
**UNIVERSIDAD TÉCNICA DE AMBATO**

**FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL**

**CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS**

TEMA:

---

# INTERFAZ DE PROGRAMACIÓN DE APLICACIONES PARA LA GENERACIÓN AUTOMÁTICA DE PROCEDIMIENTOS ALMACENADOS EN MYSQL

---

Trabajo de Graduación. Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero en Ingeniero en Sistemas Computacionales e Informáticos

SUBLÍNEA DE INVESTIGACIÓN:

Orientación Objetos

AUTOR: Alexander Ivan Quinaluiza Arias.
TUTOR: Ing. Edison Homero Álvarez Mayorga.

Ambato - Ecuador
Abril, 2018

## CERTIFICACIÓN DEL TUTOR

En mi calidad de Tutor del Trabajo de Investigación sobre el Tema:

"INTERFAZ DE PROGRAMACIÓN DE APLICACIONES PARA LA GENERACIÓN AUTOMÁTICA DE PROCEDIMIENTOS ALMACENADOS EN MYSQL", del señor ALEXANDER IVAN QUINALUIZA ARIAS, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el Art. 16 del Capítulo II, del Reglamento de Graduación para Obtener el Título Terminal de Tercer Nivel de la Universidad técnica de Ambato

Ambato, Abril de 2018

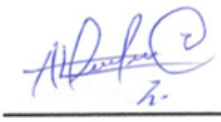
Ing. Edison Homero Álvarez Mayorga

EL TUTOR

## AUTORÍA DEL TRABAJO

El presente trabajo de investigación titulado: INTERFAZ DE PROGRAMACIÓN DE APLICACIONES PARA LA GENERACIÓN AUTOMÁTICA DE PROCEDIMIENTOS ALMACENADOS EN MYSQL. Es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, Abril de 2018



Alexander Ivan Quinaluiza Arias

CC: 1805240304

## APROBACIÓN DEL TRIBUNAL DE GRADO

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ing. Hernán Naranjo y Ing. Carlos Núñez , revisaron y aprobaron el Informe Final del trabajo de graduación titulado "INTERFAZ DE PROGRAMACIÓN DE APLICACIONES PARA LA GENERACIÓN AUTOMÁTICA DE PROCEDIMIENTOS ALMACENADOS EN MYSQL", presentado por el señor Alexander Ivan Quinaluiza Arias de acuerdo al Art. 17 del Reglamento de Graduación para obtener el título Terminal de tercer nivel de la Universidad Técnica de Ambato.

Ing. Elsa Pilar Urrutia, Mg.

PRESIDENTE DEL TRIBUNAL

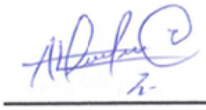
Ing. Hernán Naranjo, Mg.
DOCENTE CALIFICADOR

Ing. Carlos Nuñez, Mg.
DOCENTE CALIFICADOR

## DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este trabajo de titulación como un documento disponible para su lectura, consulta y procesos de investigación. Cedo los Derechos de mi Trabajo de Titulación, con fines de difusión pública, además autorizo su reproducción dentro de las regulaciones de la Universidad.

Ambato, Abril del 2018



Alexander Ivan Quinaluiza Arias

CC: 1805240304

# DEDICATORIA

Dedico este trabajo a mi familia, quienes me han apoyado y han permanecido a mi lado todo este tiempo que he trabajado en este proyecto.

A mi novia L. Toapanta, con quien he compartido sueños, ideas, a ella que me ha prestado su apoyo sincero en este camino educativo.

A los docentes que han compartido sus conocimientos y experiencias para llevar a cabo el desarrollo de este proyecto.

A mis amigos, con quienes compartí mas que una botella, entre alegrías y lamentos que deja la vida de un estudiante universitario.

Alexander Quinaluiza

# AGRADECIMIENTO

Suponen los cimientos de mi desarrollo, todos y cada uno de ustedes –mi familia– han destinado tiempo para enseñarme nuevas cosas, para brindarme aportes invaluables que servirán para toda mi vida.

Especialmente estuvieron presentes en la evolución y posterior desarrollo total de mi tesis, les agradezco con creces. Los quiero.

Alexander Quinaluiza

# ÍNDICE

## ÍNDICE DE TABLAS

## ÍNDICE DE FIGURAS

## RESUMEN EJECUTIVO

El tema del proyecto se centra en desarrollar una Interfaz de Programación de Aplicaciones(API) para la generación automática de Procedimientos Almacenados en MySQL. Los objetivos del proyecto son: ejecutar una metodología ágil que facilite el diseño de la API y de una aplicación gráfica que sirva para poner a prueba los aspectos funcionales de la misma y garantizar la calidad del proyecto propuesto.

En el proyecto se analizá la arquitectura de MySQL, con ese análisis se toman los requerimientos necesarios para el desarrollo de la API. Por consiguiente se realiza una comparativa sobre las metodologías ágiles más adaptables al proyecto, de las cuales se toman dos para la comparativa, estas son: OpenUP y Scrum. En el análisis de las metodologías se toma en cuenta ventajas y desventajas, a la vez un estudio bibliográfico de proyectos desarrollados con cada uno de ellas, que sean relacionados con el tema. Entonces una vez concluida la etapa de análisis de las metodologías, se escogió OpenUP como metodología de desarrollo para este proyecto.

La Interfaz de Programación de Aplicaciones para la generación automática de Procedimientos Almacenados en MySQL usa la metodología OpenUp. En este contexto, se desarrollan los siguientes documentos: Plan de desarrollo de software, Arquitectura de Software, Plan de Pruebas de Software, Instalación de la API, Estudio de aceptación de la API.

La API fue desarrollada utilizando las siguientes herramientas: MySQL Server 5.7, Entorno de Desarrollo Integrado(IDE) Microsoft Visual Studio en su versión Community, API de conexión de MySQL para .NET y C# como lenguaje de programación. Para el desarrollo de la aplicación de prueba se usaron las herramientas ya nombradas. Esta aplicación fue desarrollada en paralelo a la API, esta permite poner a prueba las funcionalidades de la misma.

## ABSTRACT

The theme of the project focuses on developing an Application Programming Interface (API) for the automatic generation of Stored Procedures in MySQL. The objectives of the project are: to execute an agile methodology that facilitates the design of the API and a graphic application that serves to test the functional aspects of the same and guarantee the quality of the proposed project.

In the project the architecture of MySQL is analyzed, with this analysis the necessary requirements for the development of the API are taken. Therefore a comparative is made on the agile methodologies more adaptable to the project, of which two are taken for the comparison, these are: OpenUP and Scrum. In the analysis of the methodologies, advantages and disadvantages are taken into account, at the same time a bibliographic study of projects developed with each of them, which are related to the subject. Then, once the methodological analysis stage was completed, OpenUP was chosen as the development methodology for this project.

The Application Programming Interface for the automatic generation of Procedures Stored in MySQL uses the OpenUp methodology. In this context, the following documents are developed: Software development plan, Software architecture, Software testing plan, API installation, API acceptance study.

The API was developed using the following tools: MySQL Server 5.7, Integrated Development Environment (IDE) Microsoft Visual Studio in its Community version, MySQL connection API for .NET and C # as a programming language. For the development of the test application, the aforementioned tools were used. This application was developed in parallel to the API, this allows you to test the functionalities of it.

# INTRODUCCIÓN

Al hablar de conectividad, de programas informáticos, de bases de datos y de desarrollo de software, se abre las puertas a la imaginación de miles de herramientas existentes en la actualidad. Estas facilitan el desarrollo de más aplicaciones que coexisten entre ellas, a ese tipo de software se lo llama API de forma abreviada o Interfaz de Programación de Aplicaciones. Es importante encontrar una metodología que facilite el desarrollo de un software de este tipo. El auge del uso de MySQL en el ámbito de bases de datos, inspiran a investigar y a crear una nueva herramienta que permita generar código SQL en un escenario donde millones de programdores necesitan disminuir el tiempo de desarrollo de las aplicaciones.

Entonces bajo el criterio de funcionalidad y de programación rápida, limpia e atendible, surge la idea del crear una API, una Interfaz de Programación de Aplicaciones para generar de forma automática procedimientos almacenados entendibles para el motor de base de datos MySQL. La API esta diseñada para generar los procedimientos almacenados de una manera rápida y fácil. La API está apoyando al programador a centrarse más tiempo en su creatividad y menos tiempo en traducir el lenguaje natural a maquina. Además que al usar procedimientos almacenados en el desarrollo de una aplicación cliente hace posible controlar de forma limpia y profesional el trabajo desarrollado.

Por esas razones se plantea este proyecto con objetivos de búsqueda e investigación de metodologías ágiles que se adapten de mejor manera al desarrollo de la idea planteada, más una aplicación gráfica diseñada para poner a prueba la API y su funcionalidad, todo hecho sobre un IDE de libre acceso y con una licencia GPL. En este contexto se contribuye con el mejoramiento continuo de este campo apasionante como lo es el desarrollo de software.

# CAPÍTULO 1

# El problema

## 1.1. Tema de Investigación

Interfaz de Programación de Aplicaciones para la generación automática de Procedimientos Almacenados en MySQL.

## 1.2. Planteamiento del problema

Durante las décadas de los 60 y 70 surge el concepto de las bases de datos; sin embargo, el objetivo principal siempre ha sido la administración óptima de la información y el uso que se le puede dar a la misma. Pero, en los últimos años, la velocidad y la facilidad en el diseño y desarrollo de una base de datos es un punto clave para un trabajo óptimo y eficaz dentro del mundo de las aplicaciones informáticas.

En este contexto empresas han desarrollado herramientas de Ingeniería de Software Asistida por Computadora (CASE)[1]. Empresas como Datamatic que brinda software privativo costoso para modelamiento de base de datos y generación de código automático para Lenguaje de Definición de Datos (DDL)[2]; vinculado con la mayoría de SGBD [3]. Además, los productos de Datamatic aun al ser costosos, no brinda una API (Interfaz de Programación de Aplicaciones) para facilitar el acceso de los recursos de la base de datos y desarrollo óptimo [4].

MySQL es el SGBD más usado para las aplicaciones basadas en la web, utilizada por Facebook, Twitter, LinkedIn, Yahoo!, Amazon Web Services y todas las empresas web más importantes que tuvieron inicios exitosos de manera virtual[5]. Este SGBD vinculado con Oracle posee productos como: conectores, herramientas de desarrollo, también el famoso workbrench, este último es un software administrador de base de datos diseñado para el entorno de MySQL. Sin embargo, workbrench no cuenta con un plugin o aplicación integrada para la generación automática de procedimientos almacenados. Esto lleva al desarrollador escribir código SQL para la creación de Procedimientos Almacenados, esto le lleva tiempo y esfuerzo.

La comunidad de desarrolladores busca una forma más rápida de agilizar los procesos de acceso y administración de recursos de una base de datos, desean

perder el menor tiempo en procesos tradicionales que deben quedar en la historia. El tiempo vale oro y mucho más para un desarrollador que debe entregar su producto de software en tiempos extremadamente cortos.

## 1.3. Delimitación

### 1.3.1. De contenidos

**Área Académica:** Software
**Línea de investigación:** Desarrollo de Software
**Sublíneas de investigación:** Orientación Objetos

### 1.3.2. Espacial

Esta API (Interfaz de Programación de Aplicaciones) no está delimitado para un área geográfica específica, el objetivo es publicar el proyecto en GitHub y contribuir con una API en el ambiente del desarrollo de software. En este contexto la delimitación espacial del proyecto es a nivel global.

### 1.3.3. Temporal

La presente investigación se desarrollará durante los 6 meses posteriores a la aprobación del proyecto por parte del Consejo Directivo.

## 1.4. Justificación

Con millones de bases de datos MySQL en producción en el mundo y miles de descargas diarias, no cabe ninguna duda de la importancia de este motor de bases de datos en el mercado [5, 6].
Los desarrolladores de aplicaciones informáticas, que integran MySQL como gestor de base de datos, se han incrementado en los últimos años. Esto hace que crezca la demanda de herramientas para facilitar el diseño, modelamiento y generación de código.
En este contexto se ve la necesidad en la comunidad de programadores de un software que genere Procedimientos Almacenados (SP) para los principales operadores DML (Lenguaje de Manipulación de Datos) que son: Select, Insert, Update, Delete [4]. Además esta API (Interfaz de Programación de Aplicaciones) puede integrarse a Microsoft Visual Studio u otro software dedicado al desarrollo de aplicaciones. Esta API fácilmente referenciará los procedimientos almacenados generados y los relacionará con sus parámetros.

El impacto del desarrollo de este proyecto es un ahorro de tiempo en el proceso de desarrollo, y también en la seguridad, ya que permitirá validar los datos que se ingresen a la base de datos y controlar los usuarios.
Además, minimiza el esfuerzo al escribir código y encontrar los recursos necesario de la base de datos en el desarrollo de aplicaciones informáticas. Entonces el programador puede dedicarse a otros procesos del desarrollo del software con más tiempo. Sin embargo, en cualquier etapa del desarrollo se debe tomar en cuenta la siguiente frase: "Él que no aplique nuevos remedios, debe esperar nuevos males, porque el tiempo es el máximo innovador"[7].

## 1.5. Objetivos

### 1.5.1. General

Desarrollar una Interfaz de programación de aplicaciones para la generación automática de Procedimientos Almacenados en MySQL.

### 1.5.2. Específicos

- Analizar la arquitectura de MySQL y sus componentes.
- Seleccionar una metodología ágil aplicada al campo del desarrollo de Interfaces de Programación de Aplicaciones.
- Diseñar una API (Interfaz de Programación de Aplicaciones) que se pueda usar desde un software de desarrollo como Microsoft Visual Studio.
- Implementar la API en una aplicación de ejemplo.

# CAPÍTULO 2

## Marco Teórico

### 2.1. Antecedentes Investigativos

Existen experiencias relacionadas con el desarrollo de API (Interfaz de Programación de Aplicaciones), entre las más destacables se tiene:
La Escuela de Ingeniería de Antioquia, Colombia, en el año 2013, cita en su investigación un diagrama conceptual para una herramienta que genere código automático y semiautomático basándose en lenguaje natural o esquemas conceptuales [8]. En la Figura(1), se muestra la síntesis del modelo conceptual de esta investigación.

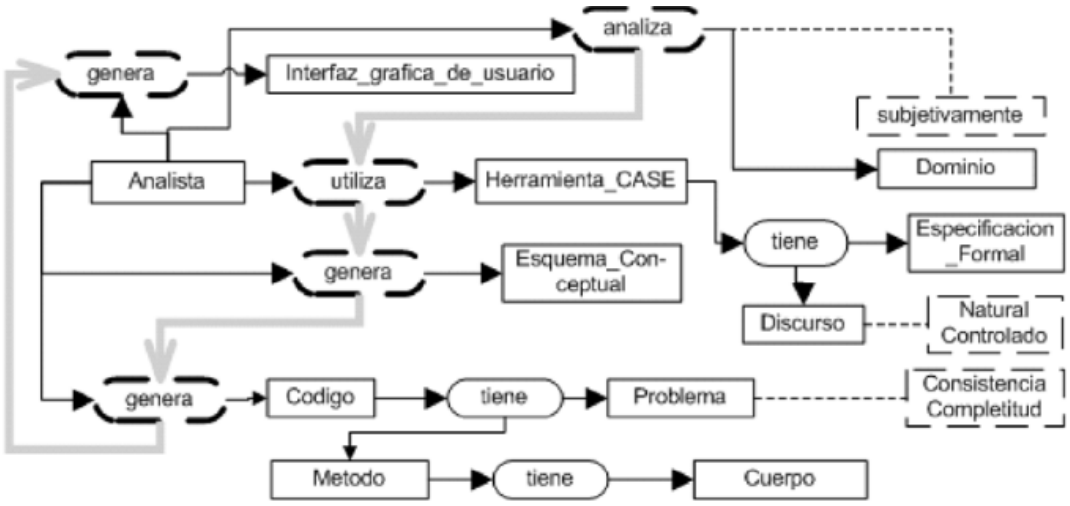



Figura 1: Diagrama conceptual para la generación automática de código.
Fuente: [8].

En la Universidad de Murcia, en el año 2014, el Departamento de Informática y Sistemas realizó su investigación sobre la interoperabilidad e herramientas basadas en MDE (Ingeniería de Software Dirigida por Modelos). Esta investigación presenta un estudio sobre la integración de la herramienta DB-Main y APIs para el desarrollo de una herramienta case para la interoperabilidad haciendo uso de modelos [9].
En el año 2014, la University of British Columbia, Canadá, en el departamento de Ciencias de la Computación se realizó una investigación sobre Integración y

virtualización de SQL Relacional y Sistemas NoSQL Incluyendo MySQL y MongoDB. Esta investigación presenta una arquitectura genérica basada en estándares que permiten los sistemas NoSQL, con un enfoque específico en MongoDB, que son consultados utilizando SQL y que interactúen sin problemas con cualquier software que soporte JDBC. Para ello, desarrollaron una API que se encargue de administrar las conexiones e integrar MongoDB y MySQL [10].

En la Universidad de Chulalongkorn, Bangkok, Tailandia, en el año 2015, en la Facultad de Ingeniería de dicha institución se desarrolló una API (Interfaz de Programación de Aplicación) para un Arduino basado en un sensor inalámbrico Mote. Con la API se busca minimizar la complejidad en el manejo del hardware a nivel de codificación del sensor inalámbrico [11].

En el año 2016, en la Habana, Cuba, la Universidad de las Ciencias Informáticas propone el desarrollo de una extensión de la herramienta Visual Paradigm for UML para la evaluación de Casos de Usos para futuros proyectos. Esta herramienta usa la biblioteca OpenAPI y la metodología de desarrollo OpenUP. Además eligieron como lenguaje de programación Java y el Entorno de Desarrollo Integrado NetBeans [12].

En el año 2017, Greenville, USA, la Universidad de Carolina del Este, centra su trabajo de investigación en los desarrolladores y forma de crean consultas, que luego se utilizan con la biblioteca original de la API de MySQL. Basado en una colección de Aplicaciones open-source PHP, logran analizar las consultas; las clasifican y las evalúan [13].

Finalmente revisando los proyectos de investigación realizadas en la Facultad de Ingeniería en Sistemas Electrónica e Industrial de la Universidad Técnica de Ambato no se encontraron investigaciones que tengan relación con el tema en estudio.

## 2.2. Fundamentación teórica

### 2.2.1. Concepto de API

API es una sigla de la lengua inglesa que apunta a la expresión Application Programming Interface (cuya traducción es Interfaz de Programación de Aplicaciones). Este concepto hace referencia a las funciones y métodos que brinda una determinada biblioteca de programación a modo de capa de abstracción, es decir, que pueda ser empleada por otro software [14].

Las API son un conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos con ciertas

funcionalidades específicas. Las API simplifican en gran medida el trabajo de un programador, ya que no tiene que escribir códigos desde cero. Estas permiten al informático usar funciones predefinidas para interactuar con el sistema operativo o con otro programa [15]. Un ejemplo de API es Java, esta se ejecuta en millones de dispositivos y programas que los usuarios comunes no saben qué hace, ni para qué sirve. EL usuario nunca ve las API en pleno proceso de trabajo, pero sí detalles de sus acciones.
La API es una interfaz que sólo otorga funcionalidades al software que lo emplea. Es decir, el usuario no ve eso. Con las API, plataformas como Twitter, Facebook, YouTube se pueden comunicar entre ellas sin que el usuario tenga que intervenir o incluso, percatarse [3, 15].

### 2.2.2. MySQL

MySQL es un sistema de gestión de base de datos relacional (RDBMS) de código abierto, basado en lenguaje de consulta estructurado (SQL) [16].
MySQL se ejecuta en prácticamente todas las plataformas, incluyendo Windows, UNIX y Linux. A pesar de que se puede utilizar en una amplia gama de aplicaciones, MySQL se asocia más con las aplicaciones basadas en la web, es un componente importante de una pila empresarial de código abierto llamado LAMP (Linux, Apache, MySQL, PHP).
MySQL ofrece la facilidad de uso, la escalabilidad y el alto rendimiento, así como un conjunto completo de controladores de base de datos y herramientas visuales para ayudar a los desarrolladores y DBA a construir y administrar sus aplicaciones de negocio críticas para MySQL. Este producto es desarrollado, distribuido y soportado por Oracle[16].

#### 2.2.2.1. Caracteristicas de MySQL

MySQL proporciona las siguientes características:

- Alto rendimiento y escalabilidad para satisfacer las demandas de cargas de datos y usuarios que crecen exponencialmente.
- Clústeres de replicación de auto-recuperación para mejorar la escalabilidad, el rendimiento y la disponibilidad.
- Cambio de esquema en línea para cumplir con los requisitos cambiantes del negocio.

- Esquema de rendimiento para supervisar el rendimiento de usuarios y aplicaciones y el consumo de recursos.
- SQL y NoSQL Access para realizar consultas complejas y operaciones de valor por claves simples y rápidas.
- Independencia de la plataforma que le da la flexibilidad para desarrollar e implementar en múltiples sistemas operativos.
- Interoperabilidad Big Data utilizando MySQL como almacén de datos operativos para Hadoop y Cassandra.

Estas características mencionadas se han agregado con apoyo de empresas de software privativo, y también con intervención de la comunidad de desarrollo de software libre [17].

### 2.2.3. Procedimientos Almacenados en MySQL

Un procedimiento almacenado es un conjunto de comandos SQL que pueden almacenarse en el servidor. Estos procedimientos almacenados una vez creados, los programadores de aplicaciones no necesitan escribir nuevamente el código SQL , en su lugar solo deben referirse al procedimiento almacenado ya creado para cumplir con un propósito o alguna regla de negocio establecida[18].
Algunas escenarios en que los procedimientos almacenados son particularmente útiles:

- Cuando varias aplicaciones cliente se desarrollan en distintos lenguajes o funcionan en distintas plataformas, pero necesitan realizar la misma operación sobre la base de datos.
- Al pensar siempre en seguridad. Por ejemplo, en los bancos usan procedimientos almacenados para todas las operaciones comunes. En este contexto, se proporciona un entorno seguro y consistente, de esta manera, los procedimientos pueden asegurar que, cada operación se realice de manera apropiada. En tal entorno, las aplicaciones no tendrán acceso a las tablas de forma directa, pero los programadores puede ejecutar procedimientos almacenados en sus aplicaciones.

Los procedimientos almacenados mejoran sustancialmente el rendimiento de las aplicaciones, ya que se necesita enviar menos información entre el servidor y el cliente. Entonces la carga de trabajo aumenta en el servidor de base de datos ya

que la mayoría del trabajo se realiza en el mismo[18].
Los procedimientos almacenados permiten tener bibliotecas en el servidor de base de datos. Esta característica es compartida por los lenguajes de programación modernos que tienen soporte de conexión con MySQL. Usando estas características del lenguaje de programación cliente es beneficioso para el programador incluso fuera del entorno de la base de datos[18].

#### 2.2.3.1. Sintaxis de Procedimientos Almacenados

Los procedimientos almacenados se crean con el comando CREATE PROCEDURE. Una rutina es un procedimiento o una función. Un procedimiento se invoca usando un comando CALL , y sólo puede pasar valores usando variables de salida[18]. Las rutinas almacenadas pueden llamar otras rutinas almacenadas. En la Figura (2) se expone la sintaxis de un procedimiento almacenado y de una función en MySQL Server.

```
CREATE PROCEDURE sp_name ([parameter[,...]])
    [characteristic ...] routine_body

CREATE FUNCTION sp_name ([parameter[,...]])
    RETURNS type
    [characteristic ...] routine_body

parameter:
    [ IN | OUT | INOUT ] param_name type

type:
    Any valid MySQL data type

characteristic:
    LANGUAGE SQL
  | [NOT] DETERMINISTIC
  | { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'

routine_body:
    procedimientos almacenados o comandos SQL válidos
```

Figura 2: Sintaxis de Rutinas en MySQL.
Fuente: [18].

Al crear una función se debe especificar la cláusula RETURNS, que es obligatorio.

Se usa para indicar el tipo de dato que retornará la función, y el cuerpo de la función debe contener un comando RETURN con un valor adjunto.
Al crear un procedimiento almacenado, la lista de parámetros entre paréntesis debe estar siempre presente. Si en el caso que el procedimiento no necesite parámetros, no se debe omitir los paréntesis () . Cada parámetro es un IN por defecto, es decir de entrada. Para especificar otro tipo de parámetro, use la palabra clave IN, OUT, o INOUT antes del nombre del parámetro [18].
Un procedimiento o función se considera "determinista" cuando genera el mismo resultado para los mismos parámetros de entrada, y "no determinista" en cualquier otro caso. MySQL establece por defecto NOT DETERMINISTIC[18].
Las siguientes características proporcionan información sobre la naturaleza de los datos en el procedimiento, tales como:

- CONTAINS SQL indica que la rutina no contiene comandos que leen o escriben datos.
- NO SQL indica que la rutina no contiene comandos SQL .
- READS SQL DATA indica que la rutina contiene comandos que leen datos, pero no comandos que escriben datos.
- MODIFIES SQL DATA indica que la rutina contiene comandos que pueden escribir datos.

CONTAINS SQL es el valor por defecto si no se dan explícitamente ninguna de estas características[18].
La característica SQL SECURITY puede usarse para especificar si la rutina debe ser ejecutada usando los permisos del usuario que crea la rutina o el usuario que la invoca. El valor por defecto es DEFINER .
La cláusula COMMENT es una extensión de MySQL, y puede usarse para describir el procedimiento almacenado. Esta información se muestra con los comandos SHOW CREATE PROCEDURE y SHOW CREATE FUNCTION respectivamente[18].
MySQL permite a las rutinas que contengan comandos DDL (tales como CREATE y DROP ), tambien comandos de transacción SQL (como COMMIT )[18].
Cabe recalcar que los procedimientos almacenados no se pueden usar el comando LOAD DATA INFILE.
El comando DELIMITER se utiliza para cambiar el delimitador del comando, de ; a // mientras se define el procedimiento . Esto permite pasar el delimitador

; usado en el cuerpo del procedimiento a través del servidor en lugar de ser interpretado por el mismo mysql[18].

#### 2.2.3.2. Cambiar las características de un procedimiento

En la Figura (3) puede usarse para cambiar las características de un procedimiento o función almacenada[18]. Se puede especificar, varios cambios en las características y en el cuerpo del procedimiento almacenado.

```
ALTER {PROCEDURE | FUNCTION} sp_name [characteristic ...]

characteristic:
    { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }
  | SQL SECURITY { DEFINER | INVOKER }
  | COMMENT 'string'
```

Figura 3: Sintaxis para editar un procedimiento almacenado.
Fuente: [18].

La cláusula IF EXISTS es una extensión de MySQL .Evita que ocurra un error si la función o procedimiento no existe. Se genera una advertencia que puede verse con el comando SHOW WARNINGS[18].

#### 2.2.3.3. Revisar el código SQL de un procedimiento

En la Figura (4) se expone el comando SHOW CREATE PROCEDURE que es una extensión de MySQL. Similar a SHOW CREATE TABLE, retorna la cadena exacta que puede usarse para recrear la rutina nombrada[18].

```
SHOW CREATE {PROCEDURE | FUNCTION} sp_name
```

Figura 4: Sintaxis para revisar el código SQL de un procedimiento.
Fuente: [18].

#### 2.2.3.4. Eliminar un procedimiento

En la Figura (5), se presenta la sintaxis del comando DROP PROCEDURE que se encarga de eliminar un procedimiento.

```
DROP {PROCEDURE | FUNCTION} [IF EXISTS] sp_name
```

Figura 5: Eliminar un procedimiento almacenado sintaxis.
Fuente: [18].

La cláusula IF EXISTS es una extensión de MySQL . Evita que ocurra un error si la función o procedimiento no existe. Se genera una advertencia que puede verse con el comando SHOW WARNINGS[18].

#### 2.2.3.5. Ejecución de un procedimiento

En la Figura (6) se expone la sintaxis del comando CALL que, ejecuta un procedimiento almacenado deseado.

```
CALL sp_name([parameter[,...]])
```

Figura 6: Sintaxis para ejecutar un procedimiento.
Fuente: [18].

El comando CALL puede pasar valores al llamador usando parámetros declarados como OUT o INOUT. En este contexto, si el procedimiento previamente creado tiene parámetros de entrada establecidos, se debe ingresar separados por una coma (,)[18].

#### 2.2.3.6. Sentencia compuesta BEGIN-END

En la Figura (7), se muestra la sintaxis de la sentencia BEGIN END. Esta se utiliza para escribir sentencias compuestas que se pueden aparecer en el interior de procedimientos almacenos y triggers. Una sentencia compuesta puede contener múltiples sentencias, estas sentencias están encerradas por las palabras claves BEGIN y END[18].

```
[etiqueta_inicio:] BEGIN
    [lista_sentencias]
END [etiqueta_fin]
```

Figura 7: Sintaxis de la sentencia BEGIN-END.
Fuente: [18].

Cada sentencia de la lista_sentencias debe terminar con un punto y coma (;). Esto se gestiona en el cliente de línea de comandos MySQL con el comando delimiter. Cambiar el delimitador de fin de sentencia ; (por ejemplo con //) permite utilizar ; en el cuerpo de una rutina[18].

#### 2.2.3.7. Declaración de variables locales

La Figura (8), expone la sintaxis del comando DECLARE para declarar variables locales. Esta sentencia define variables que se pueden usar dentro de la clausula

BEGIN-END. A la variable se le proporciona un valor por defecto con la cláusula DEFAULT. El valor puede especificarse como una expresión, no necesita ser una constante. Si la cláusula DEFAULT no está presente, el valor inicial es NULL[18].

```
DECLARE var_name[,...] type [DEFAULT value]
```

Figura 8: Declaración de variables locales.
Fuente: [18].

#### 2.2.3.8. Sentencia SET para variables

En los procedimientos almacenados el comando SET es una versión extendida del comando general SET.
Las variables referenciadas pueden ser las declaradas dentro de una rutina, o variables de servidor globales. EL comando SET asigna valores a las variables, permite la operación entre variables locales y globales[18]. En la Figura(9), se presenta la sintaxis del comando SET.

```
SET var_name = expr [, var_name = expr] ...
```

Figura 9: Sintaxis del Comando SET.
Fuente: [18].

#### 2.2.3.9. Sentencia SELECT-INTO

En la Figura(10), se expone la sintaxis de SELECT-INTO. Esta almacena columnas seleccionadas directamente en las variables locales. Es decir que, la consulta debe retorna un solo valor[18].

```
SELECT col_name[,...] INTO var_name[,...] table_expr
```

Figura 10: Sintaxis de la sentencia Select-Into.
Fuente: [18].

#### 2.2.3.10. Cursores

MySQL Server soporta cursores simples dentro de procedimientos almacenados. La sintaxis es la de SQL empotrado. Los cursores no son sensibles, son de sólo lectura, y no permiten deslizamiento (scrolling). Al decir que no es sensible, significa que el servidor puede o no hacer una copia de su tabla de resultados[18]. En la Figura(11), se presenta un ejemplo de un cursor simple.

```
CREATE PROCEDURE curdemo()
BEGIN
  DECLARE done INT DEFAULT 0;
  DECLARE a CHAR(16);
  DECLARE b,c INT;
  DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
  DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
  DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done = 1;

  OPEN cur1;
  OPEN cur2;

  REPEAT
    FETCH cur1 INTO a, b;
    FETCH cur2 INTO c;
    IF NOT done THEN
       IF b < c THEN
          INSERT INTO test.t3 VALUES (a,b);
       ELSE
          INSERT INTO test.t3 VALUES (a,c);
       END IF;
    END IF;
  UNTIL done END REPEAT;

  CLOSE cur1;
  CLOSE cur2;
END
```

Figura 11: Ejemplo de Cursores.
Fuente: [18].

Los cursores se beben declarar antes de declarar las variables y condicionales.

**Declarar cursores** En la Figura (12), se expone el comando que permite declarar un cursor en MySQL. Este comando puede declarar varios cursores en la rutina[18].

```
DECLARE cursor_name CURSOR FOR select_statement
```

Figura 12: Declaración de cursores.
Fuente: [18].

**Sentencia OPEN del cursor** En la Figura(13), presenta el comando que permite abrir un cursor declarado en el procedimiento almacenado[18].

```
OPEN cursor_name
```

Figura 13: Abrir un cursor.
Fuente: [18].

**Sentencia FETCH del cursor** La Figura(14), expone el comando FETCH,el mismo que trata el siguiente registro, siempre y cuando existan registros, usando el cursor abierto que se especifique y avanza el puntero del cursor[18].

```
FETCH cursor_name INTO var_name [, var_name] ...
```

Figura 14: Sintaxis sentencia FETCH cursores.
Fuente: [18].

Si no existen más registros disponibles, ocurrirá una condición de Sin Datos con el valor SQLSTATE 02000. En este caso, se puede configurar un manejador (handler) para detectar esta condición (o para una condición NOT FOUND)[18].

**Cerrar un cursor** El comando CLOSE cierra un cursor previamente abierto[18]. En la Figura(15), se muestra la sintaxis de dicho comando.

```
CLOSE cursor_name
```

Figura 15: Cerrar un Cursor.
Fuente: [18].

#### 2.2.3.11. Constructores de control de flujo

Los constructores IF, CASE, LOOP, WHILE, ITERATE, y LEAVE están completamente implementados. Estos constructores pueden contener un comando simple, o un bloque de comandos usando el comando compuesto BEGIN-END. Los constructores pueden estar anidados[18].

**Sentencia IF** La sentencia IF es constructor condicional básico. En la Figura(2.2.3.11), se presenta la sintaxis de la misma. En este contexto, search_condition se evalúa, si se cumple el condicional, el comando SQL correspondiente listado se ejecuta. Si no coincide ninguna search_condition se ejecuta el comando listado en la cláusula ELSE. statement_list puede consistir en varios comandos[18].

```
IF search_condition THEN statement_list
    [ELSEIF search_condition THEN statement_list] ...
    [ELSE statement_list]
END IF
```

Figura 16: Sintaxis sentencia IF.
Fuente: [18].

**Sentencia CASE** El comando CASE para procedimientos almacenados implementa un constructor condicional complejo. Si una search_condition se evalúa a cierto, el comando SQL correspondiente se ejecuta. Si no coincide ninguna condición de búsqueda, el comando en la cláusula ELSE se ejecuta[18]. En la Figura(17), se expone la sintaxis del comando ya nombrado.

```
CASE case_value
    WHEN when_value THEN statement_list
    [WHEN when_value THEN statement_list] ...
    [ELSE statement_list]
END CASE
```

Figura 17: Sintaxis de la sentencia CASE.
Fuente: [18].

**Sentencia LOOP** La sentencia LOOP implementa un constructor de bucle simple que permite la ejecución repetida de comandos particulares. El comando dentro del bucle se repite hasta que el mismo termine, que sucede usualmente con un comando LEAVE[18]. En la Figura(2.2.3.11), se expone la sintaxis de la sentencia LOOP.

```
[begin_label:] LOOP
    statement_list
END LOOP [end_label]
```

Figura 18: Sintaxis de la sentencia LOOP.
Fuente: [18].

Un comando LOOP puede etiquetarse. end_label no puede darse hasta que esté presente begin_label , y si ambos lo están, deben ser el mismo.

**Sentencia LEAVE** Este comando se usa para abandonar cualquier control de flujo etiquetado. Puede usarse con BEGIN-END o bucles[18]. En la Figura (19), se presenta la sintaxis del comando ya nombrado.

```
LEAVE label
```

Figura 19: Sintaxis del comando LEAVE.
Fuente: [18].

**Sentencia ITERATE** La sentencia ITERATE sólo puede usarse en comandos como: LOOP, REPEAT, y WHILE . ITERATE significa “vuelve a hacer el bucle”[18]. En la Figura (20), se muestra un ejemplo del uso de este comando.

```
CREATE PROCEDURE doiterate(p1 INT)
BEGIN
  label1: LOOP
    SET p1 = p1 + 1;
    IF p1 < 10 THEN ITERATE label1; END IF;
    LEAVE label1;
  END LOOP label1;
  SET @x = p1;
END
```

Figura 20: Sentencia ITERATE.
Fuente: [18].

**Sentencia REPEAT** Las sentencias que se encuentren dentro de un comando REPEAT se repite hasta que la condición search_condition sea verdadera. Este comando REPEAT puede etiquetarse. El end_label no puede darse a no ser que begin_label esté presente, y si lo están, deben ser el mismo[18]. La Figura (21) muestra la sintaxis del comando expuesto. Del mismo modo, en la Figura (22), se presenta un ejemplo del uso del comando.

```
[begin_label:] REPEAT
    statement_list
UNTIL search_condition
END REPEAT [end_label]
```

Figura 21: Sintaxis de la sentencia REPEAT.
Fuente: [18].

```
mysql> delimiter //

mysql> CREATE PROCEDURE dorepeat(p1 INT)
    -> BEGIN
    ->   SET @x = 0;
    ->   REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)

mysql> CALL dorepeat(1000)//
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @x//
+------+
| @x   |
+------+
| 1001 |
+------+
1 row in set (0.00 sec)
```

Figura 22: Ejemplo del comando REPEAT.
Fuente: [18].

**Sentencia WHILE** Todos las sentencias que se encuentren dentro de un comando WHILE se repite mientras la condición search_condition se cumpla. Este comando WHILE puede etiquetarse. end_label no puede darse a no ser que begin_label también esté presente, y si lo están, deben ser el mismo[18]. En la Figura(23), se muestra la sintaxis del comando expuesto. Del mismo modo, en la Figura, se expone un ejemplo del uso del comando.

```
[begin_label:] WHILE search_condition DO
    statement_list
END WHILE [end_label]
```

Figura 23: Sintaxis de WHILE.
Fuente: [18].

```
CREATE PROCEDURE dowhile()
BEGIN
  DECLARE v1 INT DEFAULT 5;

  WHILE v1 > 0 DO
    ...
    SET v1 = v1 - 1;
  END WHILE;
END
```

Figura 24: Ejemplo del comando WHILE.
Fuente: [18].

### 2.2.4. Tipos de datos MySQL

Los tipos de datos que puede haber en un campo, se pueden agrupar en tres grandes grupos: Numéricos, de fecha y de cadena.

#### 2.2.4.1. Tipo numéricos

Existen tipos de datos numéricos, que se describen en la Tabla (1).

| Tipo | Almacenamiento | Rangos |
|---|---|---|
| TINYINT | 1 byte | -128 a 127 y 0 a 255 |
| SMALLINT | 2 bytes | -32768 a 32767 y 0 a 65535 |
| MEDIUMINT | 3 bytes | -8.388.608 a 8.388.607 y 0 a16777215 |
| INT | 4 bytes | -2147483648 a 2147483647 y 0 a 429.4967.295 |
| INTEGER | 4 bytes | -2147483648 a 2147483647 y 0 a 429.4967.295 |
| BIGINT | 8 bytes | -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 y 0 a 18.446.744.073.709.551.615. |
| FLOAT(X) | 4 ú 8 bytes | -3.402823466E+38 a -1.175494351E-38 y 1.175494351E-38 a 3.402823466E+38 |
| FLOAT | 4 bytes | -3.402823466E+38 a -1.175494351E-38 y 1.175494351E-38 a 3.402823466E+38 |
| DOUBLE | 8 bytes | -1.7976931348623157E+308 a -2.2250738585072014E-308 y 2.2250738585072014E-308 a 1.7976931348623157E+308 |
| DOUBLE PRECISION | 8 bytes | - |
| REAL | 8 bytes | - |
| DECIMAL(M,D) | M+2 bytes sí $D > 0$, M+1 bytes sí $D = 0$ | - |
| NUMERIC(M,D) | M+2 bytes if $D > 0$, M+1 bytes if $D = 0$ | - |

Tabla 1: Tipos de datos numéricos.

Fuente: [19].

#### 2.2.4.2. Tipo fechas

Al hablar de fechas, hay que tener en cuenta que MySQL no comprueba de una manera estricta si una fecha es válida o no. Simplemente comprueba que los meses esta comprendidos entre 0 y 12 , en el caso de los días, estos están comprendido entre 0 y 31[19]. En la Tabla (2), se expone los tipos de datos para el manejo de fechas.

| Tipo | Almacenamiento | Descripción |
|---|---|---|
| DATE | 3 bytes | El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. |
| DATETIME | 8 bytes | El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. |
| TIMESTAMP | 4 bytes | El rango va desde el 1 de enero de 1970 al año 2037. |
| TIME | 3 bytes | Almacena la hora. |
| YEAR | 1 byte | Almacena el año. |

Tabla 2: Tipos de datos para fechas.

Fuente: [19].

#### 2.2.4.3. Tipo cadenas

MySQL cuenta con datos para el manejo de caracteres. En la Tabla (3), se citan varios tipos de datos encargados en el manejo de caracteres.

| Tipo | Almacenamiento |
|---|---|
| CHAR(n) | n bytes |
| VARCHAR(n) | n +1 bytes |
| TINYBLOB, TINYTEXT | Longitud+1 bytes |
| BLOB, TEXT | Longitud +2 bytes |
| MEDIUMBLOB, MEDIUMTEXT | Longitud +3 bytes |
| LONGBLOB, LONGTEXT | Longitud +4 bytes |
| ENUM('value1','value2',...) | 1 ó dos bytes dependiendo del número de valores |
| SET('value1','value2',...) | 1, 2, 3, 4 ó 8 bytes, dependiendo del número de valores |

Tabla 3: Tipos de datos para caracteres.

Fuente: [19].

### 2.2.5. Operadores en MySQL

Los operadores en MySQL son expresiones que pueden usarse en varios puntos de los comandos SQL, tales como en las cláusulas ORDER BY o HAVING de los comandos SELECT , en la cláusula WHERE de los comandos SELECT, DELETE, INSERT, o UPDATE o en comandos, SET . Las expresiones pueden escribirse usando valores literales, valores de columnas, NULL, funciones[18]. En la Figura (25),se extiende una lista de operadores por su precedencia.

```
:=
||, OR, XOR
&&, AND
NOT
BETWEEN, CASE, WHEN, THEN, ELSE
=, <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
|
&
<<, >>
-, +
*, /, DIV, %, MOD
^
- (resta unaria), ~ (inversión de bit unaria)
!
BINARY, COLLATE
```

Figura 25: Lista de operadores.
Fuente: [18].

### 2.2.6. Funciones en MySQL

MySQL cuenta con un gran número de funciones que facilitan el trabajo al programador. En la Tabla4 se presenta un resumen con las funciones dividas por grupos específicos de aplicación.

| Grupo | Descripción | Funciones Ejemplo | Total de Funciones |
|---|---|---|---|
| Matemáticas | Grupo de funciones encargadas en el manejo de expresiones numéricas. | ABS, SIN, ASIN | 36 |
| Cadena | Funciones que trabajan con cadenas de texto y permiten su manipulación. | TRIM, CONCAT | 60 |
| Fechas | Estas funciones permiten trabajar con expresiones referentes al tiempo, es decir con: Fechas, meses, días, años, horas. | ADDDATE, MONTH | 83 |
| Control de flujo | Funciones condicionales que controlan el flujo de las consultas | IF, IFNULL, CASE | 5 |
| Funciones de conversión | Dos funciones encargadas de realizar conversiones explicitas de expresiones. | CAST Y CONVERT | 2 |
| Funciones de encriptado | Un grupo de funciones especializadas en la encriptación y desencriptación de contraseñas usando diferentes algoritmos. | MD5, SHA1, SHA | 12 |
| Funciones de información | Las funciones pertenecientes a este grupo retornar información sobre el sistema. | VERSION, USER | 13 |
| Funciones generales | Grupo de funciones para verificar y establecer información de manera general. | DEFAUT, FORMAT | 10 |
| Funciones de grupos | Se aplica a un grupo de registros especifico, retorna un solo valor. | MIN, MAX, SUM | 12 |

Tabla 4: Funciones en MySQL.

Fuente: [20, 18].

### 2.2.7. Arquitectura de MySQL

En la Figura (26) se muestra la arquitectura de MySQL Server con cada uno de sus componentes divididos en capas.

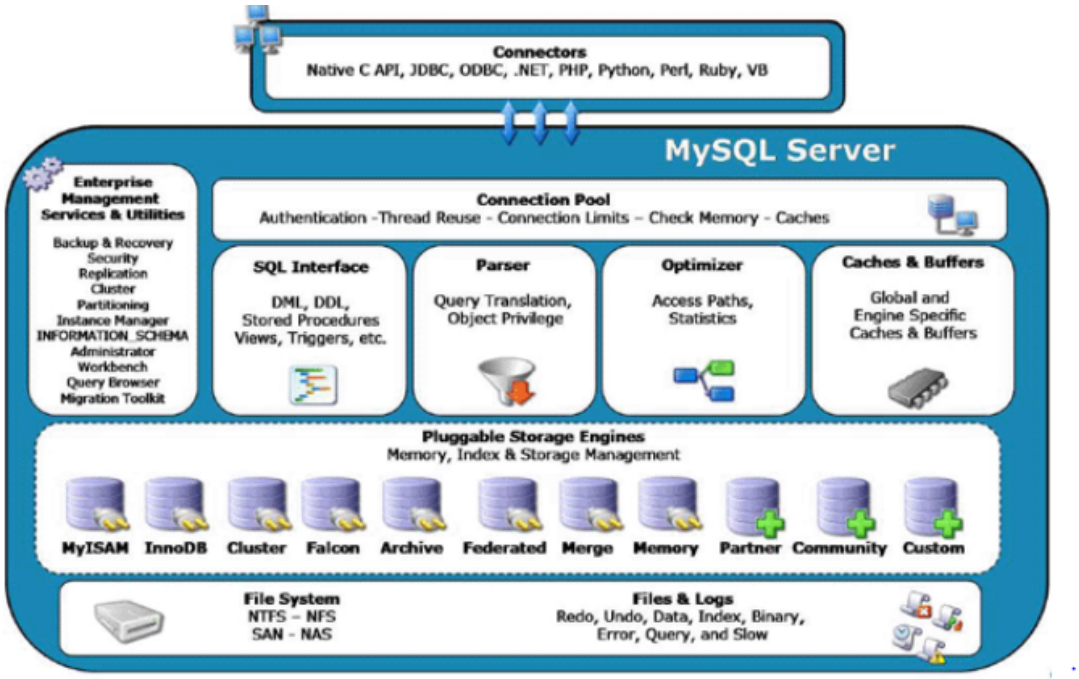



Figura 26: Arquitectura de MySQL Server.
Fuente: [21].

Las herramientas y servicios de MySQL Server son los programas que se incluyen con la distribución del gestor, o que pueden instalarse como aplicaciones. Estas incluyen las herramientas de replicación, seguridad, particionamiento, ademas del navegador de consultas (QueryBrowser) y la aplicación administrativa de MySQL Workbench, entre otras[22].

#### 2.2.7.1. Motores de almacenamiento

Los motores de almacenamiento son los componentes del servidor de bases de datos que llevan a cabo acciones en los datos[21]. La idea de esa arquitectura es hacer una interfaz abstracta con funciones comunes de gestión de datos en el nivel físico. De ese modo, el gestor de almacenamiento puede intercambiarse, e incluso un mismo servidor MySQL puede utilizar diferentes motores de almacenamiento para diferentes bases de datos o para diferentes tablas en la misma base de datos. Esto permite utilizar el motor de almacenamiento más adecuado para cada necesidad concreta[22]. En la tabla (5) lista los motores de almacenamiento

disponibles de MySQL Server. En este contexto TR hace referencia al soporte de transacciones, XA hace referencia al soporte de la interfaz de transacciones en MySQL y SP que se refiere al soporte de un punto de guardado de transacción. Esta tabla se puede revisar con la siguiente sentencia sql "*show engines*" en MySQL.

| **Motores** | **Soporte** | **Comentario** | **TR** | **XA** | **SP** |
|---|---|---|---|---|---|
| InnoDB | SI | Soporta transacciones, bloqueo a nivel de fila y claves externas. | SI | SI | SI |
| MRG_MYISAM | SI | Colección de tablas MyISAM idénticas | NO | NO | NO |
| MEMORY | SI | Hash basado, almacenado en la memoria, útil para las tablas temporales | NO | NO | NO |
| BLACKHOLE | SI | /dev/null motor de almacenamiento nulo (cualquier cosa que escriba a él desaparece) | NO | NO | NO |
| MyISAM | SI | Motor de almacenamiento MyISAM | NO | NO | NO |
| CSV | SI | Motor de almacenamiento CSV | NO | NO | NO |
| ARCHIVE | SI | Motor de almacenamiento Archive | NO | NO | NO |
| PERFORMANCE_SCHEMA | SI | Esquema de rendimiento | NO | NO | NO |
| FEDERATED | NO | Motor de almacenamiento de MySQL federado | - | - | - |

Tabla 5: Motores de Almacenamiento de MySQL.

#### 2.2.7.2. Conectores

Los conectores son bibliotecas en diferentes lenguajes de programación que permiten la conexión (remota o local) con servidores MySQL y la ejecución de consultas. Para este proyecto se usa el conector para aplicaciones desarrolladas en .NET. Este conector es una API que administra la interacción de las aplicaciones clientes con servidores MySQL[23].

#### 2.2.7.3. Procesamiento y Optimización de consultas

Cada vez que una consulta llega al gestor de MySQL, se analiza sintácticamente y se produce una representación intermedia de la misma. A partir de esa representación, MySQL toma una serie de decisiones, que pueden incluir el determinar el orden de lectura de las tablas, el uso de ciertos índices, o la re-escritura de la consulta en una forma más eficiente.
En este ámbito se puede utilizar sentencias en las consultas para ayudar al optimizador de una manera más eficiente, o bien se solicita al servidor información sobre cómo ha planificado las consultas. Esto permite entender el funcionamiento de las consultas que se ejecuten en el gestor.
La optimización de las consultas depende directamente del motor de almacenamiento que se elige, el optimizador consulta al gestor si soporta ciertas características que se desea ejecutar, de este modo, puede decidir y ejecutar sentencias mas optimas[22].

#### 2.2.7.4. Caché de consultas

MySQL entre sus componentes o funciones tiene la de implementar una caché de consultas, donde guarda consultas y sus resultados. De esta manera, el procesador de dichas consultas, antes de planificar la optimización, busca la consulta en la caché, y recupera la información mucho más rápida que la primera vez que se ejecuto la consulta[21, 22].

#### 2.2.7.5. Control de Concurrencia

El control de concurrencia en un gestor de bases de datos es el mecanismo que se utiliza para evitar que lecturas o escrituras simultáneas a un mismo dato o conjuntos de datos. Este componente utiliza un mecanismo para controlar este acceso, son los bloqueos (Locks). La idea es que, cada vez que una aplicación quiera acceder a un conjunto de datos, se establezca un bloqueo sobre los mismos. En este contexto, varias aplicaciones que quieran ejecutar acciones simultáneas, no tendrían ningún problema en hacerlo, ya que para la lectura se proporcionan bloqueos compartidos o (shared locks). Sin embargo, cuando la o las aplicaciones escriben sobre la base de datos de manera simultánea los lectores pueden tener problemas. Por esa razón, MySQL proporciona bloqueos exclusivos o (exclusive locks) para la escritura de datos sobre el gestor. El manejo simultaneo de datos puede dar paso a inconsistencias o efectos no deseados y este componente busca evitar problemas de dicha índole[22].

#### 2.2.7.6. Gestión de transacciones y recuperación

La gestión de transacciones permite dotar de semántica "todo o nada" a una consulta o a un conjunto de consultas que se declaran como una sola transacción. Es decir, si hay algún problema y parte de la consulta o algunas de las consultas no consiguen llevarse a cabo, el servidor anulará el efecto parcial de la parte que ya haya sido ejecutada. La recuperación permite "volver hacia atrás" (rollback) partes de una transacción[22].

#### 2.2.7.7. Análisis de la Arquitectura de MySQL

Una vez documentado cada uno de los componentes que intervienen en la Arquitectura de MySQL, se observa que es necesario tener en cuenta los siguientes aspectos para el desarrollo de la Interfaz de Programación de Aplicación (API) para la generación automática de procedimientos almacenados en MySQL.
El primer aspecto es establecer un conector, este es necesario para la comunicación con la base de datos. Este es una API que permite hacer uso de servicios y recursos de una base de datos especifica.El segundo aspecto es el Connection Pool encargada de administrar los usuarios y el proceso de autenticidad para el acceso a una base de datos. Como siguiente punto es el manejo de servicios; necesarios para desarrollar la API, es el acceso al esquema publico INFORMATION_SCHEMA, este esquema tiene alojados todos los metadatos de MySQL.
El componente SQL Interface, procesa las consultas que se desea ejecutar, en este caso son todas las posibles dentro de un Procedimiento Almacenado. Siguiendo la linea del proceso esta el componente Parser, este se encarga del control de concurrencia, es decir que evita la lectura y escritura de datos sobre una tabla de forma simultanea.
El optimizador y el cache de consultas son dos componentes relacionados. El primero antes de planificar la ejecución de una consulta, este busca si dicha consulta ya fue efectuada, la busca en memoria cache. Si la encuentra utiliza esa información. En cache se guarda las consultas sus resultados, esto evita el doble trabajo por parte del gestor.
Por ultimo tenemos los motores de almacenamiento de base de datos, estos almacenan las bases de datos. En la arquitectura serán usados al momento de generar un Procedimiento Almacenado. En el caso de el motor de almacenamiento de las bases de datos, es importante saber que se debe usar INNODB. Este soporta bases de datos relacionales, y la API esta pensada para este tipo de base de datos ya que son las más complejas de desarrollar. Sin embargo una base de datos puede tener una o varias tablas en diferentes motores de almacenamiento. Entonces

todos los trabajos efectuados sobre las bases de datos son registrados en ficheros Redo, Error, etc. Estos son almacenados en el sistema de archivos del sistema operativo en donde fue instalado MySQL.

### 2.2.8. Metodologías de Desarrollo de Software

Las principales diferencias de una Metodología Ágil respecto de las Metodologías Tradicionales se presentan en la Tabla (6) , diferencias que no se refieren sólo a los procesos, sino también al contexto de equipo y organización, que es más favorable a cada uno de estas filosofías de desarrollo de software[24].

| **Metodología Ágil** | **Metodología Tradicional** |
|---|---|
| Pocos Artefactos. El modelado es prescindible, modelos desechables. | Más Artefactos. El modelado es esencial, mantenimiento de modelos. |
| Pocos Roles, más genéricos y flexibles. | Más Roles, más específicos. |
| No existe un contrato tradicional, debe ser bastante flexible. | Existe un contrato prefijado. |
| Cliente es parte del equipo de desarrollo. | El cliente interactúa con el equipo de desarrollo mediante reuniones. |
| Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños (< 10 integrantes) y trabajando en el mismo sitio. | Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos. |
| La arquitectura se va definiendo y mejorando a lo largo del proyecto. | Se promueve que la arquitectura se defina tempranamente en el proyecto. |
| Énfasis en los aspectos humanos: el individuo y el trabajo en equipo. | Énfasis en la definición del proceso: roles, actividades y artefactos. |
| Se esperan cambios durante el proyecto. | Se espera que no ocurran cambios de gran impacto durante el proyecto. |

Tabla 6: Diferencias entre metodologías ágiles y no ágiles.

Fuente: [24].

El desarrollo ágil de software se refiere a métodos de Ingeniería del Software basados en el desarrollo iterativo e incremental, estas metodologías son

imprescindibles en un mundo en el que los desarrolladores se exponen a cambios con frecuencia. Siempre hay que tener en cuenta como programadores que, lo que es última tendencia hoy, puede que no exista mañana y por esto existe la metodología ágil donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto organizados y multidisciplinarios[25]. Sin embargo, las metodologías tradicionales se usan hoy en día para proyectos a largo plazo, como desarrollo de sistemas operativos y semejantes. En la Tabla (7) se presentan ejemplos de metodologías.

| **Metodología Ágil** | **Metodología Tradicional** |
|---|---|
| Desarrollo de Software Adaptable (ASD) | Prototipato |
| OpenUP | Espiral |
| Scrum | Cascada |

Tabla 7: Ejemplos de metodologías.

#### 2.2.8.1. Metodologías ágiles para el desarrollo de una API

Para escoger una metodología ágil para el desarrollo de este proyecto, se realiza una comparativa de dos metodologias, esas son: Scrum y OpenUP. Estas fueron escogidas para realizar un análisis que facilite la selección de una de ellas. En la tabla (8) se exponen características sobre cada una.

| **Scrum** | **OpenUP** |
|---|---|
| Es apropiado para proyectos con equipos competitivos productivos. | Es apropiado para proyectos pequeños y de bajos recursos. |
| Es iterativo con tiempos de incrementos fijos. | Permite detectar errores tempranos a través de un ciclo iterativo |
| Demostración de los resultados de proyecto en cada iteración. | Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP. |
| Gestión regular de las expectativas del cliente. | Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas. |
| Entregales funcionales cada 30 días. | Los avances del proyecto se realizan con micro-incrementos. |

Tabla 8: Comparación entre Scrum y OpenUP.

Scrum es una metodología de desarrollo de software ágil que aplica de manera regular un conjunto de prácticas para trabajar en equipo, y obtener el mejor resultado posible en un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos [26]. En este contexto, la metodología no contempla un registro de trabajos realizados en el desarrollo de Interfaces de Programación de Aplicaciones (API).
Existe experiencias de desarrollo con la metodología OpenUP en proyectos relacionados. Estos son citados en la sección de 2.1 Antecedentes Investigativos. Al existir proyectos relacionados con el tema, facilita la selección y uso de la metodología. Ademas que la misma propone la generación de documentos estrictamente necesarios para el desarrollo del proyecto. Por estas razones se escogió OpenUp como metodología de desarrollo de una Interfaz de Programación de Aplicación para la generación automática de Procedimientos Almacenados en MySQL.

#### 2.2.8.2. OpenUP

OpenUP es un Proceso Unificado que aplica enfoques iterativos e incrementales dentro de un ciclo de vida estructurado, utiliza una filosofía ágil que se enfoca en la naturaleza de colaboración en el desarrollo de software. Es una herramienta que puede extenderse para hacer frente a una amplia variedad de proyectos. Está basado en casos de uso, la gestión de riesgos, y una arquitectura centrada a impulsar el desarrollo[21].

**Elementos** OpenUP se organiza en dos dimensiones: Contenido metodológico y contenido procedimental. El contenido metodológico es el que define elementos metodológicos, tales como: Disciplinas, tareas, artefactos y procesos, independientemente de como se usen estos o se combinen. El contenido procedimental, por el contrario, es donde se aplican todos estos elementos metodológicos dentro de una dimensión temporal, pudiéndose crear multitud de ciclos de vida diferentes a partir del mismo subconjunto de elementos metodológicos[21]. En la Figura (27), se presentan los elementos que son parte de metodología que se usara en este proyecto.

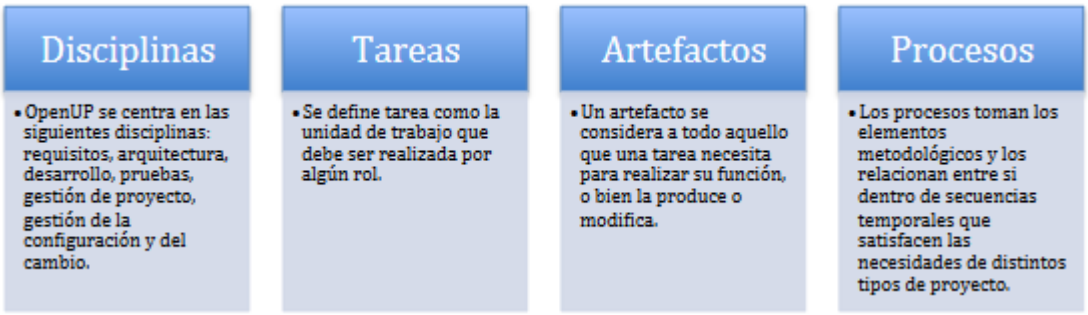



Figura 27: Elementos de OpenUP.
Fuente: [21].

**Ciclo de vida** En todo proyecto que se desarrolle con la metodología OpenUP consta de cuatro fases, que son: Inicio, elaboración, construcción y transición. Cada una de estas se divide a su vez en iteraciones[21]. En la Figura (28), se expone las diferentes fases y las relaciones entre las mismas.

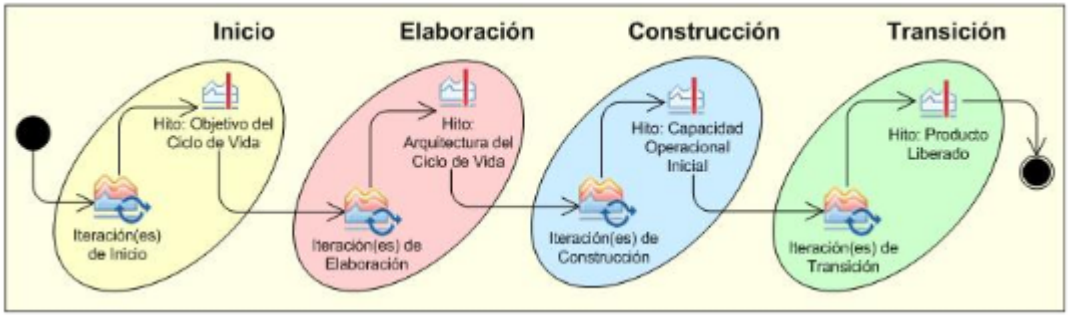



Figura 28: Ciclo de vida en OpenUP.
Fuente: [21].

- **Fase de inicio:** En esta fase, las necesidades de cada participante del proyecto son tomadas en cuenta y plasmadas en objetivos del proyecto. Se definen para el proyecto: el ámbito, los límites , el criterio de aceptación, los casos de uso críticos, una estimación inicial del coste y un boceto de la planificación.

- **Fase de elaboración:** En esta fase se realizan tareas de análisis del dominio y definición de la arquitectura del sistema. Se debe elaborar un plan de proyecto, estableciendo unos requisitos y una arquitectura estables. Por otro lado , el proceso de desarrollo, las herramientas, la infraestructura a utilizar y el entorno de desarrollo también se especifican en detalle en esta fase. Al final de la fase se debe tener una definición clara y precisa de los casos de

uso, los actores, la arquitectura del sistema y un prototipo ejecutable de la misma.

- **Fase de construcción:** Todos los componentes y funcionalidades del sistema que falten por implementar son realizados, probados e integrados en esta fase. Los resultados obtenidos en forma de incrementos ejecutables deben ser desarrollados de la forma más rápida posible sin dejar de lado la calidad de lo desarrollado.

- **Fase de transición:** Esta fase corresponde a la introducción del producto en la comunidad de usuarios, cuando el producto está lo suficientemente maduro. La fase de la transición consta de las subfases de pruebas de versiones, y capacitación de los usuarios finales. En función de la respuesta obtenida por los usuarios puede ser necesario realizar cambios en las entregas finales o implementar alguna funcionalidad más[21].

**Practicas** OpenUP es una metodología basada en RUP (Proceso Unificado de Desarrollo), y por lo tanto, comparte las mismas prácticas que subyacen por debajo del flujo de trabajo y los roles de OpenUP[21]. En la Figura (29), se presentan las practicas de trabajo de esta metodología.

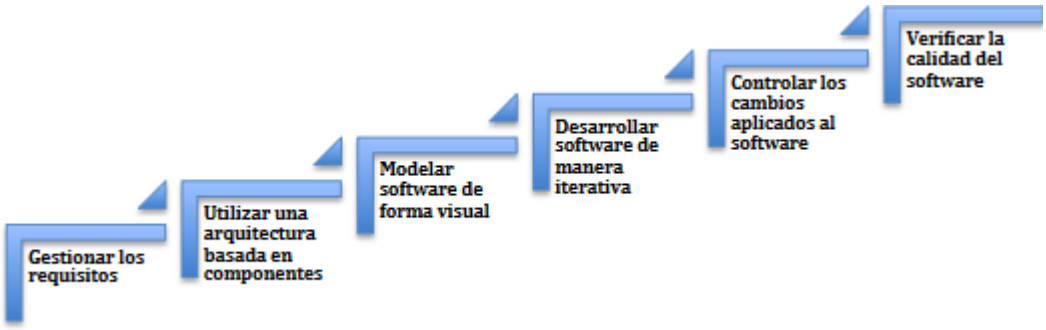



Figura 29: Practicas de OpenUP.
Fuente: [21].

### 2.2.9. Lenguajes de programación Orientado a Objetos

Entre todos los lenguajes de programación orientado a objetos, se nombra a los más importantes y afines con el desarrollo de este proyecto. En este contexto la Tabla2.2.9 expone una tabla comparativa entre C#, Java y Python.

| TABLA COMPARATIVA | | | |
|---|---|---|---|
| **LENGUAJES** | **CARACTERÍSTICAS** | **FORTALEZAS** | **DEBILIDADES** |
| C# | Esta estandarizado por Microsoft como parte de su plataforma net. | Se desempeña de forma plena en los sistemas operativos Windows.<br>Sintaxis más en comparación con C y C++.<br>Posibilidad de realizar aplicaciones web, de escritorio y móviles. | Requiere un mínimo de 4 GB para su instalación |
| JAVA | Multiplataforma. | Al ser orientado a objetos permite su modulación.<br>Permite la creación de aplicaciones de escritorio Tiene soporte a desarrollo de aplicaciones móviles y web. | Es un lenguaje interpretado así que es relativamente lento en comparación con otros lenguajes |
| PYTHON | Permite la creación de todo tipo de programas incluso sitios web.<br>No requiere de compilación es un código interpretado | Libre y código fuente abierto.<br>Lenguaje de propósito general.<br>Multiplataforma.<br>Orientado a objetos.<br>Portable. | Los lenguajes interpretados suelen ser relativamente lentos. |

Tabla 9: Comparativa entre lenguajes de Programación.

Fuente: [27].

MySQL brinda un conector para la plataforma de desarrollo .Net. Esta característica es importante porque facilita el desarrollo de una aplicación ejemplo haciendo uso del lenguaje C# en un entorno integrado de desarrollo (IDE) como Microsoft Visual Studio Community.

El IDE de desarrollo Microsoft Visual Studio en su versión Community permite desarrollar proyectos Open Source. El mismo integra herramientas fáciles de usar en el desarrollo, además de documentación sobre MySQL y su integración en el

entorno de desarrollo.
En este contexto, para el desarrollo de este proyecto se escogió como lenguaje de programación a C# por las características antes nombradas y basadas en las ventajas presentadas en la Tabla 2.2.9.

# CAPÍTULO 3

# Metodología

## 3.1. Modalidad Básica de la investigación

El presente trabajo tiene las siguientes modalidades de investigación:
**Modalidad Bibliográfica o Documentada** Se considera esta modalidad ya que se recurre a diferentes fuentes obtenidas de libros, artículos científicos, tesis desarrolladas en Universidades para profundizar enfoques con respecto al tema de la investigación.
**Modalidad aplicada** Por la utilización de los conocimientos adquiridos a lo largo de la carrera universitaria.

## 3.2. Recolección de información

Para la recolección de información se utilizará la técnica de investigación documental de tipo informativa (expositiva), se buscará información relevante de diferentes fuentes confiables, la información analizada y sintetizada servirá de apoyo para el desarrollo de este trabajo.

## 3.3. Procesamiento y análisis de datos

Para el procesamiento de la información se realizará las siguientes actividades:

- Recolección de datos mediante investigación documental.
- Revisión y análisis de la información recogida.
- Interpretación de los resultados mediante gráficos y cuadros informativos.

## 3.4. Desarrollo del Proyecto

Para el desarrollo de este proyecto son necesarias las siguientes actividades:

| Nivel | Actividades |
|---|---|
| 1 | Análisis de la arquitectura de MySQL. |
| 1.1 | Estudio de los componentes de MySQL. |
| 2 | Estudio bibliográfico de metodologías agiles de desarrollo de software. |
| 2.1 | Estudio bibliográfico de lenguajes de programación orientado a objetos. |
| 3 | Diseño de un esquema de funcionamiento de la API, orientado a objetos. |
| 3.1 | Priorizar los objetos, módulos de programación de la API |
| 3.2 | Desarrollo de los construibles. |
| 3.3 | Pruebas de cada construible. |
| 4 | Integración de los construibles. |
| 4.1 | Creación del manual del programador de la API. |
| 5 | Diseño de una GUI ejemplo con la implementación de la API. |
| 5.1 | Desarrollo de la GUI de ejemplo. |
| 5.2 | Publicación de la API en GitHub. |

Tabla 10: Actividades del Proyecto de Investigación.

# CAPÍTULO 4

## Desarrollo de la propuesta

### 4.1. Plan de Desarrollo de Software

#### 4.1.1. Introducción

##### 4.1.1.1. Propósito

El propósito de este documento es describir las actividades que se desarrollarán en la fase de elaboración y de la fase de construcción de la Interfaz de Programación de Aplicaciones para la generación automática de Procedimientos Almacenados en MySQL.

##### 4.1.1.2. Alcance

En este documento se especifica de forma detallada, las actividades a realizar para alcanzar lo descrito como objetivos de este proyecto de investigación. En este contexto el desarrollo de una API para la generación automática de Procedimientos Almacenados en MySQL. Para ello, se describe un flujo de trabajo a seguir para la implementación de los casos de uso.

##### 4.1.1.3. Definiciones, Acrónimos y Abreviaciones

- **API:** abreviatura de Interfaz de Programación de Aplicaciones.
- **Casos de Uso:** formato que proporciona escenarios que indican la interacción del usuario con el software o con otros componentes necesarios para conseguir un objetivo especifico.
- **Iteración:** Iteración significa el acto de repetir un proceso con la intención de alcanzar una meta deseada.
- **OpenUP:** metodología de desarrollo de software.
- **Recurso:** elementos disponibles para el desarrollo de un proyecto.
- **Usuario:** persona que utilizará el software, en este caso, un programador o desarrollador de software.

### 4.1.2. Resumen Plan del Proyecto OpenUP

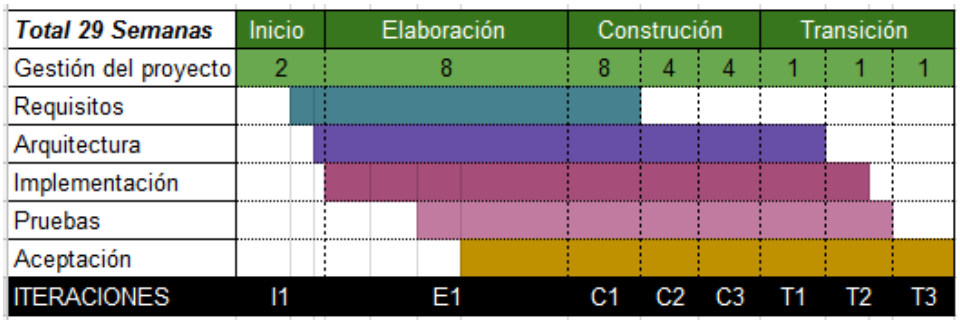



Figura 30: Resumen del Plan del Proyecto.

### 4.1.3. Productos del proyecto

| **Actividad** | **Iteración** |
|---|---|
| Plan de desarrollo de Software | 1 |
| Especificación de los requisitos del Software | 3 |
| Documento de arquitectura del Software | 1 |
| Definición de pruebas para el Software | 1 |
| Desarrollo del primer prototipo | 3 |
| Desarrollo de pruebas | 3 |
| Primer prototipo funcional | 3 |
| Refinamiento de los requerimientos funcionales | 3 |
| Refinamiento de las pruebas | 3 |
| Segundo prototipo funcional/(API completa) | 3 |
| Estudio de Aceptabilidad | 1 |

Tabla 11: Productos del proyecto.

#### 4.1.3.1. Organización del proyecto

La estructura organizacional del proyecto incluye a todos los participantes que se estimen necesarios para proporcionar información, colaboración con el desarrollo y validación del proyecto. El personal del proyecto estará formado por los siguientes personas y sus puestos de trabajo asociados.

| | |
|---|---|
| **Nombre** | Alexander Quinaluiza |
| **Rol** | Desarrollador del Proyecto de Investigación |
| **Información de contacto** | alexanderq@hotmail.com.ar |

Tabla 12: Desarrollador del proyecto de investigación.

| | |
|---|---|
| **Nombre** | Edison Álvarez |
| **Rol** | Tutor del Proyecto de Investigación |
| **Información de contacto** | ealvarez@uta.edu.ec |

Tabla 13: Tutor del proyecto de investigación.

#### 4.1.3.2. Estimación del proyecto

Los costos generados por el proyecto están detallados en la propuesta del mismo. No obstante para estimar el costo, esfuerzo y el tiempo que llevará el desarrollo del software se usó la metodología de estimación por puntos de función. En el siguiente gráfico se detalla el cálculo de los Puntos de Función Sin Ajustar (PFSA).

| | Valor optimista | Valor probable | Valor pesimista | Estimado (op+4*pr+pe) /6 | Complejidad | Valor de complejidad | Total PF |
|---|---|---|---|---|---|---|---|
| Número de entradas | 10 | 15 | 20 | 15 | simple | 3 | 45 |
| Número de salidas | 11 | 17 | 21 | 16,67 | media | 5 | 83,33 |
| Número de consultas | 25 | 30 | 35 | 30 | simple | 4 | 120 |
| Ficheros lógico | 2 | 4 | 7 | 4,17 | simple | 7 | 29,17 |
| Ficheros interfaz(externos) | 9 | 15 | 18 | 14,5 | media | 5 | 72,5 |
| Total Puntos de Función Sin Ajustar (PFSA) | | | | | | | 350 |

Figura 31: Puntos de Función Sin Ajustar (PFSA).

Para el cálculo de los Puntos de Función Ajustados (PFA) se debe tomar en cuenta una serie de factores de ajuste de complejidad. En la siguiente figura se muestra el Total de los PFA.

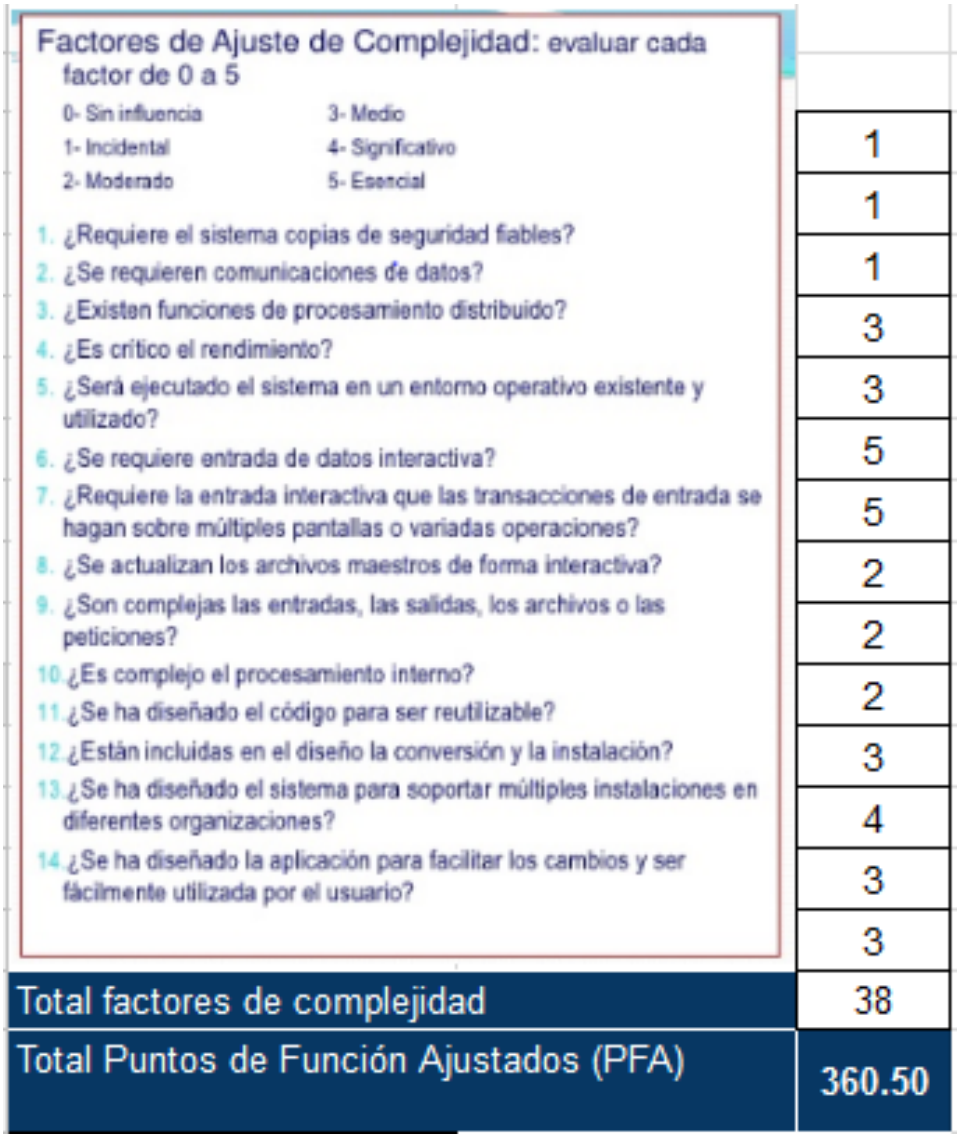



Figura 32: Puntos de Función Sin Ajustar (PFSA).

En base al total de los PFA se realizan los cálculos de estimación del costo, esfuerzo y duración del proyecto. En la siguiente figura se detallan los resultados obtenidos para el desarrollo del software.

| Lenguaje | Horas PF promedio | Lineas de código por PF |
|---|---|---|
| Ensamblador | 25 | 300 |
| COBOL | 15 | 100 |
| Lenguajes 4ta Generación | 8 | 20 |

| Datos | | Resultados de los cálculos | |
|---|---|---|---|
| Personas | 1 | PFA | 360,50 |
| Horas*PF | 8 | Esfuerzo(persona-horas) | |
| Horas que se pretende trabajar al día (horas) | 5 | 2884,00 | |
| Días de trabajo a la semana (días) | 7 | Duración(horas,meses) | |
| Al mes (horas) | 140 | 2884,00 | |
| Costos | | 20,60 | |
| Sueldo (dolares) | 500 | Costo del software(dolares) | |
| Otros (dolares) | 200 | 10500,00 | |

Figura 33: Puntos de Función Sin Ajustar (PFSA).

#### 4.1.3.3. Plan de proyecto

El desarrollo se llevará a cabo basadas en fases con más de una iteración de ser necesarias. La tabla 14muestra la distribución de las fases con sus iteraciones y el tiempo estimado que llevara cada una de las mismas.

| **Fase** | **No. Iteraciones** | **Duración** |
|---|---|---|
| Fase de Inicio | 1 | 2 Semanas |
| Fase de Elaboración | 1 | 8 Semanas |
| Fase de Construcción | 3 | 16 Semanas |
| Fase de Transición | 3 | 3 Semanas |

Tabla 14: Plan de Fases.

#### 4.1.3.4. Objetivos de las Iteraciones

Al terminar la primera iteración de la fase de elaboración se habrá construido los siguientes documentos:

- Documento de especificación de requisitos de software
- Plan de desarrollo de software

- Documento de arquitectura del software
- Mapa de interacción con el software

Al finalizar la segunda iteración de la fase de construcción se habrá concretado las siguientes actividades:

- Desarrollo del primer prototipo
- Desarrollo de pruebas
- Primer prototipo Funcional

Al culminar la iteración 3 de la fase de construcción se habrá cumplido las siguientes actividades:

- Refinamiento de los requisitos funcionales, de acuerdo a las experiencias obtenidas en la presentación del primer prototipo funcional.
- Refinamiento de pruebas
- Segundo prototipo funcional /(API completa)
- Estudio de aceptación

#### 4.1.3.5. Liberación de prototipos

Se tiene prevista la liberación de dos prototipos antes de la publicación final de la API para generación automática de procedimientos almacenos en MySQL:

| **Prototipo** | **Funcionalidad** | **Liberación** |
|---|---|---|
| Prototipo 1 | 50 % de la funcionalidad total:<br>Permite generar procedimientos almacenados para operaciones como: Insert, Delete, y Update de manera automática a través del mapeo de la base de datos. | Semana 18 |
| Prototipo 2 | 100 % de la funcionalidad total:<br>Se incluyen funciones propias de MySQL, generación de procedimientos almacenados con consultas elaboradas, controles de flujo, y más procesos que se pueden realizar en un procedimiento almacenado. | Semana 24 |

Tabla 15: Liberación de prototipos.

## 4.2. Especificación de requisitos de Software

### 4.2.1. Introducción

#### 4.2.1.1. Propósito

El presente documento de especificación de requisitos de software (ERS) tiene como propósito definir las especificaciones funcionales, no funcionales de la API para la generación automática de procedimientos almacenados en MySQL que será utilizada por estudiantes, programadores, docentes y mas interesados con el área de desarrollo de software y la gestión de base de datos MySQL.
Este documento esta estructurado en base al estándar IEEE Recommended Practice for Software Requirements Specification ANS/IEEE 830 1998.

#### 4.2.1.2. Alcance

La API para la generación automática de Procedimientos Almacenados en MySQL basará su desarrollo en los requerimientos funcionales y las posibilidades existentes y permitas por MySQL en la creación de rutinas.

#### 4.2.1.3. Definiciones, Acrónimos y Abreviaciones

- **API:** abreviatura de Interfaz de Programación de Aplicaciones.
- **ERS:** abreviatura de Especificación de Requisitos Software.

#### 4.2.1.4. Características de los usuarios

La API para la generación automática de Procedimientos Almacenados en MySQL contara con los tipos de usuarios descritos en las siguiente tabla.

| Tipo de Usuario | Programador |
|---|---|
| Formación | Persona con conocimientos sobre desarrollo de programas informáticos y administración de base de datos MySQL. |
| Habilidades | Conocimientos básicos de lenguajes orientados a objetos y SQL. |
| Actividades | Persona encargada de desarrollar una aplicación que tienen conexión a una base de datos alojada en MySQL.<br>Ademas de modelar las reglas de negocios de los sistemas informáticos a través de procedimientos almacenados o rutinas. |

Tabla 16: Características del programador.

#### 4.2.1.5. Restricciones

- **Políticas Regulatorias:** La API se soportará en cualquier aplicación informática desarrollada en lenguaje C# que necesite de generación de procedimientos almacenados en MySQL. Esta tendrá una licencia GNU GLP (Licencia Pública Genera de GNU), es decir que la organización, persona que obtenga este producto de software, puedan estudiarlo, modificarlo, distribuirlo. La utilización de esta API estará rígida por las políticas establecidas en el licenciamiento impuesta en la misma.

- **Limitaciones de Hardware:** La API para la generación automática de procedimientos almacenados en MySQL podrá ser utilizada en cualquier equipo de computo que soporte un entorno de desarrollo como Visual Studio Community, con un lenguaje de programación C#.

- **Integración con otras API's:** Debido a que MySQL cuenta con una API para el manejo de bases de datos para Visual Studio. Esta formará parte del producto de software a desarrollar. Es decir, todo lo necesario para la generación de este software, usará las bibliotecas del ensamblado MySql.Data.

- **Consideraciones de Seguridad**: La seguridad de la API para la generación automática de procedimientos almacenados en MySQL hereda las clases de conexión, validación, y restricciones que maneja el ensamblado propio de MySQL para Visual Studio. Sin embargo, la ejecución de setencias SQL son analizadas por la API antes de que surjan cambios importantes sobre la base de datos.

#### 4.2.1.6. Suposiciones y Dependencias

- La API se adjuntará en el entorno de desarrollo de software Microsoft Visual Studio 2013 o superior en cualquiera de sus versiones.
- La versión de C# que se utilizará es v6.0 o superior para la integración con cualquier aplicación que necesite de la API.
- Los programadores usarán la GUI desarrollada como ejemplo para la generación de procedimientos almacenados en MySQL.

### 4.2.2. Requisitos Funcionales

#### 4.2.2.1. Modulo: Conexión

**Conexión con una base de datos**

| | |
|---|---|
| **Número de requisito** | RF42211 |
| **Nombre de requisito** | Conexión con una base da datos |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe establecer la conexión con la base de datos a través de un objeto MySQLConnection. Esta debe recibir como argumentos las propiedades de dicho objeto referenciado anteriormente. |

Tabla 17: Requisito funcional: Conexión con una base de datos.

#### 4.2.2.2. Módulo: Base de Datos

**Listar las bases de datos existentes**

| | |
|---|---|
| **Número de requisito** | RF42221 |
| **Nombre de requisito** | Listar las bases de datos existentes |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe proporcionar un método que retorne un objeto tipo List con la información perteneciente a la bases datos existentes. En este contexto, el usuario de la API debe poder acceder a cada una de las bases de datos existentes en MySQL. |

Tabla 18: Requisito funcional: Listar las bases de datos existentes.

#### 4.2.2.3. Módulo: Tablas

**Listar las tablas de una base de datos**

| **Número de requisito** | RF42231 |
|---|---|
| **Nombre de requisito** | Listar las tablas de una base de datos |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe proporcionar un método que retorne un objeto tipo List con la información perteneciente a las tablas de una base a la que se conecto. En este contexto, el usuario de la API debe poder acceder a cada una de las tablas. |

Tabla 19: Requisito funcional: Listar las tablas de una base de datos.

**Listar la estructura de las tablas**

| **Número de requisito** | RF42232 |
|---|---|
| **Nombre de requisito** | Listar la estructura de las tablas |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe proporcionar un método que retorne un objeto tipo List<Tabla> con la información de los campos de la tabla. En este contexto, el usuario de la API debe poder acceder a cada uno de los campos. |

Tabla 20: Requisito funcional: Listar la estructura de las tablas.

#### 4.2.2.4. Módulo: Vistas

**Listar las vistas de una base de datos**

| | |
|---|---|
| **Número de requisito** | RF42241 |
| **Nombre de requisito** | Listar las vistas de una base datos |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe proporcionar un método que retorne un objeto tipo List con la información perteneciente a la vistas de una bases datos en particular. En este contexto, el usuario de la API debe poder acceder a la información de la vista en MySQL. |

Tabla 21: Requisito funcional: Listar las vistas de una base datos.

**Listar la estructura de las vistas**

| | |
|---|---|
| **Número de requisito** | RF42242 |
| **Nombre de requisito** | Listar la estructura de las vistas |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe proporcionar un método que retorne un objeto tipo List<Tabla> con la información de los campos de la vista. En este contexto, el usuario de la API debe poder acceder a cada uno de los campos. |

Tabla 22: Requisito funcional: Listar la estructura de las vistas.

#### 4.2.2.5. Módulo: Funciones MySQL

**Listar las funciones propias de MySQL**

| | |
|---|---|
| **Número de requisito** | RF42251 |
| **Nombre de requisito** | Listar las funciones propias de MySQL |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe proporcionar un método que retorne un objeto tipo List<Funcion> con la información de los campos importantes de la función. En este contexto, el usuario de la API debe poder acceder a cada uno de los campos y trabajar con ellos sin restricciones. |

Tabla 23: Requisito funcional: Listar las funciones propias de MySQL.

**Listar las funciones propias de una base de datos**

| | |
|---|---|
| **Número de requisito** | RF42252 |
| **Nombre de requisito** | Listar las funciones propias de una base de datos |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe proporcionar un método que retorne un objeto tipo List<Funcion> con la información de los campos importantes de la función. En este contexto, el usuario de la API debe poder acceder a cada uno de los campos y trabajar con ellos sin restricciones. |

Tabla 24: Requisito funcional: Listar las funciones propias de MySQL.

#### 4.2.2.6. Módulo: Relaciones

**Listar las relaciones entre las tablas**

| | |
|---|---|
| **Número de requisito** | RF42261 |
| **Nombre de requisito** | Listar las relaciones entre las tablas |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe proporcionar un método que retorne un objeto tipo List<Relacion> con la información de los campos importantes de las relaciones existentes. En este contexto, el usuario de la API debe poder acceder a cada uno de los campos y trabajar con ellos sin restricciones. Este Modulo permitirá generar cogido SQL de consultas sobre una base de datos. |

Tabla 25: Requisito funcional: Listar las relaciones entre las tablas.

#### 4.2.2.7. Módulo: Operaciones

**Mapeo de campos**

| | |
|---|---|
| **Número de requisito** | RF42271 |
| **Nombre de requisito** | Mapeo de campos |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe realizar operaciones de mapeo sobre la base de datos , estas operaciones usaran módulos como Tablas, Base de Datos, Funciones y entre otros descritos anteriormente. Cada uno de los módulos serán integrados en este, para cumplir las expectativas referentes al tipo de sentencia que se desea generar. En este contexto, el usuario de la API debe poder obtener un código SQL usando todos los objetos posibles referenciados a la base de datos. |

Tabla 26: Requisito funcional: Mapeo de Campos.

**Interpretación de código SQL**

| | |
|---|---|
| **Número de requisito** | RF42271 |
| **Nombre de requisito** | Interpretación de código SQL |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe realizar operaciones de transcripcion de variables integradas por la misma para facilitar la creacion de procedimientos almacenados con todos los tipos de parametros soportados por MySQL. Estas varibles empezaran con '#', el tipo de operación sera definida por los siguientes caracteres: (i, o , io) representando a parámetros IN, OUT, INOUT. Ademas se debe agregar el nombre de la variable. un ejemplo de la variable es: '#icd.nombre' donde 'i' es el tipo de parámetro, 'cd' es el alias de la tabla, 'nombre' es el campo de la tabla. Al integrar estas variables en el código SQL se generarán los parámetros del procedimiento automáticamente gracias al modulo de Mapeo de Campos. |

Tabla 27: Requisito funcional: Interpretación de código SQL.

**Validación de Procedimientos Almacenados Existentes**

| | |
|---|---|
| **Número de requisito** | RF42273 |
| **Nombre de requisito** | Validación de Procedimientos Almacenados Existentes |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe poder validar si existe el procedimiento en la base de datos, ademas se debe poder agregar un método para borrar el procedimiento almacenado de ser necesario. |

Tabla 28: Requisito funcional: Validación de Procedimientos Almacenados Existentes.

### Generación de código SQL

| | |
|---|---|
| **Número de requisito** | RF42274 |
| **Nombre de requisito** | Generación de código SQL |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe poder generar código SQL de procedimientos almacenados con operaciones previamente establecidas como: Insert, Select, Update, Delete,y Busqueda. |

Tabla 29: Requisito funcional: Generación de código SQL.

### Ejecución de sentencias SQL

| | |
|---|---|
| **Número de requisito** | RF422734 |
| **Nombre de requisito** | Ejecución de sentencias SQL |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe poder ejecutar una sentencia SQL por medio de un método. Ademas el método debe ser capaz de reconocer que tipo de sentencia se va ejecutar y devolver los resultados correspondientes. |

Tabla 30: Requisito funcional: Mapeo de Campos.

### Crear Sentencias SQL dinámicas

| | |
|---|---|
| **Número de requisito** | RF42275 |
| **Nombre de requisito** | Crear Sentencias SQL dinámicas |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [ ] Alta/Esencial [**X**] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe poder retornar código de sentencias SQL de forma rápida y sencilla. |

Tabla 31: Requisito funcional: Crear Sentencias SQL dinámicas.

**Generar SQL de consultas**

| | |
|---|---|
| **Número de requisito** | RF42276 |
| **Nombre de requisito** | Generar SQL de consultas |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [ ] Alta/Esencial [**X**] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe poder retornar código SQL de consultas entre tablas de las bases de datos, tomando en cuenta las relaciones existentes sobre las mismas. |

Tabla 32: Requisito funcional: Generar SQL de consultas.

#### 4.2.2.8. Módulo: Procedimientos Almacenados

**Listar los Procedimientos Almacenados**

| | |
|---|---|
| **Número de requisito** | RF42281 |
| **Nombre de requisito** | Listar los Procedimientos Almacenados |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe listar los procedimientos almacenados de una base de datos en un objeto List<Procedure>. El usuario debe poder manipular cada una de las propiedades de los procedimientos. |

Tabla 33: Requisito funcional: Listar los procedimientos Almacenados.

**Generar código SQL de Procedimientos Almacenados Simples**

| | |
|---|---|
| **Número de requisito** | RF42282 |
| **Nombre de requisito** | Generar código SQL de Procedimientos Almacenados Simples |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe poder obtener el método que retorne el código SQL en base al nombre de la tabla y la operación elegida. El usuario debe poder manipular el código sin restricciones. |

Tabla 34: Requisito funcional: Generar código SQL de Procedimientos Almacenados Simples.

**Generar código SQL de Procedimientos Almacenados Complejos**

| | |
|---|---|
| **Número de requisito** | RF42283 |
| **Nombre de requisito** | Generar código SQL de Procedimientos Almacenados Complejos |
| **Tipo** | [**X**] Requisito [ ] Restricción |
| **Actor** | Programador |
| **Prioridad** | [**X**] Alta/Esencial [ ] Media/Deseado [ ] Baja/Opcional |
| **Descripción** | La API debe poder obtener el método que retorne el código SQL en base al código que será interpretada en uno de los métodos del módulo Operaciones. El usuario debe poder ser capaz de incluir dos o más sentencias SQL, las mismas que podrán ser agregadas de forma dinámica. |

Tabla 35: Requisito funcional: Generar código SQL de Procedimientos Almacenados Simples.

### 4.2.3. Requisitos no funcionales

#### 4.2.3.1. Requisitos de Rendimiento

La API para la generación automática de Procedimientos Almacenados en MySQL deberá poder conectarse a cualquier servidor de base de datos MySQL y mantener la conexión durante las operaciones necesarias que el programador desee, en este punto el rendimiento de la API es directamente proporcional a la capacidad de respuesta del servidor, ya sea que el mismo se encuentre trabajando en el mismo equipo o en uno remoto.

#### 4.2.3.2. Seguridad

La API es dependiente de las tecnologías de seguridad implantadas en el servidor de base de datos MySQL, y a su vez de las técnicas, herramientas y procesos que el usuario de la API implante en su entorno de desarrollo.

## 4.3. Documento de Arquitectura de Software

### 4.3.1. Introducción

#### 4.3.1.1. Propósito

El presente documento de arquitectura, tiene como finalidad presentar la arquitectura de la API para la generación automática de Procedimientos Almacenados en MySQL, el cual detalla en diferentes vistas el comportamiento de la API considerando los objetivos establecidos en este proyecto.

#### 4.3.1.2. Alcance

La descripción del documento se encuentra basado en diferentes perspectivas lógicas de la API para la generación automática de Procedimientos Almacenados en MySQL, por tal motivo se incluyen aspectos relevantes de cada una de las mismas.

#### 4.3.1.3. Organización del Documento

El documento se organiza en base a la plantilla Software Architecture Document del proceso de desarrollo elaborado por OpenUP, la misma que es adaptada a las características particulares del tipo de proyecto en desarrollo.

### 4.3.2. Representación de la Arquitectura

El modelo propuesto utilizas las siguientes vistas:

- **Vista de casos de usos:** lista los casos de usos que representa funcionalidades centrales de la API para la generación automática de procedimientos almacenados en MySQL que requieran una gran cobertura arquitectónica o aquellos que implican un punto delicado de la arquitectura.
- **Vista lógica:** describe las partes de la arquitectura más significativas del modelo de diseño, como son la descomposición en capas, paquetes o bibliotecas. Una vez presentadas estas unidades lógicas esenciales, se profundiza en ellas hasta el nivel requerido para la API.

- **Vista de despliegue:** describe uno o más escenarios de distribución física de la API sobre los cuales se ejecutará y hará el despliegue de la misma. Es decir, muestra la comunicación y relaciones existentes entre los diferentes nodos que componen los escenarios antes mencionados, así como el mapeo de los elementos de la Vista de Procesos en dichos nodos.
- **Vista de implementación:** describe la estructura general del Modelo de Implementación y el mapeo de los componentes y clases de la Vista Lógica que intervienen en esta vista.
- **Vista de datos:** describe los elementos principales del Modelo de Datos, brindando un panorama general de dicho modelo en términos de tablas.

### 4.3.3. Objetivos y Restricciones de la Arquitectura

El desarrollo de la API para la generación automática de Procedimientos Almacenados en MySQL esta orientado a la elaboración de un software de alta calidad, que sea intuitivo, comprensible y que aporte a las necesidades de los programadores.
A nivel de arquitectura las principales metas son las siguientes:

- Diseño basados en componentes con fines concretos, claros y con alto grado de cohesión,
- Desacoplamiento entre componentes que permita un mantenimiento optimo y de ser necesario el remplazo de los mismos,
- Componentes re-utilizables.

En la planificación de este proceso se han encontrado restricciones como:

- Restricción de diseño: debido a que la API propuesta es para programadores que usen como gestor de base de datos a MySQL para la generación automática de procedimientos almacenados, la misma debe ser adaptada a la forma de trabajo del gestor. En este contexto se debe diseñar una Aplicación gráfica (GUI) que use la API y facilite las pruebas de la misma.

### 4.3.4. Vista de Casos de Uso

#### 4.3.4.1. Introducción

La vista de Casos de Uso describe los casos de uso o escenarios que presenten funcionalidades importantes para la API final, es decir, que requieran de una

gran cobertura arquitectónica o aquellos que impliquen un punto delicado de la arquitectura. Estos casos de uso, en conjunto con los requerimientos no funcionales, permiten descubrir y diseñar la arquitectura de la API para la generación automática de Procedimientos Almacenados en MySQL.

#### 4.3.4.2. Identificación de los Casos de Uso relevantes para la arquitectura

Para el diseño de la API que genera de forma automática Procedimientos Almacenados en MySQL, se identifican como casos de uso importantes desde el punto de vista de la arquitectura de este proyecto, a los siguientes:

- Administración de Conexiones
- Administración de Tablas
- Administración de Vistas
- Administración de Funciones
- Operaciones
- Administración de Procedimientos Almacenados
- Administración de la API

**Administración de Conexiones**

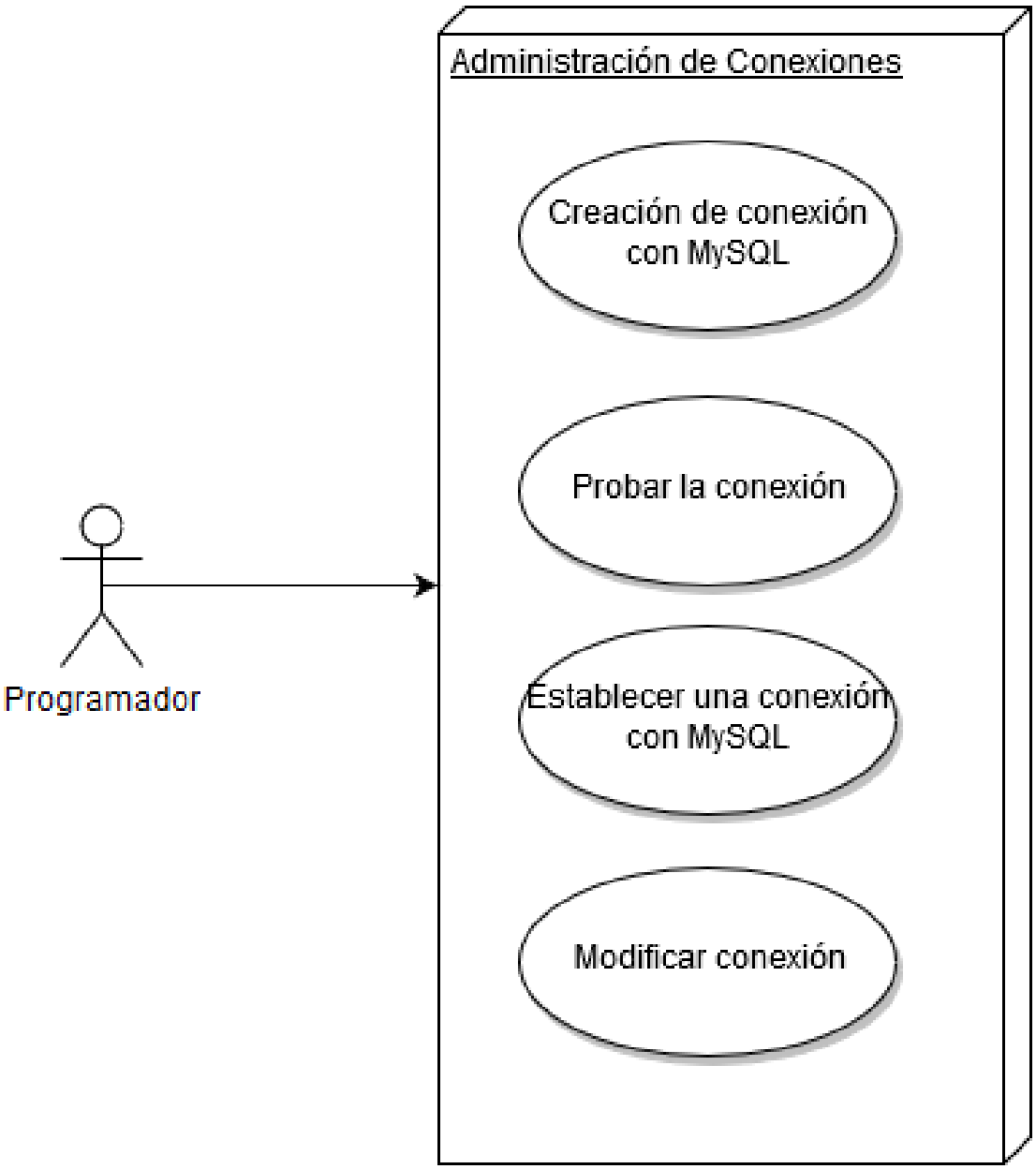



Figura 34: Casos de Uso: Administración de Conexiones.

**Administración de Tablas**

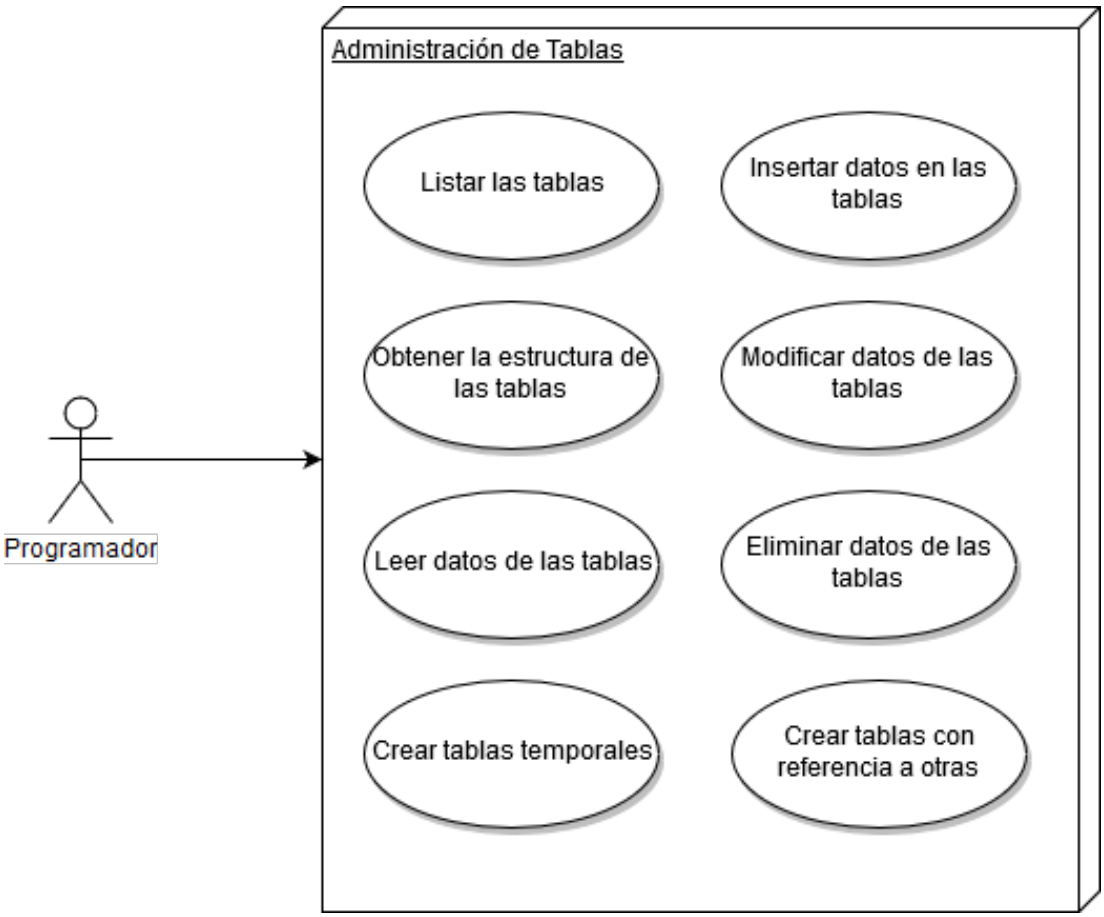



Figura 35: Casos de Uso: Administración de Tablas.

### Administración de Vistas

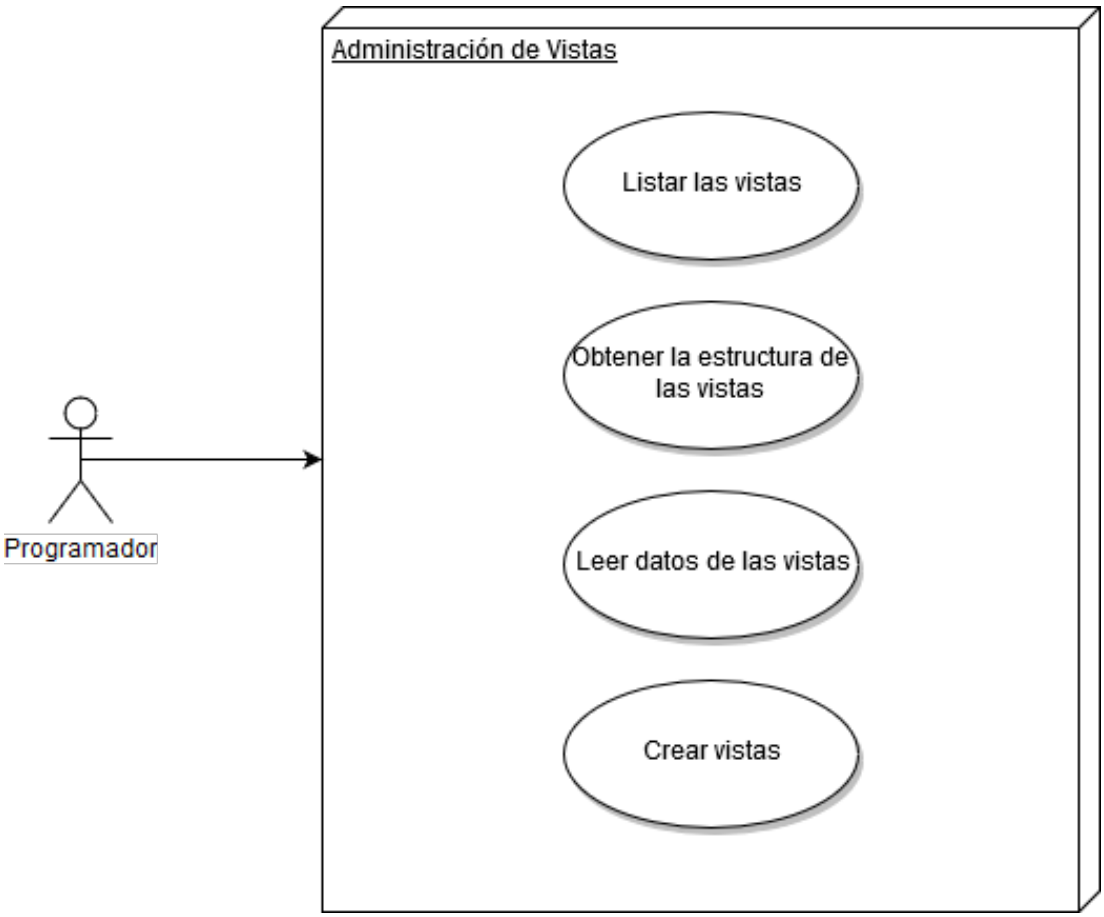



Figura 36: Casos de Uso: Administración de Vistas.

### Administración de Funciones

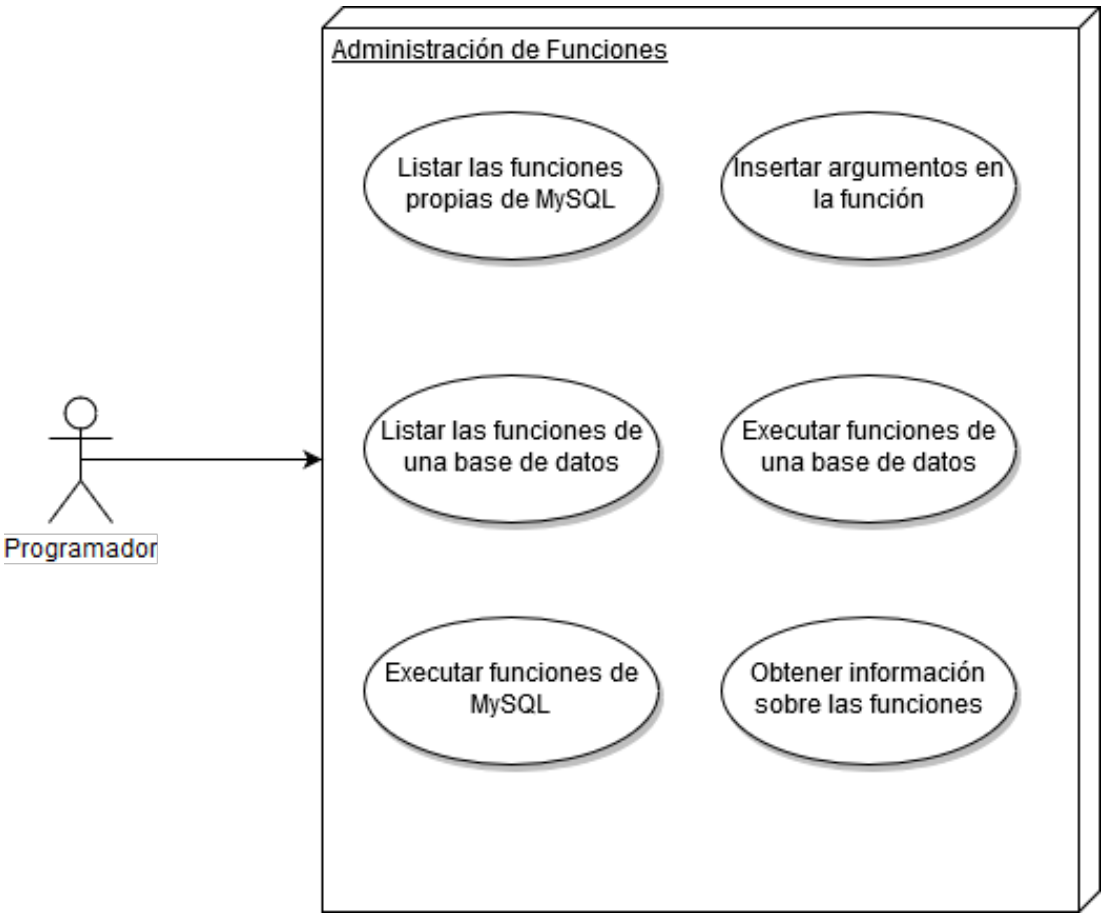



Figura 37: Casos de Uso: Administración de Funciones.

**Operaciones**

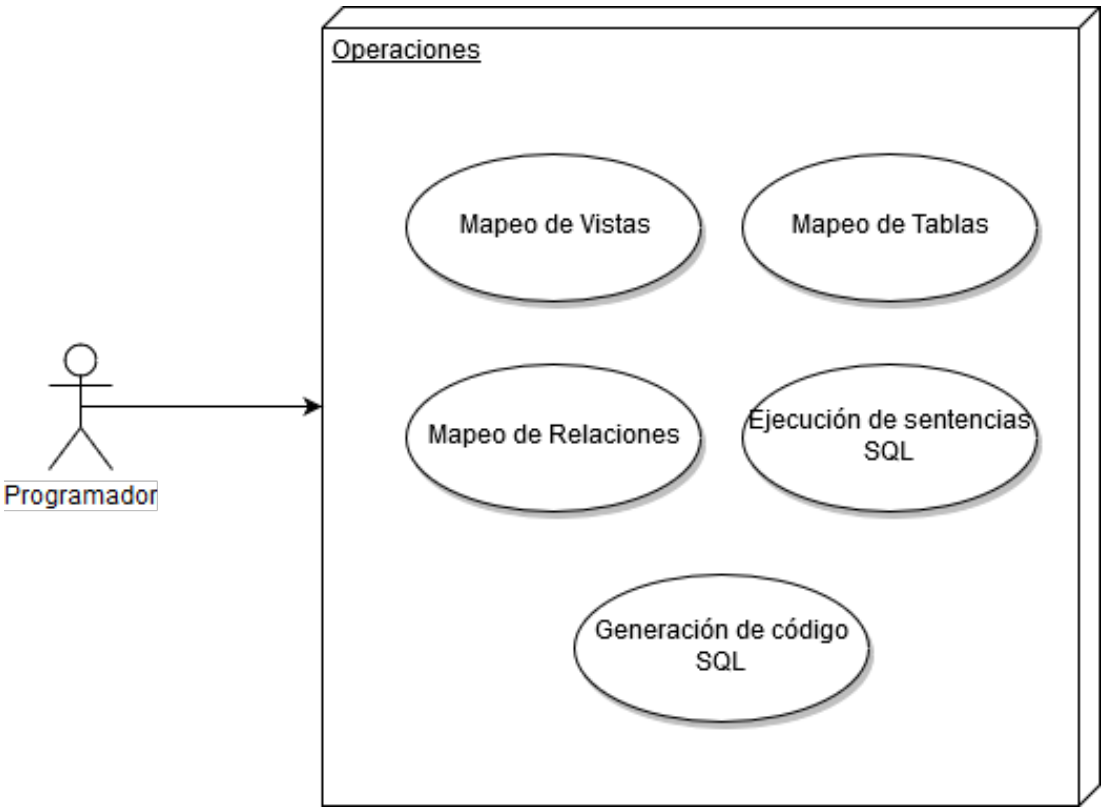



Figura 38: Casos de Uso: Generación de código SQL.

**Administración de Procedimientos Almacenados**

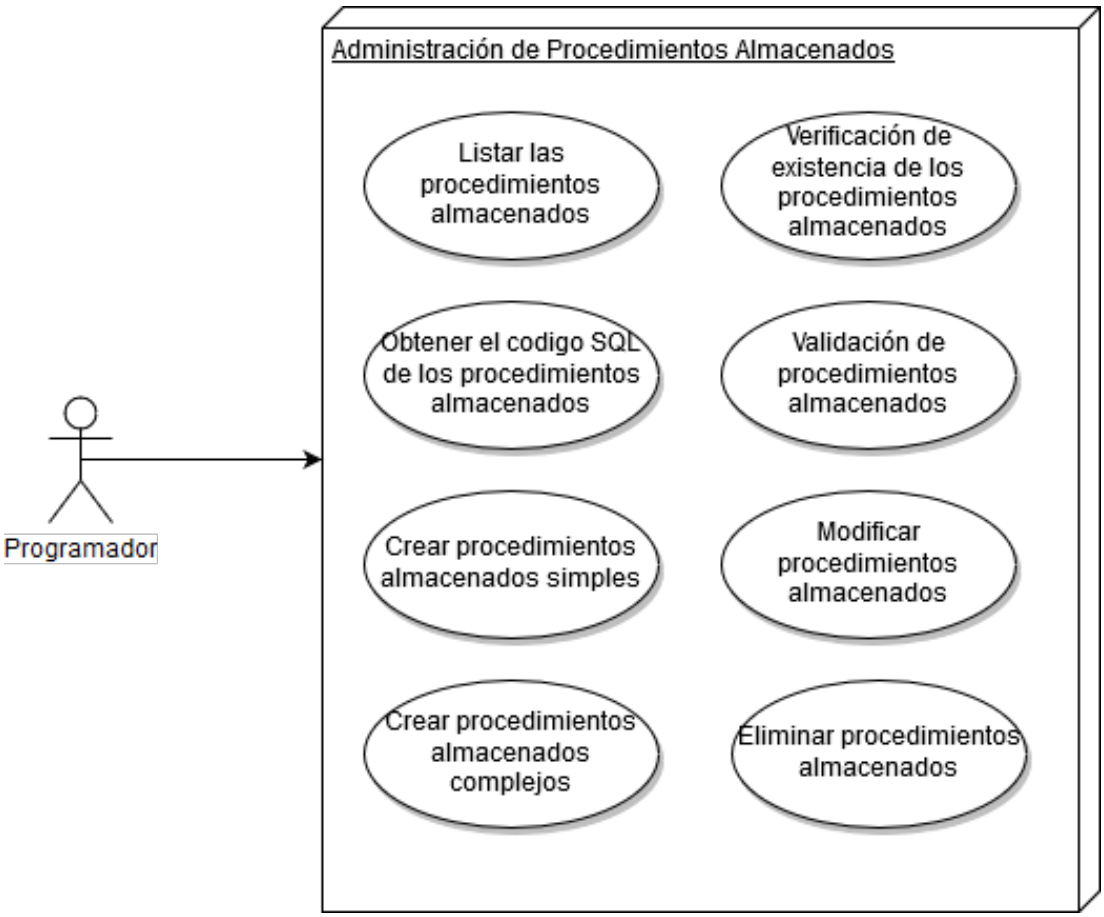



Figura 39: Casos de Uso: Administración de Procedimientos Almacenados.

### Administración de la API

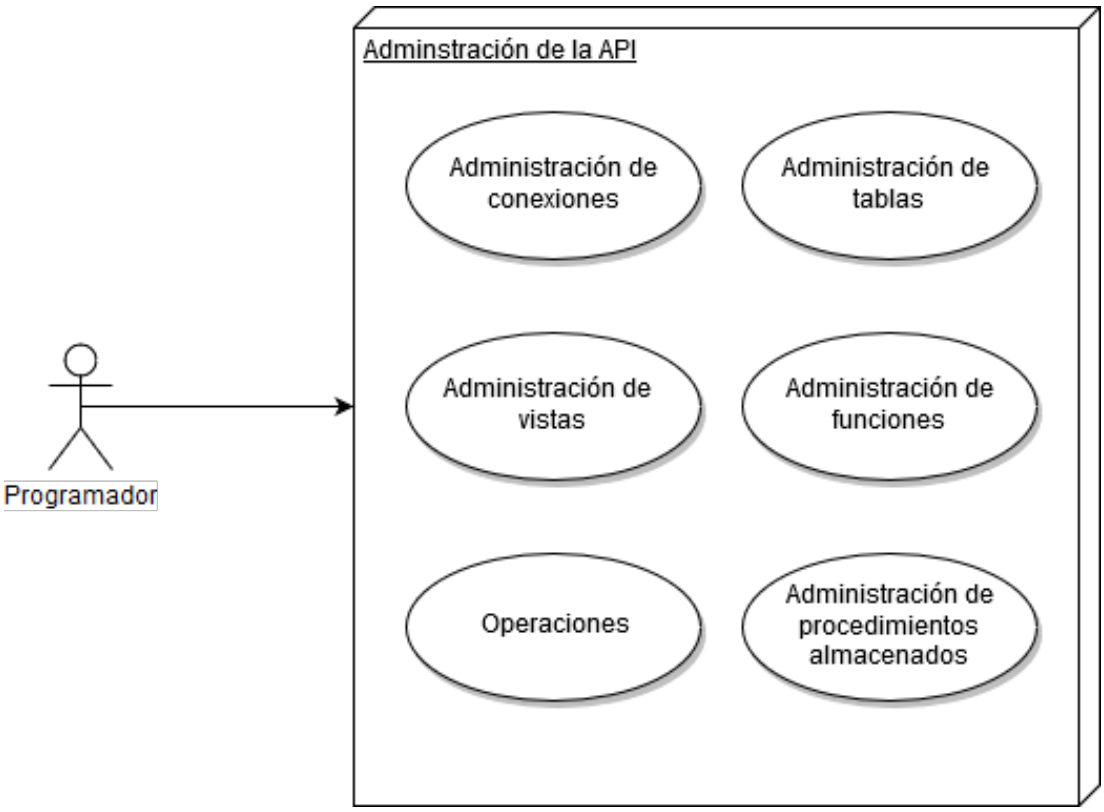



Figura 40: Casos de Uso: Administración de la API.

#### 4.3.4.3. Descripción de los casos de uso relevantes para la arquitectura

Remitirse a la sección 4.2.2 Requisitos Funcionales del documento de Especificación de Requisitos de Software.

### 4.3.5. Vista Lógica

#### 4.3.5.1. Introducción

Se describen en este punto los refinamientos que definen las diferentes unidades lógicas que componen la arquitectura de la API para la generación automática de Procedimientos Almacenados en MySQL.
El primer refinamiento realizado consiste en la descomposición de la API en subsistemas. Los subsistemas presentan cortes verticales del diseño de la API , es decir que, cada subsistema tiene un grupo de diferentes funcionalidades relacionadas entre si y posee la capacidad funcional como un sistema mismo.
Posteriormente se explora la composición de cada subsistema que integra la API.
Y, para finalizar se incluye la realización de los casos de uso previamente descritos en la sección anterior mediante los componentes arquitectónicos definidos.

#### 4.3.5.2. Descomposición en Subsistemas

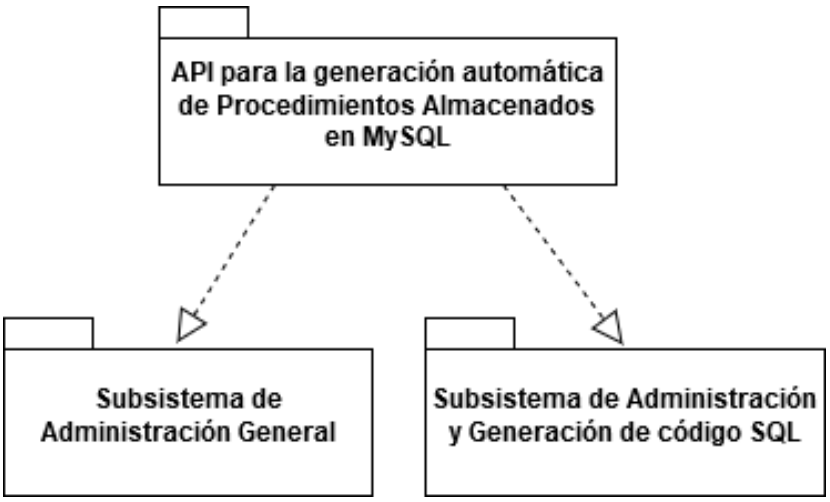



Figura 41: Descomposición en subsistemas.

#### 4.3.5.3. Descripción de los Subsistemas

- **Subsistema de Administración General:** Este subsistema permitirá la interacción entre las operaciones previamente establecidas en la API y las aplicaciones que hagan uso de la misma.
- **Subsistema de Administración y Generación de código SQL:** Este subsistema estará encargado de todas las operaciones necesarias para la creación de sentencias SQL soportadas dentro de un procedimiento almacenado en MySQL.

#### 4.3.5.4. Diseño de Subsistemas

- **Subsistema de Administración General**

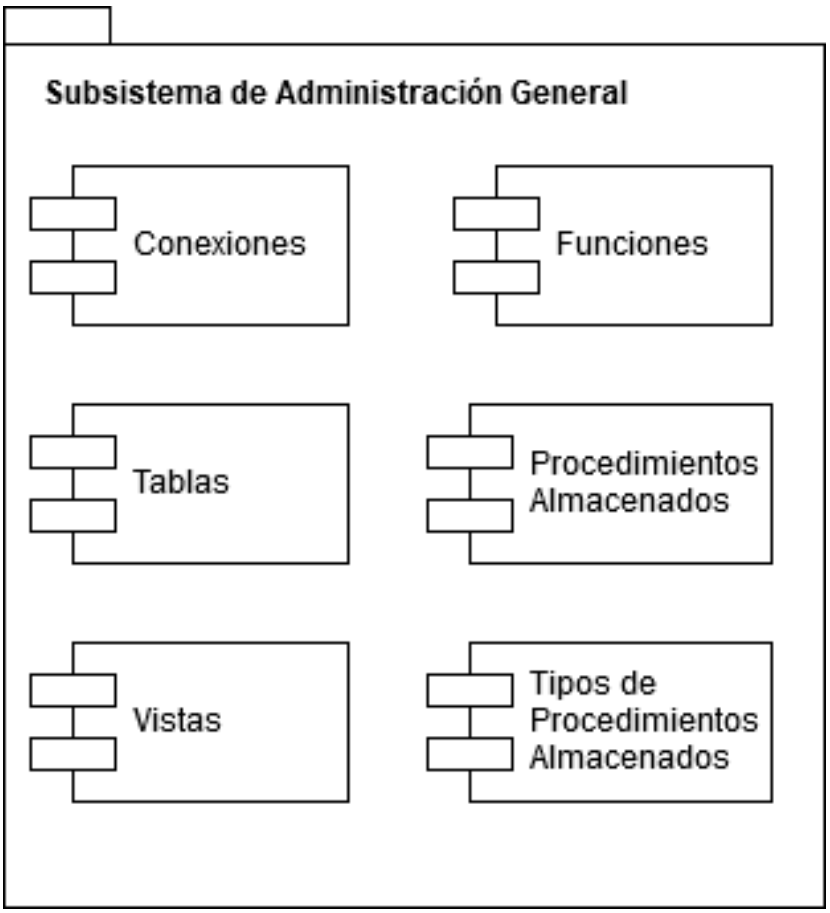



Figura 42: Subsistema de Administración General.

- **Subsistema de Administración y Generación de código SQL**

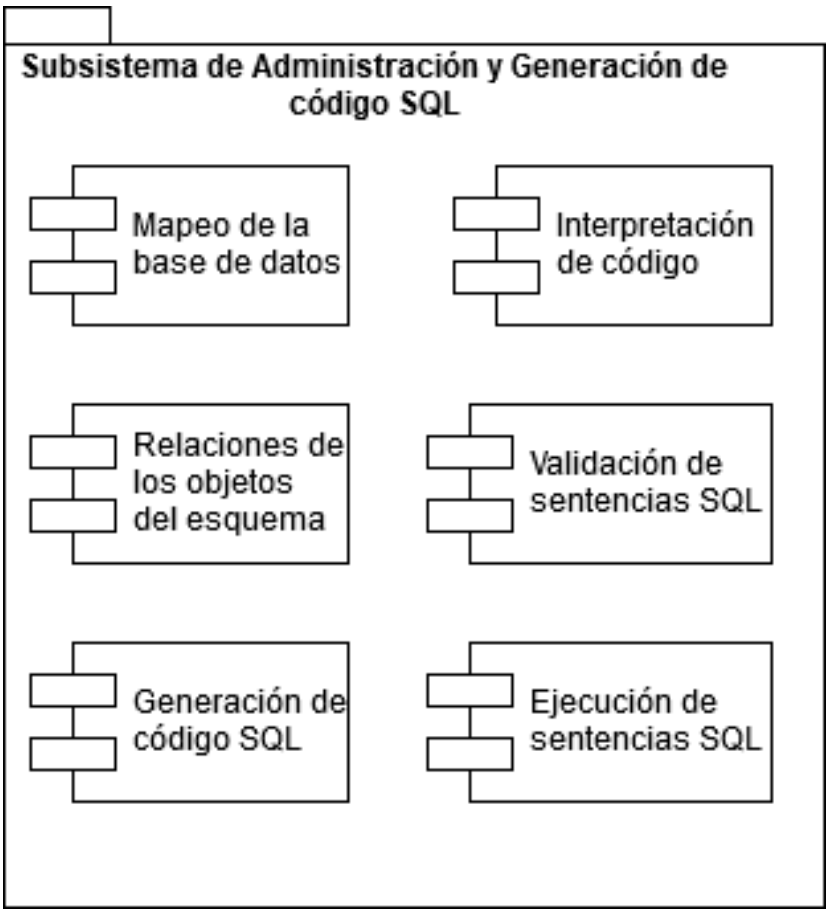



Figura 43: Subsistema de Administración y Generación de código SQL.

#### 4.3.5.5. Realización de los casos de uso relevantes para la arquitectura

**Administración de Conexiones**

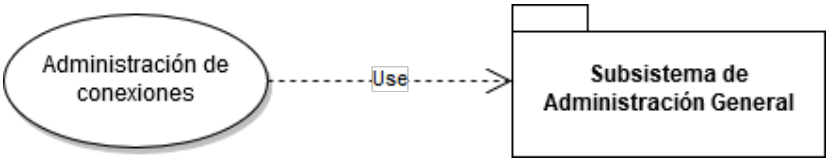



Figura 44: Realización de Casos de Uso: Administración de Conexiones.

**Administración de Tablas**

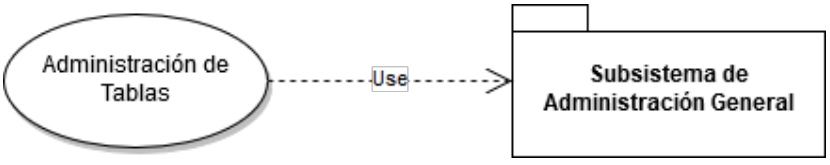



Figura 45: Realización de Casos de Uso: Administración de Tablas.

### Administración de Vistas

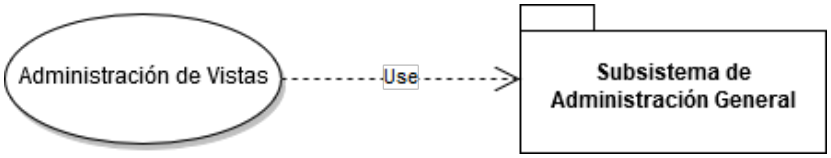



Figura 46: Realización de Casos de Uso: Administración de Vistas.

### Administración de Funciones

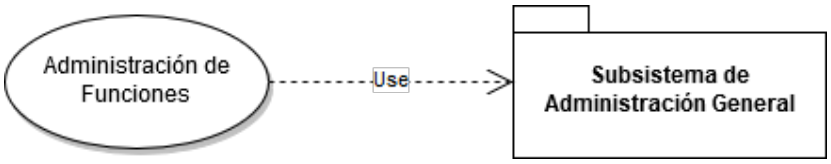



Figura 47: Realización de Casos de Uso: Administración de Funciones.

### Operaciones

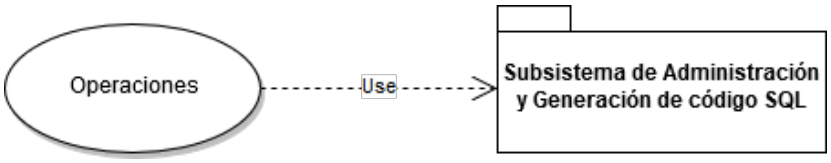



Figura 48: Realización de Casos de Uso: Operaciones.

### Administración de Procedimientos Almacenados

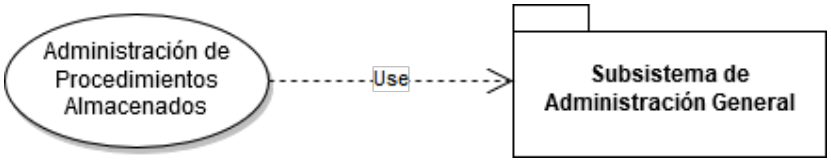



Figura 49: Realización de Casos de Uso: Administración de Funciones.

### Administración de la API

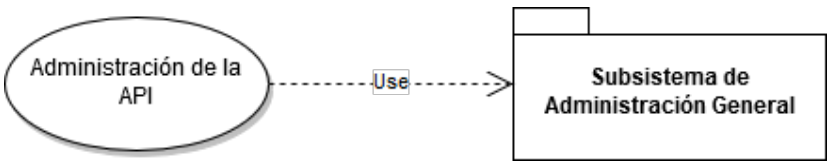



Figura 50: Realización de Casos de Uso: Administración de la API.

#### 4.3.5.6. Diagrama de Clases

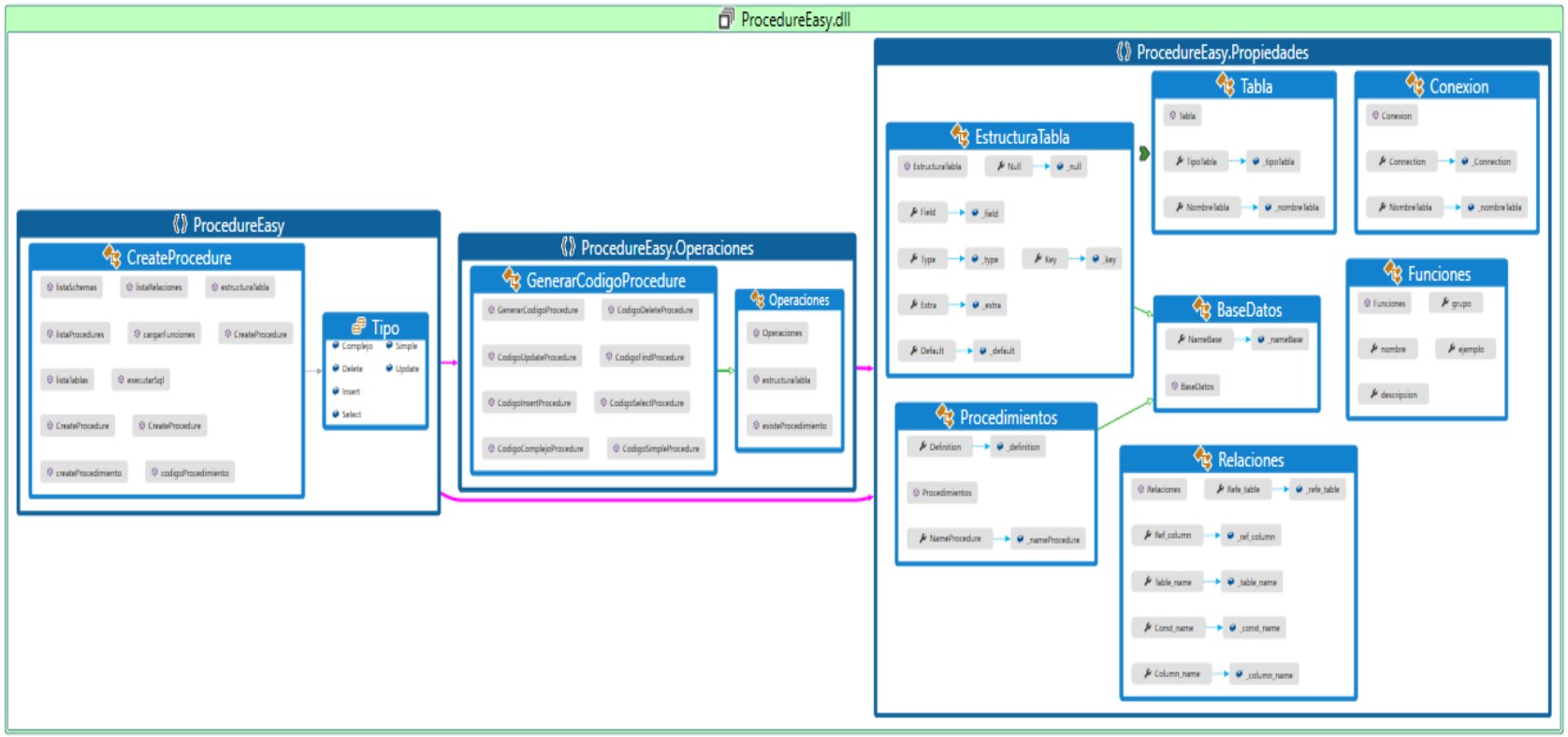



Figura 51: Diagrama de Clases.

### 4.3.5.7. Diagrama de Entidad/Relación

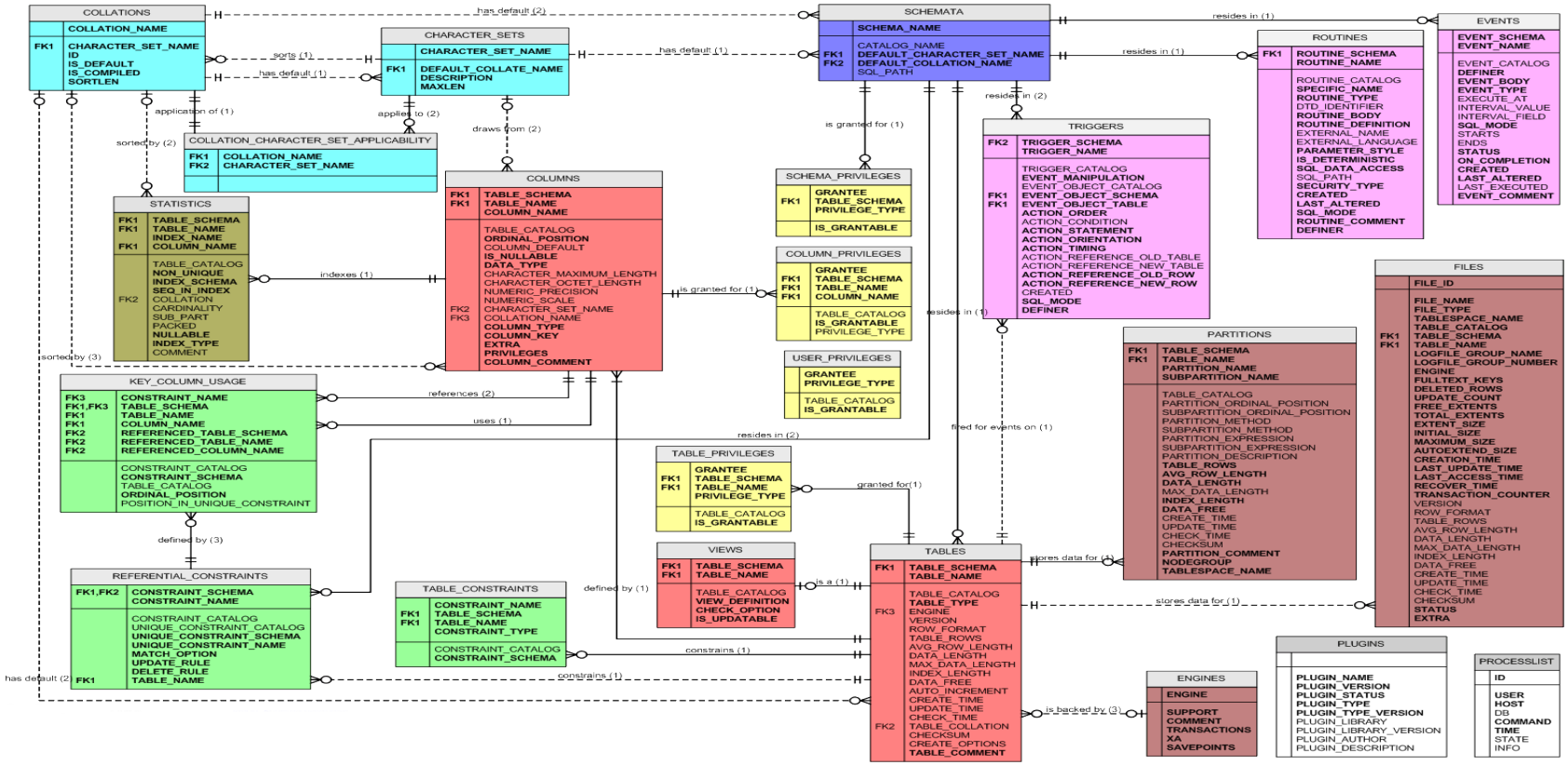



Figura 52: Diagrama de Entidad/Relación.

### 4.3.6. Vista de Despliegue

#### 4.3.6.1. Introducción

Está sección describe las configuraciones físicas sobre las cuales se realiza el despliegue del software y se lo ejecuta, ademas de la infraestructura necesaria para su instalación.
Para el caso de la API para la generación automática de Procedimientos Almacenados en MySQL se describe un escenario general de distribución esperado para los componentes de software, las características de los nodos presentados y la comunicación entre los mismos.

#### 4.3.6.2. Distribución y Despliegue

La siguiente figura muestra el escenario general de distribución esperado para la integración de la API:

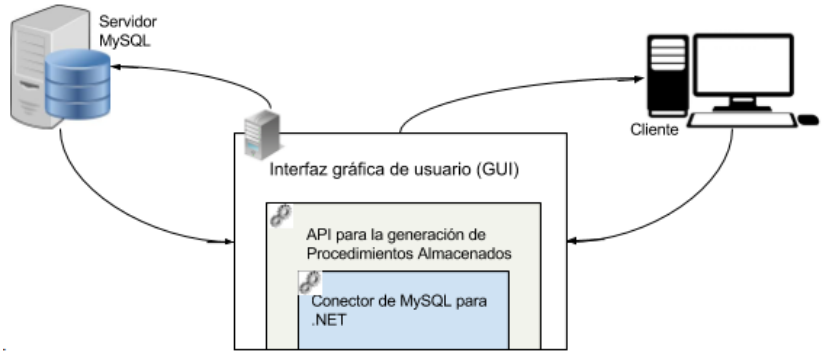



Figura 53: Distribución y Despliegue.

**Escenario del Entorno**

- **Servidor de base de datos MySQL:** un sistema de gestión de base de datos con varios motores de base de datos, multiusuario y multihilo.
- **Interfaz gráfica de usuario (GUI):** esta interfaz usa la API para la generación automática de Procedimientos Almacenados y del conector de MySQL para .NET.
- **Cliente:** es el usuario de la GUI, que depende de la tecnología que usó el programador para el desarrollo de la misma, es decir que si la aplicación fue desarrollada para ejecutarse en la web, esta necesitará de un navegador, si se desarrollo como una aplicación de escritorio las características del equipo del cliente serán diferentes. Ademas que, por cada una de las tecnologías

usadas para el desarrollo de la GUI, esta se puede implantar en otras que beneficien al usuario final.

### 4.3.7. Vista de Datos

Remitirse a la sección 4.3.5.7 referente al Diagrama de Entidad/ relación, la misma da una visión de los datos a los que se puede acceder.

## 4.4. Plan de Pruebas de Software

### 4.4.1. Introducción

#### 4.4.1.1. Objetivos

El plan de pruebas de software se elabora con el fin de especificar los componentes que se van evaluar para que el grupo de trabajo pueda realizar el proceso de validación de los requerimientos funcionales y no funcionales de la API para la generación automática de procedimientos almacenados en MySQL. Ademas, a través del plan de pruebas se tiene un control y se puede continuar con la trazabilidad de los requerimientos, con lo cual el grupo de trabajo, identifica el porcentaje de avance que se ha logrado hasta cierto tiempo.
Al desarrollar un plan de pruebas, se puede obtener información sobre los errores, defectos o fallas que tiene el prototipo, así se realizan las correcciones pertinentes, según el caso, y así se asegura la calidad del producto que se entregará al usuario final. El plan de pruebas se aplica sobre el producto, es decir, el código fuente, los resultados de las pruebas son registrados en un formato que se encuentra en el reporte de pruebas.

#### 4.4.1.2. Estrategia de pruebas

A través de los diferentes documentos que se han elaborado, se puede registrar información relacionada con las pruebas, de esta forma se asegura la calidad de estas y por ende la del producto en general. Ademas permite al responsable de las pruebas, saber exactamente los criterios que se deben tomar al poner a prueba cada uno de los elementos de la API. Abajo se explica de forma breve la relación, el aporte de cada documento con respecto a este, el plan de pruebas.

| Especificación de Requisitos de Software |
| --- |
| • Priorización: se seleccionan los requerimientos de mayor prioridad para aplicar las pruebas correspondientes.<br>• Dependencias: muestra una visualización clara de los diferentes grupos de requerimientos que deben ser evaluados dentro de las pruebas de integración.<br>• Trazabilidad: le permite saber al responsable de la prueba el estado de cada uno de los requerimientos |

Tabla 36: Descripción de la Especificación de Requisitos de Software.

| Documento de Arquitectura de Software |
| --- |
| • Diagrama de Componentes: la disgregación de la API en componentes con funcionalidades especificas, permite que las pruebas se puedan manejar y clasificar según la funcionalidad de cada uno de los módulos.<br>• Diagrama de los Casos de Uso: muestra una visualización clara de los diferentes escenarios que se deben tomar en cuenta para realizar las pruebas.<br>• Vista Física: prueba de los componentes de Hardware que tiene la API. |

Tabla 37: Descripción del Documento de Arquitectura de Software.

Con esta estrategia se garantiza llevar un seguimiento de la trazabilidad que se ha manejado desde la Especificación de Requerimientos (ERS), ademas de mantener la consistencia entre la API y su respectiva documentación.

#### 4.4.1.3. Alcance

Teniendo en cuenta los documentos hechos anteriormente, el grupo de trabajo pretende realizar las pruebas, de manera incremental por módulo. En este contexto las pruebas funcionales de la API se realizarán inmediatamente después de haber implementado una funcionalidad, es decir, que el orden corresponde al que se establece en la prioridad de los requerimientos.

### 4.4.2. Instrumentos de Prueba

#### 4.4.2.1. Módulos del Programa

En esta sección se presentan los módulos que componen la Interfaz de Programación de Aplicación para la generación automática de Procedimientos

Almacenados en MySQL, ademas de las especificaciones de las pruebas a realizar en cada uno. Cabe recalcar, que cada módulo representa un componente en la API.

| Módulo | Pruebas | Descripción |
|---|---|---|
| **Interfaz Gráfica de Usuario** | Facilidad de uso | La facilidad de uso permite evaluar si el usuario, en cada momento, tiene conocimiento sobre lo que puede y debería ser. En el caso de la interfaz gráfica, esta sirve para evaluar las funcionalidades y propiedades que brinda la API. |
| | Look & feel | Look & feel se refiere, a la apariencia que se le proporciona al usuario de la interfaz que integra la API de forma implícita para las pruebas. |
| **Lógica de Negocio** | Funcionalidad | La API debe cumplir con todos los requerimientos ya detallados en el documento referente a las especificaciones requeridas para el software. |
| **No funcionales** | No funcionales | La API debe cumplir con los requerimientos no funcionales que se han especificado en el SRS teniendo en cuenta el diseño. |

Tabla 38: Descripción de los módulos del programa.

### 4.4.3. Características a ser probadas

En esta sección se encuentran las características de la API a ser probadas, conforme al alcance del Proyecto.

| Característica | Descripción | Módulo |
|---|---|---|
| Requerimientos Funcionales | Se debe tener en cuenta el criterio de aceptación y de dependencias para realizar las pruebas de los módulos. En este contexto se debe hacer uso de la vista de los casos de uso, para tener claro los aspectos que se prueban, y si la API cumple con los mismos. | El módulo que permite probar esta característica es: Lógica de Negocios |

Tabla 39: Características a ser probadas.

### 4.4.4. Aproximación

En esta sección se exponen los tipos de pruebas a utilizar en la API para la generación automática de Procedimientos Almacenados en MySQL, para cada una de ellas se presenta un formato para el registro de los resultados.

#### 4.4.4.1. Pruebas Funcionales

En el caso de las pruebas funcionales, estas son aplicadas al los requerimientos funcionales de la API, es decir, las que se registraron en el documento de Especificación de Requerimientos del Software (ERS). En este proyecto, se desarrollo una Interfaz de Usuario (GUI) para poner a prueba los aspectos funcionales de la API.

| **Pruebas de Integración** | **Código: INT01** |
|---|---|
| **Actividades** | Validación de los requerimientos funcionales |
| **Herramientas** | EasyProcedure, herramienta con interfaz gráfica desarrollada para las pruebas. |
| **Tiempo estimado** | 30 minutos por prueba |
| **Entregable** | Informe de las pruebas funcionales realizadas |

Tabla 40: Pruebas de Integración.

#### 4.4.4.2. Pruebas de Comportamiento

Las pruebas de comportamiento se le realiza a la herramienta desarrollada para las pruebas de integración. Estas pruebas garantizan la calidad de la API y de la aplicación EasyProcedure. En este contexto, se definen los casos de uso de EasyProcedure, los mismos que son apegados a los de la API. En la siguiente sección se definirán rutas de éxitos y fallos del programa.

| **Pruebas de Comportamiento** | **Código: COM01** |
|---|---|
| **Actividades** | Realizar las pruebas de comportamiento |
| **Herramienta** | EasyProcedure |
| **Tiempo estimado** | 50 minutos por prueba |
| **Entregable** | Informe de las pruebas funcionales realizadas |

Tabla 41: Pruebas de Comportamiento.

### 4.4.5. Proceso de Pruebas

En esta sección se muestra los casos de pruebas generales que serán usados por la herramienta EasyProcedure que hace uso de la API para la generación automática de Procedimientos Almacenados en MySQL. Para este proceso se tomarán en cuenta los casos de uso de la API, es decir, que el alcance y los escenarios de la API serán los mismo para la herramienta EasyProcedure. En las tablas siguientes, se presenta los casos de pruebas:

| **Nombre** | **Administración de Conexiones** \| **Pruebas PC01** |
|---|---|
| **Propósito** | Verificar si es posible conectarse a una base de datos en MySQL |
| **Prerrequisitos** | Listar las bases de datos existentes en MYSQL |
| **Ubicación** | EasyProcedure y base de datos MySQL |
| **Entrada** | Acciones que se pueden realizar:<br>- Creación de la conexión<br>- Probar conexión<br>- Establecer conexión<br>- Modificar conexión |
| **Oráculo** | 1. CREACIÓN DE LA CONEXIÓN: Mostrar los campos necesarios de ingreso para la conexión.<br>2. PROBAR CONEXIÓN: Probar si los datos son correctos para establecer la conexión.<br>3. ESTABLECER LA CONEXIÓN: El programa mostrará la base de datos a la que se conectó.<br>4. MODIFICAR CONEXIÓN: Se actualiza la información para establecer una nueva conexión. |
| **Pasos** | 1. CREACIÓN DE LA CONEXIÓN<br>- Abrir el programa EasyProcedure<br>- Clic en la opción conexión del menú principal<br>- Llenar el formulario con los datos de conexión<br>2. PROBAR CONEXIÓN<br>- Seleccionamos una base de datos en formulario de conexión<br>- Pulsar el botón probar conexión<br>3. ESTABLECER LA CONEXIÓN<br>- Pulsar el botón conectar<br>4. MODIFICAR CONEXIÓN<br>- Clic en la opción conexión del menú principal<br>- Llenar el formulario con los datos de conexión |
| **Módulos Asociados** | - Base de Datos |

Tabla 42: Pruebas: Administración de conexiones.

| **Nombre** | **Administración de Vistas** | **Pruebas PC02** |
|---|---|---|
| **Propósito** | Verificar si es posible listar, obtener la estructura, leer los datos, y crear vistas. | |
| **Prerrequisitos** | Establecer una conexión con una base de datos en MySQL. | |
| **Ubicación** | EasyProcedure y base de datos MySQL. | |
| **Entrada** | Acciones que se pueden realizar:<br>- Listar las Vistas<br>- Obtener la estructura<br>- Leer los datos<br>- Crear vistas | |
| **Oráculo** | 1. LISTAR LAS VISTAS: Mostrar todas las vistas que pertenecen a una base de datos.<br>2. OBTENER LA ESTRUCTURA: Mostrar los campos y el tipo de dato de cada una de las vistas.<br>3. LEER LOS DATOS: Mostrar los datos que reflejan las vistas almacenadas en una base de datos especifica.<br>4. CREAR VISTAS: Crear vistas en base a código SQL generado por la API. | |
| **Pasos** | 1. LISTAR LAS VISTAS<br>- Pulsar la opción vistas del menú principal.<br>-Visualizar solo las vistas que son parte de una base de datos a la que previamente se conecto.<br>2. OBTENER LA ESTRUCTURA<br>- Seleccionar una vista.<br>- Extender los datos estructurales de la vista.<br>3. LEER LOS DATOS<br>- Visualizar los datos de una vista, ejecutando una consulta.<br>4. CREAR VISTAS<br>- Pulsar la opción herramientas de generación de código.<br>- Seleccionar la opción consultas.<br>- Generar la consulta en base a la selección de tablas y campos.<br>- Marcar la opción generar como vista.<br>- Pulsar el botón aceptar. | |
| **Módulos Asociados** | - Base de Datos<br>- Generar código SQL<br>- Ejecutar código | |

Tabla 43: Pruebas: Administración de vistas.

| Nombre | Administración de Tablas | Pruebas PC03 |
|---|---|---|
| **Propósito** | Verificar si es posible listar, obtener la estructura, leer los datos, y crear tablas, ademas de insertar, modificar, eliminar sus registros. | |
| **Prerrequisitos** | Establecer una conexión con una base de datos en MySQL. | |
| **Ubicación** | EasyProcedure y base de datos MySQL. | |
| **Entrada** | Acciones que se pueden realizar:<br>- Listar las tablas<br>- Obtener la estructura<br>- Leer los datos<br>- Crear tablas<br>- Crear tablas temporales<br>- Crear tablas con referencia a otras tablas<br>- Insertar registros<br>- Modificar registros<br>- Eliminar registros | |
| **Oráculo** | 1. LISTAR LAS TABLAS: Mostrar todas las tablas que pertenecen a una base de datos.<br>2. OBTENER LA ESTRUCTURA: Mostrar los campos y el tipo de dato, y más de cada una de las tablas.<br>3. LEER LOS DATOS: Mostrar los datos de las tablas almacenadas en una base de datos especifica.<br>4. CREAR TABLAS TEMPORALES: Crear la tabla temporal en base a código SQL generado.<br>5. CREAR TABLAS CON REFERENCIA A OTRAS TABLAS: Crear la tabla en base a código SQL generado.<br>6. INSERTAR REGISTROS: Generar código SQL para insertar registros en una tabla perteneciente a la base de datos.<br>7. MODIFICAR REGISTROS: Generar código SQL para actualizar los registros de una tabla perteneciente a la base de datos.<br>8. ELIMINAR REGISTROS: Generar código SQL para eliminar registros en una tabla perteneciente a la base de datos. | |

continua en la siguiente pagina

| | |
|---|---|
| **Pasos** | 1. LISTAR LAS TABLAS<br>- Pulsar la opción tablas del menú principal.<br>-Visualizar solo las tablas que son parte de una base de datos a la que previamente se conecto.<br>2. OBTENER LA ESTRUCTURA<br>- Seleccionar una tabla.<br>- Extender los datos estructurales de la tabla.<br>3. LEER LOS DATOS- Visualizar los datos de una tabla, ejecutando una consulta.<br>4. CREAR TABLAS TEMPORALES<br>- Pulsar la opción herramientas de generación de código.<br>- Seleccionar la opción consultas.<br>- Generar la consulta en base a la selección de tablas y campos.<br>- Marcar la opción generar como tabla temporal.<br>- Pulsar el botón aceptar.<br>5. CREAR TABLAS CON REFERENCIA A OTRAS TABLAS<br>- Pulsar la opción herramientas de generación de código.<br>- Seleccionar la opción consultas.<br>- Generar la consulta en base a la selección de tablas y campos.<br>- Marcar la opción generar como tabla.<br>- Pulsar el botón aceptar.<br>6. INSERTAR REGISTROS<br>- Pulsar la opción sintaxis.<br>- Pulsar la opción insert.<br>- Seleccionar una tabla.<br>- Generar código SQL para inserción de datos.<br>7. MODIFICAR REGISTROS<br>- Pulsar la opción sintaxis.<br>- Pulsar la opción update.<br>- Seleccionar una tabla.<br>- Generar código SQL para actualización de datos.<br>8. ELIMINAR REGISTROS<br>- Pulsar la opción sintaxis.<br>- Pulsar la opción delete.<br>- Seleccionar una tabla.<br>- Generar código SQL para eliminar datos. |
| **Módulos Asociados** | - Base de Datos<br>- Generar código SQL<br>- Ejecutar código |

Tabla 45: Pruebas: Administrar Tablas.

| Nombre | Administración de Funciones | Pruebas PC04 |
|---|---|---|
| **Propósito** | Verificar si es posible listar, argumentar y ejecutar funciones en consultas de MySQL. | |
| **Prerrequisitos** | Establecer una conexión con una base de datos en MySQL. | |
| **Ubicación** | EasyProcedure y base de datos MySQL | |
| **Entrada** | Acciones que se pueden realizar:<br>- Listar las funciones de MySQL.<br>- Listar las funciones de una base de datos.<br>- Insertar argumentos en las funciones.<br>- Ejecutar funciones de una base de datos.<br>- Obtener información sobre las funciones. | |
| **Oráculo** | 1. LISTAR LAS FUNCIONES DE MYSQL: Mostrar todas las funciones de MySQL.<br>2. LISTAR LAS FUNCIONES DE UNA BASE DE DATOS: Mostrar todas la funciones que son parte de una base de datos previamente conectada.<br>3. INSERTAR ARGUMENTOS EN LAS FUNCIONES: Se debe insertar argumentos en las funciones.<br>4. EJECUTAR FUNCIONES DE UNA BASE DE DATOS: Se debe poder ejecutar las funciones de una base de datos.<br>5. OBTENER INFORMACIÓN SOBRE LAS FUNCIONES: Se debe mostrar información sobre el uso y la utilidad de una función. | |

continua en la siguiente pagina

| | |
|---|---|
| **Pasos** | 1. LISTAR LAS FUNCIONES DE MYSQL:<br>- Las funciones se cargan previamente en el espacio de trabajo.<br>2. LISTAR LAS FUNCIONES DE UNA BASE DE DATOS:<br>- Las funciones se cargan previamente en el espacio de trabajo.<br>3. INSERTAR ARGUMENTOS EN LAS FUNCIONES:<br>- Seleccionar uno o varios argumentos del editor SQL.<br>- Doble clic derecho sobre la función a utilizar.<br>4. EJECUTAR FUNCIONES DE UNA BASE DE DATOS:<br>- Seleccionar uno o varios argumentos del editor SQL.<br>- Doble clic derecho sobre la función a utilizar.<br>5. OBTENER INFORMACIÓN SOBRE LAS FUNCIONES:<br>- Pulsar sobre la opción información del menú principal.<br>- Seleccionar una función de la lista. |
| **Módulos Asociados** | - Base de Datos<br>- Conexión |

Tabla 46: Pruebas: Administración de funciones.

| Nombre | Ejecución de Operaciones | Pruebas PC05 |
|---|---|---|
| **Propósito** | Verificar si se puede realizar las operaciones de mapeo, generación y ejecución de código SQL. | |
| **Prerrequisitos** | Conexión a una base de datos existentes en MySQL. | |
| **Ubicación** | EasyProcedure y base de datos MySQL | |
| **Entrada** | Acciones que se pueden realizar:<br>- Mapeo de vistas.<br>- Mapeo de tablas.<br>- Mapeo de Relaciones.<br>- Generación de código SQL.<br>- Ejecución de código SQL. | |
| **Oráculo** | 1. GENERACIÓN DE CÓDIGO SQL:<br>Generación de código SQL como: sentencias simples y complejas que usen el mapeo de vistas, tablas, relaciones .<br>2. EJECUCIÓN DE CÓDIGO SQL: Ejecución de código SQL que se generen para desarrollar los procedimientos almacenados. | |
| **Pasos** | 1. GENERACIÓN DE CÓDIGO SQL:<br>- Pulsar en cualquier herramienta de generación SQL.<br>- Cada herramienta desplegará una pantalla referente a la operación o utilidad de la herramienta seleccionada.<br>- Generar código SQL y adjuntar en la ubicación del cursor en el editor de código.<br>2. EJECUCIÓN DE CÓDIGO SQL:<br>- Seleccionar el editor de código SQL.<br>- Pulsar el botón ejecutar del menú del editor.<br>- Presentará una tabla de resultados de la ejecución. | |
| **Módulos Asociados** | - Tablas<br>- Vistas<br>- Funciones | |

Tabla 47: Pruebas: Operaciones.

| Nombre | Administración de Procedimientos | Pruebas PC06 |
|---|---|---|
| **Propósito** | Verificar si es posible generar procedimientos almacenados en base a las tablas existentes en una base de datos de MySQL. | |
| **Prerrequisitos** | Listar las tablas, vistas y funciones de una base de datos. | |
| **Ubicación** | EasyProcedure y base de datos MySQL. | |
| **Entrada** | Acciones que se pueden realizar:<br>- Listar los procedimientos almacenados .<br>- Obtener el código SQL de un procedimientos almacenados.<br>- Crear procedimientos almacenados simples.<br>- Crear procedimientos almacenados complejos.<br>- Verificación de existencia de los procedimientos almacenados.<br>- Validación de procedimientos almacenados.<br>- Modificar procedimientos almacenados.<br>- Eliminar procedimientos almacenados. | |
| **Oráculo** | 1. LISTAR LOS PROCEDIMIENTOS ALMACENADOS: Listar los procedimientos almacenados existentes en una base de datos.<br>2. OBTENER CÓDIGO SQL DE UN PROCEDIMIENTO ALMACENADO: Se muestra el código SQL del procedimiento almacenado.<br>3. CREAR PROCEDIMIENTOS ALMACENADOS SIMPLES: El programa debe ser capaz de crear procedimientos para operaciones como insert, update, select y delete de forma automática al seleccionar una o varias tablas.<br>4. CREAR PROCEDIMIENTOS ALMACENADOS COMPLEJOS: El programa podrá crear procedimientos almacenados con mas de una sentencia SQL validas en un Procedimiento Almacenado.<br>5. VERIFICACIÓN DE LA EXISTENCIA DE LOS PROCEDIMIENTOS ALMACENADOS: Se mostrará si el existe un procedimiento almacenado con el mismo nombre.<br>6. VALIDACIÓN DE LOS PROCEDIMIENTOS ALMACENADOS: Si existe algún error el programa lo mostrará.<br>7. MODIFICAR PROCEDIMIENTOS ALMACENADOS: Usando el punto 2 se puede modificar un procedimiento almacenado.<br>8. ELIMINAR PROCEDIMIENTOS ALMACENADOS: Elimina un procedimiento almacenado especifico. | |

continua en la siguiente pagina

| | |
|---|---|
| **Pasos** | 1. LISTAR LOS PROCEDIMIENTOS ALMACENADOS:<br>- Pulsar la opción procedimientos almacenados del menú principal.<br>-Visualizar solo los procedimientos almacenados que son parte de una base de datos a la que previamente se conecto.<br>2. OBTENER CÓDIGO SQL DE UN PROCEDIMIENTO ALMACENADO:<br>- Doble clic derecho sobre un procedimiento almacenado, el código SQL se abrirá en una pestaña nueva con un editor de código SQL.<br>3. CREAR PROCEDIMIENTOS ALMACENADOS SIMPLES:<br>- Pulsar en la opción “Herramientas” del menú principal.<br>- Seleccionar una opción de la lista, las opciones son las descritas en el homónimo de la actividad del oráculo.<br>4. CREAR PROCEDIMIENTOS ALMACENADOS COMPLEJOS:<br>- Pulsar en la opción “Herramientas” del menú principal.<br>- Desarrollar código SQL para el cuerpo del procedimiento almacenado.<br>- Libertad de selección de herramientas para generación de código SQL.<br>5. VERIFICACIÓN DE LA EXISTENCIA DE LOS PROCEDIMIENTOS ALMACENADOS:<br>- Comprobación interna realizada por la API.<br>6. VALIDACIÓN DE LOS PROCEDIMIENTOS ALMACENADOS:<br>- Operación interna realizada por la API.<br>7. MODIFICAR PROCEDIMIENTOS ALMACENADOS:<br>- Usar los pasos del ítem 2 que obtiene el código del procedimiento almacenado.<br>- Editar el código SQL, y libertad de selección de herramientas para generación de código SQL.<br>8. ELIMINAR PROCEDIMIENTOS ALMACENADOS:<br>- Pulsar sobre la función Drop.<br>- Elegir uno o varios procedimientos almacenados. |
| **Módulos Asociados** | - Tablas<br>- Vistas<br>- Funciones<br>- Generación de código SQL<br>- Ejecución de código SQL |

Tabla 48: Pruebas: Administrar Procedimientos Almacenados.

### 4.4.6. Desarrollo de bocetos para las interfaces de la aplicación de prueba

#### 4.4.6.1. Pantalla principal

En esta pantalla se encontrará los accesos a los diferentes módulos que tiene la API para la generación automática de procedimientos almacenados en MySQL. En este contexto, todos los módulos están establecidos como parte de esta pantalla.

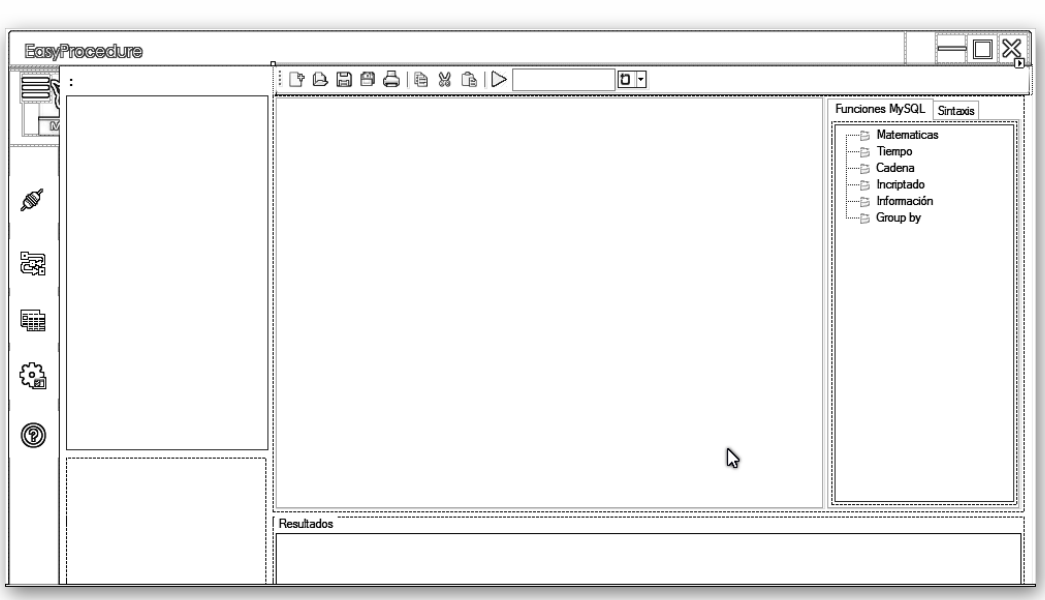



Figura 54: Bocetos: Pantalla Principal.

En los siguientes capturas se muestran los módulos que se encuentran en diferentes posiciones y secciones de la pantalla principal.

#### 4.4.6.2. Módulo: Conexión


Figura 55: Bocetos: Módulo de Conexión.

#### 4.4.6.3. Módulo: Tablas

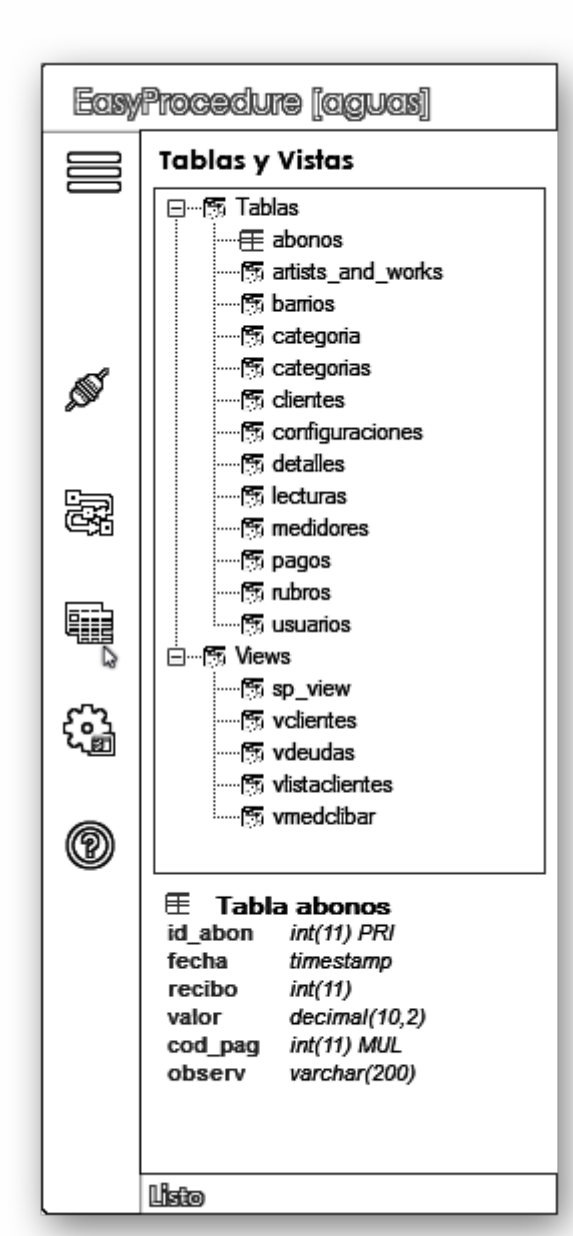



Figura 56: Bocetos: Modulo de Tablas.

#### 4.4.6.4. Módulo: Vistas

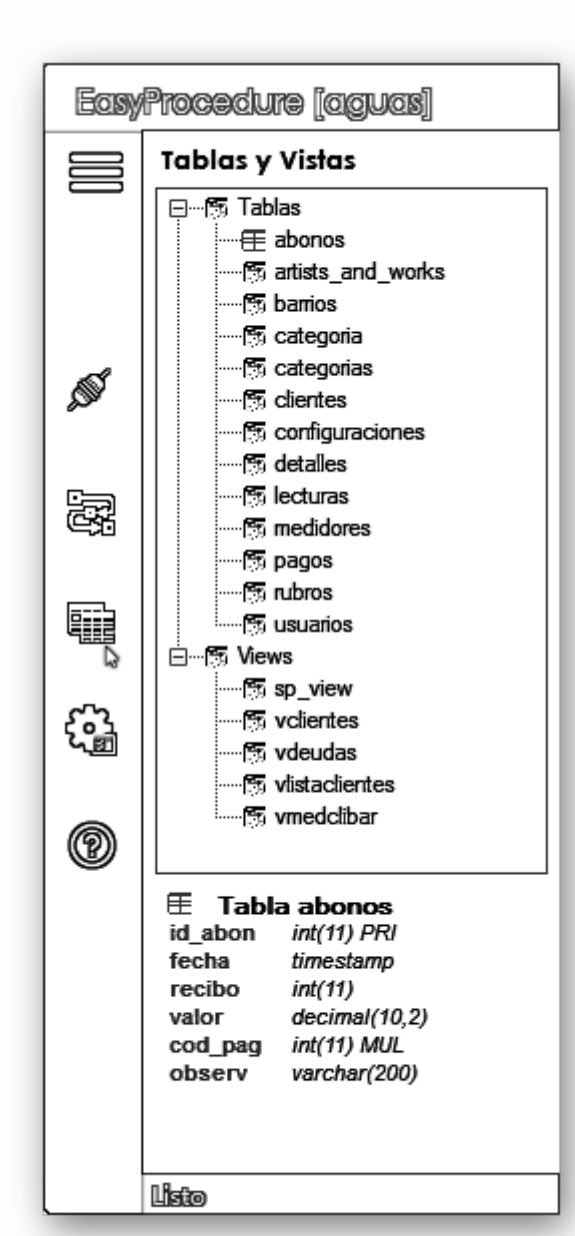



Figura 57: Bocetos: Módulo de Vistas.

#### 4.4.6.5. Módulo: Funciones


Figura 58: Bocetos: Módulo de Funciones.

#### 4.4.6.6. Módulo: Operaciones

**Pantalla para la generación de código SQL**

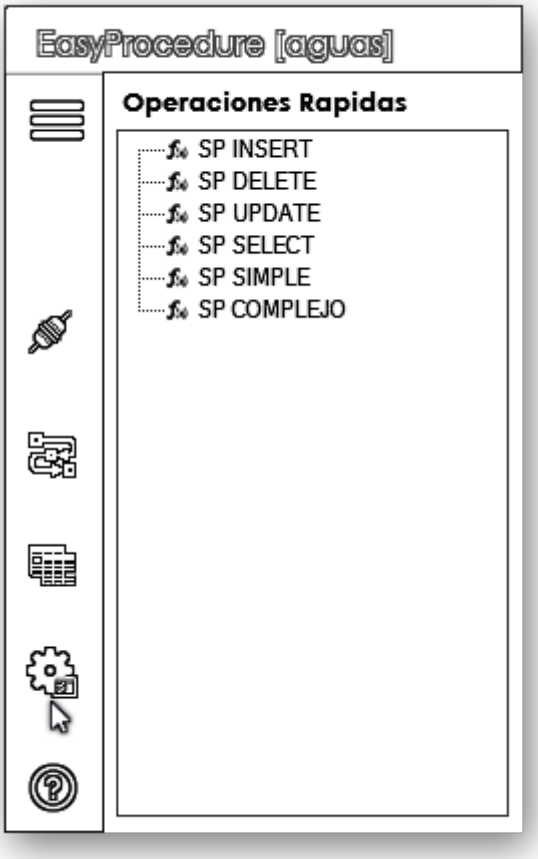



Figura 59: Bocetos: Pantalla para la generación de código.

**Pantalla para la ejecución de código SQL**

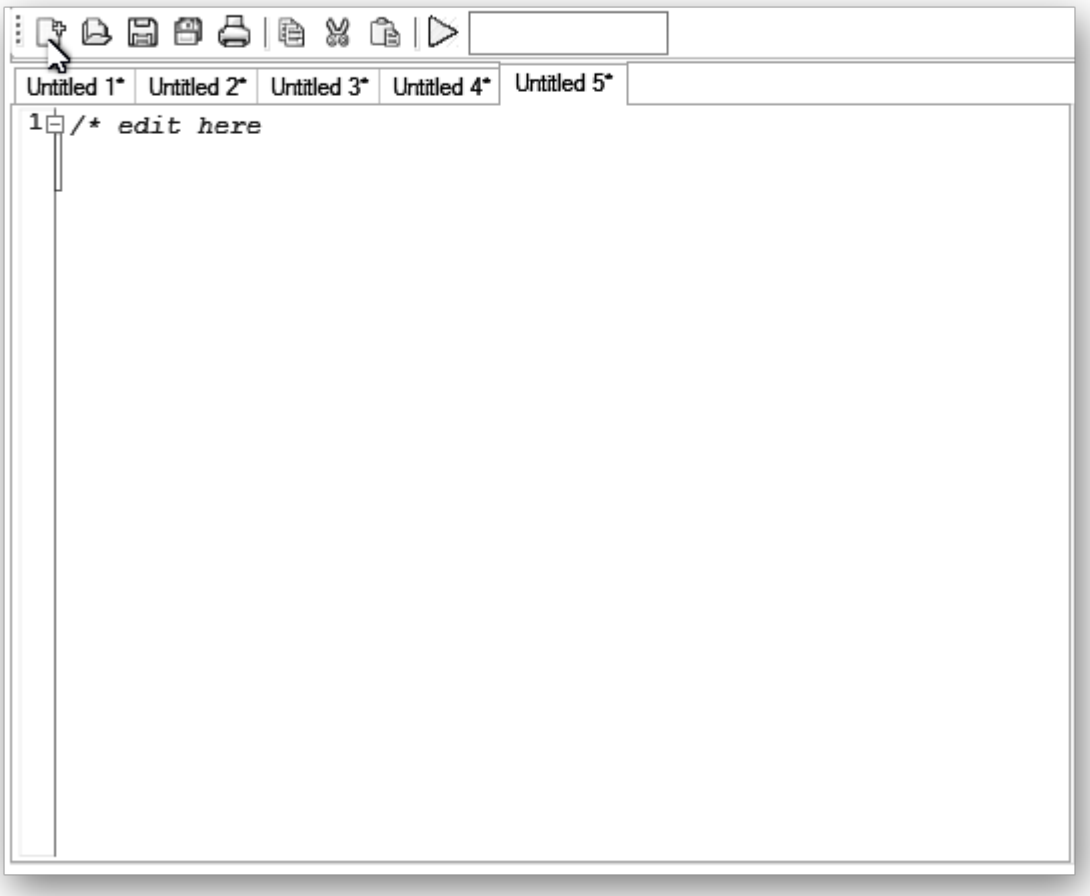



Figura 60: Bocetos: Pantalla para la generación de código SQL.

**Pantalla para la seleccionar tablas**

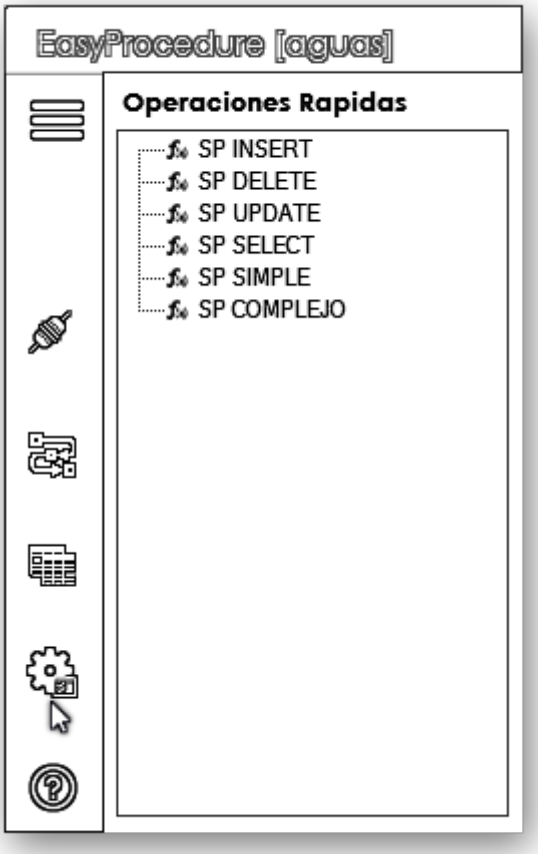



Figura 61: Bocetos: Pantalla para seleccionar tablas.

**Pantalla para generar consultas**

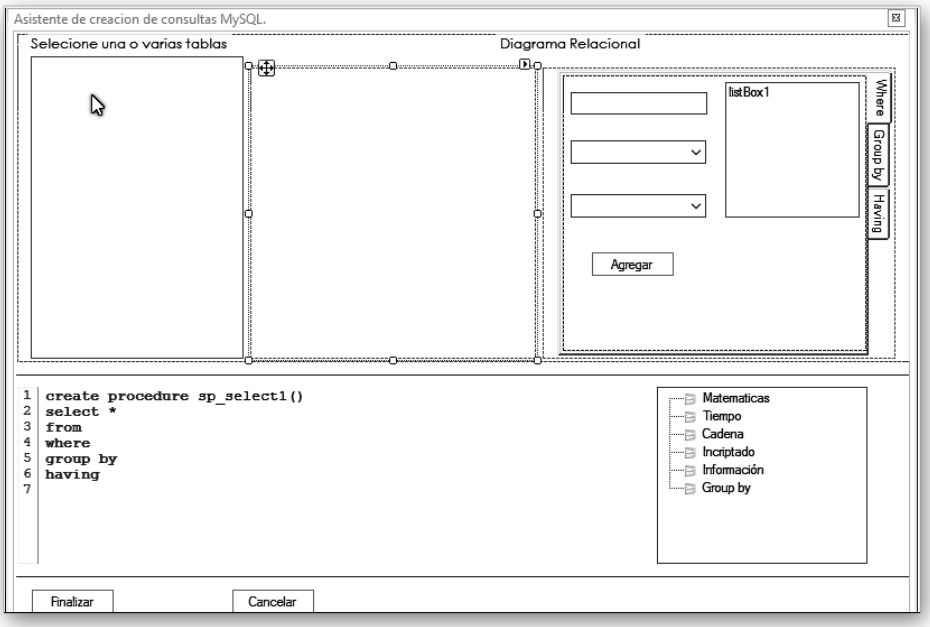



Figura 62: Bocetos: Pantalla para generar consultas.

### Pantalla de ayuda

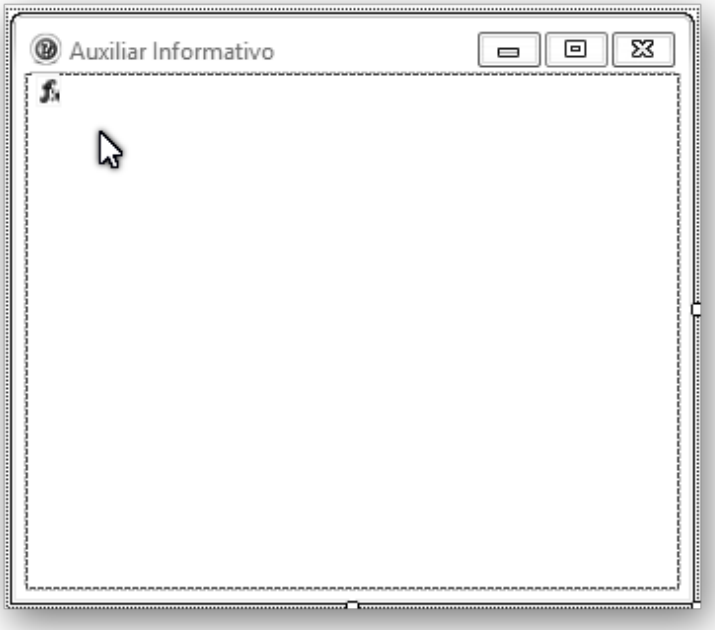



Figura 63: Bocetos: Pantalla de ayuda.

### Pantalla para la sección de resultados

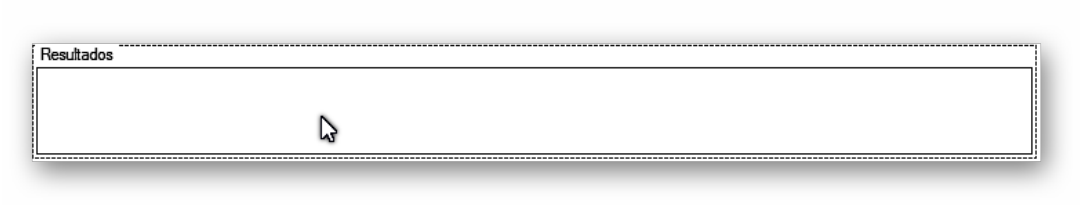



Figura 64: Bocetos: Pantalla para la sección de resultados.

#### 4.4.6.7. Módulo: Procedimientos Almacenados

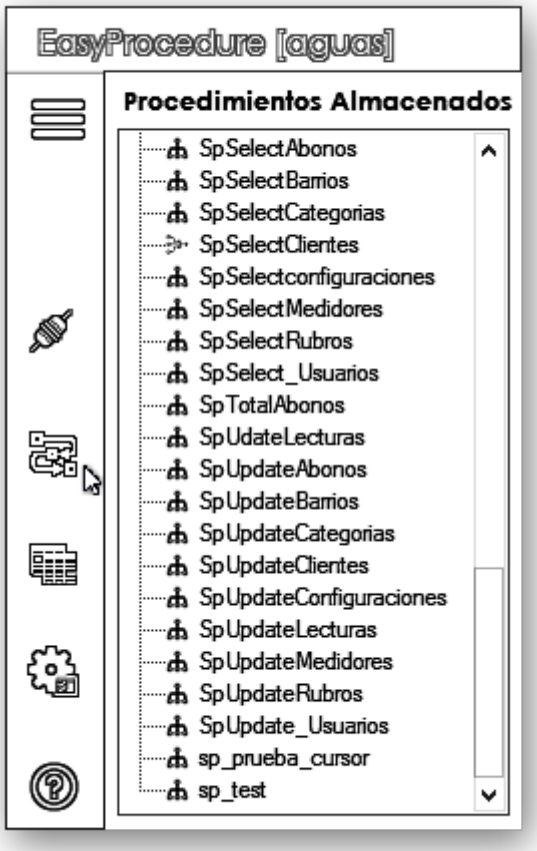



Figura 65: Bocetos: Módulo de procedimientos almacenados.

### 4.4.7. Desarrollo de las interfaces de la aplicación de prueba con Windows Forms

#### 4.4.7.1. Pantalla principal

Al igual que en el boceto presentado en puntos anteriores, la pantalla principal fue diseñada para contener una mejor accesibilidad al usuario a los módulos

y herramientas que la API para la generación automática de procedimientos almacenados en MySQL. Sin embargo este programa fue desarrollado con la finalidad de probar los módulos y operaciones que fueron programadas en la API. En este contexto se usa el escenario general de despliegue y distribución descrito en la sección 4.3.6.2 Distribución y Despliegue.

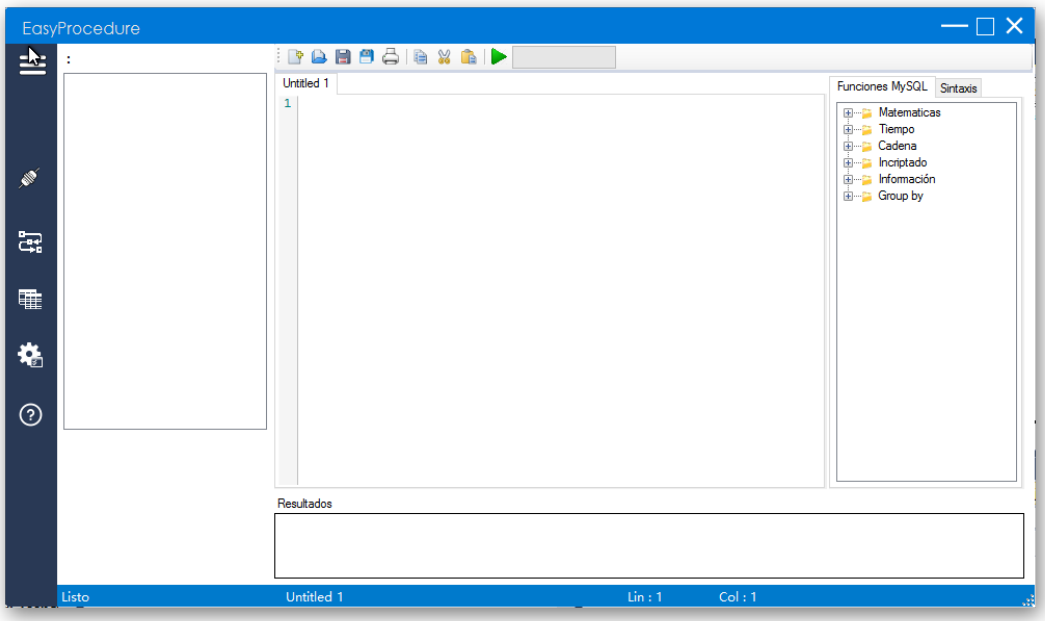



Figura 66: Interfaz: Pantalla Principal.

#### 4.4.7.2. Módulo: Conexión

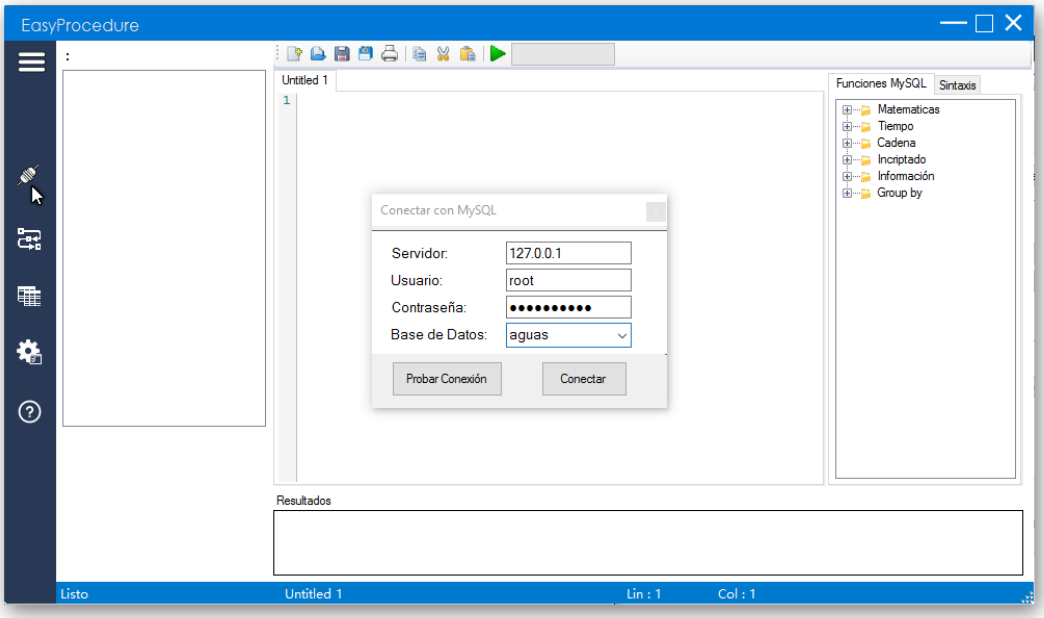



Figura 67: Interfaz: Módulo de Conexión.

#### 4.4.7.3. Módulo: Tablas

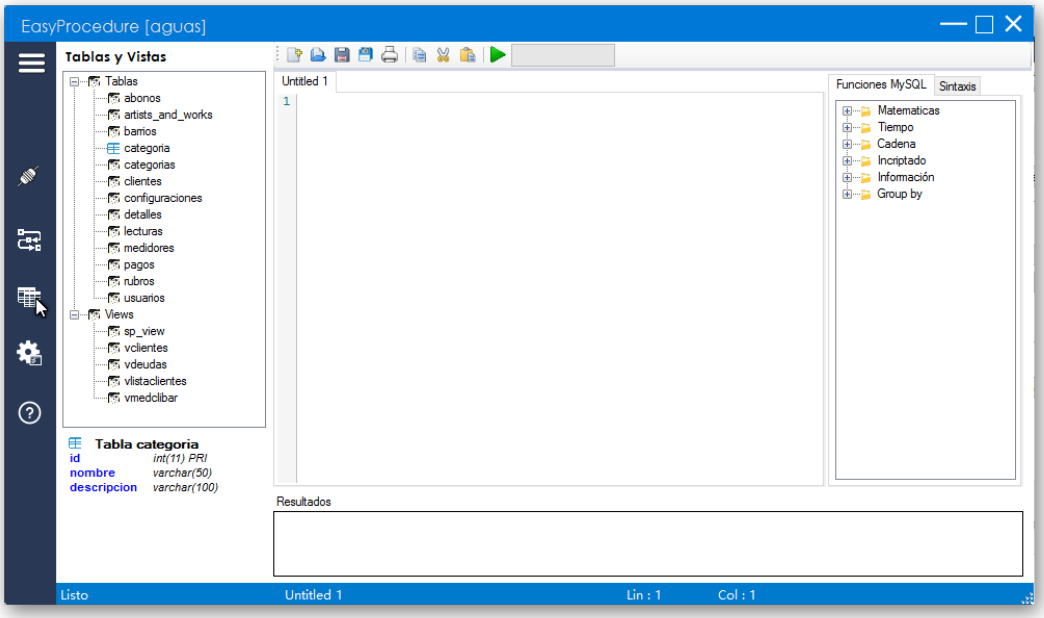



Figura 68: Interfaz: Modulo de Tablas.

#### 4.4.7.4. Módulo: Vistas

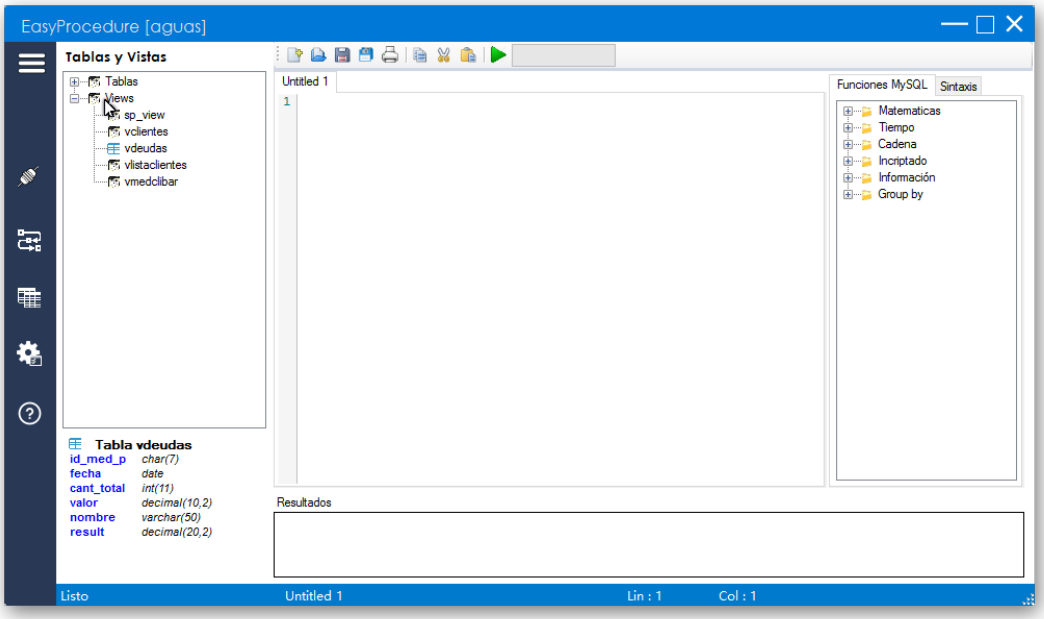



Figura 69: Interfaz: Módulo de Vistas.

#### 4.4.7.5. Módulo: Funciones

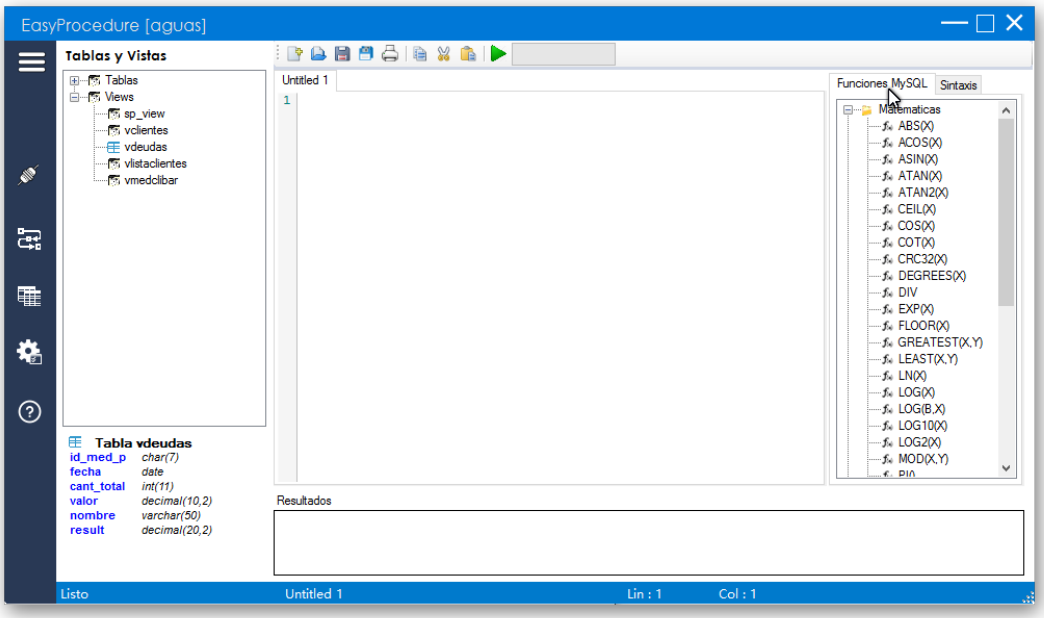



Figura 70: Interfaz: Módulo de Funciones.

#### 4.4.7.6. Módulo: Operaciones

**Pantalla para la generación de código SQL**

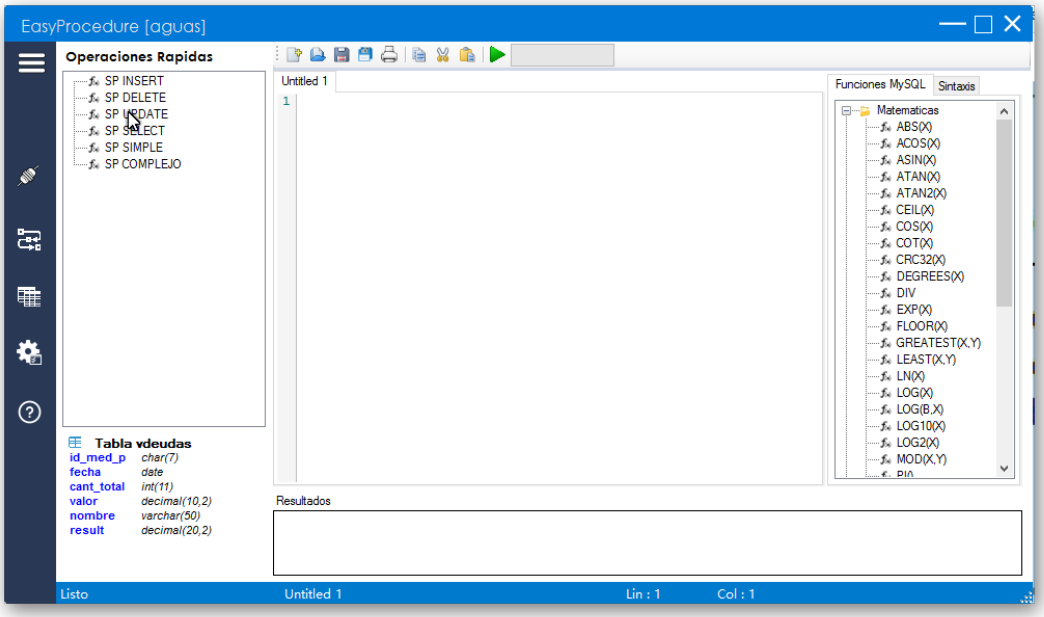



Figura 71: Interfaz: Pantalla para la generación de código.

**Pantalla para la ejecución de código SQL**

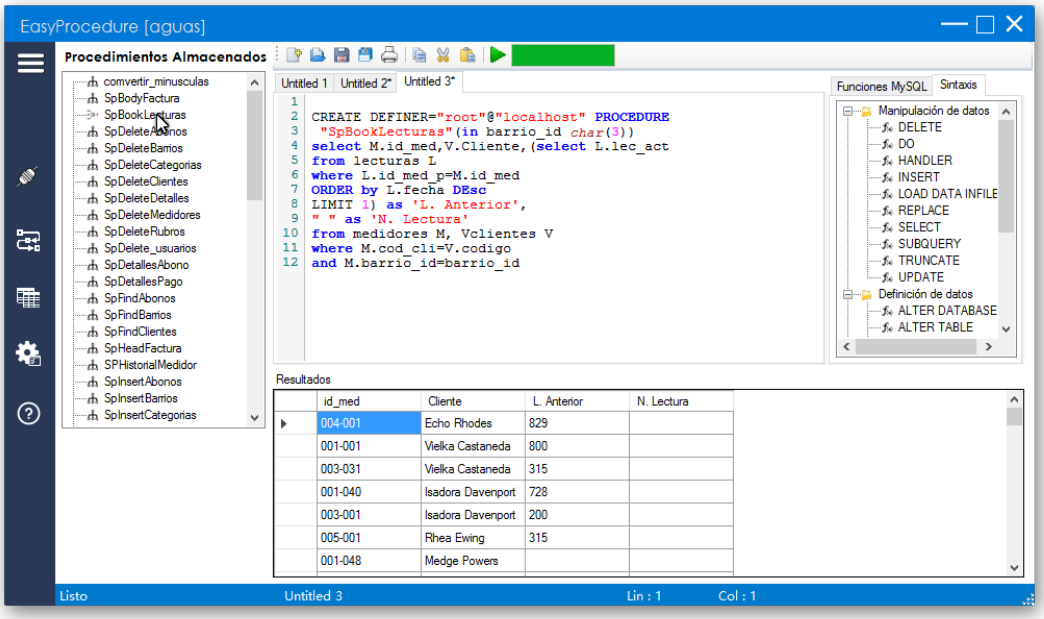



Figura 72: Interfaz: Pantalla para la generación de código SQL.

## Pantalla para la seleccionar tablas

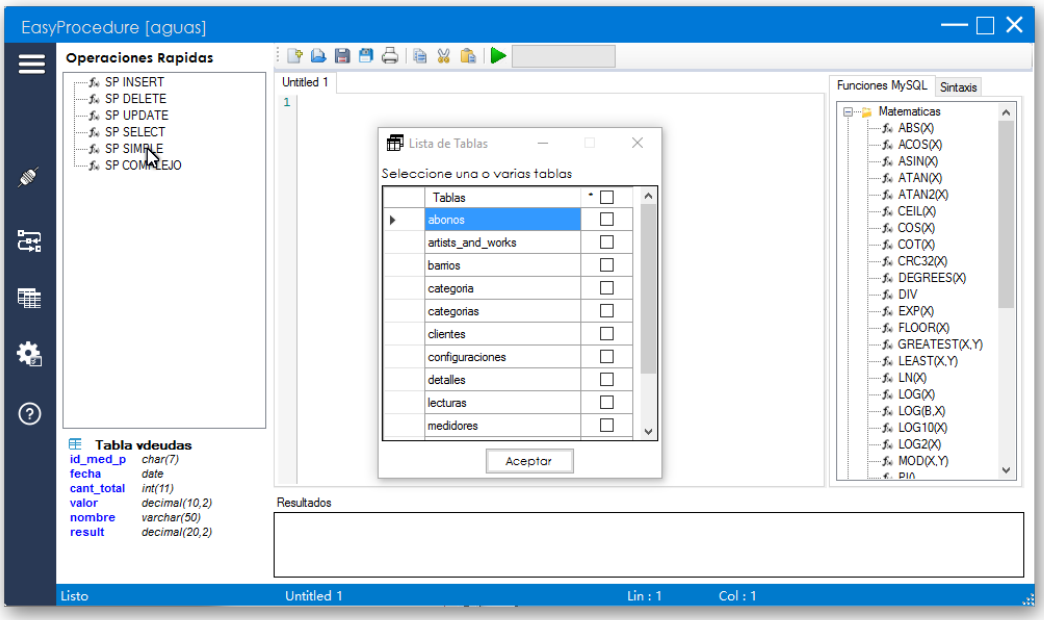



Figura 73: Interfaz: Pantalla para seleccionar tablas.

## Pantalla para generar consultas

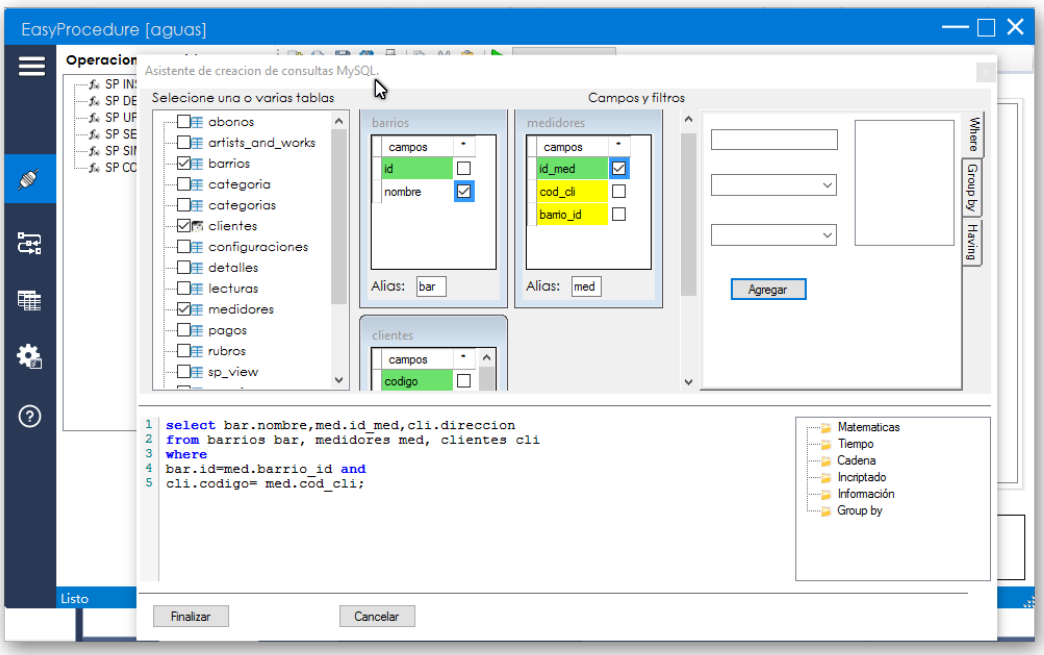



Figura 74: Interfaz: Pantalla para generar consultas.

## Pantalla de ayuda

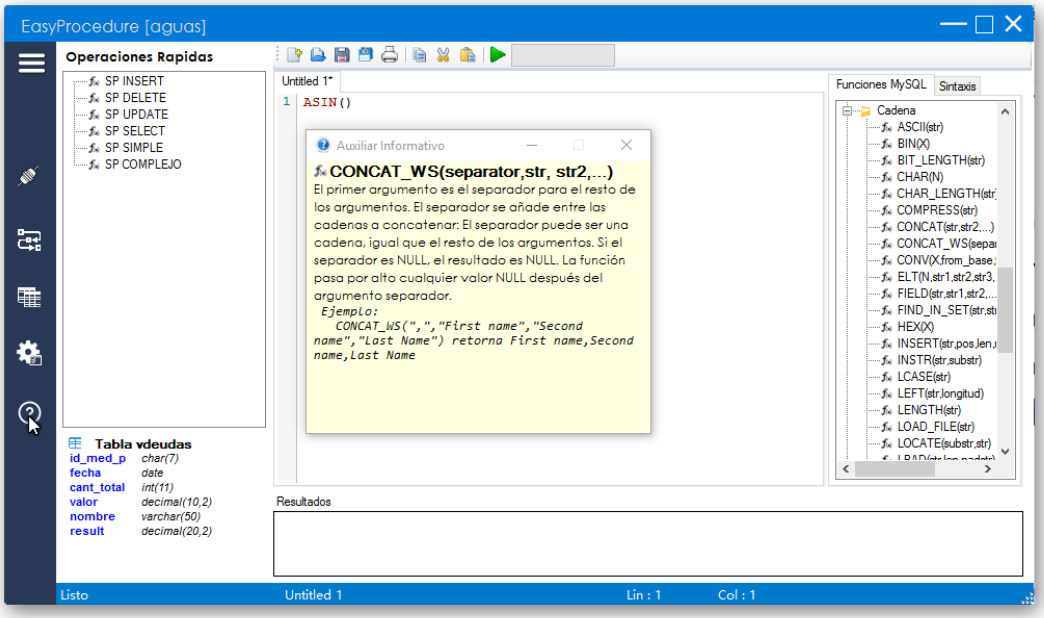



Figura 75: Interfaz: Pantalla de ayuda.

## Pantalla para la sección de resultados

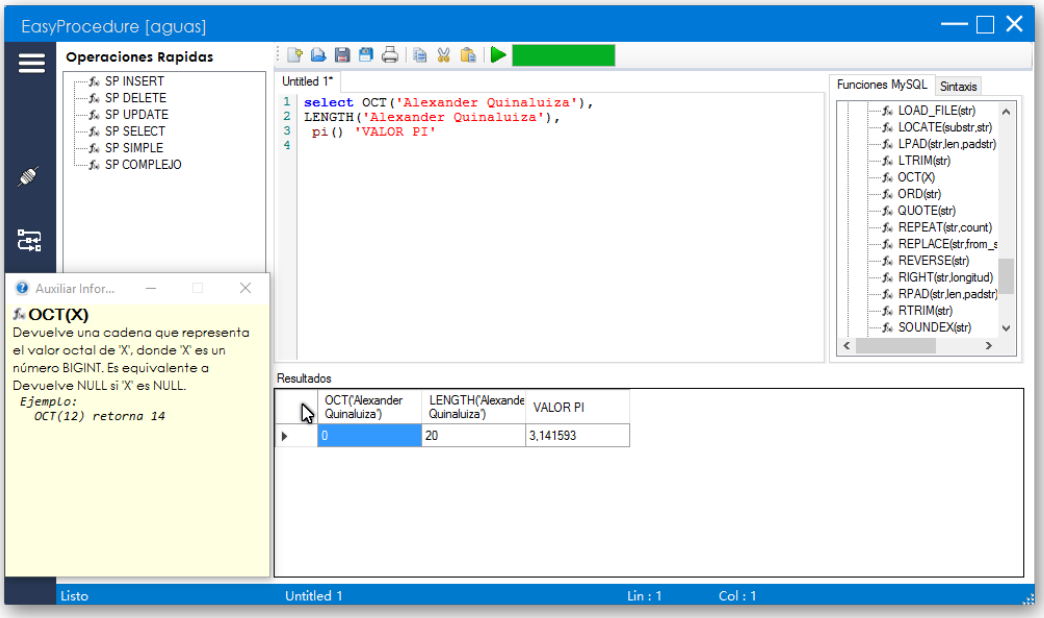



Figura 76: Interfaz: Pantalla para la sección de resultados.

#### 4.4.7.7. Módulo: Procedimientos Almacenados

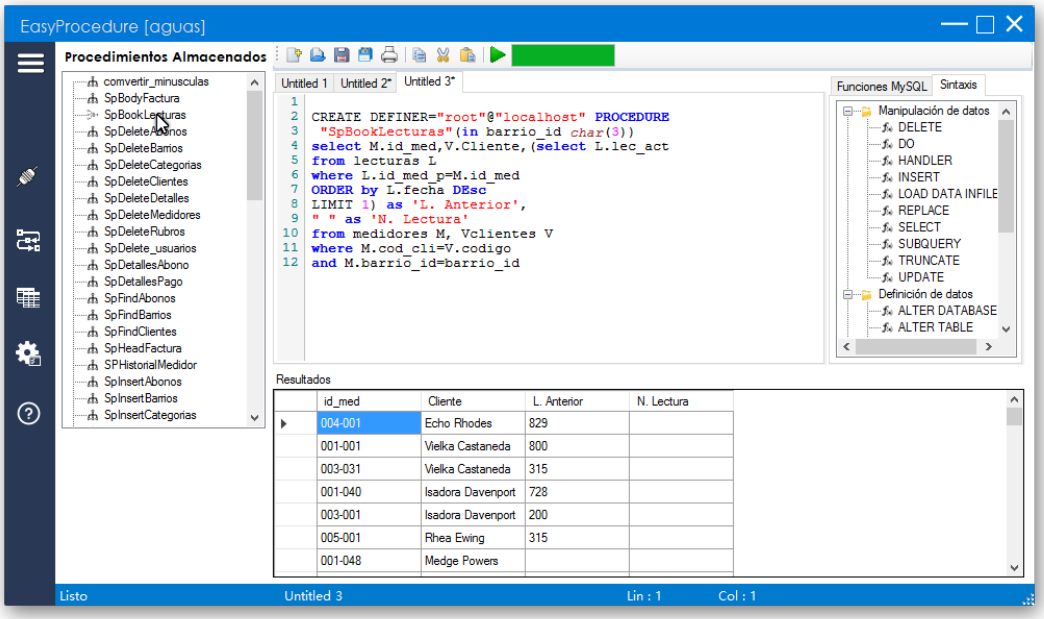



Figura 77: Interfaz: Módulo de procedimientos almacenados.

## 4.5. Instalación de la API

### 4.5.1. Requerimientos previos a la instalación de la API

#### 4.5.1.1. Servidor de base de datos

El servidor de base de datos para MySQL debe cumplir con características mínimas de hardware y software que detalla en su pagina oficial, en su versión MySQL Community Server 5.7 o superior. En este contexto, la API para la generación automática de procedimientos almacenados en MySQL es independiente de la tecnología que se use para la instalación de la base de datos. Sin embargo para este proyecto se instalo una base de datos en una maquina virtual con las siguientes características:

**Hardware**

- Memoria Ram: 2 GB
- Disco Duro: 20 GB

**Software**

- Centos 7 versión minimal
- MySQL Community Server 5.7

#### 4.5.1.2. IDE de programación

Para el desarrollo de la API y en conjunto con su aplicación gráfica de prueba, se usó el Entorno de Desarrollo Integrado (IDE) Visual Studio Community 2015. Este software para el desarrollo requiere varias características tanto de hardware como de software que se citan adelante.

**Hardware**

- Memoria Ram: 8 GB
- Disco Duro: 500 GB
- Procesador: Core i5 2.20 GHz

**Software**

- SO Windows 10 Pro
- MySQL Connect .Net
- Visual Studio Community 2015

### 4.5.2. Manual de Integración de la API

Para poder usar la API para la generación automática de procedimientos almacenados en MySQL dentro del entorno de desarrollo es necesario citar procesos fundamentales para el uso de la misma.

#### 4.5.2.1. Descargar la API

La API para la generación automática de procedimientos almacenados en MySQL esta publicada en un repositorio de acceso publico en GitHub, el mismo contiene los ensamblados necesarios para usarlo en cualquier aplicación, además de su correspondiente manual, tanto de la API, como de la aplicación de prueba desarrollada. La dirección de acceso es la siguiente: https://github.com/AlexanderQuinaluiza/EasyProcedureMySQL.git.

#### 4.5.2.2. Agregar la API en un Proyecto

Para usar la API en cualquier proyecto se debe seguir los siguientes pasos:

1. Crear un nuevo proyecto en visual studio.
2. Establecer una Conexión con MySQLConnection.

3. Referenciar la API al proyecto.

4. Instanciar la clase principal de la API.

5. Generar un procedimiento almacenado.

Una vez que se ha referenciado la API, el aprendizaje y uso de la misma será de forma facil, los datos integrados en los métodos, clases, propiedades y más objetos fueron comentados de forma que el IDE lo presente como mensajes si se tratará de una biblioteca nativa de desarrollo. A continuación se muestra un ejemplo en donde las áreas marcadas de color amarillo representan a la API usada dentro de la Aplicación de prueba llamada EasyProcedure.

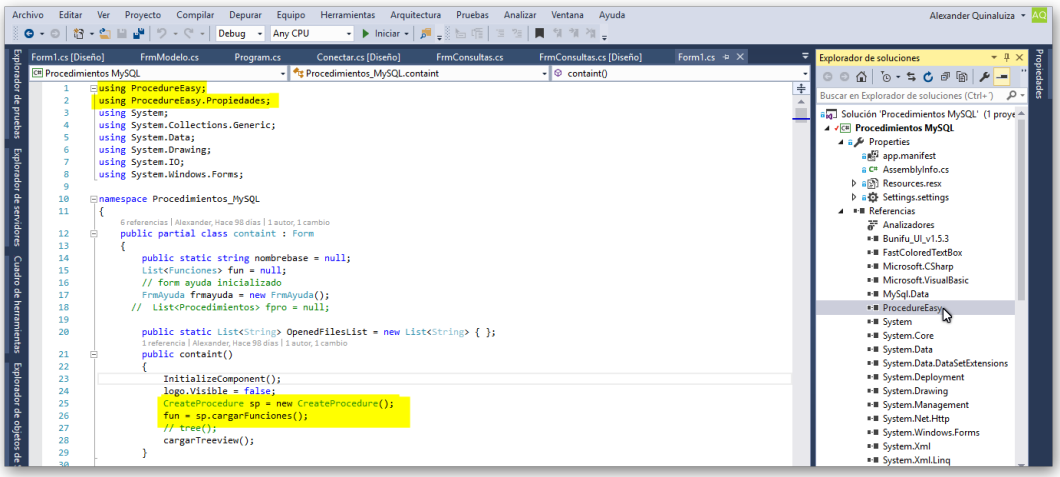



Figura 78: Referencia a la API.

## 4.6. Aceptabilidad de la Aplicación EasyProcedure

### 4.6.1. Proceso de evaluación de EasyProcedure

El proceso para determinar el grado de aceptación de la herramienta EasyProcedure, se realiza con el uso del modelo de evaluación TAM (Modelo de Aceptación de Tecnología)[28] de forma virtual, con el uso de un cuestionario en línea, a través de los formularios de Google Drive. En la siguiente figura se muestra el modelo a usar.

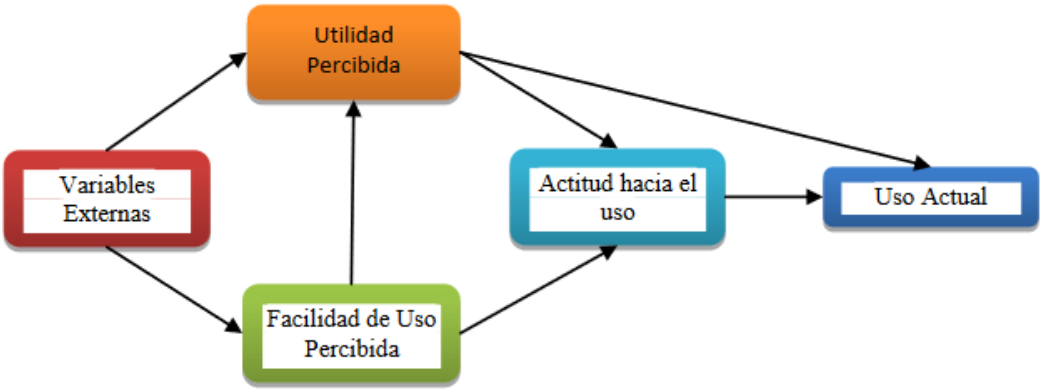



Figura 79: Modelo TAM.
Fuente: [28].

El modelo TAM sugiere usarlo cuando se presenta una tecnología nueva a los usuarios, en este caso la aplicación es EasyProcedure. En este contexto, la decisión sobre cuando y como utilizar la aplicación depende de una serie de factores, como:

- PU (Utilidad Percibida). Es el grado en que una persona cree que el uso de EasyProcedure mejora su rendimiento en el trabajo.
- FUP (Percepción de la Facilidad de Uso). Es el grado en que una persona cree que utilizando EasyProcedure, podrá liberarse del esfuerzo que le conlleva generar procedimientos almacenados de manera tradicional o con otras herramientas.

Las dos variables o factores tienen un impacto directo en la actitud de uso de la aplicación por parte de los usuarios.

### 4.6.2. Instrumento de Evaluación

Para la evaluación se aplica una encuesta a un grupo de programadores de la Facultad de Ingeniería en Sistemas Computacionales e Informáticos de la Universidad Técnica de Ambato. De acuerdo al modelo TAM la encuesta busca evaluar los siguientes aspectos: Facilidad de uso y la Utilidad percibida.

#### 4.6.2.1. Formato de los Ítems en la Encuesta

El instrumento esta formado por 10 ítems. Las repuestas para cada ítem fueron presentadas de acuerdo a la escala de Likert de cinco opciones. En la siguiente figura se presenta el formato de las preguntas y sus posibles

respuestas. La encuesta y sus resultados se encuentran en el siguiente enlace: https://goo.gl/forms/BITWF6y3ytjPgj0j2.

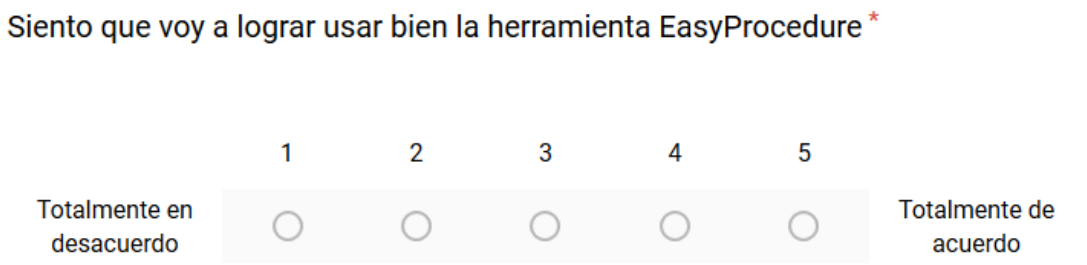



Figura 80: TAM: Formato de preguntas.

#### 4.6.2.2. Lista de Ítems de la Encuesta

En la siguiente tabla se presentan un resumen del instrumento de recolección de datos, de las dimensiones evaluadas, así como los valores máximos y esperados por cada dimensión:

| Dimensión | Ítems | Valor máximo (Likert) | Valor promedio esperado ($\mu$) |
|---|---|---|---|
| Facilidad de Uso | 5 | 25 | 3 |
| Utilidad Percibida | 5 | 25 | 3 |

Tabla 49: Dimensiones e ítems del instrumento de recolección de datos.

En la siguiente tabla se presenta los ítems que se usarán en la encuesta, cada uno de estos pertenecen a una dimensión a evaluar en la aplicación.

| Dimensión | Código | Ítems |
|---|---|---|
| Facilidad de Uso | FU01 | ¿Cree usted que esta aplicación me será complicado de utilizar? |
| | FU02 | ¿Cree usted que para utilizar esta aplicación no necesitará de ayuda? |
| | FU03 | ¿Se siente cómodo con el diseño de la aplicación? |
| | FU04 | ¿Cree usted que va lograr usar bien esta esta aplicación? |
| | FU05 | ¿Interactuar con la aplicación no requiere mucho esfuerzo mental? |
| Utilidad Percibida | UP01 | ¿Cree que trabajando con esta aplicación es más productivo? |
| | UP02 | ¿Cree que la aplicación facilitará el desarrollo de proyectos con MySQL? |
| | UP03 | ¿Si tuviera que elegir, usted seguiría usando la esta aplicación? |
| | UP04 | ¿Usaría la aplicación para generar procedimientos almacenados en MySQL? |
| | UP05 | ¿Le lleva menos tiempo crear procedimientos almacenados cuando usa la aplicación? |

Tabla 50: Dimensiones e ítems del instrumento de recolección de datos.

#### 4.6.2.3. Muestras estadísticas

En las siguientes tablas se evidencian las muestras obtenidas de la aplicación de la encuesta en una población de 12 usuarios del software EasyProcedure. La muestra es tomada de las dos dimensiones evaluadas.

| Dimensión | Ítems | $x_i$ | $(x_i - \bar{x})^2$ |
|---|---|---|---|
| Facilidad de Uso | ¿Cree usted que esta aplicación me será complicado de utilizar? | 4,4167 | 0,0101 |
| | ¿Cree usted que para utilizar esta aplicación no necesitará de ayuda? | 4,5 | 0,0003 |
| | ¿Se siente cómodo con el diseño de la aplicación? | 4,5 | 0,0003 |
| | ¿Cree usted que va lograr usar bien esta esta aplicación? | 4,5 | 0,0003 |
| | ¿Interactuar con la aplicación no requiere mucho esfuerzo mental? | 4,67 | 0,0233 |
| Datos estadísticos de la muestra | | $\bar{x}$ =4,5173 | $s$=0.0829 |

Tabla 51: Estadístico muestral sobre la Facilidad de Uso.

| Dimensión | Ítems | $x_i$ | $(x_i - \bar{x})^2$ |
|---|---|---|---|
| Utilidad Percibida | ¿Cree que trabajando con esta aplicación es más productivo? | 4,8330 | 0,0624 |
| | ¿Cree que la aplicación facilitará el desarrollo de proyectos con MySQL? | 4,0833 | 0,2500 |
| | ¿Si tuviera que elegir, usted seguiría usando la esta aplicación? | 4,4167 | 0,0277 |
| | ¿Usaría la aplicación para generar procedimientos almacenados en MySQL? | 4,6667 | 0,0070 |
| | ¿Le lleva menos tiempo crear procedimientos almacenados cuando usa la aplicación? | 4,9167 | 0,1112 |
| Datos estadísticos de la muestra | | $\bar{x}$ =4,5833 | $s$=0,3027 |

Tabla 52: Estadístico muestral sobre la Utilidad Percibida.

#### 4.6.2.4. Prueba de Hipótesis

Prueba de hipótesis de la media muestral basada en el parámetro poblacional de aceptabilidad del Software EasyProcedure tomando en cuenta la escala de Likert.

$$\mu = 3.$$

$$H_0: \bar{x} \geqq \mu.$$

$$H_1: \bar{x} < \mu.$$

$$\mu \geqq \bar{x} \leqq 5 \rightarrow \text{Se acepta } H_0: \bar{x} \geqq \mu.$$

$$\bar{x} < \mu \rightarrow \text{Se rechaza } H_0: \bar{x} \geqq \mu.$$

En este caso se tiene un nivel de confianza del 95 % de que se aceptará la hipótesis $H_0$. En este contexto el nivel de significancia es: $\alpha = 0.05$. El valor critico $t_{(0.05,4gl)}$ con 4 grados de libertad (gl) es de -2,1318.

Para este estudio se estable la siguiente regla de aceptación: Se acepta $H_0$ si (t) calculado de la dimensión de Facilidad de Uso ($t_{FUc}$) y (t) calculado de la dimensión Utilidad Percibida ($t_{UPc}$) son mayores al valor de (t) teórico ($t_T$=-2.1318) . A continuación los cálculos de (t) para cada una de las dimensiones.

$$t_{FUc} = \frac{\bar{x}-\mu}{\frac{s}{\sqrt{n}}} \; ; \; t_{UPc} = \frac{\bar{x}-\mu}{\frac{s}{\sqrt{n}}}$$

$$t_{FUc} = \frac{4{,}5173-3}{\frac{0{,}0829}{\sqrt{5}}} \; ; \; t_{UPc} = \frac{4{,}5833-3}{\frac{0{,}3027}{\sqrt{5}}}$$

$$t_{FUc} = \frac{1{,}5173}{0{,}0366} \; ; \; t_{UPc} = \frac{1{,}5833}{0{,}1337}$$

$$t_{FUc} = 41,4562 \; ; \; t_{UPc} = 11,8422$$

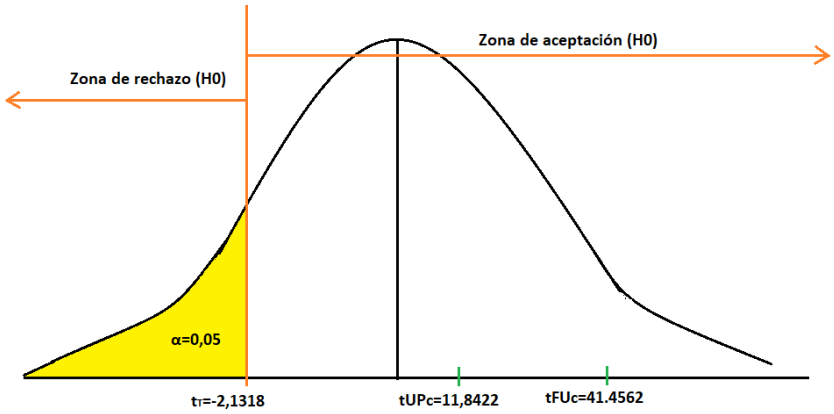



Figura 81: TAM: Distribución t-Student del estudio muestral.

La figura (81) muestra los resultados de los cálculos ($t_{FUc}$=41,4562) y ($t_{UPc} = 11,8422$) son mayores al valor (t) teórico ($t_T$=-2.1318). En base a esa regla se puede concluir que se acepta $H_0$. Por consiguiente, queda demostrado estadísticamente que la media muestral($\bar{x}$) es mayor que el valor promedio esperado ($\mu$=3). En este contexto, la muestra tomada brinda evidencia suficiente para afirmar que aplicación EasyProcedure es aceptada por los usuarios.

# CAPÍTULO 5

## Conclusiones y Recomendaciones

### 5.1. Conclusiones

- En este proyecto se evidenció que la metodología OpenUP fomenta la documentación de lo esencial para el desarrollo y para la gestión del proyecto de Software. En el caso del desarrollo de una Interfaz de Programación de Aplicaciones (API) para la generación automática de Procedimientos Almacenados en MySQL. OpenUP utiliza estándares para la generación de la documentación en cada una de sus etapas de desarrollo. Estos son: Software Planing Document OpenUp, IEEE Recommended Practice for Software Requirements Specification ANS/IEEE 830 1998, Software Architecture Document OpenUp, estándar genérico para pruebas de software ISO/ICE 19207.

- Se puede usar la arquitectura de este proyecto para aplicar a diferentes gestores de base de datos, o a su vez en la generación de nuevos módulos a la API para soporte multibase. En este contexto los alcances de la aplicación se limitan a la imaginación del programador.

- El diseño de la aplicación gráfica de prueba fue pensado en el usuario final, prestando un diseño moderno, simple, con alcance directo a las herramientas y módulos de la API que fueron puestas a prueba en la misma. En la actualidad se busca satisfacer el gusto de los usuarios finales con interfaces que integren la ingeniería y el arte, sin perder su calidad y funcionalidad en el proceso.

- Para el caso de las pruebas se desarrollo una aplicación gráfica, ya que para probar una API se necesita de otra aplicación para verificar si lo que debe hacer, lo hace bien. En este sentido se desarrolló una aplicación útil y de calidad. Esta aplicación prueba los alcances que la API puede tener al momento de generar herramientas útiles que, agilicen los tiempos de desarrollo y faciliten al programador la creación de código entendible.

- En el estudio de aceptación del software EasyProcedure que se realizó usan-

do el Modelo de Aceptación Tecnológica (TAM), se evidencia estadísticamente que la aplicación es aceptada por los programadores que la usan. Es decir que el usuario percibe la utilidad del software y la facilidad de uso. Por consiguiente se determina que el software es eficiente y que la actitud hacia el uso del mismo es positiva. El manejo de metodologías y estándares ayudan a brindar un software de calidad, pero el único que puede valorar y percibir esa calidad es el usuario final. En la siguiente figura se presenta los datos de la muestra que fueron analizados usando la distribución t-Student para la prueba de hipótesis. La misma demostró que $H_0$: $\bar{x} \geqq 3$ se acepta.

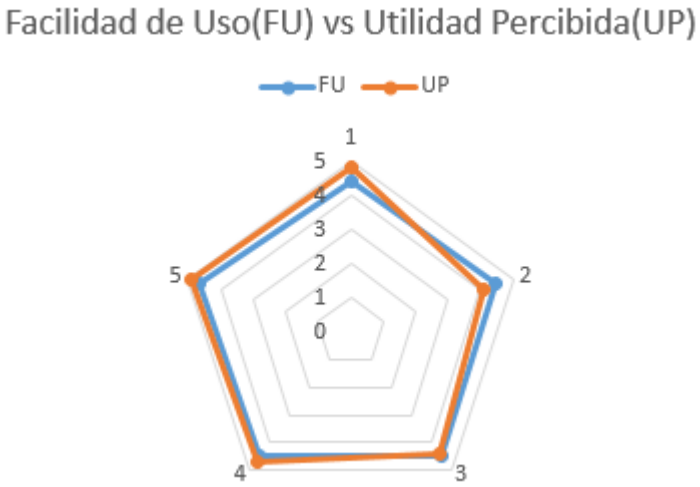



Figura 82: TAM: Facilidad de Uso vs Utilidad Percibida.

En la figura anterior se presenta la relación entre las muestras de las dos dimensiones evaluadas por el modelo TAM. Las mismas tienen un grado de significancia mínima entre ellas, y están sobre la zona de aceptación de la distribución (t-Student). Los cálculos y análisis de los datos se enceuntran en la sección de Section 4.6 subsección prueba de hipótesis.

## 5.2. Recomendaciones

- Al instalar y usar la Base de datos MySQL se recomienda usar la versión 5.6 o superior, la misma tiene mas funciones integradas al catalogo. En este contexto la aplicación gráfica cuenta con todas las funciones establecidas por MySQL en su versión más reciente.

- Para la integración de la API en un nuevo software, se recomienda usar el Framework 4.5 de .NET. Este cuenta con herramientas, controles que

facilitan el uso de la API y la integración de la misma.

- Se recomienda crear bases de datos con Constrains establecidas para indices, claves foráneas, claves primarias. Esto permite un mejor manejo de los metadatos por parte de la API.
- Se recomienda el uso de la API para bases de datos con motores de almacenamiento INNODB. Este esta diseñado para base de datos relacionales. Sin embargo se puede usar en los demás motores de almacenamiento, y generar procedimientos almacenados de ser necesarios.
- Se recomienda no usar la sentencia DATA FILE dentro de un procedimiento almacenado, este da errores, es mejor usarlas de forma independiente en conjunto con Triggers para el control de los datos.

## Bibliografia


[1] Carlos Neil, Marcelo De Vincenzi, Nicolas Battaglia, and Roxana Martínez. Herramientas colaborativas multiplataforma en la enseñanza de la ingeniería de software. In *XVIII Workshop de Investigadores en Ciencias de la Computación (WICC 2016, Entre Ríos, Argentina)*, 2016.

[2] Javier Fernández. Desarrollo de base de datos, April 2007. Available at `http://aurea.es/desarrollo-base-de-datos/`.

[3] Datanamic Solutions BV. Dezign for databases - database design and data modeling tool, 2017. Available at `http://www.datanamic.com/dezign/index.html`.

[4] EcuRed. (API) Interfaz de Programación de Aplicaciones, February 2016. Available at `https://www.ecured.cu/API`.

[5] Oracle. 10 razones para elegir MySQL para las aplicaciones web de la próxima generación, 2014. Available at `https://goo.gl/5FJdCZ`.

[6] EcuRed. MySQL, 2015. Available at `https://www.ecured.cu/Mysql`.

[7] J.L.U. Tellería. *Factores clave de dirección: orientados a la obtención de resultados*. ESIC, 2000.

[8] Carlos Mario Zapata and John Jairo Chaverra. Una mirada conceptual a la generación automática de código.(a conceptual approach to automatic generation of code). *Revista EIA*, 7(13):143–154, 2013.

[9] Francisco Javier Bermúdez Ruiz, Jesús Garcıa Molina, and Oscar Dıaz Garcıa. Db-main/models: Un caso de estudio sobre la interoperabilidad de herramientas basada en mde. *de aceptado en: Jornadas de Ingeniería del Software y Bases de Datos, Cádiz*, 2014.

[10] Ramon Lawrence. Integration and virtualization of relational SQL and NoSQL systems including MySQL and MongoDB. mar 2014.

[11] Thiti Sittivangkul, Weerasak Cheunta, Nitthita Chirdchoo, and Lunchakorn Wuttisittikulkij. Simple-api (application programming interface) for an arduino-based wireless sensor mote. In *ITC-CSCC: In-*

*ternational Technical Conference on Circuits Systems, Computers and Communications*, pages 946–949, 2015.

[12] Lionel R Baquero Hernández, Dayana Mendoza Peña, Osviel Rodriguez Valdés, and Omar Mar Cornelio. Extensión de la herramienta visual paradigm for uml para la evaluación y corrección de diagramas de casos de uso plugin of visual paradigm for uml tool for evaluation and correction of use case diagram. 2016.

[13] David Anderson and Mark Hills. Query construction patterns in php. In *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*, pages 452–456. IEEE, 2017.

[14] Julián Pérez Porto and Ana Gardey. Definición de API, 2017. Available at `http://definicion.de/api/`.

[15] Qué es una API y para qué sirve?, February 2015. Available at `http://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html`.

[16] Margaret Rouse and Rob McCormack. Qué es MySQL?, January 2015. Available at `http://searchdatacenter.techtarget.com/es/definicion/MySQL`.

[17] Oracle. MySQL, 2015. Available at `http://www.oracle.com/technetwork/database/mysql/index.html`.

[18] Oracle. *MySQL 5.0 Reference Manual*, May 2014. Available at `https://downloads.mysql.com/docs/refman-5.0-es.a4.pdf`, version 5.0.

[19] Carlos Cuenca. Tipos de Datos de MySQL, February 2003. Available at `https://desarrolloweb.com/articulos/1054.php`.

[20] Salvador Pozo. MySQL con clase, March 2015. Available at `http://mysql.conclase.net/curso/?cap=011#FUN_FLUJO`.

[21] Vele Zhingri and César Augusto. Análisis de rendimiento entre la base de datos relacional: Mysql y una base de datos no relacional: Mongodb. B.S. thesis, Universidad del Azuay, 2016.

[22] Miguel Sicilia. Visión general de la arquitectura de MySQL 5.1., 2017. Available at `https://cnx.org/contents/-jPvAviC@1/Visin-general-de-la-arquitectu`.

[23] Oracle. *MySQL Connector/Net Developer Guide*, May 2017. Available at `https://downloads.mysql.com/docs/connector-net-en.a4.pdf`.

[24] Patricio Letelier and Mª Carmen Penadés. Metodologías ágiles para el desarrollo de software: extreme programming (xp). 2006.

[25] Ophelia Pastrana. 5 beneficios de aplicar meetodologías ágiles en el desarrollo de software, 2014. Available at `https://goo.gl/dUSYGi`.

[26] EcuRed. Scrum, November 2011. Available at `https://www.ecured.cu/SCRUM`.

[27] Mathieu Fourment and Michael R Gillings. Una comparación de lenguajes de programación usados en bioinformática, 2013. Available at `https://goo.gl/fh7Krq`.

[28] Diego Leyton. Extensión al modelo de aceptación de tecnología tam, para ser aplicado a sistemas colaborativos, en el contexto de pequeñas y medianas empresas. Master's thesis, Universidad de Chile, 2013.

# Anexos y Apéndices