Parte I Introducción

Capítulo 1. PRESENTACIÓN DEL PROYECTO

1.1 Descripción del Proyecto

El proyecto consiste básicamente en la elaboración de un comunicador, es decir, la construcción de un producto software que facilite la comunicación de una persona con parálisis cerebral con el resto, mediante un sistema basado en la escritura.
La aplicación que compone el producto es un programa para un dispositivo móvil (en concreto, una PDA Pocket PC).
Este proyecto está pensado que sea una herramienta que ayude a las personas con parálisis cerebral a incorporarse en la medida de lo posible al entorno, dejando ver de esta manera que la informática además de los avances tecnológicos, también puede ayudar a la integración de este colectivo en la sociedad de hoy.
Debido a las ventajas que aporta el dispositivo móvil, como son su portabilidad, reducido tamaño, peso... el usuario va a poder ampliar su independencia, aumentar su capacidad de comunicación etc.

1.2 Alcance

La aplicación está pensada para personas con parálisis cerebral, o personas privadas de la capacidad del habla, pero que sin embargo hayan adquirido a lo largo de su vida la facultad de escribir y de leer, puesto que la aplicación se basa en el lenguaje escrito. Además para aumentar su utilidad será reproducido cada una de las expresiones del usuario, mediante un sintetizador de voz.

1.3 Acerca de esta documentación

Esta memoria pretende ser rigurosa con los contenidos teóricos y no extenderse en descripciones exhaustivas. No pretende hacer un análisis detallado del hardware o del software empleado. Se ciñe a dar los conocimientos necesarios para la comprensión del desarrollo de la aplicación bajo un entorno de las características que aquí se prestan.
La memoria se organiza en partes, y ésta a su vez en capítulos que también son divididos en puntos, de esta forma se establece un orden lógico de lectura por temas.
A continuación se describe en qué consistirá cada una de las partes de la memoria:

- **Parte I. Introducción :** Esta es la sección en la que nos encontramos actualmente, en la que se hace una breve presentación del tema que aborda el proyecto, junto con los objetivos del mismo, a la vez que se hace una pequeña descripción del contenido y estructura de la memoria.
- **Parte II. Fundamentos Teóricos:** En este apartado desarrollamos todos los contenidos teóricos necesarios para la comprensión del proyecto. Son los siguientes:
    - Parálisis Cerebral : Qué es la parálisis cerebral y el contexto donde se desarrolla la aplicación.
    - Las nuevas tecnologías de la comunicación e información y los alumnos con déficit: una rápida descripción de la situación de las nuevas tecnologías en este ámbito.
    - Introducción a los dispositivos móviles: Una breve reflexión sobre las tecnologías móviles, haciendo mención especial a la PDA empleada en este proyecto, a los procesadores empleados en este tipo de dispositivos y al sistema operativo que ejecutan.
    - Desarrollo de aplicaciones para Pocket PC: En este capítulo se recorre de manera teórica los conceptos necesarios para llevar a cabo el desarrollo de una aplicación para este tipo de dispositivos, así como una descripción de la plataforma .NET y las herramientas y programas que nos permiten el desarrollo de aplicaciones como ésta.
- **Parte III. Desarrollo de la aplicación:** Podríamos considerar que esta parte tiene dos secciones.:

  Una primera que consiste en la realización de un estudio sobre los posibles métodos de comunicación para dispositivos móviles que pueden desarrollarse basándose en las habilidades y capacidades de las que disponen las personas con parálisis cerebral, así como el aprovechamiento de las ventajas que nos ofrecen este tipo de dispositivos. Los métodos de comunicación son basados en la escritura, porque consideramos que hay un buen número de herramientas visuales, y que éstas tienen un grado de comunicación bastante limitado.

  Y una segunda parte en la que nos centraremos únicamente en las fases de análisis, diseño, implementación e instalación de uno de los métodos concretos, que con la colaboración directa del Centro Obregón nos ha permitido encuadrar nuestro proyecto en un entorno real. Un punto importante que hay que tener en cuenta, y resaltar es el hecho de que la aplicación será usada por personas que hayan adquirido la capacidad de la escritura a lo largo de su vida, sin embargo, se ha añadido al proyecto un sintetizador de voz que permite que la comunicación no se limite solo con personas que puedan leer, sino que lo podrán hacer con todas aquellas de su entorno
- **Parte IV**. **Manual de usuario**: Es una guía pormenorizada del manejo de la aplicación.
- **Parte V. Conclusiones:** Reflexión sobre los objetivos alcanzados y sobre el futuro y nuevas vías de actuación del proyecto.
- **Apéndices.**

Capítulo 2. OBJETIVOS PROPUESTOS

2.1 Objetivos propuestos

A continuación se listan los objetivos que se pretenden cubrir con este proyecto.

- Elaboración de un estudio de métodos de comunicación sobre PDA para personas con parálisis cerebral, que permita según sus características personales la aplicación de un método u otro.
- Realización de una aplicación que permita a las personas con parálisis cerebral una comunicación más natural y sencilla.
- Derivado de este objetivo, la consecución de una mayor independencia al usuario de la aplicación y la integración de estas personas que tienen problemas con la comunicación en la sociedad de hoy.
- Conseguir desarrollar una aplicación real, destinada a usuarios finales.
- Fomentar el aprendizaje del trabajo en equipo, tanto entre nosotros como con algunos usuarios finales del proyecto
- Ayudar a mejorar la calidad de vida a personas que se han visto privadas de ello, así como a las personas de su entorno, ya que tienen mucho que decir, y desgraciadamente los medios a su alcance son limitados.
- Valorar y conocer a un colectivo con pocas oportunidades en relación con las Nuevas Tecnologías y con grandes dificultades de comunicación
- Implicar al Centro Obregón, en un primer momento y a todos los colectivos de características similares, en el desarrollo y uso de nuevas tecnologías, para ayudar a acercar el mundo del futuro al día de hoy.
- Aplicación de una metodología de desarrollo software que permita la creación de esta aplicación.
- El uso correcto de las técnicas adquiridas a lo largo de estos años.
- Conocer un entorno de trabajo específico como puede ser el desarrollo de aplicaciones para dispositivos móviles.

Parte II Fundamentos Teóricos

Capítulo 3. CONTEXTO DE LA APLICACIÓN: LA PARÁLISIS CEREBRAL

3.1 ¿Qué es la Parálisis Cerebral?

3.1.1 Definición

Se define como un trastorno neuromotor no progresivo debido a una lesión o una anomalía del desarrollo del cerebro inmaduro.

La Parálisis Cerebral no permite o dificulta los mensajes enviados por el cerebro hacia los músculos, dificultando el movimiento de éstos. Es un concepto enormemente ambiguo ya que aunque sea un trastorno motor también lleva asociados otros de tipo sensorial, perceptivo y psicológico.

La Parálisis Cerebral no es progresiva, lo que significa que no se agravará cuando el niño sea más mayor, pero algunos problemas se pueden hacer más evidentes.

3.1.2 Problemas del lenguaje

La Parálisis Cerebral puede presentar una gran diversidad en cuanto a características de los problemas del lenguaje. En cuanto a la frecuencia, las estadísticas varían mucho según los autores, pero puede afirmarse que alrededor de un 60% de los casos de parálisis cerebral presentarán problemas de lenguaje.

En principio lo que caracteriza la parálisis cerebral es la dificultad motora en la ejecución del lenguaje expresivo, existiendo desde problemas muy leves hasta la imposibilidad total de emitir un sonido comprensible. No obstante, no hay que olvidar que un número importante de casos presenta desde retrasos simples o retrasos graves en la adquisición del lenguaje.

Se debe tener en cuenta, que junto con otros aspectos madurativos, el ritmo de adquisición del lenguaje en el paralítico cerebral puede presentar características especiales que oscilan desde pequeños problemas a nivel morfológico, hasta problemas graves de comprensión que requerirán una intervención especializada.

En general, se distinguen dos grandes aspectos de los posibles problemas:

1. Problemas en la adquisición del lenguaje.
2. Problemas motores de expresión, que afectan al habla y a la voz.

La adquisición del lenguaje

Los estudios sobre la evolución del lenguaje en el paralítico cerebral son más bien limitados ya que en general se ha estudiado mucho más el problema motor. No obstante en la práctica nos podemos encontrar diferentes niveles evolutivos según los casos:

a. Nivel de desarrollo normal o superior
b. Nivel de desarrollo con ligero retraso
c. Nivel de desarrollo con retraso grave

En los casos en que exista retraso del lenguaje deberemos tener en cuenta dos posibilidades:

- El paralítico cerebral que tiene deficiencia mental, ligera, media o grave como trastorno asociado.
- El paralítico cerebral que tiene un desarrollo cognitivo global normal, pero con algún grado de retraso del lenguaje.

En el primer caso, los problemas serán iguales que en la deficiencia mental correspondiente, pero con el problema motriz sobreañadido que posiblemente dificultará las interacciones verbales.

En el segundo caso, no puede decirse que haya un perfil característico de los retrasos del lenguaje en el paralítico cerebral. Esto es, no se puede decir que lo característico sean los problemas fonológicos, los semánticos, etc., sino que el grado de retraso del lenguaje y los aspectos implicados variarán en cada caso.

Problemas motores de expresión

Estas dificultades pueden afectar a las funciones de respiración, fonación, voz, articulación, con diferentes niveles de dificultad en cada uno de ellos.

Los problemas más frecuentes son:

- Mímica facial inexpresiva, pobre, lenta o, por el contrario, con gestos bruscos, exagerados, tics, etc, que muchas ocasiones aparecen con motivo de una actividad motriz voluntaria y en particular con el lenguaje.
- Movimientos asociados de una parte del cuerpo (brazos, manos, hombros, cabeza, etc.) o de todo él al hablar o intentar hablar.
- Los reflejos orales pueden no aparecer o por el contrario persistir hasta edades avanzadas.
- Con frecuencia está alterada la motricidad de la alimentación. Tarda en deglutir correctamente y en masticar.
- El balbuceo muchas veces es pobre y presenta dificultades en sus aspectos interactivos.
- Problemas de voz: se manifiestan ya desde edades tempranas dando lugar frecuentemente a síntomas disfónicos.
- Problemas de fonación y prosodia: la emisión fluida de sonidos está alterada y muchas veces la entonación, la melodía y el ritmo no son adecuados.

3.1.3 Tipos de parálisis cerebral

Existen muchos tipos y varias clasificaciones, una de ellas es en cuanto al tipo de parálisis cerebral. Es interesante nombrar las características de cada una de ellas, en cuanto al lenguaje se refiere:

- **Espasticidad** (Espásticos): aumento exagerado del tono muscular (hipertonía), por lo que hay movimientos exagerados y poco coordinados. Afecta al 70-80% de los pacientes. El lenguaje tiende a ser explosivo, interrumpido por largas pausas. En casos muy graves, puede quedar bloqueado al no poder mover sus mecanismos de fonación.

- **Atetosis** (Atetósicos): se pasa de hipertonía a hipotonía, por lo que hay movimientos incoordinados, lentos, no controlables. Estos movimientos afectan a las manos, los pies, los brazos o las piernas y en algunos casos los músculos de la cara y la lengua, lo que provoca hacer muecas o babear. Los movimientos aumentan a menudo con el estrés emocional y desaparecen mientras se duerme. Pueden tener problemas para coordinar los movimientos musculares necesarios para el habla (disartria). El lenguaje es muy variable. Los casos leves pueden presentar pequeños fallos en la articulación, mientras que los graves no hablan en absoluto. El atetósico produce, en general, un habla incordinada y carente de ritmo

- **Ataxia**: sentido defectuoso de la marcha y descoordinación motora tanto fina como gruesa. Es una forma rara en la que las personas afectadas caminan inestablemente, poniendo los pies muy separados uno del otro.

- **Mixto**: es lo más frecuente, manifiestan diferentes características de los anteriores tipos. La combinación más frecuente es la de espasticidad y movimientos atetoides.

3.1.4 Epidemiología

Es la causa más frecuente de discapacidad física en los niños después de haberse instaurado la vacuna de la poliomielitis.
Se presenta en dos de cada 1.000 nacidos vivos. En España, alrededor de 1.500 bebés nacen o desarrollan una Parálisis Cerebral cada año. Puede afectar a niños y a niñas de cualquier raza y condición social.

3.1.5 Síntomas

Los primeros síntomas comienzan antes de los tres años de edad y suele manifestarse porque al niño le cuesta más trabajo voltearse, sentarse, gatear, sonreír o caminar. Los síntomas varían de una persona a otra, pueden ser tan leves que apenas se perciban o tan importantes que le imposibilite levantarse de la cama. Algunas personas pueden tener trastornos médicos asociados como convulsiones o retraso mental, pero no siempre ocasiona graves impedimentos.

Los síntomas más importantes son las alteraciones del tono muscular y el movimiento, pero se pueden asociar otras manifestaciones:

- Problemas visuales y auditivos.
- Dificultades en el habla y el lenguaje.
- Alteraciones perceptivas:
  - Agnosias: Alteración del reconocimiento de los estímulos sensoriales.

  - Apraxias: Pérdida de la facultad de realizar movimientos coordinados para un fin determinado o pérdida de la comprensión del uso de los objetos ordinarios, lo que da lugar a comportamientos absurdos. Incapacidad para realizar movimientos útiles.
- Distractibilidad.
- Diskinesia: dificultad en los movimientos voluntarios.

Las contracturas musculares que se asocian con la Parálisis Cerebral conlleva que sea imposible que la articulación se mueva, pero también puede ocurrir que exista una falta de tono muscular, por lo que las articulaciones pueden dislocarse ya que los músculos no las estabilizan.

Nivel cognitivo
La Parálisis Cerebral Infantil (PCI) no tiene porqué suponer una afectación a nivel cognitivo, como tradicionalmente se ha creído, lo que ha abierto el camino a intervenciones psicológicas que lo han potenciado. Las personas que no son capaces de controlar bien sus movimientos, o no pueden hablar, a menudo se da por supuesto que tienen una discapacidad mental. Aunque algunas personas con Parálisis Cerebral tienen problemas de aprendizaje, esto no es siempre así, incluso pueden tener un coeficiente de inteligencia más alto de lo normal. Aproximadamente un tercio de los niños tienen un retraso mental leve, un tercio tiene incapacidad moderada o grave y el otro tercio restante es intelectualmente normal

Alteraciones visuales
El problema visual más frecuente es el estrabismo (los ojos no están alineados) que puede necesitar ser corregido con gafas o, en los casos más graves, mediante una operación quirúrgica.
Los problemas visuales más serios son menos frecuentes. Algunos niños pueden tener un defecto que provoca que la parte del cerebro que es responsable de la interpretación de las imágenes que el niño ve no funciona con normalidad. En pocos casos, el niño se puede quedar ciego pero en la mayoría de los casos los niños con este defecto sólo tienen dificultad para descifrar los mensajes que reciben desde sus ojos, por ejemplo, cuando aprenden a leer.

Comunicación
La capacidad de comunicarse de un niño afectado por Parálisis Cerebral va a depender fundamentalmente de su desarrollo intelectual, que hay que estimular desde el principio. Su capacidad de hablar también dependerá de la habilidad que adquiera para controlar los músculos de la boca, la lengua, el paladar y la cavidad bucal. Las dificultades para hablar que tienen los paralíticos cerebrales suelen ir asociadas a las de tragar y masticar.

Epilepsia
Afecta a uno de cada tres niños, es impredecible cuando puede ocurrir, pero puede ser controlada mediante medicación. Normalmente causan que los niños griten y seguidamente hay pérdida de la conciencia, sacudidas de las piernas y brazos, movimientos corpóreos convulsivos y pérdida del control de la vejiga. En el caso de convulsiones parciales simples hay sacudidas musculares, entumecimiento u hormigueo y en el caso de que sean complejas, la persona puede alucinar, tambalear o realizar movimientos automáticos y sin propósito, o manifestar una conciencia limitada.

Capítulo 4. LAS NUEVAS TECNOLOGÍAS DE LA COMUNICACIÓN Y LOS ALUMNOS CON DÉFICIT

El ser humano por naturaleza es un ser social e intenta participar de los procesos de interacción social convirtiéndose en un comunicador y receptor de mensajes (informaciones, sensaciones, conocimientos, datos, hechos, etc.). Es evidente que esta faceta humana puede desarrollarse hoy día gracias al avance de las NTIC (Nuevas Tecnologías de la Información y Comunicación) y de la aparición de nuevos dispositivos (periféricos) de comunicación que permiten y posibilitan la interacción entre personas con deficiencias auditivas, visuales, motóricas, ... siendo posible así mismo gracias al empleo de las NTIC el obtener beneficios en relación a :

- **Poseer un cierto Control ambiental,** esto es, poder manipular diferentes dispositivos domésticos (al estilo de lo que hoy día conocemos como el prototipo de las casas inteligentes) y llevar a cabo acciones como sencillas para quienes no presentamos déficit, como conectar o desconectar timbres, manejar sistemas de intercomunicación, el abrir o cerrar puertas, el comunicarse por los sistemas tradicionales como el teléfono, el poseer el control de aparatos de radio, de televisión, cassettes, las luces de la casa, el aparato de aire acondicionado, hacer todo lo más posible el acceso de las personas con dificultades de movilidad el acceso a los bienes de la cultura, etc.

- **Alcanzar la Integración laboral.** El empleo y difusión masiva de productos de hardware y software y su utilización en los ámbitos productivos y de la comunicación e información en nuestra sociedad ha modificado sustancialmente los procesos culturales, productivos y profesionales . Este cambio ha mejorado significativamente las expectativas personales y laborales de la persona con necesidades educativas especiales que ahora puede ocupar puestos de trabajo que antes del empleo masivo de las nuevas tecnologías le estaban vedados.

- **Democratización del uso del ordenador** al utilizarlo como cualquier usuario que no posea alguna discapacidad manifiesta**.** Por ejemplo, una persona tetrapléjica puede a través del habla, comunicarse con un ordenador y con el resto de las personas de su trabajo; igualmente el movimiento de su cabeza puede suponer con un casco adaptativo el modo de mover un ratón, lo que por ejemplo le permitiría realizar ciertos teletrabajos o participar en el mundo de la empresa como tele trabajador. En este sentido, los dispositivos que utilizaban hasta ahora personas con discapacidad los emplea cada vez más el resto de la población, especialmente las personas mayores. El ordenador, el fax y el correo electrónico permiten que en muchos casos

no sea necesaria la presencia física en el lugar de estudio o de trabajo. En concreto el teletrabajo, está abriendo nuevas expectativas al modificar algunos hábitos tradicionales. Igualmente para el caso de los deficientes motóricos están disponibles "teclados alternativos" los cuales les el acceso a programas y a las nuevas formas de comunicación telemática. Las nuevas tecnologías - con el uso de pantallas táctiles, conmutadores e interruptores, emuladores de teclado, ayudas para acceder al teclado estándar, lectores ópticos de tarjetas, digitalizadores de voz, tableros de conceptos, etc. - ayudan a que las limitaciones físicas no influyan en los aprendizajes ni en la capacidad de aprender.

- **Que las personas con deficiencias visuales interactúen con los demás y tengan acceso a los bienes del conocimiento,** personas para quienes cuya mayor dificultad con respecto al uso del ordenador, estriba en la salida de datos; gracias a la conexión al ordenador de sistemas alternativos al monitor o pantalla, donde las imágenes puedan ser sustituidas por sonidos, o por una línea Braille (táctil, para ir comprobando lo que se escribe) y las impresoras de Braille, pudiendo de esta manera, aprovechar el potencial de las nuevas tecnologías.

- **Desarrollar actividades comunes las persona con Deficiencias auditivas**, pongamos el caso de un estudiante sordo puede utilizar el ordenador tan fluidamente o con la misma dificultad que cualquier otra persona, ya que no necesita ninguna adaptación especial pues su deficiencia sensorial no afecta a su acceso al mismo, el elemento más utilizado para recibir la información es la pantalla, estos alumnos y alumnas podrán trabajar con él sin problemas y en algunos casos se puede sustituir cualquier mensaje sonoro por señales de tipo visual.

El empleo de las nuevas tecnologías puede ayudar a romper las barreras que dificultan la integración de las personas que presentan algún déficit o discapacidad, con todo tipo de garantías, en la sociedad.

4.1 La informática en procesos de rehabilitación y la educación terapeútica.

En la actualidad programas como **el Visualizador Fonético** desarrollado por IBM contribuyen a la **reeducaciòn del habla,**. estando dirigido a profesionales especializados en el tratamiento del habla, el lenguaje y el oído. Entre las funciones de este programa recogemos las siguientes:

- Ayuda a percibir las cualidades de la palabra articulada: ritmo, entonación, duración, pausa, intensidad y tiempo.
- Corrige y mejora la prosodia de la voz.
- Al mejorar la prosodia, hace más inteligible el habla.
- Visualiza la prosodia, favoreciendo así la corrección.
- Educa la respiración, proceso de capital importancia para la emisión del sonido articulado.

4.2 La atención a alumnos con deficiencias motóricas.

Las nuevas tecnologías pueden ser en sí mismas un recurso y un apoyo de los programas de intervención pedagógica dirigidos a alumnos con deficiencias motóricas. Podemos clasificar a estos alumnos en dos grupos: el primero sería aquel en el que existe una lesión cerebral (*parálisis cerebral, por ejemplo*) - *considerada la Parálisis Cerebral como un desorden permanente y no inmutable de la postura y del movimiento, debido a una disfunción del cerebro antes de completarse su crecimiento y su desarrollo"* - y un segundo grupo donde no hay lesión cerebral (*espina bífida, por ejemplo*).

Los trastornos más frecuentes de la parálisis cerebral son:

- Los relacionados con el lenguaje.
- Auditivos.
- Visuales.
- De la percepción.
- Del desarrollo mental.
- De la personalidad.
- De la atención.

Existen diversas máquinas y productos elaborados por la industria que podemos considerar como nuevas tecnologías y que pueden ayudar en el proceso rehabilitador de los alumnos con deficiencias motóricas, entre ellos citaremos:

* Microprocesador AUTOCOM con un funcionamiento parecido al de una máquina de escribir, el cual nos permite cambiar de renglón, volver atrás, corregir, etc., con la ayuda de una pieza magnética.

* POSUUM: permiten accionar diferentes aparatos como interruptor de luz, T.V, sonidos, etc. ... por medio de la mano, el pie o la boca.

* Máquinas de escribir eléctricas (permitir el borrado de errores).

* Tableros de letras, palabras, imágenes o símbolos adaptados al nivel del desarrollo de cada alumno y que pueden ser accionados apretando un botón, succionando, manejando una palanca o con un movimiento de cabeza.

* Ordenadores ayudan a los niños y niñas con Parálisis Cerebral en la comunicación, en el del aprendizaje y en el juego.

Los niños con parálisis cerebral, para el empleo de las máquinas, se pueden ayudar de conmutadores e interruptores, de pantallas táctiles, de los emuladores del teclado, digitalizadores de voz, etc, igualmente, se pueden emplear programas que contengan mensajes grabados que les ayuden a trabajar ciertos conceptos.

El empleo de las nuevas tecnologías con alumnos con déficit que les permita el poder comunicarse, explorar el entorno, tomar decisiones, simular situaciones,... va a aportar una mayor participación en las actividades escolares, en la dinámica del aula, y por tanto, le ayudará en su proceso enseñanza-aprendizaje, en la adquisición y desarrollo de habilidades, de conceptos y de procedimientos, así como a acercarse al conocimiento de hechos.

Capítulo 5. INTRODUCCIÓN A LOS DISPOSITIVOS MÓVILES

Para empezar, la nomenclatura utilizada es bastante retorcida, en muchos casos por motivos políticos y en otros por la gran evolución de estas tecnologías.

Como ejemplo, al principio los dispositivos de mano sin teclado eran llamados de forma genérica *Palmhelds*, mientras los dispositivos de mano con teclado se denominaban *Handhelds*. Microsoft llamó a sus dispositivos *PALM PCs* o *PPC*. Sin embargo, para evitar confusiones con la empresa *PALM*, que tenía registrado el nombre de su sistema operativo como *PALMOS*, Microsoft cambió la denominación de *Windows CE 3.0* por *PocketPc*, nombre que mantuvo con su *PocketPc 2002*.

En la nueva versión de *Windows CE 4.0* o *Windows CE .NET*, ha vuelto a cambiar el nombre para que el público relacione más su producto con su familia de sistemas operativos. Por este motivo lo ha denominado *Windows Mobile 2003*, del que ha sacado otra versión, *Windows Mobile 2003 Second Edition*.

Otra característica es la gran diversidad de máquinas que se han inventado, y que han provocado grandes incompatibilidades entre unas familias de PDAs y otras, incluso entre las mismas familias. Esto afortunadamente ha mejorado bastante en los últimos tiempos.

Para empezar, los términos para referirse a estos dispositivos

Término	Descripción
P.D.A(Personal Digital Assistant) Asistente personal Agenda Electrónica	Nombre genérico para cualquier dispositivo portátil de reducido tamaño para llevar encima.
Ordenador de mano	Lo mismo que PDA con la diferencia que aquí se suponen capacidades avanzadas de programación. Quedan excluidas las agendas sin capacidad de programación, como las de gama baja.
Handheld HPC	PDA con teclado tipo QWERTY incorporado, de más peso y con más prestaciones que un PPC
Palmheld Palm-size PC PPC Pocketpc	PDA. sin teclado. PocketPc es el nombre de uno de los sistemas operativos, pero actualmente se usa también como sinónimo de este tipo de dispositivos.
Pocketpc	PDA con S. O. Windows CE 3.0. Actualmente se usa este nombre asimismo como nombre genérico para este tipo de dispositivos.

Figura 5.1 Términos de los dispositivos móviles

5.1 Tipos de dispositivos móviles

Hay muchos tipos diferentes, pero básicamente están divididos en 4 grandes familias.

5.1.1 Newton

El primer PDA fue desarrollado por Macintosh , y se llamaba Newton. Tenía pantalla táctil, sin teclado monocromática.. Reconocía caracteres, y era mas o menos portátil. Pesaba medio Kilo, no está mal para la época, aunque sí que es un poco grande para meterlo en el bolsillo de la camisa.

.
Figura 5.2 Macintosh Newton. El primer PDA de la historia.

5.1.2 PALM

Tiene muchísima más importancia la familia de *PALM*. Sacaron unas PDAS que se convirtieron durante años en el referente de las agendas electrónicas. De echo el nombre de *PALM* se usa para describir de forma genérica este tipo de dispositivos.

Sus características eran:
Poco tamaño
Pantalla monocromática
Mucha autonomía, semanas de uso continuado sin recargarlas.
sin teclado. Para meter letras se usa la parte inferior de la pantalla, y se escribía directamente las letras a mano alzada usando el método de escritura Graffiti

Figura 5.3 PALM M505

Mismo procesador (*Motorola Dragonball*) con una velocidad limitada, y con poca memoria. Esto ha cambiado en los últimos dispositivos.
Sistema operativo *PALM OS*, que hace un uso muy eficiente de las limitadas prestaciones, y que va por la versión 6. Ha habido una gran evolución entre la versión 5 y la 6 para poder hacer un uso más eficiente de los nuevos procesadores.

Versiones compatibles entre sí. Este punto es realmente envidiable en la familia *Windows CE*. Es decir, es probable que un programa escrito hoy en día funcione en una *PALM* de hace 7 años. Esto también está cambiando con las últimas versiones de las máquinas. A pesar de eso, la compatibilidad hacia atrás es bastante buena.

En los últimos tiempos han cambiado un poco las cosas, están sacando procesadores mucho más potentes, han abandonado los procesadores *DragonBall* y se han decantado por los procesadores *StrongARM* y *Xcale*, y están sacando pantallas a color, con lo que ha perdido las ventajas de la autonomía. También hay móviles con *PalmOS*.

Figura 5.4 Palm III

5.1.3 EPOC-SYMBIAN.

Este sistema operativo, el *EPOC*, fue inventado y desarrollado por la empresa inglesa *Psion*, y son típicos sus *handheld* con un auténtico teclado, pero con una pantalla monocromática que le permitía una gran autonomía.
Las prestaciones eran mayores que las *PALM*, con muchísima mejor pantalla, y con un tamaño y peso muy reducido, y con mucha mas autonomía que las *WINDOWS CE*. Hace poco hicieron una alianza entre unas cuantas empresas, como *Psion*, *Ericsson*, *Nokia* , y *Psion* dejó de fabricar sus propios *handhelds*, centrándose en crear el sistema operativo Symbian, tomando como base el S.O. *EPOC*.

Este sistema operativo es el que llevan los *nokia* de última generación, el *nokia* 3650 y algún otro aparato.

Figura 5.4 Nokia 3660

Figura5.5 Psion Revo

Figura 5.6 Psion 5MX

5.1.4 WINDOWS CE

Llegamos a la parte que nos interesa, ya que es la máquina en la que está basado el proyecto.
Las características principales son:

Intento de copiar el Windows de sobremesa, con lo que se sobrecarga un montón el sistema operativo. Al principio eran máquinas con poca autonomía y muy lentas a pesar de llevar procesadores y hardware mucho más potente que sus competidores.

Procesadores mucho más potentes. Mientras que las *PALM* manejaban con soltura aplicaciones con 20 MHz, las que tenían *Windows CE* eran lentas con procesadores a 60 y 70 MHz

Mucha más memoria, usada con mucha menos eficiencia. Mientras tienes aplicaciones en *PALM* os de 20 o 30 kB, una aplicación básica para *pocketpc* puede llevar 200 o 300 kB.

Multimedia desde sus orígenes.

Más capacidades de expansión.

Microsoft llamó *Palmheld* a los equipos sin teclado y *handheld* a los que sí lo tenían. Más tarde, a partir de *Windows CE3.0*, para evitar equivocaciones con los *PALM,* pasó a denominar a los *Palmheld* con el nombre de *PocketPc*, luego *PocketPc 2002*, y la versión actual, *Windows Mobile 2003*, que va por su segunda edición (*Windows Mobile 2003 Second Edition*). Hay otra versión para casettes de coche. Además en los últimos años han añadido una cuarta versión, los smartphones, abandonando la versión para casettes de coche. Todos tienen la base de Windows CE, pero son productos diferentes y incompatibles.

Además la base de todos ellos, Windows CE, se usa en otros tipos de dispositivos, desde dispositivos sin pantalla para aplicaciones industriales hasta otros como la consola de videojuegos *Sony DreamCast*.

WINDOWS CE 1.0

La primera versión era bastante limitada, pero una revolución para su época. Tenían pantalla monocromática, sonido mono y bastantes ranuras de expansión. Pesaban mucho y tenían una autonomía bastante baja. No hubo muchos modelos, eran *handhelds*. (dispositivos con teclado)

Figura 5.7 Cassiopeia A20

WINDOWS CE 2.0

Aparece el color, y empieza la comercialización masiva. También los *Palmhelds*, que son la gran novedad de esta versión. Para programarlos se usaba o visual basic o visual c++, y se necesitaba el visual studio 6.0 y un añadido con la SDK de estos dispositivos.

Fig 5.8 HandHeld HP Jornada 620 LX

Velocidades de unos 60 MHz y memorias desde 4 hasta 16 megas.

Microsoft prepara la posibilidad de que se puedan compilar los programas para una gran variedad de procesadores, pero al final solo triunfan dos, *MIPS*, y *SH3*. Por supuesto, con las aplicaciones perfectamente incompatibles entre sí.

En cuanto al software, hay bastante diferencia entre la versión *handheld*, que tiene el aspecto de Windows, con su botón inicio, y sus aplicaciones típicas de office, pocket word, pocket powerpoint,pocket excel, y con escritorio, sin embargo la versión *Palmheld* ó *PPC* está mucho más limitada, sin estas aplicaciones y sin escritorio, necesitabas aplicaciones externas para conseguir hacer algo util, ya que según salía de la caja no se podía hacer mucho con ellas.

WINDOWS CE 2.11

Aparece el sonido estéreo, y empiezan a ser verdaderos aparatos multimedia, con capacidad para reproducir mp3, y archivos de video.

Hay versión para *handheld* y para *Palmheld*. Suben de memoria, empiezan a llevar 8 o 16 megas.Siguen con los dos procesadores a más velocidad, velocidades de hasta 130Mhz.

Figura5.9.Ejemplo Windows Ce2.11

WINDOWS CE 3.0

Es el verdadero despegue del Windows CE. Aparecen aparatos mucho más potentes, con procesadores *MIPS* y *SH3* velocidades entre 150, y 206Mhz , y entra en escena un nuevo procesador, el *StrongARM*, procesador que tendrá una importancia capital.

Las *PPC*, se pasan a denominar *pocketpc*. Nuestra aplicación funcionaría aquí. Memorias entre 16 y 32 Megas. Definitivamente instalan el pocket word y excel en todas las versiones, y alguna aplicación más, con lo que tienen una funcionalidad elevada nada más sacarlo de la caja.

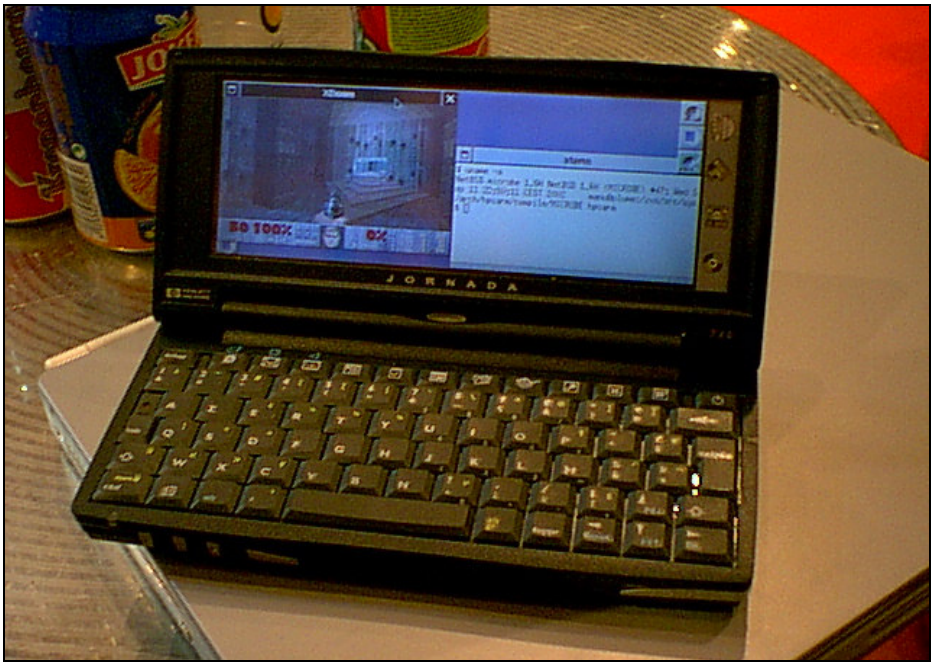


Fig 5.10 HP jornada 720, procesador StrongARM a 206, 32 Mb de memoria, ejecutando Doom2.

Para programar estos modelos, se necesitan las *Embedded Visual Tools*, que contiene sobre todo el entorno de programación de *Visual Basic* y de *VC++,* y las *SDK*(Software Development Kit), para poder compilar en una plataforma concreta una aplicación) para *pocketpc, handheld*, y para *Windows CE 2,11*. Es gratuito.. Además es independiente del *Visual Studio*.

Dentro de Windows CE 3.0, hay 3 generaciones y dos versiones principales para cada una de ellas.

La original, con *pocketpc*(sin teclado) y con "*Handheld PC 2000*".(con teclado) también se le denomina *pocketpc 2000*.

En versión sin teclado, siguen existiendo los tres procesadores, *ARM, MIPS y SH3*, pero con un cambio fundamental.

Fig 5.11 Compaq Ipaq serie 3600,

Hasta ahora, todos los modelos de agendas tenían la ROM fija, en su mayor parte no se podía actualizar, y como mucho, se podía cambiar en algunos modelos muy determinados la tarjeta de la ROM, es decir, los chips. El soporte de los fabricantes siempre fue penoso, ya que a ellos les interesaba que la gente cambiara de aparato en vez de S.O.

Sin embargo, con los procesadores *StrongARM* hay un cambio, y ponen memoria *Flash ROM,* Con lo que realmente se puede actualizar el sistema operativo sin cambiar chips. Incluso se ha llegado a poner *linux* en *IPAQs*.

Dentro de *Windows CE 3.0*, surje una nueva versión, que es *POCKETPC 2002*.. Se sube la memoria, entre 32 y 64 Mb. También aparecen los smartphones, que es una versión pensada para móviles, sin pantalla táctil, y con muchas posibilidades de comunicación.

Figura 5.12 Orange SPV con Windows CE

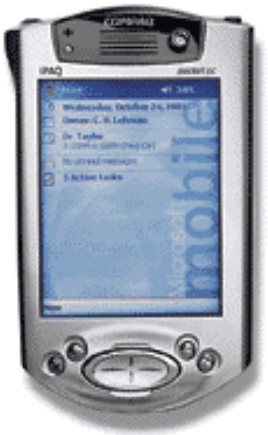


Figura 5.13 Compaq IPAQ 3960

Microsoft abandona otros procesadores que no sean de la familia StrongARM, así que sólo hay aparatos con este procesador y pocketpc 2002 en adelante. Los que tenían PocketPc 2000 y este procesador, en algunos casos se han podido actualizar a PPC2002, como las antiguas Ipaq de la serie 36XX. Los que tienen otros procesadores, no tienen esa opción. Tampoco pasa nada, ya que para actualizar los otros se necesitaría cambiar los chips de ROM.

Esto tiene una ventaja, sólo se necesita versión para un sólo procesador. Empiezan a converger las máquinas.

En esta época surge otro procesador, que es el que llevan los equipos actuales, el *Xcale*. Pero a diferencia de antes, este procesador SI es compatible plenamente con los *ARM*. Un programa escrito para un *ARM* funciona en un *Xcale*. Tienen velocidades de 200, 300 y 400MHz, y dos versiones compatibles entre sí, la 250 y la 255.

Al principio el rendimiento era decepcionante, ya que los flamantes *Xcale* a 400Mhz no conseguían superar la velocidad de los viejos ARM a 206Mhz por otras limitaciones en los buses del sistema., en parte por el intento de aprovechar el sistema operativo, y por el bajo rendimiento de la primera versión(250).

No sacaron versión nueva de handheld, y parece que Microsoft ha abandonado estos aparatos decantándose más por otras tecnologías como los Tablet PCs, basados en equipos Intel.

para programarlos se usan las Embedded Visual Tools versión 2002. también gratuitas.

WINDOWS MOBILE 2003

Está basado en otra evolución del núcleo del sistema operativo, *Windows CE 4.0 .NET*

Con pocos cambios de aspecto respecto a la versión anterior, aunque se ha partido de distinta base. Han conseguido mantener la compatibilidad con *Windows CE 3.0*

Figura 5.14

Resumiendo, tenemos:

Versión winCE 1.0 para handheld con procesador MIPS, y SH3

Versión winCE 2.0 para HPC y PPC con procesadores MIPS y SH3

Versión winCE 2.11 para HPC y PPC, procesadores MIPS y SH3

Versión winCE 3.0 2000 para Eadioce,HPC y PPC, con procesadores MIPS, SH3 y ARM

Versión winCE 3.0 2002 para PocketPc, Smartphones con procesadores de la familia ARM (StrongARM y Xcale)

Versión winCE 4.0 .NET para Windows Mobile 2003, Smartphones con procesadores de la familia ARM (StrongARM y Xcale)

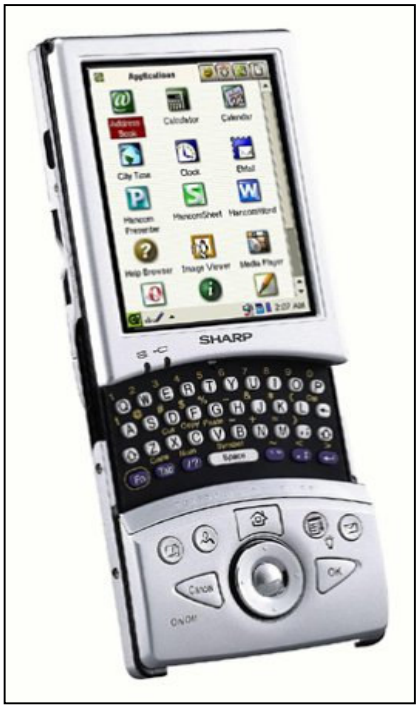


Figura 5.15 Sharp Zaurus

LINUX

Y un último comentario, hay un PDA, el *Sharp Zaurus,* que dispone de una versión reducida de LINUX.

Capítulo 6. DESARROLLO DE APLICACIONES PARA POCKET PC

6.1 Introducción

6.1.1 Inicios de la programación

La aparición de Windows a mediados de los años ochenta, sobre todo a raíz del lanzamiento de la versión 3.1, supuso una gran revolución en el mundo del PC. Los usuarios de esta plataforma, disponían ahora de un entorno gráfico de trabajo, que facilitaba en gran medida su labor y dejaba atrás paulatinamente la aridez del trabajo en el modo comando; ya no era necesario migrar a la plataforma Macintosh para disponer de un entorno de trabajo avanzado.

Sin embargo, el desarrollo de aplicaciones para el nuevo modo gráfico de modo comandos (aún no era propiamente un sistema operativo), distaba mucho de ser una tarea sencilla y rápida. Aquellos aventurados programadores, que se embarcaban en la gesta de desarrollar una aplicación para Windows, debían prácticamente, hacer borrón y cuenta nueva sobre todo lo que sabían, y comenzar casi, desde cero. Tan radical era el cambio, que hacer el más sencillo programa para que funcionara en Windows, se convertía en la más traumática de las experiencias.

Hasta ese momento, y en líneas generales, todo era más simple en la programación para modo comandos: la aplicación tomaba el control del sistema operativo, el cuál esperaba las instrucciones del programa para ir ejecutándolo; sólo podíamos tener en ejecución una aplicación en cada momento; el modo gráfico era proporcionado por librerías específicas del lenguaje que estuviéramos utilizando, etc.
Pero la nueva arquitectura de programación de Windows cambiaba todos los esquemas que pudiera conocer el programador: programación basada en eventos y orientada a objetos; modo gráfico proporcionado y gestionado por el sistema y no por el lenguaje; múltiples aplicaciones funcionando simultáneamente; y lo más novedoso, y también más traumático para los programadores, el hecho de que el sistema enviaba información mediante mensajes a nuestra aplicación, a los que debíamos dar una adecuada respuesta, lo que suponía que a partir de ese momento, era el sistema el que controlaba a la aplicación, con lo que se acabaron los tiempos en los que nuestro programa tomaba el control absoluto del sistema operativo.

En estos primeros tiempos de la programación para Windows, sólo los llamados *gurús* de C y Windows, que conocían perfectamente todos los trucos y la arquitectura del nuevo entorno operativo de Microsoft, eran capaces de desarrollar las nuevas aplicaciones, para el asombro de los más modestos *programadores de a pie*.

Uno de los grandes problemas para el programador, consistía en que debía centrarse excesivamente en el desarrollo de la parte del interfaz de la aplicación, controlando hasta el más mínimo detalle de lo que el usuario pudiera hacer con una ventana: captura y envío de mensajes desde y hacia las ventanas de la aplicación, gestión de manipuladores de ventanas y contextos de dispositivos para el dibujo de todos los elementos de la aplicación, escritura de los procedimientos de ventana, etc.; el más simple programa que mostrara un mensaje tenía un gran número de líneas de código.

En un escenario como este, en la mayor parte de casos, se desviaba la atención de lo verdaderamente importante en la aplicación: la funcionalidad que necesitábamos dar al usuario. Programar una simple entrada de datos para almacenar en un fichero era toda una odisea.
Por añadidura, tampoco existían herramientas de desarrollo que facilitaran la labor del programador, todo consistía en un puñado de aplicaciones independientes que funcionaban en modo comando:
compilador, enlazador, editor de código, etc., lo que hacía que un programador no pudiera alcanzar el mismo nivel de productividad que tenía desarrollando las aplicaciones MS-DOS de aquel entonces.
Esto suponía un grave inconveniente para Microsoft, puesto que el paso previo para popularizar su nuevo entorno de usuario para ordenadores personales, pasaba por la existencia de una comunidad de programadores lo más amplia posible, todos escribiendo aplicaciones para Windows; sin embargo, dada su dificultad, pocos eran los que se lanzaban a tal osado intento.

Conscientes del problema que entrañaba el que los desarrolladores no migraran de forma masiva a la creación de programas para Windows, Microsoft puso en marcha un proyecto con el nombre clave Thunder (Trueno), encaminado a crear una herramienta de desarrollo que facilitara la escritura de programas para Windows. En 1991, este proyecto dio como fruto la primera versión de Visual Basic

Nos encontramos en un momento muy importante en la historia de la informática en general, y la programación en particular; estamos en el punto de partida de una nueva generación de aplicaciones, que demandan una nueva tecnología, y que gracias al entorno .NET y a C#.NET, como una de sus herramientas integrantes, vamos a poder afrontar con plenas garantías de éxito.

6.2 La evolución hacia .NET

6.2.1 Cambios Realizados en la arquitectura

Los motivos que han llevado a Microsoft al desarrollo de .NET han sido tanto tecnológicos como estratégicos.

Respecto a las motivaciones tecnológicas, la necesidad de poner a disposición del programador una plataforma de desarrollo con plena potencia para abarcar los requerimientos de las nuevas aplicaciones que están a punto de llegar, y que no soporte incómodos lastres derivados de antiguos modelos de programación, ha desembocado en una tecnología totalmente nueva, que no arrastra pesadas incompatibilidades, pero que sin embargo, permite la ejecución de componentes basados en el anterior modelo de programación. Esto es .NET, una nueva arquitectura para el futuro del desarrollo de aplicaciones, y no, como en un principio pudiera pensarse, una operación más de marketing, que proporciona las herramientas ya conocidas con algunas remodelaciones y *lavados de cara*.

En cuanto a las causas estratégicas, gracias a .NET y a su modelo de distribución de software basado en servicios, Microsoft se sitúa en una posición clave en un mercado que evoluciona hacia la creación de servicios para la web, que serán utilizados por otras aplicaciones mediante un sistema de suscripción o alquiler. Se espera que en este potencial mercado, comiencen a aparecer empresas dedicadas a la producción y publicación de servicios en Internet. La propia Microsoft, ha expresado en este sentido, su intención de convertirse en proveedor de servicios.

6.2.2 Abandono de anteriores tecnologías

Los herméticos y poco flexibles modelos de programación actuales, impiden cada vez más al programador el abordaje de proyectos para Internet, que le permitan la creación de aplicaciones distribuidas más potentes.
Estos sistemas de trabajo, han evolucionado desde un esquema que integra diversas tecnologías como COM, ASP, ADO, etc., la mayor parte de ellas no pensadas inicialmente para ser ejecutadas en la Red, o que en el caso de ser diseñadas para Internet, arrastran elementos que no estaban pensados para funcionar en la web.
Todos estos elementos, conforman la arquitectura Windows DNA (Distributed interNet Architecture), que hasta la actualidad ha sido el modelo de programación para Internet propugnado por Microsoft.
Este modelo ha sido dejado a un lado para dar paso a .NET; lo que no supone una evolución de la actual arquitectura Windows DNA, sino que por el contrario, significa el nuevo comienzo de una arquitectura pensada para la Red.
Antes de describir en qué consiste .NET, hagamos un breve repaso de los problemas que plantea Windows DNA, de manera que podamos comprender mejor, los motivos por los cuales es necesaria la migración hacia la nueva plataforma de Microsoft.

6.2.3 La problemática de Windows DNA

Cuando a mediados de los años 90, Microsoft reorientó su estrategia hacia Internet, carecía de una herramienta de desarrollo potente y rápida para dicho entorno. Sin embargo necesitaba un producto para la programación en la Red y lo necesitaba ya. El resultado fue Windows DNA, que era bastante aceptable dado el apremio con el que debía dar respuesta a este sector del desarrollo de aplicaciones, aunque siempre ha adolecido de una falta de integración y facilidad de manejo, siendo un gran calvario para el desarrollador.

ASP

Las páginas ASP (Active Server Pages) son el medio con el que en Windows DNA, podemos programar aplicaciones para Internet utilizando la tecnología de Microsoft.
Aun cuando el resultado conseguido es satisfactorio, el hecho de ser código interpretado, carecer de una herramienta de depuración y poca estructuración suponen un grave paso atrás, máxime cuando todas las herramientas de desarrollo tienden progresivamente hacia un modelo orientado a objetos.

ADO

Este modelo de objetos para el acceso a datos fue diseñado inicialmente para ASP, pero dado su éxito, se traslado también a Visual Basic, para superar los inconvenientes que presentaban los obsoletos DAO y RDO.

El hecho de que se creara en un principio para ASP, puede hacernos pensar que es el medio perfecto para el acceso a datos en Internet; sin embargo, su diseño no se basa totalmente en un modo de acceso desconectado a los datos, ya que para que funcionara con mejor rendimiento dentro del mundo cliente/servidor de las aplicaciones VB, también se puede utilizar estableciendo una conexión permanente con el origen de datos del servidor, lo que supone un claro lastre a la hora de trasladarlo al mundo de Internet, en el que la conexión se establece sólo durante el tiempo que dura la operación a realizar con los datos (obtención, modificación)

Conflictos con DLL's

La instalación y mantenimiento de los componentes compilados en forma de DLL es otro de los importantes problemas existentes en la actualidad. La actualización de una DLL, cuando se produce un cambio en la misma y los conflictos de versión entre componentes, llevan a una inversión muy importante y grave de tiempo en corregir estos problemas.

COM

Una observación de la evolución de COM resulta reveladora y ayuda a comprender el camino que ha llevado hasta la creación de .NET.
El modelo de objetos basado en componentes (COM), se introdujo a mediados de los años 90 como una vía para conseguir un mayor aprovechamiento del código, al situarlo en componentes reutilizables por más de una aplicación.

A pesar de constituir un gran avance en el mundo de la programación, carecía de herencia, un aspecto muy importante y al que Microsoft anunció un próximo soporte, además de otras características, como el poder disponer de un modelo de objetos unificado que podría ser utilizado en diferentes plataformas; de hecho, se especuló con un cambio de nombre hacia *Common Object Model*, lo cual era muy significativo.

Sin embargo, y en contra de las expectativas, la siguiente versión, DCOM, siguió sin incorporar las características anunciadas, aunque eso no significaba que el equipo de desarrollo de COM no estuviera trabajando en ello.

Para la nueva versión, denominada COM+, se anunciaban cambios radicales en el panorama del desarrollo de componentes, en donde habría plenas capacidades de orientación a objetos (herencia incluida), los componentes se podrían escribir en un amplio abanico de lenguajes soportados por COM, la ejecución se realizaría en un entorno común que se haría cargo de la gestión de memoria y objetos, etc.

Aproximadamente en el mismo espacio de tiempo, otro equipo de desarrollo de Microsoft, después de la finalización de IIS 4, acumuló un conjunto de ideas para la creación de una nueva arquitectura, que provisionalmente se definió como Next Generation Windows Services (NGWS) o Nueva Generación de Servicios para Windows.

Al proyecto NGWS se incorporó Visual Studio y COM+ junto con MTS; sobre estos dos últimos, se comenzó a trabajar en todas las características comentadas antes, de forma que permitieran un entorno de ejecución común para todos los lenguajes de Visual Studio. El resultado fue .NET, y debido a los

profundos cambios sufridos por la integración de todos los elementos que lo forman, esta arquitectura no ha derivado directamente de COM, aunque muestra las principales características anunciadas para COM+.

Por todo lo anteriormente comentado, se puede afirmar que .NET es una nueva tecnología, y no una evolución del modelo Windows DNA; construida sin el peso de la compatibilidad hacia tecnologías anteriores, pero que ha sabido aprovechar las mejores ideas de los elementos existentes en la actualidad.

6.3 .NET Framework

El mundo del desarrollo de aplicaciones se encuentra sumido en una nueva etapa de transformación y evolución hacia nuevos esquemas de trabajo.

Los factores determinantes de dicho cambio los podemos encontrar en la necesidad de utilizar Internet como vehículo de intercambio por parte de diversos sectores de la economía.

Las empresas requieren establecer relaciones comerciales más dinámicas con sus clientes, de modo que su volumen de negocio se incremente a través del canal de ventas electrónico (el denominado comercio electrónico o e-commerce). Por otro lado también necesitan unas relaciones empresariales más ágiles en este mismo marco del ciberespacio (el llamado B2B o Bussiness to bussiness).

Aparte de todos estos elementos, nos encontramos con que el usuario de este medio, Internet, dispone de dispositivos cada vez más sofisticados para desplazarse por la Red, no sólo el PC; y además, exige que todos ellos permitan un acceso rápido y sencillo, a múltiples aplicaciones simultáneamente, con un mayor grado de interacción, y obteniendo información de un amplio conjunto de fuentes de datos; todo esto, naturalmente, sin los tradicionales esfuerzos de configuración que requieren algunas aplicaciones.

Con el paso del tiempo, Internet se ha convertido en el principal entorno de trabajo para el desarrollo de aplicaciones que gestionan información, haciendo que su alcance sea mayor que ningún otro medio hasta el momento. Baste pensar, que con un simple dispositivo que tenga acceso a Internet (léase un PC) y un programa navegador, es posible acceder a infinidad de sitios web basados en este paradigma.
Sin embargo, actualmente, la comunicación entre servidores es complicada (sobre todo si residen en plataformas distintas), y la integración de aplicaciones en dispositivos que no sean el típico PC, es limitada con las herramientas disponibles hasta la fecha. Pero no desesperemos, nos encontramos en un momento crucial, en el que todos esos inconvenientes pueden ser salvados gracias a un nuevo avance tecnológico: Microsoft .NET.

6.3.1 ¿Qué es .NET?

.NET es toda una nueva arquitectura tecnológica, desarrollada por Microsoft para la creación y distribución del software como un servicio. Esto quiere decir, que mediante las herramientas de desarrollo proporcionadas por esta nueva tecnología, los programadores podrán crear aplicaciones basadas en servicios para la web.

Las características principales que conforman .NET son las siguientes:

- La plataforma .NET Framework, que proporciona la infraestructura para crear aplicaciones y el entorno de ejecución para las mismas.

- Los productos de Microsoft enfocados hacia .NET, entre los que se encuentran Windows .NET Server, como sistema operativo que incluirá de forma nativa la plataforma .NET Framework; Visual Studio .NET, como herramienta integrada para el desarrollo de aplicaciones; Office .NET; b.Central para .NET, etc.
- Servicios para .NET desarrollados por terceros fabricantes, que podrán ser utilizados por otras aplicaciones que se ejecuten en Internet.

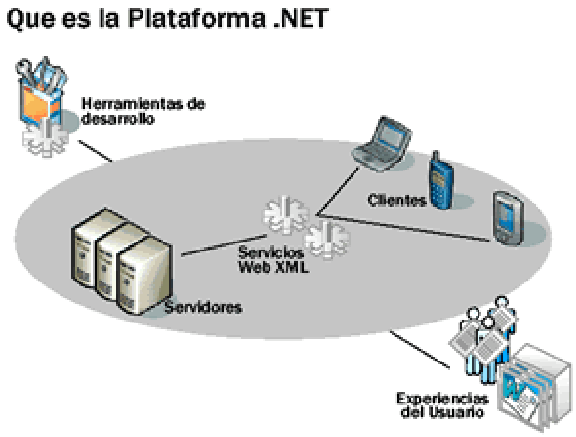


Figura 6.1 Qué es la plataforma .NET

Existen adicionalmente un conjunto de productos, que bajo la etiqueta de Servidores Empresariales para .NET (.NET Enterprise Servers) se incluyen dentro de la estrategia .NET. Entre estos productos podemos encontrar a SQL Server 2000, BizTalk Server, Commerce Server 2000, etc. Sin embargo, hemos de hacer una puntualización importante: estos productos no están basados en .NET Framework, pueden funcionar dentro del entorno de ejecución de .NET Framework, pero el único producto actualmente desarrollado bajo el nuevo entorno es Visual Studio .NET.

Gracias a .NET y a su modelo de desarrollo basado en servicios, se flexibiliza y enriquece el modo en el que hasta ahora se construían aplicaciones para Internet. La idea que subyace bajo esta tecnología, es la de poblar Internet con un extenso número de aplicaciones, que basadas en servicios para la web (Servicios Web), formen un marco de intercambio global, gracias a que dichos servicios están fundamentados en los estándares SOAP y XML, para el intercambio de información.
En este sentido, un programador puede crear Servicios Web para que sean utilizados por sus propias aplicaciones a modo de componentes (pero de una forma mucho más avanzada que empleando el modelo COM clásico), siguiendo una estructura de programación ya conocida.

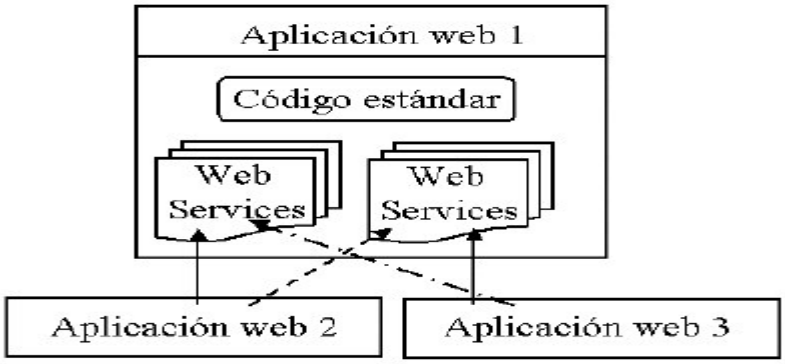


Figura 6.2 Esquema de funcionamiento de aplicación web incluyendo Servicios Web.

Sin embargo, los Web Services traen de la mano un nuevo modelo de distribución del software; el basado en el desarrollo y publicación de Web Services y en la suscripción a los mismos por parte de otras aplicaciones, potenciales usuarios de tales servicios.

Los fabricantes de software, pueden de esta manera, dedicarse a la creación de servicios web y a su alquiler. Nace de esta manera, la figura del proveedor de servicios web.

Dado el esquema anterior, el programador puede construir sus aplicaciones a base de Servicios Web, reduciendo significativamente el tiempo y esfuerzo en el desarrollo.

Los **Componentes** de la plataforma .NET son:

- Smart Clients(Clientes Inteligentes): Son dispositivos muy variados. Lo que los hace 'Smart' o inteligentes es su capacidad para hacer uso de servicios Web. Sus características son:

    - Permiten acceder a la información en el formato apropiado, en cualquier momento y lugar
    - Hacen uso de Servicios Web
    - Optimizan de distintas maneras la forma en que la información es presentada y organizada. Por ejemplo: Pueden convertir texto en sonido en un celular o reconocer la escritura en un TabletPC
    - Proveen de una interfase sencilla y natural para que el usuario acceda a la información. Pueden utilizar la identidad del usuario, su perfil y datos para adaptar la información que es presentada.
    - Pueden reconocer la presencia de otros dispositivos e intercambiar información
    - Pueden adaptarse a las características de la red donde están. Por ejemplo la velocidad de transmisión
    - Tienen capacidad de procesamiento propio, y distribuyen el procesamiento en la red haciendo uso de los servicios Web.

- Pocket PC (PC de bolsillo)

- SmartPhone (Teléfono Inteligente)

- HandHelds

- TabletPC

- XBox la consola de juegos de Microsoft

- PCs: Las computadoras personales

- NoteBooks: Las computadoras Portátiles

Muchos otros dispositivos en desarrollo

Servidores:
Proveen de la infraestructura para implementar el modelo de computación distribuida en Internet. Son Sistemas operativos y de Aplicación.

Sistemas Operativos:
- Windows 2000: Server, Advance Server y Datacenter
- Windows .NET Server: Standard, Enterprise, Datacenter y Web Server

6.4 .NET Framework

.NET Framework constituye la plataforma y elemento principal sobre el que se asienta Microsoft

.NET. De cara al programador, es la pieza fundamental de todo este nuevo modelo de trabajo, ya que proporciona las herramientas y servicios que necesitará en su labor habitual de desarrollo.

.NET Framework permite el desarrollo de aplicaciones a través del uso de un conjunto de herramientas y servicios que proporciona, y que pueden agruparse en tres bloques principales: el Entorno de Ejecución Común o Common Language Runtime (CLR a partir de ahora); la jerarquía de clases básicas de la plataforma o .NET Framework Base Classes; y el motor de generación de interfaz de usuario, que permite crear interfaces para la web o para el tradicional entorno Windows, así como servicios para ambos entornos operativos. La Figura 3 muestra un diagrama con la distribución de elementos dentro del entorno de .NET Framework.

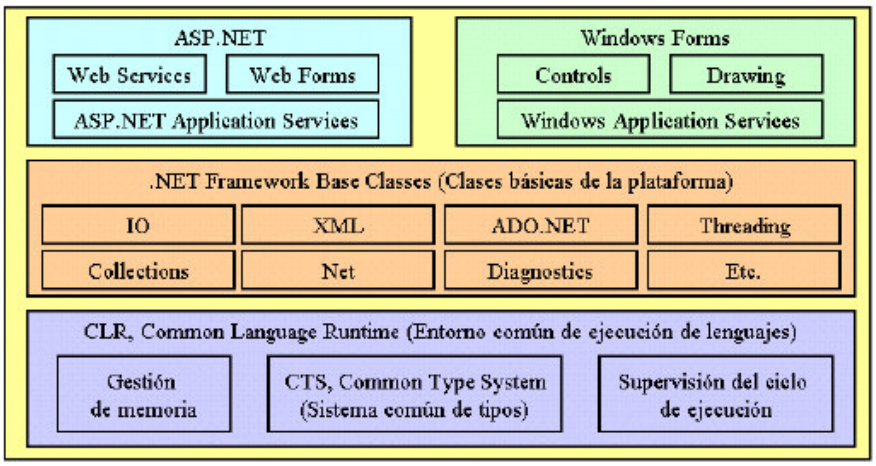


Figura 6.3 Esquema de componentes dentro de la plataforma .NET Framework.

En la base del entorno de ejecución, se encuentra el CLR, que constituye el núcleo de .NET Framework, encargándose de la gestión del código en cuanto a su carga, ejecución, manipulación de memoria, seguridad, etc.

En el nivel intermedio, se sitúa la jerarquía de clases básicas del entorno de ejecución, que constituyen un sólido API de servicios a disposición del programador, para multitud de tareas como, gestión del sistema de ficheros, manipulación multihebra, acceso a datos, etc.

Finalmente, en el nivel superior, encontramos las clases que permiten el diseño del interfaz de usuario de nuestras aplicaciones. Si necesitamos desarrollar aplicaciones para Internet, utilizaremos ASP.NET, que nos provee de todo lo necesario para crear aplicaciones para la Red: formularios web ,servicios web, etc.

Y no piense el programador tradicional de Windows, que todo en .NET Framework es programación para Internet. La plataforma no se ha olvidado de este colectivo de programadores, que necesitan desarrollar programas para este sistema operativo, y pone a su disposición los denominados Windows Forms (Formularios), la nueva generación de formularios, con características avanzadas y muy superiores a las del motor de generación de formularios de VB6.

Adicionalmente, existe la posibilidad de que necesitemos servicios del sistema que no requieran interfaz de usuario en absoluto. Este aspecto también está contemplado por la plataforma, permitiéndonos, por ejemplo, la creación de servicios para Windows 2000 y NT.

6.4.1 El CLR, Common Language Runtime

El Entorno de Ejecución Común de Lenguajes o CLR (Common Language Runtime), representa el alma de .NET Framework y es el encargado de la ejecución del código de las aplicaciones.

A continuación se enumeran algunas de las características de este componente de la plataforma:

- Proporciona un desarrollo de aplicaciones más sencillo y rápido gracias a que gran parte de las funcionalidades que tradicionalmente debía de crear el programador, vienen implementadas en el entorno de ejecución.
- Administra el código en tiempo de ejecución, en todo lo referente a su carga, disposición en memoria, recuperación de memoria no utilizada a través de un recolector de memoria, etc.
- Implementa características de gestión a bajo nivel (administración de memoria, por ejemplo), que en ciertos lenguajes, eran labor del programador.
- Proporciona un sistema común de tipos para todos los lenguajes del entorno.
- Gestiona la seguridad del código que es ejecutado.
- Dispone de un diseño abierto a lenguajes y herramientas de desarrollo creadas por terceros fabricantes.
- Facilita enormemente la distribución e instalación de aplicaciones, ya que en teoría, es posible instalar una aplicación simplemente copiando los ficheros que la componen en uno de los directorios del equipo en el que se vaya a ejecutar, eliminando los temibles conflictos de versiones entre librerías, problema conocido también con el nombre de *Infierno de las DLL* o *DLL Hell*.

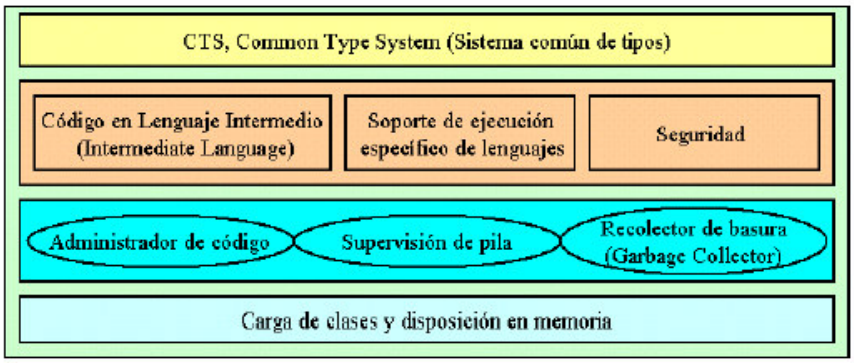


Figura 6.4 Esquema de elementos dentro del CLR.

En los siguientes apartados, haremos una descripción de los elementos y características más destacables del CLR, que permitan al lector obtener una visión global del mismo, y de las ventajas de escribir programas para este entorno de ejecución.

6.4.2 El CTS, Common Type System

El Sistema Común de Tipos o CTS (Common Type System), es el mecanismo del CLR que permite definir el modo en que los tipos serán creados y manipulados por el entorno de ejecución de .NET Framework.

Entre las funcionalidades que comprende, podemos destacar la integración de código escrito en diferentes lenguajes; optimización del código en ejecución; un modelo de tipos orientado a objeto, que soporta múltiples lenguajes; y una serie de normas que aseguran la intercomunicación entre objetos.

El sistema común de tipos (CTS a partir de ahora), como hemos indicado, permite definir o diseñar el modo en cómo el código de la aplicación será ejecutado, pero no se encarga directamente de su ejecución; dicho de otro modo, el CTS le dice al CLR cómo quiere que sea ejecutado el código.

Un ejemplo de las ventajas del CTS, consiste en que desde un lenguaje como VB.NET, podemos instanciar un objeto de una clase escrita en otro lenguaje como C#; y al hacer una llamada a uno de los métodos del objeto, no es necesario realizar conversiones de tipos en los parámetros del método, funcionando todo de forma transparente.

6.4.3 ¿Qué es un tipo dentro de .NET Framework?

Al mencionar el sistema de tipos de la plataforma .NET, podemos pensar de un modo inmediato, que se trata sólo del conjunto de tipos de datos con que podemos declarar variables en nuestro código; sin embargo, el CTS, va mucho más allá y se extiende a cualquier elemento que pueda ser ejecutado dentro del entorno.

Por tal motivo, en el contexto de .NET Framework, un tipo se puede definir como una entidad de código ejecutada dentro del CLR; entendiendo por entidad de código, aquella a partir de la cual creamos una instancia y manejamos posteriormente en el programa como un objeto.

De todo lo anterior podemos obtener dos conclusiones:

- Todas las implementaciones de clases, interfaces, estructuras, etc., ya sean nativas de la plataforma o creadas por el programador, se pueden considerar tipos válidos de .NET.
- Todos los tipos que manipulamos dentro de .NET Framework son objetos.

6.4.4 Los tipos de datos son objetos

Dentro de .NET Framework, todos los tipos de datos están implementados como clases, de ahí el hecho de que cuando declaremos una variable en el código, esa variable sea además, un objeto de la clase relacionada con el tipo de dato que contiene, disponiendo de propiedades y métodos al igual que cualquier otro objeto.

6.4.5 Categorías de tipos

Los tipos creados por el CTS pueden clasificarse en dos grupos principales, según el modo en el que se almacenan y manipulan en memoria:

- **Tipos por valor.** Un tipo creado por valor, almacena un dato que puede ser accedido de forma directa. Los tipos por valor se organizan a su vez en varios subgrupos, como son los tipos de datos nativos de la plataforma .NET, tipos de datos creados por el programador y tipos enumerados.
- **Tipos por referencia.** Un tipo creado por referencia, contiene la dirección de memoria en donde reside un dato. Para acceder a dicho dato, lo hacemos de forma indirecta utilizando esa dirección de memoria o referencia. Los tipos por referencia se organizan a su vez en varios subgrupos, como son las clases propias de la plataforma, las clases creadas por el programador, interfaces, delegates, etc.

6.4.6 La disposición de los datos en la memoria

Explicado de un modo muy elemental, cuando ejecutamos una aplicación, los datos que utiliza dicha aplicación se sitúan en memoria. La memoria se divide en dos zonas: una denominada *pila* o *stack*, pequeña y compacta pero de muy veloz acceso a datos y rápido proceso; y otra denominada *montón* o *heap*, de mayor tamaño pero más lenta.

El modo en como el CLR maneja los tipos en la memoria, tiene una gran importancia de cara a conseguir el mayor rendimiento posible en la ejecución del programa. Es una tarea que gestiona de forma automática el entorno de ejecución.

Cuando se crea un tipo por valor, una variable de tipo numérico, por ejemplo; el valor de dicha variable se sitúa en la pila, de forma que su acceso es directo. Al destruir un tipo por valor, el valor que se almacena en la pila también es destruido. Si asignamos una variable de estas características a otra, se crea en memoria una copia del valor original, con lo que tenemos en este caso dos tipos o variables con valores iguales. Un tipo por valor no puede tener un valor nulo.

Cuando creamos un tipo por referencia, la instancia de una clase (un objeto) que asignamos a una variable, por ejemplo; dicho tipo se sitúa en el montón. Una variable de este tipo contiene la referencia a un valor, no el propio valor, por lo que si asignamos una variable que contiene un tipo por referencia a otra variable, se dice que ambas apuntan o se refieren al mismo valor. Un tipo por referencia sí puede contener un valor nulo.

Como podemos comprobar, la relación que la memoria tiene con respecto a los tipos de .NET es muy importante, ya que dependiendo de donde sean ubicados, se conseguirá un rendimiento mas o menos óptimo en la ejecución del programa.

6.4.7 Metadata (metadatos)

Durante el diseño de .NET Framework, se hizo evidente que una aplicación necesitaba, además de su propio código ejecutable, información adicional sobre la propia aplicación, que pudiera ser utilizada por el entorno de ejecución para funcionalidades diversas.

Para resolver este problema, se optó por incluir toda esta información complementaria dentro de la propia aplicación. A la información que va incluida en la aplicación pero que no forma parte del código ejecutable se le denomina metadatos, y con esta técnica obtenemos aplicaciones o componentes auto-descritos.

Los metadatos son creados por el compilador del lenguaje utilizado en cada caso y grabados dentro del fichero resultante (EXE o DLL) en formato binario, siendo el CLR el encargado de recuperarlos en el momento que los necesite.

Algunos de los datos proporcionados por los metadatos de una aplicación son la descripción del ensamblado (trataremos los ensamblados posteriormente) junto a su versión, clave y tipos que lo componen (clases, interfaces, etc.).

6.4.8 Soporte multi-lenguaje

Uno de los puntos clave del CLR es que está diseñado para soportar múltiples lenguajes, permitiendo así unos elevados niveles de integración entre los mismos. Con tal motivo, .NET Framework proporciona los siguientes lenguajes con sus correspondientes compiladores para la escritura de aplicaciones:

- VB.NET.
- C#.
- C++ con Extensiones Administradas.
- JScript.NET.

Por integración de lenguajes podemos definir algo tan poderoso como el hecho de escribir una clase en C#, y heredar de dicha clase desde VB.NET. Esto permite formar grupos de trabajo heterogéneos, en los que cada integrante del grupo, puede escribir el código de una aplicación en el lenguaje de su preferencia.

Gracias a que el entorno de ejecución es común, y el código compilado no pasa directamente a código ejecutable puro, sino a un código intermedio (lo veremos más adelante), podemos crear nuestros programas en el lenguaje con el que nos sintamos más cómodos en cuanto a sintaxis y prestaciones con la ventaja de que la velocidad de ejecución será muy parecida a la obtenida habiendo escrito el código en otro lenguaje en principio más rápido como C++ o C#.

6.4.9 El CLS (Common Language Specification)

La integración entre lenguajes mencionada en el anterior apartado, puede llevar a preguntarnos cómo es posible conseguir que lenguajes de distinta naturaleza y sintaxis *se entiendan*.

La respuesta la hallamos en la Especificación Común de Lenguajes o CLS (Common Language Specification), que consiste en un conjunto de características comunes, que deben cumplir todos los lenguajes de la plataforma, para poder integrarse entre sí.

Esto tiene varias finalidades, que describimos a continuación:

- **Independencia del lenguaje**. En muchas ocasiones el programador se ve obligado a escribir el código en un lenguaje que no es de su agrado; la causa de ello es que dicho lenguaje le provee de funcionalidades de las cuales carece su lenguaje preferido. Con .NET, esto no ocurre, puesto que es la propia plataforma la que proporciona la funcionalidad de modo independiente al lenguaje, por lo que podemos escribir nuestras aplicaciones utilizando el lenguaje con el que nos sintamos más cómodos, ya que el resultado será el mismo.
- **Integración entre lenguajes**. Es posible escribir, por ejemplo, una librería de clases en un lenguaje, y utilizarla desde otro lenguaje distinto (siempre que ambos lenguajes cumplan con las normas del CLS). Este concepto no es nuevo, hasta ahora también podíamos escribir una librería en C++ y utilizarla desde VB, pero gracias al CLS, se extiende y se potencia este modo de trabajo, ya que al basarse los lenguajes en un conjunto de reglas comunes, el acceso en el caso antes mencionado, a una librería de clases, se facilita enormemente desde cualquier otro lenguaje creado en base al CLS.

- **Apertura a nuevos lenguajes**. Finalmente, al ser esta, una especificación abierta, es posible incorporar a .NET Framework nuevos lenguajes, aparte de los actualmente disponibles, y no sólo creados por Microsoft, sino por cualquier otro fabricante. Mediante el CLS, un fabricante de software sabe qué requisitos debe observar un nuevo lenguaje que él desarrolle, para poder integrase en el entorno

6.4.10 Ejecución administrada

La ejecución administrada se trata de un conjunto de elementos existentes en .NET Framework, que supervisan el código del programa durante su ejecución dentro del CLR, asegurándose de que el código cumple todos los requisitos para poder hacer uso de los servicios proporcionados por el entorno de ejecución, y beneficiarse de sus ventajas.

6.4.11 Código administrado

El código que escribamos orientado a utilizar todas las cualidades del CLR se denomina código administrado. Por defecto el código escrito en VB.NET, C# y JScript.NET es administrado, con lo que el programador no debe preocuparse en configurar de manera especial su proyecto.

Por el contrario, el código escrito en C++ no es administrado por defecto, lo que significa que el entorno no lo supervisa y no garantiza su fiabilidad al ser ejecutado por el CLR. Si el programador de C++ quiere que su código sea administrado debe utilizar las extensiones administradas que la plataforma proporciona para este lenguaje y activarlas a través de una opción del compilador.

El hecho de que el entorno realice labores de comprobación sobre el código, supone evidentemente, una labor extra que repercute sobre el rendimiento final a la hora de ejecutar el programa. Sin embargo, las pruebas realizadas ofrecen como resultado una pérdida de un 10% en el rendimiento del código administrado con respecto al código no administrado.

Teniendo en cuenta los niveles de seguridad que nos ofrece el código administrado y dado que la velocidad de los procesadores evoluciona, esta pérdida de rendimiento que supone la ejecución administrada posiblemente no sea significativa en un corto plazo de tiempo.

6.4.12 Datos administrados

De forma similar al código, los datos administrados son datos los datos de la aplicación gestionados en memoria por el CLR a través de un mecanismo denominado recolector de basura.

Al igual que en el punto anterior, los datos son administrados por defecto en las aplicaciones escritas en VB.NET, C# y JScript.NET. Si utilizamos en cambio C++, los datos de la aplicación no son administrados por defecto, debiéndolo indicar en el código del programa.

6.4.13 Recolección de memoria no utilizada

Durante la ejecución de un programa, los datos cargados en memoria por dicho programa dejan de ser utilizados en algún momento, por lo que ocupan un espacio que puede ser muy necesario para otros procesos.

En muchos lenguajes, la gestión de memoria es tarea del programador, el cual debe preocuparse de asignar y liberar memoria para el programa, escribiendo el código necesario.

6.4.14 Recolección de memoria en .NET Framework

En lo que respecta a .NET Framework, el CLR implementa un supervisor de memoria denominado Garbage collector o recolector de basura, que se ocupa de hallar los objetos (datos administrados) de la aplicación que no son utilizados y liberar la memoria que usan, realizando de manera adicional, una compactación sobre la memoria, para optimizar el rendimiento sobre el área de trabajo de la aplicación.

Cuando un objeto ya no es necesario, debemos ejecutar su método Dispose( ), en el cual se deberá incluir el código que se ocupe de liberar los recursos que utilice el objeto; esta acción comunica al CLR que hay un objeto cuya memoria no es necesitada por la aplicación, aunque esto no quiere decir que dicha memoria se libere inmediatamente.

La liberación de memoria se producirá cuando el CLR así lo requiera; esto sucede cuando la zona de memoria reservada para las instancias del objeto, denominada *montón administrado*, se llene; en ese momento, el CLR activará el recolector de basura que se encargará de liberar y compactar la memoria no utilizada.

6.4.15 La ejecución de código dentro del CLR

El proceso de ejecución del código en el entorno de .NET Framework, varía notablemente respecto al modo de ejecución tradicional de programas. En este apartado, realizaremos un repaso de los elementos y técnicas que intervienen en dicho proceso, de modo que el lector tenga un conocimiento más detallado de lo que sucede con el código, desde que termina de escribirlo, y hasta el resultado obtenido tras su ejecución.

6.4.16 El IL, Intermediate Language

Durante el proceso de compilación, el código fuente es tomado por el compilador del lenguaje utilizado para su escritura, y convertido, no directamente a código binario, sino a un lenguaje intermedio, que recibe el nombre de Microsoft Intermediate Languaje (MSIL o IL).
Este lenguaje o código intermedio (IL a partir de ahora), generado por el compilador, consiste en un conjunto de instrucciones que son independientes del sistema operativo o procesador en el que vaya a ejecutarse el programa, y que se ocupan de la manipulación de objetos, accesos a memoria, ...

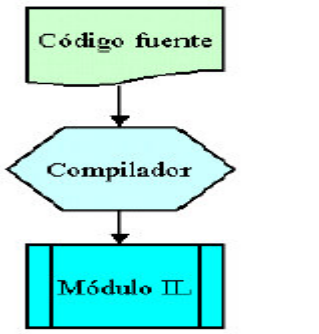


Figura 6.5 Obtención de lenguaje intermedio a partir de código fuente.

Además del código en IL, el compilador genera también metadatos, que como se ha explicado en un apartado anterior, contienen información adicional, incluida en la propia aplicación, y que serán utilizados por el CLR al ejecutar el programa.

Tanto el código en IL, como los metadatos generados, se guardan en un fichero de tipo EXE o DLL, basado en la especificación tradicional de Microsoft para ficheros con formato de ejecutable transportable (Portable Executable o PE) y objeto común (Common Object File Format o COFF). Con el desarrollo de la tecnología .NET, esta especificación ha sido ampliada para dar cabida, además de código binario, código IL y metadatos.

Ya que el código obtenido en IL es independiente del procesador, en su estado actual no es posible todavía ejecutarlo, debido a que el IL no ha sido diseñado para conocer las instrucciones específicas del procesador en el que se va a ejecutar. La ejecución se lleva a cabo, realizando el paso final de compilación que se detalla seguidamente.

6.4.17 Compilación instantánea del IL y ejecución

Como acabamos de indicar, el código en lenguaje intermedio no es directamente ejecutable, ya que desconoce la arquitectura del procesador sobre la cual va a funcionar.

Antes de realizar la ejecución, el código en IL debe ser convertido a código máquina, utilizando lo que se denomina un compilador instantáneo o compilador Just-In-Time (JIT compiler), que es el encargado de generar el código binario específico para el procesador en el que el programa será ejecutado.

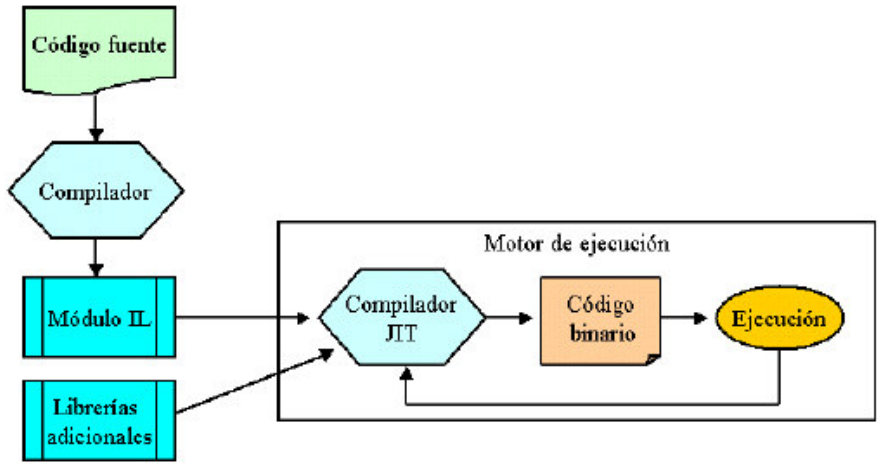


Figura 6.6 Proceso de compilación instantánea de código IL.

6.4.18 Compilación bajo demanda

Para optimizar la ejecución y mejorar su velocidad, el compilador JIT se basa en el hecho de que es posible que ciertas partes del código que compone la aplicación nunca sean ejecutadas. Por este motivo, al ejecutar la aplicación, no se toma todo su IL y se compila, sino que sólo se compila el código según se va necesitando y se almacena el código máquina resultante de modo que esté accesible en las siguientes llamadas.

6.4.19 Independencia de plataforma

Ya que el código máquina ejecutable, es obtenido a través de un compilador JIT, con las instrucciones adecuadas para un procesador determinado, .NET Framework proporciona varios compiladores JIT para cada una de las plataformas que soporta, consiguiendo así que la aplicación, una vez escrita, pueda funcionar en distintos sistemas operativos, y haciendo realidad el objetivo de que nuestro código sea independiente de la plataforma en la que se vaya a ejecutar, actuando .NET Framework como una capa intermedia, que aísla el código del sistema operativo.

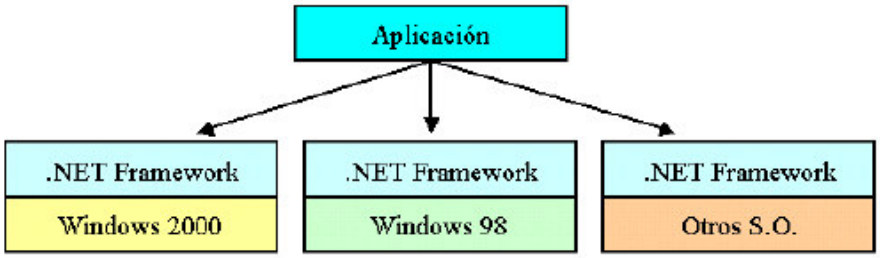


Figura 6.7 Una misma aplicación se ejecuta en distintos sistemas a través de .NET Framework.

6.4.20 Dominios de aplicación

En .NET Framework se han reforzado las características de seguridad y aislamiento hasta un nivel que permite la ejecución de múltiples aplicaciones en un mismo proceso. A este contexto de ejecución de un programa se le denomina *dominio de aplicación* (Application Domain).

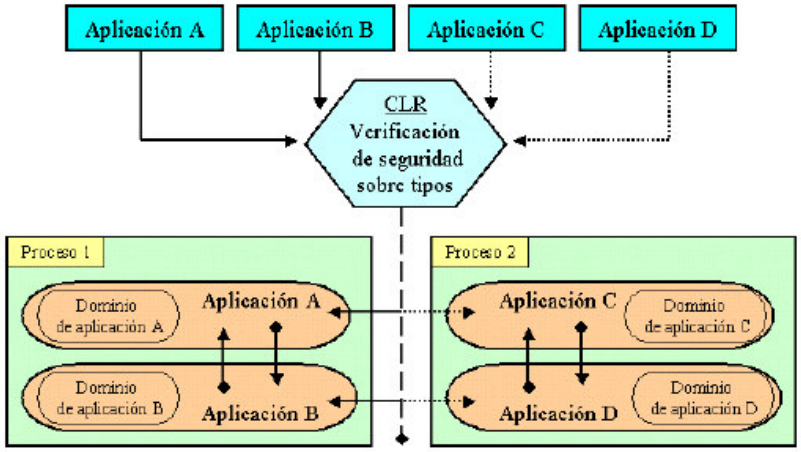


Figura 6.8 Carga de aplicaciones y creación de dominios de aplicación en procesos.

La técnica utilizada tradicionalmente para conseguir aislar las aplicaciones, de modo que no se produzcan colisiones entre las mismas, ha sido a través de procesos. Cada aplicación se carga en un proceso separado, que proporciona el adecuado nivel de aislamiento; de este modo, se evitan posibles conflictos entre las direcciones de memoria utilizadas por cada programa. Sin embargo, esto supone un gran consumo de recursos, cuando las aplicaciones deben hacer llamadas a otras aplicaciones que residan en procesos distintos, debido a que se debe de realizar un traspaso de procesos entre la

aplicación que realiza la llamada y la aplicación destino. Esta técnica ha sido mejorada en .NET, de modo que se consigue tener en un mismo proceso, varias aplicaciones en ejecución.

El código administrado en .NET Framework, para poder ser considerado como seguro, debe pasar en primer lugar una fase de comprobación, efectuada por el CLR, que asegure el hecho de que no realice ningún acceso no permitido a direcciones de memoria u otras operaciones que puedan provocar un fallo del sistema. Una vez superada dicha comprobación, el código es marcado como seguro a nivel de tipos (type-safe), y la aplicación ejecutada.

Superada esta fase de verificación, el programa se ejecutará en un dominio de aplicación, que como hemos comentado antes, consiste en una técnica que permite ejecutar varias aplicaciones en un único proceso, con el mismo nivel de aislamiento que si se estuvieran ejecutando en procesos separados, y la ventaja de eliminar la sobrecarga producida cuando distintas aplicaciones están situadas en diferentes procesos y deben hacerse llamadas entre sí. Cada aplicación se ejecuta en su propio dominio de aplicación.

Los dominios de aplicación incrementan notablemente la capacidad de crecimiento de los servidores al ejecutar múltiples aplicaciones en un mismo proceso.

6.4.21 Servidores de entorno

Un servidor de entorno o Runtime Host es el encargado de ejecutar un dominio de aplicación dentro del CLR, aprovechando las ventajas proporcionadas por este último.

Cuando el CLR se dispone a ejecutar una aplicación, un servidor de entorno crea el entorno de ejecución o shell para dicha aplicación, y lo carga en un proceso; a continuación, crea un dominio de aplicación en ese proceso y por último carga la aplicación en el dominio.

.NET Framework dispone entre otros, de los servidores de entorno relacionados a continuación:

- **ASP.NET**. Carga el entorno en un proceso preparado para gestionarse en la web; creando también, un dominio de aplicación para cada aplicación de Internet ejecutada en un servidor web.
- **Internet Explorer**. Crea un dominio de aplicación por cada sitio web visitado, en el que se ejecutan controles administrados basados en el navegador.
- **Windows Shell**. Crea un dominio de aplicación con interfaz Windows, para cada programa que es ejecutado.

.

6.4.22 Namespaces

Otro de los pilares que forman los cimientos de .NET Framework es el concepto de *espacio de nombres* o *namespaces*.

Un namespace o espacio de nombres, también denominado nombre calificado, es el medio proporcionado por la plataforma para organizar las clases dentro del entorno, agrupándolas de un modo más lógico y jerárquico

.

Cuando creamos un proyecto dentro de Visual Studio .NET, esta herramienta ya se encarga de crear de forma automática un namespace con el mismo nombre del proyecto. En el caso de que sea el

programador quien quiera crear un namespace de forma explícita, puede hacerlo mediante la palabra clave Namespace dentro del código del proyecto.

Para acceder desde el código de una aplicación, a una clase contenida dentro de un espacio de nombre, debemos indicarlo en la aplicación realizando una operación que en VB.NET se denomina *Importar*.

Existen dos medios para importar un espacio de nombre: usar la palabra clave Imports en la cabecera del módulo de código junto al nombre del namespace y clase a la que queremos acceder; o bien usar la descripción calificada completa en cada momento que necesitemos hacer referencia a la clase.

La convención sintáctica para hacer referencia a una clase contenida en un espacio de nombre, es como acabamos de ver, el espacio de nombre y la clase separados por un punto. En el caso de accedera una clase que se encuentra con varios espacios de nombre de profundidad, especificaremos dichos espacios de nombre separados por un punto, e igualmente al final, la clase. La inclusión al final del nombre de la clase, depende de si instanciamos directamente el objeto usando la lista de espacios de nombre o importamos dicha lista.

En el caso de instanciar un objeto directamente en el código, escribiremos los espacios de nombre y al final, el nombre de la clase. Si importamos los espacios de nombre, no debemos poner el nombre de la clase, sino que debemos terminar con el espacio de nombres que contiene la clase que necesitamos.

Todas las clases de la plataforma .NET están contenidas dentro de espacios de nombre, por lo que siempre que necesitemos instanciar un objeto, deberemos hacerlo usando la convención de espacios de nombre y puntos explicada anteriormente.

Las clases principales de .NET Framework están, por consiguiente, incluidas también en sus correspondientes namespaces. Como muestra el ejemplo anterior, si queremos instanciar un objeto para un formulario (Button, TextBox, etc.) debemos usar el espacio System.Windows.Forms, y dentro de este la clase que necesitemos. Como habrá podido adivinar el lector, el namespace System constituye el espacio raíz, a partir del cual, descienden el resto de espacios de nombre y clases de la plataforma, como IO, Threading, Collections, etc.

6.4.23 La jerarquía de clases de .NET Framework

El entorno de ejecución integra toda la funcionalidad y servicios necesarios a través de la jerarquía de clases base de la plataforma. La mayor parte de las necesidades básicas del programador están cubiertas por este amplio conjunto de clases, que permiten dotar a las aplicaciones de todas las características necesarias.

El desarrollador experimentado puede estar preguntándose la necesidad de implementar una nueva jerarquía de clases si las actuales ya cumplen con su cometido. Entre las posibles razones, queremos destacar las siguientes:

- El nuevo sistema de clases está mucho mejor organizado, y provee al programador de una potencia y versatilidad para sus aplicaciones nunca antes lograda en versiones anteriores de Visual Studio.
- Podemos crear una nueva clase, heredando de una clase propia de la plataforma, para extender su funcionalidad.
- Desplazando la funcionalidad de las clases fuera de los lenguajes, y haciéndolas por lo tanto, independientes de los mismos, simplifica el proceso de desarrollo.
- Al ser las clases de .NET Framework, comunes a todos los lenguajes, se eliminan las barreras tradicionales que impedían a los programadores abordar ciertos proyectos por el hecho de usar un lenguaje que no disponía de cierta funcionalidad que sí tenía otro lenguaje. Ahora cualquier

programador, con independencia del lenguaje que elija, tiene pleno acceso a todas las funcionalidades que le brinda la plataforma .NET.

Dentro de .NET Framework, System designa al espacio de nombre principal o raíz, a partir del cual, descienden todos los espacios de nombre y clases de la plataforma.

Además de las clases que proporcionan acceso a los tipos de datos intrínsecos de .NET Framework, System nos permite el acceso a otros servicios.

6.4.24 Ensamblados

Un *ensamblado* o *assembly*, consiste en un conjunto de tipos y recursos, reunidos para formar la unidad más elemental de código que puede ejecutar el entorno de .NET Framework.

De igual forma que los edificios se crean a base de la unión de un conjunto de materiales, dentro de la tecnología .NET, los ensamblados se presentan como los *bloques de construcción* software, que se unen o ensamblan para crear aplicaciones. Una aplicación desarrollada para .NET Framework debe estar compuesta por uno o varios ensamblados.

Podemos establecer una analogía entre un ensamblado y una DLL, ya que ambos contienen clases, que se exponen a otras aplicaciones. Por dicho motivo, a un ensamblado también se le da el nombre de *DLL lógica*; el término *DLL* se emplea porque tiene un comportamiento similar al de las DLL's tradicionales, y el término *lógica* porque un ensamblado es un concepto abstracto, ya que se trata de una lista de ficheros que se referencian en tiempo de ejecución, pero que no se compilan para producir un fichero físico, a diferencia de lo que ocurre con las DLL's tradicionales.

Sin embargo, un ensamblado extiende sus funcionalidades a un horizonte mucho más amplio, ya que puede contener otros elementos aparte de clases, como son recursos, imágenes, etc.

Por otro lado, simplifican los tradicionales problemas de instalación y control de versiones sobre los programas, uno de los objetivos de la tecnología .NET, en la que en teoría, para instalar una aplicación, sólo sería necesario copiar los ficheros que la componen en un directorio de la máquina que la vaya a ejecutar.

6.4.25 La problemática tradicional de los componentes

De todos son conocidos los problemas que puede acarrear la instalación de una aplicación, en la que uno de sus elementos, sea un componente que sobrescribe otro ya existente de una versión anterior, pero que en su interior no guarda compatibilidad con ciertos aspectos de versiones anteriores, provocando el que otras aplicaciones que también hacen uso de ese componente, fallen.

Este inconveniente ha sido solucionado en parte por Windows 2000, ya que permite el desarrollo de programas, cuyos componentes puedan ser instalados en el mismo directorio del programa, de forma que al ejecutarse, la aplicación busque inicialmente en su directorio los componentes necesarios, y en caso de no encontrarlos, se dirija a las rutas habituales del sistema. Adicionalmente, los componentes propios del sistema operativo, permanecen bloqueados para evitar ser reemplazados accidentalmente por la instalación de terceras aplicaciones.

A pesar de que los anteriores aspectos constituyen un importante avance, no han resuelto del todo el problema, fundamentalmente, porque las cuestiones que tienen que ver con las versiones de componentes y sus correspondientes reglas, residen en lugares distintos del propio componente:
librerías de tipos, el registro del sistema, etc.

6.4.26 Ensamblados, una respuesta a los actuales conflictos

Para solucionar las cuestiones planteadas en la anterior sección, se han desarrollado los ensamblados. Un programador debe indicar en el interior del ensamblado, mediante metadatos que se almacenan en el manifiesto del ensamblado, toda la información acerca de la versión a la que pertenece. El ensamblado, por su parte dispone de la maquinaria que supervisa el cumplimiento de las normas de versión; y gracias a su diseño, es posible ejecutar varias versiones del mismo ensamblado simultáneamente sin provocar errores en el sistema, esta es una de las grandes innovaciones que introducen.

6.5 Visual Studio .NET

Una de las piezas más importantes en el esquema de Microsoft .NET es el propio Visual Studio .NET. Éste es un entorno de trabajo tan distinto a lo hecho anteriormente que Microsoft tuvo que darle otro nombre y prácticamente salirse de la secuencia de versiones de la plataforma de Visual Studio, aunque, al final, también lo mencionó como la séptima versión de la saga.

Para comprender a Visual Studio .NET hay que pensar como desarrollador y hacer una ligera historia de la tecnología de programación. Los primeros días de la programación se distinguieron por el código lineal, poco útil en aplicaciones grandes dado que arrojaba engendros difíciles de mantener. El código lineal evolucionó al código estructurado, que mejoró la disposición del código y que, por mucho tiempo, se estableció como el rey de la codificación.

Sin embargo, ya desde 1969, se avizoró otro esquema de programación que lograría una revolución real hasta mediados de la década de los años ochenta, con la presentación de C++, y más específicamente en la de los noventa con la proliferación de los entornos gráficos que le dieron mayor razón de ser.

La máxima de la programación orientada a objetos es la reutilización; es decir, aprovechar el código ya hecho para evitar la duplicación de trabajo. Sin embargo, una de las mayores limitantes de esto es, precisamente, la diversidad de lenguajes de programación: Un objeto generado en C++ no puede usarse directamente en lenguajes como Visual Basic, Object Pascal o SmallTalk. En realidad, un objeto generado en C++ sólo podrá usarse en C++, así como uno de Visual Basic, sólo podrá usarse en Visual Basic (claro está que podría compilar el objeto y hacerlo binario para, así, poder integrarlo en un entorno de programación, pero ello traería la necesidad de usar protocolos propietarios como COM, DCOM, COM+, CORBA, SOM, etcétera, lo cual rompe con ese sueño del reuso integral).

Con esto en mente, Microsoft quiso aprovechar la oportunidad de la presentación de Microsoft .NET para generar un marco de trabajo que fuera aprovechado por cualquier lenguaje de programación que se ciñera a sus estándares. Ese marco de trabajo es el .NET Framework, que es el meollo de la tecnología .NET. En pocas palabras, el .NET Framework es un cúmulo de clases expuestas para que, quien quiera, haga uso de su funcionalidad. A su vez, este cúmulo de clases conforma un estándar abierto que puede integrarse a cualquier plataforma o lenguaje.

Esta apertura permitió el diseño de un entorno de desarrollo tan amplio como lo es el Visual Studio .NET, que no sólo incluye los lenguajes de C# .NET, Visual Basic .NET, Visual C++ .NET y, próximamente, J# .NET, sino que hay más de 20 lenguajes de otros fabricantes que pueden funcionar en él, como Pascal .NET, Cobol .NET, y muchos otros.

El que tantos lenguajes distintos puedan funcionar en un mismo entorno, tiene un beneficio adicional: puede incluirse un objeto hecho en cualquiera de estos lenguajes en un proyecto generado en otro lenguaje. Por ejemplo, pueden incluirse clases generadas con C# .NET en un proyecto de Visual Basic .NET. Las clases de C# .NET no tendrán que compilarse para que esto sea posible, dado que el entorno

interpretará adecuadamente las instrucciones que tenga para poder aprovechar su funcionalidad sin problemas.

De esta forma, Visual Studio .NET es lo más cercano a la máxima de la programación orientada a objetos: la reutilización. Claro está que todos los lenguajes tendrán que cumplir con las prerrogativas del .NET Framework, pero con la cantidad de lenguajes que están apareciendo para este entorno, esto podría ya no ser un problema.

Conclusión:

Visual Studio .NET es la herramienta integral de desarrollo para crear e integrar rápidamente servicios Web XML y aplicaciones. Visual Studio .NET le ofrece un ambiente altamente productivo para desarrollar una amplia variedad de aplicaciones que se ejecutan en la nueva plataforma Microsoft .NET. Usando el ambiente de ejecución Microsoft .NET Framework, seguro y de alto desempeño, Visual Studio .NET le proporciona herramientas poderosas para diseñar, crear, probar e implementar servicios Web XML y aplicaciones así como compartir mejores prácticas y guías a su equipo de desarrollo.

6.5.1. Instalación del Visual Studio .NET

Preparación del entorno de trabajo
Antes de poder comenzar a escribir aplicaciones para .NET Framework, debemos instalar en nuestra máquina de trabajo las herramientas que nos permitirán el desarrollo de programas para este entorno de ejecución.

.NET Framework SDK
Se trata del kit de desarrollo de software para .NET Framework (Software Development Kit o SDK), que contiene la propia plataforma .NET y un conjunto de herramientas independientes, algunas funcionan en modo comando (en una ventana MS-DOS) y otras en modo gráfico. Los elementos imprescindibles para poder desarrollar aplicaciones para .NET están contenidos en este conjunto de herramientas.

Visual Studio .NET
Es la nueva versión de la familia de herramientas de desarrollo de software de Microsoft, naturalmente orientadas hacia su nuevo entorno de programación: .NET Framework.

Si bien es posible la escritura de programas empleando sólo el SDK de .NET Framework, este último, al estar compuesto de herramientas independientes, constituye un medio más incómodo de trabajo.

Visual Studio .NET (VS.NET a partir de ahora), al tratarse de un entorno de desarrollo integrado (Integrated Development Environment o IDE como también lo denominaremos a lo largo del texto), aúna todas las herramientas del SDK: compiladores, editores, ayuda, etc., facilitando en gran medida la creación de programas

Requisitos hardware
Las características mínimas y recomendadas que debe tener el equipo en el que instalemos VS.NET.
Mínimo Recomendado
Procesador Pentium II – 450 MHz Pentium III – 733 MHz
Memoria 128 MB 256 MB
Espacio en disco duro 3 GB

Sistema operativo
VS.NET puede ser instalado en un equipo con uno los siguientes sistemas operativos:
Windows 2000 (se requiere tener instalado el Service Pack 2).
Windows NT 4.0. (se requiere tener instalado el Service Pack 5).
Windows Me.
Windows 98
Para aprovechar todo el potencial de desarrollo de la plataforma, es recomendable usar como sistema operativo Windows 2000, ya que ciertos aspectos del entorno (las características avanzadas de gestión gráfica por ejemplo) no están disponibles si instalamos .NET en otro sistema con menos prestaciones.

6.6 .NET Compact Framework

6.6.1 Introducción

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.

Tiempo de ejecución en lenguaje común

El tiempo de ejecución en lenguaje común proporciona un entorno de ejecución de código que administra el código destinado a .NET Compact Framework. La administración de código se refiere a la administración de la memoria, de los subprocesos y de la seguridad, así como a la verificación y compilación del código en otros servicios del sistema.

El tiempo de ejecución se ha diseñado para mejorar el rendimiento. Utiliza compilación directa (JIT), que permite que el código administrado se ejecute en el lenguaje del equipo nativo de la plataforma en

el que se está ejecutando la aplicación. De esta manera es posible crear aplicaciones destinadas a una gran variedad de plataformas, sin que haya que preocuparse de volver a compilar o generar ejecutables destinados a cada plataforma concreta.

Aunque la aplicación móvil esté escrita en Visual Basic .NET o C# .NET, al tratarse de código administrado, seguirá siendo posible incorporar funciones y subrutinas almacenadas externamente en bibliotecas de vínculos dinámicos (DLL), incluidas las API de Windows CE. .NET Compact Framework proporciona los tipos de datos y la compatibilidad con las estructuras necesaria para incorporar con facilidad funciones de las API de Windows CE en su aplicación.
Biblioteca de clases de .NET Compact Framework

La biblioteca de clases de .NET Compact Framework es una colección de clases reutilizables que se integra estrechamente con el tiempo de ejecución en lenguaje común. Las aplicaciones aprovechan estas bibliotecas para obtener ciertas funcionalidades.

Como es de esperar en una biblioteca de clases orientada a objetos, los tipos de .NET Compact Framework permiten llevar a cabo una serie de tareas de programación habituales, entre las que se incluye el diseño de interfaces, el uso de XML, el acceso a bases de datos, la administración de subprocesos y la E/S de archivos.

A continuación, se incluye una lista de funcionalidades habituales disponibles en .NET Compact Framework.

6.6.2 Clases relacionadas con formularios

.NET Compact Framework implementa un subconjunto de las clases **System.Windows.Forms** y **System.Drawing**, que permite crear una cómoda interfaz de usuario basada en Windows CE para la aplicación del dispositivo. El diseñador de formularios de Visual Studio.NET se ocupa automáticamente de gran parte de la interacción utilizando estas clases.

La implementación de Windows Forms con .NET Compact Framework incluye la posibilidad de utilizar formularios, la mayoría de los controles de .NET Framework, el alojamiento de controles de otros fabricantes, mapas de bits y menús. La tabla 1 indica los controles que se incluyen con .NET Compact Framework.

Figura 6.9. Controles incluidos con .NET Compact Framework

Control	Descripción
Button	botón de comando simple
CheckBox	casilla de verificación habitual
ComboBox	lista desplegable de elementos
ContextMenu	implementa un menú contextual
DataGrid	cuadrícula que se puede enlazar a un origen de datos
DomainUpDown	lista de elementos que se puede explorar mediante una barra de desplazamiento
HScrollBar	barra de desplazamiento horizontal
ImageList	contenedor que almacena imágenes
InputPanel	controla el Soft Input Panel (SIP)
Label	control simple para mostrar texto
ListBox	proporciona una lista de elementos
ListView	proporciona cuatro vistas de los datos: iconos grandes, iconos pequeños,

	lista y detalles
MainMenu	implementa un menú en un formulario
NumericUpDown	campo de entrada numérica que incluye una barra de desplazamiento
OpenFileDialog	proporciona acceso al cuadro de diálogo nativo para abrir archivo
Panel	contenedor utilizado para acoger otros controles
PictureBox	muestra imágenes
ProgressBar	indicador visual del progreso de una tarea
RadioButton	botón de opción habitual
SaveFileDialog	proporciona acceso al cuadro de diálogo nativo para guardar archivo
StatusBar	panel simple para mostrar texto
TabControl	proporciona una interfaz de fichas para una aplicación
TextBox	campo de entrada de texto estándar
Timer	componente de temporizador básico
ToolBar	implementa una barra de herramientas en un formulario
TrackBar	interfaz de control deslizante utilizado con datos numéricos
TreeView	presenta datos en formato jerárquico
VScrollBar	barra de desplazamiento vertical

Al ser .NET Compact Framework un subconjunto de .NET Framework, los controles incluidos ofrecen un subconjunto de las funcionalidades de sus equivalentes para equipos de escritorio. Debido a consideraciones de tamaño y de rendimiento, los controles de .NET Compact Framework no incluyen algunas propiedades, métodos y eventos de los controles. Con un poco de codificación, usted mismo puede implementar estas funcionalidades en caso de que sean necesarias, ya que .NET Compact Framework le permite crear sus propios controles mediante herencia de la clase base del control. A partir de esta base, puede agregar sus propios métodos, propiedades y eventos para crear exactamente el control que necesita.

6.6.3 Clases de XML y de datos

.NET Compact Framework incluye un conjunto de clases que permiten incorporar con facilidad datos (ya sea de un origen de datos relacional o no), entre los que se incluye el contenido XML, en sus aplicaciones móviles. Estas clases se definen para los espacios de nombres **System.Data** y **System.Xml**. La implementación de clases de XML y de datos en .NET Compact Framework es un subconjunto de la que se encuentra en .NET Framework.

6.6.4 Servicios Web

.NET Framework presta una atención especial a los servicios Web. En el espacio de nombres **System.Web** de .NET Compact Framework, hay una versión a escala reducida de las posibilidades y funcionalidades que ofrece el correspondiente espacio de nombres de .NET Framework. La característica más destacable es que se pueden crear clientes de servicios Web pero no se pueden alojar servicios Web en .NET Compact Framework.

Estos clientes de servicios Web XML pueden ser sincrónicos o asincrónicos. Se pueden crear fácilmente clientes de servicios Web XML destinados a .NET Compact Framework. El IDE de Visual Studio .NET hace automáticamente la mayor parte del trabajo.

6.6.5 Compatibilidad con GDI

.NET Compact Framework proporciona compatibilidad con los elementos básicos de dibujo de GDI incluidos mapas de bits, pinceles, fuentes, iconos y plumas mediante el espacio de nombres **System.Drawing**.

6.6.6 Clases base

.NET Compact Framework proporciona un conjunto robusto de clases base que expone una amplia variedad de funcionalidades que los desarrolladores pueden aprovechar. Esta infraestructura subyacente le permite escribir cómodas aplicaciones .NET que incluyen la posibilidad de crear aplicaciones con varios subprocesos (**System.Threading**), aprovechar recursos de red (**System.Net**) y trabajar con archivos (**System.IO**).

6.6.7 Compatibilidad con IrDA

Algunos dispositivos con plicac CE, por ejemplo, Pocket PC y Pocket PC 2002, incluyen la posibilidad de comunicarse mediante infrarrojos (IR). Para poder aprovechar esta posibilidad, .NET Compact Framework incluye clases que le permiten aprovechar la aplicación mediante infrarrojos desde la aplicación. Estas clases son parte del espacio de nombres **System.Net.IrDA**. Puede utilizar infrarrojos para comunicarse con equipos Pocket PC, impresoras y otros dispositivos compatibles con infrarrojos.

6.6.8 Compatibilidad con Bluetooth

.NET Compact Framework no incorpora compatibilidad nativa con Bluetooth. Puede obtener acceso a la mayoría de implementaciones de Bluetooth en equipos Pocket PC de otros fabricantes mediante las comunicaciones del puerto serie o mediante una API del proveedor.

6.6.9 Compatibilidad con Visual Basic

Visual Basic .NET utiliza bastante las funciones auxiliares de la biblioteca auxiliar de Visual Basic. .NET Compact Framework incluye también un subconjunto de estas funciones, consideradas por los desarrolladores de Visual Basic como una parte esencial del lenguaje, por lo que se ha decidido incluirlas.

Para los desarrolladores de Visual Basic o eMbedded Visual Basic que están comenzando a trabajar con .NET Compact Framework, esto significa que la mayoría de las funciones del lenguaje Visual Basic con las que está acostumbrado a trabajar también estarán disponibles en Visual Basic .NET.

6.6.10 Características a la carta

Para ahorrar recursos en el dispositivo de destino, Microsoft ha dividido .NET Compact Framework en componentes lógicos. Al suministrar los componentes como DLL independientes (o ensamblados, según se les denomina en .NET Compact Framework) Microsoft le ofrece la oportunidad de decidir y seleccionar las características que necesita y sólo aquéllas para las que el dispositivo de destino tiene espacio.

Un ejemplo de esto es el ensamblado System.SR, que contiene las cadenas de los mensajes de error. Si incluye este ensamblado en su aplicación podrá ver descripciones detalladas de todos los errores detectados, lo cual es ciertamente útil durante una sesión de depuración, pero normalmente no es necesario una vez que se ha lanzado a producción. Si no se incluye este ensamblado, no se verán afectados ni el rendimiento ni la funcionalidad de la aplicación, pero no podrá ver mensajes de error detallados.

Otro ejemplo del enfoque a la carta de .NET Compact Framework son los componentes SQL Server CE, que se entregan en un conjunto de DLL cuyo tamaño total apenas supera 1 MB. A menos que agregue explícitamente una referencia a los ensamblados System.Data.SqlServerCe, estas DLL no se incluirán en la aplicación.

6.7 Características no incluidas en .NET Compact Framework

Ha sido necesario realizar importantes recortes en .NET Framework para adaptarlo a las limitaciones de funcionamiento de Windows CE. En esta sección se abordarán las características más destacables de .NET Framework que no se han incluido en .NET Compact Framework.

6.7.1 Sobrecargas de métodos

La sobrecarga de un método ofrece manera alternativas de llamar a dicho método. También aumenta el tamaño de Framework. Por este motivo, .NET Compact Framework ha eliminado las sobrecargas de prácticamente todos los métodos.

Hay dos consecuencias básicas de esto. Primero, es bastante probable que una determinada sobrecarga de un método que solía utilizar con una aplicación de escritorio no esté disponible a la hora de desarrollar aplicaciones basadas en .NET Compact Framework. Segundo, cuando esté leyendo la documentación, preste mucha atención a si se admite un determinado método en .NET Compact Framework.

6.7.2 Controles no incluidos

Hay una serie de controles de .NET Framework que no se han incluido en .NET Compact Framework. La ausencia de la mayoría de estos controles carece de importancia para los desarrolladores de aplicaciones móviles. Teniendo en cuenta que las aplicaciones móviles apenas imprimen, no es ningún problema eliminar la familia completa de controles relacionados con la impresión. Por tanto, se han eliminado los controles CrystalReportViewer, PageSetupDialog, PrintDialog, PrintDocument, PrintPreviewControl y PrintPreviewDialog. Puede sustituir la mayoría de los cuadros de diálogo que faltan por sus propios cuadros de diálogo o bien utilizar directamente los cuadros de diálogo del sistema mediante la API de Windows CE.

También están empezando a aparecer controles de otros fabricantes que sustituyen a los controles que no se han incluido en .NET Compact Framework. Si desea ver una lista de controles de .NET Compact Framework de otros fabricantes, consulte las referencias que se incluyen al final de este artículo.

6.7.3 Funcionalidad XML

Aunque se ha incluido buena parte de la funcionalidad de XML de .NET Compact Framework, ha sido necesario recortarla. El componente clave relacionado con el XML que no se ha incluido es el espacio

de nombres **System.Xml.XPath**. El espacio de nombres XPath hacía la interpretación del XML mucho más fácil que los métodos que incorpora .NET Compact Framework. Para compensar esto, puede utilizar una combinación de las búsquedas recursivas e iterativas con el modelo de objetos de documento (DOM, Document Object Model).

Hay otro componente clave XML que no se ha incluido en .NET Compact Framework: XSLT (Extensible Stylesheet Language Transformation). Mediante XSLT, es posible convertir un documento XML en diferentes formatos.

En relación con el XML, .NET Compact Framework no ofrece actualmente compatibilidad para el desarrollo de servicios Web XML basados en dispositivos.

6.7.4 Compatibilidad con bases de datos

.NET Compact Framework ofrece un conjunto robusto de herramientas relacionadas con los datos. La compatibilidad con la base de datos local la proporciona SQL Server CE. En el servidor, .NET Compact Framework ofrece compatibilidad con SQL Server.

6.7.5 Serialización binaria.

Debido a consideraciones de tamaño y de rendimiento, no se han incluido las clases BinaryFormatter y SoapFormatter en .NET Compact Framework.

6.7.6 Acceso al registro de Windows

.NET Framework incorpora el espacio de nombres Microsoft.Win32.Registry, que facilita el trabajo con el registro de Windows desde una aplicación. Obviamente, este espacio de nombres no se ha incluido en .NET Compact Framework, ya que hace referencia a Win32 y no a Windows CE. Puede tener acceso al registro de Windows CE llamando a las correspondientes API de Windows.

6.7.7 Aprovechamiento de los componentes COM

El proceso de incorporación de objetos COM en una aplicación basada en .NET Compact Framework consta de dos pasos. Primero, debe escribir un empaquetador DLL no administrado (es decir, en eMbedded Visual C++) que exponga el objeto COM. Dependiendo de la complejidad del objeto COM, hacer esto puede ser muy sencillo o extremadamente complicado. Segundo, debe utilizar PInvoke para tener acceso al empaquetador DLL. Afortunadamente, la comunidad de desarrolladores ya ha comenzado a trabajar en proporcionar acceso a los componentes COM utilizados con más frecuencia, varios de los cuales se incluyen en las referencias que aparecen al final del artículo.

6.7.8 Seguridad

.NET Compact Framework no impide el acceso al código no administrado. Cualquier aplicación puede llamar a una API, ya sea del sistema o no.

Actualmente, no hay ningún tipo de seguridad basada en funciones en .NET Compact Framework. El objeto principal no entiende de identidad conocida o de función conocida.

6.7.9 Servicios Web XML

La exclusión más notable de las posibilidades de los servicios Web XML de .NET Compact Framework es la posibilidad de utilizar cookies. Las cookies se utilizan con frecuencia para almacenar el estado en el servidor entre diferentes llamadas desde un cliente. Aunque el uso de las cookies en los servicios Web no es tan frecuente como su uso en sitios Web, también se utilizan.

.NET Compact Framework ofrece posibilidades de cifrado limitadas respecto a los servicios Web.

6.7.10 Impresión

.NET Compact Framework no ofrece ninguna función para imprimir. No hay ninguna manera fácil de interaccionar con impresoras de red ni con impresoras externas mediante infrarrojos.
La solución para tener acceso a las impresoras de red es crear una aplicación basada en el servidor, que acepte e imprima las tareas enviadas por la aplicación móvil.

Puede enviar una salida mediante el puerto de infrarrojos directamente a impresoras compatibles con infrarrojos. Puede utilizar el espacio de nombres **System.Net.IrDA** para tener acceso al puerto de infrarrojos del dispositivo.

6.7.11 GDI+

Windows CE no es compatible de manera nativa con GDI+, por lo que las funcionalidades relacionadas con GDI+ se han eliminado de .NET Compact Framework.

6.7.12 Entorno remoto

La primera versión de .NET Compact Framework no es compatible con el entorno remoto.

6.8 Visual Studio .NET 2003 para dispositivos móviles

Visual Studio .NET 2003 ofrece un entorno de desarrollo robusto para la creación de aplicaciones destinadas a .NET Compact Framework. Junto con Visual Studio .NET se incluyen un conjunto de perfiles de dispositivos ya creados. Un perfil de dispositivo contiene la información necesaria para crear aplicaciones destinadas a determinados dispositivos. Con Visual Studio .NET, hay perfiles que le permiten crear aplicaciones para Pocket PC, Pocket PC 2002 y Windows CE .NET 4.1 y posterior. Estos perfiles le permiten crear aplicaciones que incluyen Windows Forms y ADO.NET, y ofrecen la posibilidad de consumir servicios Web.

Los perfiles pueden ser específicos de los dispositivos, como los destinados a Pocket PC, plataformas menos específicas destinadas a la plataforma Windows CE en general o perfiles genéricos destinados a cualquier plataforma a la que se haya transferido .NET Compact Framework.

Visual Studio .NET es compatible con los kits de dispositivos (anteriormente conocidos como SDK). Al igual que las versiones anteriores de las herramientas incrustadas, los kits de dispositivos están separados de Visual Studio .NET y se pueden instalar y actualizar de manera independiente.

6.8.1 Adiciones al IDE (Entorno de Desarrollo)

Además de todas las características que se encuentran de manera nativa en Visual Studio .NET, están las siguientes características específicas de los dispositivos:

- Plantillas: configuraciones predefinidas para tipos de proyectos habituales. Las plantillas se proporcionan para los dispositivos Pocket PC y Windows CE.
- Controles específicos de los dispositivos: controles específicamente diseñados para utilizarlos con Pocket PC y Windows CE. La interfaz, el consumo de recursos y la funcionalidad se han adaptado a estos entornos.
- Emuladores de dispositivos: entornos de prueba que simulan determinados dispositivos. Los emuladores se ejecutan en el equipo del desarrollador, lo que permite realizar pruebas sin utilizar directamente el dispositivo.
- Distribución automática de aplicaciones: permite realizar pruebas fácilmente en un emulador o un dispositivo, ofreciendo a los desarrolladores un entorno de pruebas perfectamente integrado.
- Depuración remota: permite aprovechar las herramientas de depuración que ofrece el IDE de Visual Studio .NET con las aplicaciones del dispositivo. Todas las herramientas de depuración se pueden utilizar con las aplicaciones basadas en .NET Compact Framework que se ejecuten en un emulador o en un dispositivo.

6.8.2 Lenguajes admitidos

.NET Compact Framework admite dos lenguajes de desarrollo, C# .NET y Visual Basic .NET. Mientras que las versiones anteriores de las herramientas de desarrollo de Windows CE favorecían el uso de lenguajes basados en C (concretamente, eMbedded Visual C++) con .NET Compact Framework apenas importa el lenguaje que se utilice, ya que todos son igualmente eficaces y funcionales.

Al ser una adición posterior al entorno de desarrollo Visual Studio .NET, .NET Compact Framework no es compatible con J#.

También debe tener en cuenta que hay otra limitación de lenguaje en .NET Compact Framework que no existe en .NET Framework. Con .NET Framework puede utilizar componentes en diferentes lenguajes en un único proyecto. En comparación, los proyectos de .NET Compact Framework están restringidos a un único lenguaje, ya sea C# .NET o Visual Basic .NET. La solución que permite superar esta limitación de utilizar un único lenguaje en cada proyecto impuesta por .NET Compact Framework es crear proyectos adicionales utilizando la plantilla Class. Agregue el código del lenguaje alternativo a la plantilla y, a continuación, sólo tiene que agregar referencias a estas clases en el proyecto de la aplicación.

6.9 Visual C# .NET 2003

Visual C# .NET 2003 es una herramienta y un lenguaje de programación modernos e innovadores que permiten generar software conectado a .NET para Microsoft Windows®, Web y una amplia gama de servicios. Debido a su sintaxis familiar, similar a la de C++, a su entorno de desarrollo integrado (IDE) de gran flexibilidad y a su capacidad para crear soluciones para una gran variedad de plataformas y dispositivos, Visual C# .NET 2003 facilita enormemente el desarrollo de software conectado a .NET.

Visual C# .NET está basado directamente en C++. Es un lenguaje de programación moderno e intuitivo orientado a objetos que ofrece mejoras significativas, incluido un sistema de tipos unificados, código

"no seguro" que permite un control total al programador y nuevas construcciones de lenguaje muy eficaces y fáciles de entender para la mayoría de los programadores.

Es un innovador lenguaje orientado a componentes con compatibilidad inherente con propiedades, indizadores, delegados, control de versiones, sobrecarga de operadores y atributos personalizados. Mediante comentarios en formato XML, los programadores de C# pueden elaborar documentación muy útil sobre el código fuente. Un modelo de herencia avanzado permite a los programadores volver a utilizar el código de cualquier lenguaje de programación compatible con .NET.

Los programadores de C# pueden incorporarse a la comunidad de desarrollo más moderna y de más rápido crecimiento, en la que pueden intercambiar código y recursos, aprovechar sus conocimientos en múltiples entornos de programación y contribuir al proceso de normalización que garantiza una participación activa y entusiasta en dicha comunidad.

La calidad del entorno de desarrollo integrado (IDE) de Visual C# .NET proporciona a los usuarios el entorno de desarrollo más completo. Ofrece valiosos recursos en línea a la comunidad de programadores. La página de inicio ofrece a los programadores un portal para lograr acceso con un solo clic a actualizaciones, preferencias, información sobre proyectos utilizados recientemente y a la comunidad MSDN en línea. Sus funciones mejoradas, como la tecnología IntelliSense, el cuadro de herramientas y la lista de tareas, proporcionan mejoras significativas de la productividad. Las ventanas de ocultación automática y la compatibilidad con varios monitores ayudan a los programadores a aprovechar al máximo el espacio de pantalla y a personalizar su entorno de desarrollo. Las nuevas reglas de generación personalizadas facilitan más que nunca el desarrollo de software eficaz y seguro.

Mediante el Diseñador de Web Forms y el Diseñador XML, los programadores pueden utilizar las características y la funcionalidad de finalización de etiquetas de IntelliSense; o bien, elegir el editor WYSIWYG (lo que se ve es lo que se imprime) para poder crear aplicaciones Web interactivas arrastrando y colocando elementos. Siguiendo unos pocos pasos sencillos, los programadores pueden diseñar, desarrollar, depurar e implementar eficaces servicios Web XML que disminuyen el tiempo de desarrollo mediante la integración de procesos de negocio accesibles desde cualquier plataforma.

6.9.1 .NET para Web

Con Visual C# .NET 2003, los programadores pueden sacar partido de Microsoft .NET e incorporar tecnología de próxima generación para la administración de recursos, tipos unificados y acceso remoto. Microsoft .NET, permite a los programadores obtener tecnología de administración de memoria de gran calidad a fin de recolectar sin problemas los elementos no utilizados y reducir la complejidad del programa. Los programadores pueden emplear el sistema de tipos comunes de Microsoft Windows .NET Framework para reciclar el código escrito en cualquiera de los más de 20 lenguajes compatibles con .NET y también realizar llamadas a procedimientos remotos con gran eficacia.

Además, los programadores pueden utilizar la biblioteca de clases de Windows .NET Framework, de total garantía y seguridad, para agregar funcionalidad integrada de gran capacidad, incluido un amplio conjunto de clases de colección, compatibilidad con redes y subprocesamiento múltiple, clases de expresión regular y cadena, así como una amplia compatibilidad con XML, esquemas XML, espacios de nombres XML, XSLT, XPath y SOAP. Asimismo, mediante el Asistente para conversión del lenguaje Java (JLCA), los programadores pueden migrar los proyectos basados en Java al entorno Microsoft .NET.

Con Visual C# .NET 2003, los programadores pueden crear eficaces servicios Web XML que integren procesos empresariales y los pongan a disposición de aplicaciones que se ejecuten en cualquier

plataforma. Los programadores pueden incorporar fácilmente cualquier número de servicios Web XML que estén catalogados y disponibles en cualquier directorio UDDI (integración, descubrimiento y descripción universal) independiente, proporcionando una base sólida de servicios y lógica empresarial para sus aplicaciones.

Visual C# .NET 2003 también permite a los programadores crear aplicaciones de próxima generación basadas en Windows. Con la herencia visual, los programadores pueden simplificar enormemente la creación de aplicaciones basadas en Windows, centralizando en formularios primarios la lógica común y la interfaz de usuario para toda la solución. Utilizando delimitación y acoplamiento de controles, los programadores pueden generar automáticamente formularios cuyo tamaño puede cambiarse, mientras el editor de menús in situ permite crear menús de manera visual directamente desde el Diseñador de Windows Forms.

La compatibilidad nativa con .NET Compact Framework, dispositivos Web móviles y aplicaciones incrustadas disponible en Visual Studio .NET 2003 Professional Edition permite a los programadores de C# trabajar para una extensa variedad de dispositivos móviles, incluidos Pocket PC, teléfonos móviles y dispositivos que utilicen el sistema operativo Windows CE .NET. Los programadores pueden alcanzar una productividad inmediata, ya que utilizarán las mismas herramientas y el mismo modelo de programación para crear versátil software basado en dispositivos que los usados para crear soluciones eficaces basadas en Windows y Web.

6.10 SQL Server 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.

SQL Server CE expone un sistema esencial de características de la base de datos relacionales, así como un procesador QUERY y una ayuda para las transacciones y los tipos de datos clasificados, mientras que mantiene una forma compacta que preserve recursos del sistema. Los datos remotos tienen acceso y la réplica de la fusión se asegura de que los datos de bases de datos del SQL Server estén entregados confiablemente, se puede manipular offline, y se puede sincronizar más adelante al servidor, haciendo el SQL Server CE ideal para los ambientes móviles y wireless.

El SQL Server CE esta diseñado para integrarse con el .NET Compact Framework y Visual Studio .Net lo que no impide que se integre con aplicaciones con las Embedded Visual Tools .

El SQL Server CE amplía la frontera de la administración de datos entregando:

- **Una plataforma familiar de la base de datos para el rápido desarrollo.** La familia de SQL Server proporciona la ayuda de administración de datos y la programabilidad a través de la empresa de los servidores más grandes directo a las estaciones de trabajo. El SQL Server CE proporciona capacidades robustas de la administración de datos en los dispositivos móviles. Exponiendo una programación y un modelo operacional constantes con el resto de la familia del SQL Server, el SQL Server se asegura de que las organizaciones pueden integrar fácilmente con los sistemas existentes y aprovecharse de capacidades existentes de desarrollo.
- **Un acuerdo con todo base de datos relacional capaz.** Aunque los dispositivos están avanzando rápidamente, los recursos de sistema tales como memoria disponible son a menudo escasos, así que es crítico que un sistema de la base de datos emparentada sea tan compacto

como sea posible mientras que todavía expone funcionalidad esencial. El SQL Server CE tiene un uso pequeña de la memoria, entregando toda su funcionalidad en aproximadamente 1 megabyte (MB). El funcionamiento se realza con un procesador QUERY óptimo. Una gama de los tipos de datos se apoya para asegurar flexibilidad, y el cifrado 128-bit que proporciona en el dispositivo para la seguridad de archivo de base de datos.

- **Acceso flexible de los datos.** El SQL Server CE permite el acceso directo, eficiente a los datos de la empresa si un dispositivo está conectado siempre o conectado intermitentemente con la computadora que funciona el SQL Server. El acceso remoto de los datos expuestos en SQL Server 6.5, el SQL Server 7.0, y bases de datos del SQL Server 2000 a través de ejecuciones remotas de las declaraciones Transact-SQL y de la capacidad de llevar registros al dispositivo del cliente para ponerse al día. Cuando está utilizado con el SQL 2000 Server, el SQL Server CE proporciona las capacidades extendidas para la sincronización a través de la réplica de la combinación. Ambas tecnologías del acceso de los datos se aprovechan bajo estándares del Internet, incluyendo el HTTP Secure Socket Layer (SSL), con la integración con el Internet Information Server (IIS). Este acercamiento asegura que los datos se pueden alcanzar confiablemente y flexiblemente, incluso a través de firewalls.

El SQL Server CE incluye actualizaciones:

- **Query Analyzer** El query analizer ha sido actualizado para mejorar el interfaz del usuario para una administración más fácil de los objetos.
- **Query Processor** El query processor le permite agilizar su código con nuevas funciones intrínsecas y consultas parametrizadas.
- **Motor de almacenamiento** El motor de almacenamiento incluye mejoras importantes a la funcionalidad del acceso remoto de los datos, incluyendo index pulls y upload only features-solamente. Además, el motor de almacenamiento ahora soporta el mismo número de índices (249) por tabla como el SQL Server.
- **Asistentes de Conectividad** Esta guía paso a paso puede ayudarle rápidamente a instalar sus opciones de seguridad y de conectividad para la réplica de los datos.
- **ISQLW.** Esta herramienta de consulta en el dispositivo le permite almacenar consultas y ver el esquema en el dispositivo.

Parte III Desarrollo de la Aplicación

Capítulo 7. ESTUDIO PREVIO

7.1 Decisiones iniciales

El objetivo de este proyecto no es otro que la realización de una aplicación capaz de permitir a las personas afectadas con parálisis cerebral, la posibilidad de comunicarse a través de un dispositivo móvil como puede ser un pocket PC, de manera sencilla a través de pulsaciones.

Debe ser una herramienta práctica y fácil de manejar, que podrán usar todas aquellas personas que dispongan de la capacidad de la escritura, puesto que la aplicación se basa en el alfabeto español.

La herramienta está pensada para que estas personas sean capaces de comunicarse con el resto del entorno, sin necesidad de que ellos hayan adquirido a lo largo de su vida la capacidad de la lectura, ya que las frases que escribe el usuario pueden ser reproducidas.

7.2 Metodología empleada

El método a seguir deberá ser apropiado para la orientación a objetos puesto que el lenguaje escogido es Visual C# .NET.

Existen muchos métodos de desarrollo orientados a objetos en ingeniería del software, no obstante no se seguirá una técnica únicamente, sino que se han tomado algunas ideas generales de estos métodos. Las ideas principales son:

Se partirá de la realidad y se irá construyendo una serie de modelos cada vez más detallados hasta llegar a un modelo implementable o solución. Para ello se usa en este caso el Lenguaje unificado de modelado (UML) al ser el único lenguaje de modelado que puede considerarse estándar.

Se intentará que el método de trabajo sea continuo e **incremental**, procurando eliminar fronteras entre las diversas fases del desarrollo, realizando modificaciones progresivas sin grandes saltos. Esto permitirá reducir el tiempo de trabajo ya que no se realizarán estudios ni diagramas que no tengan aplicación final directa en el producto acabado, a diferencia de lo que ocurría con otros métodos no orientados a objetos. Todos los esfuerzos irán dirigidos hacia la implementación final.

El método no será solo incremental, si no que será una evolución natural en la que ciertas ideas tendrán que ser reconsideradas y muchos elementos serán modificados o eliminados a medida que avanzamos en el desarrollo, siguiendo un proceso **iterativo**.

Se buscará realizar un esfuerzo en la calidad de los componentes software de la solución final (de las clases). Así mismo se buscará que dichos componentes tengan gran independencia. De esta forma se apostará por la reutilización y por la facilidad de mantenimiento. Inicialmente esto no beneficia mucho a corto plazo puesto que requiere un mayor esfuerzo pero si que da unos buenos resultados a medio / largo plazo.

EL proceso con el que se corresponde lo descrito es con "**El Proceso Unificado"**, una metodología de desarrollo software dirigida por casos de uso. Se centra en la funcionalidad que el sistema debe poseer para satisfacer las necesidades de un usuario (persona, sistema externo, dispositivo) que interactúa con él, podríamos decir que los casos de uso son el hilo conductor que orienta las actividades de desarrollo

En nuestro caso, por las características del proyecto, una aplicación orientada a la comunicación de una persona con parálisis cerebral, tenemos que tener en cuenta de manera continua los requisitos del usuario, y orientar el diseño a sus especificaciones que pueden ir cambiando a medida que cada mini-proyecto se va poniendo en marcha (cada mini proyecto es considerado una iteración)

7.3 Métodos de escritura

Hemos desarrollado diversos métodos de escritura para esta aplicación. Nos hemos basado en modificaciones de métodos de escritura existentes para ordenador, métodos usados en móviles y también en alguna idea de teclados en pantalla para PDAs ya comercializados.

7.3.1 Características de los métodos de escritura de los dispositivos portátiles

- **Teléfonos Móviles**

Los métodos de escritura para móviles tienen en cuenta las características de las que disponen estos dispositivos, como pueden ser:

Tamaño: Obligatoriamente las teclas son diminutas por el tamaño del terminal. Es muy difícil introducir todas las letras, y usan las pulsaciones repetidas del mismo botón para conseguir escribir la letra adecuada.

Suelen llevar una docena de teclas como mucho.

Posición para escribir: Lo normal es sujetarlos con una mano, y usar un solo dedo para escribir, el pulgar. Aunque existe algún modelo con teclas en ambos lados del teclado, o con teclados QWERTY. Los métodos de escritura están basados en pulsaciones de una en una.

Tipo de usuario: Son personas sin conocimientos técnicos. Están pensados para un público masivo. Tampoco hay edad concreta para tener un teléfono.

Tienen que ser capaz de usarlo personas de todas las edades y conocimientos. Son métodos de encender y escribir, sin configuraciones ni aprendizaje. En algunos casos, para usar el texto predictivo se aconseja el manual de usuario, aunque lo normal es que la persona saque el móvil de la caja y se ponga a usarlo.

- ## PDAs o asistentes personales

Las características de escritura se basan en:

Posición: Las PDAs se suelen manejar con una sola mano, mientras que con la otra se sujetan. Tampoco se usan métodos de dos manos, como en un ordenador personal de sobremesa, menos para las handheld, que son agendas electrónicas con teclado tipo QWERTY incorporado, y que no tienen mayor interés para este proyecto.

Características del dispositivo: Tienen pantalla táctil, y para usarlo se dispone de un *stylus*, o estilete de punta no agresiva. Los botones hardware de los que disponen son muy poco numerosos, entre 4 y 8, y se usan para funciones de los programas, no para introducir texto.

Tipo de usuario: Hasta ahora son usuarios habituados a las últimas tecnologías, con capacidades para aprender métodos más complicados y/o que requieran aprendizaje.

Los métodos de escritura para PDAs son mayoritariamente de dos tipos:

Métodos de escritura de rasgos: Se usa el *stylus* para escribir a semejanza de cómo se escribiría en un papel. El mejor representante de este método son las PDAs con sistema operativo PALMOS con su método de escritura *calligrapher*. Es un método que requiere aprendizaje y cierta habilidad para conseguir resultados óptimos. Como ventaja se puede nombrar que es rápido una vez que se ha habituado a él, y que se parece a la escritura natural. Además se puede tomar notas con facilidad, y no hace falta tener la mirada fija en la pantalla. El usuario puede moverse mientras escribe. La precisión no es necesaria, sino la habilidad para hacer los rasgos reconocibles a la agenda. A pesar de que se escriba correctamente, puede fallar en reconocer los caracteres.

Por necesitar *stylus*, y una alta habilidad manual, queda descartado para este proyecto.

Métodos de escritura de emulación de botones:

Se basan en diversas formas de emular la presencia de botones en la pantalla de la PDA.

El más típico y habitual es el teclado QWERTY en pantalla. Por ser el habitual en los ordenadores personales (habituales entre los usuarios de estos tipos de dispositivos) no necesitan de ningún periodo de aprendizaje, el usuario va a poder escribir nada más encenderlo. Además siempre se sabe el símbolo que va a salir, la precisión entre lo que se marca y lo que se quiere tener es absoluta.

Desventajas: como la pantalla de la PDA es reducida, el usuario tiene que tener mucha precisión para acertar en el diminuto botón que representa una tecla. Además no tiene la guía del tacto que se tiene con un teclado físico, y tiene que estar mirando fijamente la pantalla para poder escribir. Por último, es difícil escribir en movimiento o haciendo otras tareas.

Existen otros métodos basados en botones más adaptados a las características de una pda, como el método combinado de botones y rasgos.

Podríamos resumirlo así:

	Rapidez	Habilidad usuario	Tasa de Acierto	Atención necesaria	Escritura en movimiento	Velocidad	Aprendizaje
Teléfonos	Baja	Baja	alta	media	Muy alta	Muy baja	Medio
Rasgos	Alta	Muy alta	baja	baja	Alta	Alta	mucho
Emulación teclado	alta	Alta	alta	Muy alta	Muy baja	Alta	Poco
Mini Teclado físico	Muy alta	Alta	Muy alta	alta	Muy baja	Alta	poco

Figura 7.1 Métodos de escritura actuales para dispositivos móviles

Para nuestro proyecto, teníamos unas premisas claras, que afectan al diseño del método de escritura:

- Sin *Stylus o puntero* Una persona con dificultades de movimiento normalmente no tiene capacidades para manejar con la precisión suficiente un objeto tan pequeño, y mucho menos usarlo en una pantalla pequeña, además de que necesitaría hacer otras acciones que pueden tener mucha dificultad. Algo tan simple como sacar el puntero de la PDA es el primer escollo con el que se tendrían que enfrentar. Así que todos los métodos están basados en el uso del dedo como mecanismo de comunicación con la aplicación.
- Capacidad de precisión reducida. Cada usuario con movilidad reducida tiene características especiales y posiblemente únicas. La habilidad del usuario tiene que ser tenida en cuenta. Hay algunos con una buena capacidad de acertar en puntos concretos, y otros que tienen movimientos muy erráticos.
- Velocidad: Hay usuarios con una velocidad de reacción reducida, es decir, sus movimientos que puede realizar son lentos, lo que hay que considerar.
- Pantalla reducida. Hay que aprovechar al máximo la pantalla. Por eso, en todos los métodos se usa toda su superficie. No sólo los botones realizan acciones, sino todos los elementos que estén en la pantalla reaccionan a los eventos de los usuarios. Por ejemplo, en los métodos 8 botones y en el método vocales, si se presiona la caja de texto donde se puede ver lo que el usuario ha introducido hasta ese momento, el sistema buscará palabras que coincidan con las últimas letras que haya introducido el usuario. En el método puro de bases de datos, para aceptar la palabra sugerida, en vez de usar otro botón aparte, se usa la lista de palabras sugeridas para aceptar la palabra seleccionada.

Por el mismo motivo, en la aplicación definitiva se han eliminado todos los elementos que no tienen un uso en la aplicación , como la barra de tareas o la barra de menú. También se han sacrificado todos los elementos decorativos como cuadros, o separadores. Los elementos no tienen separación entre unos y otros, ni existen márgenes con los bordes de la pantalla.

- Elementos usables con los dedos y visibles desde una distancia mayor a la habitual. La PDA será usada por los usuarios normalmente desde un soporte en la silla de ruedas, estará a una distancia mayor a la habitual, que suele ser un par de PalmOS de los ojos.

Por esto, no se usarán ListBox o ComboBox típicas de Windows, ni el diminuto botón de la barra de tareas para cerrar la aplicación, ni menús, ni el tipo de letra por defecto.

- Todo tiene que poder hacerlo el usuario. Excepto para la instalación del programa, no necesitará ningún tipo de ayuda externa. Sobre todo para encender y arrancar la aplicación. Asimismo, los botones hardware del dispositivo serán usados por la herramienta, ya que su uso normal, que es el de arrancar otros programas, no tiene utilidad, porque si necesita un teclado especial, no necesitará funciones como la lista de tareas o el Pocket Outlook.

7.4 Métodos de escritura aplicables a una PDA

Están ordenados por la necesidad de capacidades físicas del usuario para manejarlas, desde capacidades muy reducidas a capacidades físicas bastante desarrolladas.

Los tres primeros métodos se distinguen del resto por no necesitar ninguna precisión para usarse. En los siguientes se necesita que los usuarios tengan capacidad para presionar en partes concretas de la pantalla.

7.4.1 Método de barrido

DESCRIPCIÓN:

Este método usa toda la PDA para reconocer la pulsación del botón. Está indicado para personas con graves trastornos de movimiento.

Su funcionamiento es análogo a los pulsadores usados habitualmente en los centros especiales.

En la pantalla salen todas las letras y símbolos que el usuario puede escribir. A través de un cursor de color diferente se le indica al usuario la tecla que en ese momento está activa. Cuando el cursor se encuentra encima del carácter deseado, el usuario sólo tiene que apretar con cualquier parte del cuerpo la pantalla, y esa letra será seleccionada. A continuación se vuelve a hacer un barrido de todas las letras.

Figura 7.2 Método del barrido

Es con diferencia el método más lento, ya que, aunque es de orden 1 (es decir, cada letra necesita sólo una pulsación) el usuario tiene que estar la mayor parte del tiempo esperando a que pase la letra buscada.

El tiempo medio será aproximadamente: $\sum_{n=1}^{Número\_de\_letras} \mathrm{Re}tardo\_Barrido * n = 7{,}25$

De donde 7.25 sale de sumar el tiempo (medio segundo) que tardaría en salir cada una de las letras, dividido por el número total de letras, suponiendo que la frecuencia de cada una de las letras fuese la misma.

Una opción para optimizar el tiempo medio, sería poner las letras más usadas, como pueden ser las vocales, al principio. El tiempo se mejoraría a cambio de tener un teclado menos intuitivo.

Así que con las 28 letras y con un retardo de medio segundo, de media se puede escribir un carácter cada 14 segundos.

Se tardaría cerca de un minuto en escribir "HOLA".

La única habilidad necesaria del usuario para usar este método consiste en poder golpear los 12 centímetros cuadrados (aproximadamente) de la pantalla de la PDA con una parte de su cuerpo con la que tenga movilidad. Ni siquiera necesita usar el dedo, podría usar la cabeza.

La agenda electrónica puede ir deletreando las teclas, este método sería útil también para personas con problemas de visión o tetrapléjicos.

7.4.2 Método del barrido de sonido.

Aprovechando la capacidad de todas las PDAs de grabar sonidos de forma autónoma, se puede hacer una modificación del método de barrido.

Para personas con una inmovilidad total, la PDA podría ir mostrando en la pantalla a pantalla completa las letras del abecedario, y con un sonido proveniente del usuario, seleccionaría la letra deseada. Para usar este método bastaría con que el usuario pudiera emitir cualquier tipo de sonido.

El orden es igual que el anterior.

7.4.3 Método del barrido por grupos

Aparecen grupos de letras. El barrido va por grupos en vez de ir por letras individuales. Cuando se selecciona un grupo, se hace un barrido sobre las letras del grupo. Se pueden hacer varios subgrupos dentro de otro grupo.

El orden sería de $\sum_{n=1}^{Número\_de\_Grupos} (\mathrm{Re}tardo * n) + \sum_{m=1}^{Numrero\_de\_Leras\_por\_grupo} (\mathrm{Re}tardo * n) = 3{,}25$

Para una implementación de 6 grupos de 5 palabras, con un retardo de medio segundo, el tiempo medio por letra sería de 2,75 segundos. Para escribir *"HOLA"* tardaría de media 11 segundos. Como se ve, es una mejora apreciable comparado con el método del barrido.

Los inconvenientes: necesita algo de entrenamiento. Y es menos intuitivo que el barrido. Además sigue siendo muy lento.

7.4.4 Método de las diagonales.

DESCRIPCIÓN:

Este método se ha desarrollado pensando en usuarios con un pulso muy errático, que sean incapaces de acertar adecuadamente en zonas concretas de la pantalla de una PDA.

La pantalla que se encontrará el usuario tendrá todos los caracteres posibles distribuidos en cuatro bloques de letras en las esquinas de la pantalla de la agenda electrónica. Si el usuario quiere seleccionar una letra, tiene que hacer un rasgo en la pantalla en esa dirección. La puede hacer en cualquier parte de la pantalla, y no necesita que sea exacta.

Al hacer la diagonal, el bloque de caracteres de esa esquina se distribuiría por las cuatro esquinas, y el usuario repetiría la diagonal en el sentido del grupo de letras donde esté la letra deseada. Entonces con la última diagonal, se seleccionaría la letra deseada.
Esta sería la secuencia de pasos necesaria para escribir la H de “HOLA”

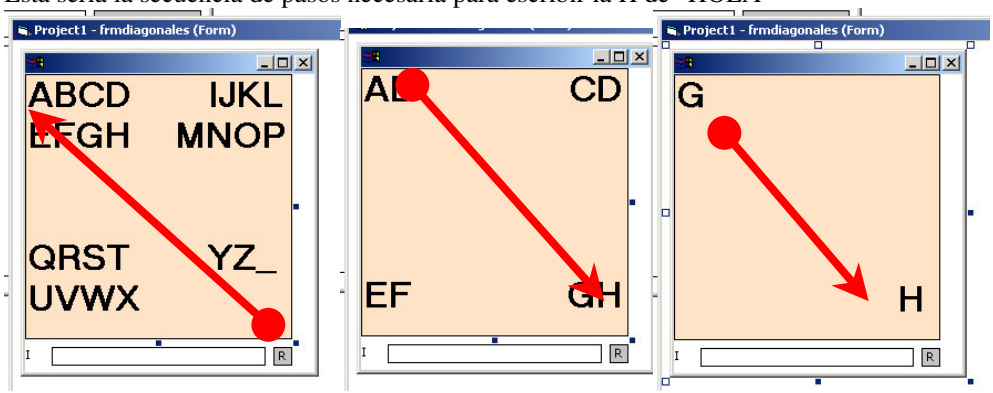


Figura 7.3 Método de las diagonales

El orden es de 3, es decir, hace falta 3 iteraciones del usuario para seleccionar una tecla.

Las capacidades del usuario para usar este método se limitan a ser capaz de hacer rasgos amplios encima de la pantalla. No necesita tener precisión para accionar zonas concretas de la pantalla.

Es mucho más rápido que el método del barrido, y necesita un usuario con más capacidades de movimiento.

Es menos intuitivo que el del barrido, y requiere cierto aprendizaje.

A partir de aquí el resto de los métodos necesitan la capacidad del usuario para dar en zonas concretas de la pantalla.

7.4.5 Método de las pulsaciones repetidas

DESCRIPCIÓN:

Es el método usado en los móviles.

Se ponen grupos de letras, y cada vez que se pulse una saldrá en orden todos los caracteres que contiene ese grupo hasta llegar a la letra deseada.

El orden depende de las letras que estén en cada grupo.

Orden $\sum_{n=1}^{NumeroLetrasPorBoton} n * \mathrm{Re}tardo$

Para un sistema con 6 grupos de 6 teclas, el orden medio sería de 3.

Hay un problema, si se desea poner dos letras que estén en el mismo grupo, el sistema tiene que distinguir si es la siguiente letra del grupo o una letra nueva. Lo normal es usar el tiempo, si pasa el tiempo adecuado, supone que quieres escribir una nueva letra.
Esto es un problema para el usuario-tipo de este tipo de aplicación, ya que alguno de los usuarios tienen una velocidad reducida.

Además necesita cierta precisión para acertar en los grupos de palabras, y es relativamente lento, más lento que otros métodos, por lo que lo hemos descartado incluso para los prototipos.

El interfaz de usuario sería idéntico al del método del agrupamiento de letras, que describiremos en el punto 7.2.7.

7.4.6 Método puro usando Bases de Datos

El usuario tiene una lista de palabras sugereridas. Además tiene una serie de botones, cada uno de ellos con grupos de letras. La lista de sugerencias tiene un barrido. El usuario tiene que pulsar en los grupos de letras, y la lista de sugerencias sólo mostrará las combinaciones que encajan con lo que el usuario ha introducido. Sólo se introducirá una pulsación por letra deseada. De esta forma, si quisiera escribir "HOLA" el usuario tendría que pulsar en

	Tecla pulsada	Sugerencias
Primera pulsación	"GHIJKL"	GATO INES LAZO GENTE....
Segunda pulsación	"MNÑOPQ"	INES LOS GÓTICO HORTERA...
Tercera pulsación	"GHIJKL"	INICIAL HOLOCAUSTO GOL HOLA
Cuarta pulsación	"ABCDEF"	HOLA INICIO INICIAR LOLA

Figura 7.4 Método puro de las bases de datos

Podemos esperar a que llegue la palabra, o seguir metiendo bloques que coincidan con la siguiente letra. Este es el sistema de texto predictivo T9 usado ampliamente en los teléfonos móviles.

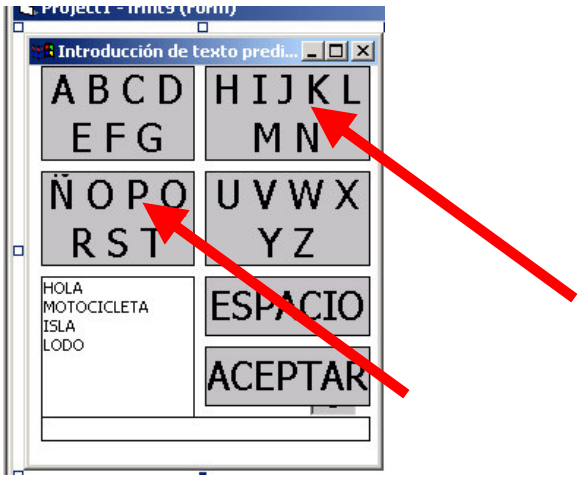


Figura 7.5 Bases de Datos con dos pulsaciones realizadas para escribir “HOLA”

Como ventaja, es bastante rápido, ya que en cuanto tengamos la palabra sugerida, la podemos seleccionar entera, sin necesidad de acabar la frase. Como gran desventaja, no podremos introducir una palabra que no exista en el dispositivo. Y en ese caso tendríamos que usar otro método de escritura para acabar lo que queremos decir.

En general, se ve cómo la ventaja de velocidad se ve muy afectada por unas pocas palabras que fallen en la búsqueda, ya que habrá que cambiar de método para completarla. Además necesita cierto entrenamiento.

Al ser un método basado en botones, es necesario que el usuario pueda tener una precisión suficiente como para acertar en ellos. Al mismo tiempo, al ser muy pocos botones son de un tamaño considerable. Para aceptar una palabra hay que pulsar en la lista de palabras sugeridas.

7.4.7 Método de Agrupamiento de Letras

Figura 7.6 Método de Escritura de Agrupamiento por letras

DESCRIPCIÓN:

Las letras están distribuidas en bloques. Cada vez que se selecciona un bloque, estas letras se redistribuyen en todos los bloques. Cuando en cada bloque queda una letra, al seleccionarse, se introducirá en la caja de texto donde se van mostrando las palabras.
Este método, con 6 bloques y 6 letras por bloque es de orden 2, ya que para cada letra se necesitan dos pulsaciones.

Para que el usuario pueda usarlo, necesita tener cierta precisión para acertar en los botones. También lleva un poco de entrenamiento, aunque es relativamente intuitivo.

7.4.8 Método de las vocales

Figura 7.7 Método de las vocales

DESCRIPCIÓN:

Este método es una variación del anterior. Se basa en usar heurística a la introducción de texto. En el español (y en la mayoría de los alfabetos) hay muchas más vocales que consonantes, de hecho, prácticamente la mitad de las letras que se usan son vocales, y mientras que las consonantes son 23, las vocales son sólo 5.
Nosotros aprovechamos este hecho para poner las vocales por separado. De esta forma, usando 5 bloques de 5 consonantes y las 5 vocales, obtenemos un método de orden aproximado de 1,5.

El funcionamiento es parecido al de agrupamiento de palabras, excepto que las vocales siempre están disponibles.Al tener más botones, éstos son más pequeños, y se requiere de más pericia por parte del usuario.

7.4.9 Método de Rasgos en Grupos de Letras

Figura 7.8 Método de rasgos en grupos de letras

DESCRIPCIÓN:

En la pantalla aparecen unos cuadros, cada uno de ellos con cuatro letras colocadas arriba, abajo, a la izquierda y a la derecha.

El usuario tiene que apretar el grupo en el que está la letra deseada, y una vez que lo tiene pulsado, tiene que hacer un movimiento en la dirección en la que esté colocada la letra dentro del cuadro. No es necesario que el movimiento acabe en el mismo grupo, puede soltar el dedo en cualquier sitio.

Con este método obtenemos un orden uno, con cada pulsación se consigue escribir una letra, como en un teclado físico. Además los bloques son relativamente grandes y no se necesita una gran precisión.

Como desventaja, necesita cierto aprendizaje, y el usuario necesita no sólo precisión para acertar en un punto sino también no hacer movimientos erráticos, y cierta habilidad manual que no necesitan los otros métodos.

7.4.10 Resumen:

	Orden	Velocidad	Capacidad física Requerida	Aprendizaje
Barrido	1	Muy Lento	Muy Poco	Muy Poco
Barrido Sonido	1	Muy lento	Muy Poco	Muy Poco
Barrido por grupos		Lento	Muy Poco	Poco
Diagonales	3	Media	Poco	Alto
Pulsaciones repetidas	3	Media	Media	Medio
Puro Bases de Datos	1	Rápida*	Media	Medio
Agrupamiento de Letras	2	Rápido	Medio	Medio
Vocales	1,5	Rápido	Medio	Alto
Rasgos en grupos de Letras	1	Muy Rápido	Alto	Alto

Figura 7.9. Resumen de los métodos de escritura

7.4.11 Conclusión

Hay multitud de características distintas en cada usuario de esta aplicación, por lo que no se puede hablar de métodos mejores que otros, sino de métodos más adecuados a problemáticas individuales.

En nuestro caso, hemos desarrollado este proyecto en el centro Obregón perteneciente a Asprona, a través del responsable de educación del centro, Víctor, que nos sugirió que hiciéramos una aplicación adaptada a una alumna.

La Alumna – usuaria de la aplicación tiene severas dificultades de pronunciación, y de movimiento. Sin embargo, aunque no tiene capacidad de desplazarse por sí misma, y necesita la ayuda de una silla de ruedas, y su capacidad de movimiento es muy reducida, dispone de un pulso bastante desarrollado, y capacidad de leer y escribir, necesario para este tipo de aplicación basada en texto.

Por ello, después de que ella probara un prototipo de los distintos métodos posibles, nos decantamos por el método de grupo de botones y el de vocales.

Por su capacidad de acertar en puntos concretos de la pantalla con gran precisión descartamos métodos más lentos, como el del barrido o el de las diagonales.

El método de las diagonales necesita tres rasgos para un carácter, mientras que con dos pulsaciones se puede obtener un carácter en el método de los grupos de caracteres.

El método puro de la base de datos comprobamos que tenía grandes problemas cuando no se acertaba a la primera. Basta ver a los usuarios de móviles cómo tienen que retroceder y escribir la palabra de forma manual para poder seguir. A pesar de eso, la capacidad de no necesitar escribir una palabra entera para poder usarla era muy interesante. Decidimos añadir esta funcionalidad al método implementado, como un añadido. Concretamente, mientras está el usuario escribiendo una palabra, en

cualquier momento puede pulsar sobre el texto que está escribiendo, para que le salga una lista con las palabras coincidentes con las últimas escritas.

Para poblar la tabla de las palabras usadas, primero buscamos una lista de las palabras españolas. Pero vimos que tenía numerosos problemas, unos problemas técnicos y otros prácticos.

Teníamos más de 80.000 palabras disponibles, y para un dispositivo portátil es una cantidad considerable, que consumía importantes recursos de memoria para este dispositivo, y con el que conseguíamos una velocidad exasperadamente lenta. Además era muy poco útil, ya que al ser todas las posibles palabras del español, y sin ninguna clasificación, la lista de palabras sugeridas era poco menos que inútil. Salían palabras muy poco usadas en el mismo lugar que palabras muy usadas.

Recortar la lista de palabras tampoco solucionaba mucho, por razones prácticas, ya que seleccionar palabras entre una lista de 80.000 es un trabajo extenuante. Además hay palabras que evidentemente van a salir antes. Por ejemplo, si el usuario ha escrito "QU" puede que la palabra "*QUERIENDO"* sea lo suficientemente común como para que le salga de sugerencia, pero es evidente que es mucho más probable que le salga la palabra "*QUE"* antes.

Por este motivo, desarrollamos una aplicación auxiliar, *GENERADOR DE DICCIONARIO,* que usaba textos como entrada, y devolvía una lista de las palabras usadas, y cada una de ellas con el número de veces que había sido usada, la "popularidad" de las palabras.

De esta forma, usamos textos de un nivel léxico sencillo, pensando en que es el tipo de lenguaje que se va a emplear. Así que empleamos textos de cuentos infantiles, y de cursos de español para extranjeros, así como textos de la E.S.O. para niños. De esta manera conseguimos tener palabras realmente usadas, y que se podían ordenar perfectamente por orden de popularidad en el español. Además las palabras que la usuaria vaya usando, serán introducidas también en el sistema, y con el tiempo, se adaptará a su forma de escribir.

7.5 Estudio para realización de la síntesis de voz

Como hemos comentado uno de los requisitos fundamentales es la reproducción del sonido, para que la comunicación pueda realizarse con todas las personas del entorno, independientemente de si adquirieron a lo largo de su vida la capacidad de lectura y de escritura, y para que sea mucho más natural. Sin embargo, nuestro proyecto inicialmente no contemplaba esta posibilidad.

Tras es el estudio de los sintetizadores de voz que se encuentran en este momento en el mercado, nos hemos dado cuenta que para este tipo de dispositivos apenas se encuentra información, por lo que hemos decidido crear nuestro propio sistema de reproducción de la voz.

La primera decisión que hay que tomar es si la reproducción se hará por palabras, por sílabas o por letras.

- Si queremos reproducir los textos por palabras deberemos disponer de una base de datos con todas las palabras reproducibles. Esto no es factible, por dos razones, en primer lugar los recursos del dispositivo son limitados, y en segundo lugar, el léxico español es demasiado amplio como para poder hacer una partición lo suficientemente buena que abarcara prácticamente todas las palabras que fuese a utilizar nuestro usuario final. Además habría que diseñar una alternativa si la palabra no se encontrase entre las reproducibles. Por tanto esta solución queda descartada.

- Si queremos reproducir por letras, es decir, por fonemas. Esta es una gran alternativa, sin embargo, es una posibilidad que constituye en sí misma un proyecto independiente. Tendríamos que hacer un estudio fonético muy profundo y todo ello aplicable a dispositivos móviles. Supone una parte demasiado amplia y descuidaríamos el sentido real de nuestro aplicación. Por lo tanto esta solución queda igualmente descartada.
- Reproducir por sílabas es una alternativa intermedia. Por un lado los recursos que consumiría un silabario serían soportables por un dispositivo de este tipo, y por otro lado consideramos que el sonido puede ser lo suficientemente bueno como para que la comunicación sea eficiente.

Decidido que la reproducción será basada en sílabas queda resolver un importante handicap: Dividir las palabras en sílabas.
Para ello vamos a hacer uso de las reglas que tiene el español para dividir las palabras.

REGLA 1.- En las sílabas, por lo menos, siempre tiene que haber una vocal. Sin vocal no hay sílaba.

REGLA 2.- Cada elemento del grupo de consonantes inseparables, mostrado en la figura 2.2, no puede ser separado al dividir una palabra en sílabas.

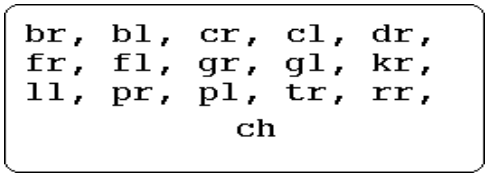


Figura 7.10 Grupo de consonantes inseparables.

REGLA 3.- Cuando una consonante se encuentra entre dos vocales, se une a la segunda vocal.

Ejemplo:

Palabra	Sílabas
Une	u+ne

Figura 7.11 Ejemplo Regla3 .

REGLA 4.- Cuando hay dos consonantes entre dos vocales, cada vocal se une a una consonante.

Ejemplo:

	Palabra	Sílabas
Ejemplo	componer	com+po+ner
Ejemplo de excepción	aprender	a+pren+der

Figura 7.12 Ejemplo Regla4

Excepción: Esto no ocurre en el grupo de consonantes inseparables (regla 2).

REGLA 5.- Si son tres las consonantes colocadas entre dos vocales, las dos primeras consonantes se asociarán con la primera vocal y la tercer consonante con la segunda vocal.

Excepción.- Esta regla no se cumple cuando la segunda y tercera consonante forman parte del grupo de consonantes inseparables.

Ejemplo:

	Palabra	Sílabas
Ejemplo	transporte	trans+por+te
Ejemplo de excepción	cumple	cum+ple

Figura 7.13 Ejemplo Regla 5

REGLA 6.- Las palabras que contienen una h precedida o seguida de otra consonante, se dividen separando ambas letras.

Ejemplo:

Palabra	Sílabas
anhelo	an+he+lo

Figura 7.14 Ejemplo Regla6

REGLA 7.- El diptongo es la unión inseparable de dos vocales. Se pueden presentar tres tipos de diptongos posibles:

(Una vocal abierta + una vocal cerrada) | (Una vocal cerrada + una vocal abierta) | (Una vocal cerrada + una vocal cerrada)

Son diptongos sólo las siguientes parejas de vocales: ai, au, ei, eu, io, ou, ia, ua, ie, ue, oi, uo, ui, iu, ay, ey, oy.

Ejemplo:

Palabra	Sílabas
Jaula	jau+la

Figura 7.15 Ejemplo Regla7

La unión de dos vocales abiertas o semiabiertas no forman diptongo, es decir, deben separarse en la segmentación silábica. Pueden quedar solas o unidas a una consonante.

Ejemplo:

Palabra	Sílabas
Aereo	a+e+re+o

Figura 7.16 Ejemplo2 Regla7

REGLA 8.- La h entre dos vocales, no destruye un diptongo.

Ejemplo:

Palabra	Sílabas
ahuyentar	ahu+yen+tar

Figura 7.17 Ejemplo Regla8

REGLA 9.- La acentuación sobre la vocal cerrada de un diptongo provoca su destrucción.

Ejemplo:

Palabra	Sílabas
María	Ma+rí+a

Figura 7.18 Ejemplo Regla9

REGLA 10.- La unión de tres vocales forma un triptongo. La única disposición posible para la formación de triptongos es la que indica el esquema:

Vocal cerrada + (vocal abierta | vocal semiabierta) + vocal cerrada

Sólo las siguientes combinaciones de vocales, forman un triptongo: iai, iei, uai, uei, uau, iau, uay, uey

Bajo estas reglas habrá que construir un método que se encargue de la división de las palabras en sílabas para posteriormente ser reproducidas.

Capítulo 8. ANÁLISIS DEL SISTEMA. DEFINICIÓN DEL PROBLEMA

El análisis se dedica a la comprensión y modelado de la aplicación y del dominio en el cual funciona. La entrada inicial de la fase de análisis es una descripción del problema que hay que resolver y proporciona una visión general conceptual del sistema propuesto. Otras entradas adicionales del análisis son un diálogo con el cliente y un conocimiento de fondo del mundo real. La salida del análisis es un modelo formal que captura los tres aspectos esenciales del sistema: los objetos y sus relaciones, el flujo dinámico de control y la transformación funcional de datos que están sometidos a restricciones.

8.1 Resultados de la entrevista

La captura de requisitos no es tan sencilla en este caso, debido a las dificultades que tienen a la hora de comunicarse las personas a las que va dedicado este proyecto, por eso, nos basamos en otras experiencias, como pueden ser las personas más cercanas (familiares, tutores, profesores ...)

También hay que destacar el hecho de que es un campo muy amplio y experimental por lo que muy difícil encontrar una aplicación genérica, ya que las características de cada persona son únicas y especiales. Hay que tener en cuenta:

- Una valoración del nivel en el área de la motricidad
- Una valoración del nivel en el área de la comunicación

Esta aplicación está basado en la escritura, es decir, la comunicación se hará a través de letras, formando palabras, puesto que hay un mayor número de herramientas que basan la comunicación en otro tipo de lenguajes visuales como puede ser el lenguaje SPC.

Debido a las limitadas habilidades de movilidad y de visión en muchos casos, los controles de la herramienta tendrán que ser lo más grandes posibles, sin poder utilizar pequeños botones o un tamaño de fuente reducido, así como un texto claro y no ambiguo en todos los formularios de la aplicación.

Requisito indispensable que nos solicitaron de cara a una mayor funcionalidad, fue la posibilidad de que los textos que el usuario escribiese pudiesen ser reproducidos, puesto que sino la comunicación sería exclusivamente entre personas que hubiesen adquirido la capacidad de la lectura y la escritura a lo largo de su vida.

Otra de las peticiones que se nos realizaron fue que los botones hardware del dispositivo que son fácilmente pulsables tuviesen su función de cara a la aplicación, a poder ser, las funciones más comunes como podía ser reproducir el sonido o borrar un carácter.

Además sería interesante, que el usuario pudiese disponer de una opción para guardar sus frases más comunes de manera que la comunicación se pudiese agilizar, así como poder borrarlas.

8.2 Definición de actores

Los actores son personas o entidades externas a la aplicación que interactúan con ella, realizando un intercambio de información. Éste podrá ser tanto de entrada como de salida.

En sistemas más complejos, un actor puede representar varios papeles. En estos casos, se definen diferentes actores para representarlos. Sin embargo, éste no es el caso de nuestro sistema, ya que sólo hay un posibles actor, que se corresponde exactamente con la persona discapacitada que usará la aplicación.

El actor de esta herramienta tiene acceso a todas las funciones que podrá manejar de forma autónoma e independiente, de manera que podrá conseguir una comunicación más natural que con los antiguos sistemas.

ACT-0001	**Alumno**
Descripción	Este actor *la persona con parálisis cerebral usuaria de la aplicación*

Figura 8.1. Descripción actor de los casos de uso

8.3 Diagramas de casos de uso

Se emplean para visualizar el comportamiento del sistema, una parte de él o de una sola clase. De forma que se pueda conocer como responde esa parte del sistema. El diagrama de uso es muy útil para definir como debería ser el comportamiento de una parte del sistema, ya que solo especifica como deben comportarse y no como están implementadas las partes que define. Un caso de uso especifica un requerimiento funcional, es decir indica esta parte debe hacer esto cuando pase esto.

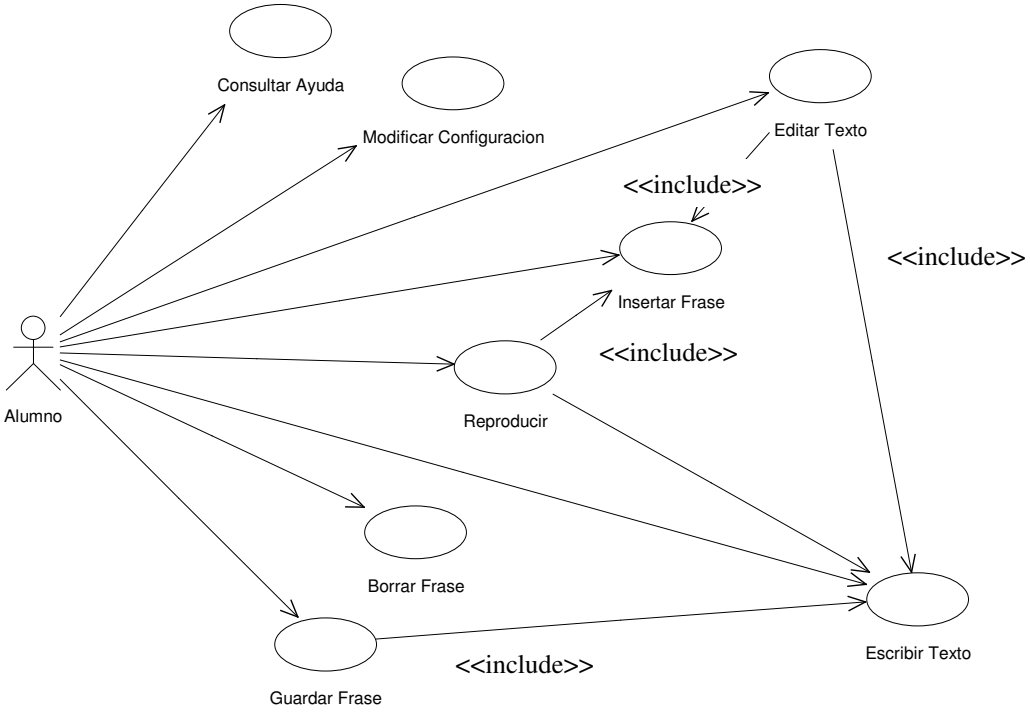


Figura 8.2 Diagrama de los casos de uso

8.4 Descripción de los casos de uso

El comportamiento de un caso de uso se puede especificar describiendo un flujo de eventos de forma textual, lo suficientemente claro para que una persona ajena al sistema lo entienda fácilmente. A la hora de describir este flujo de eventos, se debe incluir cómo y cuando empieza y termina el caso de uso en cuestión, cuándo interactúa con los actores y qué objetos intercambian, el flujo básico y los alternativos de comportamiento.

A medida que se obtiene una mejora en la comprensión de los requisitos del sistema, estos flujos de eventos se especifican gráficamente mediante los diagramas de interacción. Normalmente, se utiliza un diagrama de secuencia para especificar el flujo principal de un caso de uso.

8.4.1 Consultar Ayuda

UC-0001	**Consultar ayuda**	
Descripción	*El sistema deberá permitir que el usuario pueda consultar la ayuda sobre el uso de determinadas funciones de la aplicación en cualquier momento*	
Secuencia normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *Solicita ayuda al sistema*
	2	*El sistema le ofrece un número determinado de posibilidades sobre las que puede ayudarle*
	3	Actor Alumno (ACT-0001) *escoge un tema de ayuda*
	4	*El sistema le proporciona la ayuda correspondiente con el tema seleccionado*

Figura 8.3. Descripción caso de uso Consultar Ayuda

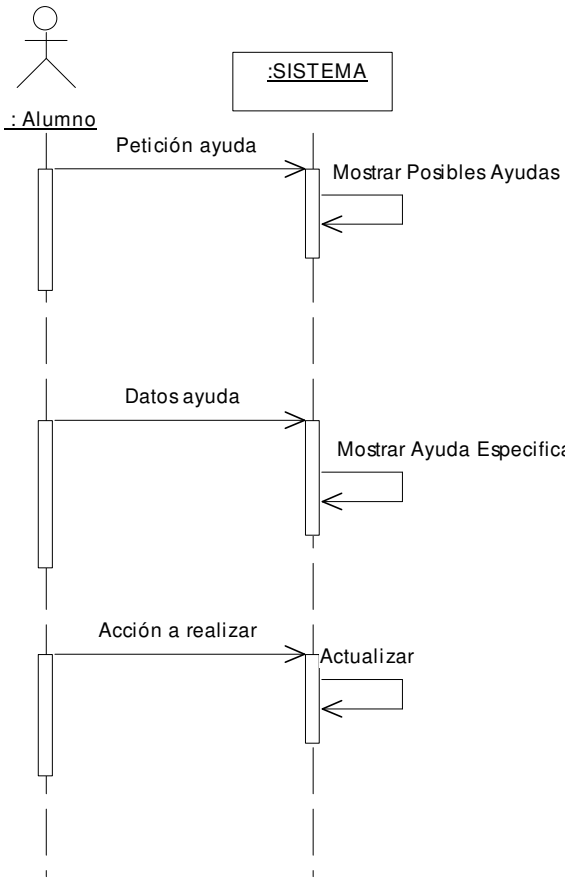


Figura 8.4. Diagrama del caso de uso Consultar Ayuda

8.4.2 Modificar Configuración

UC-0002	**Modificar configuración**	
Descripción	*El sistema deberá proporcionar la posibilidad de cambiar el método de escritura y si el usuario así lo desea*	
Secuencia normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *solicita modificar la configuración*
	2	*El sistema le ofrece la posibilidad de modificar o cambiar el método de escritura*
	3	Actor Alumno (ACT-0001) *escoge los cambios a realizar*
	4	*El sistema actualiza los cambios*

Figura 8.5 Descripción caso de uso Modificar configuración

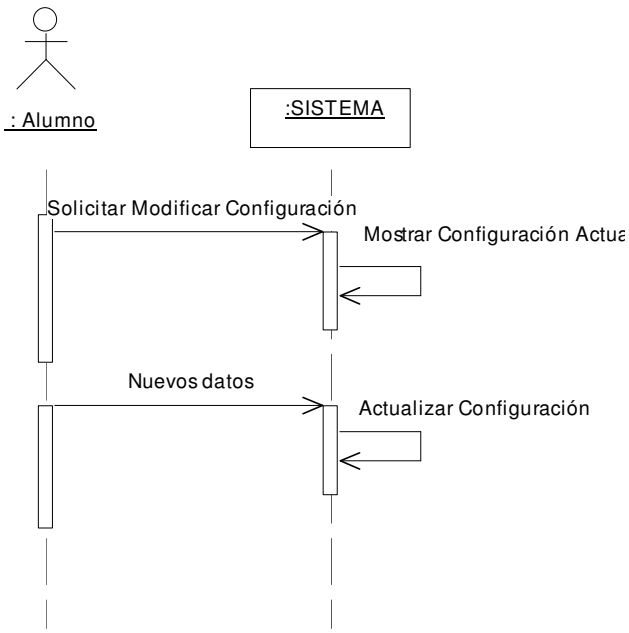


Figura 8.6 Diagrama del caso de uso Modificar configuración

8.4.3 Escribir texto

UC-0003	**Escribir texto**	
Descripción	*El sistema deberá permitir que el usuario podrá escribir en diferentes métodos de escritura cualquier texto*	
Secuencia normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *escoge las letras una a una a escribir*
	2	*El sistema actualiza el texto*
	3	Si *se equivoca en algún carácter,* actor Alumno (ACT-0001) *puede solicitar borrar el texto*

Figura 8.7. Descripción casos de uso Escribir Texto

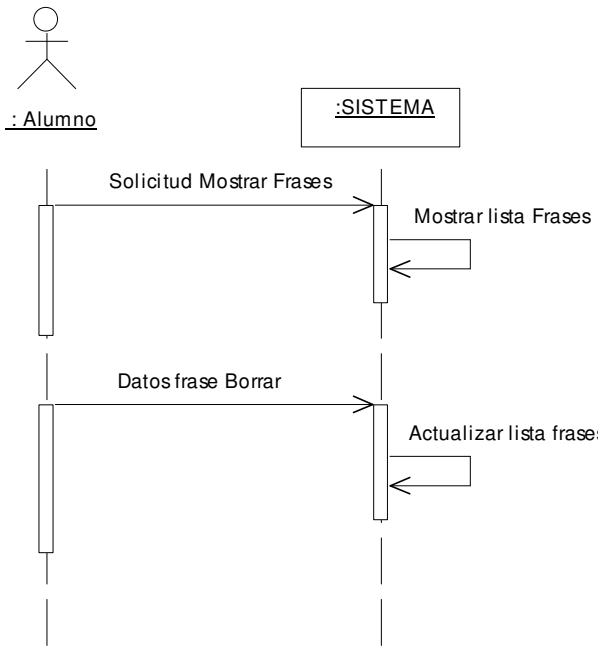


Figura 8.8 Diagrama del caso de uso Escribir Texto

8.4.4 Insertar frase

UC-0004	**Insertar frase**	
Descripción	*El sistema tiene que ofrecer al usuario una lista de frases comunes para que pueda disponer de ellas en cada momento de manera que se agilice la comunicación*	
Secuencia normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *solicita ver las frases disponibles*
	2	*El sistema muestra las frases que tiene almacenadas*
	3	Actor Alumno (ACT-0001) *escoge la frase con la que quiere comunicarse*
	4	*El sistema actualiza el texto*

Figura 8.9. Descripción caso de uso Insertar Frase

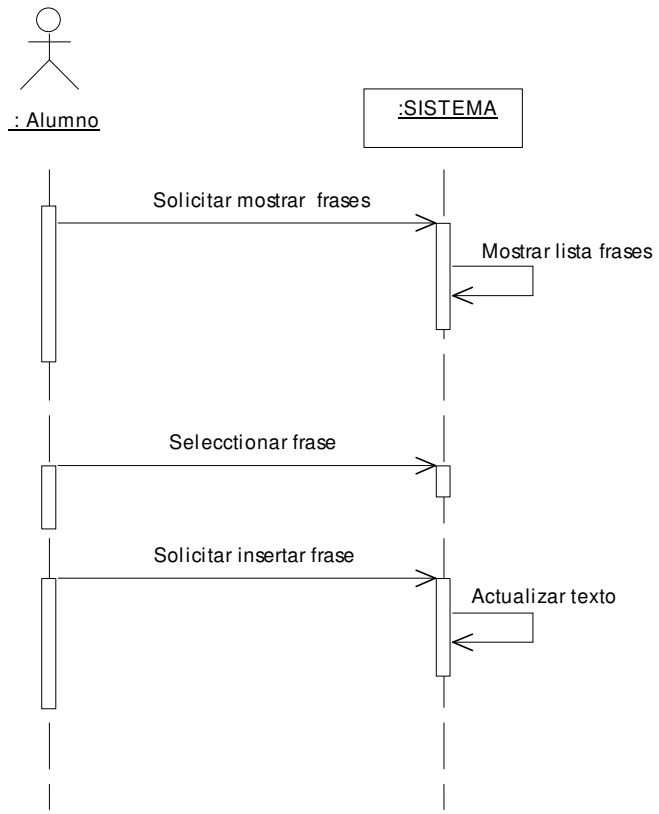


Figura 8.10 Diagrama del caso de uso Insertar Frase

8.4.5 Borrar frase

UC-0005	**Borrar frase**	
Descripción	*El sistema tiene que ofrecer la posibilidad de borrar las frases que tiene almacenadas si el usuario considera que no son lo suficientemente útiles*	
Secuencia normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *solicita borrar una frase*
	2	*El sistema muestra la lista de frases de la que dispone*
	3	Actor Alumno (ACT-0001) *escoge la frase que quiere borrar*
	4	*El sistema actualiza la lista de frases*

Figura 8.11. Descripción caso de uso Borrar Frase

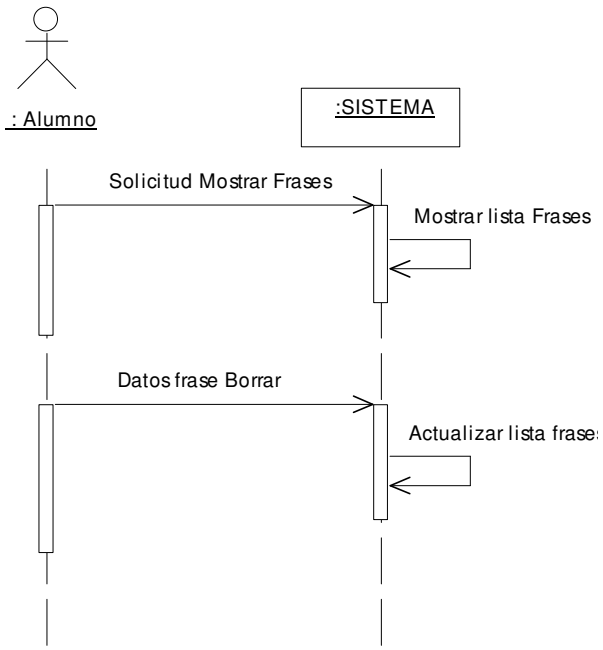


Figura 8.12. Diagrama del caso de uso Borrar Frase

8.4.6 Guardar frase

UC-0006	**Guardar frase**	
Descripción	*El sistema tiene que permitir al usuario que guarde sus frases si lo desea para su posterior uso para conseguir una comunicación más rápida y natural*	
Secuencia normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *Escribe la frase que desea guardar*
	2	Actor Alumno (ACT-0001) *solicita guarda la frase*
	3	*El sistema guarda la frase*

Figura 8.13. Descripción caso de uso Guardar Frase

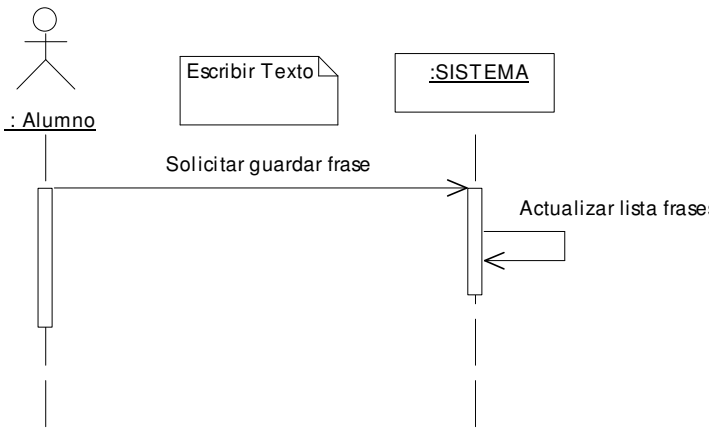


Figura 8.14. Diagrama del caso de uso Guardar Frase

8.4.7 Reproducir el texto

UC-0007	Reproducir el texto	
Descripción	*El sistema tiene que permitir reproducir*	
Secuencia normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *Escribirá el texto que desee reproducir*
	2	Actor Alumno (ACT-0001) *solicita su reproducción*
	3	*El sistema reproducirá el texto correspondiente*

Figura 8.15. Descripción caso de uso Reproducir el Texto

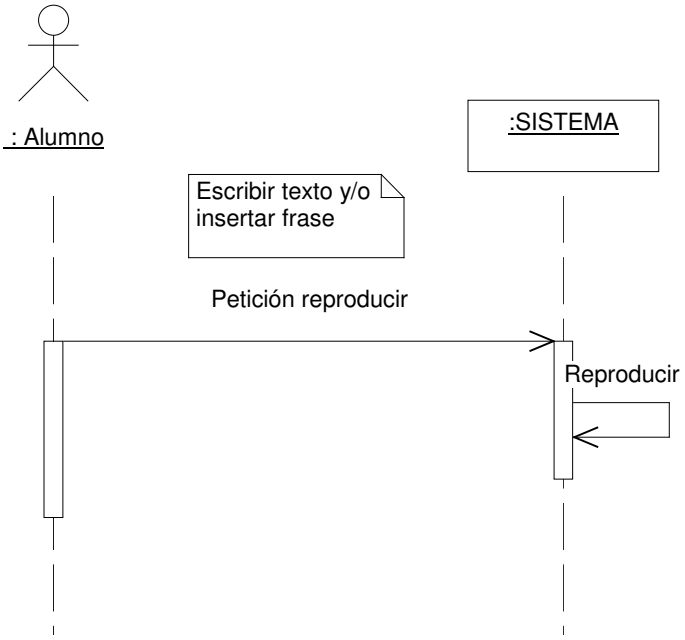


Figura 8.16. Diagrama del caso de uso Reproducir el Texto

8.4.8 Editar Texto

UC-0008	**Editar el texto**	
Descripción	*El sistema tiene que permitir Editar el texto*	
Secuencia normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *Escribirá el texto que desee editar*
	2	Actor Alumno (ACT-0001) *solicita editar dicho texto*
	3	*El sistema editará el texto correspondiente*

Figura 8.17. Descripción caso de uso Editar Texto

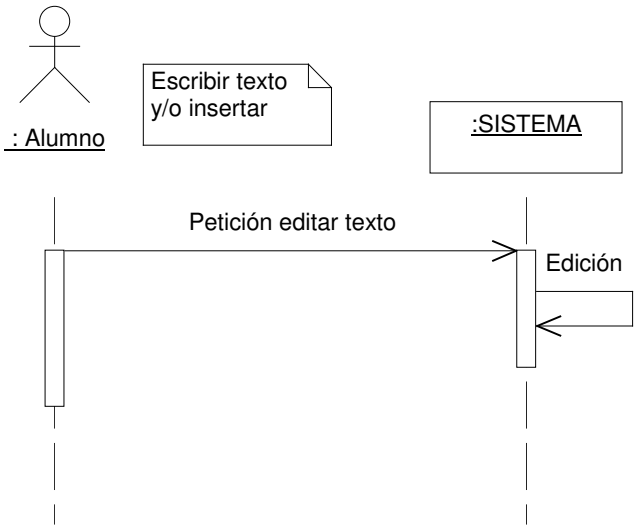


Figura 8.18. Diagrama del caso de uso Editar Texto

8.5 Modelo de Objetos

El modelo de objetos muestra la estructura estática de datos correspondientes al sistema del mundo real, y la organiza en segmentos manejables describiendo clases de objetos del mundo real, y sus relaciones entre sí. Lo más importante es la organización de más alto nivel del sistema, en clases conectadas mediante asociaciones.

8.5.1 Diagrama inicial de clases

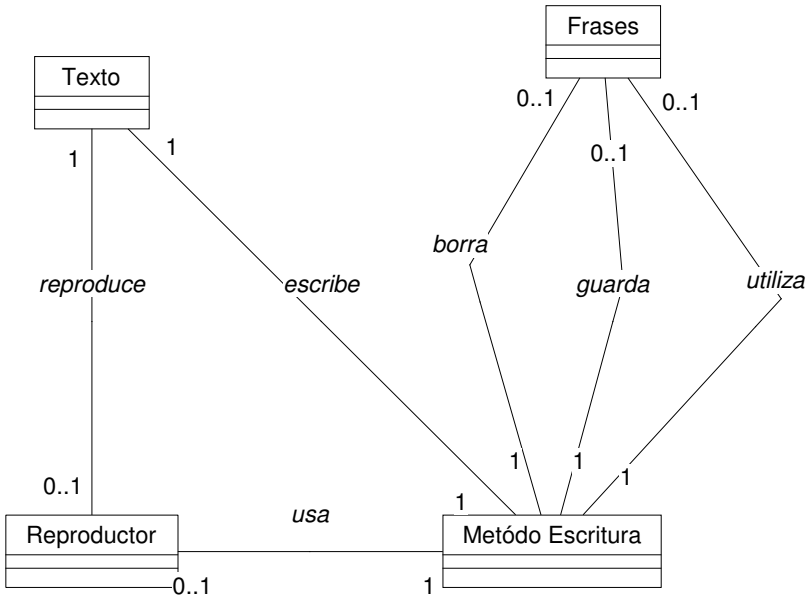


Figura 8.19 Diagrama inicial de clases

Capítulo 9. DISEÑO

Una vez completada la fase de análisis, y antes de pasar a la implementación y programación, es necesaria una actividad de nivel superior.

La fase de diseño es una fase intermedia entre la vista abstracta de un sistema y la implementación real de éste. Los productos de esta fase pueden ser más cercanos a una visión abstracta o a una visión implementable.

9.1 Casos de Uso

9.1. 1 Consultar Ayuda

UC-0001	**Consultar ayuda**	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *Este caso de uso describe la consulta de la ayuda en línea por parte del usuario*	
Secuencia normal	**Paso**	**Acción**
	1	El actor Alumno (ACT-0001) *pulsa un botón del dispositivo móvil*
	2	El sistema *despliega un menú sobre las posibles consultas de ayuda*
	3	Si *quiere consultar alguno de los temas que le ofrece la herramienta,* el actor Alumno (ACT-0001) *pulsa el botón que se corresponde con la ayuda deseada*
	4	El sistema *despliega un formulario con la información necesaria para proporcionar la ayuda*

	5	El actor Alumno (ACT-0001) *puede escoger las siguientes opciones:* *- situarse en el formulario correspondiente al tema de la ayuda deseada* *- volver al menú de ayuda* *- volver al punto donde se encontraba antes del caso de uso consultar ayuda*
Excepciones	**Paso**	**Acción**
	-	-
	-	-
Importancia	importante	
Comentarios	El actor puede hacer las consultas a la ayuda en cualquier momento de uso de la aplicación a través de los botones hardware	

Figura 9.1 Caso de uso consultar ayuda

9.1. 2 Guardar Frase

UC-0002	**Guardar frase**	
Descripción	*El sistema tiene que permitir que el actor pueda almacenar en la aplicación las frases que quiera para su posterior uso*	
Precondición	hay n frases	
Secuencia normal	**Paso**	**Acción**
	1	Use case escribir texto (UC-0006) is performed
	2	Actor Alumno (ACT-0001) *después de haber escrito el texto que desea, pulsa el botón de menú opciones que es accesibles desde cualquiera de los dos métodos de escritura*
	3	El sistema *despliega el menú opciones compuesto por varios botones, entre ellos el botón frases*
	4	Actor Alumno (ACT-0001) *pulsa el botón frases*
	5	El sistema *despliega el formulario frases que contiene varios botones y cajas de texto con las correspondientes frases almacenadas*
	6	Actor Alumno (ACT-0001) *pulsa el botón añadir*
	7	El sistema *añade el texto que se encontraba en ese instante en la caja de texto como una frase al fichero donde éstas se almacenan*
Postcondición	hay n+1 frases	
Importancia	Importante	
Comentarios	El sistema pretende tener almacenadas las frases más usadas por el usuario para ganar velocidad en la comunicación	

Figura 9.2 Caso de uso insertar frase

9.1. 3 Reproducir Texto

UC-0003	**Reproducir Texto**	
Descripción	*El sistema debe permitir que el actor pueda comunicarse, es decir, si quiere escribir algún texto y reproducirlo* deberá poder hacerlo	
Secuencia normal	**Paso**	**Acción**
	1	Caso de uso escribir texto (UC-0006) es incluido
	2	Caso de uso insertar frases (UC-0004) es incluido
	3	Si *el actor ha terminado de escribir, o si lo desea en algún momento,* actor Alumno (ACT-0001) *puede :* *- pulsar un botón hardware para reproducir el texto, por ser una función básica de la aplicación* *- pulsar en el botón menú opciones y posteriormente pulsar el botón play.*
	4	El sistema *reproduce el sonido del texto correspondiente*
Importancia	vital	
Comentarios	Es el caso de uso central de la aplicación que consiste en escribir texto y posteriormente reproducirlo si el usuario lo considera preciso. Este caso de uso incluye los casos de uso insertar frases y escribir texto	

Figura 9.3 Caso de uso reproducir texto

9.1. 4 Insertar Frases

UC-0004	**insertar frases**	
Descripción	El sistema deberá permitir que *el actor pueda incorporar una frase al texto que lleva escrito hasta el momento* o durante la realización de los siguientes casos de uso:: [UC-0003] Comunicarse	
Precondición	la longitud del texto es n	
Secuencia Normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *pulsa el botón de menú opciones al que se puede acceder desde cualquiera de los métodos de escritura*
	2	El sistema *el sistema despliega el menú opciones que contiene botones entre los que se encuentra el botón frases*
	3	Actor Alumno (ACT-0001) *pulsa el botón frases*
	4	El sistema *despliega el formulario frases que contiene el botón insertar y una caja de texto, además de 2 botones que te permiten moverte por las frases almacenadas*
	5	Actor Alumno (ACT-0001) *escoge la frase y pulsa el botón insertar*
	6	El sistema *se sitúa en el método de escritura en el que el usuario estaba trabajando*

Postcondición	la longitud del texto es >= n
Importancia	importante
Comentarios	Este caso de uso se da cuando el usuario quiere insertar una frase en la caja de texto para poder hacer uso de ella (reproducirla o editarla)

Figura 9.4 Caso de uso insertar frase

9.1. 5 Modificar configuración

UC-0005	**Modificar configuración**	
Descripción	El sistema debe permitir que si *el actor quiere pueda cambiar el método de escritura que esté empleando*	
Precondición	estado actual sin modificar	
Secuencia Normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *pulsa el botón menú opciones al que se puede acceder desde cualquiera de los métodos de escritura*
	2	El sistema *despliega el formulario opciones que contiene botones entre los que se encuentra el botón configuración*
	3	Actor Alumno (ACT-0001) *pulsa el botón configuración*
	4	El sistema *despliega un formulario con la posibilidad de cambiar de método de escritura*
	5	Si *quiere modificar algún punto de la configuración,* actor Alumno (ACT-0001) *pulsa sobre el método* de escritura con el que quiere trabajar
	6	El sistema *actualiza los cambios*
Postcondición	estado actual modificado	
Importancia	importante	
Comentarios	Este caso de uso puede realizarse en cualquier momento, forma parte del menú de opciones	

Figura 9.5 Caso de uso modificar configuración

9.1. 6 Escribir texto

UC-0006	**escribir texto**
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando *el actor quiere escribir algún texto* o durante la realización de los siguientes casos de uso: [UC-0002] Guardar frase. [UC-0003]

	Comunicarse	
Secuencia normal	**Paso**	**Acción**
	1	El sistema *proporciona dos métodos de escritura a elegir por el actor*
	2	El actor Alumno (ACT-0001) *va escogiendo mediante pulsaciones las letras, números y símbolos que desea escribir*
Excepciones	**Paso**	**Acción**
	2	- Si *escribe algún carácter equivocado,* el actor Alumno (ACT-0001) *puede borrarlo de dos formas:* *- pulsando un botón hardware por ser una función muy común en la aplicación* *- situarse en el menú opciones, pulsar editar texto y desde ahí borrar*
Importancia	Vital	
Comentarios	forma parte de la comunicación	

Figura 9.6 Caso de uso escribir texto

9.1. 7 Borrar Frases

UC-0007	**Borrar Frases**	
Descripción	*El sistema debe permitir que el usuario pueda borrar en cualquier momento alguna de las frases si así lo desea*	
Precondición	hay n frases	
Secuencia Normal	**Paso**	**Acción**
	1	Actor Alumno (ACT-0001) *pulsa sobre el botón menú opciones que es accesible desde cualquiera de los métodos de escritura*
	2	El sistema *despliega el formulario opciones que contiene botones entre los que se encuentra el botón frases*
	3	Actor Alumno (ACT-0001) *pulsa el botón frases*
	4	El sistema *despliega el formulario frases que contiene un botón borrar para eliminar una de las frases*
	5	Actor Alumno (ACT-0001) *pulsa el botón borrar, una vez escogida la frase que quiere eliminar y situándose en ella con los botones que te permiten seleccionarla*
	6	El sistema *borra la frase deseada por el usuario*
Postcondición	hay n-1 frases	
Excepciones	**Paso**	**Acción**
	5	SI *no existen frases que borrar,, el sistema no permitirá su borrado deshabilitando el botón borrar*,
Importancia	Puede quedar bien	
Comentarios	Este caso de uso se dará siempre que el actor considere que alguna de las frases que tiene almacenadas no son lo suficientemente usadas por él	

Figura 9.7 Caso de uso borrar Frase

9.1. 8 Editar Texto

UC-0008	Editar Texto	
Descripción	*El sistema debe permitir que el actor pueda editar,decir, si quiere escribir algún texto y editarlo deberá* poder hacerlo	
Secuencia normal	**Paso**	**Acción**
	1	Caso de uso escribir texto (UC-0006) es incluido
	2	Caso de uso insertar frases (UC-0004) es incluido
	3	Si *el actor ha terminado de escribir, o si lo desea en algún momento,* actor Alumno (ACT-0001) *puede : pulsar el botón editar texto en el menú opciones*
	4	El sistema *editará el texto correspondiente*
Importancia	vital	
Comentarios	Este caso de uso es especialmente bueno en situaciones en las que no es posible reproducir un texto (por causas ajenas a la aplicación) y hay necesidad de comunicarse, por lo que una solución es editar el texto para que la otra persona pueda ver lo escrito	

Figura 9.8 Caso de uso Editar Texto

9.2 Identificación de las clases

El primer paso en la construcción de un modelo de objetos es encontrar las clases necesarias para la aplicación. Las clases suelen corresponderse con sustantivos o entidades físicas. Aunque hay que evitar las estructuras de implementación propias de la computadora.

Una vez obtenidas las clases, hay que descartar todas aquellas no necesarias e incorrectas, eliminando:

- Clases redundantes: Cuando haya dos clases que expresen la misma información, hay que mantener solamente la que tenga el nombre más descriptivo.
- Clases irrelevantes: si una clase tiene muy poco o nada que ver con el problema, debe ser eliminada. Así, habrá que emplear nuestro propio criterio, porque esa clase en otro contexto podría resultar importante.
- Clases vagas: una clase debe ser algo específico, por lo que habrá que evitar clases con límites mal definidos o con un ámbito excesivo.
- Atributos: los nombres que describen objetos individuales suelen recalificarse como atributos.
- Operaciones: cuando un nombre describe una operación que se aplica a objetos y que no es manipulada en sí entonces no debe ser una clase. Sin embargo, operaciones con características propias deben ser modeladas como clases.
- Roles: el nombre de una clase no debe reflejar el papel que desempeña en una asociación.
- Estructuras de implementación: deben ser eliminadas del modelo de análisis las estructuras extrañas al mundo real, porque quizá se necesiten en una fase posterior del diseño.

A continuación se describen todas las clases encontradas junto con una breve descripción:

- Padre: Clase interfaz que surge de la necesidad de tener una clase que se comunique con las clases de interfaz de usuario.
- 8botones: Clase interfaz que se corresponde con un método de escritura concreto.

- métodoVocales: Clase interfaz que se corresponde con un método de escritura diferente al anterior.
- Texto: Clase entidad que surge de la necesidad de tener una clase que permita manipular, actualizar y guardar el texto.
- Reproductor: Clase entidad que surge de la necesidad de tener una clase que permita reproducir el texto.
- AdaptardorTexto: Clase entidad que surge de la necesidad de tener una clase que permita transformar el texto inicial en un conjunto de sonidos reproducibles.
- Ipciones: Clase interfaz que surge de la necesidad de tener una clase que agrupe las posibilidades que el usuario tiene con la aplicación.
- Frases: Clase interfaz que surge de la necesidad de tener una clase que permita el manejo de un conjunto de frases
- BotonesHardware: Clase que nos permitirá realizar los métodos de los botones externos del dispositivo móvil.
- Ayuda: Clase que permite mostrar la ayuda al usuario si así lo necesita.
- TextoPredictivo: Clase encargada de realizar las búsquedas en la base de datos y de proporcionar la lista de palabras sugeridas.
- Clase Configuración: Clase que permite escoger al usuario el método de escritura con el que desea introducir sus textos

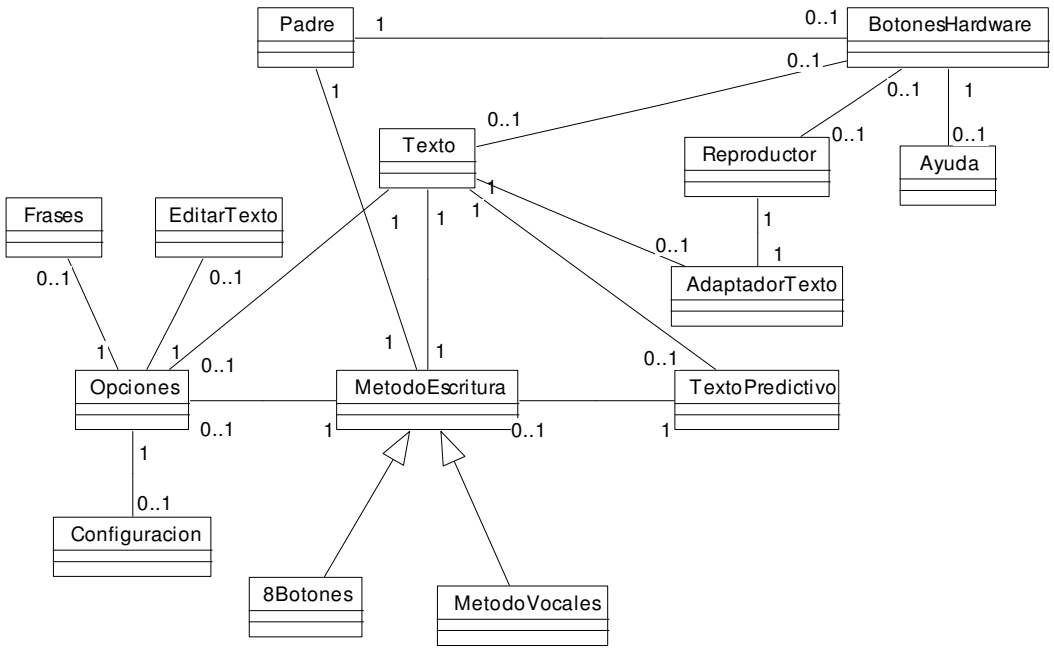


Figura 9.9 Diagrama de clases

9.3 Diccionario de Datos

- **Clase Padre:**

Es la clase controladora del sistema, encargada de arrancar la aplicación de iniciar el método de escritura que el usuario desea.

  - Atributos
    - oFrm8Botones: El formulario que contiene el método de escritura 8 botones.
    - oFrmSheila: El formulario que implementa el método de escritura Vocales.
    - Textoi: Objeto de la clase Texto que contiene el valor de la cadena escrita en cada instante.
  - Métodos
    - CreaFormularioEscritura(): como su propio nombre indica es el encargado de crear el método de escritura correspondiente.
    - ActualizaTexto(): es el que realiza la actualización del texto cuando textoi es modificada.

- **Clase Botones Hardware:**

Es la encargada de ejecutar los métodos de las funciones básicas que se encuentran en los botones externos del dispositivo

  - Atributos
    - Sin atributos
  - Métodos
    - BotónPresionado (int Boton): es el encargado de hacer la acción correspondiente dependiendo del botón que haya sido presionado

- **Clase 8 Botones:**

Es la clase que implementa uno de los métodos de escritura utilizados. Desde esta clase el usuario tiene la posibilidad de escribir sus textos

  - Atributos
    - Textoi: Objeto de la clase Texto que contiene el valor de la cadena escrita en cada instante.
    - PalabrasBuscadas que es un ArrayList que contendrá las palabras sugeridas en el caso de que haya sido pulsada esta opción
    - Pulsación: lleva la cuenta del número de pulsación, 0,1 o 2 para la colocación de los botones
    - ÍndiceP: es el índice que lleva la cuenta de la palabra sugerida en la lista de Palabras buscadas
  - Métodos
    - ActualizaTexto():es el que realiza la actualización del texto cuando textoi es modificada.
    - PonBotones(string textoBotones): es el encargado de distribuir los botones en función de la pulsación. Está sobrecargado, si se le llama sin parámetros colocará los botones por defecto
    - BotonPresionado(string textoBoton): al pulsar una tecla se encarga de hacer las acciones pertinentes.
    - Abajo(): usado para la lista de palabras sugeridas para poder desplazarse a través de ella hacia abajo.
    - Arriba():usado para la lista de palabras sugeridas para poder desplazarse a través de ella hacia arriba

- ColocarPalabra: Coloca la lista de palabras buscadas.

- **Clase Método Vocales**

Igual que la anterior pero con una optimización del método de escritura.

  - Atributos
    - Los mismos que en el caso anterior
  - Métodos
    - Los mismos que en el caso anterior

- **Clase Texto:**

Es la clase que contiene todos los métodos relacionados con el manejo del texto escrito por el usuario, así como el texto que hasta el momento se lleva escrito.

  - Atributos
    - Cadena: contiene el texto escrito hasta el momento
    - ultimaPalabra: contiene la última palabra escrita
  - Métodos
    - Añadir(string carácter): se encarga de añadir un carácter a cadena siempre que sea necesario
    - Borrar():se encarga de borrar el último carácter escrito
    - BorrarTodo(): borra todo el texto escrito hasta el momento
    - Leer():devuelve el valor de cadena

- **Clase Frases:**

Es la clase que contiene los métodos asociados con las frases.

  - Atributos
    - Textoi: Objeto de la clase Texto que contiene el valor de la cadena escrita en cada instante.
    - LineaActiva: identifica la frase seleccionada
  - Métodos
    - BorrarFrase() :Es el encargado de borrar una frase de la lista de frases
    - InsertarFrase(): es quien inserta una frase en el texto escrito hasta el momento
    - AñadirFrase(): guarda la frase seleccionada en la lista de frases
    - Arrriba(): te permite desplazarte hacia arriba
    - Abajor(): te permite desplazarte hacia abajo

- **Clase Ayuda:**

Es la clase encargada de mostrar la ayuda.

  - Atributos
    - Ningún atributo
  - Métodos
    - MostrarAyuda(int i): es el encargado de mostrarte el tipo de ayuda seleccionada por el usuario

- **Clase Opciones:**

Es la clase que centraliza las opciones de menú que usuario puede realizar sobre el texto.

  - Atributos
    - Textoi: Objeto de la clase Texto que contiene el valor de la cadena escrita en cada instante
  - Métodos
    - Reproducir
    - EditarTexto(): se encarga de mostrar el formulario EditarTexto
    - Frases(): se encarga de mostrar el formulario Frases

        - Salir(): es quien te permite salir de la aplicación

- **Clase Editar Texto:**

Es la clase encargada de la edición del texto.

  - Atributos
    - Textoi: Objeto de la clase Texto que contiene el valor de la cadena escrita en cada instante
  - Métodos
    - BorrarTodo()
    - Borrar
    - Arriba(): te permite moverte por el texto hacia arriba
    - Abajo(): te permite desplazarte hacia abajo por el texto escrito hasta el momento

- **Clase Reproductor**

Es la verdadera encargada de la reproducción del texto

  - Atributos
    - Ninguno
  - Métodos
    - Reproducir: es el encargado de proporcionar los sonidos

- **Clase Adaptador Texto**

Es la clase que lleva a cabo las modificaciones pertinentes sobre el texto para dar lugar a una lista de ficheros reproducibles, y además implementa la síntesis de voz

  - Atributos
    - Ninguno
  - Métodos
    - Diptongos(string subcadena): es el encargado de comprobar si la cadena es un diptongo
    - Triptongo(string subcadena): es el encargado de comprobar si la cadena es un triptongo
    - TipoLetra(string subcadena): es el encargado de comprobar si la cadena es una vocal o una consonante necesaria para la división en sílabas
    - dividirSilabas(string textoAdividir): lógicamente es el encargado de divididr el texto escrito hasta el momento en una serie de sílabas
    - Preprocesado(ArraryList listaSilabas): es quien elabora una lista de sonidos casi reproducibles
    - busquedaSonidos(ArrayList listaPreprocesada): es quien elabora la lista de sonidos definitiva

- **Clase Configuración**

Es la clase usada por el usuario para escoger el método de escritura con el que desea escribir.

  - Atributos
    - Ninguno
  - Métodos
    - EscogerMetodo(): es quien se encarga de mostrar el método de escritura correspondiente

- **Clase Texto Predictivo**

Es la clase encargada de hacer las búsquedas de las palabras sugeridas.

  - Atributos
    - ListaBuscada: es quien contiene la lista de palabras sugeridas
  - Métodos

- buscar(): es quien se encarga de buscar la consulta pertinente
- MostrarErrorres(sqlCeException e): es quien muestra los errores en caso de que los haya

9.4 Diagramas de Secuencia

En el diagrama de secuencia se muestra el orden de las llamadas en el sistema. Se utiliza un diagrama para cada caso de uso a representar. Es imposible representar en un solo diagrama de secuencia todas las secuencias posibles del sistema, por ello se escoge un punto de partida. El diagrama se forma con los objetos que forman parte de la secuencia, estos se sitúan en la parte superior. De estos objetos sale una línea que indica su vida en el sistema. Esta línea simple se convierte en una línea gruesa cuando representa que el objeto tiene el foco del sistema, es decir cuando el esta activo.

9.4.1 Diagrama de Secuencia Escribir Texto

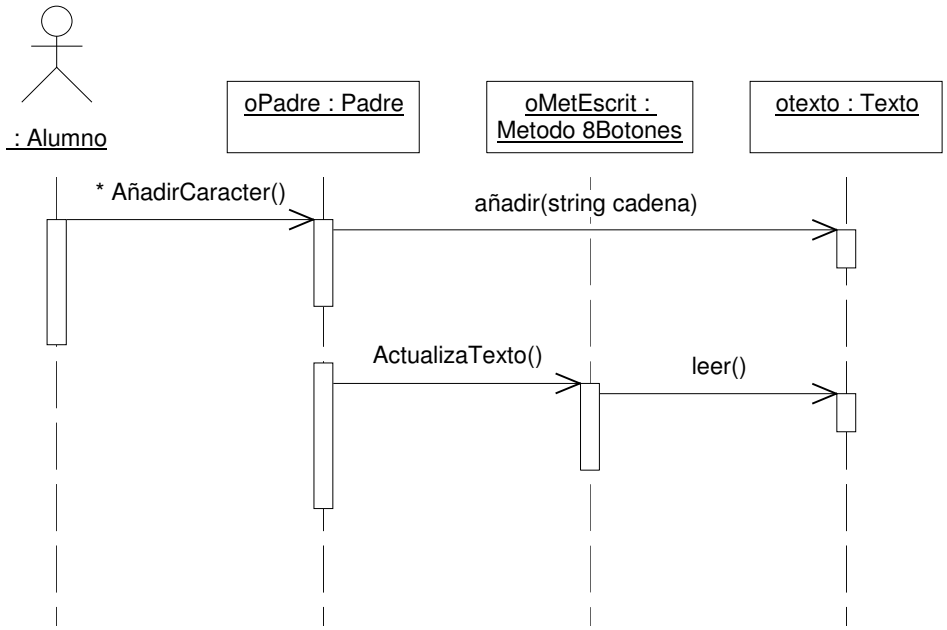


Figura 9.9 Diagrama de secuencia Escribir texto

9.4.2 Diagrama de Secuencia Borrar Carácter (Secuencia de Excepción de Escribir Texto)

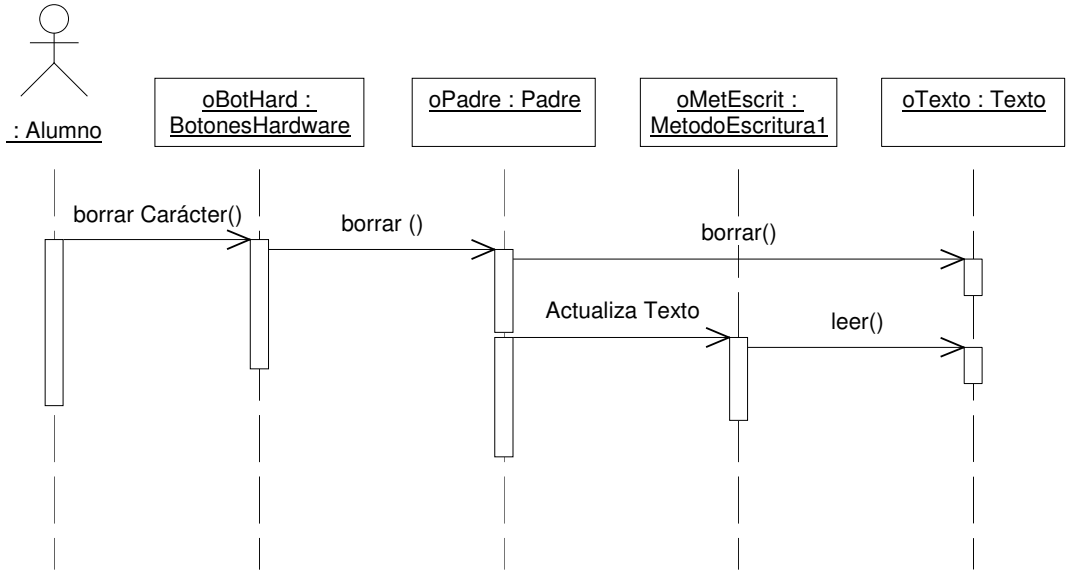


Figura 9.10 Diagrama de secuencia Borra Carácter

9.4.3 Diagrama de Secuencia Reproducir el Texto

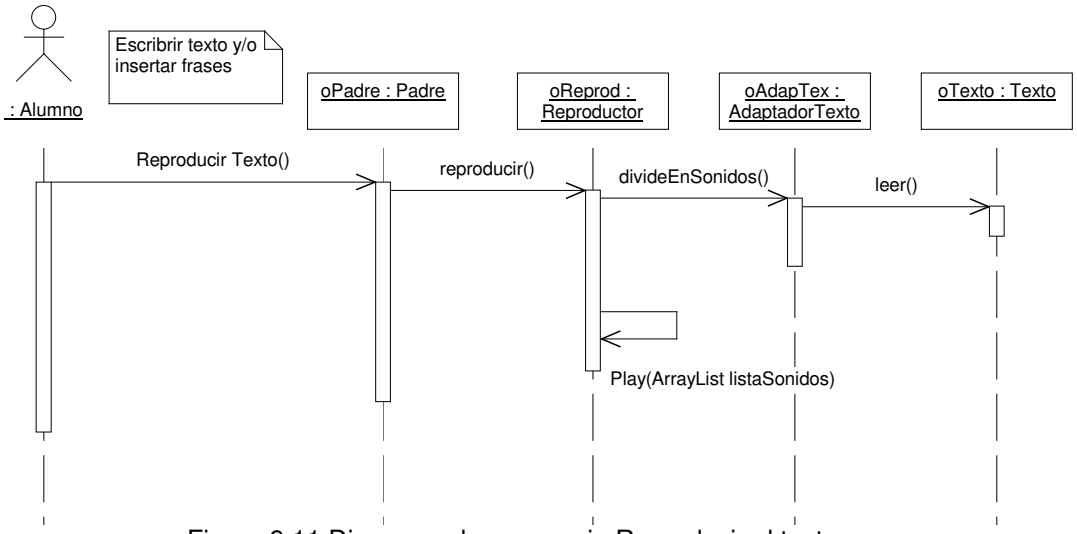


Figura 9.11 Diagrama de secuencia Reproducir el textor

9.4.4 Diagrama de Secuencia Insertar Frase

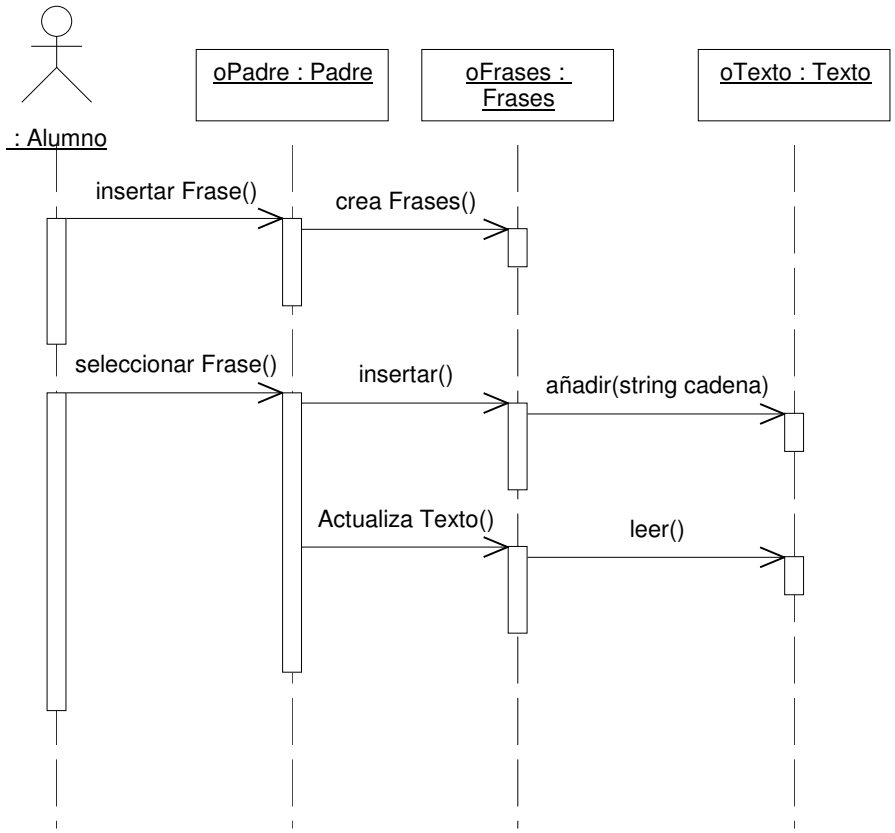


Figura 9.12 Diagrama de secuencia insertar frase

9.4.5 Diagrama de Secuencia Borrar Frase

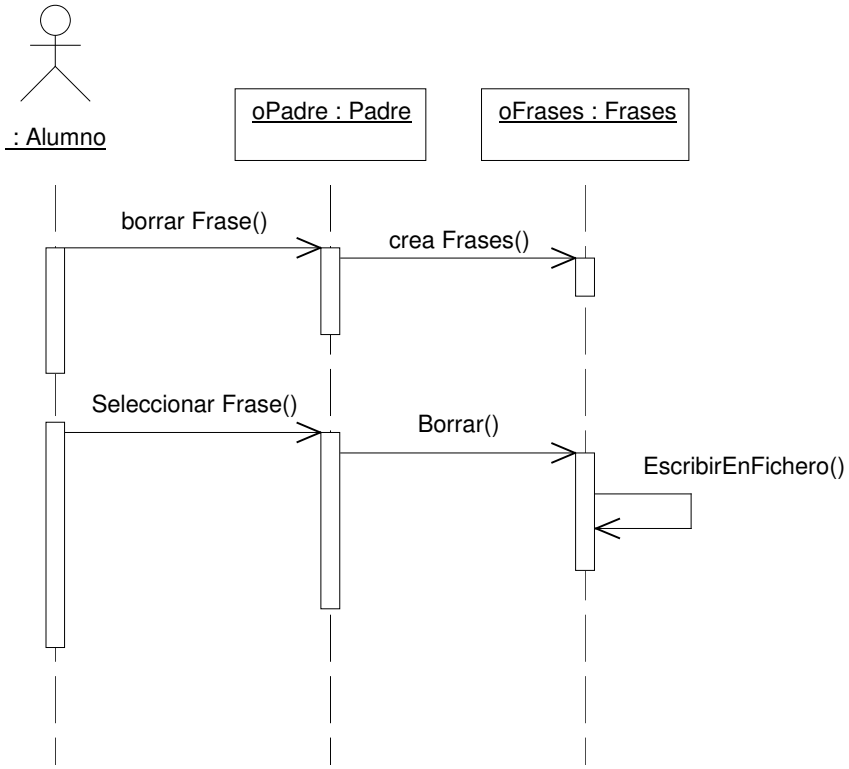


Figura 9.13 Diagrama de secuencia Borrar Fraser

9.4.6 Diagrama de Secuencia Guardar Frases

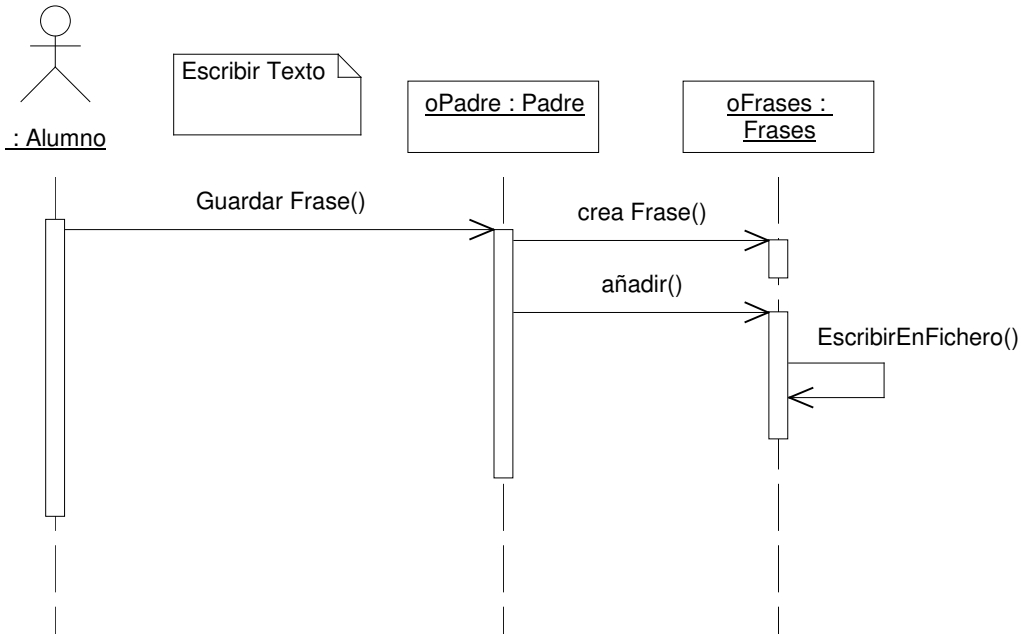


Figura 9.14 Diagrama de secuencia Guardar Frase

9.4.7 Diagrama de Secuencia Consultar Ayuda

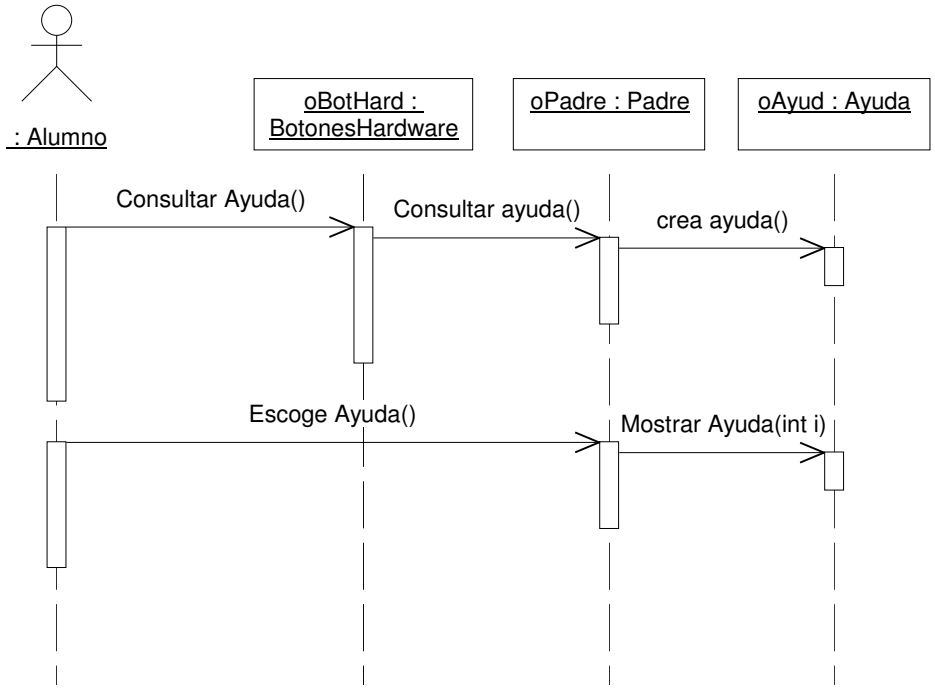


Figura 9.15 Diagrama de secuencia Consultar ayuda

9.4.8 Diagrama de Secuencia Modificar Configuración

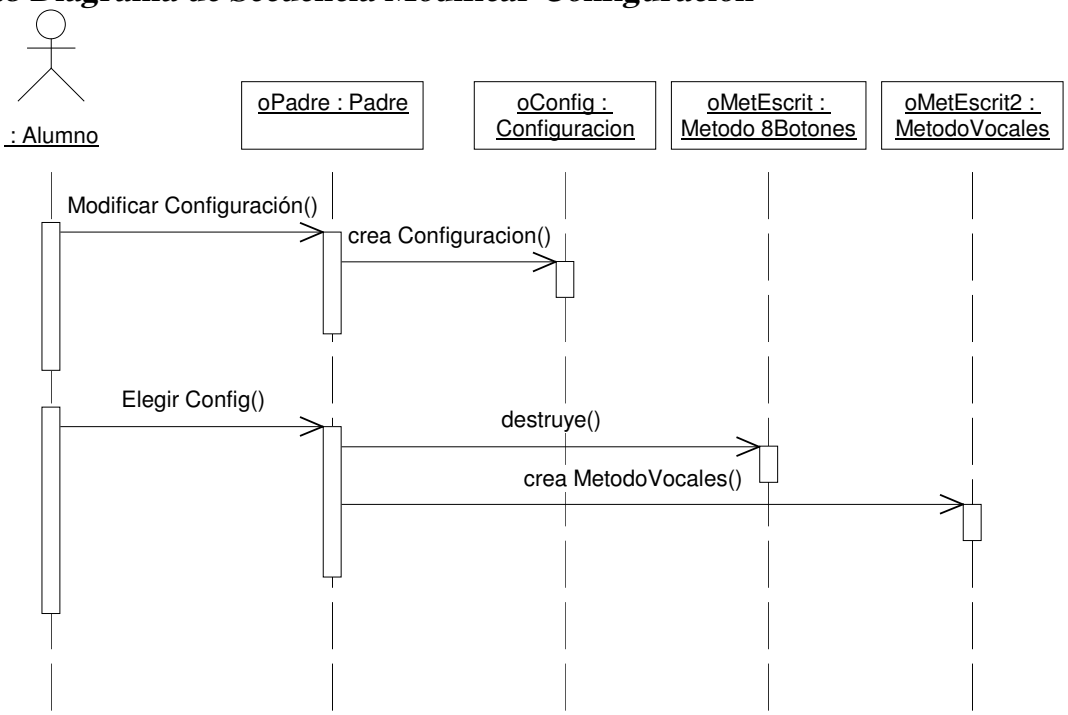


Figura 9.16 Diagrama de secuencia Modificar Configuracion

9.4.9 Diagrama de Secuencia Editar Texto

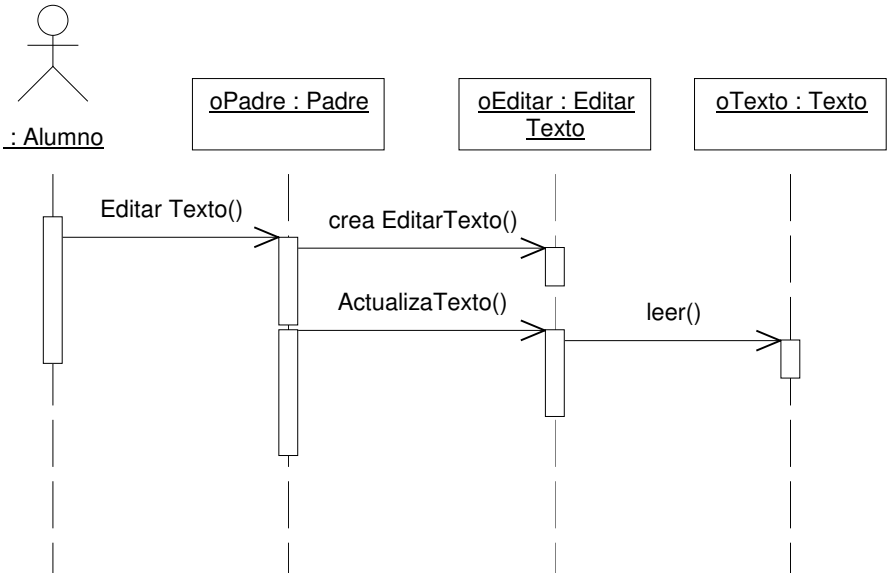


Figura 9.17 Diagrama de secuencia Editar Texto

Capítulo 10. IMPLEMENTACIÓN

Durante la fase de Implementación del Sistema las clases de objetos y las relaciones desarrolladas durante el diseño se traducen a un lenguaje de programación concreto.

10.1 El entorno de programación y el lenguaje

Ha habido una evolución a lo largo del desarrollo del proyecto en cuanto al lenguaje de programación utilizado y las herramientas de desarrollo empleadas.

Empezamos el proyecto centrándonos en el entorno de programación gratuito de Microsoft Embedded Visual Tools 3.0 con las SDK de PocketPc 2002, y con el lenguaje Embedded Visual Basic. Para tal fin recibimos ambos un curso de formación sobre Visual Basic 6.0 como base para poder realizar el proyecto con garantías. Sin embargo, a través del prototipo nos dimos cuenta de las grandes dificultades con las que nos enfrentábamos.

El Embedded Visual Basic es un entorno de programación cerrado en las PDA´s y tremendamente limitado. Se crea un código interpretado por el intérprete de Visual Basic, que es muy lento.

Al mismo tiempo, por sus grandes diferencias con respecto a Visual Basic 6.0 teníamos muchas dificultades en hacer tareas relativamente sencillas.

Eso nos llevó a un cambio crucial en el entorno de desarrollo. Nos cambiamos a Visual Studio .NET 2003, usando como lenguaje de programación C#.

Los motivos del cambio de lenguaje fueron varios, sobre todo la curiosidad de aprender nuevos lenguajes de programación, y de profundizar en uno de los más comentados y con mayor capacidad de crecimiento. Además queríamos empezar a conocer un lenguaje sin vicios adquiridos de versiones anteriores.

Además se programa prácticamente igual que para las aplicaciones de equipos de sobremesa, y nos podría servir para aprender a pogramar los dos tipos de máquinas.

La aplicación habría tenido un comportamiento prácticamente idéntico y nos hubiéramos encontrado con las mismas dificultades y ventajas de emplear Visual Basic .NET en vez de C#, que son los dos lenguajes disponibles actualmente para programar en dispositivos móviles. De echo, como ya se ha comentado, da igual en cuál de los lenguajes se programe en .NET, ya que todos se traducirán al IL, que es el bytecode que entiende el CLR (Common Language Runtime) de .NET. Por cierto, con una velocidad de ejecución muchísimo más alta que en Embedded Visual Basic, gracias a diversas técnicas que usa el CLR, como que una vez que el código está "traducido" no lo vuelve a compilar en tiempo de ejecución, etc.

Y las librerías de clases, que son el .NET COMPACT FRAMEWORK son usadas indistintamente por los dos lenguajes.
Sin embargo, consideramos que el Visual Basic .NET es una adaptación de otro lenguaje totalmente distinto, mientras que el C# ha sido diseñado recientemente con todas las necesidades en mente, y las posibilidades de copiar lo que a la gente le gusta de todos los lenguajes, y desde cero. Opinión, por supuesto, muy discutible. Tiene gran parecido con java.

Aunque se parece el Visual Basic 6.0 y el Visual Basic .NET, los cambios para adaptarlo al .NET lo han llevado a algo radicalmente diferente, por tener que incluir herencia, polimorfismo y otras características que debe llevar cualquier lenguaje que quiera ser compatible con .NET, y de los que el Visual Studio 6.0 carecían.

En cuanto a la diferencia entre el entorno de desarrollo del Visual Studio .NET con el Embedded Visual Tools, es abismal. En la versión Embedded nos encontramos tremendamente limitados. Ahora que se puede ver desde la distancia, tenemos la sensación de ser un entorno "de juguete".

Más ventajas, el Embedded está centrado en dispositivos concretos, que tienen que ser compilados especialmente para ellos. Sin embargo el .NET Compact Framework ofrece una compatibilidad nunca vista en este tipo de dispositivos. Es un subconjunto del .NET Compact Framework (a partir de aquí nos referiremos a él como .NET CF), y como tal, el exe de una aplicación hecha para él funciona en un equipo de sobremesa "casi" con la misma facilidad que otra aplicación normal.

La forma de trabajo es común para todo el Visual Studio, así que el desarrollador tiene más facilidad para adaptarse a la herramienta.

También nos hemos encontrado con algunas desventajas. El .NET CF es un subconjunto del .NET Framework, y como tal, tiene numerosos métodos ausentes, y otros métodos con muchas menos sobrecargas sobre los operadores.

10.2 Uso de las bases de datos desde PDA

Otro de nuestros grandes problemas se produjo con la base de datos. Desde los antiguos tiempos del Windows CE 2.0, se ha tenido un formato de bases de datos basadas en Pocket Access, basado en una versión reducida de la versión de Access para el escritorio. Estas bases de datos se pueden sincronizar automáticamente con el ActiveSync, el programa que distribuye Microsoft para comunicarse y sincronizar agenda y ordenador.

Cuando estábamos pasando la base de datos del PC de escritorio a la PDA, vimos que con los ficheros de Access no había posibilidad directa de comunicación, y algo que usando la sincronización del ActiveSync nos hubiera llevado minutos, con la forma de trabajo recomendada nos llevó muchísimos quebraderos de cabeza.

Para nuestra aplicación hemos usado una base de datos en SQL Server CE 2000.
Necesitábamos unas necesidades muy especiales: era útil para nuestra aplicación usar un conjunto de datos muy simples, sólo hemos usado una tabla “diccionario” con los atributos

“palabra” varchar2
“popularidad”, int

En palabra están todas las palabras que nos interesa que el cliente pueda distinguir, mientras que en popularidad tenemos la frecuencia con la que se usa.
Concretamente teníamos que manejar una sola tabla muy simple, pero con una característica extrema: Necesitábamos un tamaño de unos 80000 registros.

Al principio pensamos en hacer uso de un array para introducir esa tabla. Teníamos el inconveniente de que era un tamaño demasiado elevado como para pensar en ponerlo en tiempo de ejecución en la exigua memoria de una PDA.

Otra de las alternativas tomadas se basaba en usar un fichero XML para poder manejar la información necesaria.
Ese archivo XML ocupaba cerca de 10 Mb,y es una cifra descomunal para una PDA.
Además la velocidad para buscar un registro hacía totalmente inviable este método. Hay que ver que los dispositivos móviles tienen unas capacidades hardware y de velocidad mucho más limitada que un ordenador de sobremesa.
El archivo XML era demasiado lento para poder usarse por sí sólo para hacer búsquedas, entonces probamos a introducirlo en memoria aprovechando las facilidades de la clase DataSet de manejar XML y de usar restricciones.
Esta solución también fallaba, por agotar totalmente los recursos de la PDA en la carga, tanto de rapidez (decenas de minutos intentando cargar el archivo XML) como de memoria (acababa con toda y colgaba la aplicación)

Entonces intentamos usar una base de datos.

En el prototipo usamos ADOCE 3.0 para acceder a una base de datos, y le vimos bastante limitaciones en las consultas que podía generar. Además Microsoft ha desarrollado una base de datos relaccional que toma como punto de referencia para el manejo de las bases de datos en los dispositivos móviles.
Se llama SQL Server CE 2000, y tiene características muy interesantes como hemos explicado.
A nosotros lo que más nos ha interesado es por el tamaño y la rapidez. El motor de la BBDD ocupa sólo 1Mb de memoria, y la base de datos en el formato de SQL Server CE no llegaba a los 3 megas.
Además podía hacer las búsquedas que necesitábamos en unos pocos segundos.
Esta base de datos está pensada para dispositivos móviles. Y está muy bien capacitada para hacer sincronizaciones o actualizaciones de datos con una base de datos SQL server, de un equipo de escritorio.

Tuvimos problemas para poder poblar la base de datos con lo que necesitábamos. Teníamos los datos en una BBDD Access, y no había forma de grabarla directamente en .sdf (tipo de fichero usado en SQL Server CE)
Para poder hacer la transferencia de datos, usamos dos aplicaciones auxiliares, una en el ordenador de sobremesa que accedía a la BBDD Access, y la cargaba en un DataSet, para a continuación guardarla en un archivo XML, y otra en la PDA que recogía el archivo XML, lo sincronizaba con un DataSet en la PDA, para a continuación poblarla tabla que teníamos en el SQL CE Server.
De nuevo nos encontramos con el problema de la memoria en el dispositivo, que no nos permitía hacerlo de una sola vez.

Al final usamos un programa externo llamado Remote SQL Server, que permite manejar BBDD SQL Server CE desde el ordenador de sobremesa, y sincronizarlas con tablas realizadas en Access. Eso nos permitió solucionar nuestras necesidades.

10.3 Diferencias de programación entre PDAs y ordenadores de escritorio

En cuanto a la diferencia entre programar para un dispositivo móvil comparado con uno de sobremesa, sobre todo se encuentra la obsesión por la memoria. Como ejemplo, por un fallo que teníamos en las primeras versiones de la aplicación nos encontramos que nuestra aplicación conseguía dejar a la PDA sin memoria, gracias a un deficiente diseño de la generación de formularios. Cuando arreglamos ese error pasó de consumir toda la memoria a tener un tamaño diminuto. Este fallo hubiera pasado desapercibido en un equipo de sobremesa por sus recursos prácticamente inagotables.

Nos hemos encontrado con el grandísimo handicap de que en CF no existen los eventos MouseUp, MouseClick o MouseDown en ningún control que no sea un botón, con lo que tuvimos que buscar formas imaginativas de hacer sucedáneos de esas funciones para poder usarlas.

Algo tan fácil como poner un control con un dibujo y que responda a eventos también ha sido un gran problema.

También hay diferencias en el entorno de desarrollo ya sea para PDAs o para aplicaciones para ordenadores de sobremesa. Aunque es casi igual, cuando estás programando para dispositivos móviles, hay un montón de ayudas que no están disponibles. Volvemos a poner otro ejemplo que nos ha pasado. Si quieres un DataGrid con los datos enlazados con un DataSet, que está a su vez rellenado con una tabla de una base de datos. Esto es relativamente común para mostrar datos de una Base de datos en pantalla.
Pues bien, si se desea hacer para un equipo de sobremesa, lleva unos pocos clicks, algún asistente que te ayuda a realizarlo y 1 o 2 líneas de código. Sin embargo lo mismo en un dispositivo móvil hay que realizarlo totalmente a mano, escribiendo todo el código, con la posibilidad de equivocarse en muchos más sitios, y llevando muchísimo más tiempo.

También tiene grandes limitaciones en cuanto a periféricos.

Para usar el sonido no se puede usar ninguna clase del CF, sino que hay que acceder directamente al API de Windows CE. Esto nos pasó para poder reproducir el sonido. También conocemos la gran dificultad que tienen estos dispositivos para periféricos que no se han considerado útiles en dispositivos móviles, como impresoras. Hay aplicaciones comerciales que se pueden usar para paliar estas deficiencias.

Sin embargo, por ser dispositivos portátiles, se puede usar con facilidad redes inalámbricas, tanto en formato wifi como IrDA, con clases específicas de las que carece el .NET Framework.

10.4 Software Usado:

- Microsoft Windows 2000 Professional Service Pack 4
- Microsoft Visual Studio .NET 2003
- Microsoft Embedded Visual Tools 3.0
- Microsoft SQL Server CE 2000
- SDK de PocketPc 2002
- Adobe PhotoShop CS
- Microsoft Word 2000
- Microsoft Access 2000
- Remote SQL Server 1.1.3
- Rational Rose Demo Ver.4
- Borland Together 4
- Nero Wave Editor 2.0.0.29

10.5 Software auxiliar desarrollado para la aplicación

- **Prototipo**: Primera versión donde aplicamos los métodos de barrido, de las diagonales, y de la base de datos. Usado para comprobar la factibilidad de algunas ideas y para recibir sugerencias sobre métodos de escritura.
- **Generador Diccionario**: Aplicación para PC de sobremesa que carga un archivo de texto, y devuelve un archivo en formato Access con todas las palabras presentes en el texto, y cada una de ellas con un valor de "popularidad" o frecuencia con la que ha aparecido en él. Usada para poblar la BBDD con las palabras más frecuentes.
- **GeneradorXML**: Aplicación para PC de sobremesa que carga una base de datos Access y la convierte en un fichero XML.
- **CargaXML** Aplicación para PocketPc, que carga un fichero XML e introduce los metadatos en una BBDD SQL CE Server.

10.6 Hardware Empleado:

- Equipo principal de desarrollo
  - S.O. Microsoft Windows 2000 Professional Service Pack 4
  - Pentium III 350Mhz
  - 192 Mb Ram
  - HD 20Gb
- Equipo secundario de desarrollo
  - S.O. Microsoft Windows 2000 Professional Service Pack 4
  - Pentium Celeron 450Mhz
  - 192 Mb Ram
  - HD 40Gb
- PDA de desarrollo: HP Jornada 565
  - S.O. Pocket Pc 2002
  - Procesador StrongArm 206Mhz
  - 32Mb Ram
  - 32Mb Rom
- PDA de implementación: Acer N-10

  - S.O. Windows Mobile 2003
  - Procesador Intel Xcale 266 Mhz
  - 64Mb Ram
  - 32Mb Rom
- Otros:
  - Tarjeta de memoria Compact Flash 256Mb
  - Tarjeta de memoria Secure Digital 512Mb
  - Cámara digital HP 3660 3.1MegaPixels.
  - Impresora Epson Stylus Photo 700

Capítulo 11. PRUEBAS

Descripción	Comprobación del botón hardware Borrar
Acción Realizada	Pulsación del botón situado más a la izquierda del dispositivo móvil
Resultado Esperado	Borrar un carácter
Resultado Obtenido	Carácter borrado
Observaciones	Correcto

Figura 11.1 Comprobación del botón hardware Borrar

Descripción	Comprobación del botón hardware Reproducir
Acción Realizada	Pulsación del botón situado a la derecha del botón de borrar (segundo por la izquierda)
Resultado Esperado	Sonido del texto escrito
Resultado Obtenido	Reproducción del texto
Observaciones	Correcto

Figura 11.2 Comprobación del botón hardware Reproducir

Descripción	Comprobación del botón hardware Ayuda
Acción Realizada	Pulsación del tercer botón del dispositivo móvil
Resultado Esperado	Mostrar ayuda
Resultado Obtenido	Ayuda mostrada
Observaciones	Correcto

Figura 11.3 Comprobación del botón hardware Ayuda

Descripción	Comprobación de Insertar Frase
Acción Realizada	Pulsación del botón insertar en el formulario Frases, después de haber seleccionado la frase con los botones de desplazamiento
Resultado Esperado	Que el texto de la frase seleccionada se introduzca en la caja de texto
Resultado Obtenido	Texto introducido
Observaciones	Correcto

Figura 11.4 Comprobación de Insertar Frase

Descripción	Comprobación de Borrar Frase
Acción Realizada	Pulsación del botón borrar en el formulario Frases, después de haber seleccionado la frase con los botones de desplazamiento
Resultado Esperado	Que la frase seleccionada sea borrada de la lista de frases
Resultado Obtenido	Frase borrada
Observaciones	Correcto

Figura 11.5 Comprobación de Borrar Frase

Descripción	Comprobación de Guardar Frase
Acción Realizada	Escritura de una frase, y pulsación del botón añadir en Frases después de haber seleccionado el lugar que ocupará cuando se añada
Resultado Esperado	Frase guardada en lugar seleccionado
Resultado Obtenido	Frase guardada en lugar seleccionado
Observaciones	Correcto

Figura 11.6 Comprobación de Guardar Frase

Descripción	Comprobación del menú sugerencias de palabaras en el método de escritura 8 botones
Acción Realizada	Pulsación sobre la caja de texto superior tras hacer escrito una letra
Resultado Esperado	Lista de palabras que comienzan por esa letra, comenzando por aquellas palabras más populares
Resultado Obtenido	Lista de palabras que comienzan por esa letra
Observaciones	Correcto

Figura 11.7 Comprobación del menú sugerencias de palabaras en el método de escritura 8 botones

Descripción	Comprobación del menú sugerencias de palabaras en el método vocales
Acción Realizada	Pulsación sobre la caja de texto superior tras hacer escrito una letra
Resultado Esperado	Lista de palabras que comienzan por esa letra, comenzando por aquellas palabras más populares
Resultado Obtenido	Lista de palabras que comienzan por esa letra
Observaciones	Correcto

Figura 11.8 Comprobación del menú sugerencias de palabaras en el método vocales

Descripción	Comprobación del sonido
Acción Realizada	Introducción de un texto y pulsación del botón play en el menú opciones opciones
Resultado Esperado	Reproducir sonido
Resultado Obtenido	Sonido reproducido
Observaciones	Correcto

Figura 11.9 Comprobación del sonido

Descripción	Comprobación del botón volver en el menú opciones
Acción Realizada	Pulsación del botón volver en el menú opciones
Resultado Esperado	Volver al método de escritura desde el que fue llamado
Resultado Obtenido	Error al no actualizar la caja de texto
Observaciones	Incorrecto
Modo de actuación	Modificado y corregido en la versión 0.5B

Figura 11.10 Comprobación del botón volver en el menú opciones

Descripción	Comprobación del botón volver en el menú Frases
Acción Realizada	Pulsación del botón volver en el menú Frases después de haber insertado una frase
Resultado Esperado	Vuelta al menú opciones y caja de texto actualizada
Resultado Obtenido	Vuelta al menú opciones y caja de texto actualizada
Observaciones	Correcto

Figura 11.11 Comprobación del botón volver en el menú Frases

Descripción	Comprobación del botón salir de la aplicación
Acción Realizada	Pulsación del botón salir en el formulario opciones
Resultado Esperado	Salir de la Aplicación
Resultado Obtenido	Vuelta al formulario de escritura
Observaciones	Error
Modo de actuación	Modificado y corregido en la versión 1.0

Figura 11.12 Comprobación del botón salir de la aplicación

Descripción	Comprobación de borrar todo el texto
Acción Realizada	Pulsación del botón borrar todo en el formulario editar Texto
Resultado Esperado	Borrar todo el texto
Resultado Obtenido	Texto Borrado
Observaciones	Correcto

Figura 11.13 Comprobación de borrar todo el texto

Descripción	Comprobación de los controles de desplazamiento en editar texto
Acción Realizada	Pulsación del botón arriba y del botón abajo tras escribir al texto como para que sean activados
Resultado Esperado	Desplazamiento por el texto
Resultado Obtenido	Desplazamiento por el texto
Observaciones	Correcto

Figura 11.14 Comprobación de los controles de desplazamiento en editar texto

Descripción	Comprobación de los controles de desplazamiento en el menú Frases
Acción Realizada	Pulsación del botón arriba y del botón abajo en el formulario Frases
Resultado Esperado	Frase seleccionada cambie en función del botón presionado
Resultado Obtenido	Frase seleccionada cambia en función del botón presionado
Observaciones	Correcto

Figura 11.15 Comprobación de los controles de desplazamiento en el menú Frases

Parte IV Manual de Usuario

Capítulo 12. MANUAL DE USUARIO

12.1 Descripción de la aplicación

Este programa le permitirá llevar a cabo las siguientes opciones:

- Escribir textos en dos métodos de escritura distintos.
- Borrar el texto
- Reproducirlo
- Editarlo
- Disponer de frases para agilizar la comunicación de manera que puedes:
    - Insertar la frase en el texto escrito hasta el momento
    - Añadir una frase nueva a las que ya había para su uso
    - Borrar una frase

A continuación se detalla el uso de la aplicación , para un manejo correcto.

12.2 Pantalla de bienvenida

Esta es la pantalla que se muestra nada más arrancar la aplicación.

En ella se muestra una pequeña ayuda de los botones hardware además de otros datos de interés.

Esta pantalla sólo está unos segundos antes de mostrarse la pantalla del método de escritura de inicio.

Es importante destacar el uso de los botones del Pocket PC, puesto que en ellos se han integrado las funciones más básicas y comunes de la aplicación. La imagen inferior de la presentación indica cuál es el uso de cada uno de los botones.

Figura 12.1 y 12.2, Presentación y Pantalla de escritura

12.3 Pantalla de escritura

El uso de la aplicación se basa en esta pantalla (método de escritura) y en el uso de todos los botones que aquí se muestran y que se detallan más adelante.

Hay dos métodos de escritura similares y su manejo es parecido. A continuación describimos las funciones y el empleo de ambos

.

Texto introducido

En esta parte de la pantalla se van mostrando los últimos caracteres que se van escribiendo.

Figura 12.3 Pantalla de Introducción de texto

Además tiene otra importante función: Si se pulsa encima, saldrá una lista de palabras coincidentes con la última palabra (subcadena) que se haya introducido, a modo de texto predictivo de manera que si la palabra que deseas escribir se encuentra entre las palabras sugeridas y pulsas sobre ella, la palabra será colocada en esta parte de la pantalla, pero éste caso se detalla un poco más adelante-

Botones de las letras

Estos botones serán empleados como teclado para poder escribir el texto deseado. Deberá pulsar sobre el botón donde se encuentra la letra que usted quiera escribir .

Figura 12.4 Botones de letras

Botón Espacio

Como su propio nombre indica, introducirá un espacio en el texto introducido en el momento que sea pulsado.

Figura 12.5 Botón de espacio

Botón de los números

Al pulsar dicho botón, saldrá una nueva pantalla con los números del 0 al 9 y con la coma decimal, para que usted pueda introducir números en los textos si así lo desea.

Figura 12.6 Botón de números

Botón Menú

Despliega el menú principal de la aplicación, donde podrá encontrar las funcionalidades que no se encuentran en la página principal. Podrá pulsarlo siempre que usted lo desee.

Figura 12.7 Botón de menú

Botones Hardware

Los botones que trae la PDA. Sirven para hacer las acciones más habituales en estos dispositivos.

De izquierda a derecha, su función es:

- Borrar un carácter. Útil para corregir las equivocaciones cometidas.
- Reproducir. Reproducirá el texto que se haya escrito hasta ese momento a través del altavoz
- Ayuda: Mostrará un menú de ayuda para que pueda consultar cualquier problema que tenga con la aplicación.
- El cuarto botón arranca la aplicación. Si no se muestra la aplicación en la pantalla, este botón la mostrará.

Los botones de la PDA podrán ser pulsados en cualquier instante de la ejecución de la aplicación.

Figura 12.8 Botones Hardware

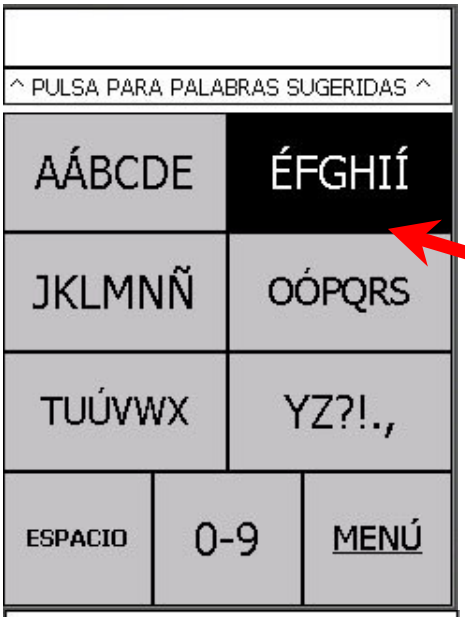
A continuación se describe cómo comenzar a escribir a través de un ejemplo de escritura.

12.4 Métodos de Escritura

Hay dos métodos de escritura muy similares. El sistema de escritura es el mismo, la diferencia se encuentra en que un método de escritura tiene las vocales en botones aislados para disminuir el número de pulsaciones.

12.4.1 Método de Escritura 8 Botones

Supongamos que se desea escribir la palabra hola. Tendremos como situación de inicio la parte superior de la pantalla en blanco, ya que todavía no hemos introducido ninguna letra.



Para escribir una letra hay que apretar apretando la tecla que contenga la letra elegida, en nuestro caso, si queremos escribir la palabra *hola* tendremos que pulsar en el botón que contenga la *h,* como se muestra aquí

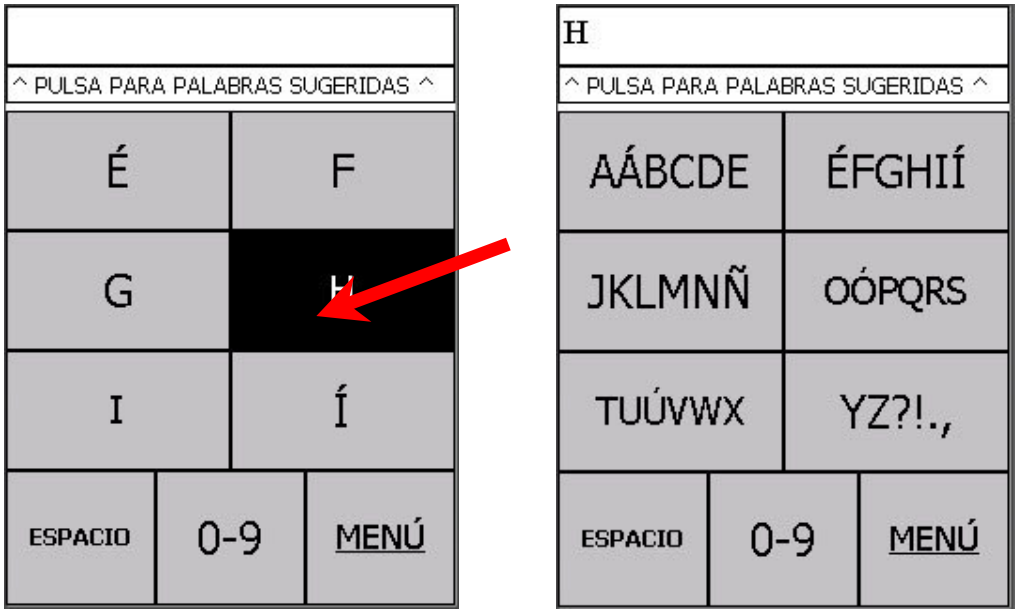
.

Figura 12.9 Ejemplo de 8 Botones, imagen 1

En el momento que se pulsa el botón deseado, todas las letras se redistribuirán en el resto del teclado, de manera que quede una letra en cada tecla.

Como está aislada la *h*, que es la primera letra de nuestra palabra, la siguiente pulsación sobre la tecla esté situada la letra, conseguirá que aparezca la *h* en el texto introducido y aparecerán de nuevo todas las letras del abecedario en los botones correspondientes.

Figura 12.10 Ejemplo de 8 Botones, imagen 2



Figuras 12.11 y 12.12 Ejemplo de 8 Botones, imagen 3 y 4

Ahora habría que hacer lo mismo con el resto de las letras:

- Pulsar en el grupo donde esté situada la letra *o*
- Las letras se redistribuirán y la letra quedará aislada de manera que si la pulsas el texto se añadirá al que hasta el momento está escrito y así sucesivamente.

12.4.2 Método de las vocales

Éste método se puede considerar una optimización del anterior, ya que las vocales están en botones separados de las consonantes de manera que el número de pulsaciones disminuye considerablemente, puesto que en una palabra por cada consonante (aproximadamente hay una vocal).

La siguiente imagen se corresponde con la pantalla inicial de este método de escritura, sin que haya sido pulsada ninguna tecla.

Figura 12.13, Método de las vocales

Veamos como es su uso, utilizando el mismo ejemplo que con el método anterior, es decir la escritura de la palabra hola.

La primera letra que tenemos que escribir es la *h*, por lo tanto tendremos que pulsar en el grupo de letras entre las que se encuentra.

Al igual que en el método anterior, la pulsación provocará que las letras correspondientes a ese botón se redistribuyan, pero esta vez lo harán sólo por los botones de las consonantes puesto que las vocales son fijas en este método de escritura.

Ahora que las letras están aisladas en los botones una segunda pulsación dará lugar a que la letra pulsada sea escrita se introduzca en la zona dedicada al texto escrito hasta el momento

Figuras 12.14 y 12.15. Ejemplo Método vocales Imagen 1 y 2

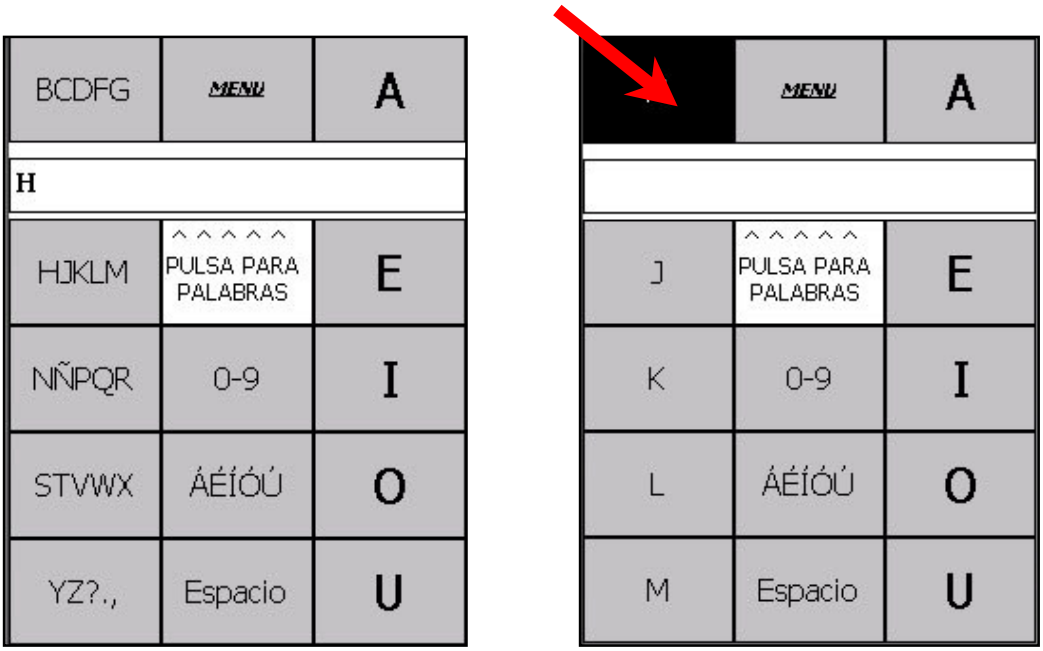


Figura 12.16 y 12.17 Ejemplo Método vocales Imagen 3 y 4

Lo siguiente sería pulsar la letra o y así sucesivamente hasta completar la palabra. La función de los botones Espacio, Menú y 0-9 es la misma que en el método de escritura anterior.

12.4.3 Para introducir números

El botón que se encuentra en ambos métodos con 0-9 escrito en su interior conduce a la misma pantalla.

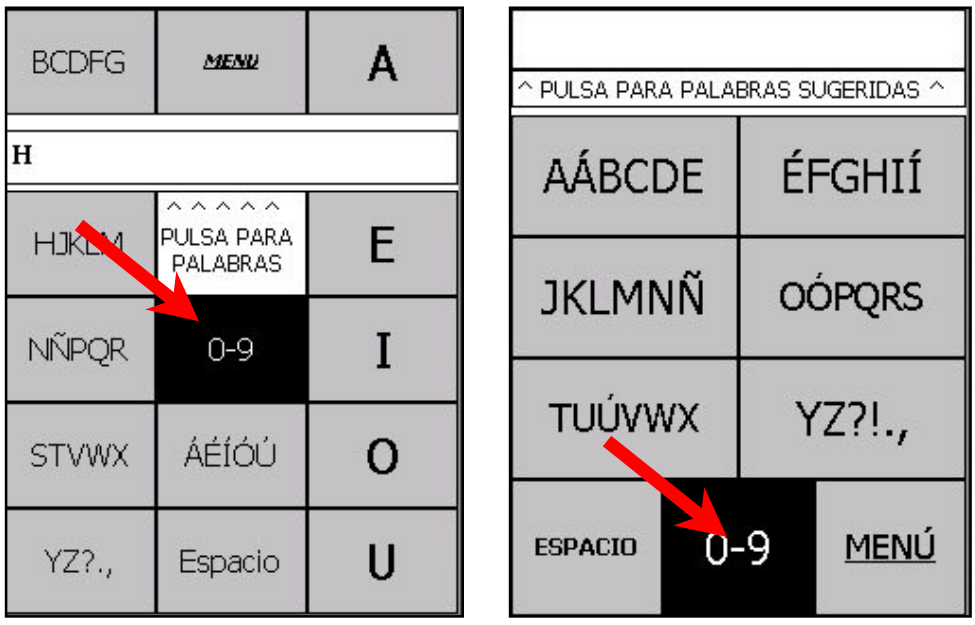
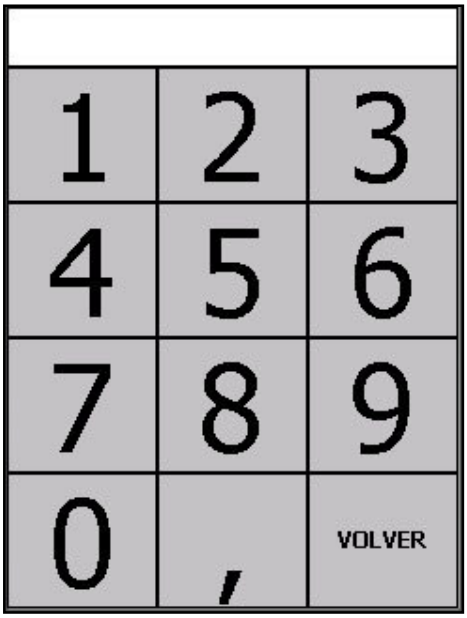
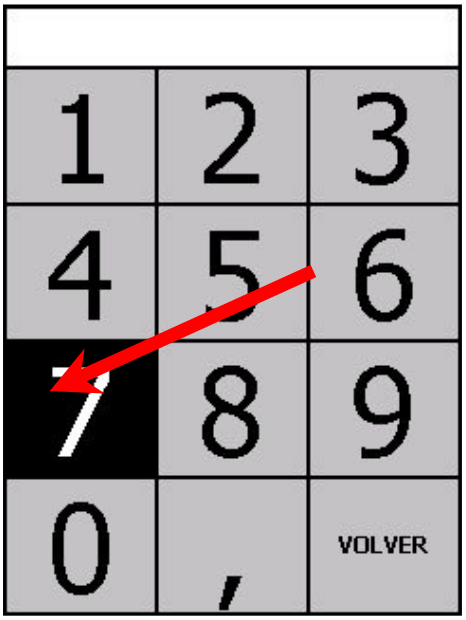


Figura 12.18 y 12.19. Introducción de números

Y el sistema para escribir números es el mismo, sólo que ahora basta con una sola pulsación puesto que hay un único dígito en cada botón.





Figuras 12.20 y 12.21. Ejemplo introducción de números imagen 1 y 2

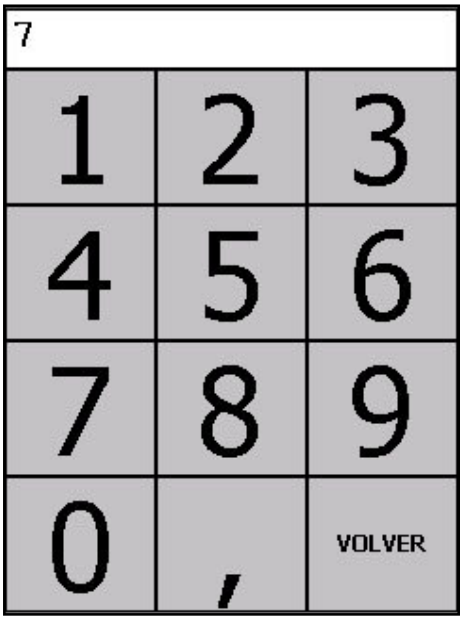


Figura 12.22 Ejemplo números imagen 3

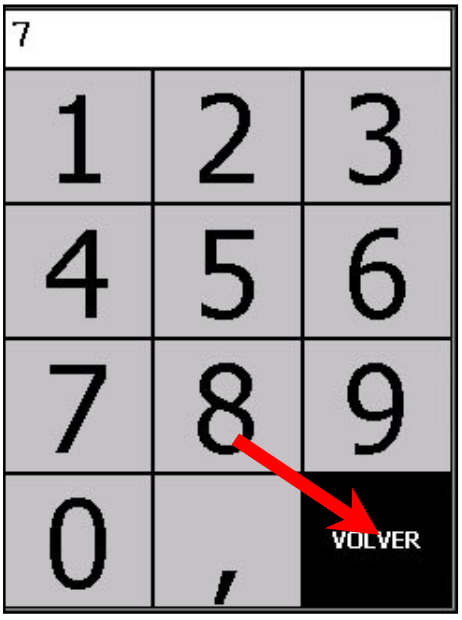


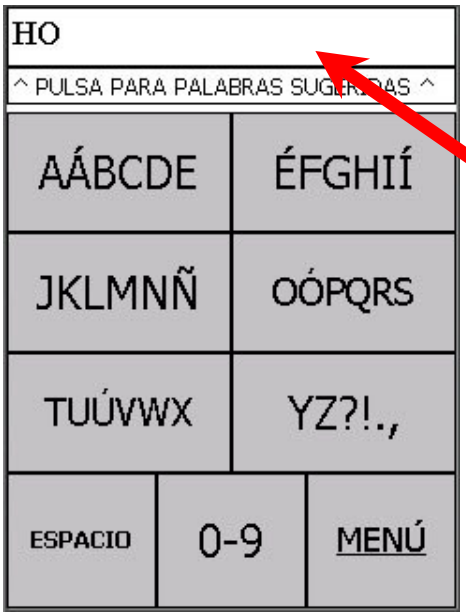
Figura 12.23 Imagen números, volver

Se incluye en esta pantalla la coma, por si se quisiera escribir un número decimal no tener que volver al método de escritura.

Si se pulsa el botón volver nos situaremos en el método de escritura desde el que se esta escribiendo.

12.5 Uso del texto Predictivo

El texto predictivo tiene como cometido reducir el número de pulsaciones que el usuario de la aplicación tenga que realizar, de esta manera, cuando se hayan escrito dos o más letras, se podrán solicitar a la aplicación sugerencias, y si la palabra que nosotros deseamos escribir se encuentra entre ellas, el número de pulsaciones hechas se habrá reducido considerablemente en la mayoría de los casos, puesto que el lenguaje castellano tiene una media de longitud de palabra entorno a 5.



Veamos a continuación un ejemplo del uso con la misma palabra para ver la diferencia. El ejemplo está basado en el primer método de escritura aunque de igual manera se realiza en el otro método.

Supongamos, que al igual que antes queremos escribir la palabra *hola*, hemos introducido la letra *h* y seguidamente la letra *o*, si pulsamos ahora en la parte donde va apareciendo el texto saldrá una pantalla con sugerencias a palabras que coinciden en sus dos primeras letras con *ho.*

Figura 12.24 Texto predictivo

Supongamos, que al igual que antes queremos escribir la palabra *hola*, hemos introducido la letra *h* y seguidamente la letra *o*, si pulsamos ahora en la parte donde va apareciendo el texto saldrá una pantalla con sugerencias a palabras que coinciden en sus dos primeras letras con *ho.*

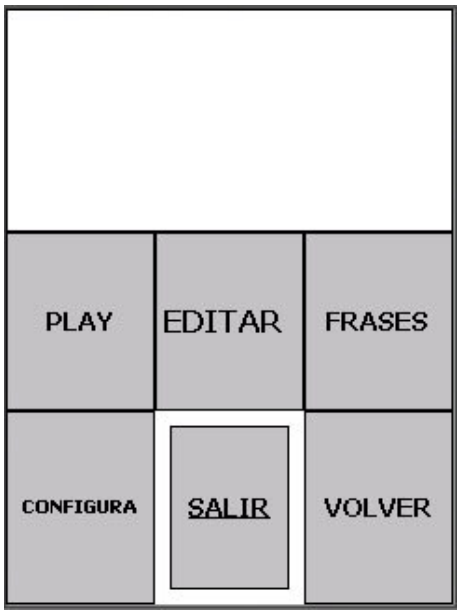
Bastaría con pulsar la palabra elegida para tenerla en la pantalla del texto introducido.

Figura 12.25 Ejemplo Texto Predictivo, imagen 1

El resultado sería éste, y ya podríamos comenzar a escribir la siguiente palabra o por el contrario, si lo que deseamos es reproducir el sonido del texto escrito.

Figura 12.26 Resultado del Texto Predictivo

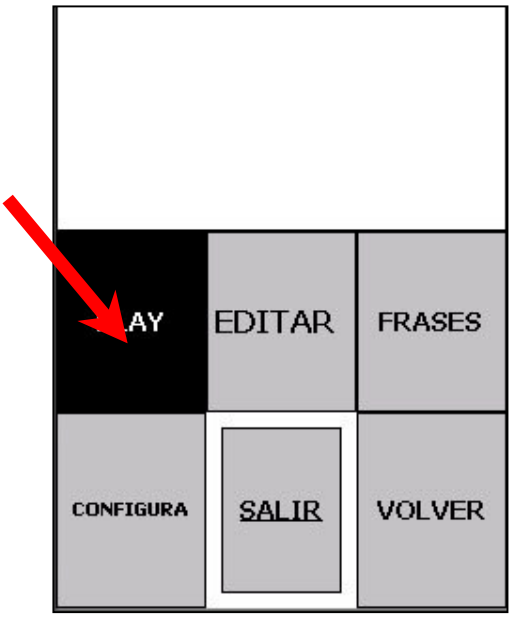
12.6 El Menú de Opciones



En el menú opciones se encuentran todas las posibilidades que ofrece la aplicación con respecto al texto que se ha escrito en alguno de los métodos de escritura, así como el submenú frases que nos permitirá insertar textos ya escritos para agilizar la comunicación.

Figura 12.27 El menú opciones

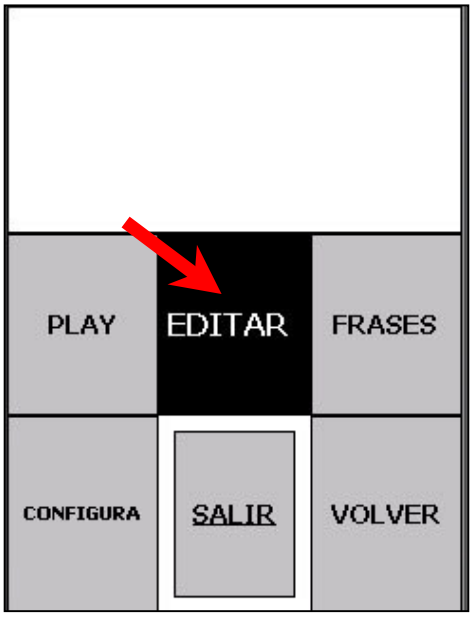
.



El menú opciones está compuesto por una parte superior en la que aparecerá el texto escrito hasta el momento.

Figura 12.28 El menú opciones: Play

Este texto se puede reproducir pulsando el botón *Play*.

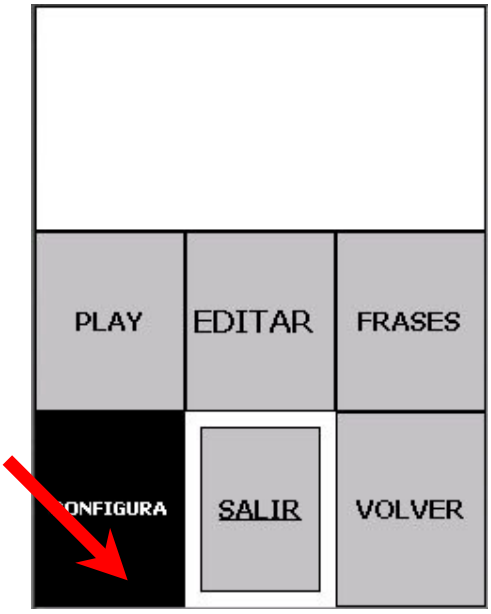


Otra de las opciones que se ofrecen es entrar en el submenú *Editar* que permitirá tratar el texto escrito

Figura 12.28 El menú opciones: Editar

Por otra parte nos encontramos con *Frases*, un submenú que describiremos más adelante que nos permite agilizar notablemente la comunicación debido a frases que se encuentran guardadas en el dispositivo y que se podrán usar para reproducir o editar, borrar las existentes o añadir nuevas.

Figura 12.29 El menú opciones: Frases



Otra de las funcionalidades que tiene el menú opciones es la de *Configurar*, que será el encargado de cambiar de método de escritura si así lo deseas.

Figura 12.30 El menú opciones: Configuración

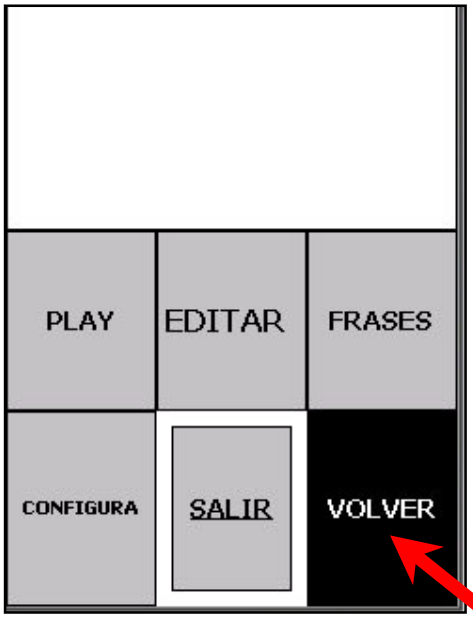


Figura 12.31 El menú opciones: Volver

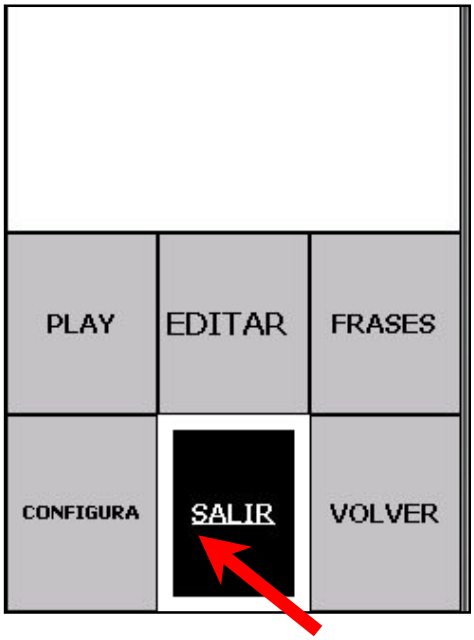
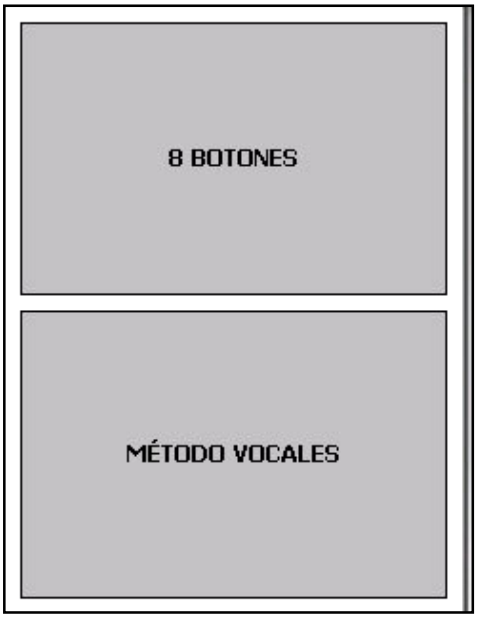


Figura 12.32 El menú opciones: salir

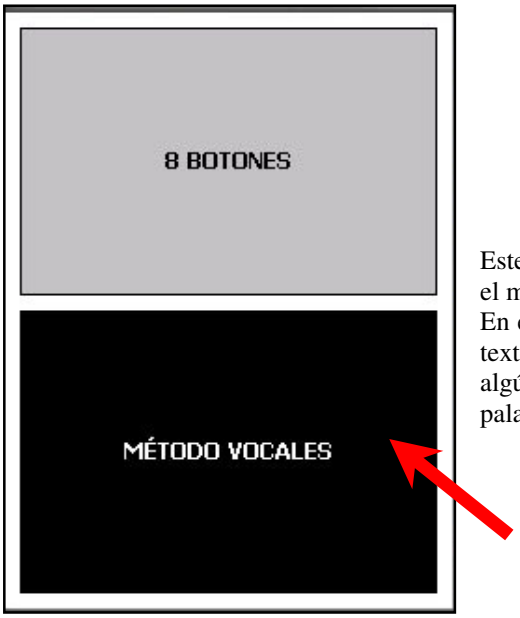
El botón *salir* ,permite abandonar la aplicación y *volver* que te devuelve al método de escritura en el que te encontrabas antes de pulsar en *Menú*

12.7 Cambiar configuración



Configuración te ofrece la posibilidad de cambiar el método de escritura si así lo deseas. Para ello tienes que pulsar en el botón configura, y posteriormente sobre uno de los dos métodos de escritura de manera que cuando éste sea pulsado automáticamente te situará en la forma de escritura deseada.

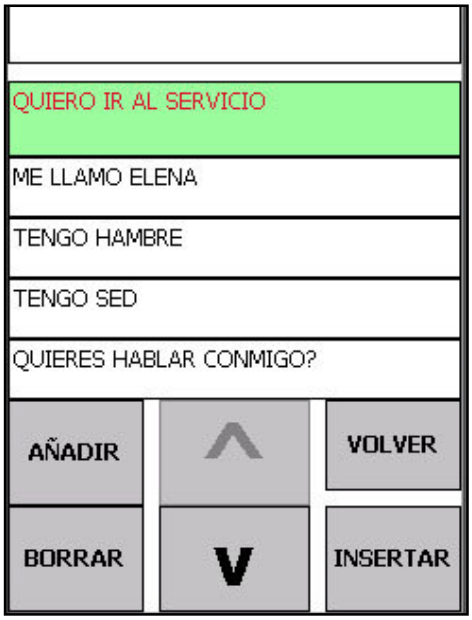
Figura 12.33 El menú configuración



Este es la pantalla que nos vamos a encontrar al pulsar en el menú opciones.
En el caso de que hubiese más métodos de introducción de texto todos se agruparían en este menú, de esta forma, si en algún momento quisieras cambiar tu forma de escribir las palabras podrías hacerlo en cualquier momento.

Figura 12.34 El menú configuración. Ejemplo

12.8 Menú frases

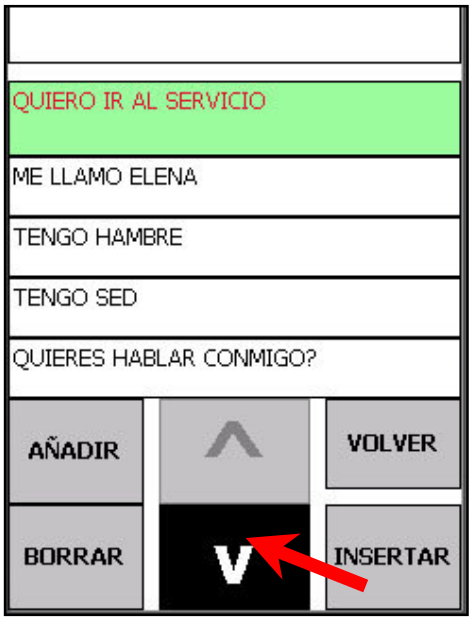


El menú frases nos da la posibilidad de agilizar notablemente la comunicación.

Podemos usar las frases que están almacenadas, o bien añadir frases nuevas o incluso borrarlas.
Para desplazarse por las frases hay dos botones, que permiten escoger la frase deseada para insertar, para borrar, o el sitio donde queremos añadir una nueva frase.

Figura 12.35 El menú Frases

12.8.1 Desplazarte por las frases



Para poder desplazarte por las frases existentes es necesario pulsar en alguno de los botones centrales que te permitirá recorrer las frases hasta situarte en la que desees.
Cuando la frase seleccionada sea la primera el botón de desplazamiento para ver las frases anteriores será desactivado, lo contrario ocurrirá si la frase sobre la que te encuentras es la última, que el botón que quedará desactivado es el de desplazamiento para ver las oraciones siguientes.

Figura 12.36 El menú Frases. Ejemplo de desplazamiento

12.8.2 Insertar una frase

Si quieres en cualquier momento usar alguna de las frases que tienes guardada, bastaría con pulsar en insertar cuando la frase que deseas esté seleccionada (mediante los botones de desplazamiento).
La frase del recuadro verde pasaría a formar parte del texto que hasta el momento se ha escrito.

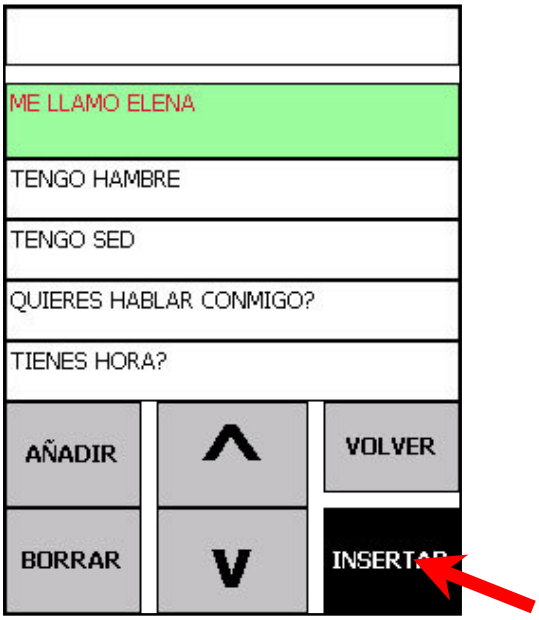


Figura 12.36 Ejemplos de inserción de una frase. Imagen 1

Cuando se ha insertado una frase, la aplicación te sitúa en el menú opciones por si deseas realizar alguna operación sobre dicho texto.

Figura 12.37 Ejemplos de inserción de una frase. Imagen 2

12.8.3 Borrar una frase

Para borrar una frase permanentemente de la lista que almacena el dispositivo hay que seleccionar la frase mediante los botones de desplazamiento y pulsar el botón borrar.

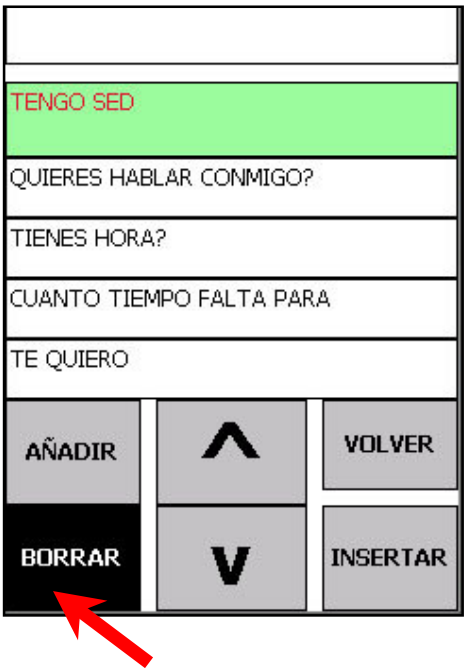
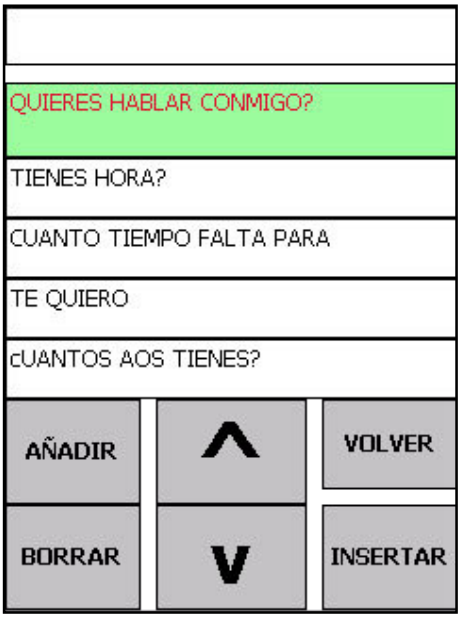


Figura 12.38 Ejemplo de Borrado de una frase. Imagen 1

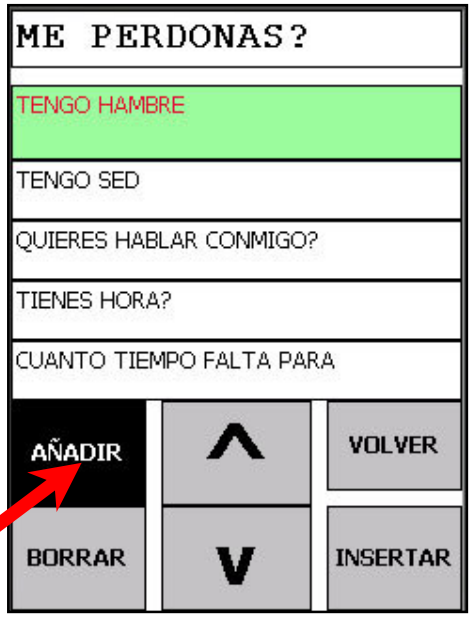


La frase será borrada del sistema, y si se quisiese volver a usar habría que escribirla de nuevo.

Figura 12.39 Ejemplo de Borrado de una frase. Imagen 2

12.8.4 Guardar una frase

Para guardar una frase y poder disponer de ella en cualquier momento, primero hay que escribir la frase en cuestión con alguno de los métodos de escritura, y posteriormente desde la pantalla de frases, seleccionar la posición que ocupará la frase que deseas guardar, y por último pulsar el botón añadir.



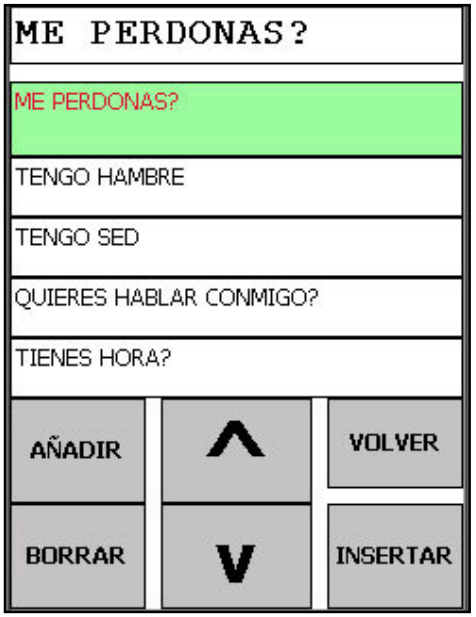
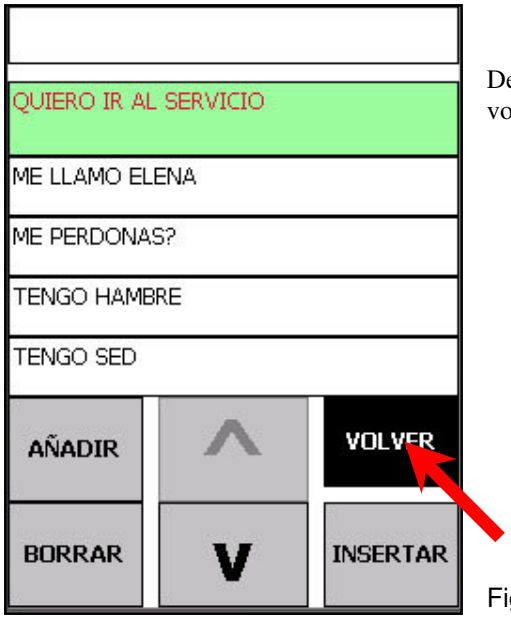


Figura 12.40 y 12.41 Ejemplo de cómo guardar una frase



Desde esta pantalla de frases si se pulsa el botón volver se volvería al menú opciones

Figura 12.42 El botón volver en el menú frases

12.9 Menú Editar Texto

El menú Editar Texto es el encargado de tratar los textos escritos, es decir, permite centralizar todas las funciones que se pueden aplicar a lo que se ha escrito y aparece en la parte superior de la pantalla.

Las funciones son: reproducir el sonido y borrar un carácter, que también se puede realizar desde los botones del dispositivo móvil, borrar todo el texto escrito, y desplazarte por el texto.

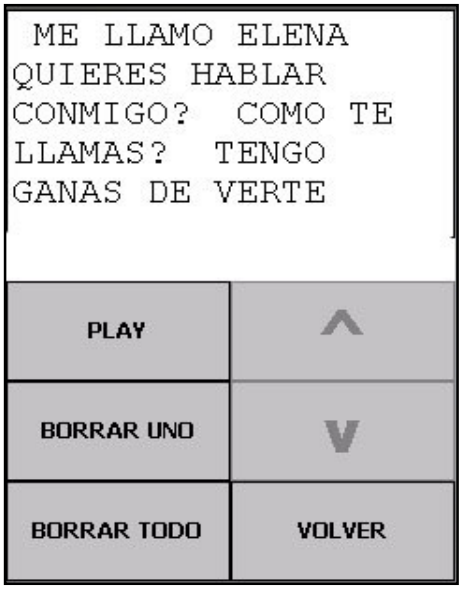


Figura 12. 43 El Menú Editar Texto

12.9.1 Borrar un carácter

Para borrar un carácter hay dos posibilidades:

- Borrarlo a través de los botones que posee el dispositivo
- Borrarlo mediante el botón que se encuentra en editar texto.

Primera forma

Si pulsas el botón señalado podrás borrar un carácter desde cualquier pantalla, ya sea el método de escritura desde donde estés escribiendo, desde opciones, desde editar texto… El carácter borrado será siempre el último escrito.

Figura 12.44 Borrar un carácter, desde un botón hardware

Segunda forma

En este caso, si pulsas el botón marcado el resultado será el mismo que en el caso anterior, el último carácter será borrado.

Figura 12.45 Borrar un carácter, desde opciones

12.9.2 Borrarlo todo el texto

Pulsar el botón borrar todo, lógicamente, tiene como consecuencia el que el texto escrito hasta ese momento se borre de la parte superior de la pantalla.

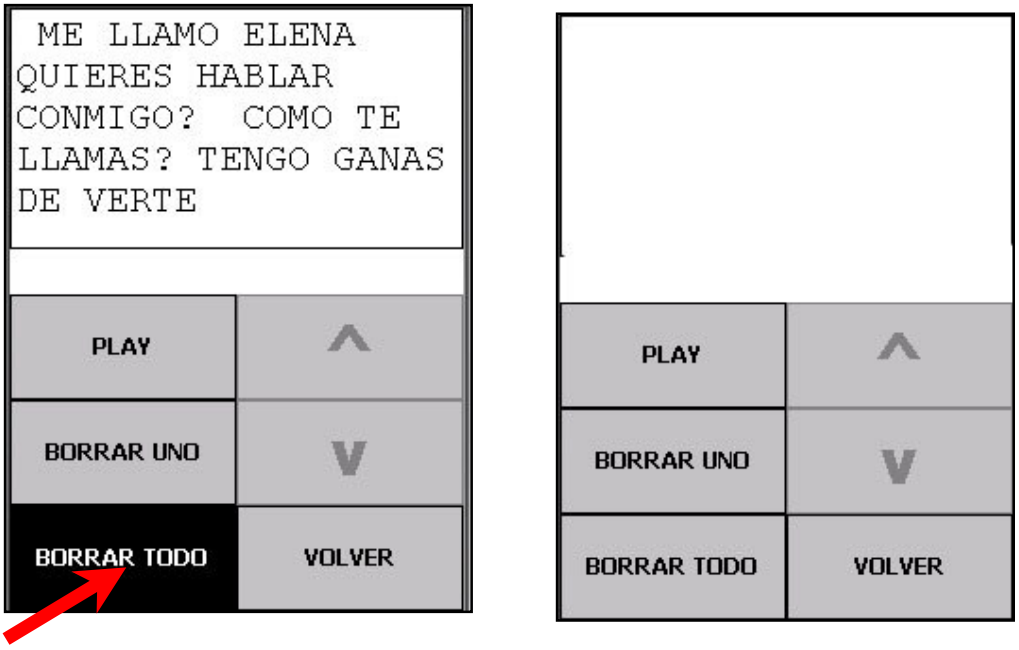
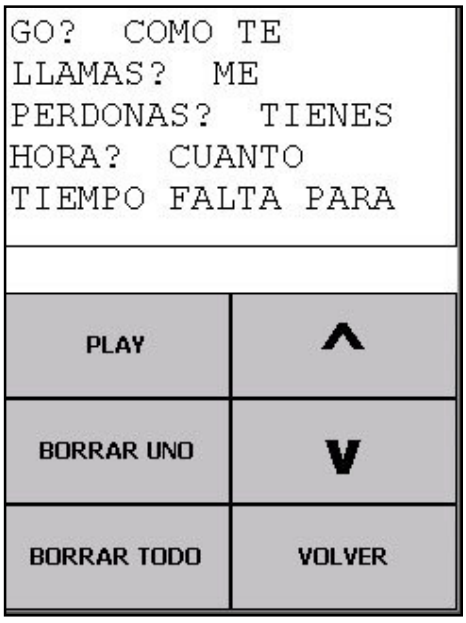


Figura 12. 46 y 12.47. Ejemplo Borrar todo



12.9.3 Desplazarte a través del texto

Estos dos botones de desplazamiento te permiten poder ver todo el texto escrito hasta el momento y moverte por él si así lo deseas.

Éstos botones sólo serán activos cuando haya el texto suficiente para que no quepa completo en una única pantalla

Figura 12.48 Menú editar texto: Desplazamiento

12.9.4 Reproducir (Play)

Para reproducir el sonido de un texto escrito un carácter hay dos posibilidades:

- Reproducirlo a través de los botones que posee el dispositivo
- Reproducirlo mediante el botón que se encuentra en editar texto.

El resultado en ambos casos es el mismo. La reproducción desde el botón del dispositivo se puede hacer en cualquier pantalla de la misma manera que borrar un carácter.

Figura 12. 49, 12.50 y 12.51 Ejemplos de reproducir texto y el Botón Volver

Si pulsas en el botón volver, te situarías en opciones, la pantalla donde te encontrabas antes de pulsar Editar Texto.

12.10 Consultar Ayuda

Consultar la ayuda sólo se puede hacer a través de los botones del dispositivo, y se puede disponer de ella en cualquier momento que lo desee el usuario.

Figura 12.52. Botón ayuda

Si pulsas el botón señalado saldrá una pantalla con las posibles ayudas que la aplicación te puede prestar.

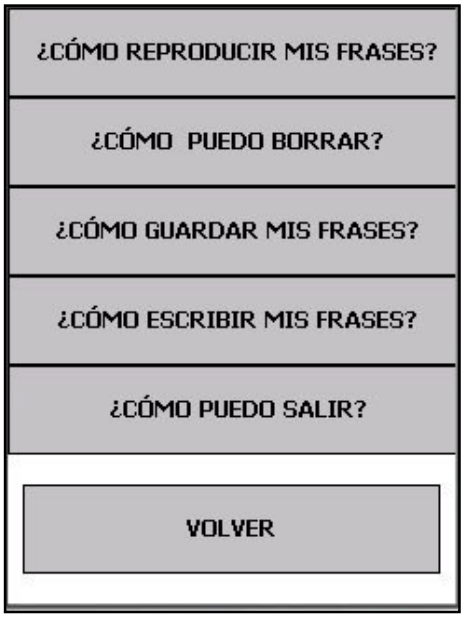


Figura 12.53. El menú ayuda

Los temas de ayuda que se ofrecen son los que aqui aparecen, pulsando en cualquiera de ellos, te saldrá la ayuda solicitada.

12.10.1 Ayuda para reproducir las frases

Figura 12.53 y 12.54. Ayuda Reproducir

La ayuda correspondiente a reproducir texto es ésta. Las opciones son que el usuario quiera reproducir las frases que él escribe en el mismo instante y lo conseguirá pulsando el botón Si, que el usuario quiera seguir consultando la ayuda de manera que pueda volver a ella si así lo desea, o por el contrario volver al mismo punto donde estaba antes de solicitar la ayuda

12.10.2 Ayuda para borrar

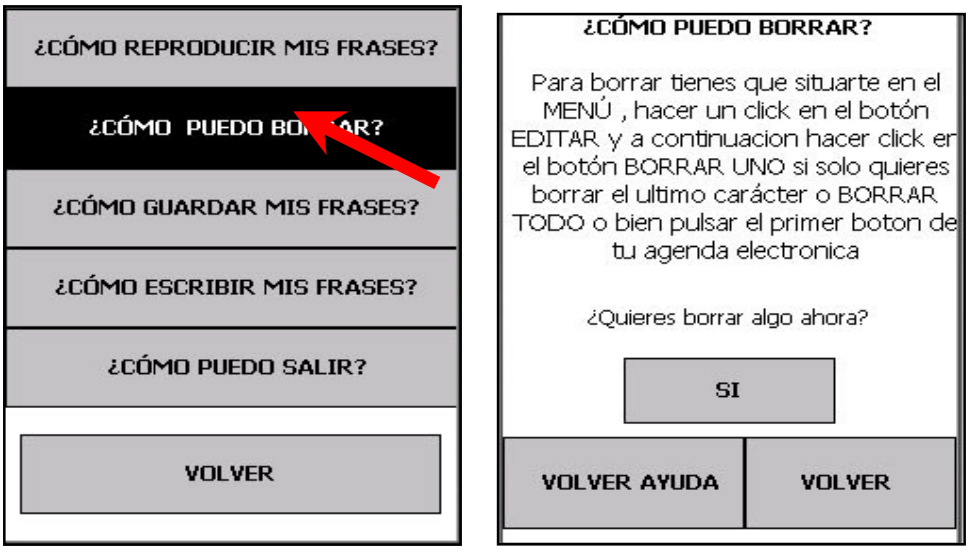


Figura 12.55 y 12.56. Ayuda Borar

Las opciones que ofrecen cada tipo de ayuda son las mismas, que son las de realizar la acción por la que se solicitó la ayuda en ese preciso momento, volver a la ayuda o volver al punto donde se estaba antes de solicitar la ayuda.

12.10.3 Ayuda para guardar frases

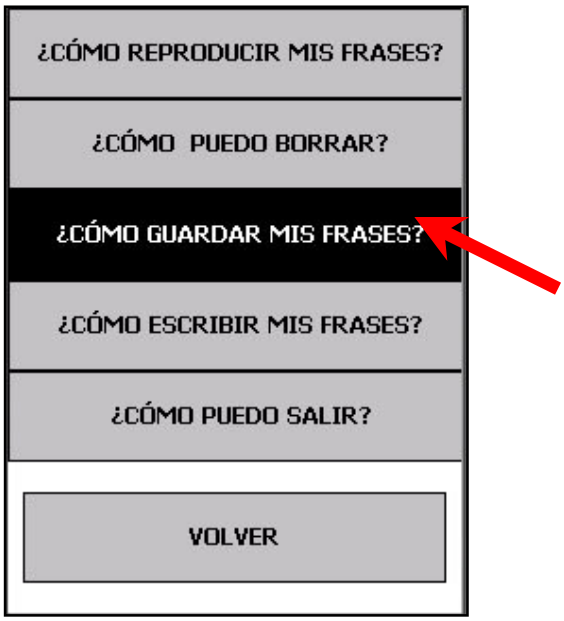


Figura 12.58 Ayuda para guardar frases

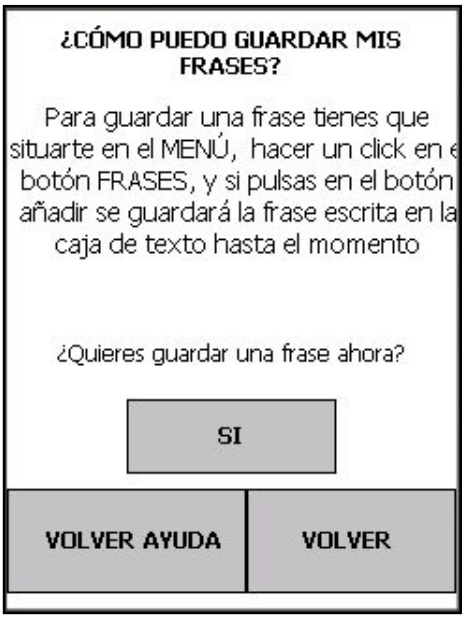


Figura 12.59 Ayuda para guardar frases

12.10.4 Ayuda para escribir las frases

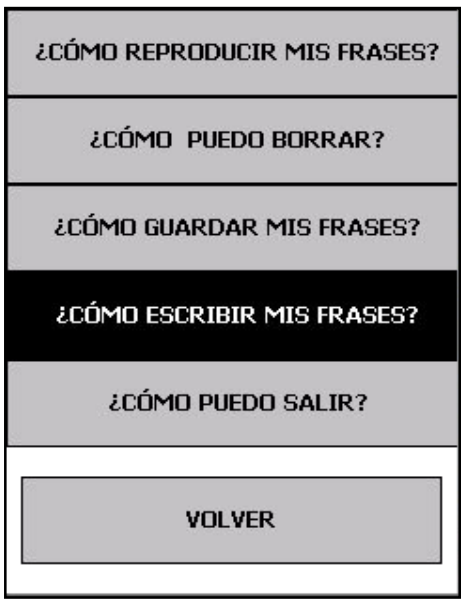


Figura 12.60 Ayuda Escribir Frases

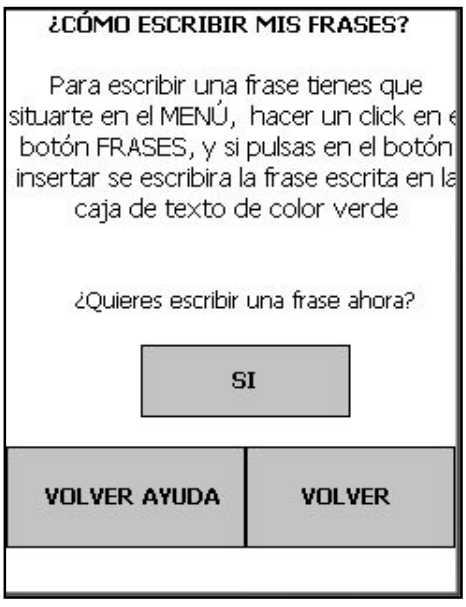


Figura 12.61 Ayuda Escribir Frases

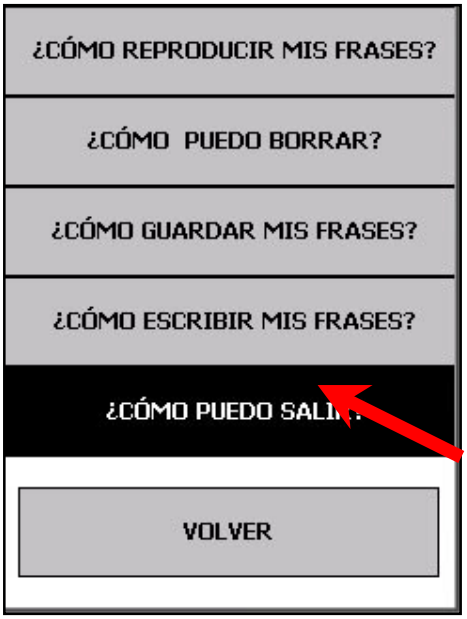


Figura 12.62 Ayuda para salir

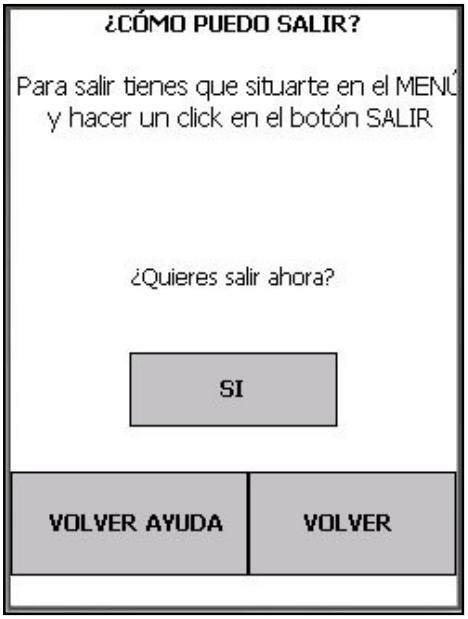


Figura 12.63 Ayuda para salir

Si pulsas el botón volver serás trasladado exactamente al punto desde donde solicitaste la ayuda.

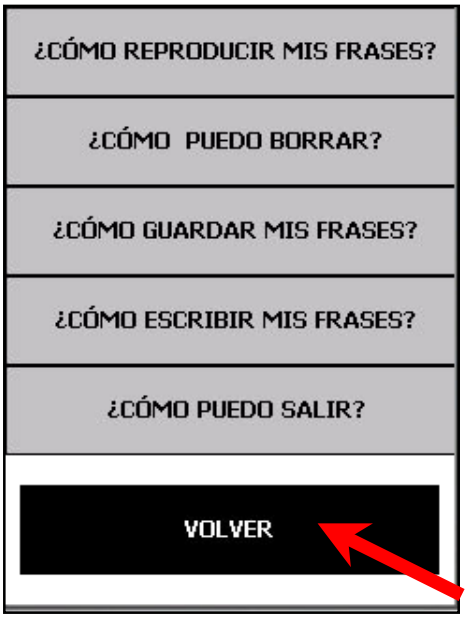


Figura 12.64 Ayuda para volver

Parte V Conclusiones

Capítulo 13. CONCLUSIONES

13.1 Objetivos alcanzados

Al principio la idea consistía en hacer un teclado que se comunicara con el ordenador de sobremesa, para poder usar los programas comerciales normales manejándolos a través de una agenda electrónica.
Sin embargo, las necesidades de los usuarios no estaban situadas en ese punto, sino en algo mucho más básico: La comunicación con el que tiene al lado.

Con este proyecto, se pretende que se pueda comunicar el usuario de una forma mucho mas fluida y natural, tanto con personas que no tengan capacidad de lectura, (sus compañeros del centro Obregón) como con gente ajena a su entorno. Gracias al sintetizador de voz, puede "hablar" con personas a través de un lenguaje común,

Por otra parte, hemos abierto el camino para que otros enfermos con necesidades diferentes se puedan beneficiar de esta alternativa. Actualmente los dispositivos móviles son mucho más accesibles, debido a su precio, que los comunicadores que usan en este tipo de centros hoy en día, sin olvidar del lenguaje tan limitado que ofrecen este tipo de herramientas. Además la aplicación estará disponible a través de la página del centro para que quien lo desee se lo pueda bajar.

La síntesis de voz no fue considerada en un principio, y al final ha sido a lo que hemos dedicado más tiempo y recursos de búsqueda de documentación de todo el proyecto.
Este cambio fue motivado por ser lo más importante para el cliente, y para sus monitores.

La usuaria para la que hemos desarrollado esta aplicación usa para comunicarse con el entorno un "silabario", consistente en una cartulina tamaño A3 donde están situados todas las sílabas del español. Para poder "hablar", necesitaba que otra persona con capacidad de lectura le sujetara el silabario delante, y que fuera leyendo en voz alta lo que ella le señalaba. Su capacidad de comunicación se limitaba a los ratos en los que una monitora o un familiar estaba con ella. Y la capacidad para conversar con alguien que no conociera ese mecanismo de comunicación era nula. Actualmente, está usando de forma habitual la agenda electrónica en su vida diaria, y ese es el objetivo más importante que hemos conseguido, que ha hecho que todos los esfuerzos realizados merecieran la pena.

Tenemos que agradecer al centro Obregón el ayudarnos a descubrir las problemáticas de un colectivo que pasa desapercibido en nuestra sociedad

13.2 Conclusiones de tipo técnico

El primer objetivo que alcanzamos fue el de tomar contacto con los dispositivos móviles y con este tipo de tecnología que al principio era ajena a nosotros. Sabemos que es un campo que está evolucionando muy rápidamente y que está en un punto de expansión, por lo que en un futuro puede sernos de gran ayuda de cara a un mercado laboral.

En segundo lugar, debido a la novedoso de esta tecnología, hemos tenido que aprender la filosofía de trabajo del .NET y de su entorno de desarrollo. Además, éste, por ser un entorno común para máquinas portátiles y equipos de sobremesa nos ha permitido adquirir conocimientos acerca de ambos.

Otra meta alcanzada, ha sido la aplicación de las técnicas aprendidas a lo largo de estos años, a un proyecto dedicado a un usuario final, en un entorno real, interactuando con dicho usuario y atendiendo los requisitos que nos solicitaba .

Capítulo 14. POSIBLES MEJORAS

En este apartado se verán las posibles mejoras que podrían hacerse sobre el producto software así como sobre el proyecto en general:

- Ofrecer la posibilidad de incorporar iconos y colores a los botones y demás elementos de la aplicación para dar una apariencia más amigable y a la vez que el contraste de los colores permita a los usuarios el uso correcto de la aplicación.

- Se podría retomar el reto de comunicar la PDA con un ordenador para conseguir manejarlo desde la silla de ruedas, a través de una tarjeta WIFI (IEEE 802.11B) o Bluetooth.

- Implementación de todos los métodos de escritura comentados, para que sea una aplicación más genérica.

- Mejora del sintetizador de voz.

- Implementación de texto predictivo a través de un árbol TRIE, para conseguir el mismo efecto de los móviles y que sea una búsqueda de palabras en tiempo real sin necesidad de tener que hacer una acción concreta para que muestre las sugerencias.

- Usar la característica de cambio de orientación de la pantalla que poseen los dispositivos con Windows Mobile 2003 Second Edition para que pueda mostrar el texto que ha escrito a un contertulio que esté situado enfrente del usuario sin necesidad de situarse detrás de él.

- Implementar la aplicación en un SmartPhone. Debería tener alguna limitación importante, sobre todo por carecer de pantalla táctil y por su tamaño

APÉNDICES

APÉNDICE A. REFERENCIAS

Bibliografía

- Grady Booch, James Rumbaugh, Ivan Jacobson, “El lenguaje unificado de desarrollo software”, Ed. Addison Wesley.
- Grady Booch, James Rumbaugh, Ivan Jacobson, “El lenguaje unificado de modelado”, Ed. Addison Wesley, 2000
- Larman, “UML y Patrones”. Ed. Pearson educacion , 2002

Fuentes Web

- www.pcdemano.com, probablemente es una de las mejores páginas sobre pocket pc en español, con multitud información muy útil y actualizada, además de usuarios muy comprometidos.
- www.edutec.es, que es una página de profesionales que se dedican al desarrollo de nuevas tecnologías para su aplicación en el campo de la educación.
- http://salud.discapnet.es, es un sitio web en el que podemos encontrar información muy completa sobre la parálisis cerebral y sus características y todo tipo de documentación acerca de esta enfermedad.
- www.microsoft.com, página oficial de Microsofr de la que se puede obtener información a cerca del entorno de desarrollo, herramientas, del sistema operativo y en general todo lo referente a Pocket pc.
- www.willydev.net, contiene información para desarrolladores, en cuanto a lenguajes de programación, ejemplos de códigos fuentes ...
- http://groups.msn.com, han sido utilizados para consultas de cuestiones muy específicas sobre las bases de datos
- www.cliekear.com, ha sido muy visitada para consultas sobre el lenguaje de programación Visual C# .NET
- www.lawebdelprogramador.com, contiene cursos, foros de consulta para el desarrollo de diversos lenguajes ...

- www.programacion.com, obviamente es una página con información acerca de la programación, lenguajes, implementaciones...
- www.msdn.microsoft.com, es la parte de la página web de Microsoft dedicada a los desarrolladores de aplicaciones para sus sistemas operativos. Con ejemplos, consejos de expertos profesionales.
- www.todopocketpc.com, es una página española dedicada a los dispositivos pocket pc , con un foro muy interesante, a demás de mucho movimiento.
- http://www.fismat.umich.mx/~karina/tesisLicenciatura/indice.html , página muy útil que nos resolvió el tema relacionado con la división de palabras en sílabas por contener las reglas en el español.
- www.mipcdebolsillo.com, es un sitio web que contiene otro de los foros que más se mueve, y con información interesante. Además tiene un buen foro de programación.

APÉNDICE B. CONTENIDOS DEL CD-ROM

Se enumera a continuación los contenidos del CD-ROM que acompaña a esta documentación:

- *Directorio Instalación*
  - *Subdirectorio Instalación PDA*: Contiene los archivos CAB necesarios para instalar la aplicación en el dispositivo. Para instalarlo directamente en la PDA hay que copiar estos ficheros a la agenda electrónica y ejecutarlos uno a uno desde allí. Para ello se puede emplear el explorador de ficheros de la propia agenda.
  - *Subdirectorio Instalación PC:* Contiene el programa de instalación que se tiene que ejecutar desde un ordenador de sobremesa que tenga Microsoft ActiveSync instalado. (presenteen los discos de instalación de cualquier PDA)
    Una vez que se ejecute el programa de instalación, desde el ActiveSync se irá a Tools→Add/Remove Programs, y desde allí se seleccionará la casilla de "proyectoTecladoAdaptado. En la siguiente sincronización de la PDA, el programa estará disponible.
- *Directorio Memoria*: Contiene la memoria en formato PDF
- *Directorio Manual de Usuario*: Contiene el manual de usuario en formato PDF.