Práctica 03

Modelado con Diagramas de Clases UML

1. Objetivo General

Aprender a generar el modelo de la base de datos que se pretende implementar utilizando diagramas de Clases UML.

El uso de diagramas de Clases permitirá integrar detalles específicos de los datos que se almacenarán, como lo son el tipo y el tamaño de los atributos de cada entidad o relación.

2. Objetivos Secundarios

- Definir gráficamente las tablas necesarias para implementar la base de datos.
- Conocer los diferentes tipos de datos que soportan los SMBD.
- Introducir el concepto de Integridad y sus tipos: Referencial, de Entidad, de Dominio y Manejo de Nulos.
- Obtener el diagrama de clases para su estudio, presentación y aprobación por parte del cliente, así como referencia para futuras complementaciones o mejoras.

3. Introducción

El Lenguaje de Modelado Unificado[1] (UML por sus siglas en inglés) es un lenguaje estandarizado de propósito general para el modelado de sistemas en Ingeniería de Software, en particular para el desarrollo de tecnología orientada a objetos. UML incluye un conjunto de notaciones gráficas para crear modelos visuales de sistemas orientadas a objetos.

En el año de 1997 el consorcio Object Management Group[2] (OMG por sus siglas en inglés) añadió UML a su lista de estándares debido a su gran popularidad y eficiencia. En el año 2000 UML fue aceptado por la Organización Internacional de Estandarización[3] (ISO por sus siglas en inglés) como un estándar para la industria en el modelado de sistemas de software.

UML es utilizado para especificar, visualizar, modificar, construir y documentar los artefactos de un sistema de software orientado a objetos. UML ofrece una manera estandarizada de visualizar los "planos arquitectónicos" de sistemas incluyendo elementos como[4]:

- Actividades
- Actores
- Procesos de Negocios

---

1. Rumbaugh, J., Jacobson, I., Booch, G.: *The Unified Modeling Language Reference Manual*. Addison-Wesley Professional, 2nd edition (2004)
2. Object Management Group. http://www.omg.org 12/10/2013
3. International Organization for Standardization. http://www.iso.org 12/10/2013
4. Fowler, M., Scott, K.: *UML gota a gota*. Addison Wesley Longman (2000)

- Esquemas de Bases de Datos
- Componentes Lógicos
- Declaraciones de Lenguajes de Programación
- Componentes reusables de software

Para los fines de esta práctica se utilizarán a las clases. Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica. Las clases son gráficamente representadas por cajas con separaciones para:

- Nombre de la Clase
- Atributos
- Operaciones

Como base para la construcción del Diagrama de Clases UML, se utilizará el diagrama Entidad – Relación que se obtuvo durante el *Modelado con Diagramas Entidad – Relación* (Práctica 02). Las tablas de la base de datos serán construidas directamente de las Clases con sus propios Atributos y Relaciones.

Un Diagrama de Clases UML en general se ve como se muestra en la Figura 3.1.

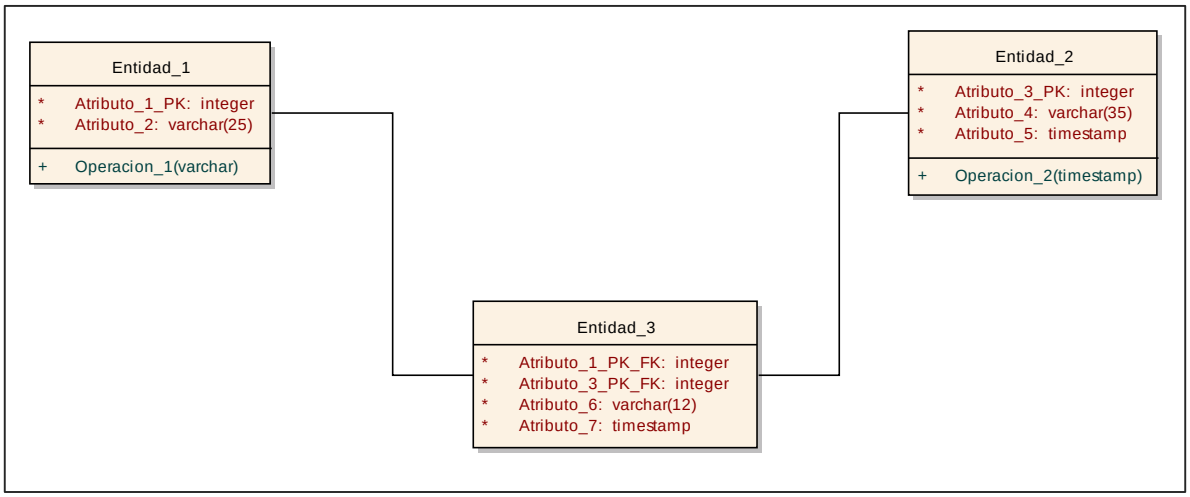

Figura 3.1 - Diagrama de Clases UML genérico.

Cuando se diagrama una base de datos partiendo de un diagrama Entidad – Relación, podemos decir que cada clase representa una tabla (entidad) con sus columnas (atributos). Podemos notar que en el diagrama de Clases se puede hacer referencia a la llave primaria de cada tabla mediante la notación PK antes del nombre del Atributo.

Aún cuando no existe una regla o algoritmo para obtener un diagrama de Clases UML a partir de un diagrama Entidad – Relación, se pueden seguir ciertos pasos para obtener de manera ordenada el diagrama de Clases que se requiere.

1. Transformar directamente Entidades en Clases.
2. Mapear Atributos (Entidad – Relación) en Atributos (Clases), indicando la llave primaria de cada tabla mediante el prefijo PK.
3. Mapear Atributos derivados (Entidad – Relación) en Operaciones, indicando el atributo que recibe como parámetro.
4. Transformar Relaciones cuya cardinalidad sea distinta a *uno a uno* en clases, junto con sus Atributos (si existieran) e identificar la llave primaria.
5. En caso de que la Relación sea *uno a uno*, analizar si es posible agregar la información contenida en ésta (atributos), en alguna de las entidades sobre la cual estaba relacionada originalmente en el diagrama Entidad – Relación. Si esto no fuera factible por el contexto o supuestos del problema, se deberá justificar la decisión y transformar esta Relación en una Clase UML junto a sus Atributos (si existieran), identificando cuales de éstos son la llave primaria.
6. Transferir las líneas que conectan Entidades con Relaciones en el diagrama Entidad – Relación en líneas que conecten Clases con Clases, respetando el orden y revisando la cardinalidad de esas conexiones en el nuevo diagrama UML.

A manera de ejemplificar los pasos anteriores, tomaremos como ejemplo el diagrama que se muestra en la Figura 3.2, éste diagrama se utilizó en durante el *Modelado con Diagramas Entidad – Relación* (Práctica 02), para convertirlo en un Diagrama de Clases.

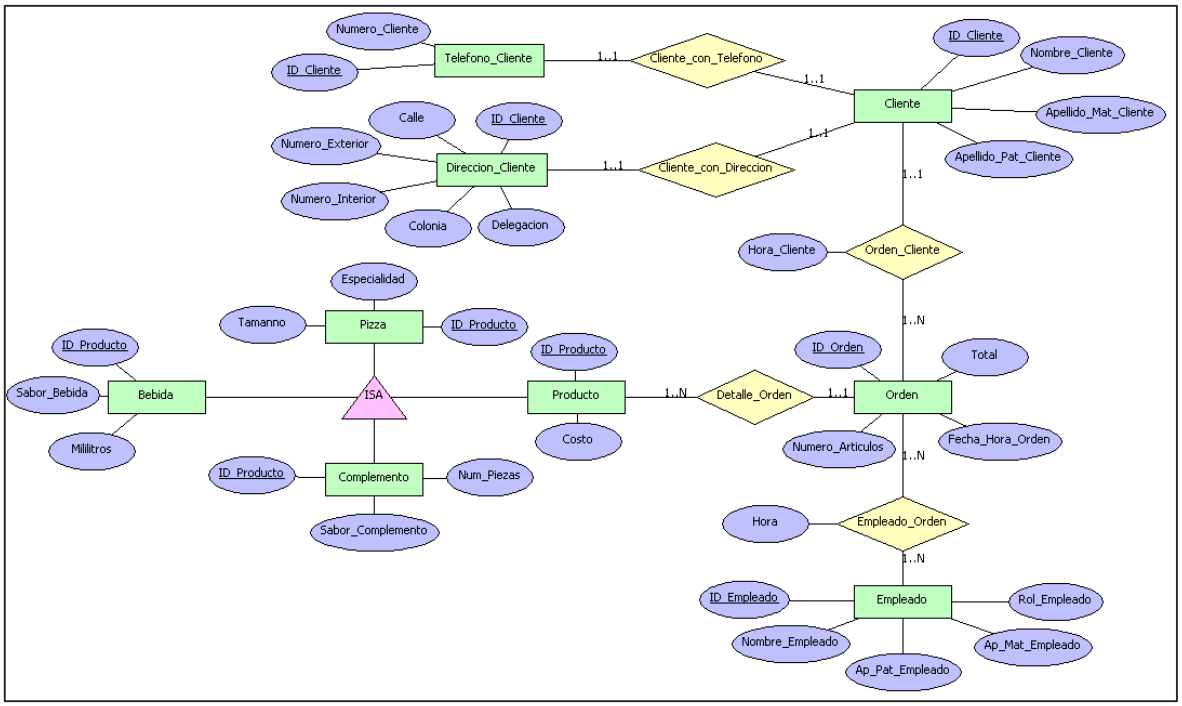

Figura 3.2 - Diagrama Entidad – Relación de la Pizzería.

Empezaremos por transformar todas las Entidades (rectángulos verdes) en Clases. Obteniendo el listado que se muestra en la Tabla 3.1.

Tabla 3.1 - Mapeo entre Entidades y Clases de la Pizzería.

Entidad	Clase
Cliente	Cliente
Teléfono_Cliente	Teléfono_Cliente
Dirección_Cliente	Dirección_Cliente
Orden	Orden
Empleado	Empleado
Producto	Producto
Complemento	Complemento
Bebida	Bebida
Pizza	Pizza

Procedemos ahora a identificar todas las Relaciones (rombos amarillos) cuya cardinalidad no sea *uno a uno* para convertirlas en Clases, el resultado obtenido se muestra en la Tabla 3.2.

Tabla 3.2 - Mapeo entre Relaciones y Clases de la Pizzería.

Relación E-R no unaria	Clase
Orden_Cliente	Orden_Cliente
Detalle_Orden	Detalle_Orden
Empleado_ Orden	Empleado_Orden

Continuar con un análisis de las Relaciones con cardinalidad uno a uno que existen en el diagrama, estas son:

- La relación ISA de la entidad *Producto.* Por su naturaleza de ser especialización, los atributos de las entidades que se encuentran involucradas en ésta relación ya contienen el atributo que heredan de la entidad *Producto*. Es por ello que la relación ISA (*Especialización_Producto)* no se convertirá en clase, ya que de hacerlo, se obtendría una clase con únicamente un atributo.
- La relación *Cliente_con_Direccion*. Desde su construcción, la entidad Dirección_Cliente tiene los atributos necesarios para relacionar una dirección al cliente que le corresponde, por lo tanto no es necesaria su transformación a una clase.
- La relación Cliente_con_Telefono. Caso análogo de la relación Cliente_con_Dirección.

Los diagramas de Clases UML ofrecen mayor información para la creación de la base de datos, como es el caso de los tipos de datos que delimitan el Dominio del atributo. Los tipos de datos varían de acuerdo al Sistema Manejador de Bases de Datos (SMBD) utilizado. La Tabla 3.3 muestra algunos de los diferentes tipos de datos que soporta el SMBD PostgreSQL versión 9.1.8.

Tabla 3.3 - Tipos de datos soportados por PostgreSQL.

Tipo de datos	**Alias**	**Descripción**
Bigint	int8	*Entero con signo de 8 bytes*
bigserial	serial8	*Autoincremento entero de 8 bytes*
bit		*Cadena de bit de longitud fija*
bit varying(n)	varbit(*n*)	*Cadena de bit de longitud variable*
boolean	bool	*Lógico (true/false)*
box		*Rectángulo en el plano*
bytea		*Datos binarios*
character varying(n)	varchar(*n*)	*Cadena de caracteres de longitud variable*
character(n)	char(*n*)	*Cadena de caracteres de longitud fija*
cidr		*Dirección IP de red (IPv4 ó IPv6)*
circle		*Círculo en el plano*
date		*Fecha (año, mes, día)*
double precision	float8	*Número de punto flotante de precisión doble*
inet		*Dirección de un host de red (IPv4 or IPv6)*
integer	int, int4	*Entero con signo de 4 bytes*
interval(p)		*Intervalo de tiempo*
line		*Línea infinita en el plano (no se aplica completamente)*
lseg		*Segmento de línea en el plano*
macaddr		*Dirección MAC de tarjeta o dispositivo de red*
money		*Moneda*
numeric [(p, s)]	decimal [(*p*, *s*)]	*Numérico exacto con precisión modificable*
path		*Trazado geométrico abierto y cerrado en el plano*
point		*Punto geométrico en el plano*
polygon		*Polígono cerrado geométrico en el plano*
real	float4	*Número de punto flotante de precisión simple*
smallint	int2	*Entero con signo de 2 bytes*
serial	serial4	*Autoincremento entero de 4 bytes*
text		*Cadena de caracteres de longitud variable*
time [(p)] [sin zona horaria]		*Hoa del día*
time [(p)] con zona horaria	timetz	*Hora del día, incluyendo la zona horaria*
timestamp [(p)] [sin zona horaria]	timestamp	*Fecha y hora*
timestamp [(p)] con zona horaria	timestamptz	*Fecha y hora incluyendo la zona horaria*

Habrá que prestar atención especial a los atributos derivados, ya que éstos se verán representados en el diagrama de Clases no como atributos, sino como Operaciones. Un atributo derivado será expresado, además de su nombre, por sus parámetros de entrada y salida, siendo la entrada el atributo del cual se derivará y la salida únicamente el tipo de dato que arrojará.

Supongamos que tenemos el atributo derivado *edad* de tipo entero, el cual se obtiene a partir del atributo *fecha de nacimiento*. La representación de este atributo en el diagrama de Clases se verá tal como lo muestra la Figura 3.3.

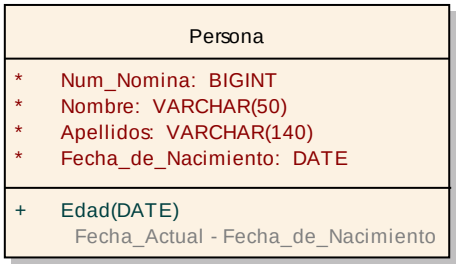

Figura 3.3 - Transformación de un atributo derivado.

En la Figura 3.4, se muestra la transformación de una entidad con atributos a una clase.

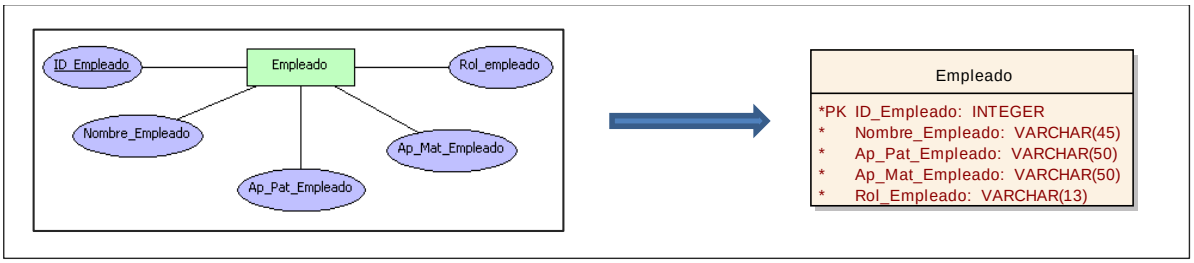

Figura 3.4 - Diagrama de transformación de una Entidad en Clase.

Transformaremos ahora el diagrama Entidad – Relación completo obtenido durante el *Modelado con Diagramas Entidad – Relación* (Práctica 02) siguiendo el mismo procedimiento. El diagrama de Clases resultantes es mostrado en la Figura 3.5.

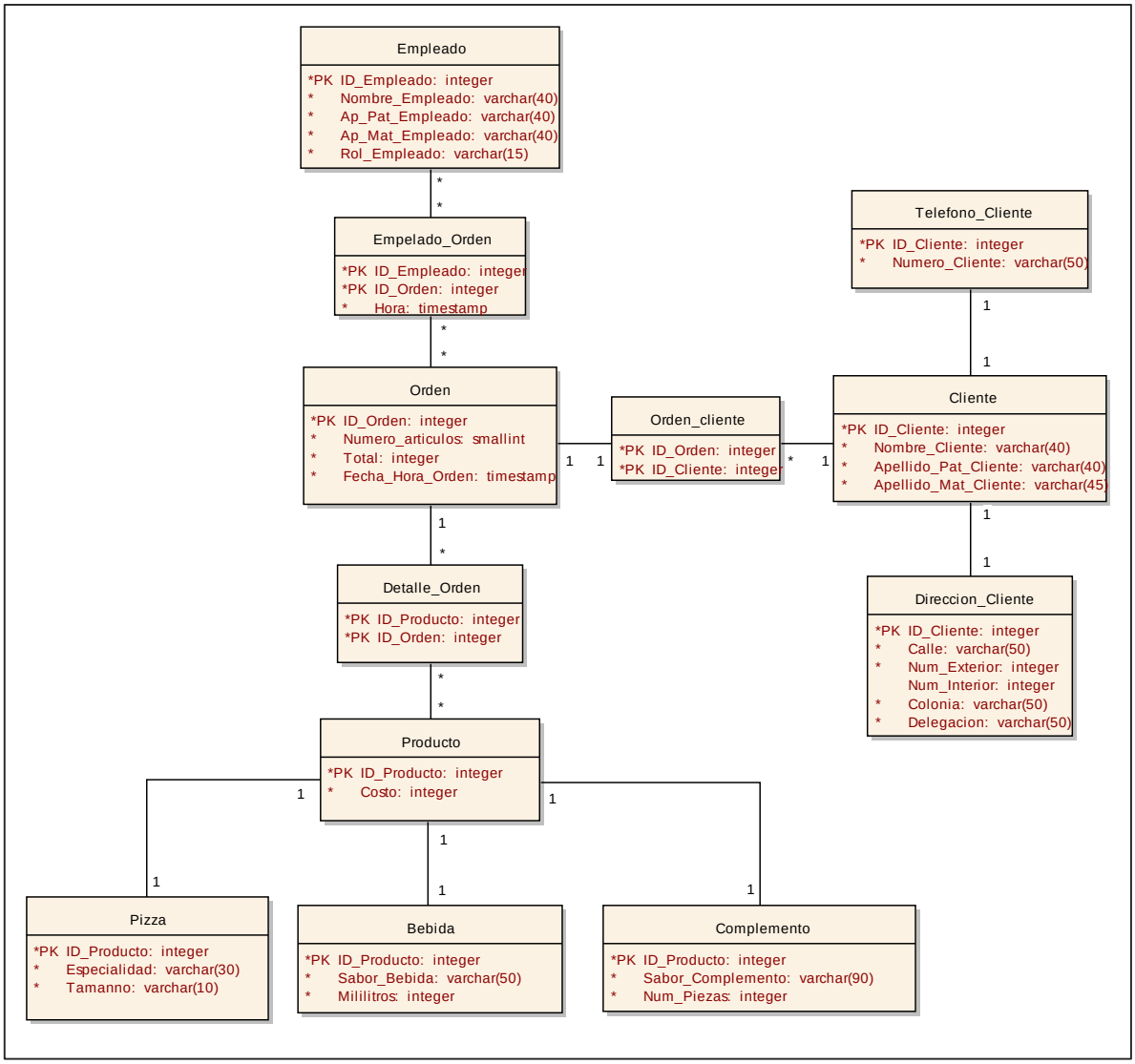

Figura 3.5 - Diagrama de Clases de la Pizzería.

4. Ejercicios

(NOTA: Resuelve los siguientes ejercicios en relación al proyecto que realizarás durante el curso, en dado caso que no tengas un proyecto, utiliza la información en el apéndice NFL-ONEFA parte 03 al final de esta práctica para realizarlos)

1. Comienza la construcción del **Diagrama de Clases** a partir del Diagrama Entidad-Relación de tu proyecto, transformando las Entidades en Clases e identificando los tipos de datos de sus Atributos. Construye el diagrama correspondiente.
2. Identifica las Relaciones que no sean uno a uno del diagrama Entidad – Relación y transfórmalas en Clases. Identifica los tipos de datos de sus Atributos. Actualiza el diagrama.
3. Identifica las Relaciones uno a uno. Después de hacer el análisis de estructura de información para cada una (es decir, si los Atributos de éstas Relaciones pueden o no incluirse en las Entidades con las cuales se relaciona), transforma en Clases las Relaciones que así lo ameriten justificando tu decisión e identifica los tipos de datos de sus Atributos. Actualiza el diagrama.
4. Diagrama las líneas que relacionan Clases de acuerdo al diagrama Entidad – Relación y agrega la cardinalidad correspondiente a cada una de ellas.
5. Realiza un listado en el que describas con tus propias palabras qué representa cada una de las clases, atributos y relaciones de tu diagrama de Clases resultante.

Entregables requeridos para prácticas subsecuentes:

- Diagrama de Clases

5. Apéndice NFL-ONEFA parte 03

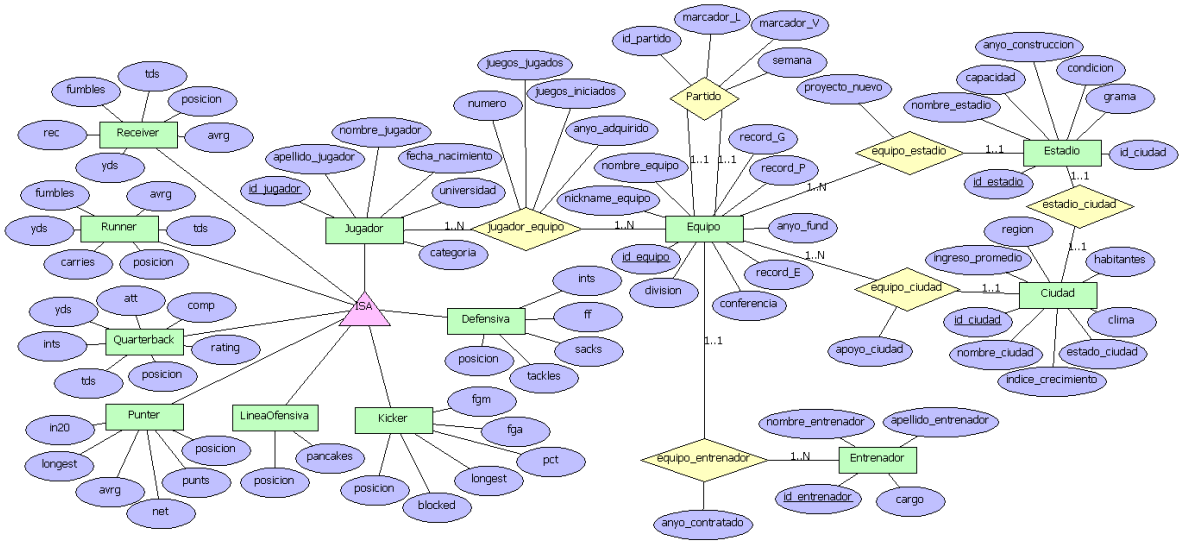

Figura 3.6 - Diagrama Entidad – Relación NFL-ONEFA.

6. Información complementaria

Los diagramas UML son de gran importancia para el proceso posterior de implementación en un sistema de bases de datos, es por ello que tener herramientas de software para el desarrollo de estos diagramas es importante para agilizar dicho proceso.

Esta sección presenta herramientas de diseño para diagramas UML.

6.1. DB Designer fork.

DB Designer fork es un software de licencia libre diseñado para crear diagramas de tipo UML, su uso es bajo Windows, ver Tabla 1.

Tabla 1. Ficha informativa de DB Designer fork

Ficha de la Herramienta	
Nombre:	DB Designer fork
Versión revisada:	1.5 Beta
Página de descarga:	http://sourceforge.net/projects/dbdesigner-fork/
Diagramas soportados:	UML
Requerimientos adicionales de instalación:	-
Licenciamiento:	Freeware
Entorno de diagramación:	Software
Formatos para exportar diagramas:	SQL Scrip/MDB XML/JPEG
Facilidad de uso:	Sencillo
Comentarios adicionales:	-

6.2. MySQL Workbench.

MySQL Workbench es de licencia gratuita, su instalación está disponible para Linux, Mac OS y Windows, este software permite entre otras cosas el desarrollo de bases de datos y administración, así como el diseño de diagramas tipo UML, ver Tabla 2.

Tabla 2. Ficha informativa de MySQL Workbench

Ficha de la Herramienta	
Nombre:	MySQL Workbench
Versión revisada:	6.1.7.11891
Página de descarga:	http://dev.mysql.com/downloads/windows/installer/ http://dev.mysql.com/downloads/workbench/
Diagramas soportados:	UML
Requerimientos adicionales de instalación:	Se necesita descargar los dos archivos para el funcionamiento
Licenciamiento:	Freeware
Entorno de diagramación:	Software
Formatos para exportar diagramas:	SQL Create/PNG/SVG/PDF/PostScript
Facilidad de uso:	Sencillo
Comentarios adicionales:	-

6.3. StarUML.

StarUML este software es de licencia libre y su instalación es bajo Mac OS X y Windows, este programa permite el diseño de diagramas UML, ver Tabla 3.

Tabla 3. Ficha informativa de StarUML

Ficha de la Herramienta	
Nombre:	StarUML
Versión revisada:	5.0.2.1570
Página de descarga:	http://staruml.sourceforge.net/en/download.php
Diagramas soportados:	Modelo Vista/Diagramas Clase/Casos de uso
Requerimientos adicionales de instalación:	-
Licenciamiento:	Freeware
Entorno de diagramación:	Software
Formatos para exportar diagramas:	JPG/JPEG/BMP/EMF/WMF/XMI
Facilidad de uso:	Sencillo
Comentarios adicionales:	Puede generar código y también generar diagramas a partir de código.