Instituto Superior Politécnico “José Antonio Echeverría”



Centro de Estudios de Ingeniería y Sistemas

**Trabajo de curso**

**Sistema informático para el control de acceso al CEIS**

**Informe**

Autores: Camilo José del Real Martell

Tutor: Ing. Fernando Díaz Santoyo

Asignaturas: Diseño y Programación Orientada a Objetos

Introducción a la Gestión de Software

Año académico: 1ro.

Carrera: Ingeniería Informática

Resumen

*Sistema informático para el control de acceso al CEIS* es una aplicación creada para controlar, como su nombre indica, la entrada de personal a los locales del Centro de Estudios de Ingeniería y Sistemas (CEIS), orientando su utilización a aquellas personas encargadas de validar las entradas o solicitudes de llaves de los locales, gestión que también posibilita. La aplicación cuenta con una interfaz amigable y estándar, sobre la cual es posible ejercer todo el trabajo de gestión con un buen desenvolvimiento por parte del ínter-actor, además de que lleva a cabo la organización de los datos bajo una buena estructura organizativa, al posibilitar una rápida y eficaz manipulación de la información.

Summary

Computer system for controlling access to the CEIS is an application created to control, as its name suggests, the staff entrance to the premises of the Centre for Engineering and Systems (CEIS), directing its use to those responsible for validating entries or requests for keys to the premises, which also allows management. The application has friendly interface and a standard on which it is possible to perform all the work of management with a good performance by the inter-actor, in addition to carrying out the organization of data in a good organizational structure, the enable quick and efficient handling of information.

Índice

[Introducción 5](#_Toc296044084)

[Capítulo 1. Descripción del negocio 6](#_Toc296044085)

[Capítulo 2. Solución propuesta 8](#_Toc296044086)

[2.1 Distribución lógica de paquetes 11](#_Toc296044087)

[2.2 Patrones de diseño implementados en la aplicación 13](#_Toc296044088)

[2.3 Manejo de errores 13](#_Toc296044089)

[2.4 Sobre la interfaz y codificación 15](#_Toc296044090)

[Capítulo 3. Tras los pasos de la ingeniería de software 16](#_Toc296044091)

[3.1 Resúmenes de tiempo y modelos de pruebas. Parte 1. 16](#_Toc296044092)

[3.2 Resúmenes de tiempo y modelos de pruebas. Parte 2. 18](#_Toc296044093)

[3.3 Resúmenes de tiempo y modelos de pruebas. Parte 3. 20](#_Toc296044094)

[3.4 Comparación de tiempos 21](#_Toc296044095)

[Capítulo 4. Sobre la aplicación 22](#_Toc296044096)

[4.1 Ventajas 22](#_Toc296044097)

[4.2 Desventajas 23](#_Toc296044098)

[Conclusiones 24](#_Toc296044099)

[Recomendaciones 25](#_Toc296044100)

Introducción

El desarrollo de *Sistema informático para el control de acceso al CEIS* no es más que la propuesta que los autores acordaron proponer como proyecto de curso. Este proyecto de curso tiene como objetivo automatizar la gestión de control del personal autorizado: docentes, alumnos ayudantes, personal técnico-administrativo y como extra, los visitantes; el sistema creado para la gestión incluye el control de acceso a los locales y las solicitudes de sus llaves, para ambos la generación de trazas; así como un conjunto de reportes o consultas para llevar a cabo búsquedas específicas sobre los datos almacenados.

Para el desarrollo de la aplicación se usó como herramienta de desarrollo el Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) Eclipse en su versión 3.2.0 empleando el lenguaje de programación Java y haciendo uso de los plug-in´s Eclipse Modeling Framework Project (EMF) versión 2.2.0, Graphical Editing Framework (GEF) versión 3.2 y Visual Editor Project (VE) versión 1.2.0.

En cuanto al modelado del sistema, se emplearon como herramientas el software de modelado, documentación y diseño de Diagramas UML Enterprise Architect, y Process Dash Board para la creación de las tablas de cálculo de tiempo, cumplimentando los objetivos propuestos para la culminación de la asignatura Introducción a la Gestión de Software en las orientaciones dadas para la realización de este informe. Se hizo uso también de la biblioteca JUnit en su versión 3.8.1 para la ejecución de pruebas de unidad en el IDE Eclipse.

Capítulo 1. Descripción del negocio

Durante el proceso de confección de las tareas a realizar para trazar una correcta gestión del desarrollo del proyecto, los autores se remitieron al Centro de Estudios de Ingeniería y Sistemas (CEIS) para obtener información acerca del negocio que se deseaba automatizar, obteniendo los datos siguientes.

1. Sobre los visitantes

Son todas aquellas personas que solicitan acceso al CEIS, pero que no están registrados en este como trabajador o estudiante. Su posibilidad de entrar está dada ante la obligatoriedad de llenar el formulario de entrada, que recoge: nombre y apellidos, número de identidad, el local al que desea dirigirse y la fecha y hora de entrada, así como que pueda acceder al local que desea, determinado por las reglas de acceso de este.

1. Sobre los trabajadores, ya sean docentes o técnico-administrativos

Personal trabajador que debe ser registrado desde su ubicación laboral en el centro. La información almacenada referente a este personal, es mucho más compleja, se registran nombres y apellidos, número de identidad, dirección, número de plaza y de expediente, salario, fecha de contrato, cargo, fecha de otorgamiento del cargo, y una serie de códigos sobre el cargo ocupacional, el grupo escala (GE), la categoría ocupacional (CO), su ubicación para la defensa (UD), el nivel escolar (NE). Estos trabajadores pueden tener acceso a los locales donde trabajan u otros, pero todo está precedido de las reglas de acceso; es el mismo caso para las solicitudes de las llaves

1. Sobre los alumnos ayudantes

Son un tipo especial de trabajadores del CEIS, pues no son trabajadores como tal. Su enfoque práctico está orientado a un grupo de desarrollo de proyectos, por este motivo es un tipo de personal que también está registrado en el centro, cuyos datos a recoger son los nombres y apellidos, su número de identificación, el grupo de desarrollo en el que se encuentra y el tutor de quien recibirá asesoría y orientaciones en sus trabajos; además, se le asigna un número identificativo

Para la gestión del personal registrado y no registrado que es capaz de acceder al CEIS; se ha de analizar también como se lleva el control de los locales y las llaves, determinado por el nivel de restricciones dados a través de las reglas de acceso o de solicitudes de llaves. ¿Qué información se obtuvo durante la investigación? es presentada a continuación.

1. Sobre los locales y llaves

Son entidades concretas donde de los locales es sabido el nombre y el personal que radica en él, respecto a la llave, esta solamente “conocerá” el local que es capaz de abrir. Respecto a la llave, únicamente está disponible una, las demás copias son para casos especiales de pérdida o de necesidad urgente.

1. Sobre las reglas

Son un objeto abstracto que determina la posibilidad de acceder a un local o solicitar una llave, teniendo en cuenta la persona, el día y los horarios. Las reglas no son absolutas, pueden cambiar con el tiempo.

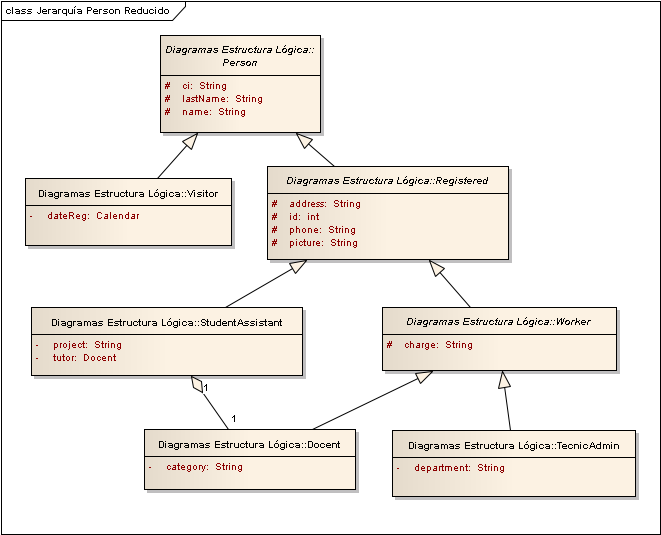
Capítulo 2. Solución propuesta

*Como nota aclarativa para este, capítulo se ha de indicar que los diagramas presentados no son copia exacta del original pues se han obviado los métodos de clase para poder presentar una mejor visualización. Para presenciar los diagramas a tamaño completo, por favor diríjase a la carpeta de anexos.*

Como parte de la modelación lógica a la gestión del personal, se decidió tomar los datos más relevantes y que fueran, o que pudiesen ser notables en el tipo de aplicación que se estaba desarrollando, concluyendo en los siguientes datos.

|  |  |  |
| --- | --- | --- |
| **Técnico-Administrativo** | **Docente** | **Alumno Ayudante** |
| Nombre y apellidos  Número de identificación  Dirección  Teléfono  ID  Foto  Cargo  Departamento en que radica | Nombre y apellidos  Número de identificación  Dirección  Teléfono  ID  Foto  Cargo  Categoría | Nombre y apellidos  Número de identificación  Dirección  Teléfono  ID  Foto  Grupo de desarrollo  Tutor |
| **Visitante** | | |
| Nombre y apellidos  Número de identificación  Fecha de ingreso a los registros (que día se registro) | | |

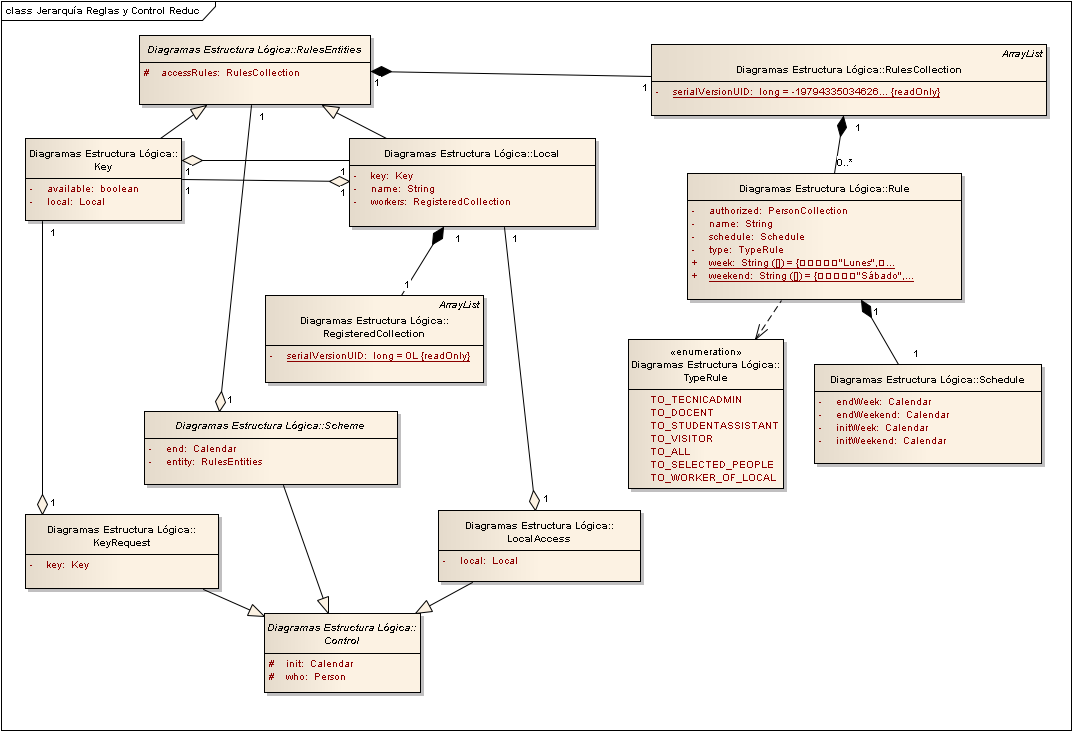
Con esta información fue posible llegar a la confección del siguiente diagrama UML de clases.



Respecto a los locales y sus llaves, el control de acceso y solicitudes de llaves, así como las trazas, se decidió modelar estos objetivos de la manera siguiente:

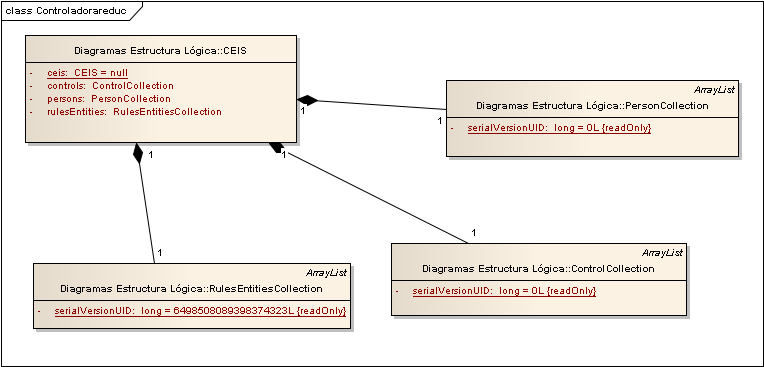
|  |  |  |  |
| --- | --- | --- | --- |
| **Local** | | **Llave** | |
| Reglas de acceso que tiene asociado  Nombre  Trabajadores que radican en él  Llave de apertura | | Reglas de solicitudes que tiene asociada  Local que abre  Disponibilidad de la llave | |
| **Accesos a locales** | **Solicitudes de llaves** | | **Trazas** |
| Fecha y hora  Persona que accedió  Local accedido | Fecha y hora  Persona que tomó la llave  Llave entregada | | Fecha y hora inicial  Persona que realizó la petición  ¿Qué fue? Local o llave  Fecha y hora final |

Con esta distribución de información se confeccionó el siguiente diagrama UML de clases.



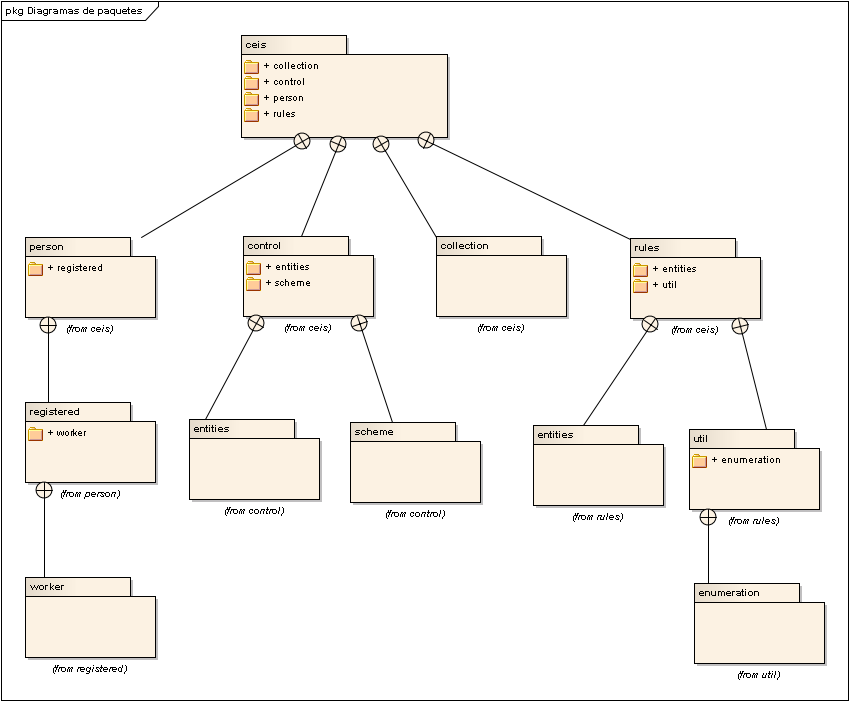
En la estructura del diagrama se evidencia una entidad denominada Rule (Regla: mediante la cual se modela el objetivo de reglas de acceso y solicitudes de llaves) cuyos campos son: nombre, a qué tipo de persona estará dirigida (Visitantes, Docentes, Técnicos-Administrativos, Alumnos Ayudantes, Todo el Personal, o un Personal Seleccionado), horario (ya puede ser un rango entre semana, otro rango en fin de semana o ambos) y por último un listado de personal autorizado en caso de ser elegida la opción de Personal Seleccionado en la opción del tipo de persona a la que estará dirigida.

Una vez definida la estructura lógica en las gestiones del personal y del control del centro respecto a locales y llaves, se ha de identificar la entidad controladora encargada de englobar los listados de todo el personal, de los locales y sus llaves, así como de las entidades controladoras de los accesos, solicitudes de llaves y trazas.



2.1 Distribución lógica de paquetes

La estructura lógica modelada se encuentra organizada en paquetes para establecer una determinada organización jerárquica; esta puede ser apreciada en el diagrama siguiente.



El contenido de los paquetes se describe de la forma siguiente.

* ceis: paquete superior, contendrá la clase controladora de la estructura lógica (CEIS) y anidará los demás paquetes.
* person: es la base de la jerarquía que se modeló para la gestión del personal, contendrá las clases más generales (Person, Visitor y Registered).
* registered: una generalización superior del personal (StudentAssistant, Worker).
* worker: contiene las dos clases identificativas del personal trabajador del centro (Docent, TecnicAdmin).
* control: comienza la jerarquía referente al control de accesos y solicitudes de llaves, tiene en su interior a la clase general Control.
* entities (de control): contiene las entidades que permiten el registro del control de acceso y de las solicitudes de llaves (LocalAccess, KeyRequest).
* scheme: contiene la clase Scheme que compondrá el registro de trazas.
* collection: almacenará las clases de colecciones (PersonCollection, RulesEntitiesCollection, RegisteredCollection, ControlCollection), necesarias para la clase controladora fundamentalmente.
* rules: base de la jerarquía que contempla las entidades concretas del CEIS (Local, Llave).
* entities (de rules): contiene las clases que modelan las entidades concretas del centro (Local, Key) y la clase base de estas.
* util: empleado para almacenar la clase Schedule con la cual se modeló los rangos de horarios, necesarios en la modelación de la entidad Rule.
* enumeration: guarda TypeRule, clase de tipo **enum** empleada para los valores del campo *type* de la clase Rule que describe al tipo de personal o a qué personal se le aplicará la regla.

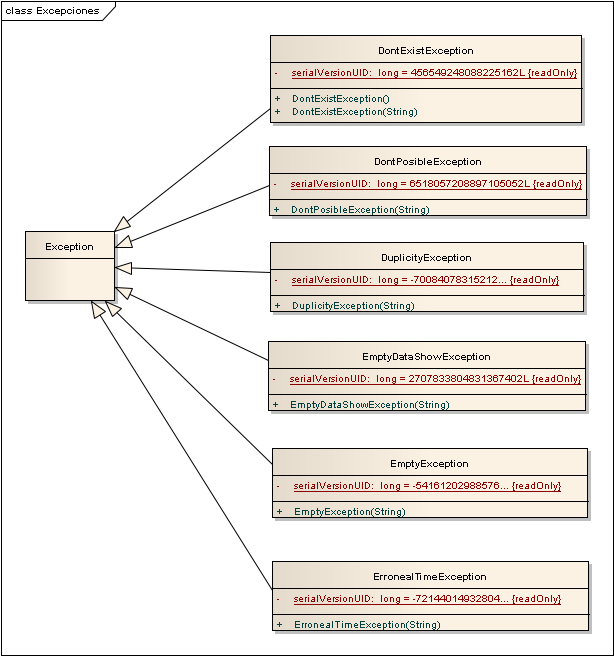
2.2 Patrones de diseño implementados en la aplicación

Patrón Singleton: empleado en la clase controladora CEIS persiguiendo el objetivo del patrón, asegurar la creación de una sola instancia de la clase determinada. (Para más información sobre el patrón de diseño Singleton diríjase a la documentación anexada)

2.3 Manejo de errores

Para el manejo de errores, la aplicación fue diseñada para validar todos los campos que el usuario introdujese con un formato erróneo, empleando para ello un sistema de alerta con ventanas pop-up´s generadas mediante el manejo de excepciones.

El siguiente diagrama muestra que clases de excepciones fueron definidas para los diversos contextos en los que se pueden detectar errores.



Descripción

* DontExistException: es enviada cuando se quiere buscar información que no existe en los registros, por ejemplo, si se desea registrar a un estudiante ayudante es necesario obtener un listado de todos los docentes para indicar cuál será su tutor, la excepción se lanzaría en caso de no existir ningún docente registrado.
* DontPosibleException: se lanza cuando se requiere ejecutar una acción que no está disponible, por ejemplo, si se desea extraer una llave, mas esta está ya en posesión de otra persona, se envía un DontPosibleException.
* DuplicityException: se genera cuando una validación detecta la duplicidad de información clave, que no puede ser repetida por su importancia identificativa, por ejemplo el número de identificación de las personas.
* EmptyDataShowException: es enviada para evidenciar que no hay información para mostrar cuando se esté tratando de acceder a determinados datos
* EmptyException: se lanza cuando hay campos de formularios vacíos y que necesitan ser llenados.
* ErronealTimeException: se lanza especialmente cuando ocurren errores en los rangos de horarios (necesarios para la modelación de la reglas), una hora inicial no debe preceder a la hora final.

2.4 Sobre la interfaz y codificación

En la codificación del proyecto, los desarrolladores emplearon estándares de codificación establecidos para el lenguaje Java, así como también se crearon las interfaces siguiendo detalles comunes de las aplicaciones de gestión de información off-line, donde superan los menús en la parte superior, barras de herramientas y vínculos a los laterales, muy clásico en aplicaciones se sistemas Windows. Casi todos los vínculos de los menús presentan comando aceleradores, atajos de teclado o teclas de acceso rápido; la mayoría de los componentes visuales que presentan la posibilidad de activar una acción, contienen etiquetas flotantes o tooltips como una herramienta de ayuda. Las interfaces están lo más agradable posible ante la uniformidad de colores, y la utilización de pequeñas imágenes para referenciar acciones o reportes ofrecen cierto toque artístico sin hacer necesidad de sobrecargar la aplicaciones de elementos innecesarios.

Capítulo 3. Tras los pasos de la ingeniería de software

Como objetivo del desarrollo de la aplicación se encuentra establecido el uso de herramientas que posibiliten la automatización de los cálculos de tiempo durante las distintas fases del desarrollo. Para el presente trabajo se estableció como pauta la utilización del software Process Dashboard con el objetivo de obtener las tablas resumen del cálculo del tiempo empleado durante las fases de: planeamiento, diseño, codificación, compilación, prueba y el tiempo de una revisión general. Seguidamente se plasman los resúmenes de los tiempos empleados por cada uno de los integrantes del equipo al encargarse de un método en particular para el cumplimiento de este objetivo; junto con la tabla se exponen los modelos de Caso de Prueba de Unidad y de Caso de Prueba funcional.

3.1 Resúmenes de tiempo y modelos de pruebas. Parte 1.

Información referente al desarrollador Camilo José del Real Martell, manejo del método *compareDateByDayOfWeekAndTime(Calendar check, Calendar init, Calendar end,* ***boolean*** *weekend)* creado para comparar objetos Calendar teniendo en cuenta solo el día de la semana y las horas y localizado en la clase RulesCollection.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Project Owner | Camilo José del Real Martell | | | | | | |
| Start Date | jun 05, 2011 8:31:15 AM |  |  |  |  |  |  |
| Completion Date | jun 05, 2011 9:37:28 AM | Completed: \* | | | | | |
| Keywords |  | | | | | | |
| Language | Java | | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time in Phase (min.)** | **Plan** | **Actual** | **To Date** | **To Date %** |
| Planning |  | 21 | 45 | 29.4 |
| Design |  | 3 | 15 | 9.8 |
| Code |  | 30 | 57 | 37.3 |
| Compile |  | 2 | 15 | 9.8 |
| Test |  | 10 | 21 | 13.7 |
| Postmortem |  | 0 | 0 | 0 |
| Total | 90 | 66 | 153 | 100 |

Modelo de caso de prueba de unidad

|  |  |
| --- | --- |
| **Caso de prueba** | Comparar fechas por día de la semana y hora |
| **Clase** | CompareTestCase |
| **Método** | testCompareDateByDayOfWeekAndTime |
| **Desarrollador** | Camilo José del Real Martell |
| **Probador** | Camilo José del Real Martell |
| **Fecha** | 9/6/2011 |
| **Versión** | 1.0 |
| **Juego de datos # 1** | |
| **Valores de entrada** | |
| **Parámetro** | **Valor** |
| Calendar init  Calendar end  Calendar check  Boolean weekend | Día actual menos dos días  Día actual más dos días  Día actual menos tres días  false |
| **Valores de salida** | |
| **Variable de salida** | **Valor esperado** |
| Done | false |

Modelo de caso de prueba funcional

|  |  |
| --- | --- |
| **Caso de Prueba** | Comparar fechas por día de la semana y hora |
| **Caso de Uso** | Acceso al centro |
| **Desarrollador** | Camilo José del Real Martell |
| **Curso a Probar** | Validar día y hora actual a partir de las reglas de acceso |
| **Probador** | Camilo José del Real Martell |
| **Fecha** | 9/6/2011 |
| **Versión** | 1.0 |
| **Objetivo de la Prueba** | Comprobar que las comparaciones de los campos de los objetos tipo Calendar se ejecuten correctamente |
| **Descripción de la Prueba** | Se introducirán los datos necesarios para solicitar un acceso al centro |
| **Condiciones:** |  |
| Para poder acceder, los datos deben estar en un rango de días y horarios de al menos una de las reglas de acceso al local, en caso de no suceder esto no se puede acceder al centro | |
| **Condiciones de Entrada** | **Resultado Esperado** |
| El día de la semana y la hora deben coincidir con al menos una regla del local seleccionado | Que sea posible acceder al local |
| **Observaciones** | |
|  | |

3.2 Resúmenes de tiempo y modelos de pruebas. Parte 2.

Información referente al desarrollador Jasiel Domínguez Jomarrón, manejo del método *find(String ci)* de la clase controladora CEIS; se encarga de buscar a la persona que tenga como número de identidad uno idéntico al parámetro.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Project Owner | Jasiel Domínguez Jomarrón | | | | | | |
| Start Date | may 27, 2011 10:06:08 AM |  |  |  |  |  |  |
| Completion Date | may 27, 2011 10:41:55 AM | Completed: \* | | | | | |
| Keywords |  | | | | | | |
| Language | Java | | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time in Phase (min.)** | **Plan** | **Actual** | **To Date** | **To Date %** |
| Planning |  | 10 | 19 | 32.2 |
| Design |  | 2 | 7 | 11.8 |
| Code |  | 19 | 28 | 47.6 |
| Compile |  | 1 | 2 | 3.4 |
| Test |  | 3 | 3 | 5.0 |
| Postmortem |  | 0 | 0 | 0.0 |
| Total | 35 | 35 | 59 | 100 |

Modelo de caso de prueba de unidad.

|  |  |
| --- | --- |
| **Caso de prueba** | Encontrar persona |
| **Clase** | FindTestCase |
| **Método** | testFind |
| **Desarrollador** | Jasiel Domínguez Jomarrón |
| **Probador** | Jasiel Domínguez Jomarrón |
| **Fecha** | 6/6/2011 |
| **Versión** |  |
| **Juego de datos # 1** | |
| **Valores de entrada** | |
| **Parámetro** | **Valor** |
| String ci | ci = "52020754879" |
| **Valores de salida** | |
| **Variable de salida** | **Valor esperado** |
| Person person | Que person apunte al mismo objeto |

Modelo de caso de prueba funcional.

|  |  |
| --- | --- |
| **Caso de Prueba** | Encontrar Persona |
| **Caso de Uso** | Buscar Información de una persona |
| **Desarrollador** | Jasiel Domínguez Jomarrón |
| **Curso a Probar** | Buscar que la persona existe en los datos de la aplicación |
| **Probador** | Jasiel Domínguez Jomarrón |
| **Fecha** | 7/6/2011 |
| **Versión** | 1.0 |
| **Objetivo de la Prueba** | Identificar errores en las comparaciones de datos para el posible perfeccionamiento del mmétodo |
| **Descripción de la Prueba** | Se obtendrán los datos de una persona y se les buscará en la “base de datos” |
| **Condiciones:** |  |
| La persona debe existir | |
| **Condiciones de Entrada** | **Resultados Esperados** |
| Campo Person no deba ser nulo | La función debe retornar un valor distinto de nulo |
| **Observaciones** | |
|  | |

3.3 Resúmenes de tiempo y modelos de pruebas. Parte 3.

Información referente al desarrollador Yasiel Pérez Díaz, manejo del método addByInsertion(Person person) de la clase PersonCollection cuyo objetivo es ordenar los campos por el número de identidad empleando el algoritmo de ordenamiento por inserción.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Project Owner | Yasiel Pérez Díaz | | | | | | |
| Start Date | jun 05, 2011 5:14:02 PM |  |  |  |  |  |  |
| Completion Date | jun 05, 2011 6:04:57 PM | Completed: \* | | | | | |
| Keywords |  | | | | | | |
| Language | Java | | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time in Phase (min.)** | **Plan** | **Actual** | **To Date** | **To Date %** |
| Planning |  | 21 | 32 | 31.7 |
| Design |  | 12 | 18 | 17.8 |
| Code |  | 19 | 25 | 24.8 |
| Compile |  | 1 | 9 | 8.9 |
| Test |  | 12 | 17 | 16.8 |
| Postmortem |  | 0 | 0 | 0 |
| Total | 50 | 65 | 101 | 100 |

Modelo de caso de prueba de unidad

|  |  |
| --- | --- |
| **Caso de prueba** | Añadir persona a los registros del centro |
| **Clase** | AddTestCase |
| **Método** | testAddByInsertion |
| **Desarrollador** | Yasiel Pérez Díaz |
| **Probador** | Yasiel Pérez Díaz |
| **Fecha** | 6/6/2011 |
| **Versión** | 1.0 |
| **Juego de datos # 1** | |
| **Valores de entrada** | |
| **Parámetro** | **Valor** |
| Person person | person->name = “Camilo”  person->ci = “91100228189” (…) |
| **Valores de salida** | |
| **Variable de salida** | **Valor esperado** |
| Int index | 0, 1, 0, 0 (hay variadas salidas) |

Modelo de caso de prueba funcional

|  |  |
| --- | --- |
| **Caso de Prueba** | Añadir persona a los registros del centro |
| **Caso de Uso** | Nuevo ingreso de docente en el centro |
| **Desarrollador** | Yasiel Pérez Díaz |
| **Curso a Probar** | Almacenamiento de los datos |
| **Probador** | Yasiel Pérez Díaz |
| **Fecha** | 7/6/2011 |
| **Versión** | 1.0 |
| **Objetivo de la Prueba** | Que los datos se guarden ordenados por el número de identificación |
| **Descripción de la Prueba** | Se insertará una nueva persona en el centro para que se guarde en una posición específica según se número de identificación |
| **Condiciones:** |  |
|  | |
| **Condiciones de Entrada** | **Resultados Esperados** |
| Se deben de introducir todos los datos solicitados de la persona y el número de identificación debe estar validado a solo números. | Según el número de carnet, se guardará en una determinada posición, pero sobre todo se debe almacenar |
| **Observaciones** | |
|  | |

3.4 Comparación de tiempos

Los tres métodos divididos para ejecutar el control de tiempos presentaban diferentes niveles de complejidad, por lo que los tiempos tendrán valores un poco aislados. La realización de los métodos en tiempo es significativa pues evidencia cierto dominio de los autores sobre el diseño y desarrollo de aplicaciones, para esto caso, está reducido a métodos, pero aún así puede servir de métodos comparativos no solo del tiempo de desarrollo, sino también los jefes de proyecto pudieran aplicar estos resultados para crear escalas de eficiencias para evaluar a su equipo.

Capítulo 4. Sobre la aplicación

4.1 Ventajas

|  |  |
| --- | --- |
| Uso de una interfaz estándar en aplicaciones de trabajo, que comprende un menú superior y dos paneles deslizantes desde los cuales se puede acceder a las tareas más comunes. |  |
| Uso de pestañas tabuladas para mantener la visualización de diferentes reportas. |  |
| Empleo de un único conjunto de controles para la navegación en los datos de los reportes. |  |
| Empleo de teclas de acceso rápido para usuarios expertos. |  |
| Existencia de un archivo de ayuda de la aplicación para principiantes. |  |

4.2 Desventajas

* Necesidad de instalación de la Máquina Virtual de Java versión 6 actualización 10 ó superior ante al uso del Look and Feel (LAF) Nimbus. En caso de ser la JRE una versión inferior, la aplicación tratará de cargar el LAF desde al archivo adjunto a la aplicación, pero esto podría traer lentitud en la aplicación y que no se carguen todos los componentes visuales como es debido.
* El uso de las clases especiales tipo **enum** requiere una versión de la JRE 1.6 o superior.

Conclusiones

Los objetivos trazados fueron cumplimentados a tiempo, gracias al seguimiento continuo que se realizó al proceso y al empleo de métricas de medición de calidad. Así también, objetivos propios de los autores fueron asistidos, donde el aumento de los conocimientos de diseño y programación se tornaron muy elevados en comparación con el inicio.

El desarrollo de la aplicación demostró varios temas a tener en cuenta para futuros trabajos:

* El empleo de herramientas para el diseño, el modelado y el control del tiempo agiliza el proceso de desarrollo de software.
* El desarrollo de tareas aplicables a la sociedad conlleva a la maduración laboral y toma de responsabilidades.
* El uso de estándares proporciona limpieza, legibilidad y organización.
* Es de suma importancia el trabajo de mesa antes de comenzar el trabajo digital.
* La implantación de reglas de trabajo son muy efectivas, en especial cuando el régimen de estudio es demasiado alto y es necesario aprovechar el tiempo.

Recomendaciones

El presente trabajo, a pesar de haber sido finalizado con todos los objetivos planteados, los autores consideran la posible ampliación del software, agregando funcionalidades tales como: el almacenaje “permanente” en memoria, opciones de configuración, un mejor trato a la memoria del sistema, así como otros aspectos que pudiesen surgir ante una nueva modelación del sistema para actualizaciones o por el hecho de una posibilidad de implantación. Es de considerar que la modelación del negocio ha sido bastante acertada por lo que serían pocos los cambios sobre el sistema.

**Bibliografía**

1. Eckel, Bruce. “Thinking in Java, 3rd. edition, Beta”, Editorial Félix Varela, La Habana 2008.
2. Recopilación Colectivo de Profesores CEIS, “Buenas Prácticas para el Diseño de Interfaz”, La Habana 2002.
3. Recopilación Colectivo de Profesores CEIS, “Principales componentes para el diseño de interfaces”, La Habana 2009.
4. S. Pressman, Rogers. ”Ingeniería del Software: Un enfoque práctico”, Quinta Edición, Editorial Félix Varela, La Habana 2005.
5. Zukowski, Jonh. The Definitive Guide to Java Swing, Third Edition