Universidad Rey Juan Carlos

Escuela Técnica Superior de Ingeniería de Telecomunicación

S I G E L

SISTEMA DE GESTIÓN DE LOCALIDADES

Documentación del diseño

Francisco Canela González
Laura Núñez González
Javier Benito Díaz
Mario Fernández Hevia
David Marín del Pino

Octubre 2010

Índice

1. Introducción

El presente documento tiene como finalidad documentar los conceptos, tecnologías y procedimientos aplicados en el diseño y desarrollo de Sigel. Sigel es un sistema de gestión de localidades de un teatro desarrollado en el marco de la asignatura *Metodología de la Programación* de la Universidad Rey Juan Carlos.

El texto se encuentra dividido en las siguientes partes:

1. Propósito: Breve resumen de los objetivos y la ideología seguida.
2. Herramientas y tecnologías: Sinopsis de los lenguajes, entornos de desarrollo integrado y herramientas usadas.
3. Diseño: Diagramas y conceptos usados luego en el desarrollo y explicación de los elementos internos que componen el software..

[1]Se puede acceder al resumen en http://creativecommons.org/licenses/by/3.0/deed.es y a la versión completa del texto legal en http://creativecommons.org/licenses/by/3.0/legalcode

2. Propósitos y retos planteados

2.1. Software como en la vida real

Para el desarrollo de Sigel se ha tratado, pese a ser una práctica universitaria, de llevar una metodología de trabajo lo más parecida a la aplicada en el entorno laboral.

2.2. Software libre a partir de sofware libre

Los miembros del equipo de desarrollo de la aplicación decidieron desde el primer momento que, dado que se trataba de una aplicación con finalidad educativa, esta iba a ser liberada como software libre.

No obstante, se decidió ir un paso más allá e intentar utilizar únicamente herramientas libre durante todas las fases del desarrollo de la aplicación.

3. Herramientas y tecnologías implicadas

3.1. Git: Sistema de control de versiones

Git es un sistema de control de versiones ideado por Linus Torvalds y que es ampliamente usado en el desarrollo de software libre, como Gnome o el Kernel de Linux. Git permite al equipo de Sigel editar código de manera concurrente, mantener un historial de versiones y volver atrás cuando se comete algún error.

Como repositorio central u origen se ha hecho uso de GitHub[2], que aloja proyectos *open source* de manera gratuita. Es posible acceder al repositorio a través de la siguiente dirección: *https://github.com/linze/Sigel*

3.2. LaTeX: Lenguaje de edición de documentos

LaTeX es un sistema de composición de textos usado ampliamente en el ámbito científico y desarrollo de software profesional. Permite abstraerse del formato y centrarse en el contenido del documento. Al ser un lenguaje de macros multiplataforma es ideal para el desarrollo colaborativo de documentación.

Para la documentación de Sigel se ha hecho un uso exclusivo de LaTeX con la ayuda del editor Kile [3].

3.3. UML y Dia: Desarrollo de diagramas

Se ha tratado de hacer uso de UML, el lenguaje unificado de modelado, en la etapa del diseño. Disponer de diagramas de clases aclarativos capacita al equipo para dividirse el trabajo y funcionar sin conocer los detalles internos de la implementación que está realizando otro miembro del equipo.

El diseño de los diagramas UML se ha realizado con Dia[4].

3.4. FreePascal y Lazarus

3.4.1. ¿Por qué FreePascal?

Para la elección del lenguaje se ha tenido en cuenta la familiaridad de los desarrolladores con la sintaxis, la potencia del lenguaje y la cercanía a los propósitos y retos planteados.

[2] http://www.github.com
[3] http://kile.sourceforge.net/
[4] http://projects.gnome.org/dia/

FreePascal tiene una sintaxis parecida a la de Pascal y la potencia de ser una alternativa a Delphi [5].

3.4.2. Lazarus. Una potente alternativa libre a Delphi

Lazarus implementa las características del entorno de desarrollo integrado de desarrollo rápido de aplicaciones Delphi. Permite realizar con relativa sencillez aplicaciones con interfaz gráfica de usuario.

[5]Puesto número 12 por mayor número de lineas de código escritas por Tiobe.Noviembre 2010. Más detalles en:http://www.tiobe.com/content/paperinfo/tpci/index.html

4. Diseño

4.1. Modulos necesarios

Es necesario el diseño de un módulo para cada tipo de dato que se desea almacenar en la aplicación. Se requiere por tanto un módulo para manejar el estado de las localidades, otro para las reservas y otro para las esperas.

4.2. Descripción de los módulos del programa

4.2.1. TTipoLocalidad

TTipoLocalidad es un tipo enumerado que indica el tipo de localidad del teatro, pudiendo encontrarse entre los siguientes valores:

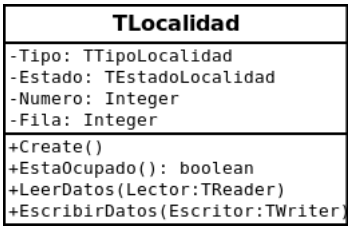
- Patio
- Primera planta
- Palco

4.2.2. TEstadoLocalidad

TEstadoLocalidad es un tipo enumerado que establece el estado de la localidad con respecto a su ocupación. Puede tener uno de estos valores:

- Libre
- Comprada
- Reservada

4.2.3. TLocalidad

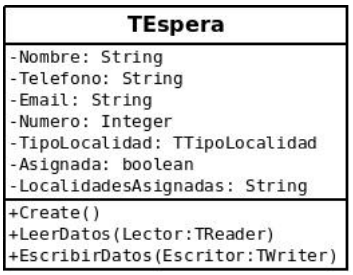


TLocalidad es una de las clases fundamentales de la aplicación. En esta clase se guardan toda la información relacionada con una localidad como su tipo, su estado, el número y la fila.

Las funciones y procedimientos son los siguientes:

- Create(): Constructor de la clase
- EstaOcupado(): Devuelve verdadero si está comprada o reservada y falso si está en otro estado.
- LeerDatos(Lector: TReader): Es un procedimiento que obtiene mediante parámetros un buffer por el cual va a poder leer los datos de una localidad en un fichero.
- EscribirDatos(Escritor: TWriter): Es un procedimiento que obtiene mediante parámetros un buffer por el cual va a poder escribir los datos de una localidad en un fichero.

4.2.4. TEspera



TEspera es la clase que gestiona la lista de espera. Guarda cada uno de los elementos que componen la lista de esperas, que nos servirá para gestionar los datos de las personas que están en ella.

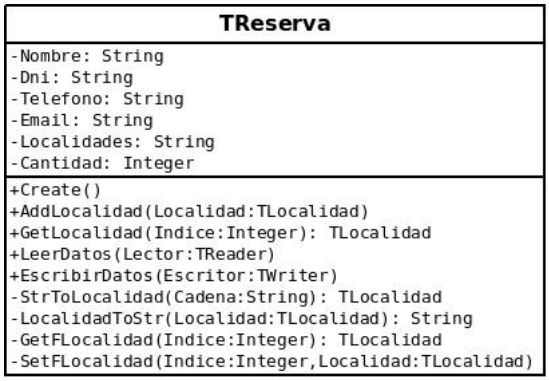
Para ello necesitaremos unos atributos privados que serán: Nombre, Telefono, Email, Numero y Tipo de Localidad. Estos datos son los que el software guarda para identificar a los usuarios, poder contactar con ellos y saber el tipo de localidad deseada. Además guarda si se le ha asignado alguna localidad y cuales.

Las funciones y procedimientos que posee esta estructura de datos son:

- Create() : Es un constructor que crea una nueva espera vacía.
- LeerDatos(Lector: TReader) : Es un procedimiento que obtiene mediante parámetros un buffer por el cual va a poder leer los datos de una espera de un fichero.

- EscribirDatos(Escritor: TWriter): Es un procedimiento que obtiene mediante parámetros un escritor de buffer por el cual va a poder escribir los datos de una espera en un fichero.

4.2.5. TReserva



TReserva es una estrucutra de datos que implementa cada uno de los elementos que componen la lista de reservas. Mediante esta estructura podremos gestionar los datos de las personas que han realizado una reserva.

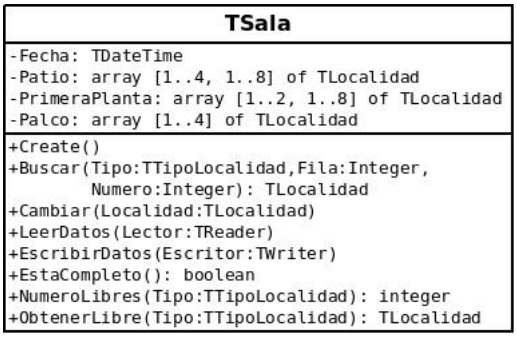
TReserva posee una serie de atributos privados que son: Nombre,DNI, Telefono, Email, Localidades, Cantidad. El nombre y dni nos servirán para identificar a los clientes. El teléfono y el email para contactar con los cilentes si es necesario. El atributo de localidades contiene un string con las localidades y los tipos. Por último cantidad contiene el número de localidades que contiene la reserva.

Esta estructura implementa una serie de funciones y procedimientos para poder gestionar adecuadamente las reservas:

- Create() : Es un constructor que crea una nueva resreva vacía.
- AddLocalidad(Localidad: TLocalidad) : Es un procedimiento que recibe un TLocalidad y lo añade a la reserva.
- GetLocalidad(Indice: Integer):TLocalidad : Se trata de una función que recibe un índice(entre 1 y 4), y busca en el array que contiene las localidades de la reserva y devuelve el TLocalidad que se encuentra en la posición del índice dado.
- LeerDatos(Lector: TReader) : Es un procedimiento que obtiene mediante parámetros un buffer por el cual va a poder leer los datos de una reserva de un fichero.

- EscribirDatos(Escritor: TWriter): Es un procedimiento que obtiene mediante parámetros un buffer por el cual va a poder escribir los datos de una reserva en un fichero.

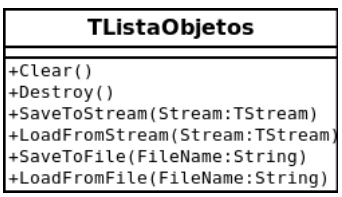
4.2.6. TSala



TSala es una estructura de datos que se encargará de gestionar las posibles operaciones de la sala. Para ello distinguimos los siguientes atributos con los que trabajaremos: Fecha, Patio, PrimeraPlanta y Palco. Nos serviremos de los siguientes métodos:

- Create(): Constructor que inicializará los variables de la sala
- Buscar(Tipo: TTipoLocalidad, Numero: Integer, Fila: Integer): Método que devolverá un tipo TLocalidad que permitirá buscar a través del tipo de localidad, el número y la fila.
- Cambiar(Localidad: TLocalidad): Método que cambiará la localidad
- LeerDatos(Lector: TReader): Es un procedimiento que obtiene mediante parámetros un buffer por el cual va a poder leer los datos de una sala en un fichero.
- EscribirDatos(Escritor: TWriter): Es un procedimiento que obtiene mediante parámetros un buffer por el cual va a poder escribir los datos de una sala en un fichero.

4.2.7. TListaObjetos



Es una clase que hemos creado y que implementa una lista que se puede guardar en ficheros. Es capaz de guardar y cargar ficheros que hereden de TPersistent y contengan los métodos LeerDatos y EscribirDatos.

- Clear(): Limpiará la lista
- Destroy(): Liberará espacio en el disco
- SaveToStream(Stream: TStream): Guardará a Stream el objeto
- LoadFromStream(Stream: TStream) Cargará de Stream el objeto
- SaveToFile(FileName: String): Nos permite guardar en un fichero la lista de objetos
- LoadFromFile(FileName: String): Nos permite cargar de un fichero la lista de objetos

4.2.8. Visión general