Herramientas y Lenguajes

Se decidió iniciar por determinar las mejores prácticas de codificación y la herramienta que nos ayudaría a escribir ese código, es decir el Entorno de Desarrollo Integrado (IDE). Los IDE existentes para desarrollo de aplicaciones en dispositivos móviles, en este caso con sistema operativo Android, fueron los siguientes:

a) Eclipse SE con Android SDK Plug-in

b) NetBeans con Android SDK

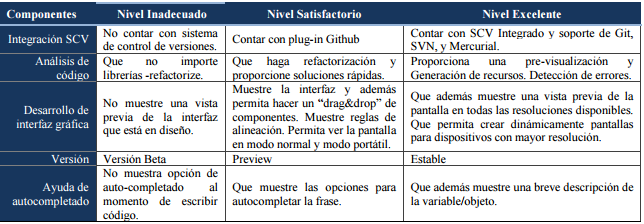
c) IntelliJ IDEA con Android Plugin

d) Google Android Studio con IntelliJ IDEA

f) .NET con Xamarin Studio

g) .NET con Visual Studio

A estos a se les aplicó los siguientes criterios de evaluación



**a) Eclipse:**

La definición que da el proyecto Eclipse “http://es.wikipedia.org/wiki/Fundaci%C3%B3n\_Eclipse” acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular". Es un [programa informático](http://es.wikipedia.org/wiki/Programa_inform%C3%A1tico) compuesto por un conjunto de herramientas de programación de [código abierto](http://es.wikipedia.org/wiki/C%C3%B3digo_abierto) multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar [entornos de desarrollo integrados](http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado) (del inglés IDE), como el IDE de [Java](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java) llamado Java Development Toolkit(JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

El [entorno de desarrollo integrado](http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado) (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. En cuanto a las aplicaciones clientes, Eclipse provee al programador con [frameworks](http://es.wikipedia.org/wiki/Framework) muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc.

El **SDK** de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. Mediante diversos plugins estas herramientas están también disponibles para otros lenguajes como C/C++ (Eclipse CDT) y en la medida de lo posible para lenguajes de script no tipados como PHP o Javascript. El IDE también hace uso de un espacio de trabajo, en este caso un grupo de [metadatos](http://es.wikipedia.org/wiki/Metadatos) en un espacio para archivos planos, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.

http://www.ecured.cu/index.php/Eclipse,\_entorno\_de\_desarrollo\_integrado

**Características de Eclipse**

**Perspectivas, editores y vistas**: en Eclipse el concepto de trabajo está basado en las perspectivas, que no es otra cosa que una preconfiguración de ventanas y editores, relacionadas entre sí, y que permiten trabajar en un determinado entorno de trabajo de forma óptima.

**Gestión de proyectos**: el desarrollo sobre Eclipse se basa en los proyectos, que son el conjunto de recursos relacionados entre sí, como puede ser el código fuente, documentación, ficheros configuración, árbol de directorios. El IDE proporcionará asistentes y ayudas para la creación de proyectos.

**Depurador de código**: se incluye un potente depurador, de uso fácil e intuitivo, y que visualmente ayuda a mejorar el código. Para ello sólo se debe ejecutar el programa en modo depuración .Tenemos una perspectiva específica para la depuración de código, la **perspectiva depuración**, donde se muestra de forma ordenada toda la información necesaria para realizar dicha tarea.

**Extensa colección de *plug-ins***: están disponibles en una gran cantidad, unos publicados por Eclipse, otros por terceros. Al haber sido un estándar de facto durante tanto tiempo (no el único estándar, pero sí uno de ellos), la colección disponible es muy grande. Los hay gratuitos, de pago, bajo distintas licencias, pero casi para cualquier cosa que imaginemos tenemos el *plug-in* adecuado.

**Ventajas en la utilización de Eclipse**

1- El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la [Plataforma](http://www.ecured.cu/index.php/Plataforma) de [Cliente](http://www.ecured.cu/index.php/Cliente) rico, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no.

2- Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente a permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/[C++](http://www.ecured.cu/index.php/C%2B%2B) y [Python](http://www.ecured.cu/index.php/Python), permite a Eclipse trabajar con lenguajes para procesado de texto como LaTeX, aplicaciones en red como Telnet y Sistema de gestión de base de datos.

3-La arquitectura plug-in permite escribir cualquier extensión deseada en el ambiente, como sería Gestión de la configuración. Se provee soporte para Java y CVS en el SDK de Eclipse. Y no tiene porqué ser usado únicamente para soportar otros [Lenguajes de programación](http://www.ecured.cu/index.php/Lenguajes_de_programaci%C3%B3n).

4- El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código.

5- El IDE también hace uso de un espacio de trabajo, en este caso un grupo de metadata en un espacio para archivos plano, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.

**Desventajas en la utilización de Eclipse:**

Su principalmente inconveniente, es el consumo de recursos del sistema. Y también que carece de mucho soporte para webapps (.war, jsp y servlets).

**b)**

**MICROSOFT .NET http://es.wikipedia.org/wiki/Microsoft\_.NET**

Es un [framework](http://es.wikipedia.org/wiki/Framework) de [Microsoft](http://es.wikipedia.org/wiki/Microsoft) que hace un énfasis en la transparencia de [redes](http://es.wikipedia.org/wiki/Red_de_computadoras), con independencia de plataforma de [hardware](http://es.wikipedia.org/wiki/Hardware) y que permita un rápido desarrollo de [aplicaciones](http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_inform%C3%A1tica).

Los principales componentes del marco de trabajo son:

* El conjunto de [lenguajes de programación](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n).
* La biblioteca de clases base o BCL.
* El entorno común de ejecución para lenguajes, o CLR por sus siglas en inglés.

Debido a la publicación de la norma para la infraestructura común de lenguajes (CLI por sus siglas en inglés), el desarrollo de lenguajes se facilita, por lo que el marco de trabajo .NET soporta ya más de 20 lenguajes de programación y es posible desarrollar cualquiera de los tipos de aplicaciones soportados en la plataforma con cualquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje.

Algunos de los lenguajes desarrollados para el marco de trabajo .NET son: [C#](http://es.wikipedia.org/wiki/C_Sharp), [Visual Basic .NET](http://es.wikipedia.org/wiki/Visual_Basic_.NET), [Delphi](http://es.wikipedia.org/wiki/Delphi) ([Object Pascal](http://es.wikipedia.org/wiki/Object_Pascal)), [C++](http://es.wikipedia.org/wiki/C%2B%2B), [F#](http://es.wikipedia.org/wiki/F_Sharp), [J#](http://es.wikipedia.org/wiki/J_Sharp), [Perl](http://es.wikipedia.org/wiki/Perl), [Python](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Python), [Fortran](http://es.wikipedia.org/wiki/Fortran), [Prolog](http://es.wikipedia.org/wiki/Prolog) (existen al menos dos implementaciones, el [P#](http://es.wikipedia.org/wiki/P_Sharp)[1](http://es.wikipedia.org/wiki/Microsoft_.NET#cite_note-1) y el [Prolog.NET](http://es.wikipedia.org/w/index.php?title=Prolog.NET&action=edit&redlink=1)[2](http://es.wikipedia.org/wiki/Microsoft_.NET#cite_note-2) ), [Cobol](http://es.wikipedia.org/wiki/Cobol) y [PowerBuilder](http://es.wikipedia.org/wiki/PowerBuilder)

## Common Language Runtime CLR

El CLR es el verdadero núcleo del framework de .NET, entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios del sistema operativo. Permite integrar proyectos en distintos lenguajes soportados por la plataforma .Net, como C++, Visual Basic, C#, entre otros.

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .NET en un código intermedio, el [CIL](http://es.wikipedia.org/wiki/Common_Intermediate_Language)(Common Intermediate Language) antes conocido como [MSIL](http://es.wikipedia.org/wiki/MSIL) (Microsoft Intermediate Language). Para generarlo, el compilador se basa en la especificación [CLS](http://es.wikipedia.org/w/index.php?title=CLS&action=edit&redlink=1) (Common Language Specification) que determina las reglas necesarias para crear el código MSIL compatible con el CLR.

Para ejecutarse se necesita un segundo paso, un compilador [JIT](http://es.wikipedia.org/wiki/Compilaci%C3%B3n_en_tiempo_de_ejecuci%C3%B3n) (Just-In-Time) es el que genera el [código máquina](http://es.wikipedia.org/wiki/C%C3%B3digo_m%C3%A1quina) real que se ejecuta en la plataforma del cliente. De esta forma se consigue con .NET independencia de la plataforma de hardware. La compilación JIT la realiza el CLR a medida que el programa invoca métodos. El código ejecutable obtenido se almacena en la memoria caché del ordenador, siendo recompilado de nuevo sólo en el caso de producirse algún cambio en el código fuente.



Estructura interna del entorno de ejecución en lenguaje común.

**f) .NET con Xamarin Studio**

Xamarin es un CrossPlatform que permite escribir aplicaciones móviles tanto para Android como para iOS y Windows Phone, consiste en una implementación libre de la plataforma de desarrollo .NET para dispositivos Android, iOS y GNU/Linux. Es decir, con Xamarin se [evita tener que utilizar Java](http://www.elandroidelibre.com/2012/05/es-java-en-android-tan-importante-como-parece.html) para desarrollar una aplicación para Android. Xamarin permitirá generar aplicación para iOS (.APP) y para Android (.APK), la cual correrá de forma nativa. Gracias a esto, surge una de las grandes ventajas de Xamarin: la reutilización de código. Al poder desarrollar aplicaciones nativas para Windows Phone, Windows 8 y web en C# y .NET, resulta que la reutilización del código se exporta a aún más plataformas que iOS o Android. Según Xamarin REFERENCIA, se puede reutilizar el 90% del código aproximadamente. Otra de las ventajas de la plataforma es que dispone de tantas librerías de terceros como el código nativo:hay una gran comunidad detrás. Pero uno de los detalles más importantes es que Xamarin nos proporciona acceso total a la API estándar de Android.

**http://www.researchgate.net/profile/Alejandro\_Tamayo/publication/257365979\_Desarrollando\_aplicaciones\_nativas\_para\_Android\_con\_CSharp/links/0c9605251a25dbeb9d000000.pdf**

**g) .NET con Visual Studio**

**Lenguaje de Programación C sharp:**