Emilio Silva - Nicolas Stucki

**HelpDesk**

Diseño de la implementación: Con el fin de implementar los requerimientos funcionales del programa en cuestión de la forma más eficiente posible, se efectuó un análisis que dio como resultado una estructura jerárquica basada en una clase principal y dos subordinadas correspondientes a cliente y empleado, y finalmente la clase Ticket que depende de todas las anteriores. No se encontró necesidad de implementar una clase para el usuario administrador, pues este requisito puede ser cubierto fácilmente por la interfaz gráfica y algunos requerimientos funcionales implementados como métodos de la clase principal. Para almacenar los dos tipos de usuario, se optó por implementar listas simplemente encadenadas, porque permiten agregar fácilmente usuarios, pueden ser reordenadas en cualquier momento, y esto de manera eficiente. De igual forma se implementó las tablas de hashing para almacenar tickets y poder obtenerlos de la forma más rápida posible, con una complejidad que no excede el O (1). Por otro lado, se implementaron las secuencias ordenadas de tickets y de incidentes gracias a árboles AVL, que permiten asimismo una gran accesibilidad.

Esquema abstracto de la solución:

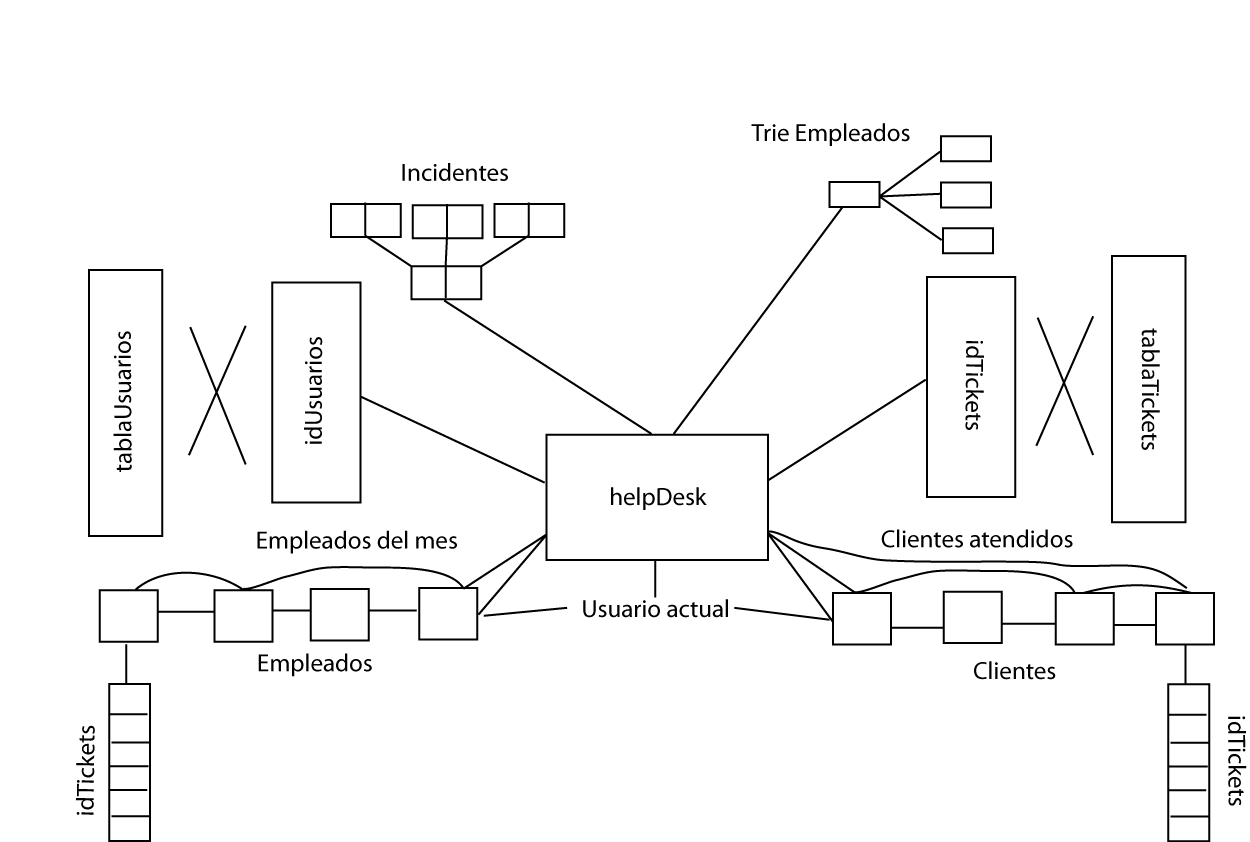
HelpDesk

Empleado

Cliente

Ticket

Diagrama de estructuras utilizadas



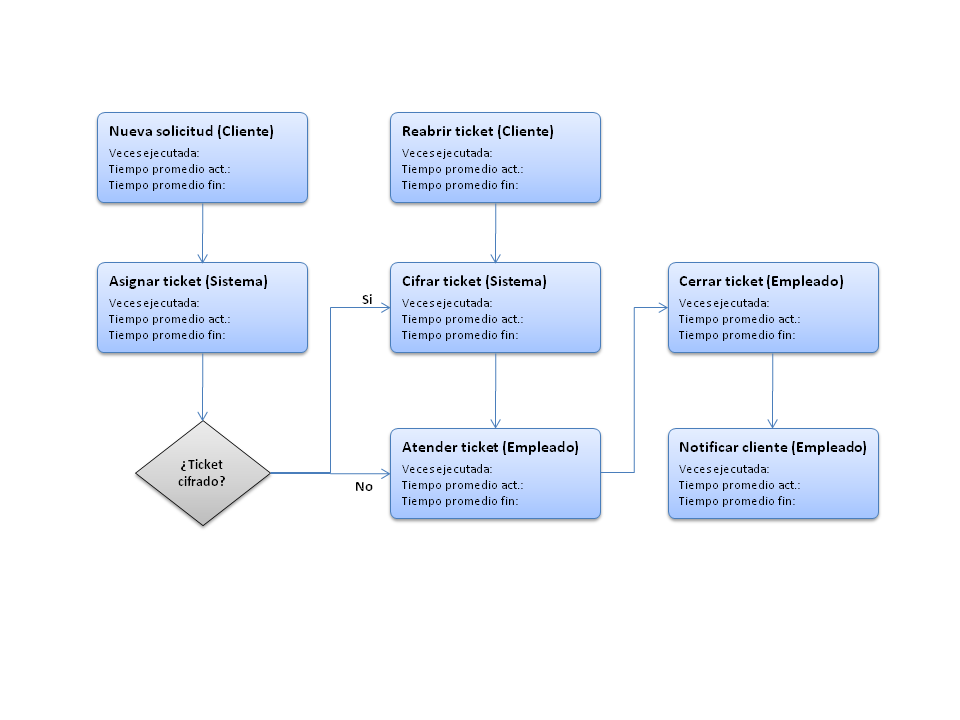
Invariante

1. Los clientes deben estar ordenados por fecha de ingreso.

2.La lista de IDs de tickets para los usuarios debe estar ordenada ascendentemente.

3.El índice de tickets e incidentes están ordenados por fecha de creación.

GRAFO



Este grafo se constituye como una estructura adicional al resto, que pretende mostrar los distintos procesos y sus estadísticas en cuestiones de tiempos y de número de veces que han sido consultados.

**N18:** Implementación de interfaces Servlet para el servicio web.

Es importante resaltar la importancia capital que ha adquirido la implementación de servlets para el uso integrado de éstos junto con interfaces web tanto estáticas como dinámicas. Ocho son los puntos a resaltar acerca de esta innovadora tecnología que puede ser representada de forma dinámica gracias al uso de los mencionados Servlets.

En nuestro documento nos enfrentamos al reto de crear formularios HTML con varios botones, cada uno de los cuales enviaba información al mismo servlet. Esto causa el problema de saber cuál botón fue oprimido por el Servlet. Pues bien, la solución a lo anterior consistió en lo siguiente:

Se efectuó varias comprobaciones de excepciones, las cuales estaban específicamente diseñadas con el fin único de evaluar los patrones de reconocimiento del formulario, es decir, por medio de excepciones controladas se pudo dominar todas las posibles combinaciones de casos.

Se hizo asimismo una constatación poco afortunada, y en efecto parece que los scripts embebidos en las páginas HTML no funcionan como deberían. Esto constituye una gran limitación del JBoss que es completamente irremplazable.

A continuación se muestra un ejemplo de cómo usar el WEB-INF:

**<web-app>**

**<servlet>**

**<servlet-name>**

**Simple**

**</servlet-name>**

**<servlet-class>**

**uniandes.cupi2.SimpleServlet**

**</servlet-class>**

**</servlet>**

**</web-app>**

**N18: Singletons**

*“Hay una infinidad de mundos posibles, es decir colección de compatibles y, entre tantos, necesariamente uno debe contener una materia más rica y mejor organizada.”*

Gottfried von Leibniz, Teodicea

Las sabias palabras de Leibniz nos dicen que *hay una infinidad de mundos posibles*, pero que no todos estos mundos son necesariamente deseables, dado que en realidad existe uno sólo que contiene *una materia más rica y mejor organizada*. A pesar de que Leibniz no sabía programar en Java, su genio mucho más avanzado para su época se pone de manifiesto en que esta frase extraída de su gran tratado teológico muestra lo inconveniente que resulta la posibilidad de que existan muchos mundos constantemente, y que por consecuente uno y sólo un mundo contiene una materia más rica que en este caso son los datos introducidos anteriormente por el usuario.

A la luz de la informática, Leibniz puede ser considerado como el primero en sugerir la necesidad de implementar singletons, que justamente son mecanismos que permiten garantizar la existencia de mundos únicos. Y ¿cuál es el mundo único en cuestión? Pues el mejor de todos los mundos posibles, el mundo que representa la interfaz del usuario.