UNIVERSIDAD DE BURGOS

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

Proyecto01

Agenda En Un Contexto Persistente

Alumnos	Roberto Miranda y Asier Alonso Daniel Lozano y Jorge Laguna
Tutor	Carlos López Nozal DEPARTAMENTO DE INGENIERÍA CIVIL Área de Lenguajes y Sistemas Informáticos

Burgos, 13 de noviembre de 2014

Índice de contenido

Índice de ilustraciones

Índice de tablas

1 INTRODUCCIÓN

Se va a proceder en la creación de una agenda persistente. En la que se contara con una serie de Contactos, mediante los cuales se podrán almacenar una serie de Llamadas.

Estos Contactos se podrán filtrar a partir de una serie de Tipos de Contacto.

Se realizará un proyecto Java junto a hsqlDB. Mediante el cual se gestionará dicha base de datos.

Se implementara el proyecto de manera Binaria en la que se gestionaran los elementos de la Base de Datos a partir de lectura y escritura de archivos. Por otra parte como anteriormente se ha citado, se gestionara la base de datos median hsqlDB.

2 REQUISITOS

Se quiere implementar una aplicación que gestione los datos de una agenda (Contactos, Tipo de contactos,Llamadas). La funcionalidad de la aplicación permite:

- Insertar
  - Un nuevo Contacto
  - Una nueva Llamada
  - Un nuevo Tipo de contacto
- Actualizar
  - Un Contacto
  - Una Llamada
  - Un Tipo de contacto
- Consultar
  - Todos los Contactos
    - Filtrados por apellido
  - Todas Llamadas
    - Filtradas por contacto
  - Tipos de contacto

3 DISEÑO ARQUITECTÓNICO

El diseño arquitectónico de la aplicación está compuesto por tres subsistemas de la aplicación y uno de pruebas.

La descripción de los subsistemas es la siguiente:

- subsistema ubu.lsi.dms.agenda.persistencia:
  - Contiene las clases que permiten acceder al sistema de persistencia de la

agenda.

- Implementa dos formas de persistencia, una basada en persistencia de base de datos y otra basada en persistencia de flujos de ficheros binarios.
- Permite configurar al cliente con uno de los dos subsistemas sin que tenga dependencias sobre las implementaciones concretas. El cliente usa una propiedad de la aplicación para seleccionar la implementación concreta.

- subsistema ubu.lsi.dms.agenda.gui
  - Contiene las clases que permiten al usuario interactuar la agenda (consultar, insertar, actualizar)
- subsistema ubu.lsi.dms.agenda.modelo
  - Contiene las clases de tipo entidad (entity class) del contexto de la aplicación de agenda
- subsistema ubu.lsi.dms.test.agenda.persistencia
  - Contiene las clases que sirven para probar la funcionalidad del subsistema ubu.lsi.dms.agenda.persistencia.

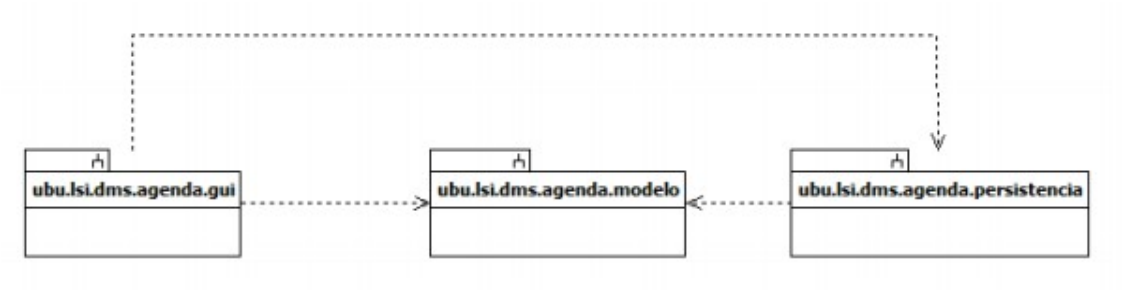

Ilustración 1: Diseño Arquitectonico Agenda

4 DISEÑO DE DATOS

Un contacto realiza cero o varias llamadas siempre asociadas a un contacto. Los contactos tienen asociado un único tipo de contacto.

Ilustración 2: Diagrama de Clases - Subsistema Modelo.

5 CREACIÓN BASE DE DATOS

A partir de los siguientes comando SQL creamos la base de datos en hsqlDB.

```
CREATE TABLE Contactos (
 IdContacto INT NOT NULL PRIMARY KEY IDENTITY,
 Nombre VARCHAR(50) NULL,
 Apellidos VARCHAR(50) NULL,
 Estimado VARCHAR(50) NULL,
 Direccion VARCHAR(255) NULL,
 Ciudad VARCHAR(50) NULL,
 Prov VARCHAR(20) NULL,
 CodPostal VARCHAR(20) NULL,
 Region VARCHAR(50) NULL,
 Pais VARCHAR(50) NULL,
 NombreCompania VARCHAR(50) NULL,
 Cargo VARCHAR(50) NULL,
 TelefonoTrabajo VARCHAR(30) NULL,
 ExtensionTrabajo VARCHAR(20) NULL,
 TelefonoMovil VARCHAR(30) NULL,
 NumFax VARCHAR(30) NULL,
 NomCorreoElectronico VARCHAR(50) NULL,
 IdTipoContacto INT NULL,
 Notas VARCHAR(512) NULL
)
```

```
CREATE TABLE Llamadas (
 IdLlamada INT NOT NULL PRIMARY KEY IDENTITY,
 IdContacto INT NOT NULL,
 FechaLlamada TIMESTAMP NOT NULL,
 Asunto VARCHAR(255) NOT NULL,
 Notas VARCHAR(255) NULL,
)
```

```
CREATE TABLE Tiposdecontacto (
 IdTipoContacto INT NOT NULL PRIMARY KEY IDENTITY,
 TipoContacto VARCHAR(50) NOT NULL,

)
```

En el caso de la implementación binaria se gestionará mediante lectura escritura en archivos .dat

6 PATRONES IMPLEMENTADOS

- Singleton

Este patrón es utilizado para controlar la situación en la que dos clientes quieres utilizar la agenda. De la forma que solo se genera una instancia para todos los clientes.

- Fabrica Abstracta

Un sistema debe ser independiente de cómo se crean o componen cada uno de los productos. Por lo que configuramos el sistema para que únicmaente se revele sus interfaces y no sus implementaciones. Para ello se ha declarado Fabrica Presistencia, FabricaBD.java y FabricaBin.java

- Fachada

Simplifica el acceso a un conjunto de objetos relacionados proporcionando un único

objeto con el que los objetos externos pueden interactuar con el subsistema.

El fin por el que se utiliza este tipo de patrón es porque el cliente no necesita tener el conocimiento de la información que el sistema va a utilizar. Se proporciona un objeto adicional mediante el cual dicho cliente interactuará con el sistema.

De esta manera se podría cambiar la implementación del sistema sin que se cree una repercusión directa con el cliente.

7 PRUEBAS REALIZADAS

Partimos de un conjunto de archivos vacios en el caso de la implementacion binaria y en en el caso de la implementación SQL se creará la BD como anteriormente se ha mostrado en el punto 5. Creación de Base de Datos.

Creamos las estructuras de datos correspondientes a cada uno de los tipos mediante los cuales vamos a gestionar nuestra agenda y las introducimos en la BD o los ficheros Binarios.

- Prueba Contactos.

1. Extraemos un contacto con el apellido solicitado
2. Se hace la comprobación mediante assertos que todos los atributos de contacto extraido coinciden con el contacto solicitado. De este mismo modo comprobamos que la inserción y la obtención de los contactos son correctas.
3. En cuanto a la actualización, se modifica el apellido de un contacto solicitado, y se comprueba de la misma manera que el apellido a sido modificado.

- Pruebas Llamadas

1. Extraemos un contacto con un cierto apellido random.
2. Se realiza la comprobación del número de llamadas realizadas por ese contacto.
3. En cuanto a la actualización, se actualizan los atributos de una llamadas y se comprueban que no sean iguales mediante assertos.

- Pruebas Tipo Contacto

1. Extraemos un tipo de contacto a partir de una Id.
2. Se realiza la comprobación del número de tipos de contacto que tenemos en la BD.
3. En cuanto a la actualización, se actualizan los atributos de un tipo de contacto y se comprueban que no sean iguales mediante assertos.

8 REFERENCIAS BIBLIOGRÁFICAS

[1] Carlos López. Apuntes de la asignatura diseño y mantenimiento del software, 2014. disponible en <https://ubuvirtual.ubu.es/>.

[2] OpenOffice.org, "Creating large documents with OpenOffice.org Writer," s.d., http://wiki.services.openoffice.org/wiki/Documentation.