**<MEDISYS>**

**ESTUDIO Y ANÁLISIS DE LAS PLATAFORMAS, AMBIENTES Y FRAMEWORKS DE DESARROLLO**

**Versión <1.0>**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 20/04/2012 | 1.0 | Estudio y análisis de las plataformas, ambientes y frameworks de desarrollo | Andrea Quisobone |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**ESTUDIO Y ANÁLISIS DE LAS PLATAFORMAS, AMBIENTES Y FRAMEWORKS DE DESARROLLO**

1. **DESCRIPCIÓN DEL CONTEXTO**

El doctor Eisner Ivan Osorio ha decidido contratarnos para la realización de un software que le permita la sistematización y automatización de procesos propios de su consultorio clínico de medicina interna.

En este momento los procesos se realizan de forma manual, agenda, citas, consulta médica, facturación particular, no se llevan estadísticas y no se recepcionan reportes de exámenes y estos procesos son muy importantes para el cliente.

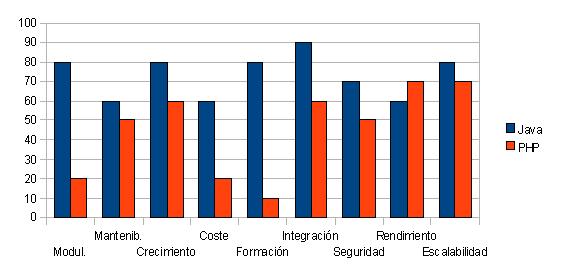
Adicional a esto en este momento se evidencian reprocesos en la generación de formulas medicas, de formulaciones pos y poss debido a los procesos manuales de estos.

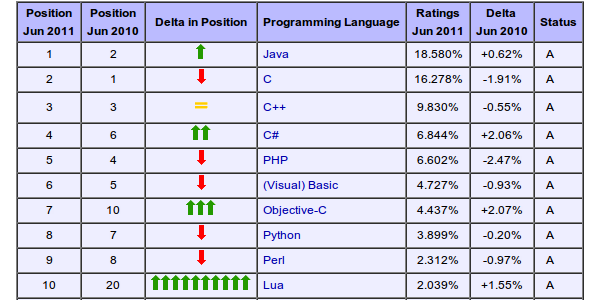
Hemos evidenciado que en este momento el consultorio debe realizar un informe a las EPS proceso esencial para la facturación se hace a través de los de RIPS. Por ahora se realizan de manera manual y luego se pasan a un medio magnético de acuerdo a la EPS es por esto que se genera la necesidad de Medisys software que solucionara estos inconvenientes actuales.

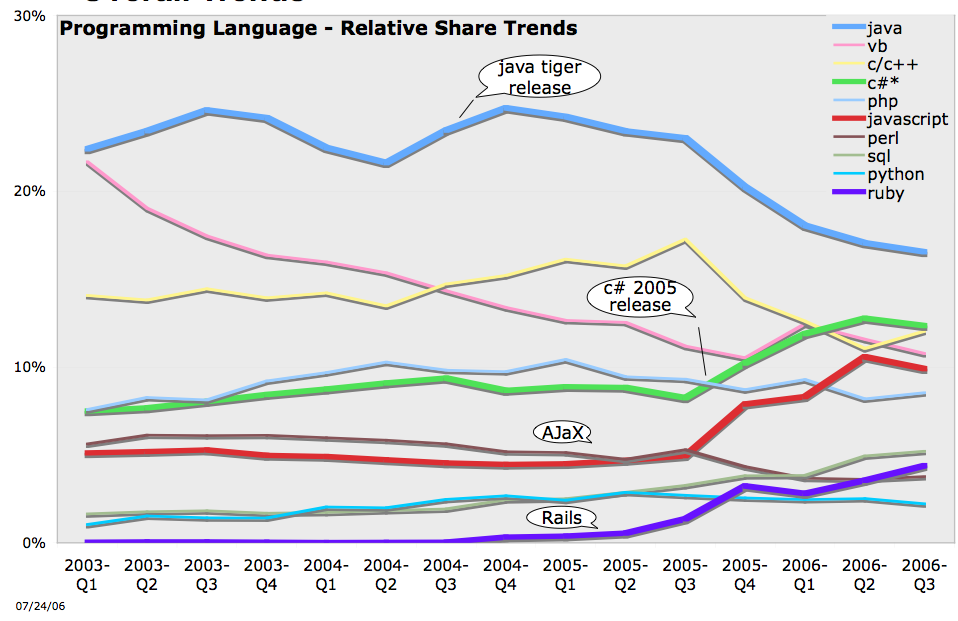
Para nuestro equipo de desarrollo es muy importante realizar un software medico con un entorno amigable por esto se eligieron las siguientes herramientas, basados en sus características y en la experiencia en adelante se describirán las más importantes.

Claro antes se incluirá unas tablas comparativas que apoyaron la decisión

De trabajar con java por ser el más usado y por su popularidad lo cual consideramos es beneficioso en caso de resolución de problemas. Más colaboradores en caso de ser necesario en la red.

**

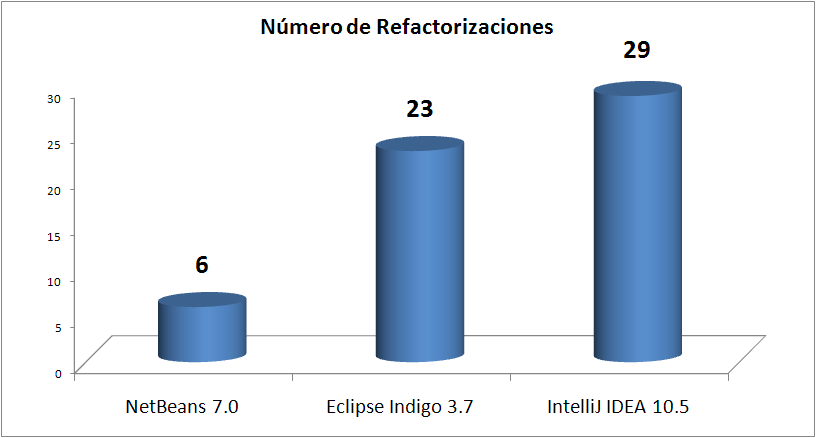


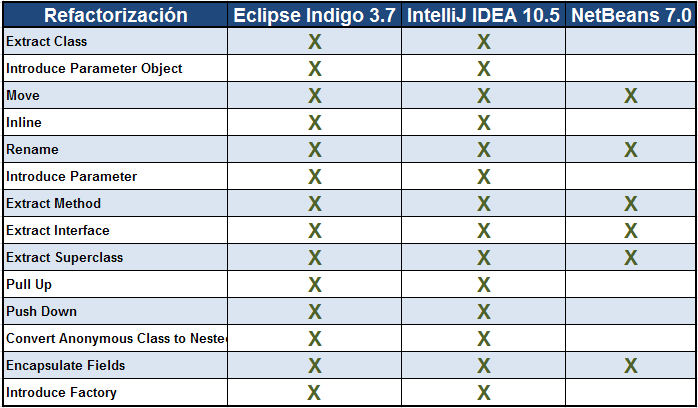


* *ECLIPSE*
* *JUNIT*
* *MYSQLWORKBENCH*
* *JPA*

*Para elegir eclipse se tuvo en cuenta la experiencia y además las siguientes tablas*

*El porcentaje de uso es adecuado para el equipo de desarrollo*

**

**

 **IDE Eclipse:**

Es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar "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 (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (usado también para desarrollar el mismo Eclipse.

Se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los plugins adecuados. La arquitectura de plugins de Eclipse permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, etc.

El desarrollo del IDE Eclipse como de algunos de los plugins mas importantes (como el JDT, plugin para el lenguaje Java, o el CDT, plugin para el lenguaje C/C++).Este proyecto también alcanza a las librerías que sirven como base para la construcción del IDE Eclipse (pero se utilizan de forma completamente independiente), como por ejemplo, la librería de widgets SWT.

Eclipse dispone de un [Editor de texto](http://es.wikipedia.org/wiki/Editor_de_texto) con resaltado de sintaxis. La compilación es en tiempo real. Tiene pruebas unitarias con [JUnit](http://es.wikipedia.org/wiki/JUnit), control de versiones con [CVS](http://es.wikipedia.org/wiki/CVS), integración con Ant, asistentes (*wizards*) para creación de proyectos, clases, tests, etc., y [refactorización](http://es.wikipedia.org/wiki/Refactorizaci%C3%B3n).

Así mismo, a través de "plugins" libremente disponibles es posible añadir control de versiones con subversión e integración con [Hibernate](http://es.wikipedia.org/wiki/Hibernate" \o "Hibernate).

**FRAMEWORK SWT**

SWT es un framework para crear interfaces gráficas en Java (como Swing o Awt) que crea a través de JNI (Java Native Interface) interfaces gráficas nativas del Sistema Operativo en donde ejecutemos nuestra aplicación SWT. Esto quiere decir que con el mismo código visualizaremos en cada Sistema Operativo nuestras ventanas como si hubieran sido creadas para ese SO en específico.

SWT soporta varias plataformas, como por ejemplo Windows, Linux y Mac OS X. El objetivo del diseño de SWT es permanecer cerca al sistema operativo, por lo que la API de SWT (Application Programming Interface) está muy cerca de la API nativa.

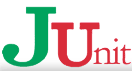
**Ventajas:**

* Ya que crea nativamente la GUI dependiendo del SO, es más rápido que Swing
* Consume menos recursos
* La interfaz gráfica se verá como las demás aplicaciones de tu SO
* Está en constante desarrollo

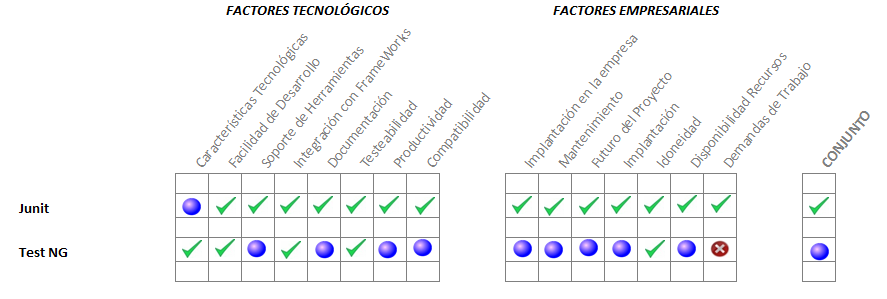
**Desventajas:**

* Muy escasa documentación
* Ya que SWT fue creado por la Fundación Eclipse y pensada para el desarrollo de Eclipse IDE, hay controles o widgets que no consideraron ya que Eclipse no los ocupa, aunque sigue en desarrollo.
* Se tiene que agregar la biblioteca al proyecto a desarrollar, no viene por default en las librerías de Java

El entorno de desarrollo Eclipse, incluyendo sus plugins, está desarrollado por completo en el lenguaje Java. Un problema habitual en herramientas Java (como NetBeans) es que son demasiado “pesadas”. Es decir, necesitan una máquina muy potente para poder ejecutarse de forma satisfactoria. En gran medida, estas necesidades Vienen determinadas por el uso del API Swing para su interfaz gráfico. Swing es una librería de widgets portable a cualquier plataforma que disponga de una máquina virtual Java pero a costa de no aprovechar las capacidades nativas del sistema donde se ejecuta, lo cual supone una ejecución sensiblemente más lenta que la de las aplicaciones nativas.SWT es una librería de widgets equivalente a Swing en la cual, se aprovechan los widgets nativos del sistema sobre el que se ejecuta. El hecho de aprovechar los widgets nativos, permite que la ejecución de interfaces de usuario sea mucho más rápida y fluida que si se utilizase Swing y, además, siempre dispone del “Look and Feel” del sistema.



La siguiente tabla describe algunas características de JUNIT divididos en factores tecnológicos y empresariales que nos apoyan nuestra decisión de trabajar con esta herramienta como por ejemplo facilidad de desarrollo ,soporte de herramientas, etc.

****

Es un conjunto de herramientas para la realización de pruebas de software JUnit es un conjunto de clases ([*framework*](http://es.wikipedia.org/wiki/Framework)) que permite realizar la ejecución de clases Java de manera controlada, para evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

JUnit es también un medio de controlar las [pruebas de regresión](http://es.wikipedia.org/wiki/Pruebas_de_regresi%C3%B3n), necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación.

El propio *framework* incluye formas de ver los resultados (*runners*) que pueden ser en modo texto, gráfico ([AWT](http://es.wikipedia.org/wiki/AWT) o [Swing](http://es.wikipedia.org/wiki/Swing_%28biblioteca_gr%C3%A1fica%29)) o como tarea en [Ant](http://es.wikipedia.org/wiki/Apache_Ant).

JUnit es un framework para realizar y automatizar pruebas de aplicaciones Java. Es decir, JUnit se sitúa en la fase de pruebas dentro del ciclo de Ingeniería del Software.

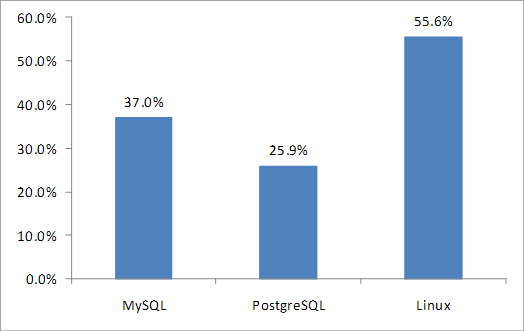
Al igual que la gran mayoría de las tecnologías del mundo Java, JUnit evoluciona para aportar nuevas mejoras y hacernos la vida más fácil a los programadores, diseñadores de pruebas, etc.

JUnit incorpora las siguientes ventajas y modificaciones respecto a versiones anteriores

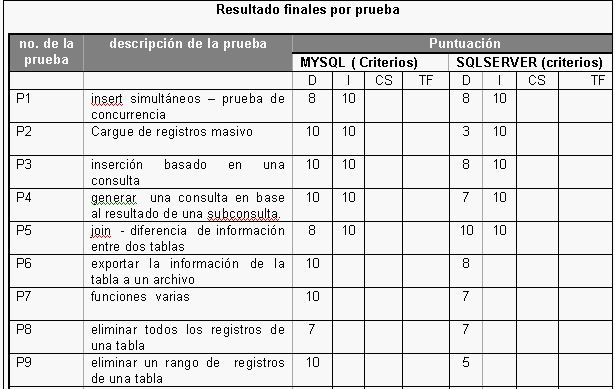
* Las clases que contienen los métodos que representan las pruebas ya no tienen que ser subclases de junit.framework.TestCase
* Los métodos que representan las pruebas a realizar ya no tienen que tener el prefijo "test" en su nombre sino que se indican con la anotación @Test
* Sustitución del método 'setUp' por la anotación o anotaciones @Before.
* Sustitución del método 'tearDown' por la anotación o anotaciones @After.
* Se pueden realizar pruebas en donde el tiempo de ejecución es crítico. De manera que una prueba falle si tarda más de X milisegundos en ser ejecutada.
* Se pueden realizar pruebas en donde se debe controlar que una excepción es lanzada.
* Se pueden desactivar pruebas, a través de la anotación @Ignore.

** MySQLWorkbench**

El porcentaje de uso de MySQL aunque no es el primero es suficiente para nosotros en vista de que es libre y cumple con los requerimientos de una base de datos adecuada para nuestra aplicación

****

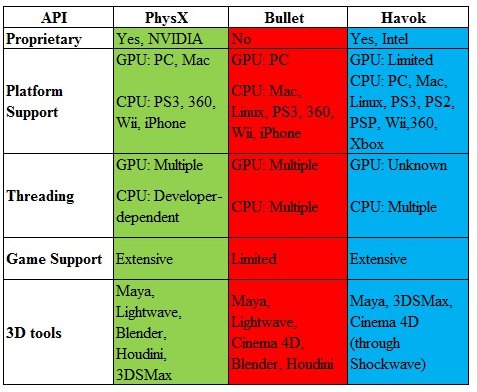
Además y como podemos apreciar en la siguiente tabla las puntuaciones en carga de registros, inserción basado en consultas, exportar la información y otras es la conveniente para el programa.



MySQL Workbench permite a un DBA, desarrollador o arquitecto de datos para diseñar visualmente, modelo, generar y gestionar bases de datos. Incluye todo lo que un modelador de datos necesarios para la creación de complejos modelos ER, hacia adelante y la ingeniería inversa, y también ofrece funciones clave para llevar a cabo la gestión del cambio difícil y tareas de documentación que normalmente requieren mucho tiempo y esfuerzo.  En esta aplicación elabora una representación visual de las tablas, vistas, procedimientos almacenados y claves foráneas de la base de datos. Además, es capaz de sincronizar el modelo en desarrollo con la base de datos real, ingeniería inversa para importar el esquema de una base de datos ya existente el cual haya sido guardado o hecho copia de seguridad con MySql Administrador.

MySQLWorkbench también puede generar el código  para crear la base de datos que se ha dibujado en el esquema; es compatible con los modelos de base de datos de DBDesigner 4 y soporta las novedades incorporadas en MySQL 5.

**JPA (Java Persistence API)**

****

Una interfaz de programación representa la capacidad de comunicación entre componentes de software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla. De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio. Las APIs asimismo son abstractas: el software que proporciona una cierta API generalmente es llamado la implementación de esa API.

La JPA fue originada a partir del trabajo del JSR 220 ExpertGroup. Ha sido incluida en el estándar [EJB](http://es.wikipedia.org/wiki/EJB)3.

Persistencia en este contexto cubre tres áreas:

* La API en sí misma, definida en javax.persistence.package
* La Java PersistenceQueryLanguage (JPQL)
* Metadatos objeto/relacional

El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos (siguiendo el patrón de [mapeo objeto-relacional](http://es.wikipedia.org/wiki/Mapeo_objeto-relacional)), como sí pasaba con EJB2, y permitir usar objetos regulares (conocidos como [POJOs](http://es.wikipedia.org/wiki/POJO)).

Analizando la naturaleza del aplicativo desarrollado, podemos evidenciar que una de las características más importantes es su optimo desempeño, es por ello que tomamos la decisión de no utilizar este framework para la conexión de la base de datos y la aplicación, debido a la gran demanda de recursos hardware que implica usar esta herramienta en conjunto de un servidor como JBosso GlashFish. En reemplazo a este framework utilizaremos una estrategia de conexión más sencilla y menos exigente computacionalmente hablando.