Parte I Introducción

Capitulo 1. PRESENTACION 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 esta pensado para que sea una herramienta que facilite a las personas con parálisis cerebral la integración lo más posible a su entorno, mostrando a su vez como la informática y los avances tecnológicos con su versatilidad, pueden ayudar a la integración de este colectivo en la sociedad de hoy, conseguir que sus discapacidades tengan la menor influencia negativa posible sobre su vida y magnificar sus capacidades.
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 del Proyecto.

La aplicación esta pensada para personas con parálisis cerebral que como consecuencia de esta enfermedad tengan discapacidades motoras severas y estén privadas de la capacidad de habla.
Dentro de la aplicación se distinguirán dos secciones en función de la capacidad de lectura y escritura que hayan desarrollado:

- Primera sección basada en el lenguaje escrito, diseñada para aquellas personas que tengan la facultada de leer y escribir.
- Segunda sección basada en imágenes dirigida para aquellas personas que no hayan desarrollado la lectura ni la escritura.

1.3 Documentación Presentada.

Esta memoria será rigurosa con los contenidos teóricos y no se extenderá en descripciones exhaustivas. No pretende hacer un análisis detallado de hardware o 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 presentan.
La memoria estará dividida en partes que a su vez se dividirán en capítulos que contiendan puntos estableciendo así un orden de lectura por temas.
A continuación se describirá el contenido de cada una de las partes en que se estructura esta memoria:

- **Parte I. Introducción:** En esta sección en la que nos encontramos, se hace una breve presentación del tema que aborda el proyecto, colectivo de personas al que esta dirigido, objetivos del mismo, así como una breve descripción del contenido de esta memoria.

- **Parte II. Fundamentos Teóricos:** En este apartado se desarrollaran los contenido teóricos necesarios para comprender el proyecto. Son los siguientes:

    - Parálisis Cerebral: Definición de la enfermedad, síntomas, posibles causas etc. Contexto en el que se desarrolla la aplicación.

    - Las nuevas tecnologías de la comunicación y los alumnos con déficit: describe la situación actual de esta tecnología enfocada para persona con discapacidades físicas y mentales.

    - Introducción a los dispositivos móviles: Breve explicación sobre las tecnologías de comunicación móvil, centrándose en la PDA, dispositivo utilizado para este proyecto; características, procesadores utilizados, sistemas operativos que ejecutan, tipos, avances etc.

    - Desarrollo de aplicaciones para Pocket PC: En este capitulo se recogen los conceptos necesarios para el desarrollo de una aplicación para Pocket PC, descripción de la plataforma .NET de Microsoft, programas, herramientas, lenguajes que nos permiten la creación de aplicaciones como esta.

- **Parte III. Desarrollo de la Aplicación:** Esta parte esta constituida por tres capítulos:

    - Estudio Previo: en el se realiza un estudio sobre los posibles métodos de escritura para dispositivos móviles y mas detalladamente para PDA diseñados para las capacidades y habilidades de personas con parálisis cerebral.

    - Análisis del Sistema: En este capitulo nos centraremos en un método de escritura para PDA concreto, en colaboración con el Centro Oregon que nos ha permitido encuadrar nuestro proyecto en un entorno real.

- Diseño: En esta parte se realizara el diseño, implementación e instalación del método de escritura para PDA seleccionado.

- **Parte IV. Manual de usuario:** Es una guía pormenorizada del manejo de la aplicación.

- **Parte V. Conclusiones:** Reflexión sobre los objetivos alcanzados y sobre es futuro y nuevas vías de actuación del proyecto.

- **Apéndices.**

Capitulo 2. OBJETIVOS PROPUESTOS

2.1 Objetivos Propuestos.

Los objetivos que se pretenden alcanzar en este proyecto son los siguientes:

- Elaboración de un estudio de métodos de comunicación sobre PDA para personas con parálisis cerebral, que permita según sus capacidades la utilización de un método u otro.
- Realizar una aplicación que permita a la personas con parálisis cerebral una comunicación mas natural y sencilla.
- Conseguir que la aplicación sea lo mas configurable posible para que se pueda adaptar a las necesidades de cada persona.
- Ayudar a mejorar la calidad de vida de personas con discapacidades en muchos casos severas, así como a las personas de su entorno, desgraciadamente los medios a su alcance normalmente son limitados.
- Valorar y conocer a un colectivo de personas a las que las nuevas tecnologías pueden servir de gran ayuda en su dia a dia.
- Conseguir desarrollar una aplicación final, destinada al uso de personas reales.
- Colaborara con el Centro Oregon, conociendo la verdadera situación en la que se encuentran las personas con parálisis cerebral sus capacidades, limitaciones, algo fundamental para el diseño de la aplicación.
- Fomentar el trabajo en equipo, no solo entre nosotros, también con usuarios finales del proyecto.
- Acercar la Tecnología de la Información a personas con grandes dificultades de comunicación.
- El uso correcto de las técnicas adquiridas durante estos años.
- Aplicación de una metodología de desarrollo del software que permita la creación de esta aplicación.
- Conocer el entorno de trabajo para la realización para aplicaciones para dispositivos móviles.

Parte II. Fundamentos Teóricos

Capitulo 3. CONTEXTO DE LA APLICACIÓN: LA PARALISIS CEREBRAL

3.1 Parálisis Cerebral.

3.1.1 Definición.

En la actualidad se considera a la parálisis cerebral como un trastorno persistente del movimiento y de la postura causada por una lesión no evolutiva del sistema nervioso central durante un período temprano del desarrollo cerebral, limitado generalmente a los tres primeros años de vida.

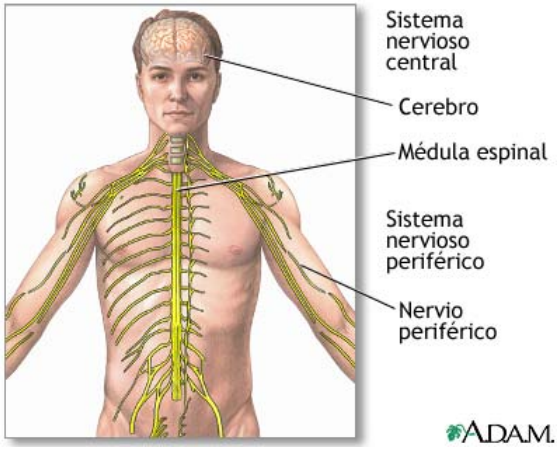


Figura 3.1 Sistema Nervioso Central

Los síntomas de la parálisis cerebral son de severidad variable y difieren de una persona a otra, también pueden cambiar en el individuo con el tiempo. Algunas personas están afectadas de otros trastornos médicos incluyendo convulsiones o retraso mental.

La parálisis cerebral no es contagiosa y usualmente tampoco es hereditaria de una generación a otra. Actualmente ésta no puede ser curada, aunque la investigación científica sigue buscando mejores tratamientos y métodos de prevención.

3.1.2 Tipos de Parálisis Cerebral.

La diversidad de casos existentes se clasifican tradicionalmente en función de la zona del encéfalo afectada, además de los síntomas motores, diferentes para cada grupo:

- **Espástica:** En esta forma de parálisis cerebral, que afecta de 70 a 80 por ciento de los pacientes, los músculos están rígidos y contraídos permanentemente. A menudo los médicos describen la clase de parálisis cerebral que el paciente padece basándose en las extremidades afectadas. Los nombres asignados para estas clases de enfermedad combinan una descripción latina de las extremidades afectadas con el término *plejia o paresis* para significar paralizado o débil respectivamente. Las cuatro clases de parálisis cerebral espástica diagnosticadas más comúnmente están ilustradas en la figura de arriba.

Cuando ambas piernas se afectan de espasticidad, éstas puedan encorvarse y cruzarse a las rodillas. Esta postura anormal de las piernas, de apariencia de tijeras, puede interferir con el caminar.

Los individuos con hemiparesis espástica pueden experimentar también temblores hemiparéticos, en los cuales sacudidas incontrolables afectan las extremidades de un lado del cuerpo. Si estas sacudidas son severas pueden obstaculizar seriamente el movimiento

- **Atetosis:** Esta forma de parálisis cerebral se caracteriza por movimientos retorcidos lentos e incontrolables. Estos movimientos anormales afectan las manos, los pies, los brazos o las piernas y en algunos casos los músculos de la cara y la lengua, causando el hacer muecas o babear. Los movimientos aumentan a menudo durante períodos de estrés emocional y desaparecen mientras se duerme. Los pacientes pueden tener problemas coordinando los movimientos musculares necesarios para el habla, una condición conocida por *disartria.* La parálisis cerebral atetoide afecta aproximadamente de 10 a 20 por ciento de los pacientes.

- **Atáctica:** Esta forma rara afecta el equilibrio y la coordinación. Las personas afectadas caminan inestablemente con un modo de caminar muy amplio, poniendo los pies muy separados uno del otro, y experimentan dificultades cuando intentan movimientos rápidos y precisos como el escribir o abotonar una camisa. Los pacientes pueden exhibir temblores de intención. En esta forma de temblor, el empezar un movimiento voluntario, como agarrar un libro, causa un temblor que afecta la parte del cuerpo usada. El temblor empeora según el individuo se acerca al objeto deseado. Se estima que la forma atáctica afecta de 5 a 10 por ciento de los pacientes con parálisis cerebral.

- **Formas Combinadas:** Es muy común que los pacientes tengan síntomas de más de una de las formas de parálisis cerebral mencionadas arriba. La combinación más común incluye espasticidad y movimientos atetoides, pero otras combinaciones son posibles.

3.1.3 ¿Qué causa la Parálisis Cerebral?

La parálisis cerebral no es una sola enfermedad con una sola causa, como varicela o rubéola. Más bien, es un grupo de trastornos relacionados entre sí que tienen causas diferentes.

Infecciones durante el embarazo	El sarampión alemán o rubéola es causada por un virus que puede afectar a la mujer embarazada, y por consiguiente, al feto en el vientre causando daño al sistema nervioso en desarrollo.
Ictericia en los infantes	Ictericia grave y sin tratar puede hacer daño a las células cerebrales.
Asfixia Perinatal	Durante el parto, la falta de oxígeno en la sangre o flujo reducido de sangre al cerebro, o ambas condiciones pueden causar una deficiencia de oxígeno en el cerebro del recién nacido, causando así la condición conocida como asfixia perinatal.
Incompatibilidad del Rh	En esta condición sanguínea, el cuerpo de la madre produce células inmunológicas llamadas anticuerpos que destruyen las células sanguíneas del feto, conduciendo así a una forma de ictericia en el recién nacido.
Apoplejía o hemorragia intracraneal	La hemorragia intracraneal, cuando el cerebro sangra, tiene varias causas — entre las que se incluyen la ruptura de los vasos sanguíneos del cerebro, la obstrucción de los vasos sanguíneos o células sanguíneas anormales — y es una forma de apoplejía. La insuficiencia respiratoria en el recién nacido, un trastorno respiratorio que es muy común en los bebés prematuros, es una de las causas

3.1.4 Síntomas.

Muchos individuos con parálisis cerebral no tienen otros trastornos médicos asociados. Sin embargo, los trastornos que involucran el cerebro y obstaculizan su función motora pueden causar también convulsiones y menoscabar el desarrollo intelectual del individuo, su atención al mundo exterior, la actividad, la conducta, su visión y la audición. Los trastornos médicos asociados con parálisis cerebral incluyen:

- **Retraso Mental:** Aproximadamente un tercio de los niños con parálisis cerebral tienen una limitación intelectual leve, un tercio presenta incapacidad moderada o grave y el tercio restante es intelectualmente normal.

- **Convulsiones o epilepsia:** Tantos como la mitad de todos los niños con parálisis cerebral tienen convulsiones. Durante una convulsión, el modo normal y ordenado de la actividad eléctrica en el cerebro se interrumpe por estallidos incontrolables de electricidad. Cuando las convulsiones resurgen sin causa directa, tal como tener fiebre, la condición se llama epilepsia. En la persona con parálisis cerebral y epilepsia, esta interrupción puede difundirse a través de todo el cerebro causando síntomas variables por todas las partes del cuerpo o puede ser limitada a una sola parte del cerebro y causar síntomas más específicos como convulsiones parciales.

- **Problemas de crecimiento:** El síndrome llamado "failure to thrive" en inglés, o fracaso de medrar, es común en niños con parálisis cerebral de moderada a grave, en especial en aquellos con cuadriparesis espástica. Fracaso de medrar es el término general usado por los médicos para describir a los niños que muestran falta de crecimiento o desarrollo a pesar de recibir suficiente alimento. En los bebés este retraso se manifiesta en peso bajo, en los niños en estaturas bajo lo normal y en los jóvenes puede presentarse en una combinación de estatura baja y falta de desarrollo sexual. Es probable que el fracaso de medrar tenga muchas causas, incluyendo en particular, mala nutrición y daño a los centros cerebrales que controlan el crecimiento y el desarrollo.

- **Visión y audición limitadas:** Un gran número de niños con parálisis cerebral tienen estrabismo, una condición en la cual los ojos no están alineados debido a diferencias en los músculos del ojo izquierdo y el derecho. En un adulto, esta condición causa doble visión. Sin embargo, el cerebro de los niños a menudo se adapta a la condición ignorando las señales del ojo desalineado. Sin tratamiento, ésta puede conducir al deterioro de la vista de un ojo y puede interferir en ciertas habilidades visuales, como el juzgar distancias. En algunos casos, los médicos pueden recomendar cirugía para corregir el estrabismo.

- **Sensibilidad y percepción anormales:** Algunos niños con parálisis cerebral tienen deficiencias en la habilidad para sentir sensaciones simples como las del tacto o el dolor. También pueden tener estereoagnosia, es decir, dificultades en percibir o identificar objetos usando el sentido del tacto. Por ejemplo, sin observar el objeto, un niño con estereoagnosia tiene dificultades en identificar una pelota, esponja u otro objeto puesto en su mano.

- **Problemas comportamentales:** Ansiedad, obsesión, inseguridad, hiperactividad, dificultades de adaptación.

- **Alteraciones del lenguaje:** entre el 70 y el 80 % de los casos.
  A pesar de todos estos trastornos, las características individuales pueden variar en el tiempo sí las diferentes rehabilitaciones aprovechan la plasticidad que ofrece un cerebro en desarrollo. Entendiendo como plasticidad, la capacidad de reestructuración funcional y estructural del sistema nervioso central, tras una agresión. En el niño con P.C.I, tras lesiones en algunos casos puntuales y en otros muchos muy diversas, se comprueba la noción de plasticidad cerebral y como zonas del cerebro no afectadas, asumen parte de las funciones de las áreas lesionadas. De esta forma niños con lesiones corticales o bulbares importantes pueden llegar a caminar, escribir o tener un habla funcional.

3.1.5 Problemas del Lenguaje.

El desarrollo del lenguaje en el niño se va realizando de forma ininterrumpida desde el nacimiento. Durante el primer año el niño desarrolla las bases de la comunicación, por medio de las interacciones que realiza con la familia, en las que son muy importantes la mímica facial, la entonación, la prosodia, el balbuceo, la coordinación sonido- vista...etc. Todo ello asociado al contexto y dentro de lo que se denominan funciones de comunicación. Además durante esta época se desarrolla la percepción auditiva y las habilidades fonológicas, empezando a adquirir el lenguaje de su entorno, con aspectos específicos a nivel comunicativo y gramatical. Se produce un desarrollo muy importante del lenguaje entre los 2 y los 3 años.

Después del primer año de vida las características de adquisición y desarrollo del lenguaje serán muy diferentes de unos casos a otros. Aproximadamente un 20 % de los niños con parálisis cerebral, no tendrán ningún problema en cuanto a la adquisición del lenguaje, pero en el resto los problemas del lenguaje irán desde pequeñas dificultades, a alteraciones de la comunicación realmente graves. No se puede referir un patrón general en cuanto a estas dificultades. Además estos problemas pueden afectar de forma desigual a los diferentes componentes del lenguaje (fonética, morfología, sintaxis, semántica y pragmática). Se diferencian dos grandes apartados en relación a lenguaje:

- **Aspectos motores del lenguaje:** En el niño con parálisis cerebral, la evolución de las funciones motoras relacionadas con el habla no se produce en general de forma adecuada debido a la lesión del S.N.C, presentan retraso o imposibilidad de realizar el desarrollo normal, persistiendo reflejos primitivos (ex. reflejo de Moro a nivel general y el reflejo de nausea a nivel de la zona oral), además de patrones de movimiento anormales. Básicamente estará alterado lo que Aronson y cols (1980), califican como habla motora. El término "habla motora", pretende englobar los procesos neuromusculares requeridos para el acto del habla.

Los procesos neuromusculares implicados necesitan precisión en el tiempo, colocación, exactitud en su dirección y fuerza de movimiento. En las personas con problemas neurológicos, estos procesos pueden estar alterados. En la P.C.I lo que está afectado no es uno de estos aspectos, sino múltiples procesos que implican varias funciones relacionadas con el habla. Las alteraciones en el habla motora reflejan el problema neurológico de base. Algunas de las funciones motoras que más comúnmente están afectadas son:

- Reflejos anormales en la zona oral, esto es reflejo de succión, deglución, morder y nausea que pueden estar exaltados, ser insuficientes o no estar presentes. Persistencia de estos reflejos debido a que el niño es incapaz de inhibirlos debido a la lesión cerebral.

- Respiración. La respiración debe proporcionar suficiente aire para mantener el control de la fonación, a nivel de intensidad y duración. En el paralítico cerebral puede haber una capacidad respiratoria insuficiente y más frecuentemente una mala coordinación.

- Fonación. La fonación se refiere al paso de aire a través de los repliegues vocales para emitir sonido. Muchas veces se produce un movimiento de aducción incompleto de los repliegues vocales lo cual produce alteraciones en la intensidad, timbre y sonoridad de la voz.

- Articulación. La articulación esta en función de la fuerza, precisión y coordinación de los movimientos de la lengua, labios y maxilar. En general se trata de una afectación global que varía con el tipo de P.C.I y el grado de afectación. Las alteraciones en la articulación van acompañadas` de alteraciones en los movimientos de la mandíbula, labios y lengua, que pueden ser reflejos no aislados, mal coordinados y mal graduados. Puede ser que un sonido se llegue a producir aislado, pero no dentro de una palabra o frase en donde es necesario gran precisión de movimientos a una velocidad determinada.

- Prosodia. Se refiere a la entonación, melodía, ritmo. El paralítico cerebral puede presentar alteraciones muy diversas en algunos casos con exceso prosódico, en otros, habla monótona o una inadecuada utilización del ritmo y del acento.

- **Aspectos Lingüísticos:** Sus características son muy diversas y a diferencia de las alteraciones en las funciones motoras del habla, no tienen relación con el tipo de parálisis cerebral. Tampoco tienen relación con la intensidad de la lesión motriz. Puede tratarse de un paralítico cerebral, muy grave que necesite expresarse por medio de Sistemas Aumentativos de Comunicación (S.A.C) y su lenguaje sea totalmente normal. Las alteraciones del lenguaje son frecuentes en el paralítico cerebral, pero al igual que los aspectos cognitivos, son difíciles de observar y de evaluar debido precisamente a la problemática motriz que impide pasarles pruebas estándar, incluso pruebas basadas en la observación. Los datos existentes si apuntan a la posibilidad e que igual que en los aspectos cognitivos, estos trastornos se incrementan con el tiempo.

Estos problemas estarían relacionados con dificultades del individuo para acceder a la información necesaria y procesarla. En general algunas de las alteraciones más frecuentes son:

- A nivel fonético. Las dificultades mas frecuentes son de tipo articulatorio y estas sí están directamente relacionadas con el problema motor.

- A nivel morfo- sintáctico. Las dificultades en esta área se relacionan con problemas para mantener la respiración o con otras causas motoras y ambientales. Pueden tener tendencia a una longitud media de los enunciados verbales (LMEV), reducida en relación a sus posibilidades.

- Léxico. No hay problemas específicos, pero si tendencia a utilizar un léxico restringido, en relación al nivel receptivo. Puede ser debido a que con ello el programa motor les resulta más fácil de llevar a cabo.

- Pragmática. Desde pequeño puede haber u uso reducido en las funciones del lenguaje, posteriormente en relación a diferentes aspectos pragmáticos, con un lenguaje muy concreto y bajo nivel de uso.

- **Neuropsicología y Lenguaje:** No puede hablarse de unas características comunes a todos los casos de parálisis cerebral debido a los aspectos comentados anteriormente de la gran diversidad de manifestaciones y subtipos clínicos. Las alteraciones en las funciones motoras relacionadas con el lenguaje expresivo son un denominador común, pero no afectan a todos los casos. En los que sí hay problemas motores, los síntomas pueden ser muy diversos de unos a otros. A continuación se citan algunos aspectos neuropsicológicos presentes en un número importante de casos:

  - Lentitud en la recepción de estímulos.
  - Dificultades en decodificar estímulos complejos.
  - Lentitud en dar una respuesta determinada.
  - Dificultades de asociación

Capitulo 4. LAS NUEVAS TECNOLOGIAS DE LA COMUNICACIÓN Y LOS ALUMNOS CON DISCAPACIDADES

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

Los trastornos que provoca la parálisis cerebral infantil no se limitan a una sola área o aspecto de la vida, sino que influyen de manera global sobre el desarrollo de la personalidad y que, indirectamente, condicionan la calidad de vida.
Las posibles aportaciones que la informática puede ofrecer, siempre que sea introducida de forma eficaz y adecuada, en los procesos educativos de los alumnos con parálisis cerebral serian:

1. Posibilitar la realización de determinadas actividades limitadas o imposibles de realizar debido a su problemática motora.
2. Ayuda educativa en cuanto facilita una adecuada interacción con los procesos instructivos.
3. Ayuda pedagógica para el profesorado de cara a optimizar la relación, a menudo complicada, entre el profesor / el alumno / los procesos de enseñanza-aprendizaje.
4. Ayuda pedagógica de cara a posibilitar el acceso a currículums más amplios y normalizados.
5. Medio posibilitador o favorecedor de la comunicación.
6. Medio posibilitador de la integración personal, escolar, social y laboral.
7. Ayuda eficaz para la realización de psicodiagnósticos y otros tipos de valoraciones.
8. Medio liberador de determinados trastornos psíquicos.

1. Posibilitar la realización de determinadas actividades limitadas o imposibles de realizar debido a su problemática motora.

Si bien es cierto que la mayor problemática de la Parálisis Cerebral es un trastorno del tono muscular, de la postura y del movimiento ésta no es la única que condiciona su vida, aunque sea la que, en un principio y dentro del marco educativo, le provoque mayores dificultades y limitaciones:

-Abrir los libros y cuadernos.
-Escribir.
-Contar con los dedos y/o otros mecanismos convencionales del cálculo.
-Agarrar correctamente los libros y cuadernos.

-Manipular, explorar,…, realizar las actividades necesarias para conocer un objeto.
-Acceder a las maletas, estanterías, librerías,…, para coger el material que le es necesario.
-Abrir los libros y cuadernos.

2. Ayuda educativa en cuanto facilita una adecuada interacción con los procesos instructivos.

Considerando como entorno educativo habitual un aula de educación especial en la que atienden a niños con PCI valoraremos como importantes las aportaciones que la utilización del ordenador puede introducir en los procesos instructivos.

La capacidad para controlar los estímulos consiguiendo un control efectivo de la atención, es una de las aportaciones que facilita la posibilidad de aprovechar al máximo las posibilidades de cada alumno ya que, debido a problemas posturales y de control de atención, se dificulta en gran medida la eficacia de los procesos educativos. Los programas, sobretodo los dirigidos a personas con retraso intelectual o a niños en edades tempranas, pueden incluir sistemas para captar la atención de acuerdo con las características de la población a la que se dirigen: sonidos, dibujos, efectos especiales,..., preguntas inoportunas, contraseñas, claves,..., que pueden hacer, incluso, difícil conseguir que los usuarios abandonen el programa (afortunadamente, este hecho se da de forma habitual en este tipo de trabajo).

Otra posibilidad que ofrecen los ordenadores es fomentar la motivación y el interés por el aprendizaje, que a menudo ya se ha perdido o no ha existido nunca debido al aburrimiento o frustración que provoca el no poder realizar prácticamente ninguna de las actividades consideradas como normales en un aula. Es muy fácil que los mismos niños comparen sus trabajos con los de sus hermanos, amigos,..., y lo consideren de inferior volumen (teniendo en cuenta la cantidad) o de inferior calidad (si ya son capaces de valorar las características y el grado de participación necesaria para la realización de aquella tarea).

El hecho de poder realizar un copiado, una redacción, un dibujo,..., podérselo llevar a casa, enseñarlo a la familia,..., hace que este interés perdido aumente cada vez más.

Es importante también disminuir las diferencias entre los procesos educativos de estos y otros alumnos sin ningún tipo de problemática. Aumentar las opciones curriculares, García L. (1990), accediendo a nuevas áreas y posibilitar las adaptaciones que faciliten el acceso a nuevos sistemas educativos.

La posibilidad de ofrecer unos medios educativos que atiendan a las características individuales de cada alumno permite cubrir las amplias y numerosas diferencias que existen entre cada uno de ellos.

Posibilitar la individualización, facilita poder dedicar una mayor atención a las diferencias que condicionan los mecanismos de aprendizaje de cada uno de los alumnos.

3. Ayuda pedagógica para el profesorado de cara a optimizar la relación, a menudo complicada, entre el profesor / el alumno / los procesos de enseñanza-aprendizaje.

Ya las ventajas que por si solas acompañan a la utilización del ordenador debido a las características intrínsecas de flexibilidad, versatilidad, variedad de imágenes, sonidos y efectos,..., se pueden considerar como una valiosa ayuda a la hora de trabajar con niños en edad temprana o con algún tipo de problema de aprendizaje. La versatilidad, paciencia, capacidad para ofrecer un refuerzo inmediato, ofrecer una constancia gráfica de los resultados,..., son ejemplos de las prestaciones que destacan como importantes de cara a introducir al niño en estos nuevos medios educativos.

El ordenador considerado tanto como soporte de la pedagogía tradicional, o como medio para introducir las nuevas tecnologías en las aulas, es un instrumento válido para ampliar las experiencias del educador, aportando nuevos elementos o nuevas estrategias de enseñanza que faciliten o mejoren su tarea educativa.

4. Ayuda pedagógica de cara a posibilitar el acceso a currículo más amplios y normalizados.

El desarrollo de la inteligencia en los alumnos con PCI no sigue unas pautas normales y, aunque es posible la no existencia de un deterioro intelectual, estos alumnos se encuentran con grandes dificultades para asimilar un gran número de contenidos.

Lógicamente, los problemas de incorporación de esquemas de lateralizad, direccionalidad, esquema corporal, conocimiento del propio cuerpo,..., son factores que condicionan y dificultan la adquisición de aprendizajes. Estos problemas pueden solventarse, en gran medida, con la utilización y práctica de determinados programas informáticos.

Los ordenadores permiten además trabajar aspectos como la percepción, memoria, desarrollo intelectual, capacidad de observación, práctica con sonidos y ritmos, establecimiento de relaciones causa efecto, sensibilidad musical, entrenamiento neuro-motor, simulación, dibujo, conceptos de semejanzas, diferencias, discriminaciones, cálculo elemental, aprendizaje de conceptos básicos,...).

5. Medio posibilitador o favorecedor de la comunicación.

Los problemas de lenguaje que padecen la mayoría de niños afectados de PCI suponen uno de los mayores handicaps a superar para conseguir una verdadera integración en cualquiera de sus posibles vertientes: familiar, escolar, social, laboral,...

La posibilidad de emitir un mensaje es una condición indispensable para poderse comunicar con las personas que nos rodean; en muchos casos, las personas con PCI no pueden comunicarse por si solos y, en el mejor de los casos, pueden sustituir el lenguaje tal como nosotros lo entendemos por otros llamados "alternativos o aumentativos".

Estos lenguajes consisten en un soporte (de diferentes tamaños y características) con símbolos que representan sujetos, cualidades, acciones,..., en el que, mediante un sistema de señalización se va indicando aquello que se desea comunicar. Un problema con el que nos encontramos en muchos casos es la dificultad o imposibilidad de conseguir una señalización funcional debido a los handicaps motores.

Afortunadamente existe la posibilidad de informatizar estos sistemas sustitutorios del lenguaje oral (sistemas ideográficos, pictográficos, Bliss,...) posibilitando y facilitando su utilización y ampliando las posibilidades comunicativas.

La posibilidad de ofrecer un medio que facilite una forma de comunicación (p.e. sintetizadores de voz) y que, a su vez, permita superar todas y cada una de las barreras que el espacio nos impone (especialmente a las personas con problemas de movilidad) posibilitando el acceso a numerosas fuentes de información, trabajar desde casa, desde el colegio,..., es ya hoy una realidad que, inexplicablemente, está desaprovechada, sobretodo por aquellas personas a las que les puede ofrecer un mayor número de servicios y que puede constituir una forma para superar las barreras entre la integración y la desintegración.

6. Medio posibilitador de la integración personal, escolar, social y laboral.

Posiblemente todos estamos de acuerdo cuando pensamos en la integración como el objetivo final, como la única meta válida para considerar como efectiva nuestra tarea profesional en el campo de la discapacidad.

La enseñanza debe preparar al discente precisamente para la vida postescolar, proporcionándole aprendizajes que le permitan la máxima independencia y supervivencia social.

El ordenador puede ayudar a no crear diferencias entre los alumnos con problemas y los alumnos considerados "normales"; puede ayudar a que las diferencias entre ambos procesos educativos disminuyan, ayudando a trabajar de la misma forma y con los mismos medios a los dos grupos de estudiantes. Se trata de un recurso tecnológico que ayuda a normalizar la modalidad educativa de las persona con minusvalía. Es decir, mediante el ordenador no buscamos objetivos educativos distintos a los de otros alumnos, sino posibilitar el acceso a los objetivos planteados en el currículum escolar como finalidades del sistema educativo para todos los estudiantes.

Debemos tener en cuenta que el concepto de integración está extraordinariamente ligado al concepto de ocupación profesional. Es necesario mostrarse competente en la sociedad en general y en el trabajo en particular.

El acceso al mundo del trabajo sólo se consigue demostrando el nivel de competencia necesario para la realización de las actividades requeridas. Los campos profesionales a los que las personas con algún tipo de minusvalía pueden acceder no son tan numerosos como debiera. La posibilidad de conseguir un trabajo es muy reducida considerando las desventajas con las que este grupo de personas se encuentran debido a sus dificultades.

La mayor dificultad para conseguir un trabajo para muchas personas discapacitadas se debe directamente a sus limitaciones físicas. Con la utilización de los ordenadores y otras ayudas tecnológicas, muchas limitaciones motoras, manipulativas y comunicativas se han podido superar, capacitando a las personas discapacitadas para lograr cualificarse para trabajos a los que antes les era imposible acceder.

Es por tanto evidente que las innovaciones tecnológicas pueden contribuir de forma efectiva a la ocupación de personas con discapacidad, ofreciendo nuevas posibilidades de trabajo y posibilitando que las valoraciones de sus capacidades laborales puedan resultan ampliadas, Navarro (1991).

Otro campo en el que las nuevas tecnologías ofrecen numerosas aplicaciones es en el control del entorno.

Una de las necesidades más profundamente sentidas por las personas con grandes handicaps físicos es la superación de su propia dependencia de otras personas para poder realizar los actos más elementales de modificación del entorno: encender o apagar la luz, contestar o llamar por teléfono, manejar la televisión, la radio,..., son acciones imposibles de realizar. Hoy en dia existen numerosas aplicaciones en este campo, por ejemplo, el sistema INCE (Interface de Control del Entorno) que permite la realización de un gran número de actividades mediante una función residual muscular mínima.

7. Ayuda eficaz para la realización de psicodiagnósticos y otros tipos de valoraciones.

Es muy difícil realizar una valoración psicopedagógica del niño con PCI, ya que las dos áreas más afectadas son la de comunicación verbal y la manipulativa, y son precisamente éstas, las que componen la mayoría de texts y pruebas psicopedagógicas; consecuentemente los resultados obtenidos no siempre se corresponden con el estado real del alumno. Por este motivo los ordenadores, ofreciendo un medio de comunicación y reduciendo al mínimo las actuaciones motoras, pueden ser de gran utilidad como fuente de determinadas destrezas.

8. Medio liberador de determinados trastornos psíquicos.

La frustración que padece el niño paralítico cerebral (sobretodo en los casos en que no existe deterioro de la inteligencia) puede dar lugar a numerosas conductas patológicas como pasividad, indiferencia, resignación,... En otros casos, la no-aceptación de su estado también puede dar lugar a conductas agresivas. Es muy importante evitar reacciones depresivas aumentando el nivel de autoestima y ayudándole a crear una buena imagen de sí mismo.

En gran medida, estos trastornos psíquicos se deben al bajo nivel de autoestima que les produce el no poder participar en la mayoría de actividades que se desarrollan en su entorno. La posibilidad de controlar los fenómenos de forma activa, eliminando el sentido del fracaso ayuda a que el niño con PCI de un cambio total en la percepción de su autoimagen, aumentando el nivel de autoestima y olvidando los sentimientos negativos sobre su propia persona. Las depresiones, de esta forma, tienden a disminuir y, incluso, pueden llegar a desaparecer.

La informática puede poner en manos del usuario la posibilidad de memorizar, presentar, manipular, analizar, calcular, investigar, dialogar y controlar toda clase de equipos periféricos de entrada, salida y almacenamiento de información,

4.2 Sistemas de ayuda para personas con discapacidades.

Las nuevas tecnologías pueden ser en si mismas un recurso y un apoyo de los programad de intervención pedagógica dirigidos a alumnos con deficiencias motóricas. Podemos clasificar los sistemas de ayuda basados en la tecnología de la información para personas con parálisis cerebral en:

- **Sistemas Alternativos y Aumentativos de Acceso a la Información:** Son ayudas para personas con discapacidad visual y/o auditiva, por ejemplo :

    - Reconocedores de voz y conversores texto- voz.
    - Sistemas multimedia que procesan, almacenan y transmiten de forma integrada imágenes, voz, texto y datos.
    - Videotelefonía, teléfonos de texto, fax.
    - Equipos de video que transmiten telefónicamente imágenes en movimiento de suficiente calidad para producir el acceso a una comunicación mediante lenguaje de signos e incluso lectura labial.
    - Sistema multimedia y pantallas táctiles, sin necesidad de utilizar teclado, trackball ni mouse, permiten en algunos casos detener el deterioro cognitivo y recuperar algunas funciones cerebrales superiores.

- **Sistemas de Acceso:** Son Interfaces adaptativas que permiten a las personas con discapacidad física o sensorial utilizar una computadora. Entre las ventajas podemos de estos sistemas podemos mencionar: desarrollo de la habilidad sensorial y motriz, aumento de la creatividad, reducción de la fatiga, desarrollo de nociones temporales etc. Ejemplos de estos sistemas:

    - Telelupas: posibilitan la lectura a personas con disminución visual.
    - Sintetizador Braille: permite a una persona invidente escribir información simulando a una máquina Perkins y verificar luego la misma.
    - Sistema de Reconocimiento óptico de caracteres: este dispositivo permite a una persona con discapacidad visual reproducir la información desde una computadora utilizando un scanner que lee cualquier texto mediante un programa OCR y los retransmite por medio de un sintetizador de voz o una línea Braille.
    - Teclado de Conceptos: fue pensado para personas con discapacidad motriz, y

consiste en una cuadrícula en blanco que se puede agrupar de acuerdo a varios conceptos temáticos asignados por los terapeutas.
- Mouses: Los hay tipo palancas, pedal, esférico (track ball), touch, etc.
- JoyMouse: permite utilizar el joystick como mouse para desplazar el puntero a través de la pantalla.
- Pantallas táctiles: Permiten que personas con dificultades motrices puedan acceder a los movimientos del cursor con la presión de un dedo o mano.
- Navegadores: navegadores que funciona con comandos verbales.

- **Sistemas Alternativos y Aumentativos de comunicación:**

  - El programa conocido como L.A.O. (Logopedia Asistida por Ordenador): programa de texto escrito, con imágenes y signos, fue pensado para la atención de alumnos que tienen un dominio de la lectura pero que aún tienen dificultades de comprensión.
  - El PHONOS: orientado específicamente a los atributos del habla (ritmo, entonación, articulación) y a la competencia lingüística por medio de imágenes sonoras o escritas.
  - El Programa IMASON, de aplicaciones informáticas para la intervención y/o rehabilitación de la percepción auditiva, discriminación y asociación del sonido a través de la computadora, asociando las fuentes de sonidos con las imágenes.
  - Visualizador Fonético Speechwiever (IBM-España) que trabaja todo lo relacionado con la prosodia y las cualidades de la palabra articulada.
  - Sistema Videovoz (Copextel-Cuba) que cumple funciones similares para la formación, corrección y desarrollo del lenguaje, registrando los fonemas en pantalla. Su adaptación permite el uso en el hogar por medio de televisores y monitores en forma indistinta.

- **Sistemas de Movilidad:** Son aquellos relacionados a la movilidad personal y las barreras arquitectónicas.
  - Chip para parapléjicos: dispositivo electrónico implantado en los músculos y nervios produce una electroestimulación nerviosa que podría solucionar el problema de la parálisis.

- **Sistemas de Control de Entornos:** Son aquellos que, - *con fines comunicativos* - permiten la manipulación de dispositivos que ayudan a controlar un entorno.
  - guantes sensitivos.
  - dispositivos de seguimiento de movimientos oculares.
  - posicionadores de 3 D.

Capitulo 5. INTRODUCCIÓN A LOS DISPOSITIVOS MOVILES

5.1 ¿Qué es un dispositivo móvil?

Los dispositivos móviles son aquellos suficientemente pequeños para ser transportados y empleados durante su transporte. Normalmente se sincronizan con un sistema de sobremesa para actualizar aplicaciones y datos. Las características básicas de este tipo de dispositivos serian:

- Son aparatos pequeños,
- Con algunas capacidades de procesamiento,
- Móviles o no,
- Con conexión permanente o intermitente a una red,
- Con memoria limitada,
- Diseñados específicamente para una función, pero que pueden llevar a cabo otras más generales.
- Normalmente se asocian al uso individual de una persona, tanto en posesión como en operación, el cual puede adaptarlos a su gusto.
- La mayoría de estos aparatos pueden ser transportados en el bolsillo del propietario y
- Otros están integrados dentro de otros mayores, controlando su funcionalidad (como puede ser el ordenador integrado en una lavadora)

Algunas de las características que hacen que estos dispositivos sean diferentes de los ordenadores de sobremesa son los siguientes:

- Funcionalidad limitada.
- No necesariamente extensible y actualizable.
- En pocos años el usuario deberá cambiarlo.
- Más barato.
- Menos complicado en su manejo.
- Fácil de aprender su operación.
- No se requieren usuarios expertos.

Algunos de estos dispositivos son los siguientes:

- Paginadores.
- Comunicadores de bolsillo.
- Teléfonos con pantalla para Internet (Internet Screen Phones).
- Sistemas de navegación de automóviles.
- Sistemas de entretenimiento.
- Sistemas de televisión e Internet (WebTV).
- Teléfonos móviles.
- Organizadores y asistentes personales digitales (Personal Digital Assistant o PDA).

5.2 Personal Digital Assistant o PDA.

PDA, son las iniciales en Ingles de Personal Digital Assistant, cuya traducción en castellano seria (Asistente Personal Digital).

En la actualidad son potentes ordenadores personales de bolsillo (según versiones), aunque originariamente se iniciaron como agendas electrónicas, hoy en día son capaces de realizar la mayoría de tareas que realiza un ordenador convencional como agenda electrónica, diccionario, conversor de divisas y medidas, calculadora, acceso a Internet, reproductor de MP3 (ficheros de sonido que puede bajarse de Internet con muchas canciones), grabadora de sonidos (que incluso nos permite recordatorios sonoros), e incluso leer libros electrónicos, etc.

Salvando las lógicas diferencias de rendimiento, un PDA es parecido a un PC. Hay una placa base, un procesador, memoria RAM, memoria permanente (ROM) También hay un sistema operativo en el cual se ejecutan los programas que deseemos (juegos, agendas, hojas de cálculo, navegadores Web…). La forma de comunicarse con el PDA es la propia pantalla, que en todos los modelos actuales es táctil. El lápiz que incorpora el aparato se utiliza a modo de ratón. Para escribir podemos utilizar un teclado virtual que se muestra en la pantalla o bien hacer uso de la característica de reconocimiento de escritura. En esta última existen dos sistemas: trazado aprendido (cada carácter tiene asociado un movimiento del lápiz) y trazado natural (el PDA aprende nuestra forma de escribir).

En resumen, Los PDAs son ordenadores que se caracterizan por:

- Su tamaño: Caben en la palma de la mano y se pueden llevar en un bolsillo.
- Su interface de entrada/salida: La entrada de datos se realiza a través de una pantalla táctil en la que se puede escribir con un lápiz que incluye un punzón de plástico, siendo ésta capaz de reconocer letras y números manuscritos, la salida se realiza a través de la misma pantalla.
- Su interconectividad: los PDAs se conectan con otros PDAs, teléfonos móviles, etc.

- mediante una interconexión de infrarrojos y con un PC mediante un soporte que se conecta a través de un puerto serie o USB.
- La alimentación: mediante pilas o baterías.
- Y por tanto nos permiten utilizar las funciones de un ordenador allí donde las necesitemos.

Los PDAs tienen una serie de programas básicos:

- Agenda
- Calendario
- Listado de tareas
- Reloj y alarma
- Calculadora
- Cliente de e-mail
- Lector de e-books (Libros electrónicos, etc.), del tipo Adobe Acrobat Reader

Y se pueden incorporar otros como:

- Mando a distancia para la TV, video, etc.
- Diccionario
- Juegos de todo tipo y en particular de Ajedrez, incluso problemas de puzzles, para jugar a través de Internet, para consulta de base de datos de ajedrez
- etc.

Además los PDAs son capaces de sincronizar sus datos con los de programas como MS Outlook, MS Schedule, Lotus Notes, etc.
Tienen de 16MB a 64 MB de memoria suficientes para las necesidades de cualquier persona.

Todavía hay muchas personas que asocian PDA con agenda electrónica, pero como hemos indicado anteriormente en el párrafo anterior una PDA es mucho más una agenda electrónica, las versiones de gama alta son auténticos ordenadores de bolsillo.

- **Elementos Básicos de una PDA:** Los elementos principales que forman una PDA son los siguientes:

    - Procesador: Dos son los fabricantes que incorporan su tecnología a los procesadores de una PDA, Intel que acompaña a las PDAS Pocket PC de Microsoft y Texas Instruments para los procesadores incorporados en las PDAS de PALM OS, también están los ARM y los HItachi.
      Las velocidades máximas actuales alcanzan los 400MHz.

    - Memoria: Las memoria estándar que utilizan los dispositivos con Pc Pocket incluido es de 64 Mbs, aunque se pueden añadir más memoria con tarjetas de expansión.

- Teclado y Lápiz Óptico: No existe lo que entendemos como teclado convencional en la mayoría de las PDAs, a excepción de algunos modelos. Disponen de unos botones en su base para ejecutar las ordenes fundamentales y un "ratón" tipo Touch Pad para movernos por la pantalla con el puntero.

- Pantalla: Las pantallas TFT de alta resolución son las más usuales en los PDAs

- Batería: ahora todas incorporan batería de Litio-polímetro o Ion-litio, que han aumentado enormemente su duración, siendo ahora el tiempo medio de ejecución- duración sin necesidad de recargarlas de 8 a 12 horas.

- Tarjetas de Expansión.

Otros elementos existentes en una PDA son:

- Puertos: las formas habituales de conexión son el puerto USB y el puerto de infrarrojos, con los que podemos conectar el PDA con un ordenador para intercambiar datos (por ejemplo, para mantener sincronizada agenda, citas, tareas; para transferir documentos, música…).

- Bluetooht: es principal tecnología sin cables que actualmente incorpora cualquier dispositivo que se precie.

- Ranuras de expansión: los usuarios que demandan mayores capacidades en movilidad es importante que el PDA disponga de ranura de expansión para tarjetas Secure Digital, Compact Flash ó Multimedia Card, algunas de las que existen en el mercado. Con ellas se solucionan posibles limitaciones iniciales de los dispositivos que pueden ser ampliables paulatinamente.

Los PDAs también permiten la conexión a Internet a través del interface con el PC, es posible enviar y recibir e-mails, descargar e-books y periódicos.

Los términos para referirse a estos dispositivos móviles son los siguientes:

Término	Descripción
P.D.A (Personal Digital Assistant) Asistente Personal Agenda Electrónica	Nombre genérico para cualquier dispositivo portátil de reducido tamaño para llevar encima.
Handheld HPC	PDA con teclado tipo QWERTY incorporado, de mas peso y con mas 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 sistema operativo 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

5.3 Tipos de Personal Digital Assistant o P.D.A.

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 SecondEdition*.

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.

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

5.3.1 Newton.

El primer PDA fue desarrollado por Macintosh, y se llamaba Newton. Tenía pantalla táctil, sin teclado monocromática. Reconocía caracteres, y era más 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
-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 5.3 Palmome_treo650

5.1.3 EPOC-SYMBIAN.

Este sistema operativo, el *EPOC*, fue inventado y desarrollado por la empresa inglesa *Psion*, y sontípicos 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 más 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*.

Figura 5.4 Psion Revo

Figura 5.5Psion 5MX

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

Figura 5.6 Nokia 3660

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 Cassiopea 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.
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 útil, ya que según salía de la caja no se podía hacer mucho con ellas.

Figura 5.8 HandHelp HP Jornada 620 LX

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.

Figura 5.9. Ejemplo Windows Ce 2.11

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.

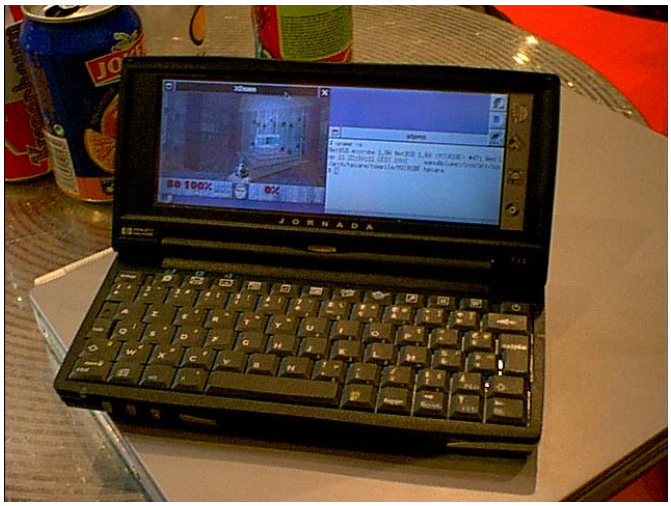


Figura 5.10 HP Jornada 720

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 *WindowsCE 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 cambiofundamental.

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*, surge 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 5.11 Compaq Ipaq serie 3600

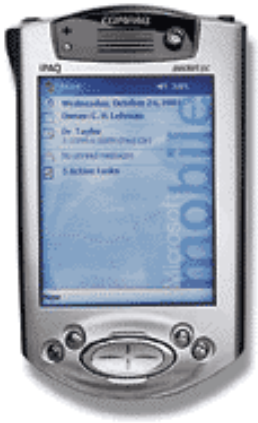


Figura 5.12Compaq Ipaq 3960

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*
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).

Figura 5.13

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

Actualmente existe una PDA, *el Sharp Zaurus*, que dispone de una versión reducida de LINUX. Este tipo de PDA cuenta con las siguientes características:

1. Contiene un teclado integrado que facilita la escritura de comandos y captura de datos .
2. Cuenta con una ranura para tarjetas de memoria secure digital lo que permite almacenamiento muy superior a la memoria con que originalmente viene este equipo.
3. También tiene una ranura para tarjetas compact flash en donde se puede instalar una tarjeta de red inalámbrica.
4. El entorno gráfico de esta PDA, Qtopia, cuenta con soporte de herramientas para desarrollo de aplicaciones en C++ de la compañía Trolltech.
5. Esta PDA incluye en el ROM un runtime de Personal Java lo que permite la ejecución de programas en Java.
6. Cuenta con el navegador Opera para que permite ejecutar aplicaciones en PHP.

The Sharp Zaurus PDA using Qt/Embedded

Figura 5.14 Sharp Zaurus

Capitulo 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 pocos 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 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.

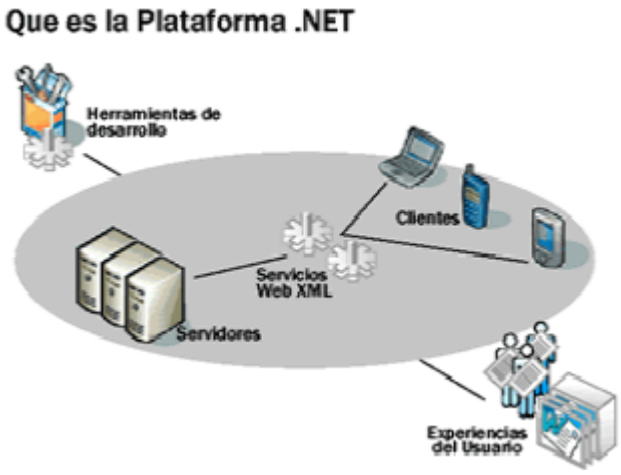


Figura 6.1 Plataforma.NET

Existen adicionalmente un conjunto de productos, que bajo la etiqueta de Servidores Empresariales para .NET (.NET Enterprise Server) 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

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 Componentes de .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 es un componente integral de Windows que admite la creación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.
- Ofrecer un entorno de ejecución de código que fomente la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos.
- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.
- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

.NET Framework contiene dos componentes principales: Common Language Runtime y la biblioteca de clases de .NET Framework. Common Language Runtime es el fundamento de la tecnología. El motor de tiempo de ejecución se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la interacción remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que fomentan su seguridad y solidez. De hecho, el concepto de administración de código es un principio básico del motor de tiempo de ejecución. El código destinado al motor de tiempo de ejecución se denomina *código administrado*, a diferencia del resto de código, que se conoce como *código no administrado*.

La biblioteca de clases, el otro componente principal de .NET Framework, es una completa colección orientada a objetos de tipos reutilizables que se pueden emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario (GUI) o de línea de comandos hasta las aplicaciones basadas en las innovaciones más recientes proporcionadas por ASP.NET, como los formularios Web Forms y los servicios Web XML.

.NET Framework puede alojarse en componentes no administrados que cargan Common Language Runtime en sus procesos e inician la ejecución de código administrado, con lo que se crea un entorno de software en el que se pueden utilizar características administradas y no administradas.

En .NET Framework no sólo se ofrecen varios hosts de motor de tiempo de ejecución, sino que también se admite el desarrollo de estos hosts por parte de terceros.

Por ejemplo, ASP.NET aloja el motor de tiempo de ejecución para proporcionar un entorno de servidor escalable para el código administrado. ASP.NET trabaja directamente con el motor de tiempo de ejecución para habilitar aplicaciones de ASP.NET y servicios Web XML, que se tratan más adelante en este tema.

Internet Explorer es un ejemplo de aplicación no administrada que aloja el motor de tiempo de ejecución (en forma de una extensión de tipo MIME). Al usar Internet Explorer para alojar el motor de tiempo de ejecución, puede incrustar componentes administrados o controles de Windows Forms en documentos HTML. Al alojar el motor de tiempo de ejecución de esta manera se hace posible el uso de código móvil administrado (similar a los controles de Microsoft ActiveX), pero con mejoras significativas que sólo el código administrado puede ofrecer, como la ejecución con confianza parcial y el almacenamiento aislado de archivos.

En la ilustración siguiente se muestra la relación de Common Language Runtime y la biblioteca de clases con las aplicaciones y el sistema en su conjunto. En la ilustración se representa igualmente cómo funciona el código administrado dentro de una arquitectura mayor**.**

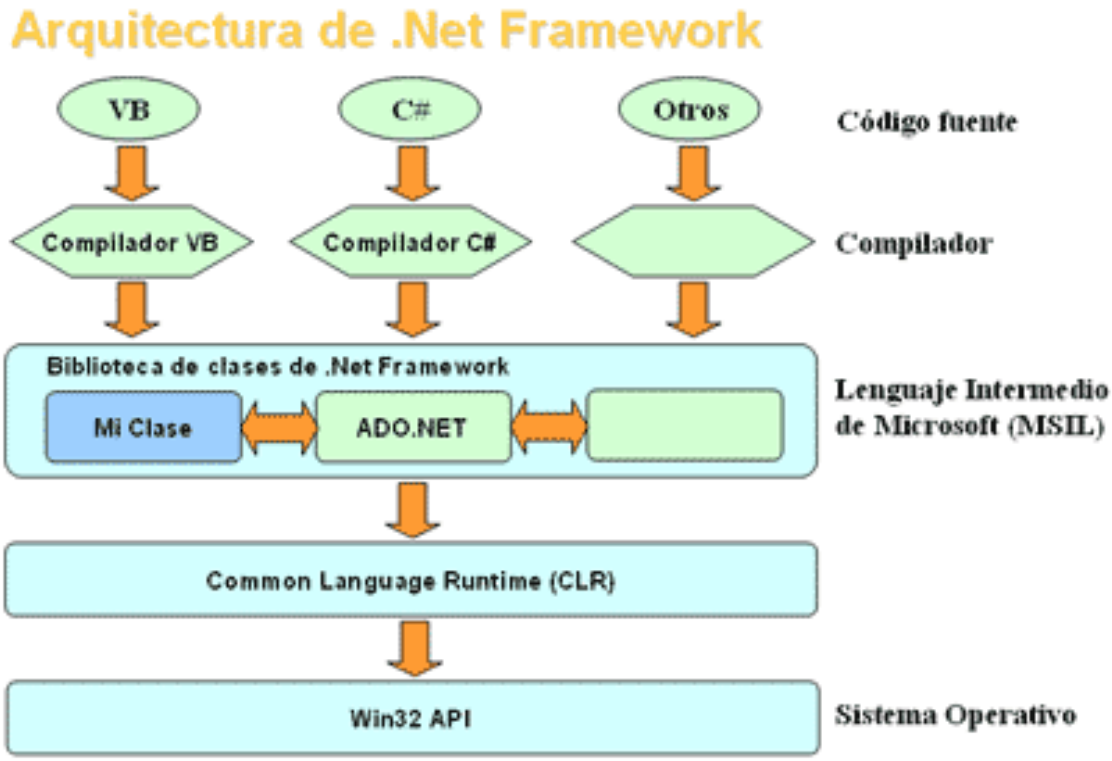


Figura 6.2 .NET Framework

6.4.1 El CLR, Common Lenguaje Runtime

El **Common Language Runtime** (**CLR**) es el núcleo de la plataforma .NET.
Common Language Runtime administra la memoria, ejecución de subprocesos, ejecución de código, comprobación de la seguridad del código, compilación y demás servicios del sistema.
Estas características son intrínsecas del código administrado que se ejecuta en Common Language Runtime.

- **Modelo de programación consistente:** A todos los servicios y facilidades ofrecidos por el CLR se accede de la misma forma: a través de un modelo de programación orientado a objetos

- **Modelo de programación sencillo:** Con el CLR desaparecen muchos elementos complejos incluidos en los sistemas operativos actuales (registro de Windows, GUIDs, HRESULTS, IUnknown, etc.) El CLR no es que abstraiga al programador de estos conceptos, sino que son conceptos que no existen en la plataforma .NET

- **Ejecución multiplataforma:** El CLR actúa como una máquina virtual, encargándose de ejecutar las aplicaciones diseñadas para la plataforma .NET. Es decir, cualquier plataforma para la que exista una versión del CLR podrá ejecutar cualquier aplicación .NET.

- **Integración de lenguajes:** Desde cualquier lenguaje para el que exista un compilador que genere código para la plataforma .NET es posible utilizar código generado para la misma usando cualquier otro lenguaje tal y como si de código escrito usando el primero se tratase.

- **Gestión de memoria:** El CLR incluye un **recolector de basura** que evita que el programador tenga que tener en cuenta cuándo ha de destruir los objetos que dejen de serle útiles. Este recolector es una aplicación que se activa cuando se quiere crear algún objeto nuevo y se detecta que no queda memoria libre para hacerlo, caso en que el recolector recorre la memoria dinámica asociada a la aplicación, detecta qué objetos hay en ella que no puedan ser accedidos por el código de la aplicación, y los elimina para limpiar la memoria de “objetos basura” y permitir la creación de otros nuevos

- **Seguridad de tipos:** El CLR facilita la detección de errores de programación difíciles de localizar comprobando que toda conversión de tipos que se realice durante la ejecución de una aplicación .NET se haga de modo que los tipos origen y destino sean compatibles.

- **Aislamiento de procesos:** El CLR asegura que desde código perteneciente a un determinado proceso no se pueda acceder a código o datos pertenecientes a otro, lo que evita errores de programación muy frecuentes e impide que unos procesos puedan atacar a otros.

- **Tratamiento de excepciones:** En el CLR todos los errores que se puedan producir durante la ejecución de una aplicación se propagan de igual manera: mediante excepciones.
  El CLR permite que excepciones lanzadas desde código para .NET escrito en un cierto lenguaje se puedan capturar en código escrito usando otro lenguaje, e incluye mecanismos de depuración que pueden saltar desde código escrito para .NET en un determinado lenguaje a código escrito en cualquier otro.

- **Soporte multihilo:** El CLR es capaz de trabajar con aplicaciones divididas en múltiples hilos de ejecución que pueden ir evolucionando por separado en paralelo o intercalándose, según el número de procesadores de la máquina sobre la que se ejecuten. Las aplicaciones pueden lanzar nuevos hilos, destruirlos, suspenderlos por un tiempo o hasta que les llegue una notificación, enviarles notificaciones, sincronizarlos, etc.

- **Distribución transparente:** El CLR ofrece la infraestructura necesaria para crear objetos remotos y acceder a ellos de manera completamente transparente a su localización real, tal y como si se encontrasen en la máquina que los utiliza.

- **Seguridad avanzada:** El CLR proporciona mecanismos para restringir la ejecución de ciertos códigos o los permisos asignados a los mismos según su procedendecia o el usuario que los ejecute.

- **Interoperabilidad con código antiguo:** El CLR incorpora los mecanismos necesarios para poder acceder desde código escrito para la plataforma .NET a código escrito previamente a la aparición de la misma y, por tanto, no preparado para ser ejecutando dentro de ella. Estos mecanismos permiten tanto el acceso a objetos COM como el acceso a funciones sueltas de DLLs preexistentes (como la API Win32)

Como se puede deducir de las características comentadas, el CLR lo que hace es gestionar la ejecución de las aplicaciones diseñadas para la plataforma .NET. Por esta razón, al código de estas aplicaciones se le suele llamar **código gestionado**, y al código no escrito para ser ejecutado directamente en la plataforma .NET se le suele llamar **código no gestionado**.

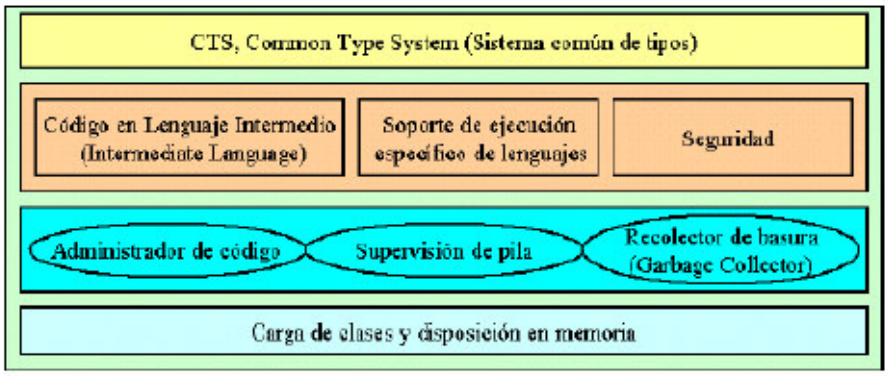


Figura 6.3 Esquema de componentes dentro del CLR.

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

Todos los lenguajes de programación que cumplen las normas de .NET tienen muchas cosas en común, una de ellas es el conjunto de tipos de datos. Hay que destacar que estos tipos de datos están implementados como clases, de manera que una variable declarada de un tipo determinado, tendrá la capacidad de usar tanto los métodos como las propiedades que pertenezcan a la clase del tipo de dato.

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**: los tipos por valor almacenan datos a los que se puede acceder de forma directa, a su vez dentro de esta categoría encontramos mas subcategorías como los tipos nativos de .NET, los tipos de datos creados por el programador y los enumerados. Los tipos por valor no pueden tener valores nulos
- **Tipos por referencia**: Los tipos creados por referencia almacenan la dirección de memoria en la que se encuentra un dato determinado de manera que usaremos esa dirección de memoria para acceder de forma indirecta al dato. Los tipos por referencia se dividen en varios subgrupos como son las clases propias de la plataforma, interfaces, clases creadas por el programador, etc.

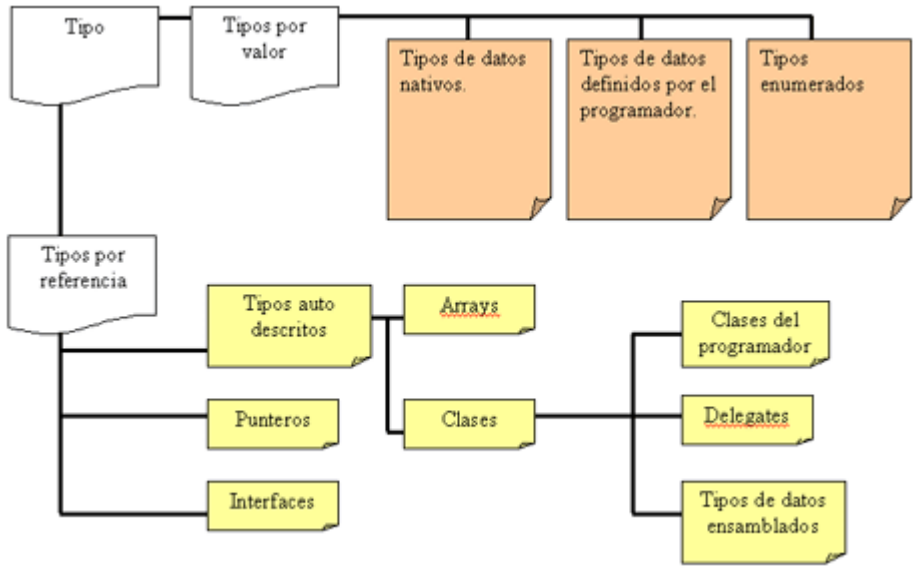


Figura 6.4 Tipos de Datos en .NET

6.4.6 La disposición de los datos en la memoria

Cuando ejecutamos una aplicación es necesario que los datos se sitúen en la memoria del ordenador, la cual esta divida en dos partes, una llamada Stack, de pequeño tamaño pero de un acceso muy rápido y otra llamada Heap que cuenta con un mayor tamaño pero con una velocidad de acceso inferior.

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 creamos tipos por valor, el valor de la variable de este tipo se almacena en el Stack, si asignamos una variable de estas características a otra, se crea una copia en el Stack. Al destruir un tipo por valor, se destruye también el valor que se guardo en el Stack.

Cuando creamos un tipo por referencia, en realidad lo que guardamos en el Heap es una dirección de memoria que apunta a un valor, pero no al valor en si mismo. Si asignamos una variable que contiene un tipo por referencia a otra variable, se dice que ambas se refieren al mismo valor. Los tipos por referencia si pueden contener valores nulos.

6.4.7 Metadata (metadatos).

Los metadatos son información binaria que describe un programa, almacenada en un archivo ejecutable portable (PE) de Common Language Runtime o en memoria. Cuando se compila el código en un archivo PE, los metadatos se insertan en una parte del archivo, mientras que el código se convierte en lenguaje intermedio de Microsoft (MSIL) y se inserta en otra parte del archivo. Cada tipo y miembro definido, o al que se hace referencia, en un módulo o ensamblado se describe en los metadatos. Cuando se ejecuta código, el motor de tiempo de ejecución carga los metadatos en la memoria y hace referencia a ellos para obtener información acerca de las clases, miembros, herencia, etc., del código.

Los metadatos describen todos los tipos y miembros definidos en el código mediante un lenguaje neutro. Los metadatos almacenan la siguiente información:

- Descripción del ensamblado
  - Identidad (nombre, versión, referencia cultural, clave pública).
  - Los tipos que se exportan.
  - Otros ensamblados de los que depende éste.
  - Permisos de seguridad que hay que ejecutar.

- Descripción de los tipos.
  - Nombre, visibilidad, clase base e interfaces implementadas.
  - Miembros (métodos, campos, propiedades, eventos, tipos anidados).

- Atributos.
  - Elementos descriptivos adicionales que modifiquen los tipos y los miembros.

Los metadatos proporcionan las siguientes ventajas principales:

- Archivos autodescriptivos: Los módulos y ensamblados de Common Language Runtime son autodescriptivos. Los metadatos de un módulo contienen todo lo necesario para interactuar con otro módulo. Los metadatos proporcionan automáticamente la

funcionalidad del IDL en COM, lo que permite usar un archivo para la definición y la implementación.

- Interoperabilidad de lenguajes y diseño más sencillo, basado en el componente: Los metadatos proporcionan toda la información necesaria sobre el código compilado para derivar clases de archivos PE escritos en otro lenguaje.

- Atributos: .NET Framework permite declarar tipos específicos de metadatos, denominados atributos, en el archivo compilado. Los atributos se encuentran en todo .NET Framework y se usan para controlar más minuciosamente el comportamiento del programa en tiempo de ejecución.

6.4.8 Interpolabilidad entre lenguajes.

La interoperabilidad entre lenguajes es la posibilidad de que el código interactúe con código escrito en un lenguaje de programación diferente. La interoperabilidad entre lenguajes puede ayudar a maximizar la reutilización de código y, por tanto, puede mejorar la eficacia del proceso de programación.

Common Language Runtime ofrece la base necesaria para la interoperabilidad entre lenguajes al especificar e imponer un sistema de tipos común, y al suministrar metadatos. Debido a que todos los lenguajes dirigidos a Common Language Runtime siguen las reglas del *sistema de tipos común* para definir y utilizar los tipos, la utilización de tipos es coherente entre todos los lenguajes.

.NET ofrece los siguientes lenguajes:
- VB.NET
- C#
- C++ con Extensiones Administradas
- JScript.NET

El código administrado se beneficia de que el CLR admita la interoperabilidad entre lenguajes de las maneras siguientes:

- Los tipos pueden heredar la implementación de otros tipos, pasar objetos a los métodos de otro tipo y llamar a métodos definidos para otros tipos, sea cual sea el lenguaje en que se implementen los tipos.

- Los depuradores, generadores de perfiles u otras herramientas deben reconocer un solo entorno, es decir, MSIL (Microsoft intermediate language, Lenguaje intermedio de Microsoft) y los metadatos de Common Language Runtime, para poder ser compatibles con cualquier lenguaje de programación dirigido al motor de tiempo de ejecución.

- El control de excepciones es coherente entre todos los lenguajes. El código puede iniciar una excepción en un lenguaje y esa excepción puede ser recibida y reconocida por un objeto escrito en otro lenguaje.

6.4.9 El CLS (Common Language Specification).

Para poder interactuar completamente con otros objetos, sea cual sea el lenguaje en que se hayan implementado, los objetos deben exponer a los llamadores sólo aquellas características que sean comunes para todos los lenguajes con los que deben interoperar.
Por este motivo, se ha definido Common Language Specification (CLS), que es un conjunto de características de lenguaje básicas requeridas por la mayoría de las aplicaciones.

Las reglas de CLS definen un subconjunto del *sistema de tipos común* es decir, todas las reglas aplicables al sistema de tipos común se aplican a CLS, excepto cuando se definan reglas más estrictas en CLS.
Esto tiene varias finalidades, que describimos a continuación:

- CLS ayuda a mejorar y garantizar la interoperabilidad entre lenguajes mediante la definición de un conjunto de características en las que se pueden basar los programadores y que están disponibles en una gran variedad de lenguajes.
- CLS también establece los requisitos de compatibilidad con CLS; estos requisitos permiten determinar si el código administrado cumple la especificación CLS y hasta qué punto una herramienta dada admite la programación de código administrado que utilice las características de CLS.
- Si un componente sólo utiliza las características de CLS en la API que expone a otro código (incluidas las clases derivadas), se garantiza que se puede obtener acceso al componente desde cualquier lenguaje de programación que admita CLS. Los componentes que cumplen las reglas de CLS y usan sólo las características incluidas en CLS se conocen como componentes compatibles con CLS.

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.
El proceso de ejecución administrada incluye los pasos siguientes:

1. Elegir un compilador: Para obtener los beneficios que proporciona Common Language Runtime, se deben utilizar uno o más compiladores de lenguaje orientados al tiempo de ejecución.
2. Compilar el código a Lenguaje intermedio de Microsoft (MSIL): La compilación convierte el código fuente en MSIL y genera los metadatos requeridos.
3. Compilar MSIL a código nativo
   En tiempo de ejecución, un compilador Just-In-Time (JIT) convierte MSIL en código nativo. Durante esta compilación, el código debe pasar un proceso de comprobación que examina el MSIL y los metadatos para averiguar si el código garantiza la seguridad de tipos.
4. Ejecutar código: Common Language Runtime proporciona la infraestructura que permite

que la ejecución tenga lugar, así como una amplia gama de servicios que se pueden utilizar durante la ejecución.
.

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 El Lenguaje Intermedio de Microsoft (MSIL).

Cuando se compila a código administrado, el compilador convierte el código fuente en Lengua intermedio de Microsoft (MSIL), que es un conjunto de instrucciones independiente de la CPU que se pueden convertir de forma eficaz en código nativo. MSIL incluye instrucciones para cargar, almacenar, inicializar y llamar a métodos en los objetos, así como instrucciones para operaciones lógicas y aritméticas, flujo de control, acceso directo a la memoria, control de excepciones y otras operaciones. Antes de poder ejecutar código, se debe convertir MSIL al código específico de la CPU, normalmente mediante un compilador Just-It-Time (JIT). Common Language Runtime proporciona uno o varios compiladores JIT para cada arquitectura de equipo compatible, por lo que se puede compilar y ejecutar el mismo conjunto de MSIL en cualquier arquitectura compatible.

Cuando el compilador produce MSIL, también genera metadatos. Los metadatos describen los tipos que aparecen en el código, incluidas las definiciones de los tipos, las firmas de los miembros de tipos, los miembros a los que se hace referencia en el código y otros datos que el motor de tiempo de ejecución utiliza en tiempo de ejecución. El lenguaje intermedio de Microsoft (MSIL) y los metadatos se incluyen en un archivo ejecutable portable (PE), que se basa y extiende el PE de Microsoft publicado y el formato Common Object File Format (COFF) utilizado tradicionalmente para contenido ejecutable. Este formato de archivo, que contiene código MSIL o código nativo así como metadatos, permite al sistema operativo reconocer imágenes de Common Language Runtime.

La presencia de metadatos junto con el Lenguaje intermedio de Microsoft (MSIL) permite crear códigos autodescriptivos, con lo cual las bibliotecas de tipos y el Lenguaje de definición de interfaces (IDL) son innecesarios. El motor de tiempo de ejecución localiza y extrae los metadatos del archivo cuando son necesarios durante la ejecución.

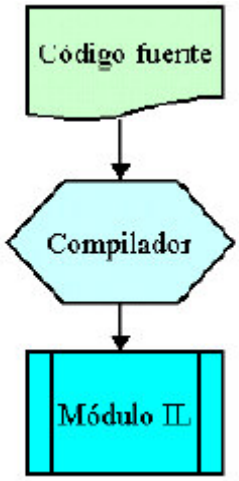


Figura 6.5 Obtención de lenguaje intermedio a partir de código fuente

6.4.14 Compilar MSIL a codigo nativo.

Para poder ejecutar el lenguaje intermedio de Microsoft (MSIL), primero se debe convertir éste, mediante un compilador Just-In-Time (JIT) de .NET Framework, a código nativo, que es el código específico de la CPU que se ejecuta en la misma arquitectura de equipo que el compilador JIT. Common Language Runtime proporciona un compilador JIT para cada arquitectura de CPU compatible, por lo que los programadores pueden escribir un conjunto de MSIL que se puede compilar mediante un compilador JIT y ejecutar en equipos con diferentes arquitecturas. No obstante, el código administrado sólo se ejecutará en un determinado sistema operativo si llama a las API nativas específicas de la plataforma o a una biblioteca de clases específica de la plataforma.

Como parte de la compilación MSIL en código nativo, el código debe pasar un proceso de comprobación, a menos que el administrador haya establecido una directiva de seguridad que permita al código omitir esta comprobación.

En esta comprobación se examina el MSIL y los metadatos para determinar si el código garantiza la seguridad de tipos, lo que significa que el código sólo tiene acceso a aquellas ubicaciones de la memoria para las que está autorizado.
La seguridad de tipos ayuda a aislar los objetos entre sí y, por tanto, ayuda a protegerlos contra daños involuntarios o maliciosos. Además, garantiza que las restricciones de seguridad sobre el código se aplican con toda certeza.

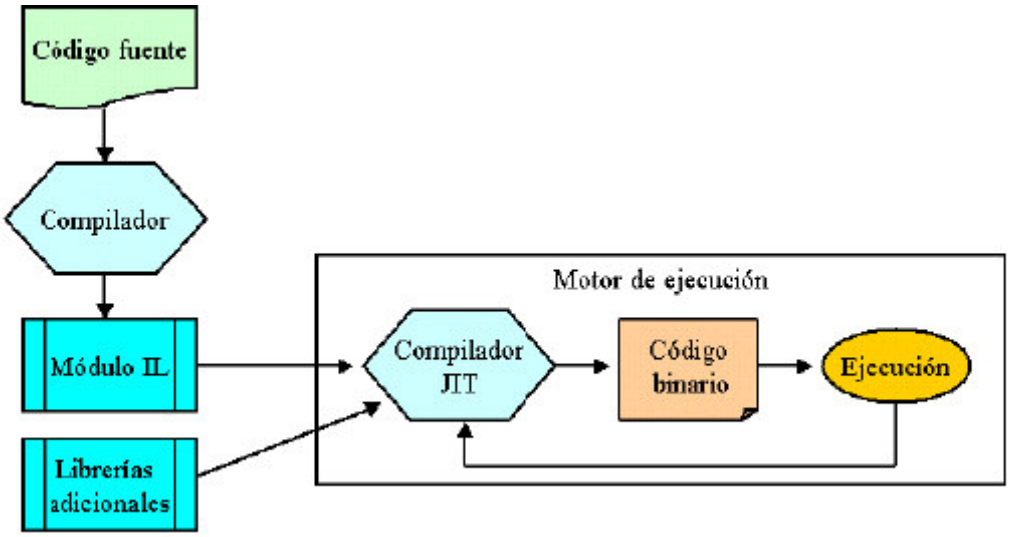


Figura 6.6 Proceso de compilación instantánea de codigo en MSIL

6.4.15 Ejecución de codigo.

Common Language Runtime proporciona la infraestructura que permite que la ejecución administrada tenga lugar, así como una gran cantidad de servicios que se pueden utilizar durante la ejecución. Para poder ejecutar un método, primero se debe compilar a código específico del procesador. Los métodos para los que se genera el Lenguaje intermedio de Microsoft (MSIL) se compilan mediante un compilador JIT cuando se les llama por primera vez y, a continuación, se ejecutan. La próxima vez que se ejecuta el método, se ejecuta el código nativo existente resultante de la compilación JIT. El proceso de compilación JIT y, a continuación, la ejecución de código se repite hasta completar la ejecución.

Durante la ejecución, el código administrado recibe servicios como la recolección de elementos no utilizados, seguridad, interoperabilidad con código no administrado, compatibilidad de depuración entre lenguajes diferentes y compatibilidad mejorada con el control de versiones y la implementación.

6.4.16 Administración de memoria automática.

La administración de memoria automática es uno de los servicios que proporciona Common Language Runtime durante *la ejecución administrada*. El recolector de elementos no utilizados de Common Language Runtime administra la asignación y liberación de la memoria de una aplicación. Esto significa que los programadores no tienen que escribir código para realizar tareas de administración de memoria al programar aplicaciones administradas. La administración automática de la memoria puede eliminar problemas frecuentes, como olvidar liberar un objeto y causar una pérdida de memoria, o intentar tener acceso a la memoria de un objeto que ya se ha liberado. A continuación se describe cómo asigna y libera memoria el recolector de elementos no utilizados.

- **Asignar memoria**
  Cuando se inicializa un nuevo proceso, el CLR reserva una región contigua de espacio de direcciones para el proceso. Este espacio de direcciones reservado se denomina montón administrado. El montón administrado mantiene un puntero a la dirección a la que se asignará el siguiente objeto del montón.

- **Liberar memoria**
  El motor de optimización del recolector de elementos no utilizados determina cuál es el mejor momento para realizar una recolección basándose en las asignaciones realizadas. Cuando el recolector lleva a cabo una recolección, libera la memoria de los objetos que ya no usa la aplicación.

- **Generaciones y rendimiento**
  Para optimizar el rendimiento del recolector de elementos no utilizados, el montón administrado se divide en tres generaciones: 0, 1 y 2.
  El recolector de elementos no utilizados del motor de tiempo de ejecución guarda los nuevos objetos en la generación 0. Los objetos creados en las primeras etapas de la duración de la aplicación y que sobreviven a las recolecciones se promueven y se almacenan en las generaciones 1 y 2.

- **Liberar memoria para recursos no administrados**
  En el caso de la mayoría de los objetos creados por la aplicación, puede utilizar el recolector de elementos no utilizados para realizar automáticamente las tareas de administración de memoria.

6.4.17 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.

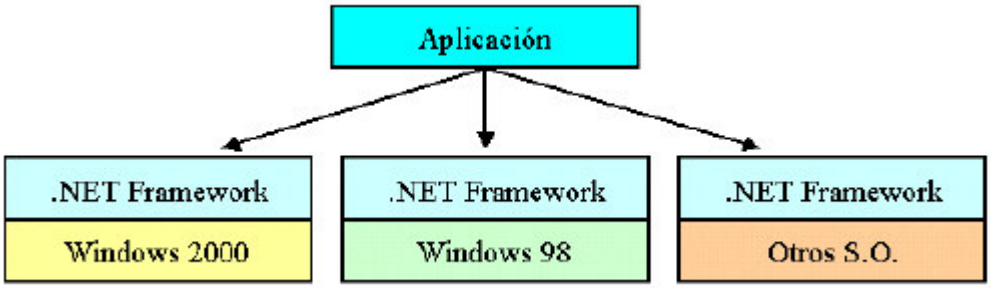


Figura 6.7 Una misma aplicación se ejecuta en distintos sistemas a través de .NET Framework

6.4.18 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).

Los dominios de aplicación constituyen una unidad de procesamiento más segura y versátil que puede utilizar Common Language Runtime para proporcionar el aislamiento entre las aplicaciones. Las aplicaciones se aíslan porque las direcciones de memoria son específicas de cada proceso; un puntero de memoria pasado de un proceso a otro no se puede utilizar de ninguna manera coherente en el proceso de destino. Tampoco se pueden realizar llamadas directas entre dos procesos. En su lugar, se deben utilizar servidores proxy, que proporcionan un nivel de direccionamiento indirecto.

En un solo proceso se pueden ejecutar varios dominios de aplicación con el mismo nivel de aislamiento que existiría en procesos independientes, sin incurrir en la sobrecarga adicional que supone realizar llamadas entre procesos o cambiar de un proceso a otro. La posibilidad de ejecutar múltiples aplicaciones en un solo proceso aumenta la escalabilidad del servidor de manera importante.
Aislar las aplicaciones es también importante para la seguridad de las mismas. Por ejemplo, en un solo proceso de explorador se pueden ejecutar controles de varias aplicaciones Web de tal forma que no puedan tener acceso a los datos y recursos de los demás controles.
Éstas son las ventajas del aislamiento que ofrecen los dominios de aplicación:

- Los errores de una aplicación no pueden afectar a otras aplicaciones: el uso de dominios de aplicación garantiza que el código que se ejecute en un dominio no afectará a las demás aplicaciones del proceso.

- Es posible detener aplicaciones concretas sin detener todo el proceso: El uso de dominios de aplicación permite descargar el código que se ejecuta en una sola aplicación.

- El código que se ejecuta en una aplicación no puede tener acceso directo al código o a los recursos de otra aplicación: Common Language Runtime impone este aislamiento al impedir que se realicen llamadas directas entre objetos de dominios de aplicación diferentes.

- La aplicación en la que se ejecuta el código establece el comportamiento del mismo: El dominio de aplicación proporciona valores de configuración tales como las directivas de versión de la aplicación, la ubicación de los ensamblados remotos a los que tiene acceso e información sobre dónde encontrar los ensamblados que se cargan en el dominio.

- El dominio de aplicación en el que se ejecuta el código puede controlar los permisos que se conceden al código.

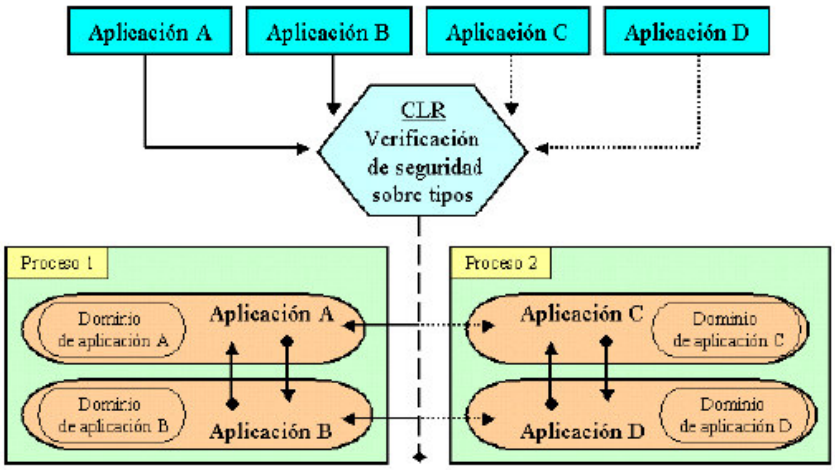


Figura 6.8 Carga de aplicaciones y creación de dominios de aplicación en procesos.

6.4.19 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.

6.4.20 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 acceder a 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 explicados 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.

6.4.21 Biblioteca de clases en .NET Framework.

La biblioteca de clases de .NET Framework es una colección de tipos reutilizables que se integran estrechamente con Common Language Runtime. La biblioteca de clases está orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Esto ocasiona que los tipos de .NET Framework sean sencillos de utilizar y reduce el tiempo asociado con el aprendizaje de las nuevas características de .NET Framework. Además, los componentes de terceros se pueden integrar sin dificultades con las clases de .NET Framework.

Por ejemplo, las clases de colección de .NET Framework implementan un conjunto de interfaces que puede usar para desarrollar sus propias clases de colección. Éstas se combinarán fácilmente con las clases de .NET Framework.

Como en cualquier biblioteca de clases orientada a objetos, los tipos de .NET Framework permiten realizar diversas tareas de programación comunes, como son la administración de cadenas, recopilación de datos, conectividad de bases de datos y acceso a archivos. Además de estas tareas habituales, la biblioteca de clases incluye tipos adecuados para diversos escenarios de desarrollo especializados. Por ejemplo, puede utilizar .NET Framework para desarrollar los siguientes tipos de aplicaciones y servicios:

- Aplicaciones de consola
- Aplicaciones GUI de Windows (Windows Forms)
- Aplicaciones de ASP.NET
- Servicios Web XML
- Servicios de Windows

Por ejemplo, las clases de Windows Forms son un conjunto completo de tipos reutilizables que simplifican enormemente el desarrollo de interfaces GUI para Windows. Si escribe una aplicación Web Forms de ASP.NET, puede utilizar las clases de Web Forms.

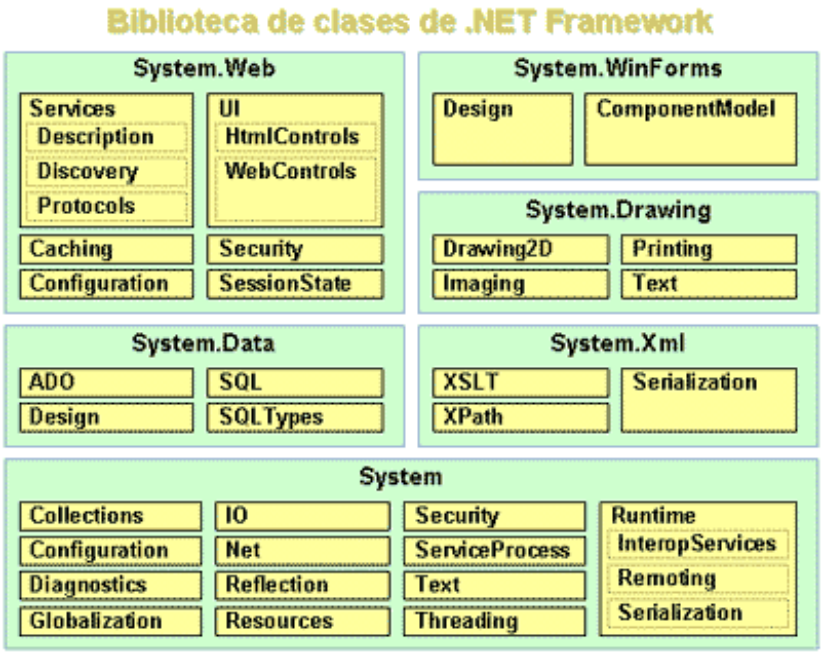


Figura 6.9 Biblioteca de clases en .NET Framework

6.4.22 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.

Los ensamblados son una parte fundamental de la programación con .NET Framework. Un ensamblado realiza las funciones siguientes:

- Contiene el código que ejecuta Common Language Runtime. El código del lenguaje intermedio de Microsoft (MSIL) de un archivo ejecutable portable (PE) no se ejecuta si no tiene asociado un manifiesto de ensamblado. Recuerde que cada ensamblado sólo puede tener un punto de entrada.

- Crea un límite de seguridad. Un ensamblado es la unidad en la que se solicitan y conceden los permisos.

- Crea un límite de tipos. La identidad de cada tipo incluye el nombre del ensamblado en que reside. Por ello, un tipo MyType cargado en el ámbito de un ensamblado no es igual que un tipo denominado MyType cargado en el ámbito de otro ensamblado.

- Crea un límite de ámbito de referencia. El manifiesto del ensamblado contiene los metadatos del ensamblado que se utilizan para resolver tipos y satisfacer las solicitudes de recursos. Especifica los tipos y recursos que se exponen fuera del ensamblado. El manifiesto también enumera otros ensamblados de los que depende.

- Forma un límite de versión. El ensamblado es la unidad versional más pequeña de Common Language Runtime; todos los tipos y recursos del mismo ensamblado pertenecen a la misma versión.

- El manifiesto del ensamblado describe las dependencias de versión que se especifiquen para los ensamblados dependientes.

- Crea una unidad de implementación. Cuando se inicia una aplicación, sólo deben estar presentes los ensamblados a los que llama la aplicación inicialmente. Los demás ensamblados, como los recursos de localización o los ensamblados que contengan clases de utilidad, se pueden recuperar a petición. De este modo, se puede mantener la simplicidad y transparencia de las aplicaciones la primera vez que se descargan.

- Es la unidad que permite la ejecución simultánea.

Los ensamblados pueden ser estáticos o dinámicos.

- Los ensamblados estáticos: pueden incluir tipos de .NET Framework (interfaces y clases), así como recursos para el ensamblado (mapas de bits, archivos JPEG, archivos de recursos, etc.). Los ensamblados estáticos se almacenan en el disco, en archivos ejecutables portables PE

- Los ensamblados dinámicos: También se puede utilizar .NET Framework para crearlos, se ejecutan directamente desde la memoria y no se guardan en el disco antes de su ejecución. Los ensamblados dinámicos se pueden guardar en el disco una vez que se hayan ejecutado.

6.4.23 La solución de los Ensamblados.

Los ensamblados están diseñados para simplificar la implementación de las aplicaciones y para solucionar los posibles problemas de versiones de las aplicaciones basadas en componentes.

El usuario final y los programadores conocen perfectamente los problemas de versiones e implementación que surgen hoy en día con los sistemas basados en componentes.

Algunos usuarios finales se han visto frustrados al instalar una nueva aplicación en su equipo y ver cómo deja de funcionar otra aplicación que ya tenían instalada. Muchos programadores han estado horas y horas intentando mantener la coherencia de todas las entradas del Registro necesarias para activar una clase COM.

Con el uso de ensamblados en .NET Framework se han resuelto muchos problemas de implementación. Debido a que son componentes autodescriptivos que no dependen de las entradas del Registro, los ensamblados permiten instalar las aplicaciones sin problemas. También simplifican la desinstalación y replicación de las aplicaciones.

Con el objetivo de solucionar los problemas de las versiones, así como los problemas restantes que desembocan en conflictos de DLL, el CLR utiliza ensamblados para los siguientes fines:

- Permitir a los programadores especificar reglas de versiones entre distintos componentes de software.
- Proporcionar la infraestructura para que se cumplan las reglas de versiones.
- Proporcionar la infraestructura que permita que varias versiones de un componente se puedan ejecutar simultáneamente, lo que se conoce como ejecución simultánea.

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 incorrectamente 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

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.

En conclusión podríamos decir que Visual Studio .NET es la herramienta de segunda generación de Microsoft para crear e implementar software seguro y eficaz para la plataforma Microsoft .NET, que permite a los programadores:

- Crear aplicaciones basadas en Windows rápidas y eficaces.
- Crear aplicaciones para Pocket PC rápidas y eficaces.
- Crear aplicaciones Web sofisticadas y seguras.
- Crear aplicaciones Web inteligentes, sofisticadas y seguras para dispositivos móviles.
- Utilizar servicios Web XML en cualquiera de las aplicaciones mencionadas.
- Evitar conflictos entre archivos .DLL.
- Eliminar los costosos problemas de implementación y mantenimiento de las aplicaciones.

Visual Studio .NET es el único entorno de desarrollo creado exclusivamente para permitir la integración con servicios Web XML. Al hacer posible que las aplicaciones compartan datos a través de Internet, los servicios Web XML permiten a los programadores ensamblar aplicaciones a partir de código nuevo y existente, independientemente de la plataforma, el lenguaje de programación o el modelo de objetos.

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 Software
Cierto tipo de proyectos y funciones de Visual Studio precisan de la instalación de componentes de software específicos, que aparecen como opcionales durante la configuración, antes de poder utilizar dicho proyecto o función. Algunos de estos componentes deben instalarse en el equipo de desarrollo, mientras que otros pueden instalarse en un equipo remoto.

Desea	Windows 2000	Windows XP, Windows Server 2003 o posterior	Windows NT4, Windows 98, Windows Me, Windows XP Home
Desarrollar aplicaciones Web ASP y servicios Web XML	Servicios de Internet Information Server (IIS)	IIS	No soportado
Compilar código relacionado con Microsoft Windows Message Queuing (MSMQ)	Servicios de Message Queue Server	Servicios de Message Queue Server	No soportado
Depurar código en equipos remotos	Depurador remoto de Visual Studio	Depurador remoto de Visual Studio	Depurador remoto de Visual Studio
Usar el control de código fuente para las versiones de procedimientos almacenados	Versiones de procedimientos almacenados de Visual Studio 6.0 Visual SourceSafe Microsoft SQL Server	Versiones de procedimientos almacenados de Visual Studio 6.0 Visual SourceSafe Microsoft SQL Server	No soportado

Figura 6.10 Requisitos software para instalación de Visual Studio.NET

Requisitos Hardware.
En este apartado se describen los requisitos de hardware para instalar las distintas ediciones de Visual Studio .NET, las ediciones Standard disponibles para los lenguajes de programación de Visual Studio .NET y sugerencias para mejorar el rendimiento de Visual Studio .NET.

El equipo en el que instale la edición de Visual Studio .NET debe cumplir los requisitos siguientes:

Requisito	Enterprise Architect	Enterprise Developer	Professional	Academic
Procesador	PC con procesador de tipo Pentium II, 450 MHz Se recomienda: tipo Pentium III, 600 MHz	Igual	Igual	Igual
RAM	Windows 2000 Professional: 96 MB; Windows 2000 Server: 192 MB; Windows XP Home: 96 MB; Windows XP Professional y Windows Server 2003: 192 MB *Se recomienda*: 128 MB para 2000 Professional, 256 MB para 2000 Server, 160 MB para XP Home y 256 MB para XP Professional y Server 2003	Igual	Igual	Igual
Espacio disponible en el disco duro	900 MB en la unidad del sistema y 4,1 GB en la unidad de instalación	Igual	Igual	Igual
Sistema Operativo	Windows® 2000, Windows XP, Windows Server 2003 o Windows NT 4.0[3],[4],[5]	Igual	Igual	Igual
Unidad de CD-ROM o DVD-ROM	Requerida	Requerida	Requerida	Requerida
Video	800 x 600, 256 colores Se recomienda: color de alta densidad, 16 bits	Igual	Igual	Igual
Mouse (Ratón)	Microsoft Mouse o dispositivo señalador compatible	Igual	Igual	Igual

Figura 6.11 Requisitos hardware para instalación de Visual Studio.NET

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.

Control	Descripción
Button	botón de comando simple
ComboBox	lista desplegable de elementos
ContextMenu	implementa un menú contextual
DataGrid	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

CheckBox	casilla de verificación habitual
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

Figura 6.12. Controles incluidos con .NET Compact Framework

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 aplicación 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 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.

6.7.4 Serialización binaria.

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

6.7.5 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.

6.7.6 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.

6.7.7 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.

6.7.8 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.

6.7.9 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.

6.8 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.

6.8.1Adiciones 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.

C# (leído en inglés “C Sharp” y en español “C Almohadilla”) es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg, éste último también conocido por haber sido el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi.

Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el **lenguaje nativo de .NET**

La sintaxis y estructuración de C# es muy parecida a la de C++ o Java, puesto que la intención de Microsoft es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su aprendizaje a los desarrolladores habituados a ellos. Sin embargo, su sencillez y el alto nivel de productividad son comparables con los de Visual Basic

En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la

BCL usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el *.NET Framework SDK*

Características de C#.

A continuación se recoge de manera resumida las principales características de C#:

- **Sencillez:** C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET
    - El código escrito en C# es *autocontenido*.
    - El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile.
    - No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple
- **Modernidad:** C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones.
- **Orientación a objetos:** Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos. C# soporta todas las características propias del paradigma de programación orientada a objetos*: encapsulacion, herencia y polimorfismo.*
- **Orientación a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas.
- **Gestión automática de memoria:** Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos.
- **Seguridad de tipos:** C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente.
- **Instrucciones Seguras:** Para evitar errores muy comunes, en C# se ha impuesto una serie de restricciones en el uso de las instrucciones de control mas comunes.
- **Sistema de tipos unificado:** A diferencia del C++, en C# todos los tipos de datos que se definan, siempre derivaran aunque sea de manera implícita de una clase base común denominada *System Object*
- **Extensibilidad de tipos básicos:** C# permite definir, a través de *estructuras*, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos.
- **Extensibilidad de operadores:** Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje.

- **Extensibilidad de modificadores:** C# ofrece, a través del concepto de *atributos*, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente información adicional a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la librería de reflexión de .NET.
- **Versionable:** C# incluye una *política de versionado* que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.
- **Eficiente:** En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros.
- **Compatible:** Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados *Platform Invocation Services (PInvoke)*, la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32

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 Emplear XML en .NET Framework.

6.10.1 Objetivos de diseño XML en .NET Framework.

Los objetivos de XML en .NET Framework son los siguientes:

- Compatibilidad con los estándares del W3C (World Wide Web Consortium): La compatibilidad con estándares significa que las clases se ajustan totalmente a los estándares actuales recomendados por el W3C (World Wide Web Consortium) en relación con XML, espacios de nombres, XSLT, XPath, esquemas y DOM (Document Object Model, Modelo de objetos de documento). La compatibilidad asegura la interoperabilidad y facilita el desarrollo de aplicaciones entre plataformas.

- Extensibilidad: En .NET Framework, XML se puede extender mediante el uso de las clases bases abstractas y los métodos virtuales. Dicha extensibilidad, o capacidad de creación de subclases, se ilustra en las clases abstractas *XmlReader*, *XmlWriter* y *XPathNavigator*, que permiten el desarrollo de nuevas implementaciones en diferentes almacenes u orígenes de datos, y exponen el código XML.

- Arquitectura Conectable: En .NET Framework, XML tiene una arquitectura conectable. En esta arquitectura basada en secuencias, *conectable* significa que los componentes basados en las clases abstractas de .NET Framework se pueden sustituir de forma sencilla. Una arquitectura conectable significa también que los datos se pueden transmitir en secuencias entre los componentes y los nuevos componentes insertados en la secuencia pueden alterar el procesamiento.

- Rendimiento: Las clases XML de .NET Framework representan componentes de procesamiento XML de bajo nivel que se utilizan no sólo como parte de .NET Framework, sino también para integrar XML en las aplicaciones. Las clases deben ofrecer un rendimiento extremadamente alto.
  Las clases XML de .NET Framework están diseñadas para admitir un modelo basado en secuencias.

- Integración con ADO.NET: Los datos relacionales y XML se combinan en .NET Framework mediante una estrecha integración entre las clases XML y ADO.NET.

6.10.2 Arquitectura de XML en .NET Framework.

Las clases XML de .NET Framework representan un conjunto coherente diseñado e integrado de clases que permiten crear de forma sencilla aplicaciones preparadas para XML. Los objetivos que se describen en la sección anterior abordan problemas reales que encuentran los programadores al utilizar XML: no sólo la creación de aplicaciones orientadas al Web, sino también las demás áreas de que se ocupa XML, como un formato de serialización común, la representación de objetos, la interoperabilidad y la mensajería, por nombrar sólo algunas.

6.10.3 Información General sobre mejoras de seguridad para System XML.

Son muchos los cambios que se han introducido en la versión 1.1 de .NET Framework con relación a la versión 1.0.

- XmlReader y sus clases derivadas, XmlTextReader y XmlValidatingReader: La petición de herencia se ha colocado en el nivel de clase de *XmlTextReader* y de *XmlValidatingReader*. Se deben tener derechos de plena confianza para heredar de estas clases.
- XmlDocument: Se ha modificado el comportamiento del método Load en *XmlDocument.* Ahora su comportamiento depende de si la clase es de plena confianza o de confianza parcial. Para obtener más información acerca de cómo se ve afectado el método *Load*.

- XmlResolver: La petición de herencia para plena confianza se ha colocado en el método ResolveUri de la clase *XmlResolver*.

  Se ha agregado una nueva clase derivada, *XmlSecureResolver*, a la versión 1.1 de .NET Framework. Esta clase permite a las aplicaciones que son de plena confianza proporcionar evidencia cuando obtienen acceso a datos y recursos externos.

- XmlTransform: La clase *XslTransform* ha adquirido diversas diferencias en el comportamiento.

6.10.4 Modelo de objetos de documentos XML (DOM).

La clase DOM (Document Object Model, Modelo de objetos de documento) es una representación en la memoria de un documento XML. DOM permite leer, manipular y modificar un documento XML mediante programación. La clase *XmlReader* también lee XML, aunque proporciona acceso de sólo avance y de sólo lectura sin almacenamiento en memoria caché. Esto significa que no hay funciones para editar los valores de un atributo o contenido de un elemento, ni la posibilidad de agregar y quitar nodos con *XmlReader*. La edición es la función principal de DOM. Es la forma común y estructurada mediante la que se representan datos XML en la memoria, aunque los datos XML reales se almacenan de forma lineal cuando se encuentran en un archivo o proceden de otro objeto.

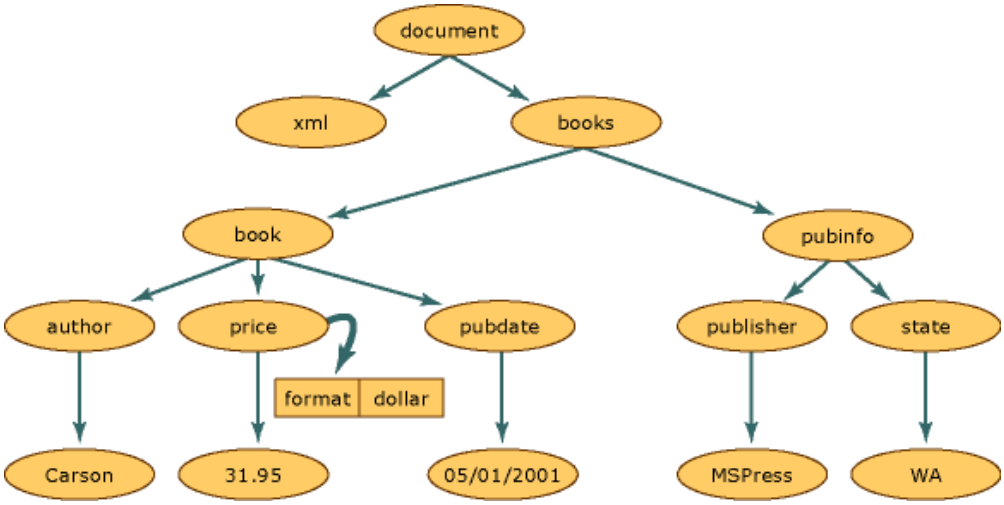


Figura 6.13 Estructura de Documentos XML.

6.10.5 Leer fragmentos XML con XMLReader.

XmlReader es una clase base abstracta que proporciona acceso de sólo avance y de sólo lectura sin almacenamiento en caché. La clase comprueba que el código XML tenga un formato correcto e inicia una excepción *XmlExceptions* si se encuentra un error.

Puede leer una secuencia o un documento, e implementa los requisitos de los espacios de nombres descritos en la recomendación proporcionada por el W3C (World Wide Web Consortium) org/TR/REC-xml-names.

Al ser una clase base abstracta, permite al usuario personalizar su propio tipo de sistema de lectura o extender las implementaciones actuales de las clases *XmlTextReader*, *XmlValidatingReader* y *XmlNodeReader*. Sin embargo, las restricciones de seguridad implementadas en .NET Framework versión 1.1 para *XmlReader* limitan la posibilidad de heredar de *XmlTextReader* y *XmlValidatingReader*. En .NET Framework versión 1.0, cualquier componente puede heredar de *XmlTextReader* o *XmlValidatingReader*. Sin embargo, al haberse implementado una demanda de herencia en el constructor de *XmlTextReader* y *XmlValidatingReader*, en .NET Framework versión 1.1 sólo los componentes que disponen de confianza tienen la posibilidad de heredar de *XmlTextReader* o *XmlValidatingReader.*

La clase *XmlReader* define métodos que permiten extraer datos de código XML u omitir registros no deseados. La clase *XmlReader* se diferencia de SAX, que es un modelo de inserción en el que el analizador inserta eventos en la aplicación. Para obtener más información sobre la comparación con SAX

La clase *XmlReader* contiene métodos que permiten:

- Leer contenido XML cuando éste está disponible en su totalidad, por ejemplo, un archivo de texto XML.
- Buscar la dimensión de la pila de elementos XML.
- Determinar si un elemento tiene contenido o está vacío.
- Leer y explorar atributos.
- Omitir elementos y su contenido.

La clase *XmlReader* tiene propiedades que devuelven información, como:

- Tipo y nombre del nodo actual.
- Contenido del nodo actual.

6.10.6 Escribir XML con XMLWriter.

XmlWriter es una clase base abstracta que define una interfaz para escribir código XML. La clase *XmlWriter* proporciona un modo de sólo lectura.

En la lista siguiente se muestra el propósito de los métodos y propiedades que incluye la clase *XmlWriter:*

- Especificar si se debe permitir el uso de espacios de nombres.
- Escribir código XML con un formato correcto.
- Codificar bytes binarios en Base64 y BinHex, y escribir el texto resultante.
- Administrar la salida, lo que incluye a los métodos para determinar su progreso, con la propiedad *WriteState.*
- Escribir varios documentos en una secuencia de salida.
- Vaciar o cerrar la secuencia de salida.

- Informar del prefijo de espacio de nombres actual, *xml: lang*, o el ámbito de *xml: space*.
- Escribir nombres válidos, nombres completos y símbolos (*tokens*) de nombres.

En la lista siguiente se identifican los elementos que la clase *XmlWriter* no comprueba:

- Los caracteres no válidos de los nombres de atributos y elementos.
- Los caracteres Unicode que no se ajustan a la codificación especificada. Si los caracteres Unicode no se ajustan a la codificación especificada, la clase XmlWriter no establece secuencias de escape para ellos en entidades de carácter.
- Atributos duplicados.
- Caracteres del identificador público DOCTYPE o del identificador del sistema.

Como *XmlWriter* no comprueba estos cuatro elementos, esto permite producir XML que no tenga el formato correcto.

6.10.7 Modelo de objetos de esquemas XML.

Un esquema XML es una herramienta eficaz y compleja para crear y validar la estructura de documentos XML compatibles. De forma parecida al modelado de datos de una base de datos relacional, un esquema proporciona una forma de definir la estructura de los documentos XML al especificar los elementos que se pueden utilizar en ellos, así como la estructura y tipos que estos elementos deben tener para ser válidos con respecto al esquema específico.

Un esquema es un archivo XML, que suele tener la extensión .xsd, en el que se describe el contenido de los elementos XML mediante código XML válido: los elementos y los atributos se declaran con los elementos *element* y *attribute*, y la estructura se crea con los elementos *simpleType* y *complexType*.

Un esquema es un documento XML en el que se define una clase de documentos XML mediante la especificación de la estructura o el modelo de los documentos XML para un esquema determinado. En un esquema se identifican las restricciones en el contenido de los documentos XML y se describe el vocabulario (reglas o gramática) que deben seguir los documentos XML compatibles para ser considerados válidos con respecto al esquema en particular. La validación de un documento XML es el proceso que asegura que el documento se ajusta a la gramática especificada en el esquema.

Los esquemas proporcionan las siguientes mejoras sobre las DTD (Document Type Definitions, definiciones de tipo de documento):

- Al usar un esquema se dispone de tipos de datos adicionales.
- Con un esquema se pueden crear tipos de datos personalizados.
- Un esquema utiliza la sintaxis XML.
- Un esquema admite conceptos orientados a objetos, como polimorfismo y herencia.

6.10.8 Validación de XML con esquemas.

Para definir la estructura de un documento XML, así como las relaciones entre sus elementos, los tipos de datos y las restricciones de contenido, se utiliza una DTD (Document Type Definition, definición de tipo de documento) o esquema. Aunque se considera que un documento XML tiene un formato correcto si cumple todos los requisitos sintácticos definidos por la recomendación de Extensible Markup Language (XML) 1.0 del World Wide Web Consortium (W3C), no se considerará válido a menos que, además de tener un formato correcto, se ajuste a las restricciones definidas por su DTD o su esquema. Por lo tanto, aunque todos los documentos XML válidos tienen un formato correcto, no todos los documentos XML con un formato correcto son válidos.

La validación de documentos y fragmentos XML se exige mediante la clase *XmlValidatingReader*, que proporciona servicios de validación de esquemas de lenguaje XSD (Schema Definition, definición de esquemas XML), XDR (XML-Data Reduced, reducido de datos XML) y DTD mediante la implementación de restricciones de validez definidas por las recomendaciones del W3C.

La clase *XmlValidatingReader* implementa la clase *XmlReader* y la validación se realiza sólo hacia delante sobre una secuencia de XML. *XmlValidatingReader* puede aceptar un *XmlTextReader* como entrada. Los niveles de *XmlValidatingReader* admiten *XmlTextReader* y también la posibilidad de analizar fragmentos de XML con la *clase XmlParserContext.*

6.11 Dibujar y Editar imágenes en .NET Framework.

GDI+ para .NET Framework es una interfaz de programación de aplicaciones (API) basada en clases para programadores que utilizan código administrado. Permite que las aplicaciones utilicen gráficos y texto con formato tanto en la pantalla como en la impresora. Las aplicaciones basadas en la API de Microsoft Win32 no tienen acceso directamente a hardware de gráficos. En su lugar, GDI+ interactúa con los controladores de dispositivos en nombre de las aplicaciones. GDI+ también es compatible con el sistema operativo Windows de 64 bits.

GDI+ puede utilizarse en todas las aplicaciones basadas en Windows. GDI+ es una nueva tecnología incluida en los sistemas operativos Microsoft Windows XP y Windows Server 2003. Es un redistribuible obligatorio para las aplicaciones que se ejecutan en los sistemas operativos Windows NT 4.0 SP6, Windows 2000, Windows 98 y Windows Millennium Edition.

La API de GDI+ está diseñada para su uso por parte de los programadores que utilizan lenguajes que siguen protocolos de código administrado. Es necesario conocer la interfaz gráfica de usuario y la arquitectura controlada por mensajes de Windows.

6.11.1 Información general acerca del GDI+.

GDI+ es el subsistema del sistema operativo Microsoft Windows XP que se encarga de mostrar información en pantallas e impresoras. Tal y como su nombre indica, GDI+ es la sucesora de GDI, la interfaz de dispositivo gráfico incluida en versiones anteriores de Windows. GDI+ es una interfaz de programación de aplicaciones (API) que se expone a través de un conjunto de clases implementadas como código administrado. A este conjunto de clases se le denomina la *interfaz de clases administradas* en GDI+.

Una interfaz de dispositivo gráfico como GDI+ permite a los programadores de aplicaciones mostrar información en una pantalla o impresora sin tener que preocuparse de los detalles de un dispositivo de presentación específico. El programador de aplicaciones llama a los métodos suministrados por las clases de GDI+ y esos métodos, a su vez, llaman a los controladores de dispositivo específicos. GDI+ aísla a la aplicación del hardware gráfico, y este aislamiento es el que permite a los programadores crear aplicaciones independientes del dispositivo.

6.11.2 Las Tres Partes del GDI+.

Los servicios de GDI+ se engloban en tres amplias categorías:

- Gráficos vectoriales 2D
- Imágenes
- Tipografía

6.11.3 Estructura de la Interfaz basada en clases.

La interfaz de clases administradas en GDI+ contiene en torno a 60 clases, 50 enumeraciones y 8 estructuras. La clase *Graphics* es la base de la funcionalidad de GDI+; es la clase que realmente dibuja líneas, curvas, figuras, imágenes y texto.

6.11.4 Lo nuevo de GDI+.

GDI+ difiere de GDI en un par de puntos. En primer lugar, GDI+ expande las características de GDI ya que proporciona nuevas características como, por ejemplo, pinceles degradados y mezcla alfa. En segundo lugar, se ha revisado el modelo de programación en GDI+ para que la programación de gráficos sea más fácil y flexible.

6.11.5 Imágenes, mapa de bits y metarchivos

La clase *Image* es una clase base abstracta que proporciona métodos para trabajar con imágenes de trama (mapas de bits) e imágenes vectoriales (metarchivos). Tanto la clase *Bitmap* como la clase *Metafil*e se heredan de la clase *Image.*

La clase *Bitmap* expande las capacidades de la clase *Image* ya que proporciona métodos adicionales para cargar, guardar y manipular imágenes de trama. La clase *Metafile* expande las capacidades de la clase *Image* ya que proporciona métodos adicionales para registrar y examinar imágenes vectoriales.

6.11.6 Dibujar, colocar y clonar imágenes.

La clase *Bitmap* puede utilizarse para cargar y mostrar imágenes de trama y la clase *Metafil***e** puede utilizarse para cargar y mostrar imágenes vectoriales. Las clases *Bitmap* y *Metafile* se heredan de la clase *Image*. Para mostrar una imagen vectorial, son necesarios un objeto Graphics y un objeto *Metafile*. Para mostrar una imagen de trama, son necesarios un objeto *Graphics* y un objeto *Bitmap*. El objeto *Graphics* proporciona el método *DrawImage*, que recibe el objeto *Metafile* o *Bitmap* como argumento.

La clase *Bitmap* proporciona un método *Clone* que puede utilizarse para hacer una copia de un objeto *Bitmap* existente. El método *Clone* tiene un parámetro de rectángulo de origen que puede utilizarse para especificar la parte del mapa de bits original que se desea copiar. En el siguiente ejemplo se crea un objeto *Bitmap* mediante la clonación de la mitad superior de un objeto *Bitmap* existente. Después, se dibujan ambas imágenes.

6.11.7 Recortar y ajustar la escala de las imágenes.

El método *DrawImage* de la clase *Graphics* puede utilizarse para dibujar y colocar imágenes vectoriales e imágenes de trama. *DrawImage* es un método sobrecargado, por lo que existen varias formas de suministrar argumentos a dicho método. Una variación del método *DrawImage* recibe un objeto *Bitmap* y un objeto *Rectangle*. El rectángulo especifica el destino para la operación de dibujo, es decir, especifica el rectángulo en el que se va a dibujar la imagen. Si el tamaño del rectángulo de destino difiere del tamaño de la imagen original, se ajusta la escala de la imagen para que encaje en el rectángulo de destino.

Capitulo 7. SISTEMAS DE COMUNICACIÓN BASADOS EN SIGNOS E IMÁGENES.

7.1 Sistemas de Comunicación.

En la sociedad actual hay diversas formas de comunicarse no sólo la oral, también está la escrita, la gestual, la gráfica y todas son formas pacificas de comunicarse con el resto de mortales que habitan alrededor nuestro o a miles de kilómetros.

Los sistemas de comunicación pueden ser:

	VERBAL	**NO VERBAL**
VOCAL	• Lengua hablada • Comunicador (si se oye)	• Lloro • Silbido • Grito
NO VOCAL	• Lengua escrita, • Braile • LSC • Código Morse, • Comunicador (si se transcribe)	• Mímica • Dibujo y pintura • Mosman • picsyms • SPC • Bliss • lenguaje de signos • Rebuss • Piscyms • PIC

Figura 7.1 Sistemas de Comunicación.

7.2 Sistemas Alternativos de Comunicación.

Se entiende por sistemas alternativos de comunicación a las distintas estrategias o ayudas puestas al servicio de las personas seriamente discapacitadas en el ámbito del lenguaje oral para establecer relaciones comunicativas.

Vamos a considerar algunos aspectos de los sistemas de comunicación:

- La utilidad de los sistemas consiste en favorecer la capacidad de expresión, por lo que cualquier posibilidad de lenguaje oral ha de ser explotada al máximo.
- Un sistema alternativo y complementario de comunicación no es como el lenguaje oral, pero no por ello se ha de considerar inútil su implantación, al contrario, a pesar de las limitaciones de cualquier sistema de comunicación en comparación con el habla, su uso acerca a las personas a las ventajas que el habla conlleva.
- Una clara limitación es que se necesita el conocimiento del sistema por parte de las personas que quieren interactuar con el niño o niña, y en ocasiones aprenderlo no es tan sencillo.

Es preciso clasificar los sistemas de comunicación según sean sistemas complementarios a la comunicación, como es el caso de la comunicación bimodal, la palabra complementada o Cued-Speech, o sistemas alternativos a la comunicación, en el caso de que el alumno no se pueda beneficiar del lenguaje oral, como el lenguaje de signos, SPC (Símbolos pictográficos para la comunicación) o Bliss.

Algunos de estos sistemas, para poder ser utilizados, precisan apoyos externos como un tablero de letras, palabras, imágenes o símbolos para la comunicación, como es el caso del SPC.

Los sistemas de comunicación alternativa pueden ser de 2 tipos:

- **Sin Ayuda:** Son sistemas de comunicación que se emplea el gesto, la mímica o el signo natural sin utilizar un elemento foráneo a tu propio cuerpo.
  Pueden ser de varias categorías:

  - Los lenguajes manuales utilizados por los no oyentes
  - Los códigos o sistemas pedagógicos creados a través de los anteriores.
  - Los códigos gestuales que no son lenguas sino solo habla (pero entendiendo como habla no oral).

- **Con Ayuda:** Necesitas un sistema de ayuda para comunicarte. Necesitan de algún instrumento o técnica foránea para poderte comunicar. Según el sistema gráfico que se emplee para representar la idea, podemos distinguir varios tipos de sistemas de signos con ayuda

Entre los sistemas alternativos de comunicación estarían:

- El lenguaje de signos.
- La comunicación bimodal.
- SPC y Bliss.
- Palabra Complementada o Cued-Speech.

7.3 Tipos de Símbolos.

Hay tres tipos de símbolos:

- **Pictográficos:** aquellos que representan la idea o forma de la realidad a la que se refiere.
- **Ideográfico:** fruto de un acuerdo social, por lo tanto, no tiene una relación lógica entre lo que se representa (símbolo) y la forma concreta.
- **Símbolos de relación:** son aquellos que se utilizan para relacionar los demás símbolos del sistema

7.4 Sistema de Comunicación SPC.

Es un sistema con muchas posibilidades ya que se puede adaptar al nivel de desarrollo, comunicación y necesidades del niño o niña.

Este sistema es el más utilizado exclusivamente en España.

El SPC tiene como objetivo principal facilitar la comunicación en sujetos no orales con dificultades motoras y auditivas. Consta de pequeñas tarjetas con dibujos muy sencillos y representativos para el alumno que están acompañados de la palabra escrita, también podemos añadir otros que no tenga el sistema y que consideremos útiles para el chico.

Para llevar a la practica este sistema se eligen los símbolos según el nivel del alumno y se colocan sobre un tablero, para seleccionar este vocabulario inicial se tienen en cuenta, ante todo, las tarjetas con sus necesidades básicas (aseo, alimentación...), las actividades cotidianas y sus gustos o preferencias. Después se van incorporando al vocabulario existente aquellas palabras que vaya necesitando cada persona a medida que van cambiando sus necesidades comunicativas. Entonces le enseñaremos al niño o niña a encadenar palabras para ir formando frases.

Estos símbolos utilizados son símbolos pictográficos sencillos e icónicos, que tienen las siguientes características:

- Representan las palabras.
- Están pensados e indicados para diferentes grupos de personas con diferentes dificultades.
- Los dibujos se pueden reproducir fácilmente.
- Se pueden separar los dibujos muy fácilmente confeccionando el plafón a las necesidades de cada usuario.

Los dibujos/símbolos están divididos por categorías que están sistematizadas por colores, estas categorías son las siguientes:

- Personas (amarillo)
- Verbos (verde)
- Términos descriptivos/adverbios (azul)
- Nombres (naranja)
- Términos sociales (rosado)
- Términos diversos como letras del alfabeto, números o colores (blanco)

Los motivos por los que suele utilizarse este código (SPC) en España, son:

- Color: facilitan localización rápida del símbolo.
- Porque ayudan a estructurar la mente
- Porque los dibujos son fácilmente identificables
- Porque al estar escritos ayudan y facilitan la lectura/escritura.

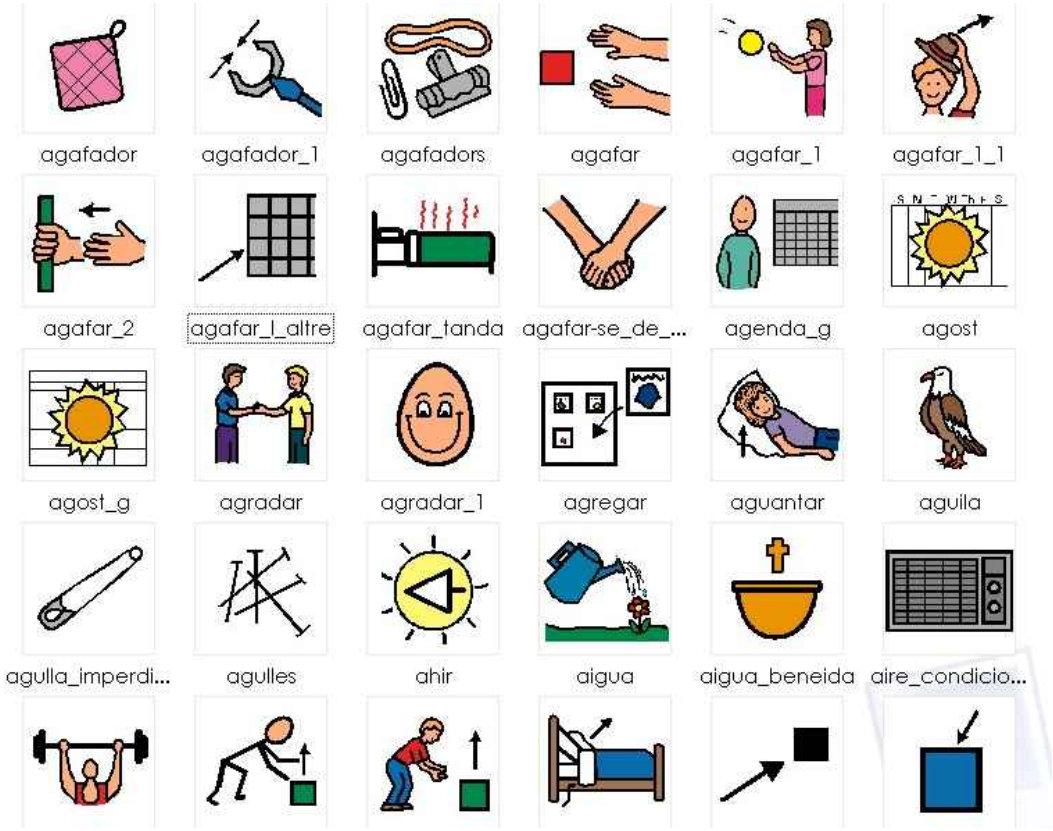


Figura 7.2 Imágenes SPC.

Parte III. Desarrollo de la Aplicación

Capitulo 8. ESTUDIO PREVIO.

8.1 Decisiones Iniciales.

El objetivo de este proyecto es la realización de una aplicación que permita a las personas con Parálisis Cerebral, cuyos problemas motores sean bastantes severos, comunicarse con las personas de su entorno a través de un dispositivo móvil como es una Pocket PC, mediante la utilización de un pulsador o conmutador adaptado.

Uno de los objetivos principales de esta aplicación es, poder adaptarse a las necesidades y gustos de cada persona.

A su vez pretende ser una herramienta práctica y sencilla, que pueda ser manejada por aquellas personas que no poseen precisión en sus movimientos y no pueden pulsar determinada posición de una pantalla táctil, utilizar un stylo etc. para escribir en una PDA.
Mediante un pulsador o conmutador se facilitara la comunicación de estas personas con trastornos graves en su motricidad.
Seleccionaran en la pantalla posiciones que contendrán letras para poder construir frases para su comunicación.

La aplicación también esta pensada para aquellas personas que no hayan desarrollado la lectura/escritura, ya que incluirá una configuración basada en el sistema de comunicación SPC (imágenes con palabras asociadas), en el momento que se seleccione una imagen la palabra asociada a ella aparecería en la pantalla de la PDA.

8.2 Metodología Utilizada.

El método utilizado deberá ser apropiado para la orientación a objetos puesto que el lenguaje a utilizar 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.

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)

8.3 Métodos de escritura.

8.3.1 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 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.

Debido a la precisión que requiere, se descartara también el uso de pantalla táctil.

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	Alto
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 8.1 Métodos de escritura actuales para dispositivos móviles.

Para este proyecto, teníamos una serie de especificaciones claras que el proyecto debía cumplir, que afectaban al diseño del método de escritura:

- Sin stylus o puntero: El proyecto esta dirigido a personas con dificultades severas del movimiento que no tienen capacidades para manejar con la precisión suficiente un objeto tan pequeño, y mucho menos usarlo en una pantalla pequeña, además el uso de un stylus tendría asociadas otras acciones con dificultad para estas personas como sacar el puntero de la PDA.

- Sin pantalla táctil: Entre la personas que padecen parálisis cerebral nuestro proyecto esta enfocado aquellas con problemas motores bastantes severos y que por lo general no tendrán suficiente precisión en sus movimiento como para apuntar a una posición determinada de la pantalla de la PDA y presionarla.

- Capacidades de precisión muy reducidas: como se ha indicado en los apartados anteriores nos centraremos en las personas con mas limitaciones en sus movimientos.

- Velocidad: Los usuarios tendrán una velocidad de reacción bastante reducida, sus movimientos serán lentos.

- Pantalla reducida: Se aprovechara al máximo la pantalla. Por eso se usara toda su superficie no podemos olvidar que las personas con parálisis, frecuentemente tienen problemas visuales. En la aplicación definitiva se han eliminado todos los elementos que no tienen un uso en la aplicación, como barra de tareas o 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 margene son los bordes de la pantalla.

- Sin Botones hardware: Los botones hardware existentes en las PDAs no tendrán en esta

aplicación ningún uso. Son botones que para ser utilizados necesitan tener cierta precisión en el movimiento de los dedos así como un poco de fuerza para presionarlos, por esta razón no se les dará ninguna función. Las funciones que pudieran tener, serán asignadas a botones software que estarán dentro de la pantalla de la PDA.

- Inclusión de imágenes: El proyecto esta dirigido para las personas más afectadas por la enfermedad de la parálisis cerebral y muchas de estas no han desarrollado la capacidad de lectura y escritura, siendo a las que más difícil resulta la comunicación. La existencia de imágenes sencillas les ayudara a comunicarse.

- Configurable: La aplicación será lo mas configurable posible, esto será un de los principales objetivos, para así adaptarse a los gustos y necesidades de cada usuario, podrán elegir los colores, el numero de botones por pantalla, el contenido de los botones, etc. dentro de cada configuración . Habrá varias configuraciones para elegir.

- Elementos 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.

- Todo podrá hacerlo el usuario, excepto la instalación de la aplicación y la variación de alguna de las configuraciones existentes.

8.4 Métodos de escritura aplicables a una PDA.

Según las premisas expuestas anteriormente, hemos considerado como posibles métodos de escritura para esta aplicación, basándonos en métodos de escritura usados en móviles, métodos de escritura existentes para ordenador y alguna idea de teclados en pantallas para PDAs ya existentes, los siguientes:

8.4.1 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 el pulsador conectado a la PDA, y esa letra será seleccionada. A continuación se vuelve a hacer un barrido de todas las letras.

Es un método 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.

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.
Se tardaría cerca de un minuto en escribir “HOLA”.

La única habilidad necesaria del usuario para usar este método consiste en poder golpear un pulsador conectado a la PDA con una parte de su cuerpo con la que tenga movilidad. Ni siquiera necesita usar el dedo, podría usar la cabeza, el codo, la barbilla etc.

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

8.4.2 Método del barrido por sonidos

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.

8.4.3 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.

8.4.4 Método del barrido de imágenes.

Este método no utiliza letras si no imágenes .Esta indicado para personas con dificultad en la lectura y escritura.

En la pantalla de la PDA no salen letras, salen imágenes asociadas a una palabra. Mediante un cursor de otro color se le indica al usuario que imagen esta activa. Si el usuario da al pulsador conectado a la PDA cuando el cursor esta encima de una imagen, esta quedaría seleccionada y la palabra asociada a ella aparecería en la pantalla de la PDA.

Al igual que el barrido de letras es un método lento pero no requiere precisión en los movimientos como se ha indicado anteriormente el pulsador puede apretarse con la parte del cuerpo con mayor movilidad y no se necesita saber leer ni escribir, pudiendo incluso resultar mas divertido sobre todo si el usuario es un niño.

8.4.5 Método de las pulsaciones repetidas.

DESCRIPCIÓN:

Es el método usado en los móviles.
Se ponen grupos de letras, y por cada pulsación en un mismo grupo irán saliendo en orden todos los caracteres que contiene hasta llegar a la letra deseada. Es decir con una pulsación en un grupo se selecciona la primera letra de ese grupo, con dos pulsaciones sobre ese grupo la segunda letra y así sucesivamente.
El orden depende de las letras que estén en cada grupo.

Este método queda descartado porque usando un pulsador conectado al PDA si la letra que queremos ocupa la quinta posición dentro de un grupo seria necesario hacer 5 pulsaciones consecutivas para seleccionarla.
Además tiene el inconveniente que las pulsaciones deben ser consecutivas si tardamos en realizar una pulsación dentro de un grupo quedar seleccionada la letra anterior a esa pulsación.

8.4.6 Método puro usando Bases de Datos.

El usuario tiene una lista de palabras sugeridas. 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 INCOMPLETO LANA JOTA
Segunda Pulsación	“MNÑOPQ”	LOS PORTICO GOTICO HORTERA
Tercera Pulsación	“GHIJKL”	GOL IMPERFECTO HORTELANO HORALIZA
Cuarta Pulsación	“ABCDEF”	HOLA INICIO INICIAR LOLA

Figura 8.2 Método puro de las bases de datos.

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

Puede ser rápido, si encontramos pronto la palabra que buscábamos, la podemos seleccionar entera, sin necesidad de acabar la frase. Como gran desventaja, no podremos introducir una palabra que no exista en el dispositivo, tendríamos que usar otro mecanismo de escritura para poder acabar lo que queríamos decir.

Otra de sus desventajas seria; si en cada grupo de letras hay que hacer un barrido por todas las palabras que contiene ese grupo, en este caso podría ser muy lento. Además necesitaría algo de entrenamiento.

8.4.7 Resumen.

	Orden	Velocidad	Capacidad física Requerida	Aprendizaje
BARRIDO	1	Muy lento	Muy Poco	Muy Poco
BARRIDO SONIDOS	1	Muy Lento	Muy Poco	Muy Poco
BARRIDO GRUPOS		Lento	Muy Poco	Poco
BARRIDO IMÁGENES	1	Lento	Muy Poco	Muy Poco
PULSACIONES REPETIDAS	3	Media	Alta	Medio
PURO BASES DE DATOS		Lento	Media	Medio

Figura 8.3 Resumen de los métodos de escritura.

8.4.8 Conclusión.

Los usuarios a los que va destinada esta aplicación pueden tener características muy distintas debido a la enfermedad, por lo que no podemos considerar que métodos son mejores que otros, eso dependerá de la persona que vaya a utilizarla y sus capacidades.

Nuestro proyecto no esta dirigido a una persona concreta, si no que ha sido realizado para un colectivo de alumnos que debido a la Parálisis Cerebral tienen en otras las siguientes discapacidades:

- La capacidad de habla es prácticamente nula.
- Los problemas físicos son muy severos, teniendo una capacidad de movimiento muy reducido en todo su cuerpo.
- Habrá Usuarios que hallan desarrollado la lectura y escritura, y otros que no.

Para tener un contacto directo con las personas que debido a la Parálisis Cerebral se encuentran en esta situación hemos trabajado en colaboración con el Centro Oregon perteneciente a Asprona, a través de su responsable Víctor.

En el Centro Oregon hemos podido conocer niños con estos trastornos y las dificultades que encuentra para relacionarse con su entorno, aquello que a nosotros nos puede resultar lo más sencillo para ellos requiere un gran esfuerzo.

Teniendo en cuenta que estos niños no pueden mover casi la totalidad de su cuerpo incluyendo sus manos y dedos, el método que hemos considerado mas adecuado para implementar en la PDA es el Método basado en barrido. Con este método conectar un pulsador adaptado a la PDA para que puedan presionarlo con aquella parte de su cuerpo que sean capaces; barbilla, codo, muñeca etc. y así seleccionar el botón, la tecla que en ese momento este activa en la pantalla

El método basado en barrido puede resultar un poco lento, pero para estas personas lo importante y esencial es comunicarse, el tiempo para ellos no tendrá tanta importancia.

Para este proyecto hemos desarrollado varios métodos de escritura diferentes todos ellos basados en el método del barrido.Como se explico anteriormente en este método un cursor ira pasando por los elementos que aparezcan en la pantalla de la PDA cambiando su color y ese elemento donde se encuentra el cursor será el elemento que esta seleccionado .Algunos de estos métodos seguirán el método del barrido individual, otros el barrido por grupos.

Los métodos de escritura implementados en la aplicación se pueden clasificar en:

- Métodos que necesitan que el usuario haya desarrollado la lectura/escritura: estos métodos se basaran en el lenguaje escrito, utilizando letras, símbolos, números etc. para la comunicación
- Métodos que no será necesario que el usuario tenga la capacidad de leer y escribir: estos métodos utilizaran para la comunicación el sistema de imágenes SPC, (este sistema ha sido explicado en capítulos anteriores) , las imágenes están clasificadas por categorías con un color especifico y tendrán asociado una palabra, expresión.

Todos los métodos serán a su vez configurables, se podrá elegir:

- El contenido de los botones. Se podrá cambiar el carácter o la imagen que tiene el botón por el deseado. Si algún carácter se usa más que otro se podrá colocar en los primeros botones de la pantalla para que el cursor lo seleccione antes y así ahorrar tiempo y facilitar la escritura.

  En los métodos basados en imágenes se podrán cambiar las imágenes que aparecen en los botones de la pantalla y colocar aquellas con las que un usuario concreto.

- La función de los botones: Se podrá cambiar la función que desempeña un botón asignándole la función deseada, borrar por inserta espacio etc.

La aplicación contara con una lista de frases hecha que el usuario podrá utilizar cuando lo desee y podrá variar esa lista, guardando frases nuevas y eliminando aquellas que no considere útiles.

El reproductor de voz existente ha sido reutilizado de un proyecto anterior realizado por alumnos de esta escuela. Este reproductor pasara el texto escrito en la PDA a sonido mediante la emisión de silabas

Capitulo 9. ANALISIS DEL SISTEMA. DEFINICION 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.

9.1 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 dispondrá de dos métodos de comunicación diferentes: uno estará basado en la escritura, la comunicación se hará a través de letras y un segundo método basado en imágenes.

Debido a las limitaciones de movilidad y visión de los usuarios de esta aplicación todos los elementos que aparecen será lo mas grande posible. Entre las configuraciones existentes habrá variaciones en relación al tamaño de los elementos para adaptarse mejor a las necesidades de cada usuario.
Con referencia al texto que aparece será claro en todas las configuraciones disponibles.

Requisito indispensable que nos solicitaron fue una aplicación lo mas configurable posible donde todo se pudiese elegir: la colocación de los elementos el la pantalla, su contenido, la función que

desempeñada por cada uno de ellos, el tamaño del texto, los colores.

Otro de los principales requisitos era la existencia de un método de escritura que permitiese la comunicación de personas sin capacidad para leer y escribir. Por esta razón la aplicación contara con un método basado en imágenes SPC.
En relación al requisito anterior otra petición que además supondrá una ventaja referente al uso de imágenes será que las imágenes usadas no serán nuevas para los usuarios, serán imágenes con las que ya están familiarizados y que ya han utilizado lo que facilitara su comprensión y uso. Estarán clasificadas por categorías para un acceso más rápido a la imagen deseada.

9.2 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 este sistema habrá tres actores que interactuaran con la aplicación, estos serán:

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

ACT-0002	**Profesor**
Descripción	Este actor realizara funciones que no podrán ser ejecutadas por el actor alumno

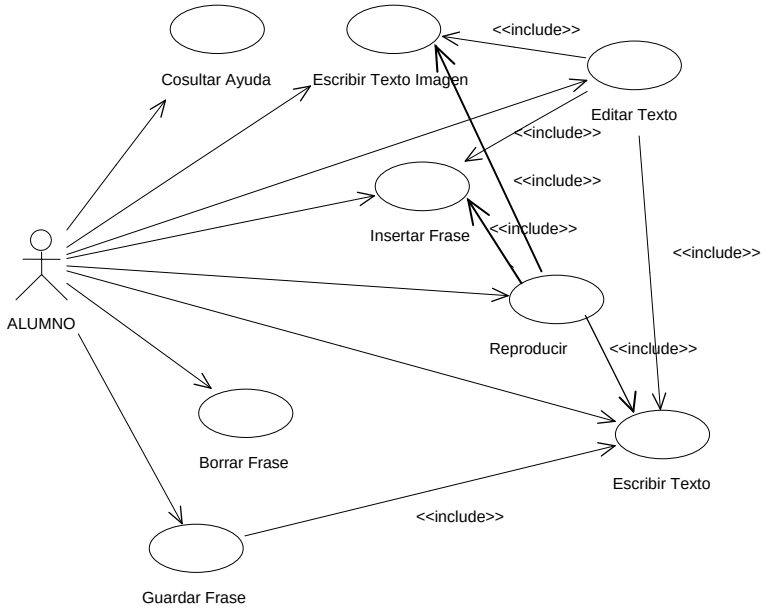
ACT-0003	**Administrador**
Descripción	Este actor es el encargado de hacer los cambios precisos en las configuraciones del sistema.

Figura 9.1 Descripción de los actores del sistema.

9.3 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.



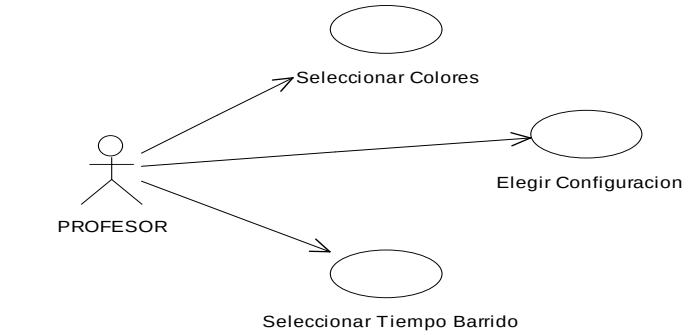


Figura 9.2 Diagrama de casos uso

9.4 Descripción de los casos de uso.

Un caso de uso describe una funcionalidad más un interacción entre un actor y un sistema en forma de secuencia de acciones.

La descripción se centra en lo que debe hacerse, no en la manera de hacerlo, buscando sencillez y claridad. Debe incluir cuando comienza el caso de uso, cuando finaliza, la interacción entre el caso de uso y los actores y el intercambio de datos realizado.

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.

9.4.1 Consultar ayuda.

UC-0001	**Consultar Ayuda**	
Descripción	El sistema deberá permitir al usuario poder consultar la ayuda sobre el uso de determinadas funciones de la aplicación en cualquier momento según se describe en el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor alumno(ACT-001) solicita ayuda al sistema
	2	El sistema le ofrece la ayuda referente a todos los temas Dispone
	3	El actor alumno (ACT-001) selecciona de entre todos los temas uno.
	4	El sistema le proporciona la ayuda referente a ese tema en concreto

Figura 9.3 Descripción del caso de uso Consultar Ayuda.

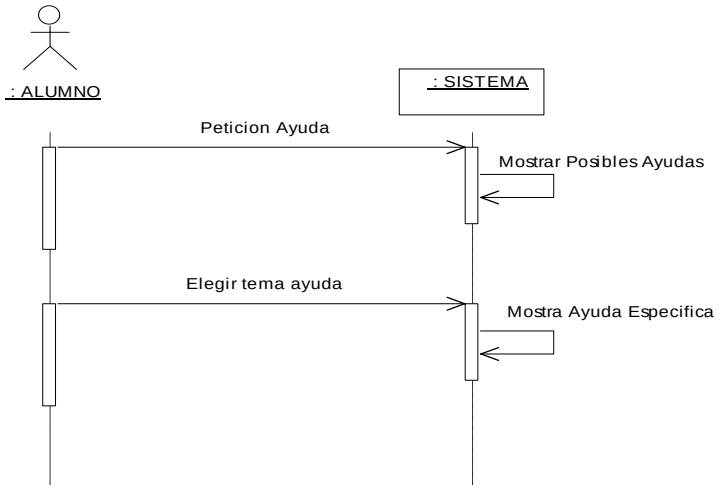


Figura 9.4 Diagrama del caso de uso Consultar Ayuda.

9.4.3Escribir Texto Imagen.

UC-0002	Escribir Texto Imagen	
Descripción	El sistema deberá permitir al usuario escribir texto mediante la selección de una imagen según el siguiente caso de uso	
Secuencia Normal	Paso	Acción
	1	El actor alumno (ACT-0001) escoge una imagen
	2	El sistema inserta en el texto la expresión o palabra asociada a esa imagen
	3	El sistema actualiza el texto
	4	Si se equivoca en algún carácter el actor Alumno (ACT-001) puede solicitar borrar texto asociado a la imagen.

Figura 9.5 Descripción del caso de uso Escribir Texto Imagen.

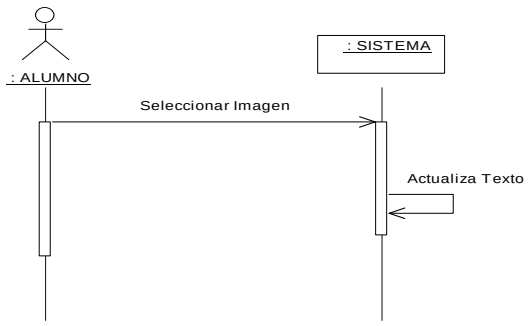


Figura 9.6 Diagrama del caso de uso Escribir Texto Imagen.

9.4.3 Editar Texto.

UC-0003	Editar Texto	
Descripción	El sistema deberá permitir al usuario editar el texto escrito según el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor alumno(ACT-001) escribe el texto que desea editar
	2	El actor alumno(ACT-001) solicita editar texto
	3	El sistema editara el texto correspondiente.

Figura 9.7 Descripción del caso de uso Editar Texto.

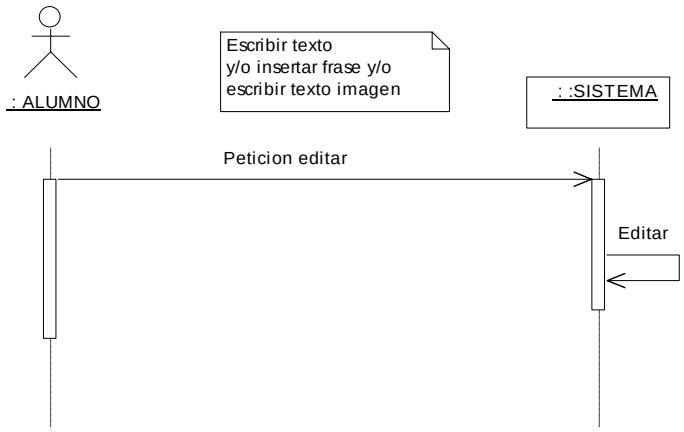


Figura 9.8 Diagrama del caso de uso Editar Texto.

9.4.4 Insertar Frase.

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

Figura 9.9 Descripción del caso de uso Insertar Frase.

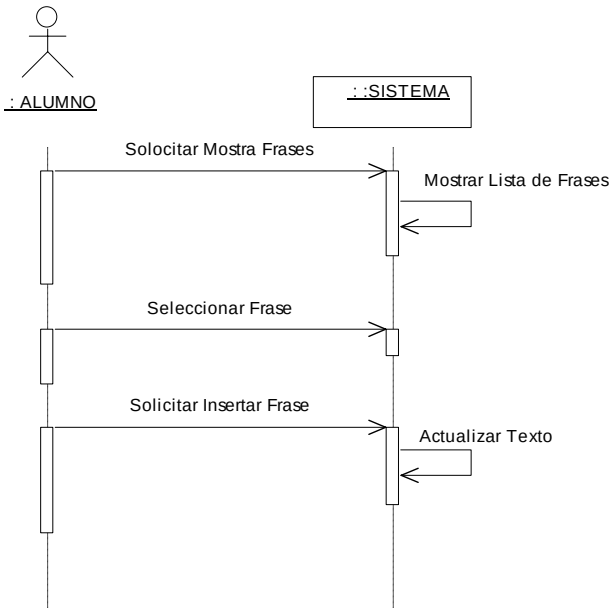


Figura 9.10 Diagrama del caso de uso Insertar Frase.

9.4.5 Reproducir.

UC-0005	Reproducir	
Descripción	El sistema permitir al usuario reproducir el texto escrito	
Secuencia Normal	Paso	Acción
	1	El actor alumno (ACT-0001) escribe el texto que desea reproducir.
	2	El actor alumno (ACT-0001) solicita su reproducción.
	3	El sistema reproduce el texto correspondiente.

Figura 9.11 Descripción del caso de uso Reproducir.

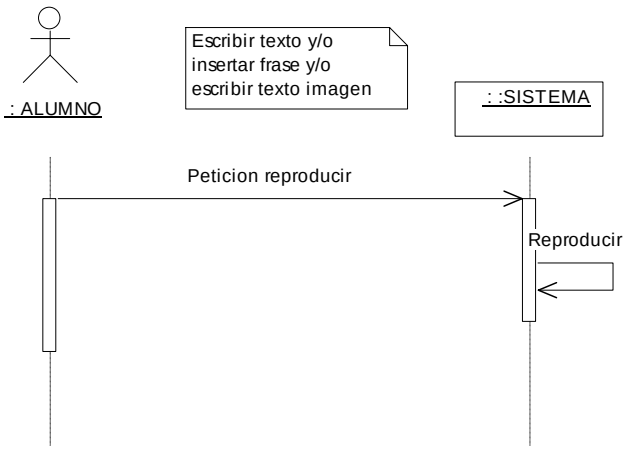


Figura 9.12 Diagrama del caso de uso Reproducir.

9.4.6 Borrar Frase.

UC-0006	Borrar Frase	
Descripción	El sistema permitir al usuario borrar las frases que tiene almacenadas si considera que no son suficientemente útiles	
Secuencia Normal	**Paso**	**Acción**
	1	El actor alumno(ACT-0001) solicita borrar una frases
	2	El sistema muestra la lista de frases que dispone
	2	El actor alumno(ACT-0001) escoge la frase con la que quiere borrar
	3	El sistema actualiza la lista de frases.

Figura 9.13 Descripción del caso de uso Borrar Frase.

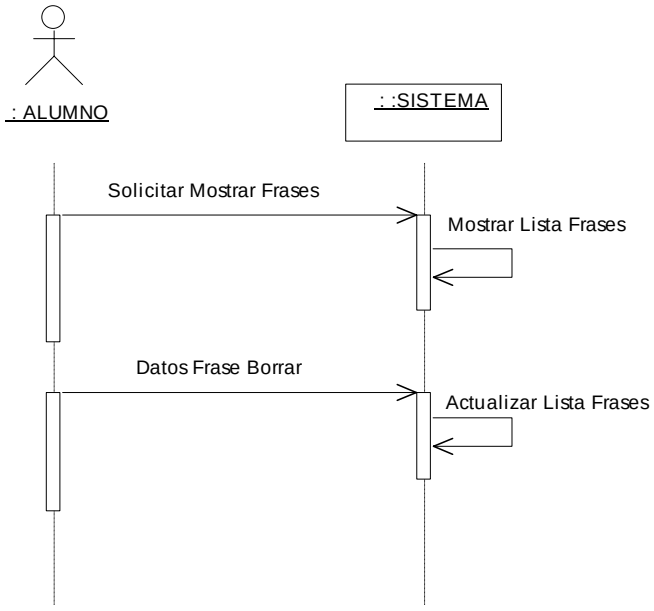


Figura 9.14 Diagrama del caso de uso Borrar Frase.

9.4.7 Escribir Texto.

UC-0007	Escribir Texto	
Descripción	El sistema deberá permitir al usuario escribir en diferentes métodos de escritura cualquier texto según el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor alumno (ACT-0001) escoge la letra que quiere escribir
	2	El sistema actualiza el texto

Figura 9.15 Descripción del caso de uso Escribir Texto.

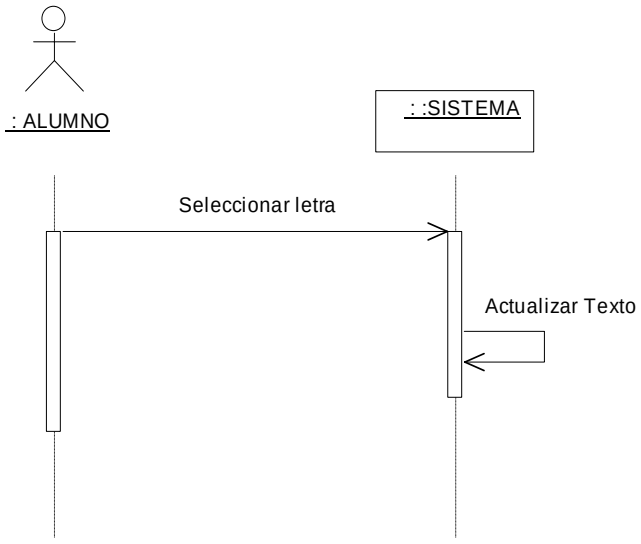


Figura 9.16 Diagrama del caso de uso Escribir Texto.

9.4.8 Guardar Frase.

UC-0008	Guardar Frase	
Descripción	El sistema permitir al usuario guardar sus frases si lo desea para su posterior utilización	
Secuencia Normal	Paso	Acción
	1	El actor alumno(ACT-0001) escribe la frase que desea guardar
	2	El actor alumno(ACT-0001) solicita guardar frase
	3	El sistema guarda la frase.

Figura 9.17 Descripción del caso de uso Guardar Frase.

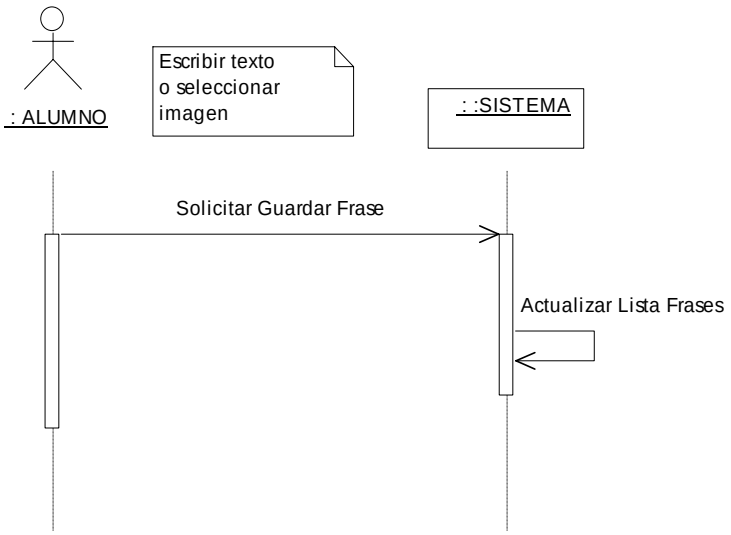


Figura 9.18 Diagrama del caso de uso Guardar Frase.

9.4.9 Seleccionar Colores.

UC-0009	Seleccionar Colores	
Descripción	El sistema deberá permitir al usuario elegir los colores de todos los elementos que aparecen en la aplicación según se describe en el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor profesor (ACT-002) solicita elegir colores
	2	El sistema le muestra todos los elemento que disponen de colores
	3	El actor profesor (ACT-002) selecciona el tipo de elemento que quiere cambiar de color
	4	El sistema actualiza los cambios

Figura 9.19 Descripción del caso de uso Seleccionar Colores.

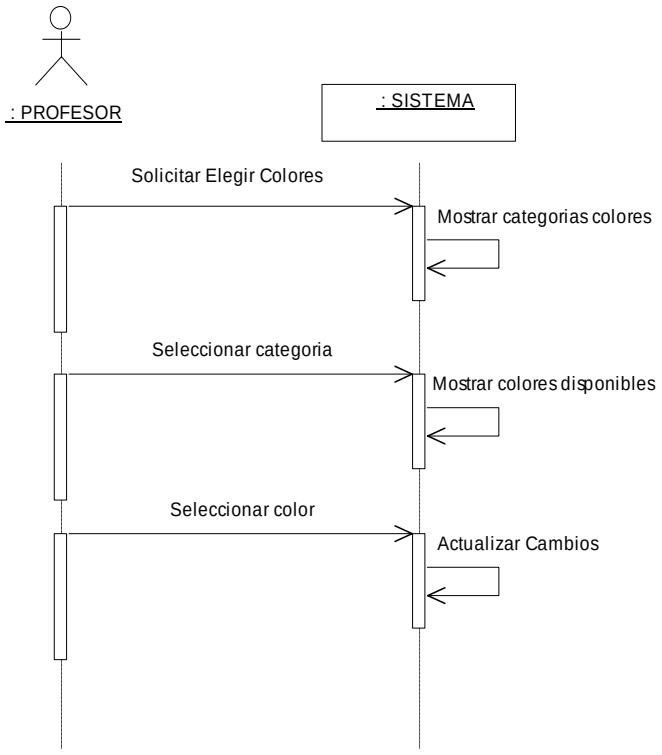


Figura 9.20 Diagrama del caso de uso Guardar Frase.

9.4.10 Elegir Configuración.

UC-00010	Elegir Configuración	
Descripción	El sistema deberá permitir al profesor el método de escritura que considere mas conveniente según el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor profesor (ACT-002) solicita modificar el método de escritura
	2	El sistema muestra todas las configuraciones existentes
	3	El actor profesor (ACT-002) selecciona uno de las configuraciones.
	4	El sistema actualiza los cambios

Figura 9.21 Descripción del caso de uso Elegir Configuración.

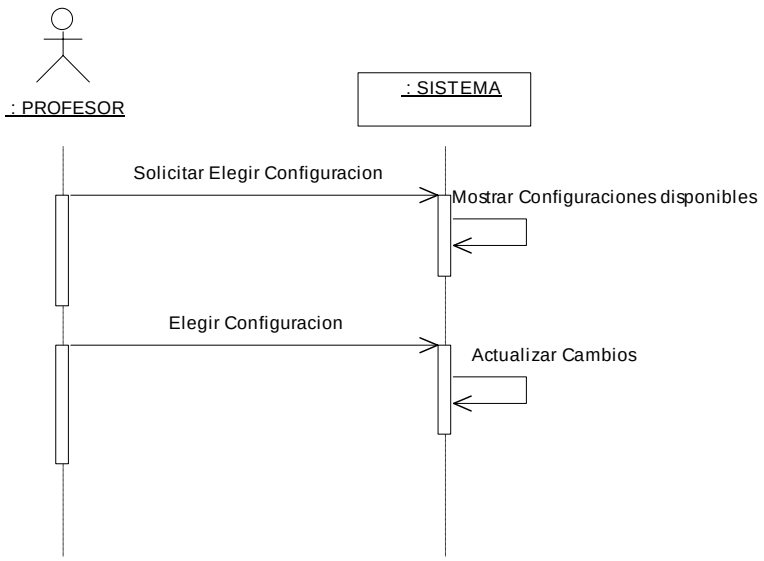


Figura 9.22 Diagrama del caso de uso Elegir Configuración.

9.4.11 Elegir Tiempo de Barrido.

UC-00011	Elegir Tiempo de Barrido	
Descripción	El sistema deberá permitir al profesor seleccionar el tiempo de barrido	
Secuencia Normal	Paso	Acción
	1	El actor profesor (ACT-002) solicita modificar el tiempo de barrido
	2	El sistema muestra todos los tiempos disponibles
	3	El actor profesor (ACT-002) elige un tiempo.
	4	El sistema actualiza los cambios

Figura 9.23 Descripción del caso de uso Elegir Tiempo de Barrido.

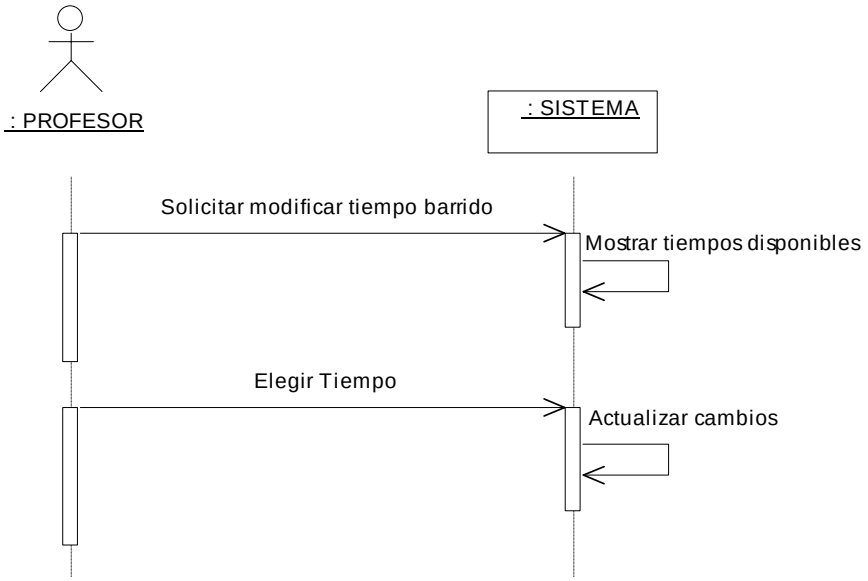


Figura 9.24 Diagrama del caso de uso Elegir Tiempo Barrido.

9.4.12 Administrar Sistema.

UC-0007	Administrar Sistema	
Descripción	El sistema deberá permitir al administrador realizar cambios en el aplicación según se describe en el siguiente caso de uso	
Secuencia Normal	Paso	Acción
	1	El administrador (ACT-003) realiza cambios en la aplicación
	4	El sistema actualiza los cambios

Figura 9.25 Descripción del caso de uso Administrar Sistema.

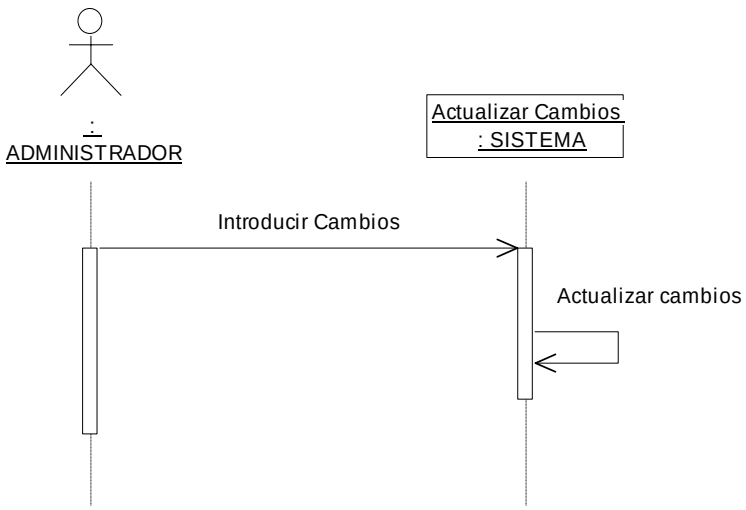


Figura 9.26 Diagrama del caso de uso Administrar Sistema.

9.5 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.

9.5.1 Diagrama inicial de clases.

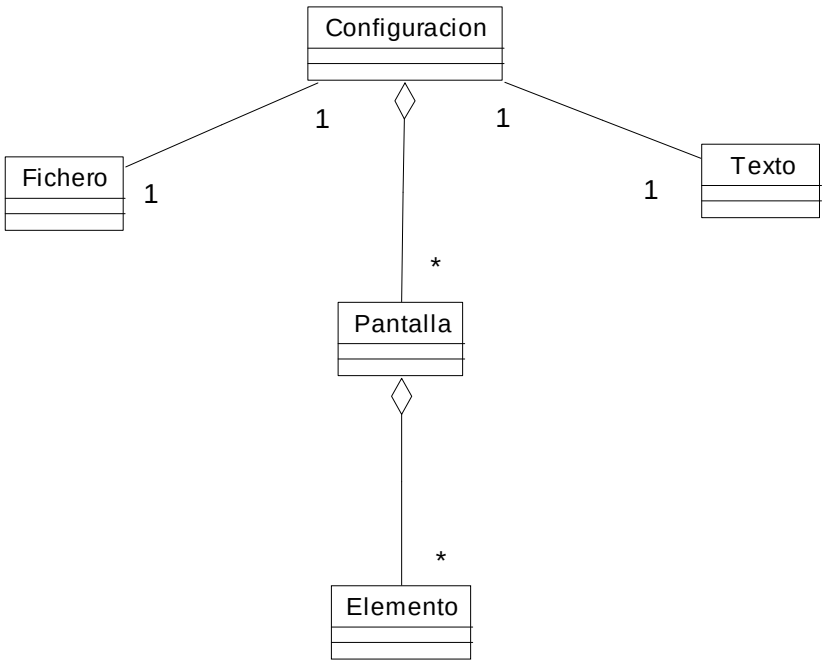


Figura 9.27 Diagrama Inicial de Clases.

Capítulo 10. 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.

En esta parte del diseño se desarrollan los casos de uso que habían sido descritos anteriormente.
También serán desarrollados los diagramas de secuencia mostrando el orden de las llamadas en el sistema. Se utilizara un diagrama de secuencia para cada caso de uso a representar y para las excepciones que puedan presentar estos casos.
Es imposible representar en un solo diagrama de secuencia todas las secuencias posibles del sistema, por ello de escoge un punto de partida.

El diagrama se realiza 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 esta activo.

Los diagramas de secuencia mostraran los mensajes que los objetos participantes intercambian entre ellos a lo largo del tiempo.

10.1 Casos de Uso.

10.1. 1 Consultar Ayuda

UC-0001	**Consultar Ayuda**		
Descripción	El sistema deberá permitir al usuario poder consultar la ayuda sobre el uso de determinadas funciones de la aplicación en cualquier momento según se describe en el siguiente caso de uso		
Secuencia Normal	**Paso**	**Acción**	
	1	El actor Alumno(ACT-0001) selecciona con el pulsador el botón de ayuda	
	2	El sistema despliega todos los temas referentes a la aplicación que disponen de ayuda	
	3	El actor Alumno (ACT-0001) si quiere consultar algún tema de ayuda existente presiona el pulsador cuando el botón referente a ese tema este seleccionado	
	4	El sistema despliega un formulario con la información referente a ese tema	
	5	El actor Alumno (ACT-0001) puede elegir las siguientes opciones:	
		5a	situarse en el formulario correspondiente de la ayuda deseada
		5b	volver al menú de ayuda
		5c	volver al punto donde se encontraba antes de solicitar la ayuda
Importancia	importante		
Comentarios	El actor puede solicitar la ayuda en cualquier método de escritura ya que junto con las letras aparecerá el botón AYUDA o podrá encontrarla dentro del menú principal de la aplicación.		

Figura 10.1 Caso de uso Consultar Ayuda.

10.1.2 Diagrama de Secuencia Consultar Ayuda.

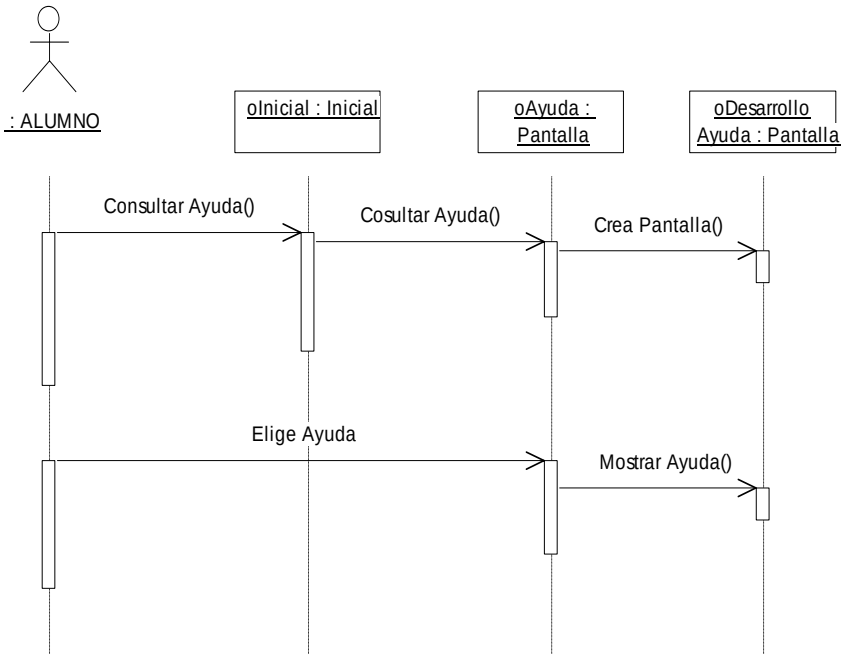


Figura 10.2 Diagrama de secuencia Consultar Ayuda.

10.1. 3 Escribir Texto Imagen.

UC-0002	Escribir Texto Imagen	
Descripción	El sistema deberá permitir al usuario escribir texto mediante la selección de imágenes que tienen palabras o frases asociadas según se describe en el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor Alumno (ACT-0001) elige una de las imágenes que el sistema dispone
	2	El sistema inserta el texto asociado a esa imagen en la caja de texto
Excepciones	**Paso**	**Acción**
	2	Si elige una imagen equivocada, el actor alumno (ACT-0001) puede borrarlo el texto asociado a esa imagen pulsando el botón con el dibujo de una “X” que se encuentra junto a las imágenes
Importancia	Vital	
Comentarios	Es la base de la comunicación para las personas que no hallan desarrollado la lectura y escritura	

Figura 10.3 Caso de uso Escribir Texto Imagen.

10.1.4 Diagrama de Secuencia Escribir Texto Imagen.

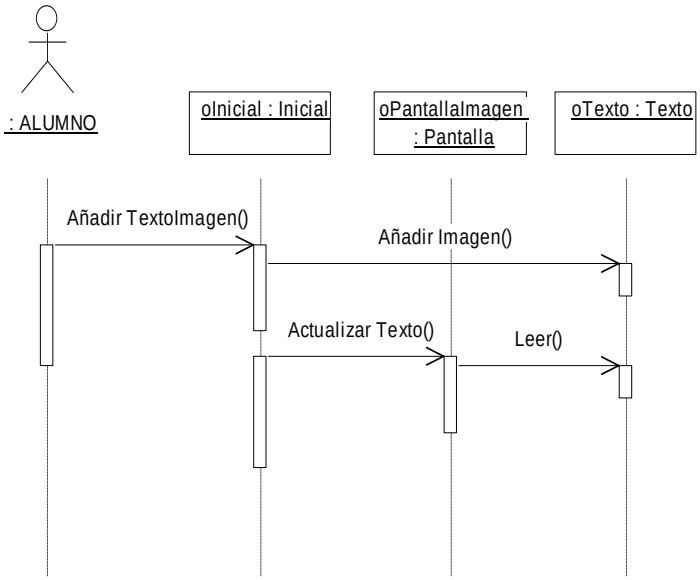


Figura 10.4 Diagrama de secuencia Escribir Texto Imagen.

10.1.5 Diagrama de Secuencia Borrar Texto Imagen.

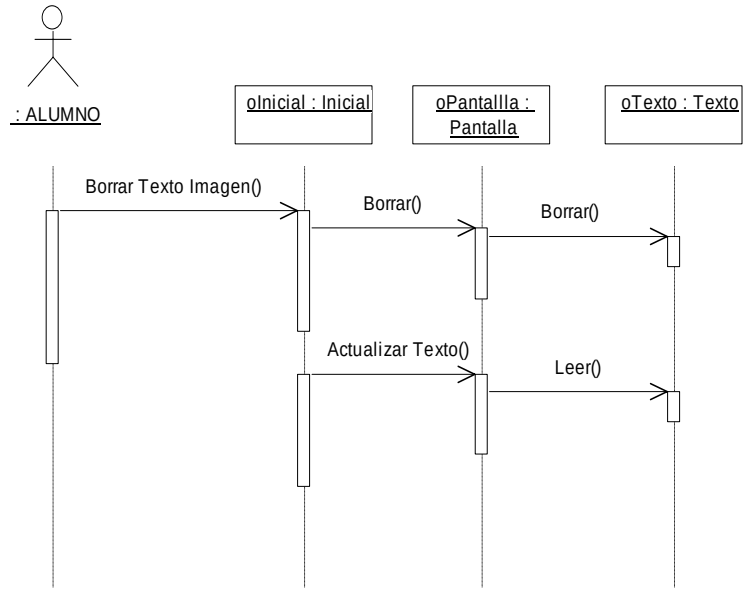


Figura 10.5 Diagrama de secuencia Borrar Texto Imagen.

10.1.6 Editar Texto.

UC-0003	Editar Texto	
Descripción	El sistema deberá permitir al usuario editar el texto escrito según el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor alumno (ACT-0001) escribe el texto que desea editar, *Caso de uso Escribir Texto* (UC-0006).
	2	El actor alumno(ACT-0001) puede en el momento que lo desee editar el texto pulsando en el menú principal EDITAR
	3	El sistema editara el texto correspondiente.
Importancia	vital	
Comentarios	Este caso de uso es importante en la comunicación ya que la otra persona podrá ver el texto escrito. Dentro de este caso de uso también se podrán realiza modificaciones al texto.	

Figura 10.6 Caso de uso Editar Texto

10.1.7 Diagrama de Secuencia Editar Texto.

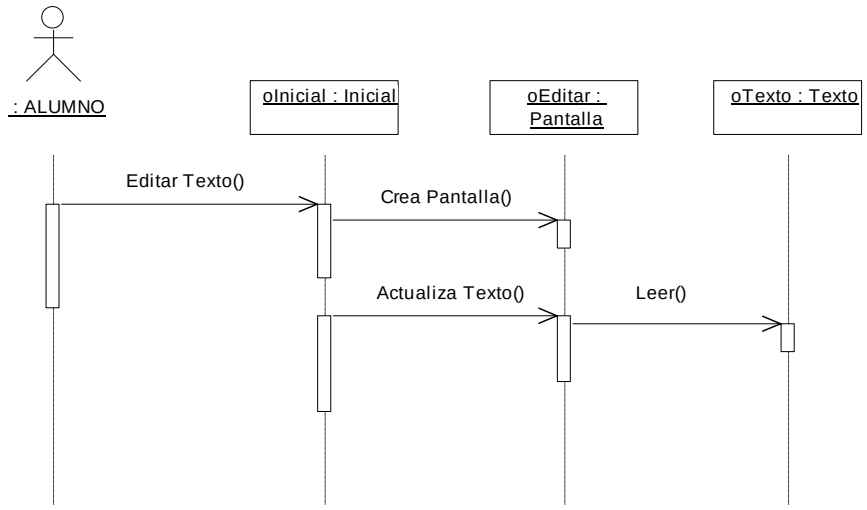


Figura 10.7 Diagrama de secuencia Editar Texto.

10.1.8 Insertar Frase.

UC-0004	**Insertar Frase**	
Descripción	El sistema deberá permitir al usuario incorporar una frase al texto que lleva escrito según se describe en el siguiente caso de uso	
Precondición	Longitud del texto es n	
Secuencia Normal	**Paso**	**Acción**
	1	El actor Alumno (ACT-0001) pulsa el botón de MENU dentro de cualquiera de los métodos de escritura existentes
	2	El sistema despliega el menú de opciones que contiene un botón FRASES
	3	El actor Alumno (ACT-0001) pulsa el botón de FRASES
	4	El sistema despliega el formulario frases que contiene el botón de insertar y una caja de texto
	5	El 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
Poscondición	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 10.8 Caso de uso Insertar Frase.

10.1.9 Diagrama de Secuencia Insertar Frase.

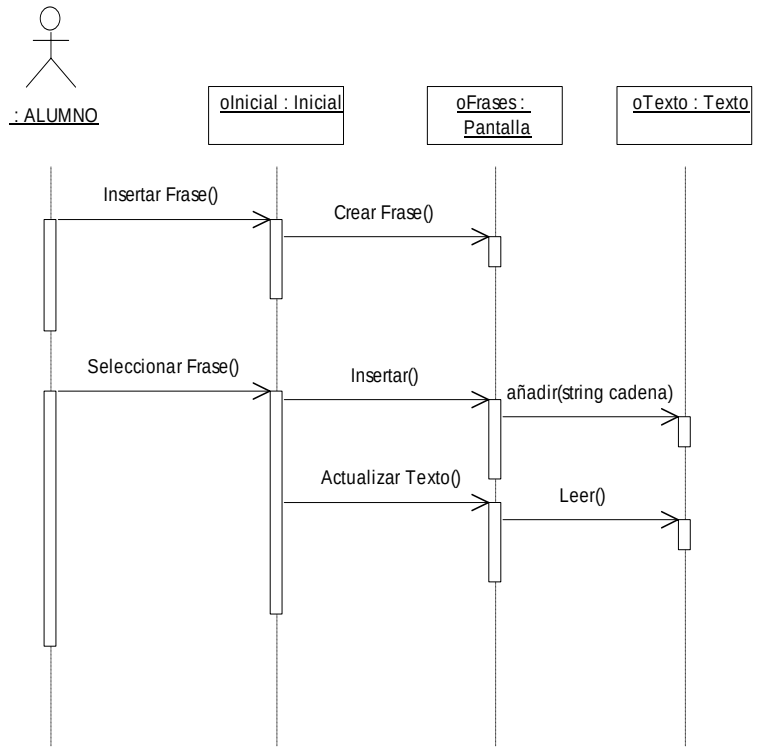


Figura 10.9 Diagrama de secuencia Insertar Frase.

10.1.10 Reproducir.

UC-0005	**Reproducir**	
Descripción	El sistema deberá permitir al usuario reproducir el texto escrito según se describe en el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	Realizar caso de uso (UC-0007) *Escribir Texto* y/o *Escribir Texto Imagen* y/o *Insertar Frase*
	2	Si el actor ha terminado de escribir, o si lo desea en algún momento, el actor alumno (ACT-0001) puede reproducir el texto pulsando dentro del menú de opciones seleccionando el botón PLAY
	3	El sistema reproduce el sonido del texto correspondiente
Importancia	Vital	
Comentarios	Este caso de uso es esencial en la comunicación ya que reproduce lo que el usuario quiere expresar no es necesario que la otra persona mire la caja de texto para saber que es lo que el usuario quiere decir	

Figura 10.10 Caso de uso Reproducir.

10.1.11 Diagrama de Secuencia Reproducir.

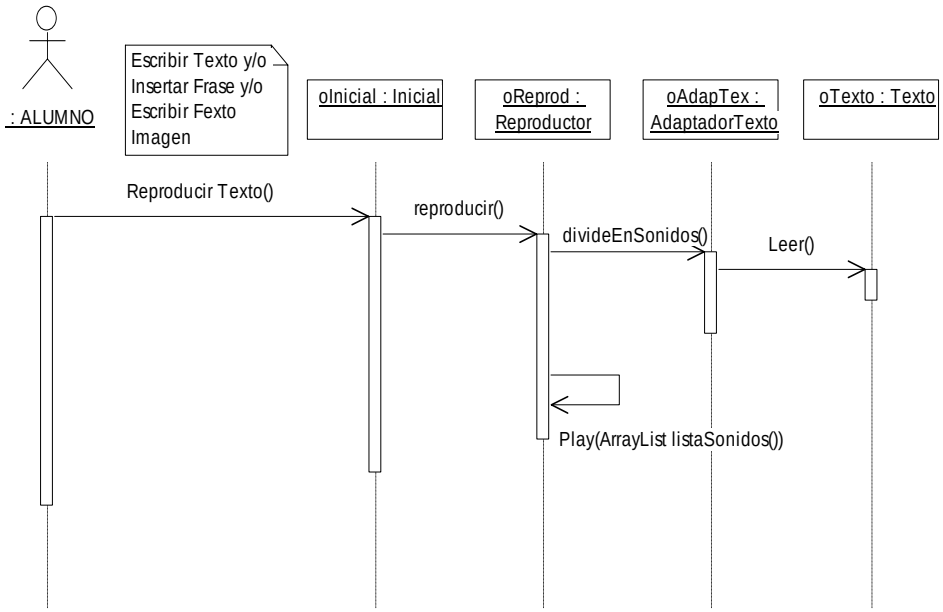


Figura 10.11 Diagrama de secuencia Reproducir.

10.1.12 Borrar Frase.

UC-0006	**Borrar Frase**	
Descripción	El sistema deberá permitir que el usuario pueda borrar en cualquier momento alguna de las frases si lo desea según se describe en el siguiente caso de uso	
Precondición	Hay n frases	
Secuencia Normal	**Paso**	**Acción**
	1	El actor Alumno (ACT-0001) pulsa el botón de MENU dentro de cualquiera de los métodos de escritura existentes
	2	El sistema despliega el menú de opciones que contiene un botón FRASES
	3	El actor Alumno (ACT-0001) pulsa el botón de FRASES
	4	El sistema despliega el formulario frases que contiene el botón borrar para eliminar una de las frases
	5	El actor alumno (ACT-0001) pulsa el botón borrar, una vez escogida la frase que quiere eliminar y estando esta seleccionada
	6	El sistema borra la frase seleccionada por el usuario
Poscondición	Hay n-1 frases	
Excepción	5	Si no existen frases que borrar, el sistema no permitirá su borrado deshabilitando el botón borrar
Importancia	importante	
Comentarios	Este caso de uso se dará siempre que el usuario considere que alguna de las frases que tiene almacenadas no son lo suficientemente usadas por el	

Figura 10.12 Caso de uso Borrar Frase.

10.1.13 Diagrama de Secuencia Borrar Frase.

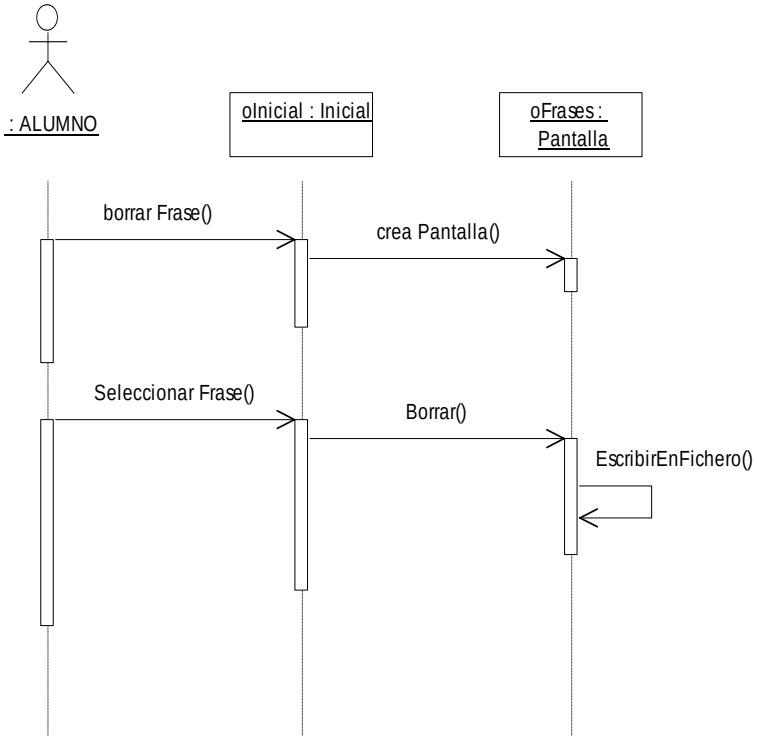


Figura 10.13 Diagrama de secuencia Borrar Frase.

10.1.14 Escribir Texto.

UC-0007	Escribir Texto	
Descripción	El sistema deberá permitir al usuario escribir en diferentes métodos de escritura cualquier texto según el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El sistema proporciona varios métodos de escritura a elegir por el actor
	2	El actor Alumno (ACT-0001) va escogiendo mediante pulsaciones letras, números, símbolos que quiere escribir
Excepciones	**Paso**	**Acción**
	2	Si se escribe algún carácter equivocado, el actor alumno (ACT-0001) puede borrarlo pulsando el botón BORRAR que se encuentra en el método de escritura junto a las letras.
Importancia	Vital	
Comentarios	Es la base de la comunicación	

Figura 10.14 Caso de uso escribir texto.

10.1.15 Diagrama de Secuencia Escribir Texto.

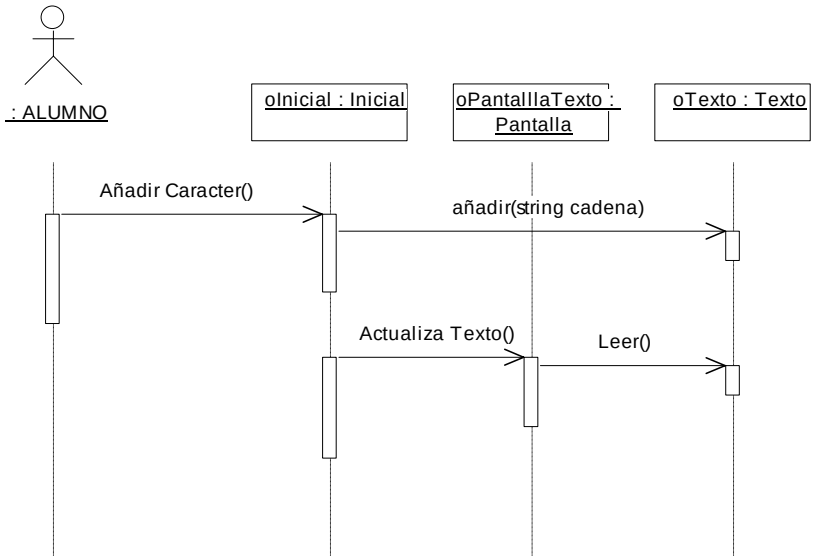


Figura 10.15 Diagrama de secuencia Escribir Texto.

10.1.16 Guardar Frase.

UC-0008	**Guardar Frase**	
Descripción	El sistema deberá permitir que el usuario pueda almacenar las frases que quiera para su posterior uso si lo desea según se describe en el siguiente caso de uso	
Precondición	Hay n frases	
Secuencia Normal	**Paso**	**Acción**
	1	Realizar caso de uso (UC-0007) *Escribir Texto* y/o (UC-0002) *Escribir Texto Imagen*
	2	El 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	El actor alumno (ACT-0001) pulsa el botón FRASES
	5	El sistema despliega el formulario frases que contiene varios botones y las frases almacenadas
	6	El 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
Poscondición	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 10.16 Caso de uso Guardar Frase.

10.1.17 Diagrama de Secuencia Guardar Frase.

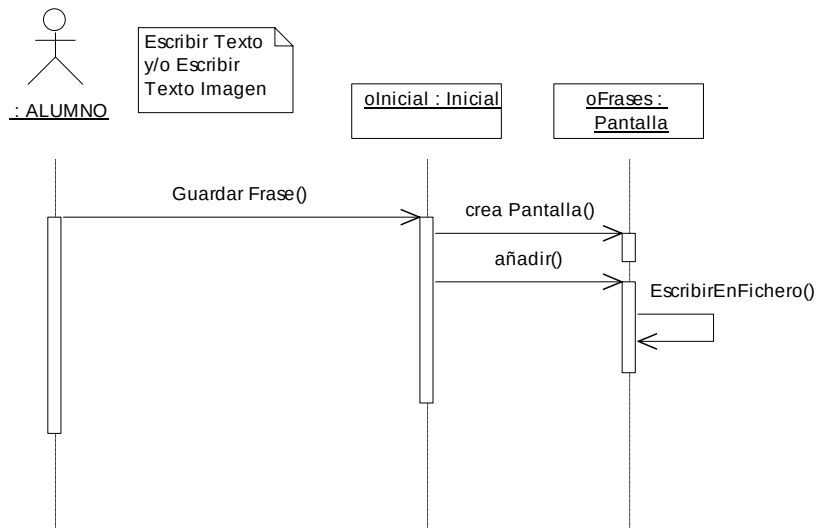


Figura 10.17 Diagrama de secuencia Guardar Frase.

10.1.18 Seleccionar Colores.

UC-0009	**Seleccionar Colores**	
Descripción	El sistema deberá permitir al usuario elegir los colores de todos los elementos que aparecen en la aplicación según se describe en el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor Profesor (ACT-0001) pulsa el botón de menú
	2	El sistema despliega los temas que contiene el menú
	3	El actor Profesor (ACT-0001) pulsa el botón de colores
	2	El sistema le muestra todos los elemento que disponen de colores
	3	El actor Profesor (ACT-0001) selecciona el tipo de elemento que quiere cambiar de color
	4	El sistema muestra los colores disponibles para ese tipo de elementos
	5	Si el actor Profesor (ACT-0001) quiere cambiar el color pulsa en el color que desea cuando este seleccionado
	6	El sistema actualiza los cambios
Poscondición	Estado actual modificado	
Importancia	Quedaría bien	
Comentarios	Si el Profesor no deseara cambiar el color podría volver la menú pulsando el botón volver.	

Figura 10.18 Caso de uso Seleccionar Colores.

10.1.19 Diagrama de Secuencia Seleccionar Colores.

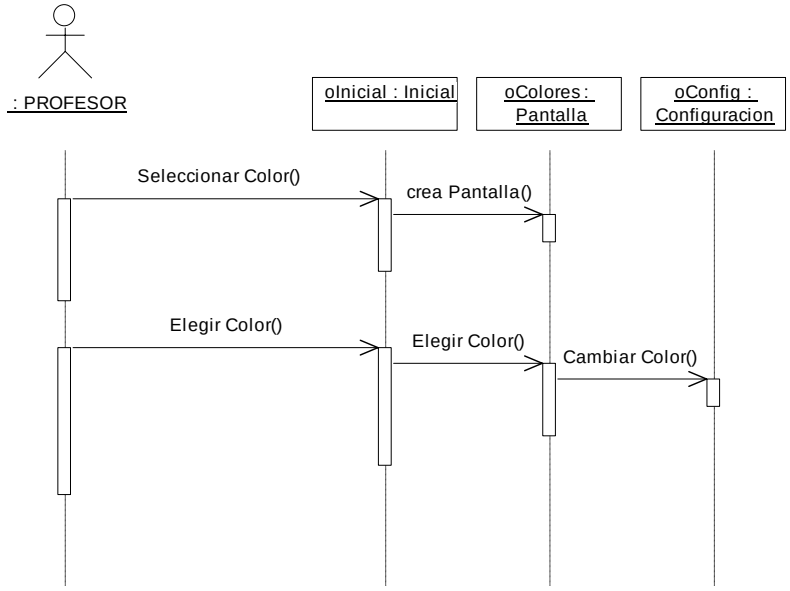


Figura 10.19 Diagrama de secuencia Seleccionar colores

10.1.20 Elegir Configuración.

UC-00010	Elegir Configuración	
Descripción	El sistema deberá permitir al profesor elegir el método de escritura que considere mas conveniente según el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor profesor (ACT-0002) presiona el pulsador cuando este seleccionado MENU dentro de cualquiera de los métodos de escritura existentes
	2	El profesor (ACT-0002) pulsa dentro de las opciones del menú CONFIGURACIONES
	3	El sistema muestra todas las configuraciones existentes
	4	Si el actor profesor (ACT-0002) quiere modificar la configuración existente pulsa sobre el método de escritura que quiera utilizar
	5	El sistema actualiza los cambios
Poscondición	Estado actual modificado	
Importancia	importante	
Comentarios	Este caso de uso se puede realizar en cualquier momento desde el menú principal	

Figura 10.20 Caso de uso Elegir Configuración.

10.1.21 Diagrama de Secuencia Elegir Configuración.

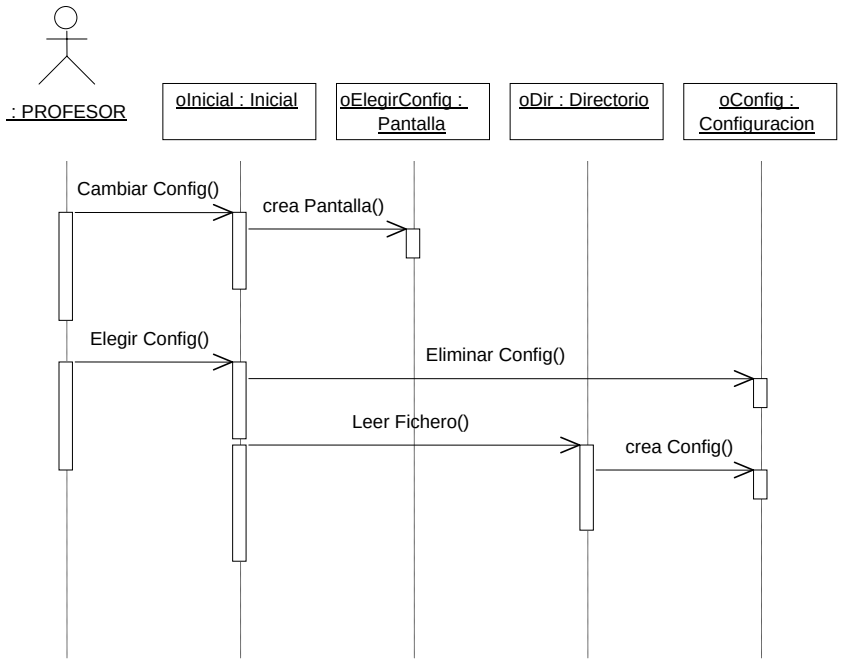


Figura 10.21 Diagrama de secuencia Elegir Configuración.

10.1.22 Elegir Tiempo de Barrido.

UC-00011	**Elegir Tiempo de Barrido**	
Descripción	El sistema deberá permitir al profesor seleccionar el tiempo de barrido, es decir el tiempo que el cursor permanece en un mismo elemento dentro de la pantalla según describe el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor profesor (ACT-0002) presiona el pulsador cuando este seleccionado MENU dentro de cualquiera de los métodos de escritura existentes
	2	El actor profesor (ACT-0002) pulsa dentro de las opciones del menú CONFIGURACIONES
	3	El sistema muestra un reloj donde se podrá elegir el tiempo de barrido
	4	El actor profesor (ACT-0002) elige un tiempo.
	5	El sistema actualiza los cambios
Poscondición	Estado actual modificado	
Importancia	importante	
Comentarios	Este caso de uso influirá en el tiempo que se tarda en escribir, cuanto mayor sea el tiempo de barrido mayor será el retardo y mayor será el tiempo medio de escritura por carácter.	

Figura 10.22 Caso de uso Elegir Tiempo Barrido.

10.1.23 Diagrama de Secuencia Elegir Tiempo de Barrido.

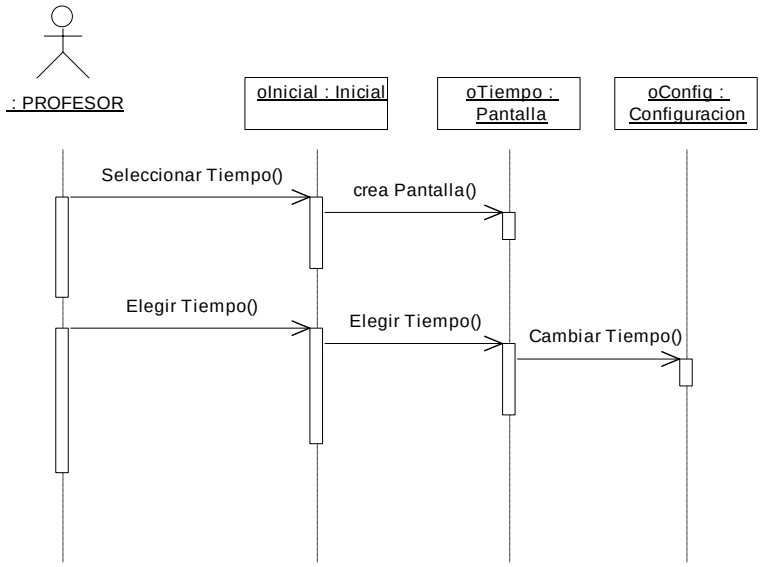


Figura 10.23 Diagrama de secuencia Elegir Tiempo de Barrido.

10.1.24 Administrar Sistema.

UC-00012	Administrar Sistema	
Descripción	El sistema deberá permitir al administrador realizar cambios en los métodos de escritura que contiene la aplicación según se describe en el siguiente caso de uso	
Secuencia Normal	**Paso**	**Acción**
	1	El actor administrador (ACT-0003) introduce los cambios que quiera realizar en el fichero que contiene toda la información referente a los métodos de escritura
	2	El sistema actualiza los cambios
Poscondición	Estado actual modificado	
Importancia	importante	
Comentarios	Este caso de uso permite la modificación de cualquiera de los métodos de escritura: orden de aparición de los elementos en la pantalla, contenido de estos, número de elementos por pantalla	

Figura 10.24 Caso de uso Administra Sistema.

10.1.23 Diagrama de Secuencia Administrar Sistema.

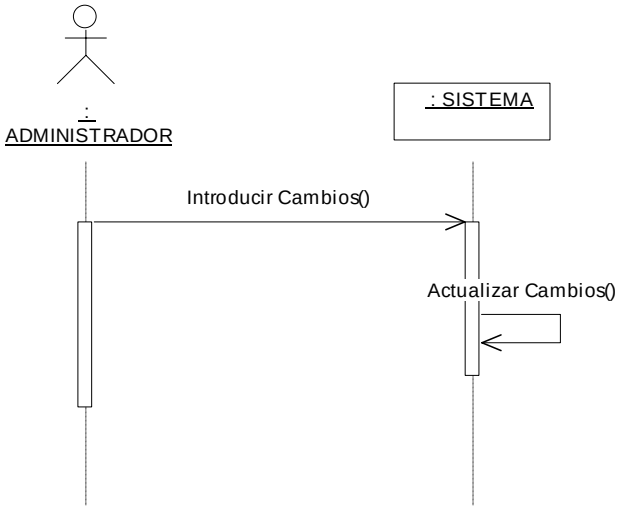


Figura 10.23 Diagrama de secuencia Administrar Sistema.

10.2 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:

- Clase Pantalla inicial: Clase encargada de iniciar la aplicación arrancando la configuración que se ejecutó por última vez.
- Clase Directorio: Clase encargada de leer el fichero que contiene los datos necesarios para crear una configuración y sus pantallas correspondientes.
- Clase Configuración: Clase encargada de almacenar las pantallas.
- Clase Texto: Clase que almacena el texto escrito en cada momento e incluye los métodos que se encargan de su tratamiento y modificación.
- Clase Pantalla: Clase que contiene los datos que caracterizan a una pantalla, así como sus elementos y los métodos que se encargan de su apariencia y funcionamiento.
- Clase PCaracteres; Clase derivada de la clase Pantalla que implementa una pantalla cuyos botones son de tipo Caracteres.
- Clase CaracterBoton; Clase que hereda de la Clase Botón y que implementa los botones de tipo Caracteres.
- Clase BotonImagen: Clase que hereda de la Clase Botón (System.Windows.Forms.Button) y que implementa los botones de tipo Imágenes.

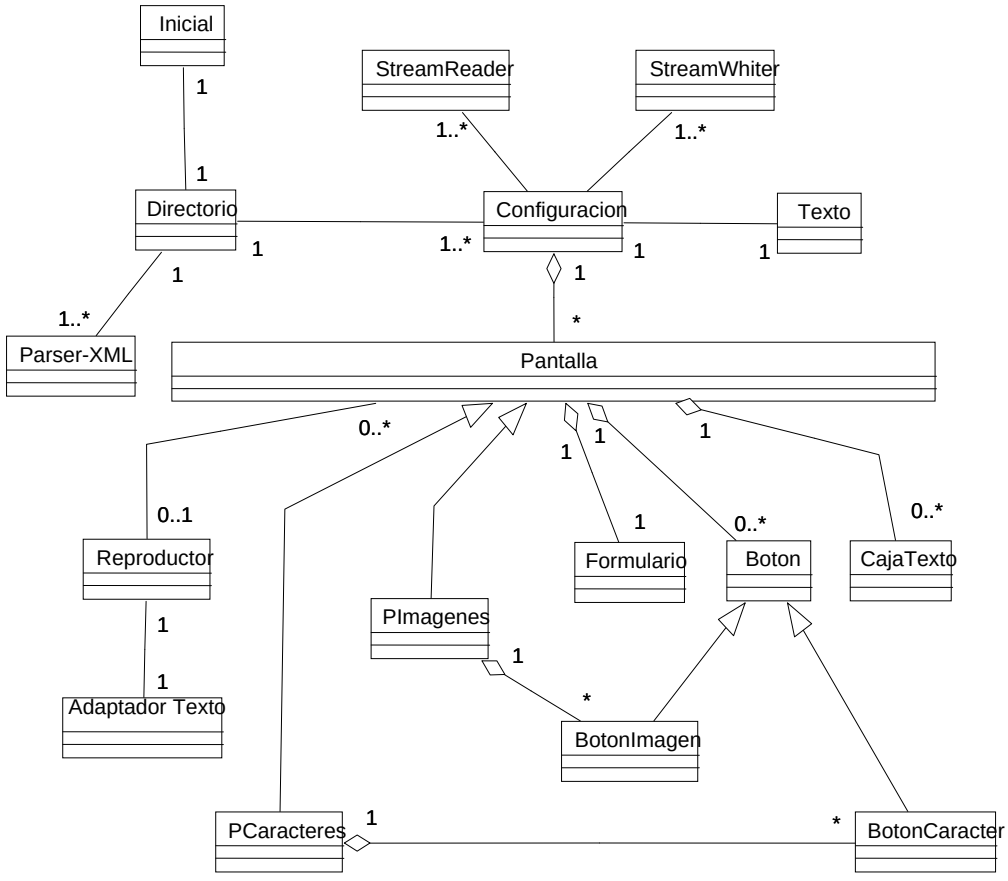


Figura 10.24 Diagrama de Clases.

10.3 Diccionario de Datos.

➢ **Clase Pantalla_inicial**

Clase encargada de iniciar la aplicación arrancando la configuración que se ejecutó por última vez.

- Atributos
  - Dir: Objeto de la clase Directorio encargado de leer el fichero que contiene los datos de la última configuración ejecutada para así poder crear las pantallas correspondientes a dicha configuración.
- Métodos
  - No contiene métodos.

➢ **Clase Directorio**

Clase encargada de leer el fichero que contiene los datos necesarios para crear una configuración y sus pantallas correspondientes.

- Atributos
  - Ficheros: Array que contiene los nombres de todos los ficheros que contienen los datos de cada configuración.
  - NombresConfig: Array que contiene los nombres de cada configuración.
  - Config: Objeto de la clase Configuración que almacena todas las pantallas necesarias.
- Métodos
  - Leer_nombres_ficheros (): se encarga de almacenar en el Array Ficheros los nombres de todos los archivos que representen alguna configuración.
  - Leer_nombres_config (): se encarga de almacenar en el Array NombresConfig el nombre de cada configuración.
  - Leer_fichero (string fichero): lee los datos del fichero correspondiente a la configuración que se va a ejecutar.
  - Crear_pantallas (string fichero): crea las pantallas asociadas a la configuración actual con los datos recogidos del fichero.

➢ **Clase Configuración**

Clase encargada de almacenar las pantallas.

- Atributos
  - Nombre: guarda el nombre de la configuración actual.
  - Tipo: guarda el tipo de la configuración actual, puede ser de “Caracteres” ó de “Imágenes.
  - Periodo: guarda el tiempo de barrido, es decir, el tiempo que permanece un botón con el foco.
  - Pantallas: Array que almacena las pantallas.
  - Fondo, color_botones, color_bot_selec, color_letra: almacenan el color actual del fondo de pantalla, de los botones, de los botones seleccionados y de las letras.
  - Textoescrito: objeto de la clase Texto que contiene el texto escrito en cada momento.

- Métodos
    - Leer_ultima_config (): lee el fichero “Configuracion” que contiene los datos necesarios para identificar la última configuración ejecutada.
    - Guardar_config(): almacena en el fichero “Configuracion” las características de la configuración actual.
    - Eliminar_config(): destruye la configuración actual y las pantallas que contiene.

➢ **Clase Texto**

Clase que almacena el texto escrito en cada momento e incluye los métodos que se encargan de su tratamiento y modificación.

- Atributos
    - Cadena: contiene el texto escrito en cada instante.
    - Ultima_palabra: contiene la última palabra escrita.
    - UltimasImagenes: contiene una lista de todas las expresiones asociadas a cada imagen seleccionada.
- Métodos
    - Añadir (string caracteres): añade a cadena lo último que se ha escrito (caracteres).
    - Añadir_Imagen (string caracteres): añade a cadena y a UltimasImagenes la expresión asociada a la última imagen seleccionada.
    - Leer (): devuelve el valor de cadena.
    - Borrar (): elimina de cadena el último carácter escrito.
    - Borrar_Imagen (): elimina de cadena y de UltimasImagenes la última expresión asociada a una imagen insertada.
    - BorrarUltima_palabra (): elimina de cadena la última palabra escrita.
    - ActualizarUltima_palabra (): busca en cadena la última palabra escrita incluyendo espacios en blanco.
    - BorrarTodo (): elimina todo el texto escrito y vacía a cadena.

➢ **Clase Pantalla**

Clase que contiene los datos que caracterizan a una pantalla, así como sus elementos y los métodos que se encargan de su apariencia y funcionamiento.

- Atributos
    - Form: formulario en el que se visualizarán los elementos que contiene pantalla.
    - CajaTexto: cuadro de texto en el que se muestra el texto escrito en cada momento.
    - Timer1: se encarga de actualizar el botón seleccionado cada período de tiempo.
    - TextoBotones: almacena una lista de los caracteres a mostrar por cada botón si la configuración es tipo “Caracteres”, y los nombres de los archivos de las imágenes si es tipo “Imágenes”.
    - Funciones: almacena una lista de las funciones de cada botón, como es el mostrar una nueva pantalla, el añadir texto a la cadena escrita o funciones específicas implementadas por métodos.

- Fil, Col: Número que indica cuántas filas y columnas de botones se van a crear.
- Nbotones: Número que indica cuántos botones contendrá la pantalla.
- Nelementos: Número que indica cuántos elementos hay asociados a una pantalla..
- Fijos_ini, Fijos_fin: Número de botones iniciales y finales de la pantalla cuyos valores y funciones no serán modificados, a los que no se les podrá modificar los elementos asignados.
- No_fijos: Número de botones de la pantalla cuyos elementos asignados son susceptibles de ser modificados.
- S: Número que indica el índice del botón que posee el foco y que va variando con el tiempo.
- Vuelta: Variable que almacena el número de ciclos que ha recorrido el foco por todos los botones.
- Marca: Variable que almacena el índice del siguiente grupo de elementos a mostrar.
- Config: Objeto de la Clase Configuración que permite acceder al resto de pantallas.

- Métodos
  - Mostrar_pantalla (int k): se encarga de activar otra pantalla y de mostrar el formulario de ésta.
  - Crear_CajaTexto (): crea un cuadro de texto nuevo y lo añade al formulario asociado a la pantalla.
  - Habilitar_timer (): pone en funcionamiento el barrido de los botones actualizando el foco al valor inicial.
  - Deshabilitar_timer (): detiene el barrido de los botones.
  - Actualizar_foco (): se encarga de trasladar el foco de un botón a otro.
  - Funcion_boton (int i): se encarga de llamar al método oportuno para que se ejecute la función asociada al elemento que contiene el botón pulsado.
  - ActualizaTexto (): se encarga de controlar el texto que se debe mostrar en el cuadro de texto cada vez que se realiza alguna modificación.

➢ Clase PCaracteres

Clase derivada de la clase Pantalla que implementa una pantalla cuyos botones son de tipo Caracteres.

- Atributos
  - Los derivados de la clase Pantalla.
  - Botones: lista de botones de tipo Caracteres, es decir, lista de objetos de la clase BotonCaracter.
- Métodos
  - Crear_Botones (): se encarga de crear un número de botones según el número de filas y columnas deseadas.
  - Poner_colores (): asigna el color correspondiente al fondo de pantalla, a los botones y a la letra.
  - con el de botones.

- Mostrar_Botones (): se encarga de asignar a cada botón el elemento que lo corresponde.
- Cambiar_Botones (): se encarga de reasignar a los botones no fijos nuevos grupos de elementos en el caso de que no coincidan el número de elementos
- Establecer_foco (): asocia el foco al botón que le corresponde.

- Arriba (): se encarga de subir una línea el texto que contiene la caja de texto de la pantalla EDITAR.
- Abajo (): baja una línea el texto que contiene la caja de texto de la pantalla EDITAR.
- Arriba_frase (): activa la frase anterior.
- Abajo_frase (): activa la frase posterior.

Clase PCaracteres

Clase derivada de la clase Pantalla que implementa una pantalla cuyos botones son de tipo Imágenes.

- Atributos
  - Los derivados de la clase Pantalla.
  - Botones: lista de botones de tipo Imágenes, es decir, lista de objetos de la clase BotonImagen.
- Métodos
  - Mismos métodos que en la clase PCaracteres.

Clase CaracterBoton

Clase que hereda de la Clase Botón (System.Windows.Forms.Button) y que implementa los botones de tipo Caracteres.

- Atributos
  - Los derivados de la clase Botón.
  - Indice: número que permite distinguir los botones unos de otros.
- Métodos
  - Los derivados de la clase Botón.

Clase BotonImagen

Clase que hereda de la Clase Botón (System.Windows.Forms.Button) y que implementa los botones de tipo Imágenes.

- Atributos
  - Los derivados de la clase Botón.
  - Image: objeto de la clase Image que contiene la imagen a representar sobre el botón.
  - Indice: número que permite distinguir los botones unos de otros.
  - BSelect: variable que indica si el botón contiene el foco o no.
- Métodos
  - Los derivados de la clase Botón.

Capitulo 11. IMPLEMENTACION.

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.

11.1 El entorno de programación y el lenguaje.

El entorno de programación utilizado durante todo el desarrollo del proyecto ha sido el Visual Studio.NET 2003, creado por Microsoft.

Esta plataforma permite desarrollar una amplia variedad de aplicaciones, entre ellas aplicaciones para dispositivos móviles y en concreto para Pocket PC que es la que nos interesaba para nuestro proyecto.

Visual Studio .NET nos ha permitido diseñar, crear, probar e implementar de una manera sencilla nuestra aplicación para Pocket PC.

Entre los lenguajes que incluye Visual Studio .NET estarían: Visual Basic .NET, Visual C++ .NET, J# .NET y C# .NET, este último es el lenguaje que ha sido utilizado para este proyecto. El C# es un lenguaje moderno ee innovador, orientado a objetos, que permite generar software para dispositivos móviles. No ha sido difícil familiarizarse con él por su parecido en la sintaxis a Java.

Para nuestra aplicación era fundamental el desarrollo de interfaces gráficas, Visual Studio .NET nos ha permitido diseñar estas interfaces aplicadas para PDA de forma sencilla, eligiendo colores, formas, tipos de letras para que el resultado fuesen interfaces prácticas.

Para el desarrollo de este proyecto se ha utilizado también el lenguaje XML e imágenes pertenecientes al sistema de comunicación SPC.

11.2 Utilización de ficheros.

La aplicación consta de los siguientes ficheros:

- **Fichero Word:** Este fichero contendrá las frases que están disponibles en la aplicación, el contenido de este fichero variará en función de la introducción y eliminación de frases por el usuario.

- **Fichero Word:** Fichero en el que se almacena la información correspondiente a la última configuración ejecutada.

- **Ficheros de imágenes:** Forman los archivos que contienen todas las imágenes SPC utilizadas en la aplicación.

- **Ficheros XML:** Archivos XML que contendrán toda la información de las configuraciones de la aplicación.

  Una de las características principales de la aplicación desarrollada es su capacidad para cambiar su configuración. Esto es posible gracias a la existencia de estos ficheros XML.

  Cada uno de estos archivos recoge la información sobre las interfaces gráficas desarrolladas.

  Los archivos XML especificarán el número de elementos que forman una pantalla y para cada uno de esos elementos se indicará una serie de componentes entre los que vamos a destacar:

  - Texto: Estaría representado por la etiqueta <texto>, esta etiqueta contiene el texto que aparece en los elementos ó el nombre de los archivos de las imágenes.

  - Contenido: Estaría representado por la etiqueta <contenido> en esta etiqueta se especifica la función desempeñada por cada elemento; si te lleva a otra pantalla o introduce una letra en el texto escrito etc.

  - Funcion: Estaría representado por la etiqueta <funcion > en esta etiqueta se indica la funcion del elemento que puede ser:

    - *Texto:* si es un elemento que introduce un carácter o texto asociado a imagen.
    - *Borrar*: si el elemento realiza la acción final de borrar.
    - *Pantalla:* si es un elemento que nos lleva a otra pantalla

Si se desea realizar algún cambio en relación a un elemento; como cambiar la letra que contiene un botón o su funcion bastaría con incorporar los cambios en este fichero para tenerlos en la interfaces de la PDA.

En relación a las imágenes se podrían cambiar para adáptalas a un usuario en concreto que conociese o manejase mejor ciertas imágenes, bastaría con cambiar el contenido de los elementos en XML referentes a las pantalla de imágenes.

10.3 Software Usado:

- Microsoft Windows XP Professional
- Microsoft Visual Studio .NET 2003
- SDK de PocketPc 2002
- Adobe PhotoShop CS
- Microsoft Word XP
- Rational Rose Demo Ver.4
- Borland Together 4
- Nero Wave Editor 2.0.0.29
- REM
- Imágenes SPC

10.4 Hardware Empleado:

- Equipo principal de desarrollo
  - S.O. Microsoft Windows XP Professional
  - Pentium III 1 GHz
  - 128 Mb Ram
  - HD 40Gb
- 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 206 MHz
  - 32Mb Ram
  - 32Mb ROM
- PDA de implementación: Acer N-10
  - S.O. Windows Mobile 2003
  - Procesador Intel Xcale 266 MHz
  - 64Mb Ram
  - 32Mb ROM

Capitulo 12. PRUEBAS.

En este capítulo se muestran las pruebas realizadas a la aplicación para comprobar su correcto funcionamiento.

Se ha realizado también como parte de estas pruebas un análisis de casos límites para ver la respuesta del programa y corregir si fuese necesario posibles errores que pudiesen aparecer

12.1 Pruebas Realizadas.

Descripción	Comprobación del botón BORRAR
Acción Realizada	Pulsación del botón BORRAR en cualquier configuración.
Resultado Esperado	Borrar el ultimo carácter
Resultado Obtenido	Carácter borrado
Observaciones	Correcto

Figura 12.1 Comprobación del botón BORRAR.

Descripción	Comprobación del botón ESPACIO
Acción Realizada	Pulsación del botón ESPACIO en cualquier configuración.
Resultado Esperado	Introducción de un espacio en el texto
Resultado Obtenido	Espacio introducido
Observaciones	Incorrecto
Modo de actuación	Modificado y corregido en la ultima versión de la aplicación

Figura 12.2 Comprobación del botón ESPACIO.

Descripción	Comprobación de inserción de un carácter
Acción Realizada	Pulsación del botón que contiene un carácter en cualquier configuración.
Resultado Esperado	Añadir carácter al texto escrito.
Resultado Obtenido	Carácter añadido
Observaciones	Correcto

Figura 12.3 Comprobación de inserción de un carácter

Descripción	Comprobación del botón MENU
Acción Realizada	Pulsación del botón MENU en cualquier configuración.
Resultado Esperado	Mostrar la pantalla menú
Resultado Obtenido	Pantalla menú mostrada
Observaciones	Correcto

Figura 12.4 Comprobación del botón MENU

Descripción	Comprobación del botón BORRARPALABRA
Acción Realizada	Pulsación del botón BORRARPALABRA en la pantalla editar.
Resultado Esperado	Borrar ultima palabra del texto escrito
Resultado Obtenido	Ultima palabra borrada
Observaciones	Correcto

Figura 12.5 Comprobación del botón BORRARPALABRA

Descripción	Comprobación de cambiar color fondo de pantalla
Acción Realizada	Pulsación del botón que contiene el color deseado
Resultado Esperado	Cambiar color de fondo
Resultado Obtenido	Color de fondo cambiado
Observaciones	Correcto

Figura 12.6 Comprobación de cambiar color fondo de pantalla

Descripción	Comprobación de cambiar color de los botones
Acción Realizada	Pulsación del botón que contiene el color deseado
Resultado Esperado	Cambiar color de botones
Resultado Obtenido	Color de botones cambiado
Observaciones	Correcto

Figura 12.7 Comprobación de cambiar color de los botones

Descripción	Comprobación de cambiar color de los botones seleccionados
Acción Realizada	Pulsación del botón que contiene el color deseado
Resultado Esperado	Cambiar color de los botones seleccionados
Resultado Obtenido	Color de botones seleccionados cambiados
Observaciones	Correcto

Figura 12.8 Comprobación de cambiar color de los botones seleccionados

Descripción	Comprobación de cambiar color de la letra
Acción Realizada	Pulsación del botón que contiene el color deseado
Resultado Esperado	Cambiar color de la letra
Resultado Obtenido	Color de letra cambiado
Observaciones	Correcto

Figura 12.9 Comprobación de cambiar color de la letra

Descripción	Comprobación de elegir configuración
Acción Realizada	Selección de una nueva configuración el la lista “Configuración”
Resultado Esperado	Cambiar la configuración
Resultado Obtenido	Configuración actualizada
Observaciones	Correcto

Figura 12.10 Comprobación de elegir configuración

Descripción	Comprobación de elección del tiempo de barrido
Acción Realizada	Seleccionar un nuevo tiempo en la lista “Tiempo”
Resultado Esperado	Cambiar tiempo de barrido
Resultado Obtenido	Tiempo de barrido actualizado
Observaciones	Correcto

Figura 12.11 Comprobación de elección del tiempo de barrido

Descripción	Comprobación de la ayuda
Acción Realizada	Pulsación del tema de ayuda deseado
Resultado Esperado	Mostrar ayuda
Resultado Obtenido	Ayuda mostrada
Observaciones	Correcto

Figura 12.12 Comprobación de la ayuda

Descripción	Comprobación del botón SALIR
Acción Realizada	Pulsación del botón SALIR en el menú
Resultado Esperado	Cerrar la aplicación
Resultado Obtenido	Aplicación cerrada
Observaciones	Correcto

Figura 12.13 Comprobación del botón SALIR

Descripción	Comprobación del botón VOLVER
Acción Realizada	Pulsación del botón VOLVER en el menú
Resultado Esperado	Volver a la pantalla anterior
Resultado Obtenido	Pantalla anterior mostrada
Observaciones	Correcto

Figura 12.14 Comprobación del botón VOLVER

Descripción	Comprobación de Imágenes
Acción Realizada	Pulsación del botón que contiene una imagen
Resultado Esperado	Insertar texto asociado a la imagen en el texto escrito
Resultado Obtenido	Texto asociado insertado
Observaciones	Correcto

Figura 12.15 Comprobación de Imágenes

Descripción	Comprobación del botón que borra una imagen
Acción Realizada	Pulsación del botón que contiene una cruz roja que representa borrar texto asociado a una imagen
Resultado Esperado	Borrar texto asociado a imagen
Resultado Obtenido	Texto borrado
Observaciones	Correcto

Figura 12.16 Comprobación del botón que borra una imagen

Descripción	Comprobación del botón volver en imágenes
Acción Realizada	Pulsación del botón que contiene la imagen de una flecha que representa la acción de volver
Resultado Esperado	Volver a la pantalla anterior
Resultado Obtenido	Pantalla anterior mostrada
Observaciones	Correcto

Figura 12.17 Comprobación del botón volver en imágenes

Descripción	Comprobación de Añadir frase
Acción Realizada	Pulsación del botón añadir en el formulario de Frases, después de haber seleccionado la frase con los botones de desplazamiento
Resultado Esperado	El texto de la frase seleccionada sea introducido en la caja de texto
Resultado Obtenido	Texto introducido
Observaciones	Correcto

Figura 12.18 Comprobación de Añadir frase

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

Figura 12.19 Comprobación de Borrar Frase

Descripción	Comprobación de Guardar Frase
Acción Realizada	Pulsación del botón guardar en el formulario de Frases, después de haber seleccionado la frase con los botones de desplazamiento
Resultado Esperado	frase sea guardada en el lugar seleccionado
Resultado Obtenido	Frase guardada en el lugar seleccionado
Observaciones	Correcto

Figura 12.20 Comprobación de Guardar Frase

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

Figura 12.21 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 texto para que sean activados
Resultado Esperado	Desplazamiento por todo el texto
Resultado Obtenido	Desplazamiento por el texto
Observaciones	Correcto

Figura 12.22 Comprobación de los controles de desplazamiento en editar texto

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

Figura 12.23 Comprobación del sonido

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 12.24 Comprobación de los controles de desplazamiento en el menú Frases

12.2 Análisis de Casos Límite.

Descripción	Comprobación de la flecha de desplazamiento hacia arriba en frases
Acción Realizada	Pulsación de la flecha subir cuando estemos situados en la primera frase
Resultado Esperado	Desactivar botón subir
Resultado Obtenido	Botón subir desactivado
Observaciones	Correcto

Figura 12.25 Comprobación de la flecha de desplazamiento hacia arriba en frases

Descripción	Comprobación de la flecha de desplazamiento hacia abajo en frases
Acción Realizada	Pulsación de la flecha de bajar cuando estemos situados en la ultima frase
Resultado Esperado	Desactivar botón bajar
Resultado Obtenido	Botón bajar desactivado
Observaciones	Incorrecto
Modo de actuación	Modificado y corregido en la ultima versión de la aplicación

Figura 12.26 Comprobación de la flecha de desplazamiento hacia abajo en frases

Descripción	Comprobación de la flecha de desplazamiento hacia arriba en editar
Acción Realizada	Pulsación de la flecha subir cuando estemos situados en la primera línea del texto escrito
Resultado Esperado	Desactivar botón subir
Resultado Obtenido	Botón desactivado
Observaciones	Incorrecto
Modo de actuación	Modificado y corregido en la ultima versión de la aplicación

Figura 12.27 Comprobación de la flecha de desplazamiento hacia arriba en editar

Descripción	Comprobación de la flecha de desplazamiento hacia abajo en editar
Acción Realizada	Pulsación de la flecha de bajar cuando estemos situados en la ultima línea del texto escrito
Resultado Esperado	Desactivar botón bajar
Resultado Obtenido	Botón bajar desactivado
Observaciones	Correcto

Figura 12.28 Comprobación de la flecha de desplazamiento hacia abajo en editar

Descripción	Comprobación del botón BORRAR cuando no se halla introducido texto
Acción Realizada	Pulsación del botón BORRAR
Resultado Esperado	No realizar ninguna acción
Resultado Obtenido	Ninguna acción realizada
Observaciones	Correcto

Figura 12.29 Comprobación del botón BORRAR cuando no se halla introducido texto

Descripción	Comprobación de borrar imagen cuando no se halla introducido texto asociado a una imagen
Acción Realizada	Pulsación del botón que borra el texto asociado a un imagen
Resultado Esperado	No realizar ninguna acción
Resultado Obtenido	Ninguna acción realizada
Observaciones	Correcto

Figura 12.30 Comprobación de borrar imagen cuando no se halla introducido texto asociado a una imagen

Parte IV. Manual de Usuario

Capitulo 13. MANUAL DE USUARIO.

13.1 Descripción de la aplicación

Esta aplicación se compone de un conjunto de métodos de escritura que difieren tanto en su forma de uso, como en la composición y apariencia de sus pantallas. En base a esto, podemos distinguir entre dos tipos de configuraciones, las configuraciones de tipo “Caracteres” y las configuraciones de tipo “Imágenes”. Ambas, dentro de sus tipos, también se diferencian en el número de botones que las componen, que se describen mediante filas y columnas (fila x columna).

Las configuraciones de tipo Caracteres se caracterizan porque los botones tienen asociados un conjunto de caracteres que se pueden distribuir en una nueva pantalla de forma independiente, y en las de tipo Imágenes los botones representan categorías que se despliegan en una pantalla que contendrá botones con imágenes asociados a una expresión.

Este programa le permitirá llevar a cabo las siguientes opciones:

- Escribir texto con una amplia variedad de métodos de escritura.
- Borrar texto.
- Reproducir texto.
- Editar texto.
- Disponer de frases almacenadas para agilizar la comunicación.

En las siguientes páginas se detalla el uso y funcionamiento de la aplicación para un eficaz manejo.

13.2 Pantalla de bienvenida.

Es la pantalla inicial y se muestra nada más arrancar la aplicación. Esta pantalla contiene el título de la aplicación y el nombre de los autores.

Se muestra durante unos segundos tras los cuales aparece la Pantalla Principal de la configuración ejecutada la última vez que se utilizó el programa.

- **Cuadro de Texto**

En cualquier configuración se podrá observar este elemento de la pantalla en el que se puede visualizar el texto introducido recientemente, así como sus modificaciones.
Es un cuadro blanco y siempre que aparece se encuentra situado en la parte superior de la pantalla.

13.3 Configuraciones tipo Caracteres.

Podemos distinguir entre dos metodologías diferentes aplicadas a este tipo de configuraciones: las que contienen botones que representan Conjuntos de caracteres y las que contienen botones que representan Categorías de caracteres. A continuación describimos el funcionamiento de ambas.

- **Botones Conjuntos de caracteres**
  - **Botones de Caracteres**

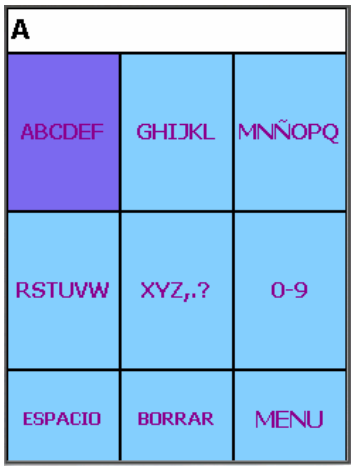


Figura 2. Pantalla Principal Conjuntos caracteres.

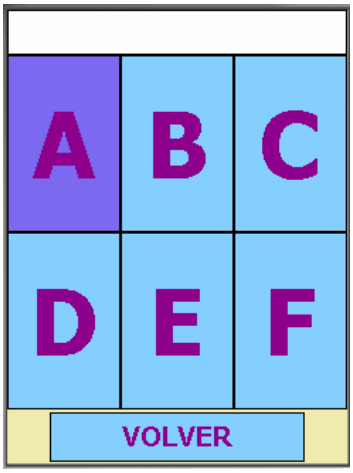


Figura 3. Pantalla Principal del Conjunto ABCDEF

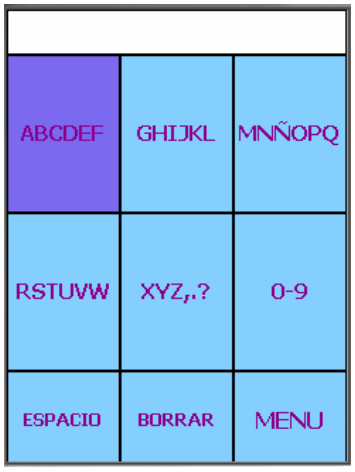
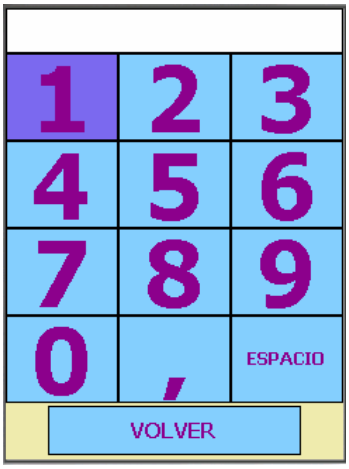


Figura 4. Pantalla desplegada con texto escrito

Los seis primeros botones representan cada uno un conjunto de caracteres, los cuales, al ser pulsados mostrarán una nueva pantalla en la que dichos caracteres se distribuirán en botones de forma independiente como se muestra en la Figura 3.

El procedimiento se detalla con el siguiente ejemplo:
Si se seleccionara el botón de la Figura 2 que contiene el grupo de caracteres ABCDEF, aparecería la pantalla de la Figura 3, en la que el conjunto de caracteres se disgrega en botones independientes. De esta manera, si el botón que representa la letra A fuera pulsado, se añadiría al cuadro de texto el carácter asociado a ese botón, que en este caso sería A. De forma automática, se pasaría a la Pantalla Principal donde podremos visualizar la letra A escrita. El resultado se puede ver en la Figura 4.

- **Botón Números**



Si se seleccionara el botón de la Figura 2 que contiene el grupo de caracteres 0-9, se mostraría la pantalla de la Figura 5, en la que aparecen los números del 0 al 9 y la coma decimal representados en botones independientes. De igual forma que en el caso anterior, al pulsar uno de los botones se añadiría al texto escrito el carácter asociado.

La única diferencia sería que al seleccionar un botón con un carácter, permaneceríamos en esta misma pantalla; para acceder a la Pantalla Principal se debe pulsar el botón VOLVER.

Figura 5. Pantalla desplegada del grupo 0-9.

- **Botones Espacio, Borrar y Menú**

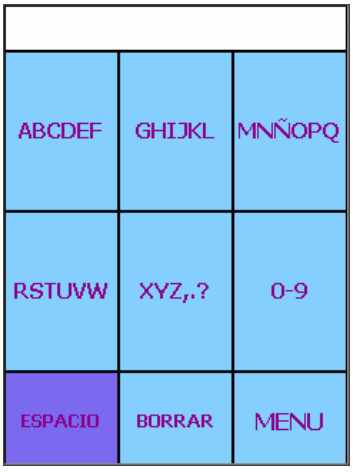


Figura 6. Botón Espacio.

Figura 7. Botón Borrar

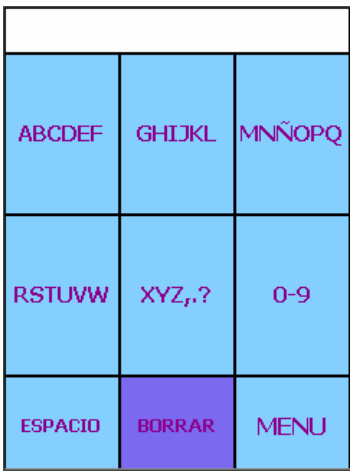


Figura 8. Botón Menú.

El **Botón Espacio**, Figura 6, introduce un espacio en blanco en el texto escrito al ser pulsado.

El **Botón Borrar**, Figura 7, tiene como función borrar el último carácter introducido en el texto.

El **Botón Menú**, Figura 8, contiene el menú principal de la aplicación. Al ser seleccionado este botón, se desplegará otra pantalla en la que se añaden nuevas funcionalidades a la aplicación contenidas en botones también. Para ver la descripción de esta pantalla véase el apartado 1.3.

- Ejemplo del resultado de aplicar el Botón Borrar:

Figura 7.1. Ejemplo Botón Borrar (1)

Figura 7.2. Ejemplo Botón Borrar (2)

En la Figura 7.1 se observa el texto escrito en la Pantalla Principal y en la Figura 7.2 se observa el texto actualizado como consecuencia de aplicar el Botón Borrar.

- **Botones Categorías de caracteres**
  - **Botones de Caracteres**

Figura 9. Pantalla Principal Categorías.

Figura 10. Pantalla secundaria de la categoría CONSONANTES

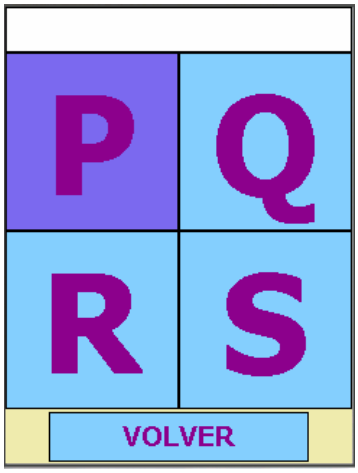


Figura 11. Pantalla desplegada Con texto escrito.

En la Figura 9 podemos observar que este tipo de botones representan las categorías “Consonantes”, “Vocales” y “Números”, de tal forma que al ser pulsados se mostrará una nueva pantalla secundaria en la que aparecerán los caracteres que pertenecen a la categoría del botón pulsado. Estas pantallas secundarias pueden contener distintas formas y número de botones, de manera que a diferente configuración tendremos distintos números y distribuciones de los botones. Este aspecto queda determinado en un número de filas y de columnas (fila x columna).

Generalmente, las categorías contienen mayor cantidad de caracteres que de botones, por lo que los caracteres se muestran de forma rotatoria por los botones.

El procedimiento que se sigue es el siguiente:
Si la pantalla secundaria contiene por ejemplo, cuatro botones asignados a los caracteres, se mostrarán los cuatro primeros caracteres pertenecientes a la categoría, el foco pasará por cada botón, y si ninguno de ellos es pulsado, cuando el foco haya recorrido todos los botones, se mostrarán los cuatro siguientes caracteres y así sucesivamente hasta mostrar todos.
En el momento que un botón es pulsado se mostrará la Pantalla Principal, por el contrario, si no se pulsa ningún botón y se han recorrido todos los caracteres de la categoría, se comenzarán a mostrar de nuevo todos los caracteres empezando por el primer grupo.

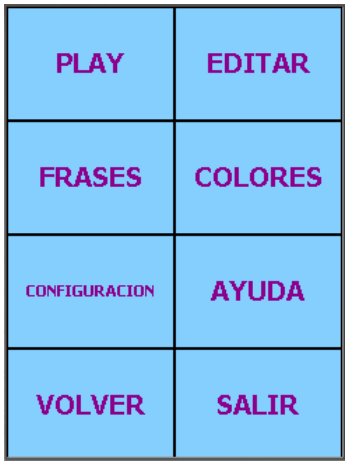
El mecanismo se detalla con un ejemplo en el que la configuración es de dos filas y dos columnas para las pantallas secundarias:
Si se seleccionara el botón de la Figura 9 que contiene la categoría CONSONANTES, aparecería la pantalla de la Figura 10, en la que se mostrarían los cuatro primeros caracteres de esta categoría, B C D y E, distribuidos en los cuatro primeros botones. De esta manera, si el carácter que deseáramos insertar fuera la letra P, esperaríamos a que el foco recorriera todos los botones y continuara mostrando nuevos cuartetos de caracteres hasta que visualicemos el que contiene este carácter, Figura 10. Una vez estuviéramos ante este carácter, pulsaríamos cuando el foco se sitúe sobre él. A continuación, aparecería la Pantalla Principal con la letra P insertada en el cuadro de texto, Figura 11.

En todas las pantallas secundarias se dispone de un botón VOLVER para el caso de no querer seleccionar ninguno de los caracteres de la categoría que corresponda. Dicho botón se puede observar en la Figura 10.

- **Botones Espacio, Borrar y Menú**

Estos botones se localizan en la parte inferior de la Pantalla Principal, Figura 9, y poseen la misma funcionalidad que en el apartado 1.2.1.

13.4 Pantalla MENU



En esta pantalla se encuentran importantes funcionalidades para tratar el texto escrito, y también ofrece la posibilidad de modificar de configuración o abandonar la aplicación.

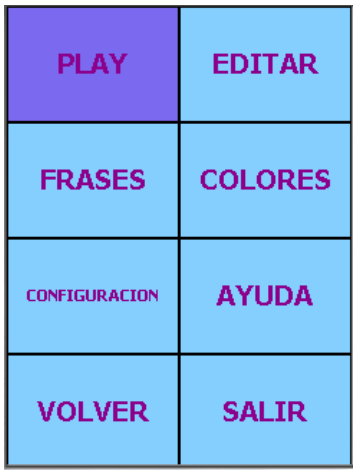
Podemos hacer uso de sus opciones en cualquier configuración, y se accede a ella pulsando el botón MENU que se encuentra en la Pantalla Principal (Ejemplo: Figura 8).

Todas las pantallas a las que se accede desde la Pantalla Menú disponen de un botón VOLVER desde el que se puede regresar de nuevo al MENU.

A continuación se describe cada una de las funcionalidades que ofrece esta pantalla.

Figura 12. Pantalla MENU

➢ PLAY (Reproducir)



El Botón Play (Figura 13) se encarga de reproducir en sonido el texto que se haya escrito hasta el momento de manera que, aunque no se pueda visualizar todo el texto escrito porque el cuadro de texto limita su aparición, éste será reproducido en su totalidad.

Figura 13. Botón PLAY

➢ Editar Texto

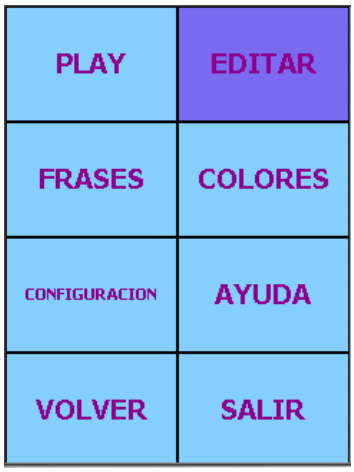


Figura 14. Botón Editar

Figura 15. Pantalla Editar

El Botón Editar muestra la Pantalla Editar que permite tratar el texto escrito hasta ese instante.

Se observa en la Figura 14 que la Pantalla Editar también ofrece la funcionalidad del botón Play.

➢ Frases

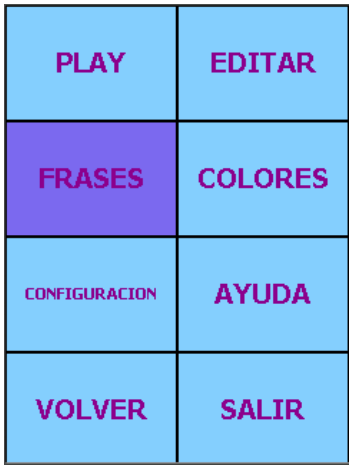


Figura 16. Botón Frases

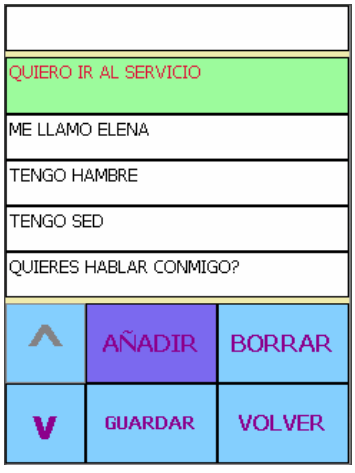


Figura 17. Pantalla Frases

El Botón Frases (Figura 16) muestra la Pantalla Frases (Figura 17) que ofrece la posibilidad de insertar frases ya creadas, además de personalizarlas para su posterior uso.

Esta opción permite agilizar la comunicación notablemente.

- **Colores**

Figura 18. Botón Colores

Figura 19. Pantalla Colores

El Botón Colores (Figura 18) muestra la Pantalla Colores (Figura 19) que permite modificar el color de fondo de la pantalla, el de los botones, el del foco de los botones o el de los caracteres.

➢ Configuración

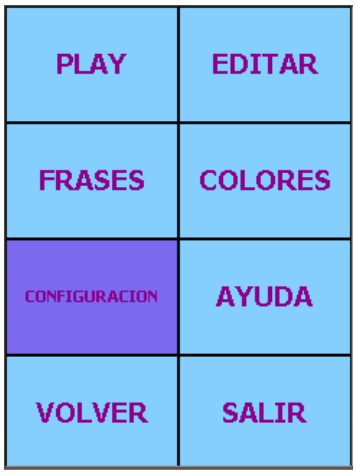


Figura 20. Botón Configuración

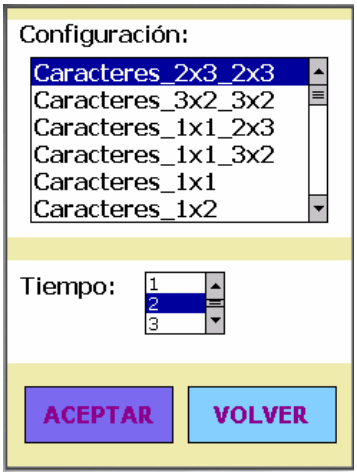


Figura 21. Pantalla Configuración

El Botón Configuración (Figura 20) muestra la Pantalla Configuración (Figura 21) que permite cambiar de configuración y elegir otra diferente.

Además ofrece la posibilidad de modificar el tiempo de barrido.

➢ Ayuda

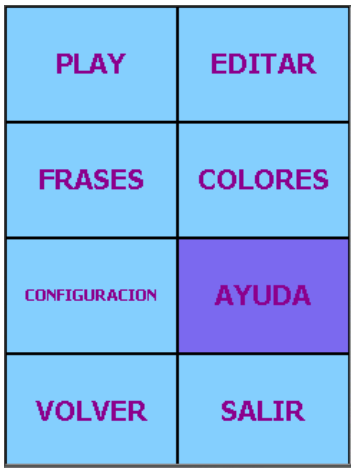


Figura 22. Botón Ayuda

Figura 23. Pantalla Ayuda

El Botón Ayuda (Figura 22) muestra la Pantalla Ayuda (Figura 23) que contiene un menú con los temas de consulta que se pueden realizar.

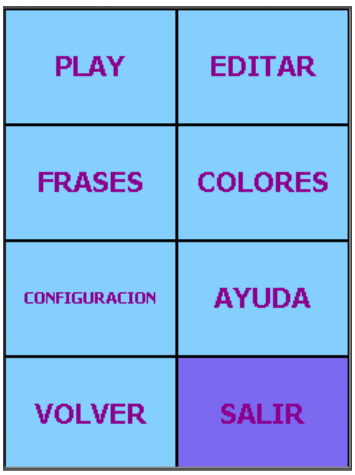
Se proporciona ayuda para facilitar el manejo de la aplicación ya que podrá ser consultada en cualquier instante desde cualquier configuración

- **Volver**

El Botón Volver (Figura 24) muestra la Pantalla Principal de la configuración en la que se encuentre. Pantalla desde donde se accedió al Menú.

Figura 24. Botón Volver

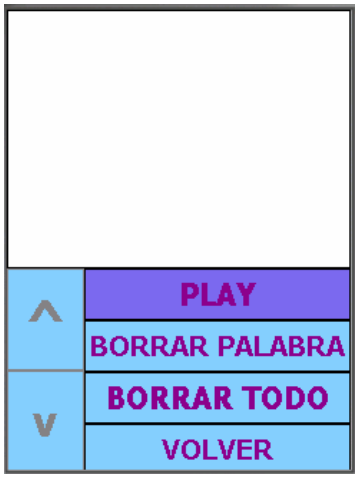
- **Salir**



El Botón Salir (Figura 25) finaliza con la ejecución de la aplicación, cerrando así el programa.

Figura 25. Botón Salir

13.5 Pantalla Editar



En esta pantalla se trata y edita el texto escrito. Contiene una Caja de Texto en la parte superior que permite visualizar una mayor proporción del texto escrito, puesto que su tamaño es mayor.

Figura 26. Pantalla Editar Texto

- **Botones y sus funciones:**
  - **Botones** de desplazamiento **^** y **V**: permiten ver todo el texto escrito hasta el momento y moverte por él de línea en línea. Estarán activos en caso de que no haya espacio suficiente en el cuadro de texto para abordar todo el texto escrito. En caso contrario, el foco no pasará por ellos además de adoptar una apariencia grisácea.
  - **Botón PLAY:** reproduce en sonido todo el texto escrito hasta el momento.
  - **Botón BORRAR PALABRA:** eliminará del texto escrito la última palabra que fue introducida. Si se añadieron espacios en blanco al final del texto, se eliminarán los espacios junto con la última palabra.
  - **Botón BORRAR TODO:** eliminará todo el texto introducido y el cuadro de texto se quedará vacío para poder empezar a escribir de nuevo.
  - **Botón VOLVER:** muestra la pantalla anterior que sería la de MENU que es desde donde se accede a la Pantalla Editar.

- Ejemplo del resultado de aplicar el Botón Borrar Palabra:

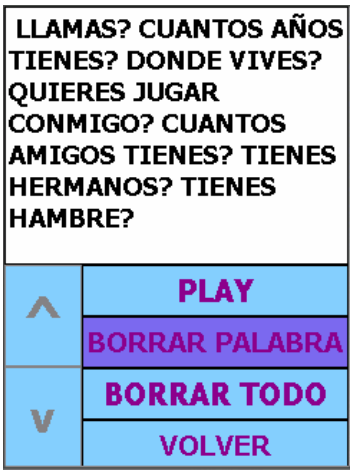


Figura 26.1. Ejemplo Botón Borrar Palabra (1)

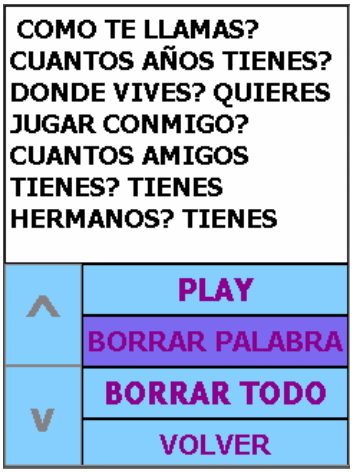
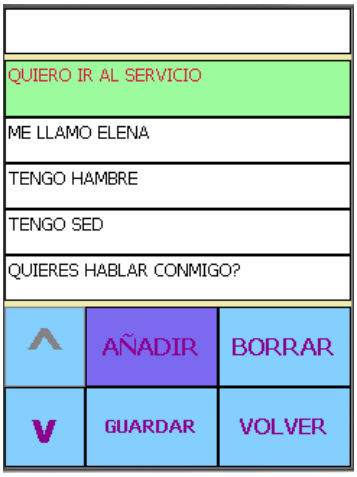


Figura 26.2. Ejemplo Botón Palabra (2)

En la Figura 26.1 se observa el texto escrito en la Pantalla Editar y en la Figura 7.2 se observa el texto actualizado como consecuencia de aplicar el Botón Borrar Palabra.

13.6 Pantalla Frases



Esta pantalla nos aporta un conjunto de frases que podemos manipular para agilizar la escritura.

Las posibilidades que tenemos son: insertar al texto frases que ya están almacenadas, guardar frases nuevas a las anteriores para poder emplearlas posteriormente y borrar las frases que no utilicemos o no creamos conveniente tener almacenadas.

Figura 27. Pantalla Frases

- **Botones y sus funciones:**
  - **Botones** de desplazamiento ^ y **V**: permiten recorrer las frases almacenadas para escoger la deseada. El botón de subir será desactivado cuando nos encontremos en la primera frase y el botón de bajar cuando nos encontremos en la última frase.
  - **Botón Añadir:** al pulsar este botón, la frase que se encuentre en el cuadro verde será añadida al texto escrito.
  - **Botón Borrar:** elimina de la lista de frases almacenada aquella que se encuentre en el cuadro verde en el momento que se pulse este botón.
  - **Botón Guardar:** añadirá el texto escrito hasta el momento a la lista de frases. Se colocará delante de la frase que se encuentre en el cuadro verde en ese instante.
  - **Botón Volver:** muestra la pantalla anterior, la Pantalla Menú.

- Ejemplo del resultado de aplicar el Botón Añadir:

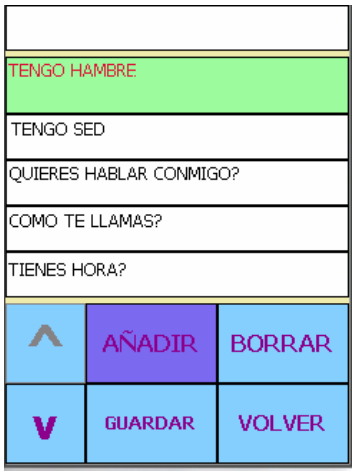


Figura 27.1. Ejemplo Botón Añadir (1)

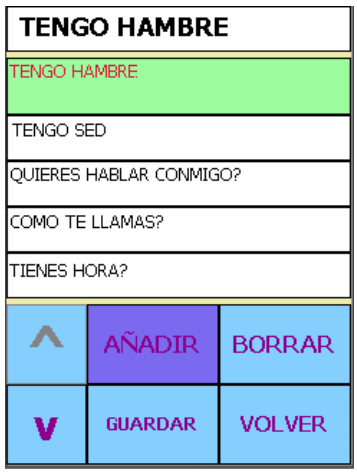
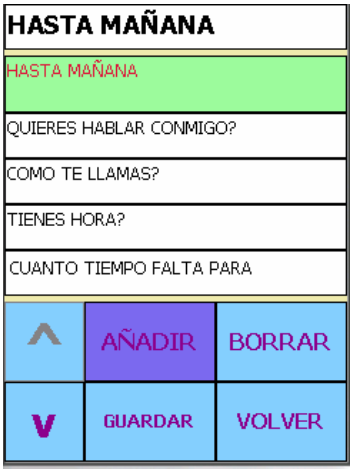


Figura 27.2. Ejemplo Botón Añadir (2)

En la Figura 27.1 se observa la Pantalla Frases con el cuadro de texto vacío y en la Figura 7.2 se observa que la frase que se encuentra en el cuadro verde ha sido insertada en la caja de texto como consecuencia de aplicar el Botón Añadir Frase.

- Ejemplo del resultado de aplicar el Botón Guardar:



Observamos que la frase escrita en el cuadro de texto ha sido añadida a la lista de frases.

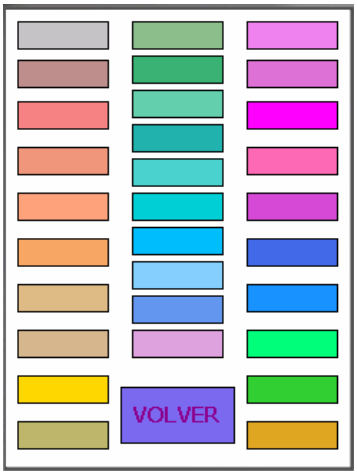
Figura 27.3. Ejemplo Botón Guardar

13.7 Pantalla Colores

Figura 28. Pantalla Colores

Esta pantalla ofrece la posibilidad de modificar el color de fondo de la pantalla, el de los botones, el de los botones que están seleccionados por el foco o el de la letra.

Al elegir cualquiera de estas cuatro opciones se desplegará una nueva pantalla, la Figura 29, con botones de diferentes colores, las tonalidades de los colores que se ofertan son diferentes para cada uno de ellos. Por ejemplo, los colores de la pantalla asociada a cambiar el color de fondo, son diferentes a los de la asociada a cambiar el color de los botones.



Al pulsar sobre cualquiera de estos botones asociados a un color, se mostrará la Pantalla Colores (Figura 28) de nuevo, y el color seleccionado será aplicado a la característica escogida.

Esta pantalla dispone de un botón VOLVER para poder regresar a la Pantalla Colores si no se desea cambiar el color o si se ha entrado erróneamente.

Figura 29. Pantalla Color Botones

13.8 Pantalla Configuración

Esta pantalla permite cambiar de configuración eligiendo otra diferente de la lista titulada “Configuración”.

Otra posibilidad que ofrece esta pantalla es la de modificar el tiempo de barrido de los botones.

Figura 30. Pantalla Configuración

En la lista “Configuración”, las configuraciones están representadas con un nombre que indica el tipo de la configuración y la forma de las pantallas:

- El tipo puede ser “Caracteres” o “Imágenes”, que son los dos únicos tipos a los que puede pertenecer una configuración.

- La forma se describe mediante el número de filas y columnas que formarán los botones destinados a la inserción de texto. De esta manera, 2x3, por ejemplo, indica que las pantallas de la configuración contendrán seis botones dedicados a la inserción de texto distribuidos en dos filas y tres columnas. En el caso de que existan dos grupos de fila x columna, por ejemplo 2x3_2x2, indica que al ser pulsado un botón de la pantalla de dos filas con tres columnas, éste desplegará una nueva pantalla en la que la distribución de los botones será 2x2, es decir de dos filas y dos columnas.

E la lista “Tiempo”, los periodos de tiempo de barrido están representados mediante números, cuanto mayor sea el número, mayor será el tiempo que el foco permanezca en un botón y, por lo tanto, el barrido será más lento.

Nota: los números son representativos, no coinciden con ninguna medida de tiempo, como los segundos.

13.9 Pantalla Ayuda

Figura 31. Pantalla Ayuda

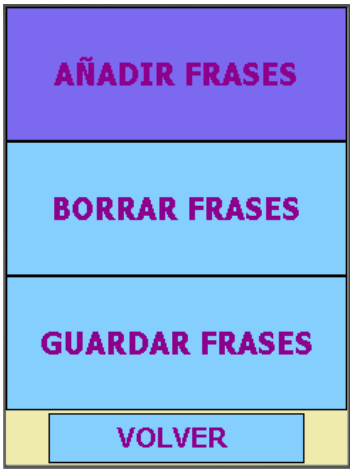


Figura 32. Pantalla Ayuda Frases

La Pantalla Ayuda, Figura 31, muestra clasificados los diferentes temas de ayuda.
Cada uno de estos temas despliega o bien otra pantalla con nuevos temas de ayuda más precisos, o bien una pantalla que contiene la información de ayuda del tema escogido.
Podemos observar un ejemplo en el que al pulsar el tema Frases en la Pantalla Ayuda (Figura 31) se despliega una nueva pantalla (Figura32) en la que se ofrecen más temas de ayuda relacionados con las funcionalidades que proporciona la Pantalla Frases. Finalmente, al seleccionar uno de estos temas, en el ejemplo “Añadir Frases”, aparece la pantalla en la que se detalla su funcionamiento.

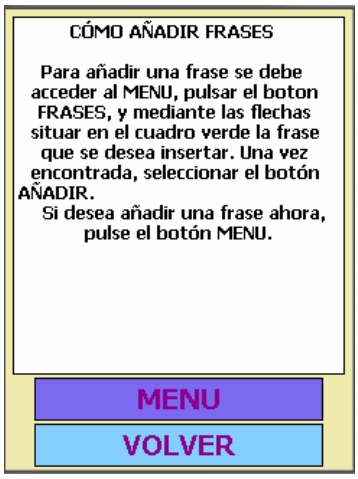


Figura 33. Pantalla de explicación Ayuda Añadir Frases

Todas las pantallas disponen del Botón Volver para poder acceder a la pantalla anterior.

13.10 Configuraciones tipo Imágenes

Este tipo de configuraciones se caracterizan porque los botones son representados mediante imágenes.

14 Botones Imágenes

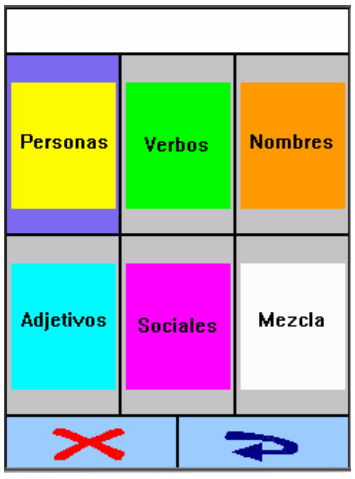


Figura 34. Pantalla Principal Imágenes.

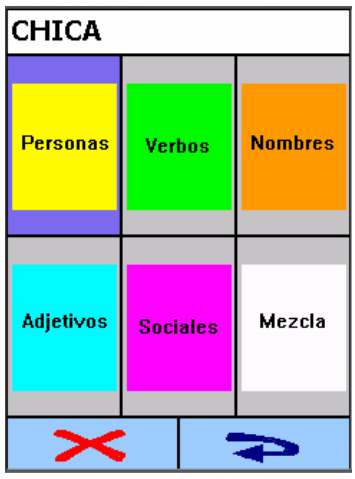


Figura 35 Pantalla Principal de Imágenes con texto escrito

Figura 36. Pantalla con imágenes de una de las categorías (Personas)

La Pantalla Principal se compone de botones que representan categorías y muestran el color que corresponde a su categoría. Podemos observar dicha pantalla en la Figura 34.
Este tipo de botones contienen un conjunto de imágenes que llevan asociadas expresiones descritas por el dibujo y que poseen el mismo color que la categoría a la que pertenecen.

El mecanismo es similar al de las configuraciones de tipo Caracteres con botones que representan categorías, apartado 13.3.
En la Pantalla Principal se incluyen las categorías “Personas”, “Verbos”, “Nombres”, “Adjetivos”, “Sociales” y “Mezcla”, de tal forma que al ser pulsados sus correspondientes botones, se mostrará una nueva pantalla en la que aparecerán las imágenes que pertenecen a la categoría del botón pulsado.
Estas pantallas secundarias pueden contener distintas formas y número de botones, de manera que a diferente configuración tendremos distintos números y distribuciones de los botones. Este aspecto queda determinado en un número de filas y de columnas (fila x columna).
Generalmente, las categorías contienen mayor cantidad de imágenes que de botones, por lo que las imágenes se muestran de forma rotatoria por los botones.

El procedimiento se detalla con un ejemplo en el que la configuración es de dos filas y dos columnas para las pantallas de segundo nivel:
Si se seleccionara el botón de la Figura 34 que contiene la categoría “Personas”, aparecería la pantalla de la Figura35, en la que se mostrarían las cuatro primeras imágenes de esta categoría, distribuidos en los cuatro primeros botones. De esta manera, si la imagen que deseamos insertar es la de una chica, debemos esperar a que el foco recorra todos los botones y continúe mostrando nuevos cuartetos de imágenes hasta que visualicemos el que contiene esta imagen, Figura 35. Una vez estemos ante esta imagen, pulsamos cuando el foco se sitúe sobre ella. A continuación, aparecería la Pantalla Principal con la expresión CHICA insertada en el cuadro de texto, Figura 36.

- **Botones Borrar Imagen y Menú Imágenes**

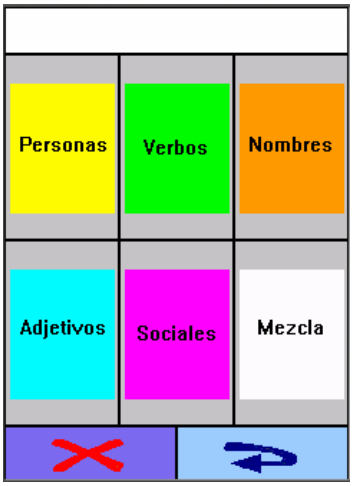


Figura 37. Botón Borrar Imagen

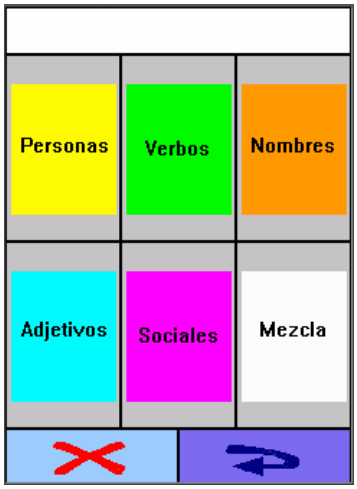


Figura 38. Botón Menú Imágenes

Estos dos botones se incluyen en la Pantalla Principal de todas las configuraciones de tipo Imágenes.

El **Botón Borrar Imagen**, Figura 37, tiene como funcionalidad borrar la última expresión asociada a una imagen introducida en el texto.

El **Botón Menú Imágenes**, Figura 38, contiene el menú principal de la aplicación. Al ser seleccionado este botón, se desplegará otra pantalla en la que se mostrarán nuevas funcionalidades representadas por botones con imágenes también.

- **Botones Anterior Imagen y Volver Pantalla Principal**

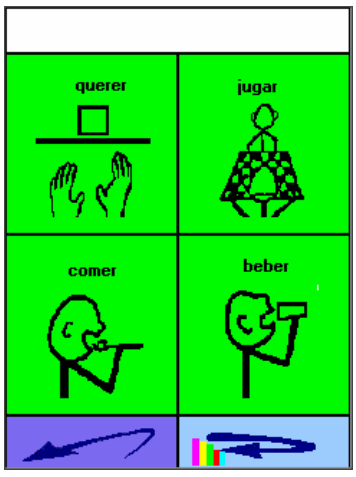


Figura 39. Botón Anterior Imagen

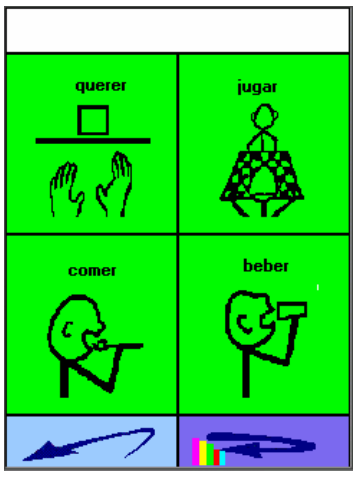


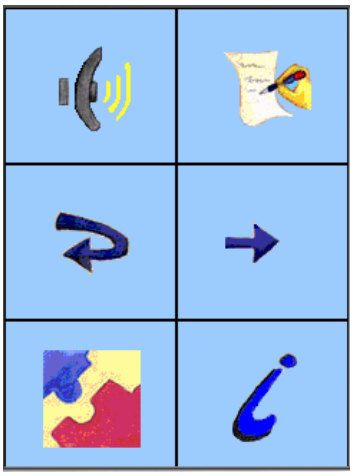
Figura 40. Botón Volver Pantalla Principal

Estos dos botones se incluyen en todas las pantallas que se despliegan al pulsar los botones de la Pantalla Principal.

El **Botón Anterior Imagen**, Figura 39, retrocede y muestra el grupo de imágenes que se ha mostrado anteriormente, puesto que al existir mayor número de imágenes a mostrar que de botones, las imágenes se van rotando en grupos por los botones.
(Procedimiento explicado en el apartado 13.3: Botones de Categorías de caracteres)

El **Botón Volver Pantalla Principal**, Figura 40, despliega la Pantalla Principal.

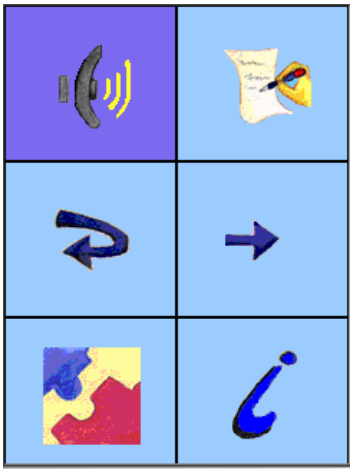
➢ Pantalla MENU Imágenes



Esta pantalla contiene las mismas opciones que el Menú de las configuraciones de tipo Caracteres, a excepción de:
Botón Colores, ya que los botones se representan mediante imágenes cuyo color es representativo, por lo tanto, es necesario que no se modifique.
Botón Frases, puesto que es una funcionalidad de nivel más avanzado que este tipo de metodologías con imágenes.

Figura 41. Pantalla MENU Imágenes

➢ PLAY Imágenes (Reproducir)



Este botón se representa mediante la imagen de un altavoz, Figura 42.
Su funcionalidad es la misma que la del Botón Play de la Pantalla Menú con caracteres.

Figura 42. Botón PLAY

- **EDITAR Imágenes**

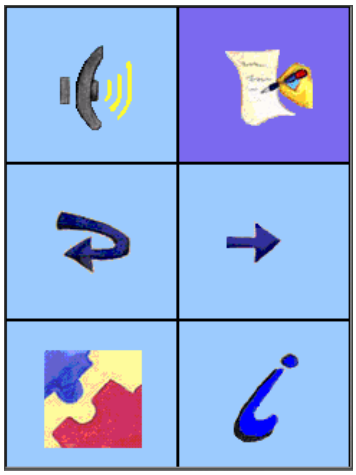


Figura 43. Botón Editar Imágenes

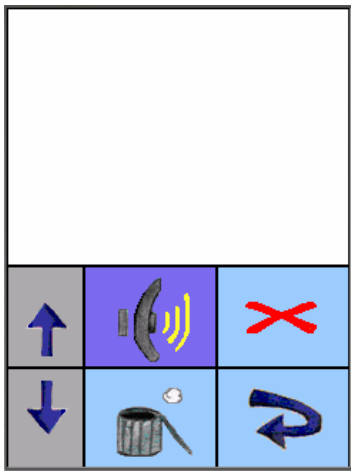
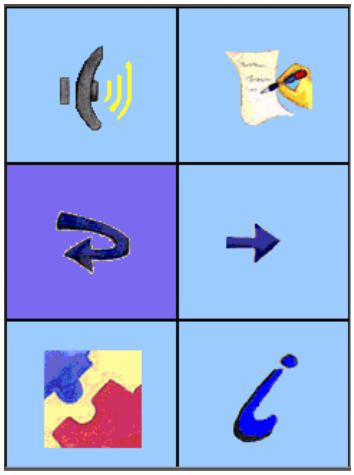


Figura 44. Pantalla Editar Imágenes

Este botón se representa mediante la imagen de papel y lápiz, Figura 43, y se encarga de mostrar la Pantalla Editar Imágenes (Figura 44).

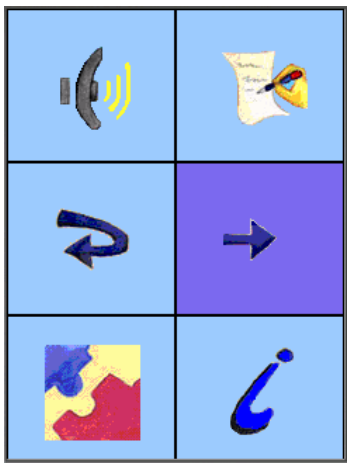
- **Volver Imágenes**



El Botón Volver se representa mediante una flecha ovalada y muestra la Pantalla Principal de la configuración actual.

Figura 45. Botón Volver Imágenes

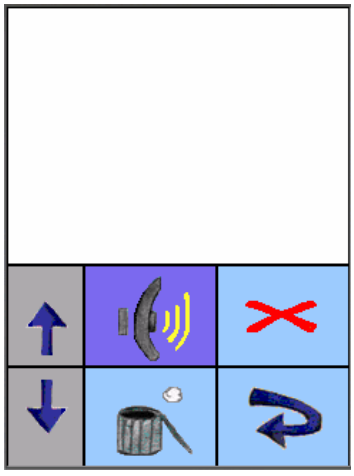
➢ Salir Imágenes



El Botón Salir se representa mediante una flecha recta con dirección hacia la derecha y finaliza la ejecución del programa de tal forma que la próxima vez que se arranque, la configuración mostrada será la actual antes de salir.

Figura 46. Botón Salir Imágenes

➢ Pantalla Editar



Esta pantalla posee las mismas funcionalidades que la Pantalla Editar de las configuraciones de Caracteres, excepto en lugar de tener la opción Borrar Palabra contiene Borrar Imagen.

Figura 47. Pantalla Editar Imágenes

- **Botones y sus funciones:**
  - **Botones** de desplazamiento **^** y **V**: representados mediante las imágenes de una flecha hacia arriba y otra hacia abajo. Permiten ver todo el texto escrito hasta el momento y moverte por él de línea en línea. Estarán activos en caso de que no haya espacio suficiente en el cuadro de texto para abordar todo el texto escrito. En caso contrario, el foco no pasará por ellos además de adoptar un fondo de imagen gris.

- **Botón PLAY:** representado mediante un altavoz. Reproduce en sonido todo el texto escrito hasta el momento. Este botón también se encuentra en la Pantalla Menú.

- **Botón BORRAR IMAGEN:** eliminará del texto escrito la expresión asociada a la última imagen que fue introducida. Este botón también se encuentra en la Pantalla Principal.

- **Botón BORRAR TODO:** representado por la imagen de una papelera. Eliminará todo el texto introducido y el cuadro de texto se quedará vacío para poder empezar a escribir de nuevo.

- **Botón VOLVER:** representado por una imagen de una flecha ovalada. Muestra la pantalla anterior que sería la de MENU que es desde donde se accede a la Pantalla Editar Imagen.

➢ Pantalla Configuración

Misma pantalla que en las configuraciones de tipo Caracteres, véase apartado 13.8.

➢ Pantalla Ayuda

Mismas pantallas que en las configuraciones de tipo Caracteres, véase apartado 13.9.

Parte V. Conclusiones

Capitulo 14. CONCLUSIONES.

14.1 Objetivos alcanzados.

Con este proyecto se pretendía realizar un comunicador para PDA, que sirviera a las personas con parálisis cerebral y en general, a las personas que tengan grandes dificultades físicas y sensoriales de comunicación con las personas de su entorno.

Otro de los objetivos que se quería conseguir, es que permitiese la comunicación de personas que no hubiesen desarrollado la lectura/escritura mediante un método sencillo y práctico pero que realizara una correcta comunicación. La utilización de imágenes pertenecientes al sistema de comunicación SPC con la que los alumnos del Centro Oregón ya conocían ha hecho mas fácil el logro de este objetivo, aportando a su vez un método de comunicación mas divertido sobre todo si el usuario es un niño.

Gracias al sintetizador de voz reutilizado, el texto que se escriba en la PDA podrá oírse en voz para una comunicación más natural.

Por otra parte se pretendía que la aplicación pudiese adaptarse a cada usuario de modo que fuera posible realizar cambios en las configuraciones, mediante la utilización de un fichero XML que recoge todo el contenido de las pantallas que conforman la aplicación es posible introducir modificaciones respecto a una serie de propiedades de los elemento que aparecen en estas pantallas como los botones.

Tenemos que agradecer al Centro Oregon y en especial a su responsable Víctor con el que hemos trabajado mas cerca por ayúdanos a descubrir los problemas de un colectivo con muchas necesidades y que pasa desapercibido en nuestra sociedad

14.2 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 lo 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 los posibles usuarios y atendiendo los requisitos que nos solicitaban.

Capitulo 15. POSIBLES MEJORAS.

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

- 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.
- Mejora del sintetizador de voz.
- En la pantalla editar que puedas moverte a cualquier carácter del texto escrito tal como se realiza en los móviles.
- Crear una aplicación que cree los ficheros XML según las características que se introduzcan.
- Creación de una base de datos con las palabras más utilizadas en el castellano para que el usuario pudiese comunicarse a través de palabras ya escritas.

APENDICES

APENDICE 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

- Basil, C., Ruíz R.: “Sistemas de Comunicación no vocal”. Madrid (1985).

- Basil, C., Soro-Camats,E., Rosell,C.: “Sistemas de signos y ayudas técnicas para la comunicación y la escritura. Principios teóricos y aplicaciones. Barcelona (1998). Ed. Masson.

Fuentes Web

- www.microsoft.com: página oficial de Microsoft 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.msdn.microsoft/library./pa: es la parte de la página Web de Microsoft en español dedicada a los desarrolladores de aplicaciones para sus sistemas operativos. Con ejemplos, consejos de expertos profesionales.

- 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.

- www.tecnoneet.org: como la dirección anterior contiene información sobre el impacto tecnológico en personas con problemas.

- www.medline.plus.org: lugar Web sobre la parálisis cerebral
- www.pdaexpertos.com: lugar Web con información sobre PDA.
- www.tecnologiaedu.com: contiene información sobre las nuevas tecnologías para alumnos con deficiencias.
- www.fortunecity.es: contiene información sobre le enfermedad de la parálisis cerebral.

APÉNDICE B. CONTENIDOS DEL CD-ROM.

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

- Directorio Instalación
- *Directorio Memoria*: Contiene la memoria en formato PDF
- *Directorio Manual de Usuario*: Contiene el manual de usuario en formato PDF.