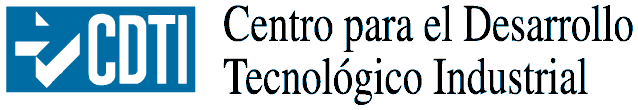


|  |
| --- |
|  |



E4.7 Formación en las herramientas de desarrollo del cliente

SIGAIM

Sistema de Generación Automática de Informes Médicos

Indice

[1.Introducción 2](#__RefHeading__559_502504632)

[2.Entorno de desarrollo 2](#__RefHeading__561_502504632)

[3.Framework de interfaz gráfica 2](#__RefHeading__577_502504632)

[4.Bibliografía 2](#__RefHeading__579_502504632)

1. Introducción

Para llevar a cabo el cliente de escritorio del sistema, ha sido necesario el análisis de diferentes herramientas, tanto del entorno de desarrollo como de los distintos frameworks de desarrollo para escritorio. Dadas las restricciones estipuladas previamente de compatiblidad y lenguaje Java.

1. Entorno de desarrollo

Los principales entornos de desarrollo considerados son Eclipse y NetBeans.

El entorno utilizado será Eclipse, ya que se trata de un entorno de desarrollo maduro y de amplia aceptación. Además dada la preferencia en el proyecto por el software libre, encaja mejor que la alternativa NetBeans, que podría presentar problemas legales en cuanto a redistribución.

Analizando ambos, encontramos que Eclipse es un entorno más flexible, por ejemplo, soportando proyectos en distintos lenguajes de programación, cuando NetBeans está restringido sólo a Java, C++ y recientemente a desarrollo web (css, php y html5). Ésto puede suponer un punto a favor en el proyecto, pudiendo utilizar la misma herramienta si en un futuro es necesario implementar algún paquete en otro lenguaje y facilitando la integración en el entorno de desarrollo de otros paquetes del proyecto SIGAIM que esperan utilizar otras herramientas y frameworks ajenos a Java.

Por otra parte, el editor de interfaces visual de eclipse genera código Java puro, frente al editor de NetBeans que se sostiene en ficheros XML de formato propio. Por lo que se genera una dependencia innecesaria de la herramienta.

1,1 Eclipse Swing Window Builder

En este sentido, se añadirá a Eclipse la extensión *Swing Window Builder*, siendo la principal opción y promovida por la propia fundación Eclipse. Se trata de un proyecto originado en Google que ha sido liberado posteriormente.

Ésta herramienta tiene la capacidad de analizar código Java ya programado e inferir de él el diseño visual que genera, pudiendo editarlo fácilmente y de forma gráfica. Como se indica más arriba, para esta labor se sostiene tan solo en el código fuente Java, por lo que se puede trabajar conjuntamente utilizando la herramienta visual o editando el código fuente correspondiente según la complejidad de la labor a realizar o la preferencia personal del desarrollador.

En cuanto al editor visual, presenta de una forma sencilla todas las capacidades inherentes del framework Swing, pudiendo añadir mediante extensiones de eclipse otros elementos que pudieran utilizarse. En cuanto al modo de diseño, es similar a la herramienta integrada en NetBeans, permitiendo arrastrar y colocar los distintos elementos y mostrando una vista en árbol de su posición lógica, así como una opción de vista previa sin compilación útil para comprobar cómo se comportan los elementos con distintos tamaños de marco y si se posicionan tal como indica el editor visual.

1. Framework de interfaz gráfica

Las principales herramientas para el desarrollo de la interfaz gráfica analizadas son: SWT, Swing, Qt y GTK+.

La herramienta escogida en este caso es Swing, ya que ofrece una opción integrada en el Toolkit de Java, multiplataforma y que mejor encaja en los objetivos de este proyecto. La alternativa nativa de Java SWT se presenta como un predecesor de desarrollo abandonado. Si bien ambos tienen soporte nativo en Java y herramientas de diseño visual tanto para Eclipse como para NetBeans, es lógico decantarse por la opción más actual y en actual mantenimiento, no suponiendo ningún problema en cuanto a compatibilidad con los equipos objetivo.

Por otro lado, se descartan frameworks como Qt y Gtk+ (U otros similares), por no ser nativos de Java requieren la redistribución de su código con la aplicación además de poder presentar problemas de compatibilidad con ciertas plataformas, como es el caso de Gtk que puede necesitar de software adicional y presenta un *look and feel* propio que podría desorientar o no convencer a los usuarios.

Si bien con otros frameworks podría lograrse una apariencia más integrada con el sistema operativo del usuario o una estética más pulida, por norma general implicaría problemas en cuanto a la adaptación a diversas plataformas y sistemas operativos.

1. Bibliografía

1. Apache UIMA. [Online] http://uima.apache.org/.