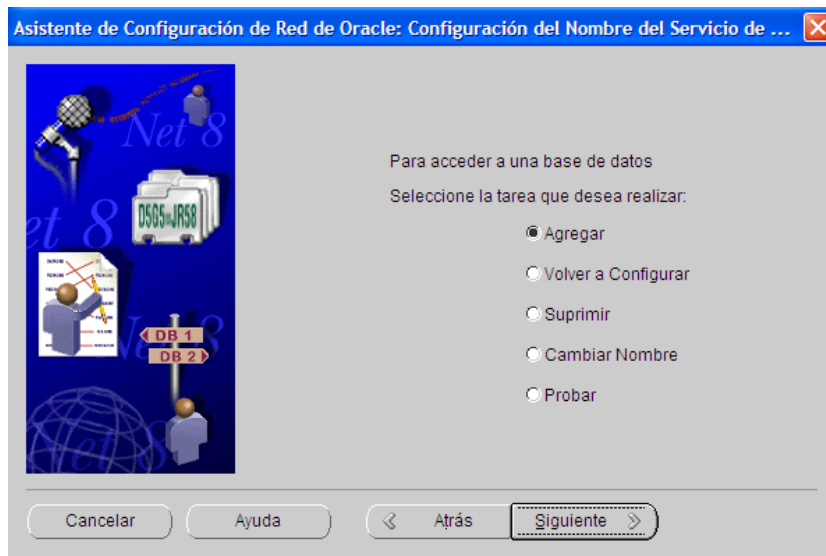


Figura A.8: Selección de la opción Agregar

8. Seleccionar la versión de Oracle, a la cual se desea conectar y después presionar el botón “Siguiente”, así:

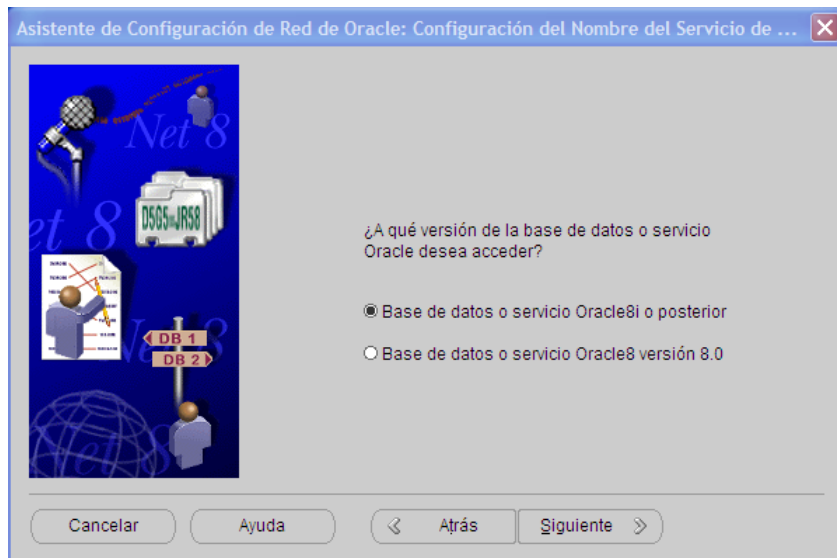


Figura A.9: Selección de la base de datos

9. Ingresar el nombre del servicio y presionar el botón “Siguiente”, así:

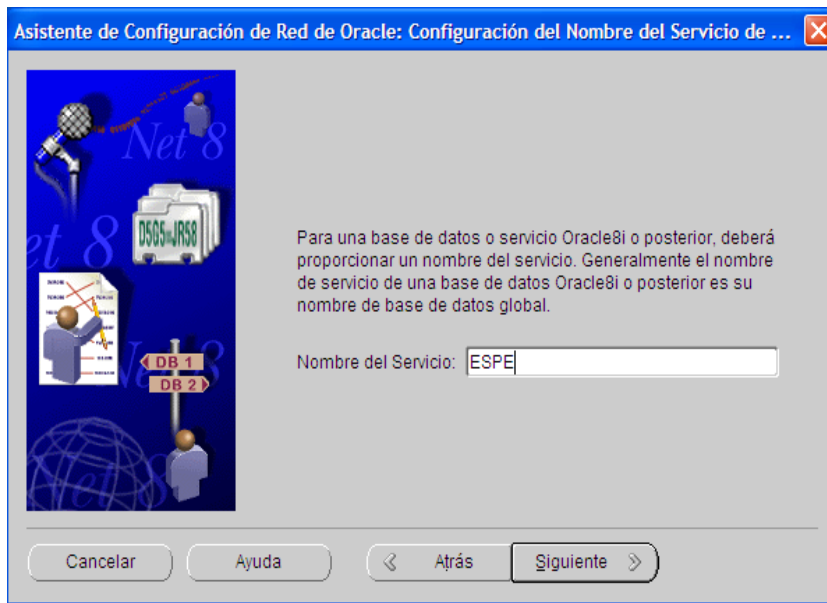


Figura A.10: Selección del nombre del servicio

10. Seleccionar un protocolo de red y presionar el botón “Siguiente”, así:

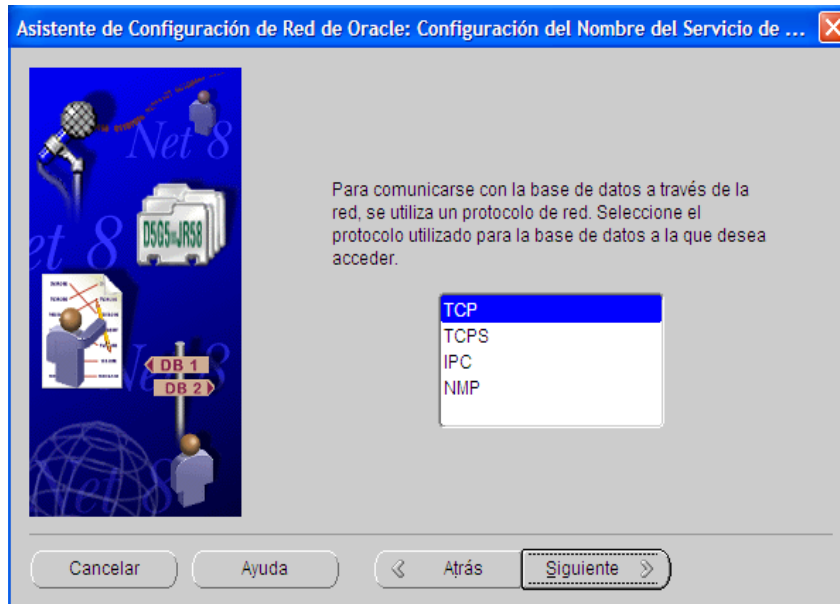


Figura A.11: Selección del protocolo de red



11. Digitar el nombre del host o la dirección IP del servidor y presionar el botón “Siguiente”, así:

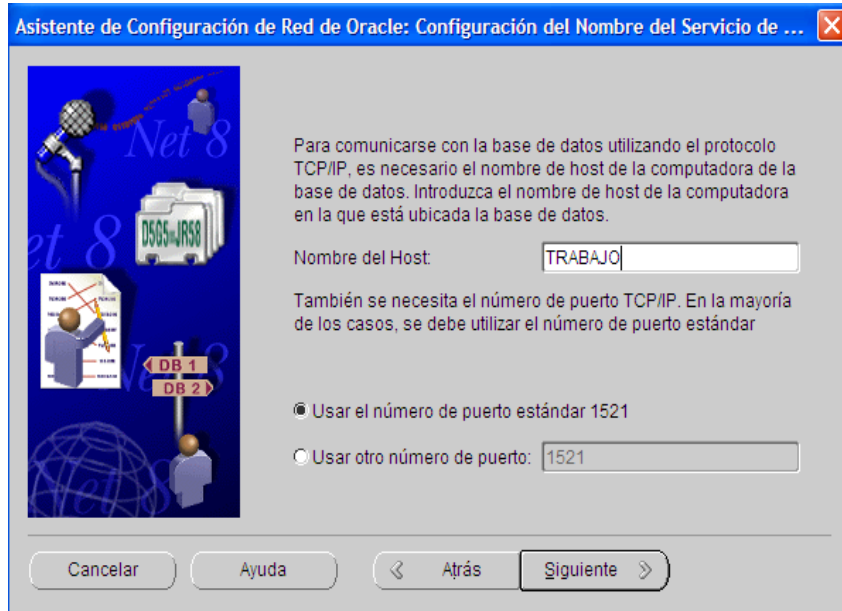


Figura A.12: Ingreso del nombre del host

12. Seleccionar si desea o no realizar la prueba de conexión y presionar el botón “Siguiente”, así:



Figura A.13: Probar conexión

## Verificación del Web Service

Para constatar que la instalación tuvo el éxito deseado, digite `http://localhost/Bienen.Business.WebServices/Authentication.asmx` en la barra de direcciones del Internet Explorer y presionar enter.

Si se muestra la siguiente pantalla la publicación se realizó con éxito

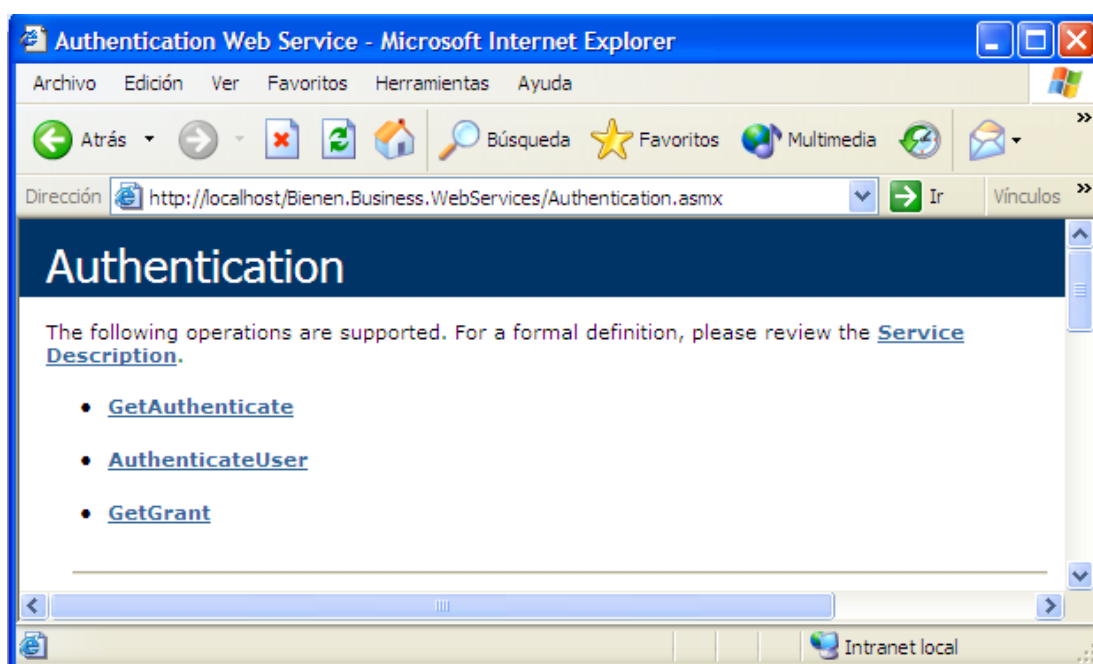


Figura A.14: Pagina de prueba del Web Services Authentication.asmx

## **B. CONSTATAION FISICA (DISPOSITIVOS MOVILES)**

Contiene el software que se ejecutara en las PDA, la base de datos que estos dispositivos manejan así como su estructura es creada en tiempo de ejecución.

Para una adecuada instalación, secuencialmente siga los siguientes pasos:

✓ **Instale Microsoft Active Sync.:**

1. Inserte el CD, cuya etiqueta es C.A.F. CD INSTALADORES
2. Busque la carpeta MICROSOFT ACTIVE SYNC, e ingrese
3. Haga doble clic en MSASYNC.EXE y siga las instrucciones que indica el instalador
4. Haga clic en siguiente, hasta finalizar la instalación.

Ha finalizado la instalación, podrá observar desde el Explorador de Windows un nuevo ítem con el nombre de Mobile Device

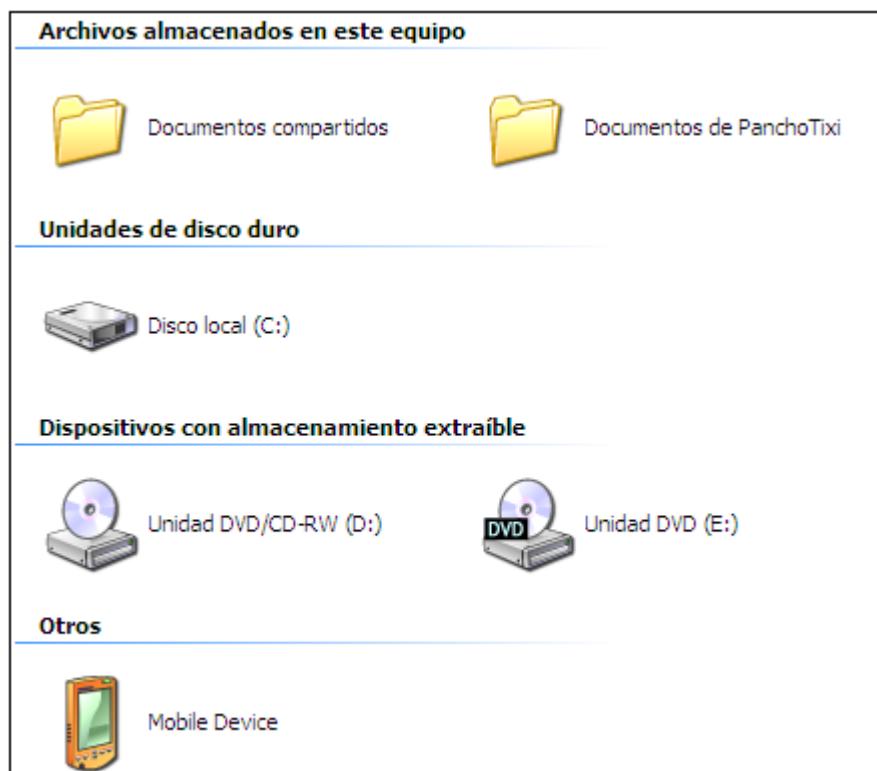


Figura A.15: Creación de Mobile Device

- ✓ **Conecte el dispositivo móvil a un PC, utilizando el cable de sincronización.**

✓ **Copie los archivos CAB al dispositivo móvil**

1. Inserte el CD, cuya etiqueta es C.A.F. CD INSTALACION
2. Utilizando el Explorador de Windows, despliegue el contenido de la unidad de CD.
3. Ingrese a la carpeta POCKET PC.
4. Seleccione los siguientes archivos para instalar: Constatacion\_PPC.CAB (Aplicación), netcf.core.ppc3.arm.cab (Compact Framework) y sqlce.ppc3.arm.CAB (SQL CE) , haga clic derecho y seleccione la opción copiar.
4. Utilice el Explorador de Windows y ubíquese en: Mobile Device\My Pocket PC\Temp
5. Haga clic derecho y seleccione la opción pegar.

✓ **Extraiga los archivos CAB e instale en los dispositivos móviles**

1. Pulse el menu Star del dispositivo móvil, seleccione y
2. Pulse en Programs, seleccione y
3. Pulse en la opción File Explorer
4. Navegue a My Device\Temp
5. Pulse el archivo netcf.core.ppc3.arm.cab  
Un mensaje indicará que el Framework ha sido instalado
6. Pulse el archivo sqlce.ppc3.arm.CAB  
Un mensaje indicará que SQL CE ha sido instalado
7. Pulse en el archivo Constatacion\_PPC.ARM.CAB

Un mensaje indicará que la aplicación ha sido instalada

Para verificar que todo el proceso de instalación del Compact Framework, Base de datos y aplicación se realizó con éxito, pulse en Start, y tendrá la siguiente pantalla, en la cual se puede apreciar la aplicación CF



Figura A.16: Pocket PC

### **C. ADMINISTRACION (WINDOWS)**

Contiene el software que permitirá gestionar los usuarios, permisos y roles que las personas pueden tener para el manejo de la aplicación.

Para su adecuada instalación, considere los siguientes aspectos

- ✓ **Haber instalado el Framework 1.1**

✓ **De clic en Administracion.msi para ello:**

- 1 Inserte el CD, cuya etiqueta es C.A.F. CD INSTALADORES
2. Utilice Windows Explorer y abra el CD
3. Ingrese en la caperta WINDOWS
3. De doble clic en el archivo Administracion.msi
4. Siga las instrucciones: haga clic en siguiente, hasta finalizar la instalación

## **ANEXO B**

# **MANUAL DEL PROGRAMADOR**

## MANUAL DEL PROGRAMADOR

El presente manual ayudara a personas con un conocimiento elevado en el manejo de base de datos Oracle 8i, Microsoft Visual Studio .Net 2003, Manejo de Web Services. Por lo que el manual se limitara exclusivamente a mostrar la funcionalidad de cada uno de los proyectos, clases y métodos. Dado que el manual lo utilizaran desarrolladores, el mismo **NO CONSTA DE CÓMO INSTALAR LAS DIFERENTES HERRAMIENTAS DE BASE DE DATOS, DESARROLLO y SERVIDOR WEB.** Lo que se indicara es la manera de poner a funcionar el código fuente de la solución, en un ambiente de desarrollo

La información que se presenta esta relacionada con la aplicación desarrollada, se indica el nombre de las clases y los métodos, el mismo que va a facilitar al programador el poder entender el código fuente y la aplicación

Para poder visualizar el contenido de la solución se debe contar con:

- Microsoft Visual Studio .NET 2003 (Plataforma de desarrollo)
- Internet Information Server (IIS) (Servidor Web)
- Oracle 8i (Base de datos)

El nombre de la carpeta que contiene el código fuente se llama Mobil, dicha carpeta se encontrara dentro del CD "C.A.F. CD CODIGO FUENTE", debe ser copiada en el disco duro de la máquina de desarrollo.



Realice de manera ordenada las siguientes actividades, previo a la visualización del código fuente.

## A. Configuración del Nombre del Servicio de Red

### 1. Hacer clic en Inicio de Windows

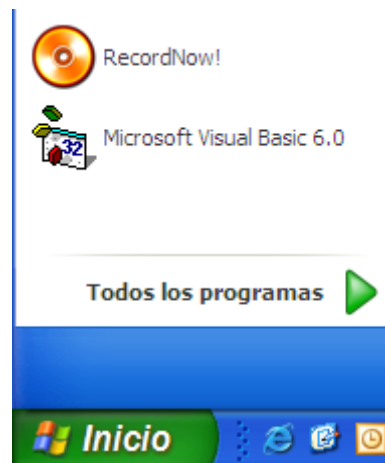


Figura B.1: Inicio de Windows

### 2. Elegir la opción “Todos los programas” del menú Inicio

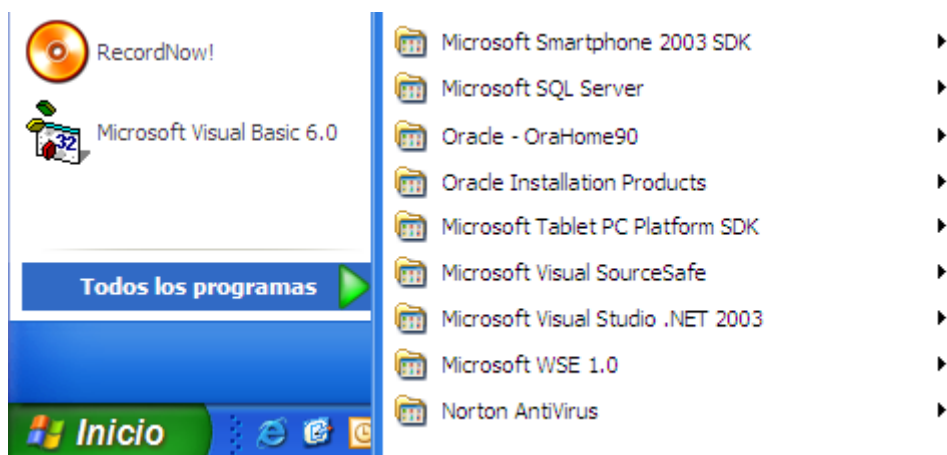


Figura B.2: Acceso a Oracle

3. Dar un clic en “Oracle – OraHome80”

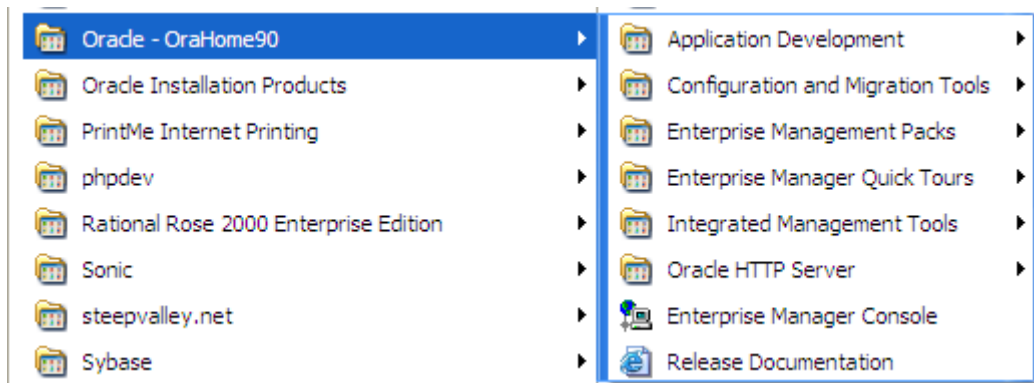


Figura B.3: Opciones instaladas en Oracle

4. Seleccionar la opción “Configuration and Migration Tools”

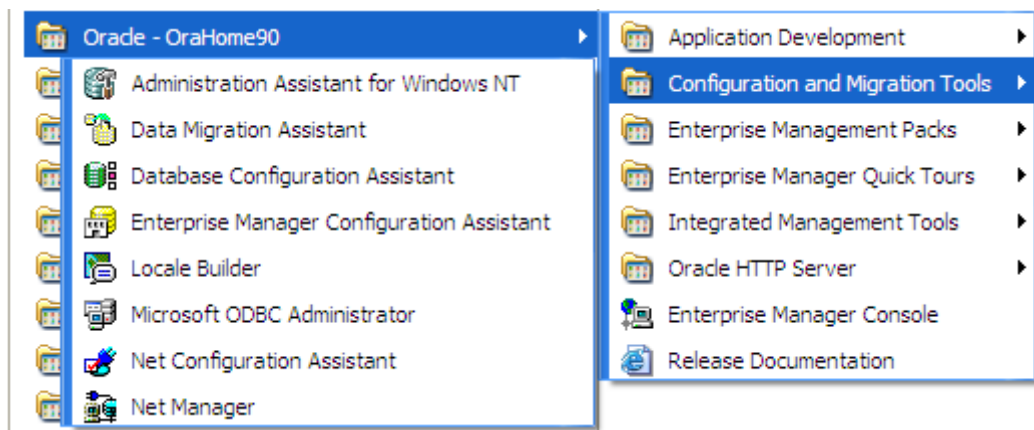


Figura B.4: Configuración de Tnsname

5. Elegir “Net Configuration Assistant”, así:

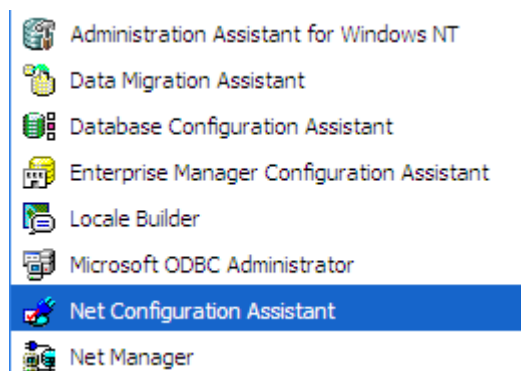


Figura B.5: Net Configuration Assistant

6. Seleccionar “Configuración de Nombre del Servicio de Red Local”, y luego presionar el botón “Siguiente”, así:



Figura B.6: Selección de Configuración de Nombre del Servicio de Red Local

7. Seleccionar “Agregar”, y luego presionar el botón “Siguiente”, así:

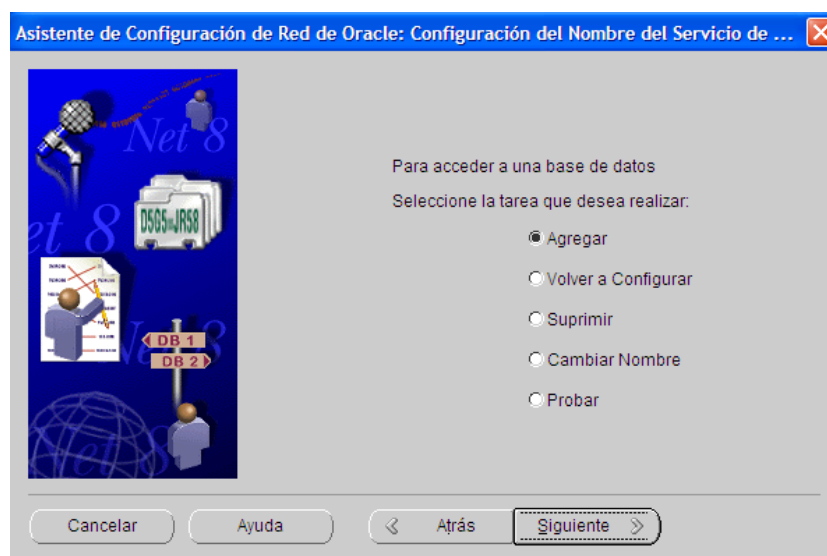


Figura B.7: Selección de la opción Agregar

8. Seleccionar la versión de Oracle, a la cual se desea conectar y después presionar el botón “Siguiente”, así:

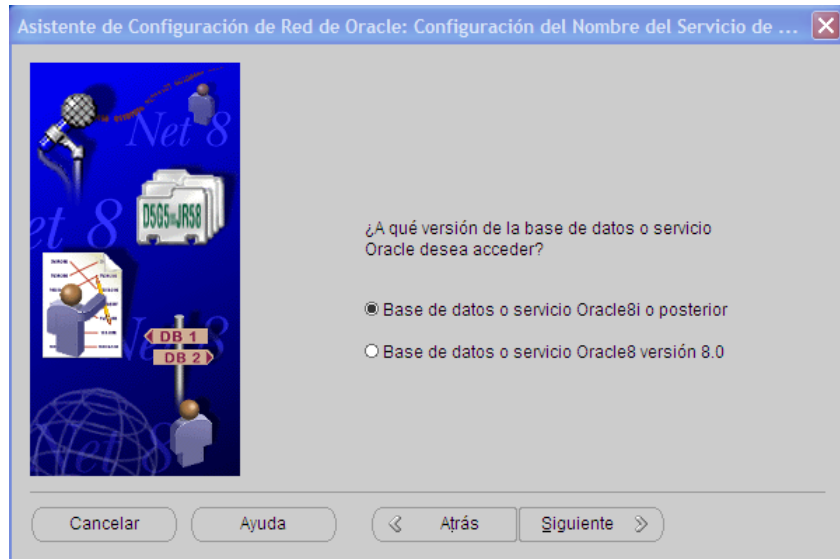


Figura B.8: Selección de la base de datos

9. Ingresar el nombre del servicio y presionar el botón “Siguiente”, así:

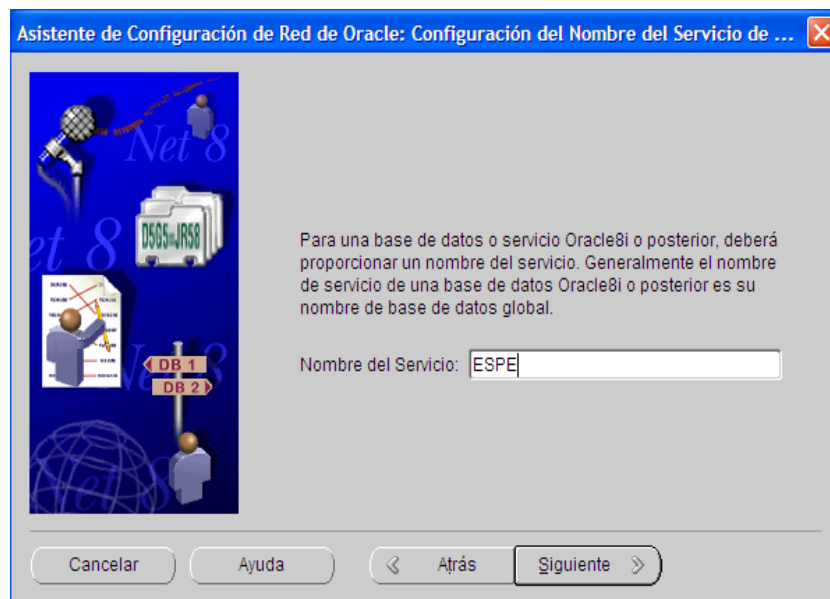


Figura B.9: Selección del nombre del servicio

10. Seleccionar un protocolo de red y presionar el botón “Siguiente”, así:

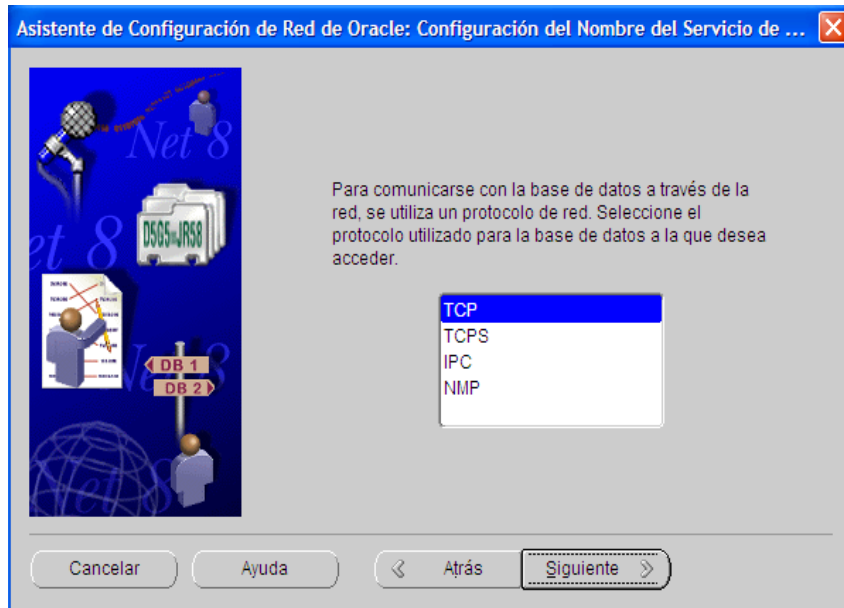


Figura B.10: Selección del protocolo de red

11. Digitar el nombre del host o la dirección IP del servidor y presionar el botón “Siguiente”, así:

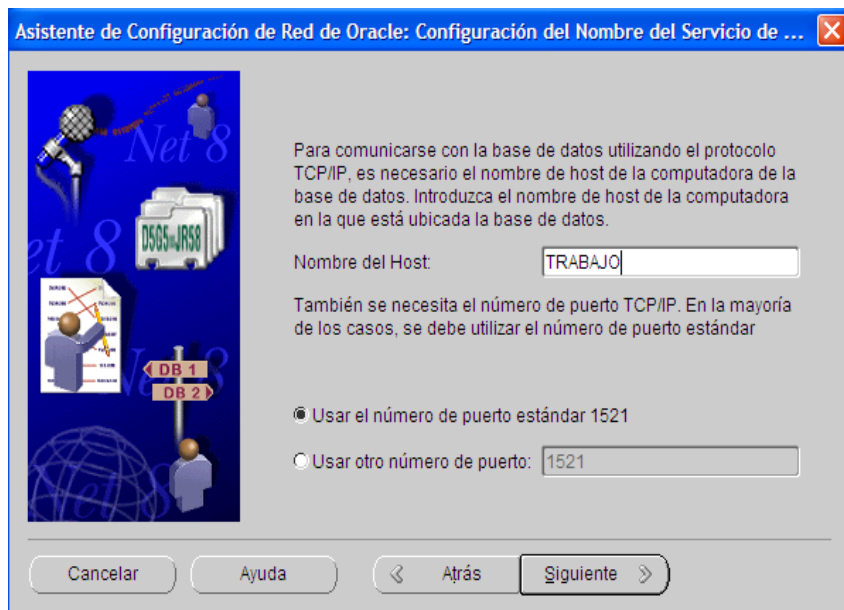


Figura B.11: Ingreso del nombre del host

12. Seleccionar si desea o no realizar la prueba de conexión y presionar el botón “Siguiente”, así:

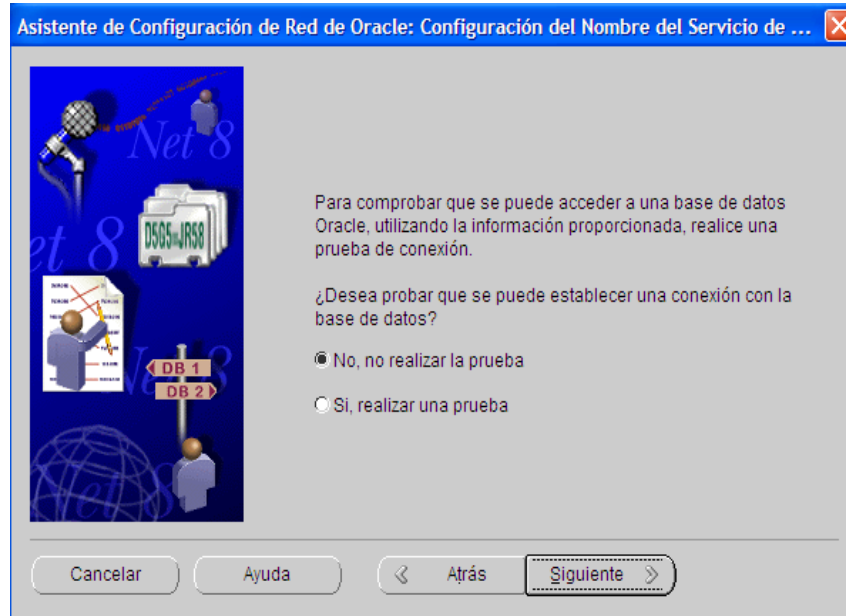


Figura B.12: Probar conexión

## B. CONFIGURACION DEL DIRECTORIO VIRTUAL

Debe crear en la maquina de desarrollo un nuevo directorio virtual con el Alias “Bienen.Business.WebServices”, ya que es el que permitirá el uso de los web services, el nombre con el cual llamara al directorio virtual no puede ser cambiado ya que la referencia al Proxy se lo realiza a ese nombre de directorio

- ✓ Creación del directorio virtual
  - Ingresar al Panel de Control
  - Seleccionar Herramientas Administrativas
  - Escoger Servicios de Internet Information Server
  - Sobre el Sitio Web hacer clic derecho
  - Elegir la opción Nuevo
  - Escoger la opción Directorio Virtual

- Digitar el nombre del directorio virtual (Alias)
- Escribir el siguiente nombre: ***Bienen.Business.WebServices***
- En Directorio: ubicar la carpeta:  
***\\Mobil\Bienen.Business.WebServices***, misma que fue copia del disco de C.A.F. CD CODIGO FUENTE.
- En Permisos del nuevo directorio virtual: dejar los valores predeterminados por parte del Servidor Web.
- Finalizar

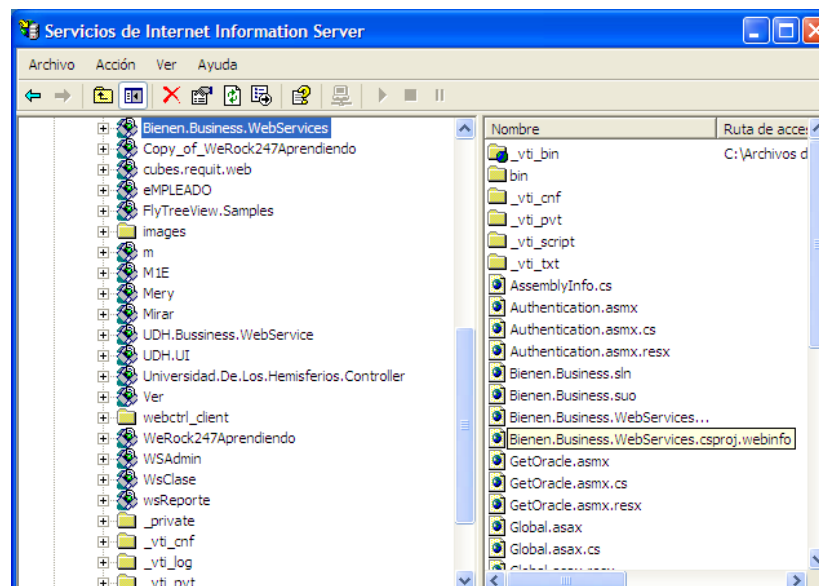


Figura B.13: Servicios de Internet Information Server

## C. CODIFICACION DE LA APLICACION

Para abrir los fuentes de la aplicación, se puede utilizar el Explorador de Windows, ubicarse dentro de la carpeta Mobil y presionar doble clic sobre el archivo Mobil.sln (Archivo de solución en .NET 2003).

El nombre de la Aplicación es ***Mobil*** y contiene los siguientes proyectos:

1. Administración (Proyecto para Windows)
2. Bienen.Business.WebServices (Proyecto de Web Services)
3. Bienes (Proyecto para Pocket PC)
4. DBHelper (Proyecto de Clases)
5. Seguridad (Proyecto de Clases)

## 1. PROYECTO ADMINISTRACION

Contiene todo la interfaz de usuario de los módulos de:

- ✓ Reportes
- ✓ Seguridad

El proyecto además contiene las siguientes Clases

- ✓ **Clase Service**

Crea el Proxy entre las páginas asmx y la aplicación.

- ✓ **Clase Kernel**

Es la que permite la creación de objetos para que pueda ser consumida por la parte visual, es de tipo estática y se la referencia dentro de otra clase o formulario de la siguiente manera:

*Kernel.Instance*

Contiene los métodos que son ejecutados desde los formularios, y las llamadas a los Métodos públicos de las paginas asmx.

### Métodos

- SignIn

Ejecuta la llamada al servicio para verificar la existencia de usuarios

```
public int SignIn(string username, string password)
{
    int respuesta = 0;
    try
```



```

        {
            respuesta =
                Service.Instance.AuthenticationService.AuthenticateUser(username, password);
        }
        catch
        {
        }

        return respuesta;
    }
}

```

- **GetGrant**

Ejecuta la llamada al servicio y solicita los permisos de los usuarios

```

public object[] GetGrant(string username)
{
    int k;
    object[] arg;
    DataRow row;
    DataSet ds = new DataSet();

    ds =
        Service.Instance.AuthenticationService.GetGrant(username);

    arg = new object[ds.Tables[0].Rows.Count];

    for (k = 0; k < ds.Tables[0].Rows.Count; k++)
    {
        row = ds.Tables[0].Rows[k];
        arg[k] = row[0].ToString();
    }

    return arg;
}

```

- **GetUser**

Ejecuta la llamada al servicio y solicita la información de los usuarios

```

public DataSet GetUser()
{
    DataSet ds = new DataSet();

    ds = Service.Instance.UserService.SearchAllUser();

    return ds;
}

```

- SaveUser

Ejecuta la llamada al servicio y solicita la inserción de un usuario

```
public string SaveUser(string cedula, string nombre, string
apellido, string direccion, string telefono)
{
    return Service.Instance.UserService.NewUser(cedula, nombre,
        apellido, direccion, telefono);
}
```

- UpdateUser

Ejecuta la llamada al servicio y solicita la actualización de un usuario

```
public int UpdateUser(string nombre, string apellido, string
direccion, string telefono, string cedula)
{
    return Service.Instance.UserService.UpdateUser(nombre,
        apellido, direccion, telefono, cedula);
}
```

- DeleteUser

Ejecuta la llamada al servicio y solicita la eliminación de un usuario

```
public int DeleteUser(string cedula)
{
    return Service.Instance.UserService.DeleteUser(cedula);
}
```

- ChangePassword

Ejecuta la llamada al servicio y solicita el cambio de clave de un usuario

```
public string ChangePassword(string username, string password)
{
    return
        Service.Instance.UserService.ChangePassword(username,
            password);
}
```

- GetProfile

Ejecuta la llamada al servicio y solicita la información de los perfiles

```
public DataSet GetProfile()
{
    DataSet ds = new DataSet();
}
```

```

        ds = Service.Instance.SecService.GetProfile();

        return ds;
    }

```

- **SaveProfile**

Ejecuta la llamada al servicio y solicita la inserción de un nuevo perfil

```

public int SaveProfile(string descripcion)
{
    return
        Service.Instance.SecService.SaveProfile(descripcion);
}

```

- **UpdateProfile**

Ejecuta la llamada al servicio y solicita la actualización de un perfil

```

public int UpdateProfile(string descripcion, int codigo)
{
    return
        Service.Instance.SecService.UpdateProfile(descripcion,
        codigo);
}

```

- **DeleteProfile**

Ejecuta la llamada al servicio y solicita la eliminación de un perfil

```

public int DeleteProfile(int codigo)
{
    return Service.Instance.SecService.DeleteProfile(codigo);
}

```

- **GetComboPrifile**

Ejecuta la llamada al servicio y solicita la información de perfil

```

public DataSet GetComboProfile()
{
    DataSet ds = new DataSet();

    ds = Service.Instance.SecService.GetComboProfile();

    return ds;
}

```

- **GetProfileModule**

Ejecuta la llamada al servicio y solicita los módulos por perfil

```
public DataSet GetProfileModule(int perfil)
{
    DataSet ds = new DataSet();

    ds = Service.Instance.SecService.GetProfileModule(perfil);

    return ds;
}
```

- **SaveProfileModule**

Ejecuta la llamada al servicio y solicita insertar los módulos al perfil

```
public int SaveProfileModule(int perfil, params object[] arg)
{
    return
        Service.Instance.SecService.SaveProfileModule(perfil, arg);
}
```

- **GetComboUser**

Ejecuta la llamada al servicio y solicita el nombre de los usuarios

```
public DataSet GetComboUser()
{
    DataSet ds = new DataSet();

    ds = Service.Instance.SecService.GetComboUser();

    return ds;
}
```

- **GetProfileAssigned**

Ejecuta la llamada al servicio y solicita el perfil asignado al usuario

```
public object GetProfileAssigned(string cedula)
{
    return
        Service.Instance.SecService.GetProfileAssigned(cedula);
}
```

- SaveProfileAssigned

Ejecuta la llamada al servicio y solicita asignar un perfil a un usuario

```
public int SaveProfileAssigned(string cedula, int perfil)
{
    return
        Service.Instance.SecService.SaveProfileAssigned(cedula,
        perfil);
}
```

### ✓ Clase TraceException

Es heredada de la clase Exception y maneja las excepciones que se pudiesen generar en el momento de producción de la aplicación

#### Métodos

- TraceException

Emite un mensaje al cliente cuando se produce un error

```
public TraceException(string msg, string form) : base()
{
    MessageBox.Show(msg, form, MessageBoxButtons.OK,
    MessageBoxIcon.Information);
    return;
}
```

## 2. PROYECTO Bienes.Business.WebServices

Esta formado por las paginas asmx es la capa de componentes de la aplicación

### ✓ Authentication.asmx

Contiene los métodos que permiten manejar la seguridad de la aplicación.

## Métodos

- AuthenticateUser

Verifica si el usuario existe

```
[WebMethod]
public int AuthenticateUser(string username, string password)
{
    string query;
    int record;

    query = string.Format("Select count(*) From AEUSM_USUMO
Where AEUSM_USERNAME = '{0}' and AEUSM_PASSWORD =
'{1}'", EscapeSqlString(username), EscapeSqlString(password))
;
    record =
Convert.ToInt32(Kernel.Instance.ExecuteScalar(query));

    return record;
}
```

- GetGrant

Obtiene los módulos a los cuales tiene acceso el usuario

```
[WebMethod]
public DataSet GetGrant(string username)
{
    string query;
    DataSet ds = new DataSet();

    query = string.Format("Select MEMOD_MODULO From
MERPM_PERMOD, MEUSP_USUPER, AEUSM_USUMO Where
MERPM_PERMOD.MEPPER_CODIGO = MEUSP_USUPER.MEPPER_CODIGO and
MEUSP_USUPER.AEUSM_CEDULA = AEUSM_USUMO.AEUSM_CEDULA and
AEUSM_USERNAME = '{0}'", username);
    ds = Kernel.Instance.Execute(query, "MERPM_PERMOD");

    return ds;
}
```

- ✓ **GetOracle.asmx**

Contiene los métodos para obtener información de los bienes desde

Oracle

## Métodos

- GetStates

Obtiene los estados de los bienes

```
[WebMethod]
public DataSet GetStates()
{
    string query;
    DataSet ds = new DataSet();

    query = string.Format("Select
    AEEST_CODIGO,AEEST_DESCRIPCION From AEEST_ESTADO Where
    AEEST_MOVIL = '{0}'",1);
    ds = Kernel.Instance.Execute(query, "AEEST_ESTADO");

    return ds;
}
```

- GetPlaces

Obtiene las ubicaciones de los bienes

```
[WebMethod]
public DataSet GetPlaces()
{
    string query;
    DataSet ds = new DataSet();

    query = string.Format("Select CEUGT_ID, CEU_CEUGT_ID,
    CEUUG_DESC From CEUUG_UNIDAD_UNIGTO Where length(CEUGT_ID)
    < '{0}' and CEUGT_ID <> '{1}'", 5, 100);
    ds = Kernel.Instance.Execute(query, "CEUUG_UNIDAD_UNIGTO");

    return ds;
}
```

- GetEmployed

Obtiene los responsables de los bienes

```
[WebMethod]
public DataSet GetEmployed()
{
    string query;
    DataSet ds = new DataSet();

    query = string.Format("Select
    PEEMD_EMPLESPE.CIU_CEDULA,CIU_APELLPAT ||'
    '||CIU_APELLMAT||' '||CIU_NOMBRE1||' '||CIU_NOMBRE2 as
    NOMBRE From MECIU_CIUADADNO,PEEMD_EMPLESPE Where
    MECIU_CIUADADNO.CIU_CEDULA = PEEMD_EMPLESPE.CIU_CEDULA");
    ds = Kernel.Instance.Execute(query, "PEEMD_EMPLESPE");
}
```

```

        return ds;
    }

```

- **GetAsset**

Obtiene la información de los bienes a constatar

```

[WebMethod]
public DataSet GetAsset(int dependencia)
{
    string query;
    DataSet ds = new DataSet();

    query = string.Format("Select
    IVN_CODIGO,IVN_DESINV,IVN_NUMSER,IVN_MARCA,IVN_MODELO,IVN_C
    ONDÍC,IVN_FECALT,INV_UNIDAD,IVN_OBSERV,INV_UNIDAD_ESPE,INV_
    INTERNO,IVN_RESPON,IVN_RESPON2,IVN_RESPON3,IVN_RESPON4,IVN_
    RESPON5 From AEINV_INVENTA Where INV_UNIDAD_ESPE = '{0} '",
    dependencia);
    ds = Kernel.Instance.Execute(query, "AEINV_INVENTA");

    return ds;
}

```

- ✓ **PutOracle.asmx**

Contiene los métodos para actualizar la base Oracle

**Métodos**

- **UpdateAsset**

Actualiza la información de cada de los bienes constatados

```

[WebMethod]
public int UpdateAsset(DataSet ds)
{
    string query;
    int result = 0;

    DataTable dt = ds.Tables[0];
    foreach(DataRow row in dt.Rows)
    {
        string codigo = row[0].ToString();
        string descripcion = row[1].ToString();
        string serie = row[2].ToString();
        string marca = row[3].ToString();
        string modelo = row[4].ToString();
        string estado = row[5].ToString();
        string fecha = row[6].ToString();
        string dia;
        string mes;
        string anio;
    }
}

```



```

        dia = fecha.Substring(0, 2);
        mes = fecha.Substring(3, 2);
        anio = fecha.Substring(6, 4);
        fecha = anio + "/" + mes + "/" + dia;
        string lugar = row[7].ToString();
        string observacion = row[8].ToString();
        string espe = row[9].ToString();
        string tipo = row[10].ToString();
        string respon1 = row[11].ToString();
        string respon2 = row[12].ToString();
        string respon3 = row[13].ToString();
        string respon4 = row[14].ToString();
        string respon5 = row[15].ToString();

        query = string.Format("Update AEINV_INVENTA Set
        IVN_DESINV = '{0}', IVN_NUMSER = '{1}', IVN_MARCA =
        '{2}', IVN_MODELO = '{3}', IVN_CONDIC = '{4}',
        IVN_FECALT = '{5}', IVN_OBSERV = '{6}',
        INV_UNIDAD_ESPE = '{7}', INV_INTERNO = '{8}',
        IVN_RESPON = '{9}', IVN_RESPON2 = '{10}', IVN_RESPON3
        = '{11}', IVN_RESPON4 = '{12}', IVN_RESPON5 = '{13}'
        Where IVN_CODIGO = '{14}'", descripcion, serie,
        marca, modelo, estado, fecha, observacion,
        Convert.ToInt16(espe), tipo, respon1, respon2,
        respon3, respon4, respon5, codigo);
        Kernel.Instance.ExecuteNonQuery(query);
    }

    result = 1;
    return result;
}

```

- **CreateRegister**

Inserta un registro con la acción realiza en cada activo

```

[WebMethod]
public int CreateRegister(DataSet ds)
{
    string query;
    int finalizado = 0;

    DataTable dt = ds.Tables[0];
    foreach(DataRow row in dt.Rows)
    {
        string bien = row[0].ToString();
        string usuario = row[1].ToString();
        string accion = row[2].ToString();
        string fecha = row[3].ToString();
        string sincronizado = row[4].ToString();

        query = string.Format("Insert Into MEREGL_REGISTRO
        values('{0}', '{1}', '{2}', '{3}', '{4}')", bien,
        usuario, accion, fecha, sincronizado);
        finalizado +=Kernel.Instance.ExecuteNonQuery(query);
    }

    return finalizado;
}

```

## ✓ Rpt.asmx

Facilita la presentación de los reportes

## ✓ Sec.asmx

Contiene los métodos que permiten realizar el control de la seguridad

### Métodos

- GetProfile

Obtiene la información de los perfiles

```
[WebMethod]
public DataSet GetProfile()
{
    string query;
    DataSet ds = new DataSet();

    query = string.Format("Select MEPER_CODIGO as CODIGO,
MEPER_DESCRIPCION as DESCRIPCION From MEPER_PERFIL");
    ds = Kernel.Instance.Execute(query, "MEPER_PERFIL");

    return ds;
}
```

- SaveProfile

Inserta la información del perfil

```
[WebMethod]
public int SaveProfile(string descripcion)
{
    string query;

    query = string.Format("Insert Into MEPER_PERFIL
values('{0}','{1}']", CountProfile(), descripcion);

    return
Convert.ToInt16(Kernel.Instance.ExecuteNonQuery(query));
}
```

- UpdateProfile

Actualiza la información de un perfil

```
[WebMethod]
public int UpdateProfile(string descripcion, int codigo)
```

```

{
    string query;

    query = string.Format("Update MEPER_PERFIL Set
MEPER_DESCRIPCION = '{0}' Where MEPER_CODIGO = '{1}'",
descripcion, codigo);

    return
    Convert.ToInt16(Kernel.Instance.ExecuteNonQuery(query));
}

```

- **DeleteProfile**

Elimina la información de un perfil

```

[WebMethod]
public int DeleteProfile(int codigo)
{
    string query;

    query = string.Format("Delete From MEPER_PERFIL Where
MEPER_CODIGO = '{0}'", codigo);

    return
    Convert.ToInt16(Kernel.Instance.ExecuteNonQuery(query));
}

```

- **GetProfileModule**

Obtiene la información de los módulos asignados a un perfil

```

[WebMethod]
public DataSet GetProfileModule(int perfil)
{
    string query;

    query = string.Format("Select MEMOD_MODULO From
MERPM_PERMOD Where MEPER_CODIGO = '{0}'", perfil);

    return Kernel.Instance.Execute(query, "MERPM_PERMOD");
}

```

- **SaveProfileModule**

Inserta la información módulos a un perfil

```

[WebMethod]
public int SaveProfileModule(int perfil, params object[] arg)
{
    string query;
    int k;

```

```

DeleteProfileModule(perfil);

for (k = 0; k < arg.Length; k++)
{
    query = string.Format("Insert Into MERPM_PERMOD
values('{0}', '{1}')" , perfil, arg[k].ToString());

    Kernel.Instance.ExecuteNonQuery(query);
}

return 1;
}

```

- **GetComboProfile**

Obtiene la información de los perfiles

```

[WebMethod]
public DataSet GetComboProfile()
{
    string query;

    query = string.Format("Select * From MEPER_PERFIL");

    return Kernel.Instance.Execute(query, "MEPER_PERFIL");
}

```

- **GetComboUser**

Obtiene los nombres de los usuarios del sistema

```

[WebMethod]
public DataSet GetComboUser()
{
    string query;

    query = string.Format("Select AEUSM_CEDULA as CEDULA,
AEUSM_NOMBRE ||' '|| AEUSM_APELLIDO as USUARIO from
AEUSM_USUMO");

    return Kernel.Instance.Execute(query, "AEUSM_USUMO");
}

```

- **GetProfileAssigned**

Obtiene el perfil asignado a un usuario

```

[WebMethod]
public object GetProfileAssigned(string cedula)
{
    string query;

```

```

query = string.Format("Select MEPER_DESCRIPCION From
MEPER_PERFIL, MEUSP_USUPER Where MEPER_PERFIL.MEPER_CODIGO
= MEUSP_USUPER.MEPER_CODIGO and AEUSM_CEDULA = '{0}'",
cedula);

```

```

return Kernel.Instance.ExecuteScalar(query);

```

```

}

```

- SaveProfileAssigned

Inserta el perfil asignado a un usuario

```

[WebMethod]
public int SaveProfileAssigned(string cedula, int perfil)
{
    string query;

    DeleteProfileAssigned(cedula);

    query = string.Format("Insert Into MEUSP_USUPER
values('{0}','{1}')" , cedula, perfil);

    return
    Convert.ToInt16(Kernel.Instance.ExecuteNonQuery(query));
}

```

- ✓ **User.asmx**

Contiene los métodos que permiten la manipulación del usuario y clave

### Métodos

- NewUser

Crea un nuevo usuario

```

[WebMethod]
public string NewUser(string cedula, string nombre, string
apellido, string direccion, string telefono)
{
    string query;
    string username = String.Empty;

    if (ExistUser(cedula) > 0)
    {
        username = "Un usuario se encuentra registrado en la
base";
    }
    else
    {
        username = NewUsername(nombre, apellido, cedula);
        query = string.Format("Insert Into AEUSM_USUMO
values('{0}','{1}','{2}','{3}','{4}','{5}','{6}')" ,
EscapeSQLString(cedula), EscapeSQLString(nombre),
EscapeSQLString(apellido), EscapeSQLString(direccion),

```

```

        EscapeSQLString(telefono), username,
        Password(cedula));
        Kernel.Instance.ExecuteNonQuery(query);
    }
    return username;
}

```

- SearchAllUser

Obtiene la información de todos los usuarios

```

[WebMethod]
public DataSet SearchAllUser()
{
    string query;
    DataSet ds = new DataSet();

    query = string.Format("Select AEUSM_CEDULA as Cedula,
    AEUSM_NOMBRE as Nombre, AEUSM_APELLIDO as Apellido,
    AEUSM_DIRECCION as Direccion, AEUSM_TELEFONO as Telefono
    From AEUSM_USUMO");
    ds = Kernel.Instance.Execute(query, "AEUSM_USUMO");

    return ds;
}

```

- DeleteUser

Elimina un la información del usuario

```

[WebMethod]
public int DeleteUser(string cedula)
{
    string query;

    query = string.Format("Delete From AEUSM_USUMO Where
    AEUSM_CEDULA = '{0}'", cedula);

    return
    Convert.ToInt16(Kernel.Instance.ExecuteNonQuery(query));
}

```

- UpdateUser

Actualiza la información del usuario

```

[WebMethod]
public int UpdateUser(string nombre, string apellido, string
direccion, string telefono, string cedula)
{

```

```

        string query;

        query = string.Format("Update AEUSM_USUMO Set AEUSM_NOMBRE
        = '{0}', AEUSM_APELLIDO = '{1}', AEUSM_DIRECCION = '{2}',
        AEUSM_TELEFONO = '{3}' Where AEUSM_CEDULA = '{4}'",
        EscapeSQLString(nombre), EscapeSQLString(apellido),
        EscapeSQLString(direccion), EscapeSQLString(telefono),
        EscapeSQLString(cedula));

        return
        Convert.ToInt16(Kernel.Instance.ExecuteNonQuery(query));
    }

```

- **ChangePassword**

Cambia el password de un usuario

```

[WebMethod]
public string ChangePassword(string username, string password)
{
    string query;

    query = string.Format("Update AEUSM_USUMO Set
    AEUSM_PASSWORD = '{0}' Where AEUSM_USERNAME = '{1}'",
    password, username);

    Kernel.Instance.ExecuteNonQuery(query);

    return "Cambiado";
}

```

Todos los Métodos que pueden ser consumidos por las aplicaciones a través de la generación del proxy están precedidos por la palabra reservada [WebMethod]

### 3. PROYECTO Bienes

Contiene la interfaz de usuario del aplicativo que funcionará en los dispositivos móviles y además posee las siguientes Clases.

## ✓ Clase Mery

Esta clase maneja la capa de datos (SQL CE) contiene los siguientes

### **Métodos**

- OpenConnection

Abre la conexión a la base de datos local

```
protected void OpenConnection()
{
    connection = (connection == null) ? new
    SqlConnection(connectionString) : connection;

    if (connection.State == ConnectionState.Closed)
    {
        connection.Open();
    }
}
```

- CloseConnection

Cierra la conexión a la base de datos

```
protected void CloseConnection()
{
    if (connection != null)
    {
        connection.Close();
    }
    connection = null;
}
```

- GetCommand

Crea un comando que contiene la sentencia sql y el objeto conexión

```
protected SqlCommand GetCommand(string sql)
{
    OpenConnection();

    if (command == null)
    {
        command = new SqlCommand();
        command.Connection = connection;
        command.CommandType = CommandType.Text;
    }

    command.CommandText = sql;

    return command;
}
```



- **GetAdapter**

Crea un adaptador que instancia un espacio de memoria físico

```
protected SqlCeDataAdapter GetAdapter(string sql)
{
    OpenConnection();
    return new SqlCeDataAdapter(sql, connection);
}
```

- **Execute**

Trae el contenido de una tabla y su estructura en un DataSet

```
public DataSet Execute(string sql, string tableName)
{
    DataSet ds = new DataSet();
    SqlCeDataAdapter da = GetAdapter(sql);
    da.Fill(ds, tableName);
    return ds;
}
```

- **ExecuteNonQuery**

Ejecuta una sentencia sql en el motor de datos

```
public int ExecuteNonQuery(string sql)
{
    SqlCeCommand cmd = GetCommand(sql);
    string men;
    try
    {
        cmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        men = ex.Message;
    }
    return 1;
}
```

- **ExecuteScalar**

Trae el resultado de un Select, primera fila primera columna

```
public object ExecuteScalar(string sql)
{
    SqlCeCommand cmd = GetCommand(sql);
    return cmd.ExecuteScalar();
}
```

## ✓ Clase LocalStore

Esta clase hereda los Métodos de la clase de datos y ejecuta todas las acciones que la aplicación requiere realizar en la base de datos local.

### Métodos

- Initialize

Crea la base de datos así como las tablas

```
public void Initialize(string dbfile)
{
    string query = null;

    try
    {
        dbname = dbfile;

        queries = new SQLQueryManager();

        connectionString = string.Format("data source =
        '{0}'", DatabasePath);

        if (!File.Exists(DatabasePath))
        {
            SqlCeEngine cengine = new
            SqlCeEngine(connectionString);
            cengine.CreateDatabase();
        }

        if (!TableExists("CEEST_ESTADO"))
        {
            query = queries.PrepareQuery("Create table
            CEEST_ESTADO(CEEST_CODIGO nchar(1) primary
            key,CEEST_DESCRIPCION nvarchar(100))");
            ExecuteNonQuery(query);
        }

        if (!TableExists("CEUSU_USUARIO"))
        {
            query = string.Format("Create table
            CEUSU_USUARIO (CEUSU_ID nvarchar(12) primary
            key,CEUSU_USERNAME nvarchar(50),CEUSU_PWD
            nvarchar(50),CEUSU_CREADO bigint)");
            ExecuteNonQuery(query);
        }

        if (!TableExists("CEBIE_BIEN"))
        {
            query = string.Format("Create table CEBIE_BIEN
            (CEBIE_CODIGO nvarchar(20) primary
            key,CEBIE_DESCRIPCION nvarchar(250),CEBIE_SERIE
            nvarchar(150),CEBIE_MARCA
            nvarchar(150),CEBIE_MODELO
```

```

        nvarchar(150),CEEST_CODIGO nchar(1),CEBIE_FECHA
        datetime,CEDEP_CODIGO
        nvarchar(10),CEBIE_OBSERVACION
        nvarchar(250),CEDEP_ESPE numeric,CEBIE_TIPO
        nvarchar(1),CEBIE_RESPON1 nvarchar(15),
        CEBIE_RESPON2 nvarchar(15),CEBIE_RESPON3
        nvarchar(15),CEBIE_RESPON4 nvarchar(15),
        CEBIE_RESPON5 nvarchar(15))");
        ExecuteNonQuery(query);
    }

    if (!TableExists("CEDEP_DEPENDENCIA"))
    {
        query = string.Format("Create table
        CEDEP_DEPENDENCIA (CEDEP_CODIGO numeric primary
        key,CEDEP_ID numeric,CEDEP_DESCRIPCION
        nvarchar(250))");
        ExecuteNonQuery(query);
    }

    if (!TableExists("CERES_RESPONSABLE"))
    {
        query = string.Format("Create table
        CERES_RESPONSABLE (CERES_CEDULA nvarchar(15)
        primary key,CERES_NOMBRE nvarchar(250))");
        ExecuteNonQuery(query);
    }

    if (!TableExists("CEREG_REGISTRO"))
    {
        query = string.Format("Create table
        CEREG_REGISTRO (CEREG_BIEN nvarchar(15),
        CEREG_USUARIO nvarchar(7),CEREG_ACCION
        nvarchar(20),CEREG_FECHA datetime,
        CEREG_SINCRONIZADO numeric)");
        ExecuteNonQuery(query);
    }
}
catch (Exception ex)
{
    throw new TraceException("LocalStore", "Initialize",
    query, ex);
}
}

```

- SaveInformation

Almacena la información de los bienes y de los parámetros

```

public void SaveInformation(DataSet ds)
{
    string query = string.Empty;

    try
    {
        if (ds.Tables.Contains("AEEST_ESTADO"))
        {

```

```

query = queries.PrepareQuery("Delete From
CEEST_ESTADO");
ExecuteNonQuery(query);

DataTable estado = ds.Tables["AEEST_ESTADO"];
foreach(DataRow row in estado.Rows)
{
    string codigo =
    row["AEEST_CODIGO"].ToString();
    string descripcion =
    row["AEEST_DESCRIPCION"].ToString();
    query = queries.PrepareQuery("Insert Into
CEEST_ESTADO values ('{0}', '{1}']", codigo,
descripcion);
    ExecuteNonQuery(query);
}
}

if(ds.Tables.Contains("CEUUG_UNIDAD_UNIGTO"))
{
    query = queries.PrepareQuery("Delete From
CEDEP_DEPENDENCIA");
    ExecuteNonQuery(query);

    DataTable dependencia =
    ds.Tables["CEUUG_UNIDAD_UNIGTO"];
    foreach(DataRow row in dependencia.Rows)
    {
        string codigo =
        row["CEUGT_ID"].ToString();
        string id =
        row["CEU_CEUGT_ID"].ToString();
        string descripcion =
        row["CEUUG_DESC"].ToString();
        query = queries.PrepareQuery("Insert Into
CEDEP_DEPENDENCIA
values ('{0}', '{1}', '{2}']", codigo, id,
descripcion);
        ExecuteNonQuery(query);
    }
}

if(ds.Tables.Contains("PEEMD_EMPLESPE"))
{
    query = queries.PrepareQuery("Delete From
CERES_RESPONSABLE");
    ExecuteNonQuery(query);

    DataTable responsable =
    ds.Tables["PEEMD_EMPLESPE"];
    foreach(DataRow row in responsable.Rows)
    {
        string cedula = row[0].ToString();
        string nombre = row[1].ToString();
        query = queries.PrepareQuery("Insert Into
CERES_RESPONSABLE values ('{0}', '{1}']",
cedula, nombre);
        ExecuteNonQuery(query);
    }
}
}

```

```

if(ds.Tables.Contains("AEINV_INVENTA"))
{
    DataTable bien = ds.Tables["AEINV_INVENTA"];
    foreach(DataRow row in bien.Rows)
    {
        string codigo =
        row["IVN_CODIGO"].ToString();
        string descripcion =
        row["IVN_DESINV"].ToString();
        string serie =
        row["IVN_NUMSER"].ToString();
        string marca =
        row["IVN_MARCA"].ToString();
        string modelo =
        row["IVN_MODELO"].ToString();
        string estado = "N";
        string fecha =
        row["IVN_FECALT"].ToString();
        string dia = fecha.Substring(0,2);
        string mes = fecha.Substring(3,2);
        string anio = fecha.Substring(6, 4);
        fecha = mes + "/" + dia + "/" + anio;
        string lugar =
        row["INV_UNIDAD"].ToString();
        string observacion =
        row["IVN_OBSERV"].ToString();
        string espe =
        row["INV_UNIDAD_ESPE"].ToString();
        string tipo =
        row["INV_INTERNO"].ToString();
        string respon1 =
        row["IVN_RESPON"].ToString();
        string respon2 =
        row["IVN_RESPON2"].ToString();
        string respon3 =
        row["IVN_RESPON3"].ToString();
        string respon4 =
        row["IVN_RESPON4"].ToString();
        string respon5 =
        row["IVN_RESPON5"].ToString();
        query = queries.PrepareQuery("Insert Into
        CEBIE_BIEN values
        ('{0}','{1}','{2}','{3}','{4}','{5}','{6}'
        ,'{7}','{8}','{9}','{10}','{11}','{12}',
        '{13}','{14}','{15}')", codigo,
        descripcion, serie, marca, modelo,
        estado, fecha, lugar, observacion, espe,
        tipo, respon1, respon2, respon3, respon4,
        respon5);
        ExecuteNonQuery(query);
    }
}
}
catch(Exception ex)
{
    throw new TraceException("LocalStore",
    "SaveInformation", query, ex);
}
}

```

- **GetStates**

Obtiene los estados de los bienes

```
public DataSet GetStates ()
{
    string query = null;

    try
    {
        query = queries.PrepareQuery("Select * From
CEEST_ESTADO");
        return Execute(query, "CEEST_ESTADO");
    }
    catch(Exception ex)
    {
        throw new TraceException("LocalStore", "GetEstados",
query, ex);
    }
}
```

- **GetPlaces**

Obtiene los lugares donde se encuentran los bienes

```
public DataSet GetPlaces (string condicion)
{
    string query = null;

    try
    {
        query = queries.PrepareQuery("Select
CEDEP_DESCRIPCION as DEPENDENCIA,CEDEP_CODIGO as
CODIGO, CEDEP_ID as ID From CEDEP_DEPENDENCIA Where
CEDEP_DESCRIPCION like '{0}'", condicion);
        return Execute(query, "CEDEP_DEPENDENCIA");
    }
    catch(Exception ex)
    {
        throw new TraceException("LocalStore",
"GetDependencias", query, ex);
    }
}
```

- **GetEmployed**

Obtiene los empleados que son responsables de los bienes

```
public DataSet GetEmployed(string condicion, string columna)
{
    string query = null;

    try
    {
```

```

        query = queries.PrepareQuery("Select CERES_NOMBRE as
        NOMBRE ,CERES_CEDULA as CEDULA From CERES_RESPONSABLE
        Where {0} like '{1}'", columna, condicion);
        return Execute(query, "CERES_RESPONSABLE");
    }
    catch (Exception ex)
    {
        throw new TraceException("LocalStore",
        "GetResponsable", query, ex);
    }
}

```

- **GetAsset**

Obtiene la información del bien

```

public DataSet GetAsset(string codigo)
{
    string query = null;

    try
    {
        query = queries.PrepareQuery("Select * From
        CEBIE_BIEN,CEEST_ESTADO,CEDEP_DEPENDENCIA Where
        CEEST_ESTADO.CEEST_CODIGO = CEBIE_BIEN.CEEST_CODIGO
        and CEDEP_DEPENDENCIA.CEDEP_CODIGO = CEDEP_ESPE and
        CEBIE_CODIGO = '{0}'", codigo);
        return Execute(query, "CEBIE_BIEN");
    }
    catch (Exception ex)
    {
        throw new TraceException("LocalStore", "GetBien",
        query, ex);
    }
}

```

- **GetAsset**

Obtiene el número de bienes

```

public int GetAsset()
{
    string query = null;
    int result = 0;

    try
    {
        query = queries.PrepareQuery("Select count(*) From
        CEBIE_BIEN");
        return
        Convert.ToInt16(ExecuteScalar(query).ToString());
    }
    catch (Exception ex)
    {
        throw new TraceException("LocalStore", "GetBien",
        query, ex);
    }
}

```

```
}
```

- **GetEmployed**

### Obtiene la información del empleado

```
public string GetEmployed(string cedula)
{
    string query = null;

    try
    {
        query = queries.PrepareQuery("Select CERES_NOMBRE
        From CERES_RESPONSABLE Where CERES_CEDULA = '{0}'",
        cedula);
        return ExecuteScalar(query).ToString();
    }
    catch (Exception ex)
    {
        throw new TraceException("LocalStore",
        "GetResponsables", query, ex);
    }
}
```

- **UpdateAsset**

### Actualiza la información del bien

```
public int UpdateAsset(string descripcion, string serie, string
marca, string modelo, string estado, string fecha, string lugar,
string observacion, int espe, string tipo, DataTable dt, string
codigo)
{
    string query = "";
    string respon1;
    string respon2;
    string respon3;
    string respon4;
    string respon5;
    DataRow row;

    try
    {
        row = dt.Rows[0];
        respon1 = row[1].ToString();
        row = dt.Rows[1];
        respon2 = row[1].ToString();
        row = dt.Rows[2];
        respon3 = row[1].ToString();
        row = dt.Rows[3];
        respon4 = row[1].ToString();
        row = dt.Rows[4];
        respon5 = row[1].ToString();

        query = queries.PrepareQuery("Update CEBIE_BIEN Set
        CEBIE_DESCRIPCION = '{0}', CEBIE_SERIE = '{1}',
```



```

        CEBIE_MARCA = '{2}', CEBIE_MODELO = '{3}',
        CEEST_CODIGO = '{4}', CEBIE_FECHA = '{5}',
        CEDEP_CODIGO = '{6}', CEBIE_OBSERVACION = '{7}',
        CEDEP_ESPE = '{8}', CEBIE_TIPO = '{9}', CEBIE_RESPON1
        = '{10}', CEBIE_RESPON2 = '{11}', CEBIE_RESPON3 =
        '{12}', CEBIE_RESPON4 = '{13}', CEBIE_RESPON5 =
        '{14}' Where CEBIE_CODIGO = '{15}'", descripcion,
        serie, marca, modelo, estado, fecha, lugar,
        observacion, espe, tipo, respon1, respon2, respon3,
        respon4, respon5, codigo);
        return ExecuteNonQuery(query);
    }
    catch (Exception ex)
    {
        throw new TraceException("LocalStore", "UpdateBien",
        query, ex);
    }
}

```

- SaveRegister

Inserta la acción realizada en el bien

```

public int SaveRegister(string bien, string username, string
accion)
{
    string query = "";
    string registro;
    string dia;
    string mes;
    string anio;

    int sincronizado = 1;

    try
    {
        registro = Convert.ToString(DateTime.Now);
        dia = registro.Substring(0, 2);
        mes = registro.Substring(3, 2);
        anio = registro.Substring(6, 4);
        registro = mes + "/" + dia + "/" + anio;

        query = queries.PrepareQuery("Insert Into
CEREG_REGISTRO
values('{0}','{1}','{2}','{3}','{4}')", bien,
username, accion, registro, sincronizado);
        return ExecuteNonQuery(query);
    }
    catch (Exception ex)
    {
        throw new TraceException("LocalStore",
"SaveRegistro", query, ex);
    }
}

```

- GetSynchronization

Obtiene los bienes que serán actualizados en Oracle

```
public DataSet GetSynchronization()
{
    string query = "";

    try
    {
        query = queries.PrepareQuery("Select * From
CEBIE_BIEN, CEREG_REGISTRO Where CEBIE_CODIGO =
CEREG_BIEN and CEREG_SINCRONIZADO = '{0}' and
CEREG_ACCION = '{1}'", 1, "ACTUALIZADO");
        return Execute(query, "CEBIE_BIEN");
    }
    catch(Exception ex)
    {
        throw new TraceException("LocalStore",
"GetSincronzacion", query, ex);
    }
}
```

- GetRegister

Obtiene la información para ser insertada en la base central

```
public DataSet GetRegister()
{
    string query = "";

    try
    {
        query = queries.PrepareQuery("Select * From
CEREG_REGISTRO Where CEREG_SINCRONIZADO = '{0}'", 1);
        return Execute(query, "CEREG_REGISTRO");
    }
    catch(Exception ex)
    {
        throw new TraceException("LocalStore", "GetRegistro",
query, ex);
    }
}
```

- UpdateRegister

Actualiza el registro que fue transferido

```
public int UpdateRegister()
{
    string query = "";

    try
```

```

    {
        query = queries.PrepareQuery("Update CEREГ_REGISTRO
set CEREГ_SINCRONIZADO = '{0}' Where
CEREГ_SINCRONIZADO = '{1}'", 0, 1);
return ExecuteNonQuery(query);
    }
    catch (Exception ex)
    {
        throw new TraceException("LocalStore",
            "UpdateRegistro", query, ex);
    }
}
}

```

### ✓ Clase Service

Creas un Proxy que enlaza las referencias a las paginas asmx que contienen la lógica de negocio

### ✓ Clase Setting

Creas dinámicamente un archivo xml, el cual contiene los campos que se encuentran en la clase que se explica a continuación

#### Métodos

- Write

Escribe el archivo Xml

```

public void Write()
{
    try
    {
        StreamWriter escribir =
File.CreateText(direccionArchivo);
escribir.WriteLine("<configuration>");
escribir.WriteLine("\t<appSettings>");

        IDictionaryEnumerator indicador =
lista.GetEnumerator();
while (indicador.MoveNext())
        {
            escribir.WriteLine("\t\t<add key=\"{0}\"
value=\"{1}\" />",
indicador.Key.ToString(),
indicador.Value);
        }

        escribir.WriteLine("\t</appSettings>");
escribir.WriteLine("</configuration>");
    }
}

```

```

        escribir.Close();
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex.Message);
    }
}

```

- **Read**

### Lee el archivo Xml

```

public void Read()
{
    try
    {
        lista.Clear();

        for (int i=0; i < atributosIniciales.GetLength(0);
            i++)
            lista[atributosIniciales[i,0]] =
                atributosIniciales[i,1];

        XmlTextReader leer = new
            XmlTextReader(direccionArchivo);

        while (leer.Read())
        {
            if ((leer.NodeType == XmlNodeType.Element) &&
                (leer.Name == "add"))
                lista[leer.GetAttribute("key")] =
                    leer.GetAttribute("value");
        }
        leer.Close();
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex.Message);
    }
}

```

- ✓ **Clase SettingValues**

Contiene dos clases

- **Clase Campos**

Contiene los campos que se almacenan en el Xml

- **Clase SetInitialValues**

Inicializa a cada uno de los campos

- ✓ **Clase SQLQueryManager**

Prepara la sentencia sql que se ejecuta en el motor de datos evitando que se infiltre sql inject.

**Métodos**

- PrepareQuery

Evita que la sentencia no pueda ser ejecutada

```
public string PrepareQuery(string query, params object[] arg)
{
    object[] temp = new object[arg.Length];

    for (int i = 0; i < arg.Length; i++)
    {
        temp[i] =
            Tool.Instance.EscapeSQLString(arg[i].ToString());
    }

    return string.Format(query, temp);
}
```

- ✓ **Clase Tool**

Es una clase estática que contiene Métodos que ayudan a conocer si la aplicación está en línea o fuera de línea.

- ✓ **Clase TraceException**

Esta clase es heredada de Exception y maneja el nivel de excepciones

- ✓ **Clase Kernel**

Contiene los Métodos que enlaza la capa de negocio con la de presentación.

## Métodos

- SignIn

Ejecuta la llamada al servicio que verifica la existencia del usuario

```
public void SignIn(string username, string password)
{
    DataSet ds = new DataSet();
    int numeroUsuario;

    try
    {
        numeroUsuario =
            Service.Instance.AuthenticationService.AuthenticateUser(
                username, Seguridad.Encrypt(password));

        if (numeroUsuario == 0)
        {
            throw new MessageException("Username o Password
                incorrectos", "Kernel");
        }

        string db = "Constatacion.sdf";

        store = new LocalStore(db);

        Setting.Instance.SetValue(Fields.User, username);

        store.CacheUser(store.UserID(Service.Instance.AuthenticationService
            .GetAuthenticate(username)), username,
            Seguridad.Encrypt(password));
    }
    catch (System.Web.Services.Protocols.SoapException sox)
    {
        throw sox;
    }
    catch (Exception ex)
    {
        throw new TraceException("Kernel", "SignIn", "", ex);
    }
}
```

- DownloadTable

Ejecuta la llamada al servicio que consulta la tablas de parámetros

```
public void DownloadTable()
{
    try
    {
        if (store != null)
        {
            DataSet estado =
                Service.Instance.GetService.GetStates();
            store.SaveInformation(estado);
        }
    }
}
```

```

        DataSet dependencia =
        Service.Instance.GetService.GetPlaces();
        store.SaveInformation(dependencia);

        DataSet responsable =
        Service.Instance.GetService.GetEmployed();
        store.SaveInformation(responsable);
    }
}
catch (Exception ex)
{
    throw new TraceException("Kernel", "DownloadTable",
    "", ex);
}
}

```

- CreateOracle

Ejecuta la llamada al servicio que actualiza la información en la base

```

public int CreateOracle(DataSet ds)
{
    return Service.Instance.PutService.CreateRegister(ds);
}

```

#### 4. PROYECTO DBHelper

Maneja la capa de datos y acciones que se realizan sobre la base de datos

ORACLE contiene la siguiente clase:

##### ✓ Clase SqlHelper

Permite la comunicación con la base de datos Oracle

##### Métodos

- Open

Abre la conexión a la base de datos local

```

public void Open()
{
    connection = (connection == null) ? new
    OleDbConnection(connectionString) : connection;
    if (connection.State == ConnectionState.Closed)
    {
        connection.Open();
    }
}

```

- Close

Cierra la conexión a la base de datos

```
public void Close()
{
    if (connection != null)
    {
        connection.Close();
    }
}
```

- GetCommand

Crema un comando que contiene la sentencia sql y el objeto conexión

```
protected OleDbCommand GetCommand(string sql)
{
    Open();

    if (command == null)
    {
        command = new OleDbCommand();
        command.Connection = connection;
        command.CommandType = CommandType.Text;
    }

    command.CommandText = sql;
    return command;
}
```

- GetAdapter

Crema adaptador que instancia un espacio de memoria físico

```
protected OleDbDataAdapter GetAdapter(string sql)
{
    Open();
    return new OleDbDataAdapter(sql, connection);
}
```

- Execute

Trae el contenido de una tabla y su estructura en un DataSet

```
public DataSet Execute(string sql, string tableName)
{
    DataSet ds = new DataSet();
    OleDbDataAdapter da = GetAdapter(sql);
    da.Fill(ds, tableName);
    return ds;
}
```



- **ExecuteNonQuery**

Consumes a sql statement in the data engine

```
public int ExecuteNonQuery(string sql)
{
    OleDbCommand cmd = GetCommand(sql);
    return cmd.ExecuteNonQuery();
}
```

- **ExecuteScalar**

Returns the result of a Select, first row first column

```
public object ExecuteScalar(string sql)
{
    OleDbCommand cmd = GetCommand(sql);
    return cmd.ExecuteScalar();
}
```

## 5. PROYECTO Seguridad

Realizes encryption and decryption of the user key.

Contains a security class and contains the following methods

- **Encrypt**

Realizes the conversion to a data string of 64 bits

```
public string Encrypt(string text)
{
    try
    {
        byte[] bytes = Encoding.ASCII.GetBytes(text);
        return Convert.ToBase64String(bytes, 0,
            bytes.Length);
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.Write(ex.Message);
        return String.Empty;
    }
}
```

- **Decrypt**

Realizes the conversion to a readable string

```
public string Decrypt(string text)
{
    try
```

```
{
    byte[] bytes = Convert.FromBase64String(text);
    return Encoding.ASCII.GetString(bytes, 0,
        bytes.Length);
}
catch (Exception ex)
{
    System.Diagnostics.Debug.WriteLine(ex.Message);
    return String.Empty;
}
}
```

## **ANEXO C**

# **MANUAL DE USUARIO**

## MANUAL DE USUARIO

### Solución Móvil

#### Ingreso al sistema

En el opción Start pulse y luego pulsar en **CF**

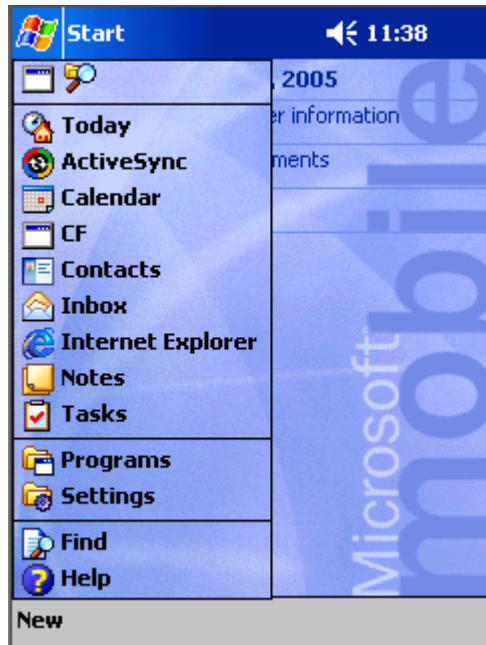


Figura C.1: Inicio del Sistema parte Móvil

A continuación se despliega la pantalla autenticación de usuario



Figura C.2: Ventana de Logon

Digitar el usuario y la contraseña respectiva y luego pulsar en la opción

▶ **Iniciar Sesión**

La siguiente figura indica que se está manteniendo una conexión al Web Service



Figura C.3: Conexión con el Web Service

## Mensajes de Error de Ingreso al Sistema

- El siguiente error se despliega cuando el nombre de usuario o la contraseña están incorrectos

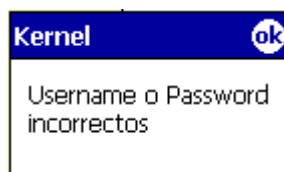


Figura C.4: Mensaje de Error: Username o Password incorrectos

- El siguiente error se despliega cuando el usuario no tiene permisos para acceder a la aplicación

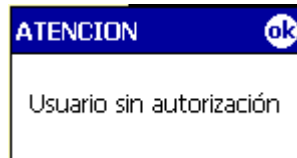


Figura C.5: Mensaje de Error: Usuario sin autorización

### Ventana de Procesos

En esta ventana se realizan las siguientes actividades: Cerrar Sesión, Descargar Base, Constatación y Sincronizar, que se explicarán a continuación.

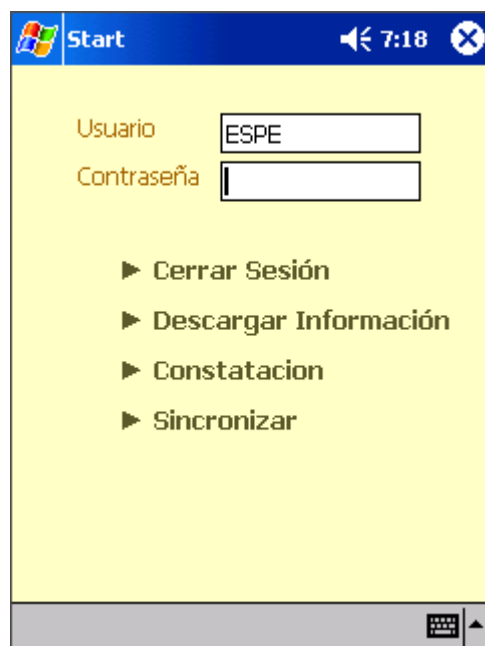


Figura C.6: Ventana de Procesos

#### ✓ Cerrar Sesión

▶ Cerrar Sesión

Cierra la sesión actual y presenta la siguiente pantalla:

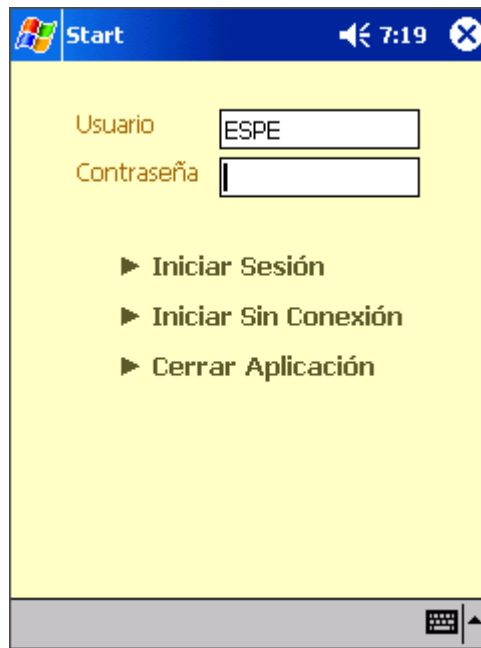


Figura C.7: Salida de Sesión

✓ **Iniciar Sesión**

▶ **Iniciar Sesión**

Permite realizar la autenticación del usuario de forma remota

✓ **Iniciar Sin Conexión**

▶ **Iniciar Sin Conexión**

Permite realizar la autenticación del usuario de forma local

Una vez que se ha autenticado el usuario localmente, aparece habilitado las opciones, así:

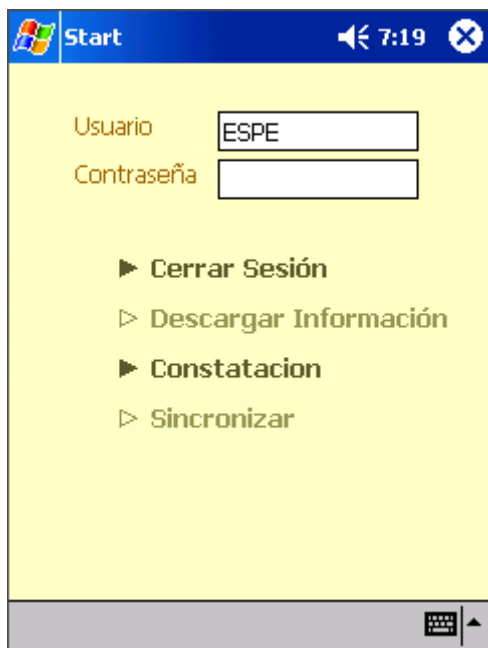


Figura C.8: Ventana de logeo fuera de línea

✓ **Cerrar Aplicación**

▶ Cerrar Aplicación

Cierra la aplicación

✓ **Descargar Base**

▶ Descargar base

Esta ventana permite copiar la información de la base de datos Oracle ubicada en el servidor y la almacena en una base de datos local SQL CE localizada en el dispositivo móvil, esto permitirá realizar el manejo de la información sin necesidad de mantener una conexión al servidor.



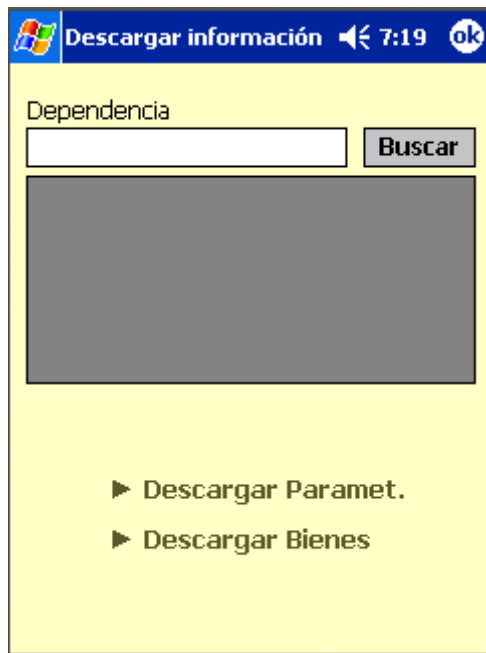


Figura C.9: Ventana de Descarga de Información

Se debe primero pulsar en la opción “Descargar parámetros” **▶ Descargar Paramet.** para que se almacene la información de los estados del bien y demás tablas de parámetros que permitirán obtener la información necesario del bien en cada uno de los dispositivos móviles, de las dependencias de la ESPE y de los empleados administrativos de la ESPE.

Para buscar una dependencia se debe digitar el texto a buscar y luego pulsar el botón **Buscar** , de esa manera se mostrara todas las dependencias que coincidan con el texto digitado, permitiendo seleccionar de cual de ellas se tomara la información para almacenarla en el dispositivo móvil.

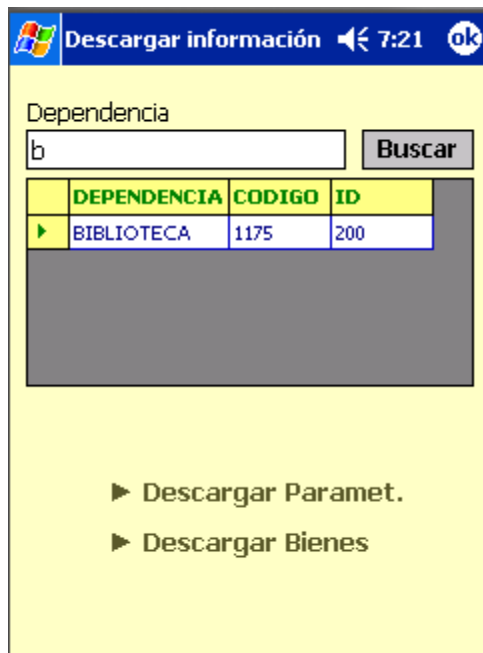


Figura C.10: Búsqueda de una determinada dependencia

La opción “Descargar Bienes” **▶ Descargar Bienes** copia la información de los bienes de una dependencia seleccionada al dispositivo móvil

### Mensajes de Error en Descargar Base

- El siguiente error se despliega cuando no se ha ingresado ningún parámetro de búsqueda en el campo dependencia

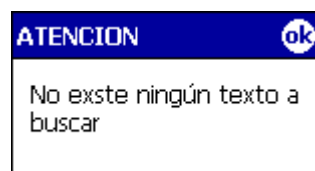


Figura C.11: Mensaje de Error: No existe ningún texto a buscar

- El siguiente error aparece cuando no se ha seleccionado ninguna dependencia

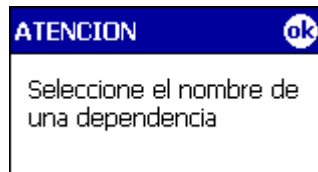


Figura C.12: Mensaje de Error: Seleccione el nombre de una dependencia

## ✓ Constatación

### ▶ Constatacion

Si no se ha descargado la información de los bienes aparecerá el siguiente mensaje:

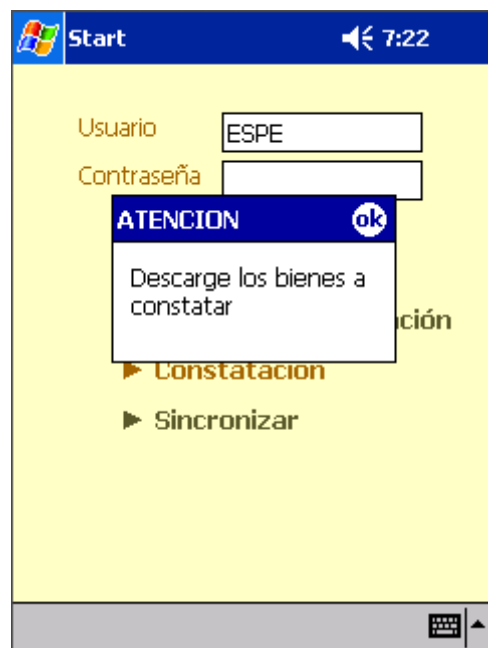


Figura C.13: Mensaje de no existencia de bienes

Esta ventana muestra la información de un bien, digitando su código de barras o utilizando el lector de código de barras.

Constatación 8:57 ok

Código  Buscar

Nombre

Serie

Marca

Modelo

Observacion

Tipo  Activo  Control

Información Estado Responsable Ubicación

▶ Grabar ▶ Cancelar ▶ Salir

Figura C.14: Ventana de constatación de bienes

Para buscar la información de un bien se debe digitar o leer el código de barras y luego pulsar el botón **Buscar**, el resultado de la búsqueda se despliega:

Constatación 9:22 ok

Código I01001010100583 Buscar

Nombre MANUAL DEL ARQUITECTO Y CONSTRUCTOR (2TOMOS)

Serie

Marca

Modelo

Observacion DOS TOMOS

Tipo  Activo  Control

Información Estado Responsable Ubicación

▶ Grabar ▶ Cancelar ▶ Salir

Figura C.15: Ventana de Datos Generales

En la siguiente figura se muestra los responsables actuales del bien

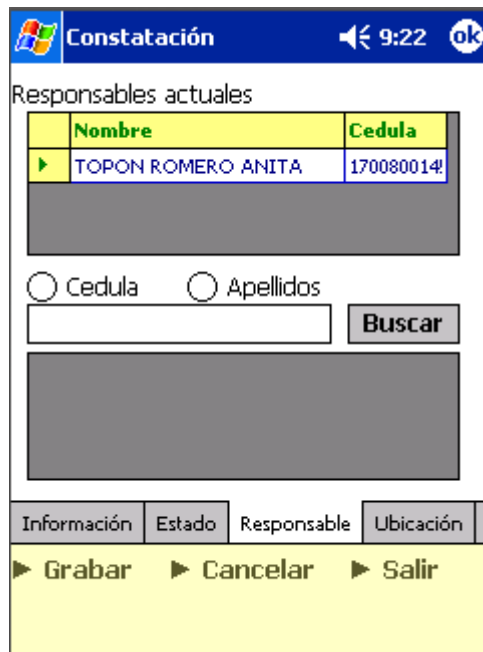


Figura C.16: Ventana de Responsables actuales del bien

Si se pulsa sobre un responsable actual permite eliminarlo, para confirmar la eliminación, aparece el siguiente mensaje

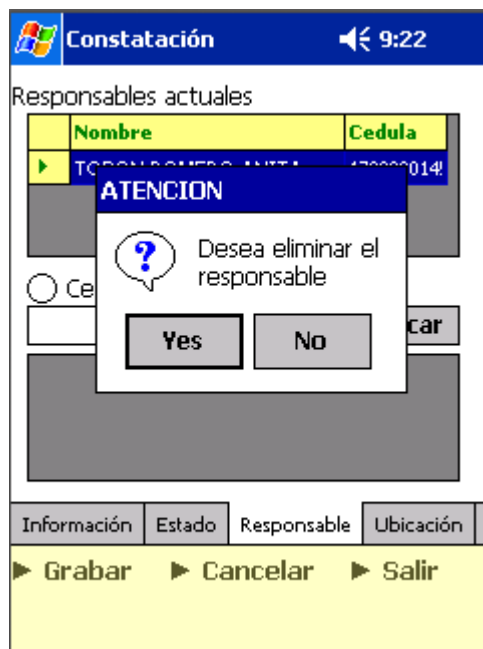


Figura C.17: Confirmación de eliminación de un responsable

Si se desea agregar un nuevo responsable al bien se tiene que buscar

un responsable ya sea ingresando la cédula o los apellidos y pulsando el botón **Buscar** como se muestra en la siguiente figura:

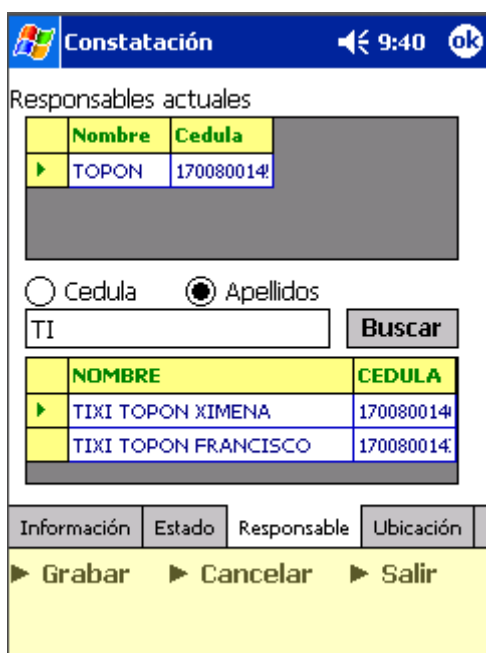


Figura C.18: Búsqueda de un nuevo responsable

Una vez que ya se ha elegido el nuevo responsable se pulsa sobre éste y se despliega un mensaje de confirmación:

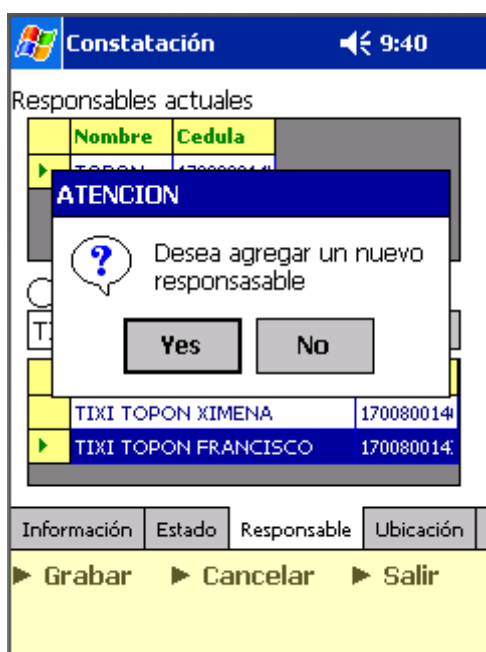


Figura C.19: Confirmación de asignación de un nuevo responsable

Nota: El número máximo de responsables por bien es de cinco

En la siguiente figura se despliega el estado actual del bien

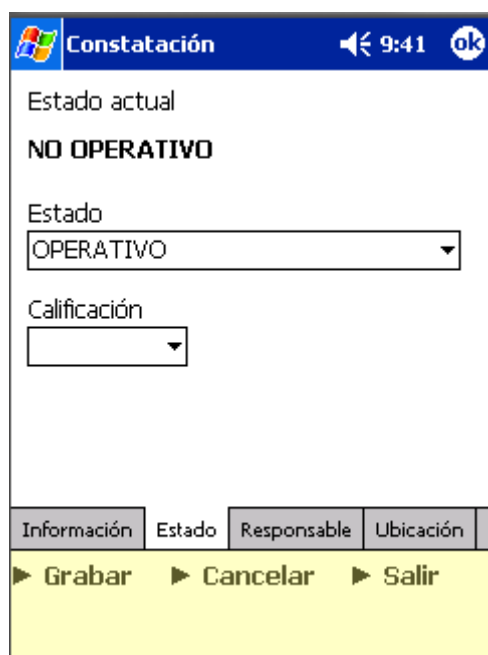


Figura C.20: Ventana del estado actual del bien

En el siguiente gráfico aparece la dependencia actual del bien

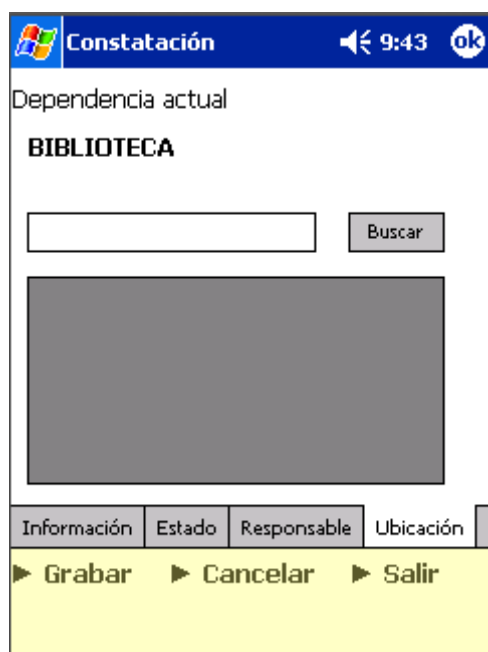


Figura C.21: Ventana de la dependencia actual del bien

Para cambiar la dependencia actual de un bien se debe buscar la nueva dependencia para esto se debe escribir el nombre de la dependencia a cambiar y luego pulsar el botón **Buscar**, así:

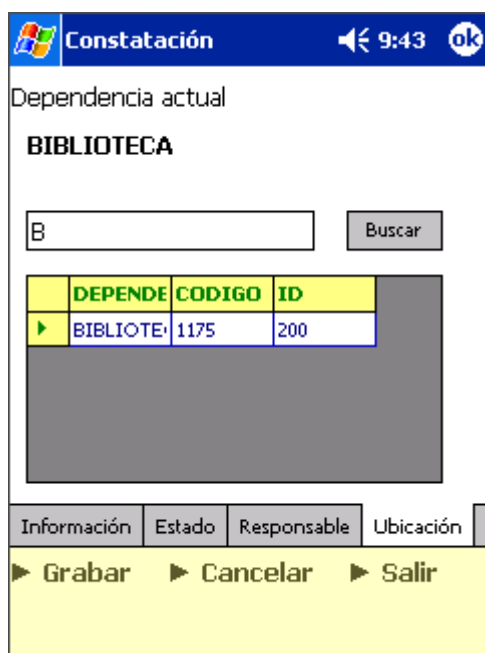


Figura C.22: Búsqueda de una nueva dependencia

Para cambiar la dependencia pulsar en una de las dependencias mostrada en la búsqueda

Para grabar los cambios realizados a la información del bien consultado se debe pulsar en la opción **▶ Grabar**, luego se despliega un mensaje confirmando la transacción, así:



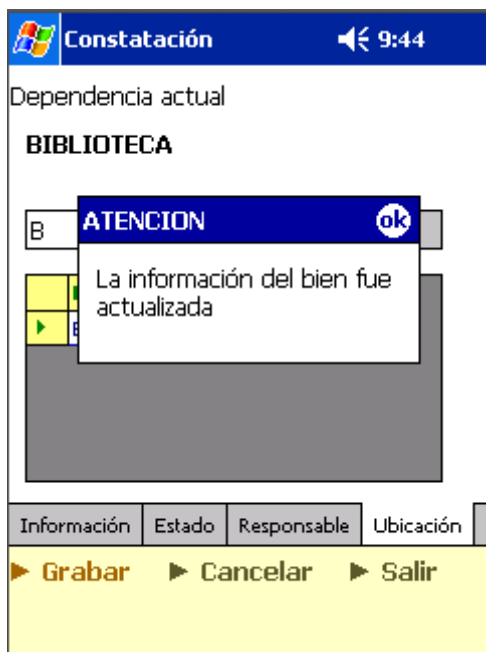


Figura C.23. Mensaje de actualización de la información del bien

Para desechar los cambios realizados a la información del bien consultado se debe pulsar en la opción **▶ Cancelar**

Para salir de la ventana de constatación se tiene que pulsar la opción **▶ Salir**

✓ **Sincronizar**

**▶ Sincronizar**

Este proceso realiza la actualización de todos los bienes modificados en el dispositivo móvil a la base de datos Oracle ubicada en el servidor

Finalizada la sincronización aparece el siguiente mensaje:

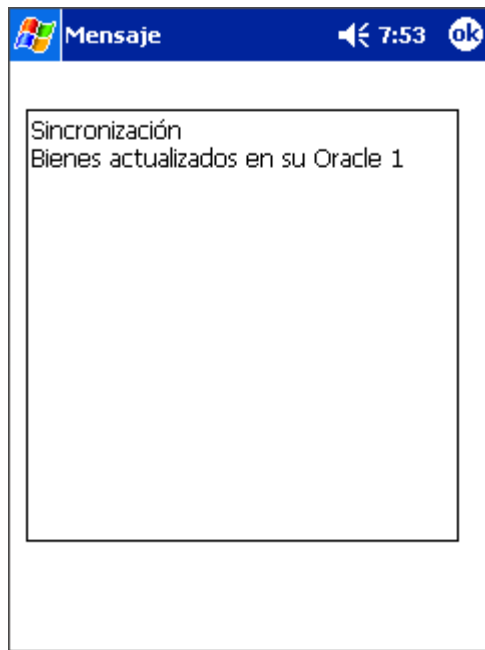


Figura C.24. Mensaje de Sincronización

Cuando no se ha realizado ninguna modificación de la información de los bienes en el dispositivo móvil aparece el siguiente mensaje

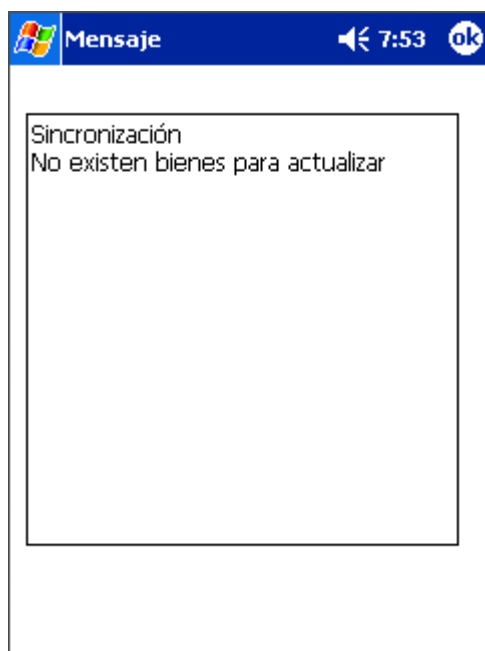


Figura C.25. Mensaje de no existencia de bienes a actualizar

## Parte Windows

### Ingreso del Sistema

A continuación se despliega la siguiente ventana

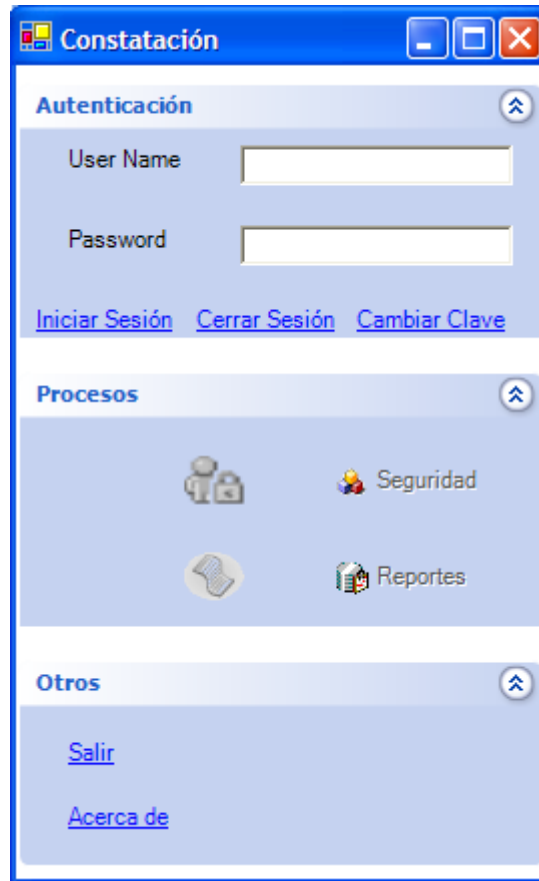


Figura C.26. Ventana de acceso al módulo de Windows

La aplicación consta de los siguientes módulos

#### **Seguridad**

Da la opción al administrador del sistema de asignar usuarios, crear perfiles, y asignar los perfiles a los usuarios.

#### **Reportes**

Da la opción de ver los reportes

Para que se habiliten las opciones de Cambio de Password, Seguridad y reportes se tiene que autenticarse el usuario

Digitar el usuario y la contraseña respectiva y luego presionar clic en la opción Iniciar Sesión [Iniciar Sesión](#)

### Mensajes de Error de Ingreso al Sistema

- El siguiente error se despliega cuando el usuario o la contraseña están incorrectos

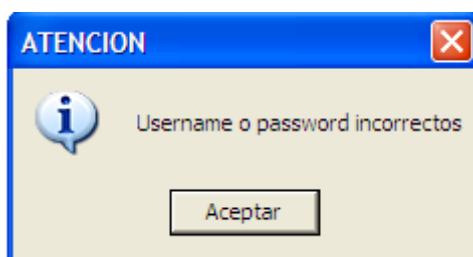


Figura C.27: Mensaje de Error PC: Username o Password incorrectos

### Cambiar Clave

Para cambiar la clave dar clic en la opción [Cambiar Clave](#) de la ventana principal, luego se desplegará el siguiente formulario:

Un formulario con un fondo azul claro y bordes redondeados. Tiene tres campos de entrada de texto: "User Name" con el valor "espe", "Nuevo Password" y "Confirmar Password". En la parte inferior del formulario hay tres botones de texto: "Aceptar", "Cancelar" y "Cerrar", todos en color azul.

Figura C.28: Ventana Cambiar Password

Se debe digitar el nuevo password y su respectiva confirmación.

Una vez que la clave ha sido actualizada, se despliega el siguiente mensaje:

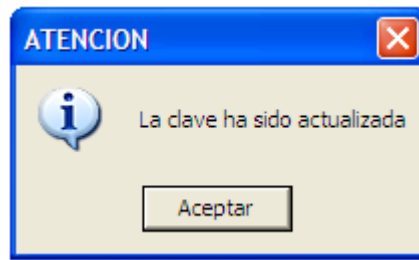


Figura C.29: Mensaje: La clave ha sido actualizada

### Mensajes de Error de Cambio de Password

- El siguiente error se despliega cuando el la nueva contraseña y su confirmación no es igual.

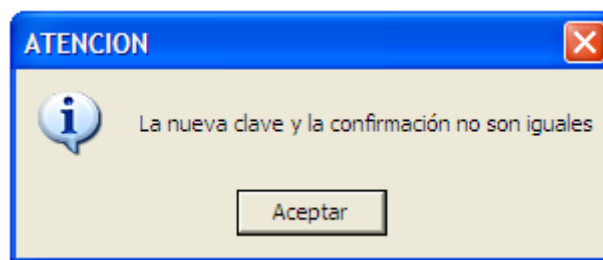
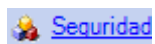


Figura C.30: Mensaje de Error PC: La nueva clave y su confirmación no son iguales

### Administración de Seguridad

Para ingresar al módulo de Administración de Seguridad, dar clic en la opción



de la ventana principal, luego se desplegará lo siguiente:

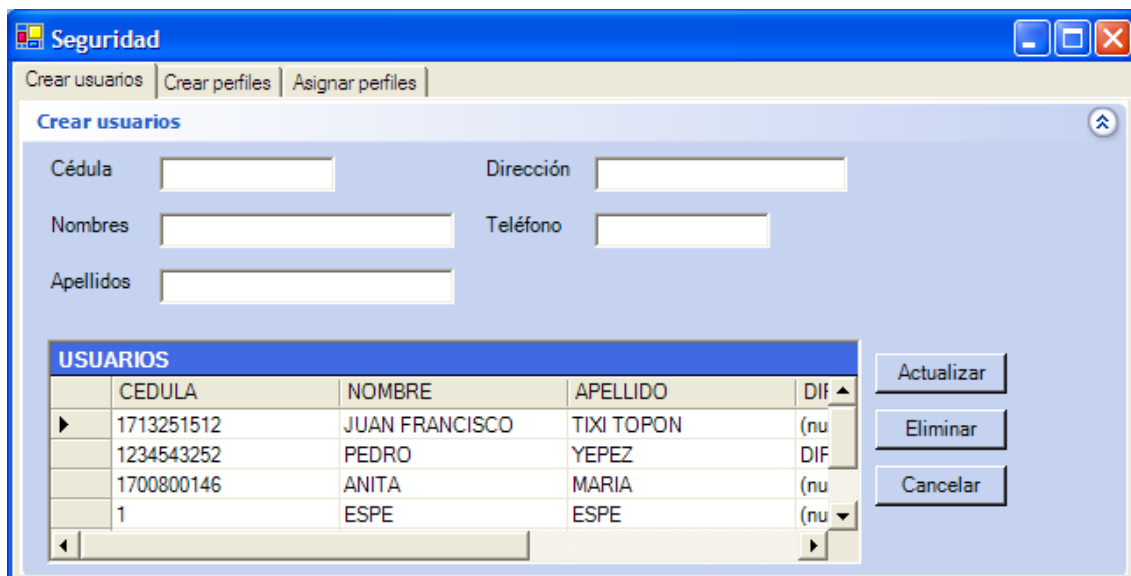


Figura C.31: Ventana de Administración de Seguridad

Esta ventana sirve para el mantenimiento de usuarios, creación de perfiles, asignación de módulos a los perfiles.

### Crear Usuarios

Se debe dar un clic en la pestaña “Crear Usuarios”, de la ventana seguridad

Se tiene que llenar la información solicitada por el formulario “Crear Usuario”, los datos requeridos son: cedula, nombres y apellidos.

Una vez que se ha ingresado la información requerida se debe dar un clic en el botón “Actualizar”, así:



Figura C.32: Ventana: Crear Usuarios

## Generación de UserName

Una vez que se ha actualizado la información, se muestra en un mensaje, el username del usuario ingresado:

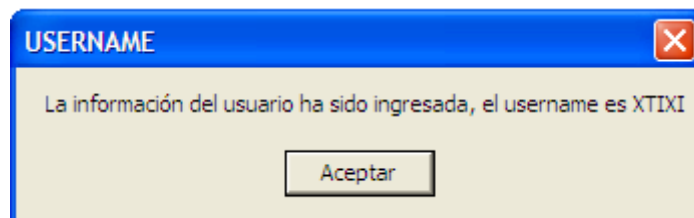


Figura C.33: Mensaje: Generación de UserName

## Actualización y Eliminación de Usuarios

Se debe elegir el registro a actualizar o eliminar, mediante un clic en la fila del grid USUARIOS, para actualizar se tiene que presionar en el botón “Actualizar”, y para borrar en el botón “Eliminar”, así:

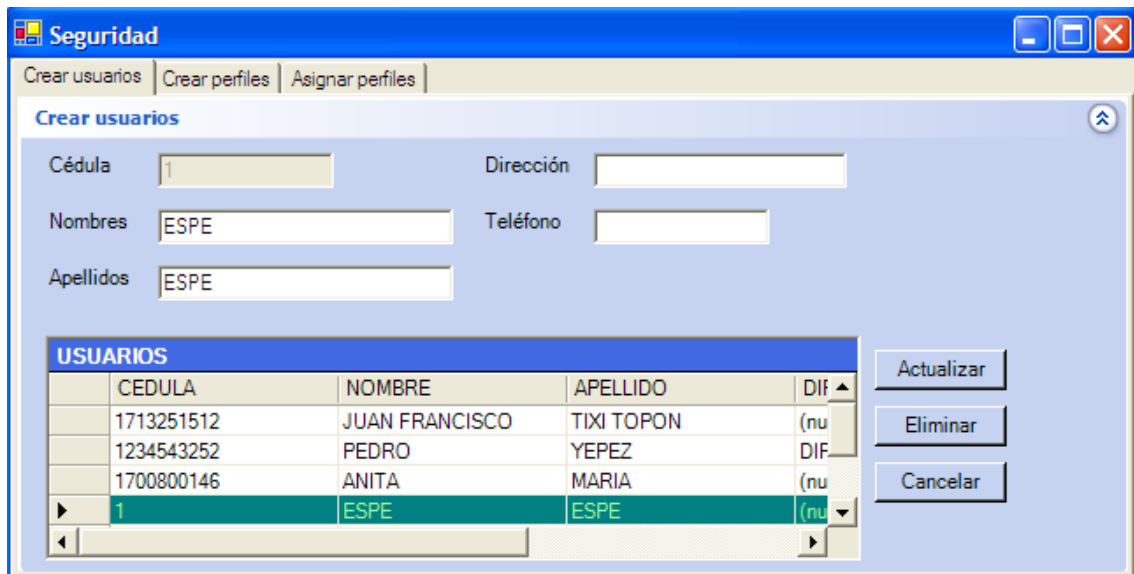


Figura C.34: Elección de Usuario a Modificar o Eliminar

### Confirmación de Actualizado y Borrado de Usuarios

El siguiente mensaje se muestra cuando se ha actualizado satisfactoriamente el registro deseado.

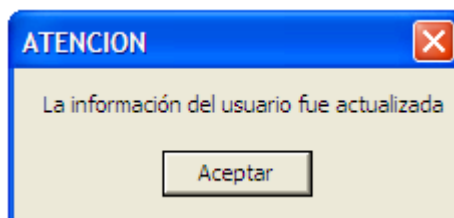


Figura C.35: Mensaje: La Información fue actualizada

El siguiente mensaje se muestra cuando se ha borrado satisfactoriamente el registro elegido.



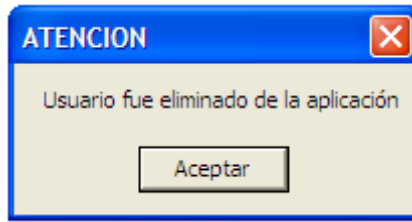


Figura C.36: Mensaje: La Información fue eliminada

## Crear Perfiles y asignación de módulos

Se debe dar un clic en la pestaña “Crear Perfiles”, de la ventana seguridad

Se tiene que llenar el nombre del nuevo perfil, y luego presionar el botón “Actualizar”.

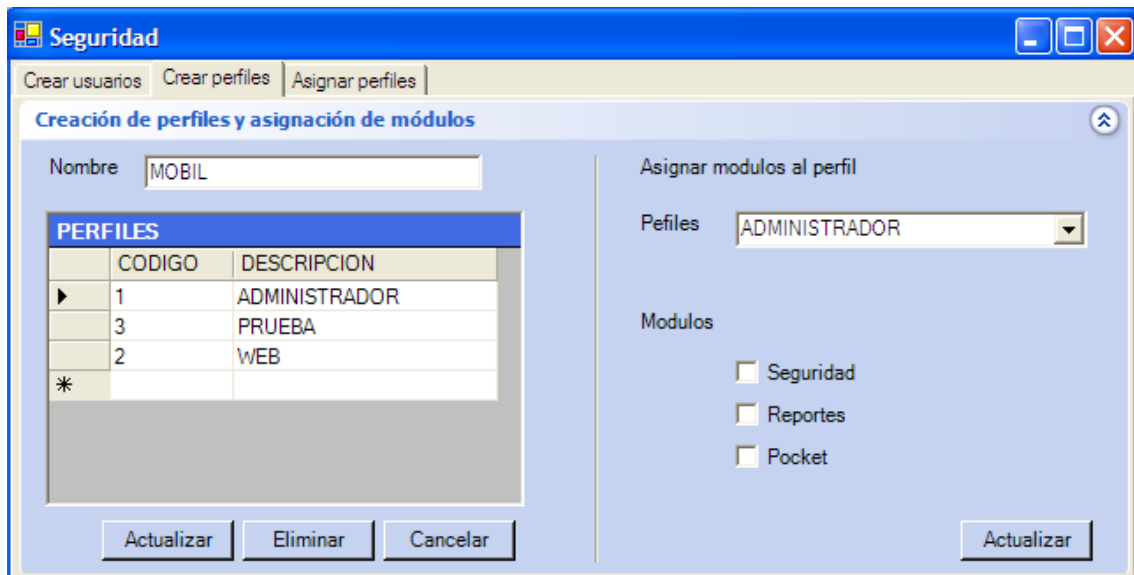


Figura C.37: Ventana: Crear Perfiles y Asignación de módulos

## Confirmación de creación de un nuevo perfil

Una vez que se ha actualizado la información, se muestra el siguiente mensaje:

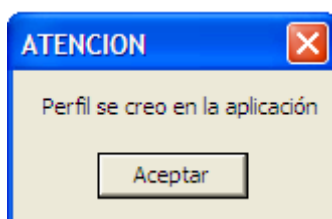


Figura C.38: Mensaje: Perfil Creado

## Actualización y Eliminación de Perfiles

Se debe elegir el registro a actualizar o eliminar, mediante un clic en la fila del grid PERFILES, para actualizar se tiene que presionar en el botón "Actualizar", y para borrar en el botón "Eliminar", así:

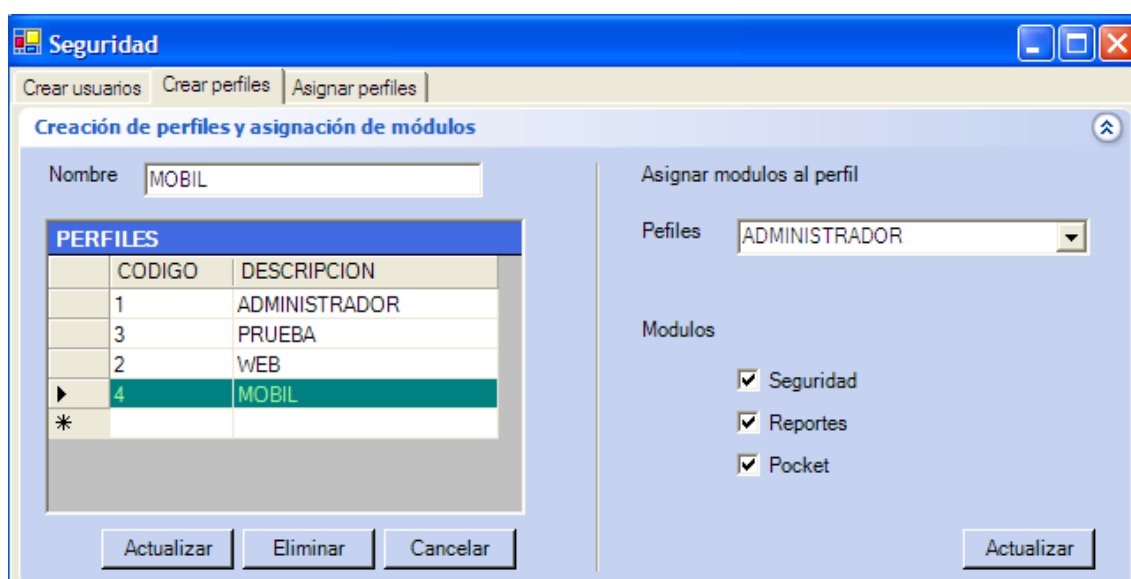


Figura C.39: Ventana: Elección de Perfil a Modificar o Eliminar

## Confirmación de Actualizado y Borrado de Perfiles

El siguiente mensaje se muestra cuando se ha actualizado satisfactoriamente el registro deseado.

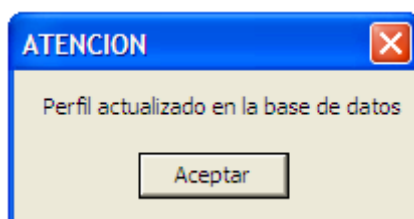


Figura C.40: Mensaje: La Información fue actualizada

El siguiente mensaje se muestra cuando se ha borrado satisfactoriamente el registro elegido.

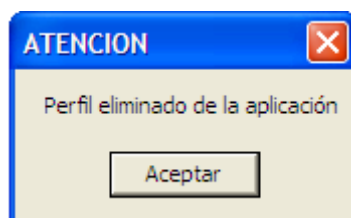


Figura C.41: Mensaje: La Información fue eliminada

## Asignación de módulos

En la parte derecha de la ventana Crear perfiles y asignación de módulos, se debe seleccionar el nombre del perfil, y los módulos que tendrá acceso dicho perfil. Luego se debe pulsar el botón "Actualizar", así:

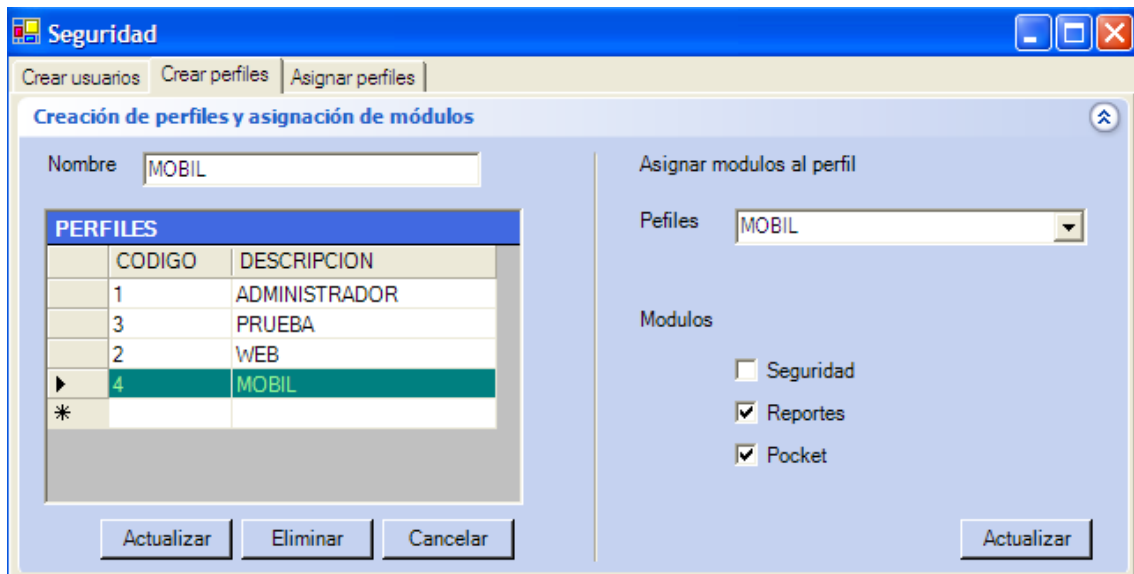


Figura C.42: Ventana: Asignación de módulos

### Confirmación de Asignación de módulos a perfil

El siguiente mensaje se muestra cuando se ha asignado satisfactoriamente los módulos al perfil deseado.

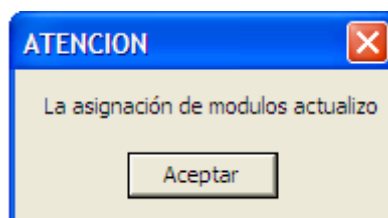


Figura C.43: Mensaje: La asignación fue actualizada

### Asignación de Perfiles

Se debe dar un clic en la pestaña “Asignar Perfiles”, de la ventana seguridad

Se debe seleccionar el nombre del Usuario, y el nombre del Perfil que se quiera asignar. Luego se debe pulsar el botón “Actualizar”, así:

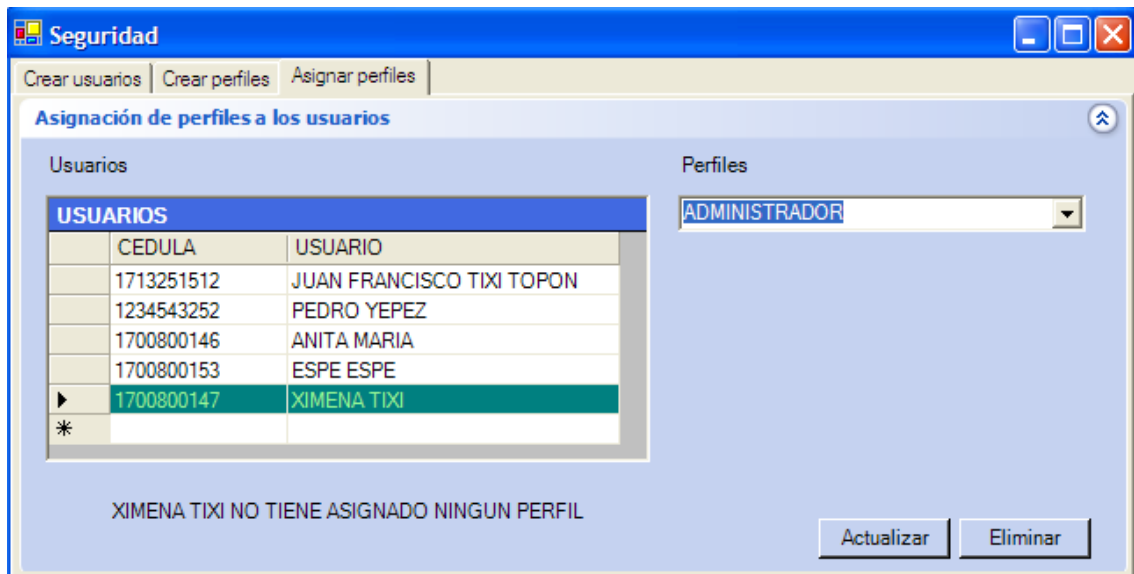


Figura C.44: Ventana: Asignación de perfiles

### Confirmación de Asignación de Perfiles

El siguiente mensaje se muestra cuando se ha asignado satisfactoriamente el perfil al usuario deseado.

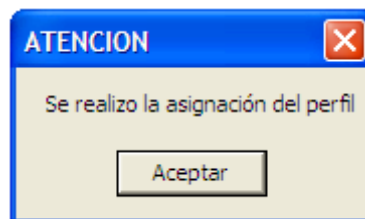



Figura C.45: Mensaje: La asignación fue realizada

## Reportes

Para ingresar al módulo de Reportes, dar clic en la opción  Reportes de la ventana principal, luego se desplegará lo siguiente:

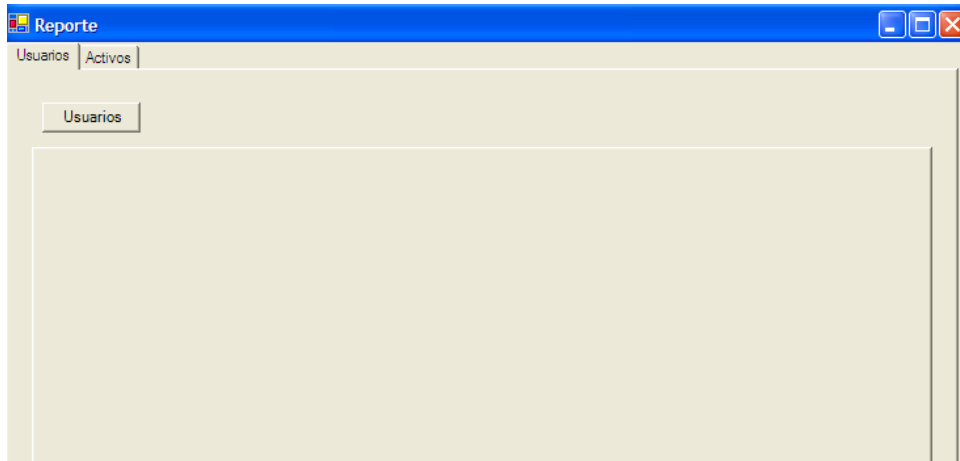


Figura C.46: Ventana de Reportes

Esta ventana sirve para la generación de Reportes de Usuarios

Para visualizar el reporte de usuarios se debe dar un clic en el botón "Usuarios". Así:

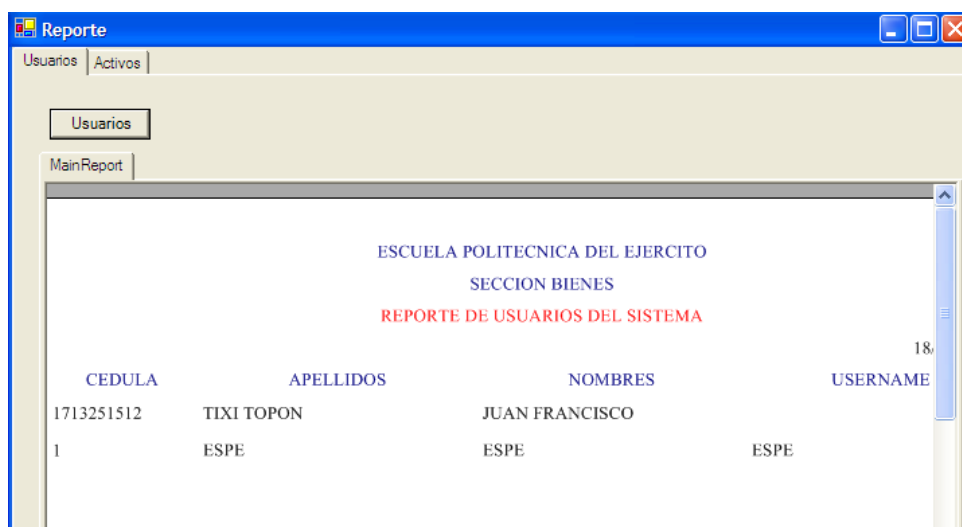


Figura C.47: Reportes de Usuarios del Sistema

## **SALIR**

El momento en que desea salir de la aplicación dar clic en [Salir](#)

## **BIBLIOGRAFIA**

[WWW.MICROSOFT.COM](http://WWW.MICROSOFT.COM)

[WWW.EXTREMEPROGRAMMING.ORG](http://WWW.EXTREMEPROGRAMMING.ORG)

[WWW.WILLYDEV.COM](http://WWW.WILLYDEV.COM)

[WWW.CSHARPCORNER.COM](http://WWW.CSHARPCORNER.COM)

[WWW.WINDOWSFORMS.COM](http://WWW.WINDOWSFORMS.COM)

[WWW.CODEPROJECT.COM](http://WWW.CODEPROJECT.COM)

[WWW.LOGICSTUDIO.NET](http://WWW.LOGICSTUDIO.NET)

[WWW.TASKVISION.COM](http://WWW.TASKVISION.COM)

[WWW.COMPACTFRAMER.COM](http://WWW.COMPACTFRAMER.COM)

[WWW.WELEARN247.COM](http://WWW.WELEARN247.COM)



## **GLOSARIO**

Custodio

Persona responsable de un bien o un activo fijo

Compac Framework

Plataforma de desarrollo para dispositivos móviles

XML

Estándar de marcado de texto

Soap

Protocolo aplicaciones orientadas a servicios

C#

Lenguaje de programación de VS 2003

Web Service

Servicios que publican métodos a ser consumidos

URL

dirección web de un sitio

Xp

Programación Extrema

Servicios

Aplicación que puede ser consumido de forma libre

Moviles

Dispositivos que soporta tecnología microsoft

**HOJA DE LEGALIZACION DE FIRMAS**

**ELABORADO POR**

---

Juan Francisco Tixi Topón

**COORDINADOR DE LA CARRERA**

---

Ing. Ramiro Delgado

Sangolquí, 22 de diciembre de 2008