Formulación de proyecto (Aplicación hibrida sobre turismo) (Febrero 2017)

York Luis Gutiérrez Navia, *Universidad Sergio Arboleda*

I. MARCO TEORICO

Diagrama de clases UML

La arquitectura orientada por modelos (ModelDriven Architecture – MDA) es el enfoque definido por el OMG para posibilitar la separación de las especificaciones de la solución a un problema de la implementación de esa solución en una plataforma de implementación definida (e.g. Oracle®, J2EE, Java, etc.). Para ello, define dos tipos de modelos (OMG, 2006A): • Modelos Independientes de la plataforma de implementación (Platform independent models – PIM), que son modelos UML que describen las características de la solución, pero que aún no poseen elementos que sean propios de una plataforma específica. • Modelos Específicos de la plataforma de Implementación (Platform Specific models – PSM), que son modelos UML enriquecidos con estereotipos que describen ciertos elementos que son propios de cada una de las plataformas de implementación disponibles para la elaboración de código ejecutable. El refinamiento se define en términos de MDA como la adición de los detalles necesarios para pasar de un modelo abstracto (por ejemplo un PIM) a un modelo más detallado (que puede ser un PSM) que en general estará más cerca de una plataforma de implementación. Una opción para la definición de esta transformación se presenta en Arango et al. (2006), donde se realiza un meta modelo simplificado del diagrama de Clases de UML y se elabora un meta modelo también simplificado del modelo relacional de Oracle® 9i, los cuales se pueden ver en las Figuras 1 y 2 respectivamente. Adicionalmente, en este trabajo se realiza una recopilación de las reglas de refinamiento entre ambos modelos, entre las cuales se cuentan las siguientes (algunas de ellas con su expresión en lógica de predicados; un mayor detalle se puede consultar en Arango et al., 2006):

• Regla 1: Toda instancia de la Metaclase UML Class, que no participe en ninguna relación de generalización se transforma en una instancia de la Metaclase Oracle Tabla Persistente si su atributo isAbstract es falso, conservando el mismo nombre.

• Regla 2: Todas las instancias de la Metaclase UML Property que pertenecen a una Clase UML (valores del atributo ownedAttribute), se convierten en instancias de la Metaclase Oracle Columna, manteniendo el mismo nombre. Esta Metaclase es abstracta y se especializa en las Metaclases Oracle Atributo Estructural y Atributo de Implementación. Las Columnas obtenidas pertenecerán a la instancia de la Metaclase Oracle Tabla Persistente a la que se mapeó la Clase y constituyen los valores del atributo atributo_tabla de dicha Metaclase Oracle.

• Regla 3: La generalización se transforma a una tabla que reúne tanto los atributos de la superclase (obligatorios y no obligatorios) como los de las subclases (opcionales). Además se debe tener una columna obligatoria llamada tipo para especificar para cada tupla de la tabla a cual clase de la jerarquía se asocia.

• Regla 4: Para las asociaciones uno a muchos, la cardinalidad uno se representa como un nueva columna en la tabla del lado de muchos, del mismo tipo de la clave primaria de la tabla del lado de uno, con sus respectivas restricciones de clave foránea y obligatoriedad. [1]

Modelo entidad relación

Entidad-relación es uno de los diagramas que se utilizan en el desarrollo de modelos para representar la información de un dominio. Con el fin de agilizar y mejorar el proceso de desarrollo de software, diferentes propuestas surgieron para contribuir en la obtención automática o semiautomática del diagrama entidad-relación. Varias de estas propuestas utilizan como punto de partida lenguaje natural o lenguaje controlado, mientras otras propuestas utilizan representaciones intermedias. Los interesados en el desarrollo de una aplicación de software no suelen comprender varias de las representaciones

utilizadas sin tener previa capacitación, lo cual restringe la participación activa del interesado en todas las etapas del desarrollo. Con el fin de solucionar estos problemas, en este artículo se propone un conjunto de reglas heurísticas para la obtención automática del diagrama entidad-relación y su representación en SQL. Se toma como punto de partida el lenguaje controlado UN-Lencep, que ya se emplea para la generación de otros artefactos en el desarrollo de aplicaciones de software.

Entidad-Relación es un diagrama de ingeniería de software que se utiliza para desarrollar un modelo de datos de alta calidad. Este diagrama, paulatinamente se está convirtiendo en la técnica universal para modelar datos. Por ello, con el fin de mejorar y agilizar el proceso de desarrollo de software, son diversas las propuestas dirigidas a permitir la obtención automática y semiautomática de los diferentes elementos del diagrama entidad-relación y su correspondiente representación en el lenguaje SQL. [2]

Programación orientada a objetos

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.
Con la POO tenemos que aprender a pensar las cosas de una manera distinta, para escribir nuestros programas en términos de objetos, propiedades, métodos y otras cosas que veremos rápidamente para aclarar conceptos y dar una pequeña base que permita soltarnos un poco con este tipo de programación.
Durante años, los programadores se han dedicado a construir aplicaciones muy parecidas que resolvían una y otra vez los mismos problemas. Para conseguir que los esfuerzos de los programadores puedan ser utilizados por otras personas se creó la POO. Que es una serie de normas de realizar las cosas de manera que otras personas puedan utilizarlas y adelantar su trabajo, de manera que consigamos que el código se pueda reutilizar.
La POO no es difícil, pero es una manera especial de pensar, a veces subjetiva de quien la programa, de manera que la forma de hacer las cosas puede ser diferente según el programador. Aunque podamos hacer los programas de formas distintas, no todas ellas son correctas, lo difícil no es programar orientado a objetos sino programar bien. Programar bien es importante porque así nos podemos aprovechar de todas las ventajas de la POO. [3]

Programación en Java

Java no se pensó originalmente como lenguaje para internet, Sun Microsystems, la empresa estadounidense creadora del lenguaje y de la plataforma, comenzó a desarrollarlo con el objetivo de crear un lenguaje independiente de la plataforma y del sistema operativo que permitiera su diseño y construcción en la floreciente electrónica de consumo (dispositivos como televisores, reproductores de video, equipos de música, etcétera). El proyecto original, denominado Green, comenzó apoyándose en C++ pero a medida que progresaba su desarrollo, el equipo creador se encontró con dificultades, especialmente de portabilidad; para evitar esto decidieron desarrollar su propio lenguaje, y en agosto de 1991 nació uno nuevo orientado a objetos y al cual llamaron Oak. En 1993, Green se renombró First Person Juc.; Sun invirtió, sin mucho éxito, un gran presupuesto y esfuerzo fundamentalmente humano para intentar vender esta tecnología, hardware y software. A mitad de 1993 se lanzó Mosaic, el primer gran navegador web, y comenzó a crecer el interés por internet (y en particular por la World Wide Web); después se rediseñó el lenguaje para desarrollar internet y, en enero de 1995, Oak se convirtió en Java. Sun lanzó el entorno JDK 1.0 (java development kit) en 1996 como primera versión del kit de desarrollo de dominio público y se convirtió en la primera especificación formal de la plataforma Java; desde entonces se han lanzado diferentes versiones, aunque JDK 1.1, la primera versión comercial, se lanzó a principios de 1997. En diciembre de 1998, Sun lanzó JDK 1.2 pero la renombró como Java 2 y comenzó a utilizarse el nombre de J2SE (Java 2 Platform, Standard Edition) para diferenciar las plataformas base de J2EE (Java 2 Platform, Enterprise Edition) y J2ME (Java 2 Platform, Micro Edition); además de la versión estándar SE, Sun lanzó otras dos ediciones populares: Micro Edition (ME) para dispositivos empotrados (embebidos) tales como teléfonos celulares (móviles) y la edición empresarial (Enterprise Edition, EE) para

procesamiento desde el servidor. Este libro se centra esencialmente en la edición SE. En mayo de 2000 se lanzó J2SE 1.3, y en febrero de 2002, la J2SE 1.4; ambas trajeron consigo un gran número de clases e interfaces a las bibliotecas estándar de Java. Sin embargo fue la versión 5.0, la primera después de la versión 1.1, la que implicó una actualización de Java de modo significativo; dicha versión originalmente se nombró 1.5, pero el número se cambió a 5.0 en la conferencia JavaOne de 2004. Después de años de investigación se añadieron tipos genéricos similares a las plantillas o templates de C++, también se agregaron propiedades de C# (el lenguaje creado por Microsoft), pre cisamente para competir con Java en internet, como el bucle "for each", así como el manejo de metadatos para su uso, principalmente, en bases de datos, y otras características como enumeraciones, varargs (número de argumentos variables), entre otras. La versión 6 (sin el sufijo 0) se lanzó en diciembre de 2006 y hoy día es la más utilizada y recomendada para su descarga del sitio web de Sun o de Oracle (su actual propietario), cuyas direcciones web se indican en el apartado 1.12. Esta versión no ha traído cambios al lenguaje, sino mejoras adicionales al rendimiento y a la interfaz gráfica, así como un nuevo marco de trabajo y API (interfaces de programación de aplicaciones) junto con soporte para servicios web e implementaciones de JavaScript, fundamentalmente para los buscadores, tales como Firefox de la fundación Mozilla. Al poco tiempo de liberar la versión 6, Sun4 comenzó un nuevo proyecto cuyo nombre clave es Dolphin, aunque ha comenzado a denominarse Java 7 y se espera que a mediados de 2011 se lance Java SE 7 y a finales de 2012, Java SE 8. Al parecer traerá una nueva biblioteca de clases, junto con soporte para XML, un nuevo concepto de superpaquete, introducción de anotaciones estándar para detectar fallos en software, nuevas API para manejo de calendario de fechas y días, etcétera. Este libro cubre fundamentalmente Java 5.0 y Java 6, pero puede utilizarse por quienes sigan utilizando la versión 2 y, a pesar de que Java 7 no se ha lanzado de forma oficial, al escribir este libro consideramos que el lector podrá migrar, si lo desea, a esta última versión, sin problemas de compatibilidad; aunque se entiende que pasarán algunos años hasta la implantación de esta futura versión, entre otras razones, porque la gran comunidad de desarrolladores de Java sigue empleando Java 2 y, con más frecuencia, la plataforma Java SE 6, ya que es la que Sun actualiza con más frecuencia; de hecho, ya se ofrece la actualización número 21.

Las características de java En C, C++ o Pascal, el compilador traduce el código fuente directamente al lenguaje máquina de su computadora, entendible por su CPU; estos lenguajes necesitan un compilador diferente para cada tipo de CPU y en consecuencia, cuando los programas escritos en estos lenguajes se traducen a código máquina no son portables, de modo que el código fuente debe ser recompilado para cada tipo de máquina o CPU. Para solucionar este problema y hacer que los programas de Java sean independientes de la máquina y se puedan transportar o "portar" fácilmente a otras máquinas y también puedan ejecutarse en navegadores web, los creadores de Java introdujeron el concepto, citado anteriormente, de máquina virtual Java y el bytecode como el lenguaje máquina de la CPU específica, donde la máquina virtual Java los ejecuta o interpreta. Existen numerosas máquinas virtuales disponibles para un gran número de plataformas que permiten a los programas ser independientes de la máquina, de modo que un programa compilado en una estación UNIX puede ejecutarse en un sistema operativo Macintosh o en Windows 7; esta característica se debe a que el intérprete de Java traduce y ejecuta una instrucción de bytecode cada vez sin traducir el código completo como sucede con otros compiladores, por ejemplo, C++, necesita un compilador diferente para cada tipo de máquina, mientras que un compilador Java traduce un programa fuente en Java a bytecode, el lenguaje máquina de la máquina virtual Java (JVM), independiente del tipo específico de CPU. El intérprete de Java traduce cada instrucción en bytecode en el tipo específico de lenguaje máquina de la CPU y, a continuación, ejecuta la instrucción; por consiguiente, Java sólo necesita un tipo diferente de intérprete para cada tipo específico de CPU, además es posible señalar que los intérpretes son programas más sencillos que los compiladores, aunque más lentos. Otra fortaleza de Java, como se verá más adelante al estudiar paquetes y bibliotecas, es que incluye bibliotecas de clases incorporadas; dichos paquetes vienen con los entornos de desarrollo JDK (Java development kit) y contienen

centenares de clases integradas con millares de métodos, como se señala en la tabla 1.4. Los creadores de Java escribieron un artículo,6 ya clásico, en el que definen el lenguaje y recogen sus once características más sobresalientes:
• Sencillo
• Orientado a objetos.
• Distribuido (características de red, especialmente internet).
• Portable.
• Interpretado.
• Robusto
• Seguro.
• Arquitectura neutra.
• Alto rendimiento.
• Multihilo (multithreaded).
• Dinámico. [4]

PostgreSQL
El lenguaje estructurado de consultas (SQL) es un lenguaje de base de datos normalizado, utilizado por la gran mayoría de los servidores de bases de datos que manejan bases de datos relacionales u objeto-relacionales. Es un lenguaje declarativo en el que las órdenes especifican cual debe ser el resultado y no la manera de conseguirlo (como ocurre en los lenguajes procedimentales). Al ser declarativo es muy sistemático, sencillo y con una curva de aprendizaje muy agradable ya que sus palabras clave permiten escribir las ordenes como si fueran frases en las que se especifica (en inglés) que es lo que queremos obtener. Por ejemplo: SELECT nombre FROM municipios WHERE poblacion>5000 ORDER BY poblacion; Devuelve el nombre de aquellos municipios con una población mayor de 5000 habitantes y los presenta ordenados por tamaño. Sin embargo los lenguajes declarativos carecen de la potencia de los procedimentales Se ha convertido, debido a su eficiencia, en un estandar para las bases de datos relacionales, de hecho el gran éxito del modelo de base de datos relacional se debe en parte a la utilización de un lenguaje como SQL. A pesar de su tesórico caracter estandar, se han desarrollado, sobre una base común, diversas versiones ampliadas como las de Oracle o la de Microsoft SQL server. Incluye diversos tipos de capacidades: • Comandos para la definición y creación de una base de datos (create table). • Comandos para inserción, borrado o modificación de datos (insert, delete, update). • Comandos para la consulta de datos seleccionados de acuerdo a criterios complejos que involucran diversas tablas relacionadas por un campo común (select). • Capacidades aritméticas: En SQL es posible incluir operaciones aritméticas así como comparaciones, por ejemplo A > B + 3. • Asignación y comandos de impresión: es posible imprimir una tabla construida por una consulta o almacenarla como una nueva tabla. 1 • Funciones de agregación: Operaciones tales como promedio (average), suma (sum), máximo (max), etc. se pueden aplicar a las columnas de una tabla para obtener una cantidad única y, a su vez, incluirla en consultas más complejas. [5]

API (Application Programming Interface)

Una API (interfaz de programación de aplicaciones) es un conjunto de rutinas y especificaciones de comunicación entre componentes de software. Existen varios tipos de API: nativas de un sistema operativo como ODBC en Windows, nativas de un lenguaje como JDBC con Java y nativas de un SGBD como OCI con Oracle. Enseguida se presentan algunas de las API de mayor utilización para el acceso a bases de datos.

- ODBC (Open Data Base Connectivity). Es un estándar planteado por Microsoft cuyo objetivo es hacer posible el acceso a cualquier dato de cualquier aplicación, sin importar qué SGBD lo almacene; ODBC logra esto al insertar una capa intermedia llamada manejador (Driver) de bases de datos, entre la aplicación y el SGBD; el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el SGBD pueda interpretar. Desde la versión 2.0 el estándar soporta SQL (Structured Query Language) [17]. Para conectarse a la base de datos se crea en el sistema operativo un origen de datos (DSN: Data Source Name) dentro del cual se definen los parámetros, la ruta y las características de la conexión, según las especificaciones del fabricante.

- JDBC (Java Data Base Connectivity). Permite la ejecución de sentencias sobre bases de datos desde Java, independiente del sistema operativo donde se esté ejecutando o del SGBD utilizado [18]. Provee servicios para acceder a una base de datos utilizando SQL como lenguaje de consulta y manipulación de datos.

Un manejador (Driver) de conexión hacia un SGBD específico es un conjunto de clases que implementan las interfaces Java definidas en JDBC y que utilizan los métodos de registro para definir un localizador de una base de datos (URL: Uniform Resource Locator). En la ejecución, el programa debe correr con la librería apropiada para su SGBD y accede a éste estableciendo una conexión y utilizando el localizador a la base de datos con los parámetros específicos.

La API de persistencia recientemente desarrollada para la plataforma Java EE se llama JPA (Java Persistence API), uno de cuyos principales objetivos es unificar la manera en que funcionan las utilidades de mapeo objeto-relacional.

- OCI (Oracle Call Interface). Permite ejecutar sentencias SQL para la consulta y manipulación de datos en un SGBD Oracle, desde un lenguaje de programación como C o PHP [19]. Las librerías OCI no necesitan ser recompiladas para su utilización y se tratan de igual forma que las aplicaciones que no son para bases de datos. Las funciones de OCI son:

- Hacer un análisis sintáctico de las sentencias SQL.

- Abrir un cursor (espacio de memoria para guardar la respuesta a una consulta).

- Introducir variables del cliente en la memoria compartida del servidor.

- Ejecutar las sentencias SQL en el espacio de memoria del cursor.

- Descargar en la aplicación cliente los registros de datos solicitados.

- Cerrar los cursores y, de esta forma, liberar los espacios de memoria reservados. [6]

II. REFERENCIAS

[1] REFINAMIENTO DEL DIAGRAMA DE CLASES UML A ORACLE. (2017). [online] redalyc. Available at: http://www.redalyc.org/articulo.oa?id=49615116 [Accessed 17 Feb. 2017].

[2] Zapata Jaramillo, C., González Calderón, G. and Chaverra Mojica, J. (2017). Automatic generation of entity-relationship diagram and its representation in SQL from a controlled language (UN-LENCEP). [online] Scielo.org.co. Available at: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1692-33242011000100014 [Accessed 17 Feb. 2017].

[3] Cox, B., Novobilski, A., García-Bermejo Giner, R. and Joyanes Aguilar, L. (1993). Programación orientada a objetos. 1st ed. Reading, Massachusetts [etc.]: Addison-Wesley [etc.].

[4] Joyanes Aguilar, L. and Zahonero Martínez, I. (2011). Programación en java 6. 1st ed. México: McGraw-Hill.

[5] Obe, R. and Hsu, L. (2012). PostgreSQL. 1st ed. Sebastopol, CA: O'Reilly Media.

[6] Quintero, J., Hernández, D. and Yanza, A. (2017). DIRECTRICES PARA LA CONSTRUCCIÓN DE ARTEFACTOS DE PERSISTENCIA EN EL PROCESO DE DESARROLLO DE SOFTWARE. [online] Scielo.org.co. Available at: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1794-12372008000100007 [Accessed 17 Feb. 2017].