Diseño de un subsistema de persistencia para una aplicación de Agenda

Índice de Contenido

1 Requisitos Software

- Usar un IDE, recomendado Eclipse IDE for Java Developers http://www.eclipse.org/downloads/
- Plugin Eclipse HSQLDB Database Server plugin https://marketplace.eclipse.org/content/hsqldb-database-server-plugin
- Driver JDBC de la base de datos HSQLDB http://hsqldb.org/, esta dependencia viene incluida en el fichero de gradle.
- Herramientas de modelado para diagramas de clase. Por ejemplo podeis usar ObjectAiD http://www.objectaid.com/ u otras herramientas que os permitan dibujar diagramas UML como http://draw.io/.

2 Objetivos Especificos

- Diseñar software para mejorar el mantenimiento de los sistemas.
- Aplicar patrones de diseño creacionales y estructurales sobre Java.

3 Enunciado

Se quiere implementar una aplicación que gestione los datos de una agenda (contactos, tipo de contactos, llamadas). La funcionalidad de la aplicación, en modo consola, permite:

- Elegir sistema de persistencia (base de datos o fichero binario)
- 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/ordenados por apellido
        - Filtrados/ordenados por nombre
    - Todas Llamadas
        - Filtrados/ordenados por contacto
        - Filtrados/ordenados por fecha de realización
    - Tipos de contacto

3.1 Diseño Arquitectónico

El diseño arquitectónico de la aplicación está compuesto por tres subsistemas de la aplicación y uno de pruebas. Las dependencias de los subsistemas de la agenda se muestran en la Ilustración 1. La descripción de los subsistemas es la siguiente:

- Subsistema `main.java.persistence`
    - 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 `main.java.gui`
    - Contiene las clases que permiten al usuario interactuar con la agenda (consultar, insertar, actualizar).
- Subsistema `main.java.modelo`
    - Contiene las clases de tipo entidad (*entity class*) del contexto de la aplicación de agenda.
- Subsistema `test.java`
    - Contiene las clases que sirven para probar la funcionalidad del resto de subsistemas.

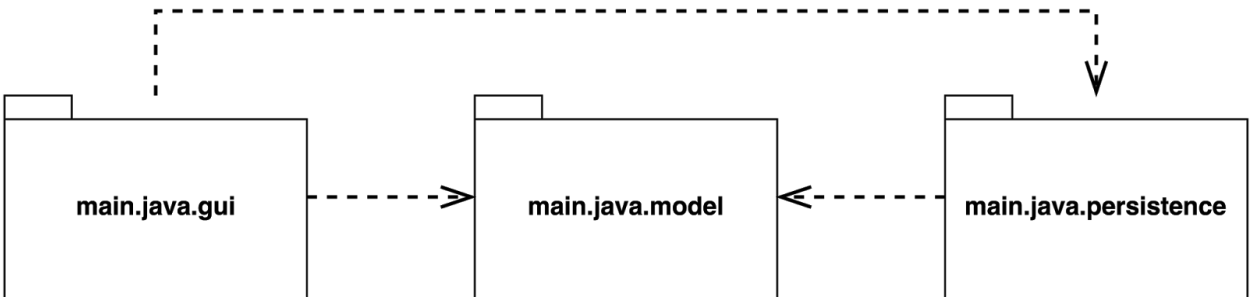

Ilustración 1: Diseño arquitectónico de la aplicación Agenda

3.2 Diseño de Datos

En el diagrama de clases de la Ilustración 2 se muestran las clases de tipo entidad y sus relaciones en la aplicación agenda. 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 del subsistema ubu.lsi.dms.agenda.modelo

En la Tabla 1 se muestran los comandos SQL concretos para crear una base de datos relacional que sirva para soportar la persistencia del subsistema `main.java.modelo`.

```
CREATE TABLE Contacts (
  id INT NOT NULL PRIMARY KEY IDENTITY,
  name VARCHAR(50) NULL,
  surname VARCHAR(50) NULL,
  title VARCHAR(50) NULL,
  address VARCHAR(255) NULL,
  city VARCHAR(50) NULL,
  province VARCHAR(20) NULL,
  postal_code VARCHAR(20) NULL,
  region VARCHAR(50) NULL,
  country VARCHAR(50) NULL,
  company_name VARCHAR(50) NULL,
  workstation VARCHAR(50) NULL,
  work_phone VARCHAR(30) NULL,
  work_extension VARCHAR(20) NULL,
  mobile_phone VARCHAR(30) NULL,
  fax_number VARCHAR(30) NULL,
  email VARCHAR(50) NULL,
  contact_type_id INT NULL,
  notes VARCHAR(512) NULL
);
```

```
CREATE TABLE Calls (
  id INT NOT NULL PRIMARY KEY IDENTITY,
  contact_id INT NOT NULL,
  call_date TIMESTAMP NOT NULL,
  subject VARCHAR(255) NOT NULL,
  notes VARCHAR(255) NULL,
);
```

```
CREATE TABLE ContactsTypes (
  id INT NOT NULL PRIMARY KEY IDENTITY,
  contact_type VARCHAR(50) NOT NULL,
);
```

Tabla 1: Comandos SQL de creación en el modelo relacional de datos

En UbuVirtual puedes obtener el código fuente de la primera versión de la práctica.

4 Productos a Entregar

Un proyecto de Gradle (bibliotecas, recursos y fuentes Java) con la implementación de los subsistemas :

- `main.java.gui` por ahora **solo interfaz en consola**.
- `main.java.persistence`
- `main.java.modelo`
- `test.java.persistence` para probar la funcionalidad del subsistema de persistencia.

Todos el código fuente debe estar comentado y bien formateado.

Además se debe entregar un documento de memoria con el formato de la plantilla disponible en UBUVirtual, donde se presenten los siguientes apartados:

- Descripción de la aplicación
- Diseño arquitectónico
- Diseño detallado del subsistema `main.java.persistence`. La documentación de diseño debe especificarse desde la perspectiva de patrones de diseño (problema, solución y participantes todos ellos en el contexto específico del software que estamos creando).
- Se deben utilizar por lo menos **dos patrones de diseño** a mayores de los que ya se usan en el código proporcionado (si se usan más de dos se tendrá muy en cuenta).
- Manual del programador donde se explique el contenido de cada carpeta (src, res, lib etc...), cómo compilar y ejecutar la aplicación.
- Manual de usuario donde se explique como funciona la aplicación.

5 Bibliografia

- Apuntes de la asignatura: Diseño y Mantenimiento del software, 2016.
- Carlos López, Raúl Marticorena, Judith Antolín y Ignacio Cruzado. «Inclusión de patrones de diseño en un plan de estudios de Ingeniería Técnica en Informática de Gestión». En *Actas de las IX Jornadas de Enseñanza Universitaria en Informática JENUI 2003. Cádiz. ISBN 84-283-2845-5*, 265-272, 2003. http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2003/loincl.pdf