
# Ingeniería Superior Informática
# Sistemas Informáticos

Laboratorio virtual de consultas SQL: Implementación de la lógica de negocio.

**Alumno:** Juan Brugera Monedero

**Tutores:** Mercedes G. Merayo y Manuel Núñez

# Resumen

## Resumen Español

Tomando como base la aplicación web SQLab, a través de la cual profesores pueden proponer ejercicios de bases de datos para que sus alumnos afiancen sus conocimientos en la materia, se ha implementado la lógica de negocio de la misma con la que se incluyen los siguientes aspectos:

- La herramienta analizará la solución del profesor y comprobará, omitiendo las palabras reservadas del gestor, que términos de la misma son las tablas que el alumno ha de consultar.
- La aplicación contrastará la consulta propuesta por el alumno con la del profesor indicando su validez en caso de que coincidan los resultados o error en caso contrario.

**Palabras clave:** SQLab, SQL, Validación, Solución, Lógica Negocio, Consulta.

## Resumen Inglés

Taking the web application SQLab as starting point, where lectures can propose database exercices to their students in order to improve their knowledge in this subject, we have developed the bussines logic of the system including the following upgrades:

- The tool will analyze the lecture's solution checking what terms of it are tables that the student should include in its solution, omitting the reserved words of the manager.
- The application will contrast the query proposed by the student with the teacher's one, indicating its validity in case results match or error otherwise.

**KeyWords:** SQLab, SQL, Validation, Solution, Bussines Logic, Query.

# Índice

# 1. Introducción

En este documento se explica todo lo referente a la implementación de la lógica de negocio sobre la aplicación SQLab así como lo referente al modelo de datos que albergan dicha herramienta.

La idea de esta solución nace en los *jueces electrónicos*, sistemas diseñados por sus creadores para plantear un entorno de aprendizaje en diversos lenguajes de programación como pueden ser Java, C ó C++. Esencialmente, el objetivo inicial de estas herramientas consiste en que un sujeto da de alta una serie de problemas que el usuario ha de resolver mediante programación en el lenguaje indicado. Una vez el usuario que pretende afianzar sus conocimientos ha introducido una solución, se compara dicha solución con la propuesta por el sujeto que dio de alta el ejercicio. Al ser lenguajes de programación orientados al tratamiento de objetos, la comprobación de la solución es relativamente sencilla, ya que basta comprobar que el objeto resultante de la solución del creador del problema coincide con el de la persona que trata de dar con la misma.

Este proceso se complica notablemente si se utiliza un lenguaje como SQL, ya que hay que tener varios aspectos en cuenta que pueden alterar el resultado final, siendo este similar al que pretendemos hallar. Es por ello que hemos de considerar un sinfín de combinaciones de la solución que daremos como válidas, como todas las combinaciones de las columnas del resultado o el origen de las mismas en caso de que estas pertenezcan a campos utilizados en el cruce de una consulta del tipo Join.

Dado el reto que planteaba el desarrollo de este sistema, decidimos comenzar con el análisis de las consultas y tratar de hallar la manera de homogeneizarlas para que el veredicto sea correcto en caso de soluciones equivalentes.

Hemos tomando como base la anterior versión de la aplicación SQLab, desarrollada por el compañero José Alfonso Contreras, en la cual la comprobación de la solución se hacía de manera directa, contrastando literalmente la solución dada por el alumno con la albergada en la aplicación. Consideramos que este tratamiento no era óptimo, ya que algo tan simple como un carácter adicional que no afectara a la solución (por ejemplo, un espacio) devolvía un veredicto de incorrecto al usuario de la herramienta. Por otro lado, teniendo en cuenta la comodidad del profesorado, se han abordado también tareas como el reconocimiento automático de las tablas que intervienen en la solución para que puedan despreocuparse de indicarlas, teniendo que introducir únicamente la solución y dejando el resto a la aplicación.

En cuanto al reconocimiento de las tablas, se han desarrollado una serie de procesos extra que dividen la consulta por palabras y toma únicamente una cada vez. Consultando sobre el gestor de la base de datos la tipología de la palabra, la aplicación comprueba si el término analizado es un elemento susceptible de recibir una consulta, es decir, una vista o una tabla, y en caso afirmativo, la incluye como elemento que forma parte de la solución.

Este tratamiento se realiza al dar de alta un ejercicio o al editarlo, manteniendo tanto la funcionalidad ya presente en la versión anterior que permitía al profesor consultar las tablas de la base de datos que pueden formar parte de sus ejercicios, como sus respectivos campos y su tipología.

Por otra parte, de cara a que el alumno pueda comprobar la validez de sus soluciones, se ha implementado un módulo de validación de las mismas a través del cual, sin importar el orden de los campos o el origen de los mismos en el caso de consultas con cruces entre tablas, el alumno pueda conocer sin necesidad de consultar al profesor, si la solución propuesta es correcta, o, por el contrario, es errónea y ha de plantearla de otra manera.

De manera adicional, se han comprobado y corregido aspectos de la versión anterior, como el tratamiento de caracteres especiales, y se ha mejorado el modelo de datos sobre el que se almacenan los datos, añadiendo claves primarias (PKs) así como índices sobre las tablas para que las consultas de las que precisa la herramienta para funcionar correctamente se ejecuten en el menor tiempo posible.

El resto de esta memoria se encuentra estructurada de la siguiente manera. En el Capítulo 2 podremos encontrar la planificación del proyecto. En el 3, presente todo lo referente al diseño de la base de datos que da soporte a la aplicación web. En el Capítulo 4, se presentan las nuevas funcionalidades añadidas a la aplicación, así como una serie de corrección de errores y mejoras para el usuario. Para finalizar, en los capítulos 5 y 6 podremos encontrar las conclusiones del alumno y la bibliografía, respectivamente.

# 2. Planificación del proyecto

## 2.1 Metodología

La metodología que se ha utilizado para llevar a cabo la implantación de las nuevas funcionalidades a la aplicación SQLab ha sido la *Metodología Ágil*. Ello es así porque las *metodologías predictivas*, a lo largo de la vida del proyecto, se ha comprobado que pueden presentar determinados inconvenientes.

- El jefe de proyecto puede no tener conocimientos técnicos y dedicarse exclusivamente al control siguiendo los procedimientos establecidos y limitándose a la generación de informes, actas, diagramas de Gantt, WBS, etc.; herramientas que facilitan la gestión pero que no forman parte del objetivo del proyecto. Un jefe de proyecto con estas características no podrá participar activamente en la toma de decisiones técnicas.
- En proyectos largos, ceñirse a un plan estático puede provocar que el producto final ya no cubra la totalidad de las necesidades del cliente dado que estas han cambiado desde el inicio. Por tanto, durante el propio desarrollo del producto, es posible que se deban ampliar las características diseñadas inicialmente con tal de que no sea obsoleto antes de su salida al mercado.
- Incertidumbre: vivimos en un entorno rápido e inestable, donde cumplir el plan inicial no garantiza el éxito. La idea de “producto terminado” puede perder su sentido en determinados sectores (por ejemplo, software), dado que el producto siempre está en evolución. La capacidad de adaptación a partir de la retroalimentación e incorporación de nuevas ideas es fundamental.

El desarrollo ágil de software considera un enfoque para la toma de decisiones en los proyectos, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. Así el trabajo se realiza mediante la colaboración de equipos auto-organizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo.

Cada iteración del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, pruebas y documentación. Teniendo gran importancia el concepto de "Finalizado" (Done), ya que el objetivo de cada iteración no es agregar toda la funcionalidad para justificar el lanzamiento del producto al mercado, sino incrementar el valor por medio de "software que funciona" (sin errores).

Los valores principales de esta metodología son los siguientes:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio sobre seguir un plan.

Por otra parte, la metodología ágil se basa en los siguientes principios:

- La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Se acepta que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Se entregará software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajarán juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos ágiles promueven el desarrollo sostenible.
- Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para, a continuación, ajustar y perfeccionar su comportamiento en consecuencia.

Dado que la lógica de negocio de SQLab la desarrollar un equipo de un único integrante, se optó por la metodología ágil por los siguientes motivos:

- El resultado del trabajo que se iba realizando día a día se podía mostrar a los tutores del proyecto para recibir un feedback por su parte, indicando además los hitos alcanzados en cada desarrollo puntual.
- La prioridad de cada hito se consensuaba por ambas partes, dedicando la totalidad del intervalo entre entrega y entrega a un único hito para garantizar la integridad del resultado final.

Esto, sumado a la necesidad de realizar pequeñas modificaciones sobre la versión anterior para poder llevar a cabo la integración de las nuevas funcionalidades, supuso tal decisión de cara poder mantener un ritmo de desarrollo flexible, combinando el tiempo de cada hito entre el desarrollo de este y la adaptación de la aplicación al mismo en ocasiones que el desarrollo lo requería.

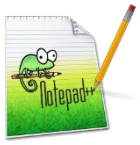
## 2.2 Recursos utilizados

Para abordar las tareas descritas en la introducción de esta memoria se han precisado de las siguientes herramientas:

- **Google Chrome:** Google Chrome es un navegador web desarrollado por Google y compilado con base en varios componentes e infraestructuras de desarrollo de aplicaciones (frameworks) de código abierto, como el motor de renderizado Blink (bifurcación o fork de WebKit). Está disponible gratuitamente bajo condiciones específicas del software privativo o cerrado. El nombre del navegador deriva del término en inglés usado para el marco de la interfaz gráfica de usuario («chrome»).

Cuenta con más de 750 millones de usuarios y, dependiendo de la fuente de medición global, puede ser considerado el navegador más usado de la Web variando hasta el segundo puesto, algunas veces logrando la popularidad mundial en la primera posición.

Por su parte, Chromium es el proyecto de software libre con el que se ha desarrollado Google Chrome y es de participación comunitaria (bajo el ámbito de Google Code) para fundamentar las bases del diseño y desarrollo del navegador Chrome (junto con la extensión Chrome Frame), además del sistema operativo Google Chrome OS.




- **Notepad++:** Programa para editar código fuente de cualquier lenguaje de programación. Como tiene soporte para una gran cantidad de lenguajes, interesará no sólo a los desarrolladores de webs, sino en general a toda la comunidad de programadores.

Notepad++ de estos editores que ofrecen ayudas muy útiles para "tirar líneas de código", como resaltado de colores, posibilidad de editar varios documentos a la vez, menús contextuales, auto-completar código, etc. Todo un regalo para los programadores, ya que además es gratuito.

En este caso, la herramienta ha sido utilizada para crear y modificar el código PHP de aquellos elementos que han requerido de cambios para implementar la nueva funcionalidad de SQLab.

- **Cygwin:** Colección de herramientas desarrollada por Cygnus Solutions para proporcionar un comportamiento similar a los sistemas Unix en Microsoft Windows.

Su objetivo es portar software que se ejecuta en sistemas POSIX a Windows con una recompilación a partir de sus fuentes. Aunque los programas portados funcionan en todas las versiones de Windows, su comportamiento es mejor en Windows NT, Windows XP y Windows Server 2003.

Se distribuye habitualmente bajo los términos de la GPL con la excepción de que permite ser enlazada con cualquier tipo de software cuya licencia esté de acuerdo con la definición de software libre. También es posible adquirir una licencia con costo para distribuirla bajo otros tipos de licencia.

Se ha requerido de Cygwin para poder establecer conexión con el servidor "Antares" alojado en la red de la Universidad Complutense de Madrid, dónde se encuentra tanto el motor de base de datos como el código de la aplicación SQLab.

- **WinSCP:** Aplicación de Software libre. WinSCP es un cliente SFTP gráfico para Windows que emplea SSH. El anterior protocolo SCP también puede ser empleado. Su función principal es facilitar la transferencia segura de archivos entre dos sistemas informáticos, el local y uno remoto que ofrezca servicios SSHNewbie.

El código fuente de WinSCP y las descargas están hospedadas en SourceForge.

Además, WinSCP tiene dos interfaces y otorga la libre elección de escoger aquella a la cual el individuo más se adapte. Ambas tienen variaciones en sus opciones de configuración y además ofrecen un alto grado de personalización.

El uso que se le ha dado a la herramienta ha sido tanto para crear y restaurar copias de seguridad de la versión anterior como para transferir los nuevos archivos PHP donde se implementan las nuevas funcionalidades adquiridas.


- **PHP:** Lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado y ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. En la actualidad, PHP se puede usar en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Este lenguaje ha sido utilizado para desarrollador los plugins nuevos que otorgan a la aplicación la mejora en la funcionalidad.

- **MySQL:** Sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo,12 y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

Se ha optado por prevalecer el uso de MySQL por su fácil integración con páginas web así con los paquetes de contenido WordPress que utilizó el autor del proyecto inicial.

# 3. Base de datos

En este capítulo se detalla todo el modelo de datos desarrollado para el correcto funcionamiento de la aplicación, así como sus claves y sus índices.

## 3.1 Tablas

### 3.1.1 Tabla Ejercicio

Es la tabla encargada de almacenar la información de los ejercicios creados por el profesor. Su estructura, así como el origen de su información, se detallan a continuación:

**NOMBRE**

- Descripción: Campo destinado a almacenar el nombre del ejercicio.
- Formato: VARCHAR(50).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-ejercicios/ -> Nombre.
- Tipificación: N/A.

**ENUNCIADO**

- Descripción: Campo destinado a almacenar el enunciado del ejercicio.
- Formato: TEXT.
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-ejercicios/ -> Enunciado.
- Tipificación: N/A.

**SOLUCIÓN**

- Descripción: Campo destinado a almacenar la solución del ejercicio.
- Formato: TEXT.
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-ejercicios/ -> Solución.
- Tipificación: N/A.

## CATEGORÍA

- Descripción: Campo destinado a almacenar la categoría del ejercicio.
- Formato: VARCHAR(25).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-ejercicios/ -> Categoría.
- Tipificación:

| VALOR | DESCRIPCIÓN |
|---|---|
| Select-Básico | La solución es del tipo SELECT FROM WHERE |
| Select-Join | La solución es del tipo SELECT FROM JOIN WHERE |
| Select-Group-Básico | La solución es del tipo SELECT FROM WHERE GROUP |
| Select-Group-Having | La solución es del tipo SELECT FROM WHERE GROUP HAVING |
| Subqueries-Valor | La solución es del tipo SELECT FROM WHERE IN |
| Subqueries-Conjuntos | La solución es del tipo SELECT FROM WHERE IN |

## NIVEL

- Descripción: Campo destinado a almacenar el nivel del ejercicio.
- Formato: VARCHAR(15).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-ejercicios/ -> Nivel.
- Tipificación:

| VALOR | DESCRIPCIÓN |
|---|---|
| Principiante | La dificultad del ejercicio es baja. |
| Intermedio | La dificultad del ejercicio es media. |
| Avanzado | La dificultad del ejercicio es alta. |

## PROFESOR

- Descripción: Campo destinado a almacenar el nombre del profesor que ha propuesto el ejercicio.
- Formato: VARCHAR(80).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-ejercicios/ -> Nombre del profesor que propone el ejercicio. Esta variable está oculta al usuario, ya que es única para cada usuario logueado con el rol de profesor.
- Tipificación: N/A.

## TABLASR

- Descripción: Campo destinado a almacenar las tablas que forman parte de la solución del ejercicio.
- Formato: VARCHAR(4000).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-ejercicios/ -> Solución
- Tipificación: N/A.
- Anotaciones: En dicho campo se almacena un objeto de PHP serializado en una cadena de texto. La obtención de la misma se genera a través de los plugins inserta.php y/o edita.php mediante los cuales se analiza el texto introducido en el campo "Solución" y se obtiene el nombre de las tablas que forman parte de la solución.
- Forma: a:N:{i:$Y_0$;s:$Z_0$:"Tabla0";…; i:$Y_{n-1}$;s:$Z_{n-1}$:"TablaN";} siendo 'N' el número de tablas almacenadas, 'Y' la posición de la tabla actual (comenzando por el 0) y 'Z' la longitud del nombre de la tabla.

## PRIMARY

- Descripción: Clave primaria de la tabla ejercicio.
- Tipo: Primary Key.
- Case Sensitive: NO.
- Campos: Nombre.

## IND_PROFESOR_EJERCICIO

- Descripción: Índice creado por profesor y ejercicio de cara a localizar ejercicios de un profesor determinado de una manera más rápida.
- Tipo: Índice.
- Case Sensitive: NO.
- Campos: Profesor, Nombre.

## 3.1.2 Tabla Hojas

Es la tabla encargada de almacenar la información de las hojas de ejercicios creadas por el profesor. Su estructura, así como el origen de su información, se detallan a continuación:

### HOJA

- Descripción: Campo destinado a almacenar el nombre de la hoja de ejercicios.
- Formato: VARCHAR(50).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-hojas/
  -> Nombre hoja
- Tipificación: N/A.

### PROFESOR

- Descripción: Campo destinado a almacenar el nombre del profesor que ha propuesto el ejercicio.
- Formato: VARCHAR(80).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-hojas/
  -> Nombre del profesor que propone el ejercicio. Esta variable está oculta al usuario, ya que es única para cada usuario logueado con el rol de profesor.
- Tipificación: N/A.

### PRIMARY

- Descripción: Clave primaria de la tabla hoja.
- Tipo: Primary Key.
- Case Sensitive: NO.
- Campos: Hoja.

### IND_PROFESOR_HOJA

- Descripción: Índice creado por profesor y hoja de ejercicios de cara a localizar hojas de ejercicios de un profesor determinado de una manera más rápida.
- Tipo: Índice.
- Case Sensitive: NO.
- Campos: Profesor, Hoja.

### 3.1.3 Tabla RelacionEjercicio

Es la tabla encargada de almacenar la solución propuesta por el usuario al ejercicio propuesto por el profesor. Su estructura, así como el origen de su información, se detallan a continuación:

**USUARIO**

- Descripción: Campo destinado a almacenar el nombre del usuario que ha resulto el ejercicio.
- Formato: VARCHAR(80).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/alumno/alumno-solucionar-ejercicio/ -> Nombre del usuario que resuelve el ejercicio. Esta variable está oculta al usuario, ya que es única para cada usuario logueado.
- Tipificación: N/A.

**EJERCICIO**

- Descripción: Campo destinado a almacenar el nombre del ejercicio.
- Formato: VARCHAR(50).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-ejercicios/ -> Ejercicio
- Tipificación: N/A.

**FECHA**

- Descripción: Campo destinado a almacenar la fecha de resolución del ejercicio.
- Formato: DATE.
- Case Sensitive: N/A.
- Nullable: NO.
- Valor: Fecha del día que se resuelve el ejercicio.
- Tipificación: N/A.

**SOLUCIONPROPUESTA**

- Descripción: Campo destinado a almacenar la solución propuesta por el usuario del ejercicio.
- Formato: TEXT.
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/alumno/alumno-solucionar-ejercicio/ -> Solución.
- Tipificación: N/A.

## VEREDICTO

- Descripción: Campo destinado a almacenar el resultado de la solución propuesta por el usuario del ejercicio.
- Formato: TINYINT.
- Case Sensitive: N/A.
- Nullable: NO.
- Tipificación:

| VALOR | DESCRIPCIÓN |
|---|---|
| 1 | CORRECTO |
| 0 | INCORRECTO |

## INTENTOS

- Descripción: Campo destinado a almacenar el número de intentos que ha realizado el usuario el ejercicio hasta dar con la solución del mismo.
- Formato: INTEGER.
- Case Sensitive: N/A.
- Nullable: NO.
- Origen: RelacionEjercicio.Intentos.
- Valor: RelacionEjercicio.Intentos + 1.
- Tipificación: N/A.

## PRIMARY

- Descripción: Clave primaria de la tabla RelacionEjercicio.
- Tipo: Primary Key.
- Case Sensitive: NO.
- Campos: Usuario, Ejercicio.

### 3.1.4 Tabla RelacionHoja

Es la tabla encargada de almacenar la relación de ejercicios que pertenece a cada hoja. Su estructura, así como el origen de su información, se detallan a continuación:

**HOJA**

- Descripción: Campo destinado a almacenar el nombre de la hoja de ejercicios.
- Formato: VARCHAR(50).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-hojas/ -> Nombre hoja
- Tipificación: N/A.

**EJERCICIO**

- Descripción: Campo destinado a almacenar el nombre del ejercicio.
- Formato: VARCHAR(50).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-ejercicios/ -> Ejercicio
- Tipificación: N/A.

**FECHA**

- Descripción: Campo destinado a almacenar la fecha en la que se incluyó el ejercicio en la hoja.
- Formato: DATE.
- Case Sensitive: N/A.
- Nullable: NO.
- Valor: Fecha del día que se incluye el ejercicio en la hoja de ejercicios.
- Tipificación: N/A.

**PRIMARY**

- Descripción: Clave primaria de la tabla RelacionHoja.
- Tipo: Primary Key.
- Case Sensitive: NO.
- Campos: Hoja, Ejercicio.

## FK_EJERCICIO

- Descripción: Clave primaria de la tabla Ejercicio.
- Tipo: Foreign Key.
- Case Sensitive: N/A.
- Campos: Ejercicio <-> Ejercicio.Nombre

## FK_HOJA

- Descripción: Clave primaria de la tabla Hojas.
- Tipo: Foreign Key.
- Case Sensitive: N/A.
- Campos: Hoja <-> Hojas.Hoja

### 3.1.5 Tabla lk_users

Es la tabla encargada de almacenar la información de los usuarios. Su estructura, así como el origen de su información, se detallan a continuación:

**ID**

- Descripción: Identificador del usuario.
- Formato: BIGINT.
- Case Sensitive: N/A.
- Nullable: NO.
- Origen: Autogenerado.
- Tipificación: N/A.

**USER_LOGIN**

- Descripción: Login del usuario.
- Formato: VARCHAR(60).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/register/ -> Usuario
- Tipificación: N/A.

**USER_PASS**

- Descripción: Password del usuario.
- Formato: VARCHAR(255).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/register/ -> Contraseña
- Tipificación: N/A.

**USER_EMAIL**

- Descripción: Email del usuario.
- Formato: VARCHAR(100).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: http://antares.sip.ucm.es/gestion_alumnos/register/ -> Correo Electrónico
- Tipificación: N/A.

## USER_URL

- Descripción: URL del perfil del usuario.
- Formato: VARCHAR(100).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: Autogenerado.
- Tipificación: N/A.

## PRIMARY

- Descripción: Clave primaria de la tabla lk_users.
- Tipo: Primary Key.
- Case Sensitive: N/A.
- Campos: ID.

## USER_LOGIN_KEY

- Descripción: Índice creado por user_login para localizar un usuario rápidamente por su login.
- Tipo: Índice.
- Case Sensitive: NO.
- Campos: user_login.

## USER_EMAIL_KEY

- Descripción: Índice creado por user_email para localizar un usuario rápidamente por su email.
- Tipo: Índice.
- Case Sensitive: NO.
- Campos: user_email.

### 3.1.6 Tabla lk_usermeta

Es la tabla encargada de almacenar la configuración de cada usuario. Su estructura, así como el origen de su información, se detallan a continuación:

#### UMETA_ID

- Descripción: Identificador del metadato.
- Formato: BIGINT.
- Case Sensitive: N/A.
- Nullable: NO.
- Origen: Autogenerado.
- Tipificación: N/A.

#### USER_ID

- Descripción: Identificador del usuario.
- Formato: BIGINT.
- Case Sensitive: N/A.
- Nullable: NO.
- Origen: Autogenerado.
- Tipificación: N/A.

#### META_KEY

- Descripción: Clave del metadato.
- Formato: VARCHAR(255).
- Case Sensitive: NO.
- Nullable: NO.
- Origen: Autogenerado.
- Tipificación: N/A.
- Anotaciones: Campos destinado a albergar distintas opciones de la configuración de cada usuario.

#### META_VALUE

- Descripción: Valor del metadato.
- Formato: LONGTEXT.
- Case Sensitive: NO.
- Nullable: NO.
- Origen: Autogenerado.
- Tipificación: N/A.
- Anotaciones: Campos destinado a albergar el valor de cada una de las distintas opciones de la configuración de cada usuario.

## PRIMARY

- Descripción: Clave primaria de la tabla lk_usermeta.
- Tipo: Primary Key.
- Case Sensitive: N/A.
- Campos: umeta_id.

## USER_ID

- Descripción: Índice creado por user_id para localizar un usuario rápidamente por su identificador de usuario.
- Tipo: Índice.
- Case Sensitive: N/A.
- Campos: user_id.

## META_KEY

- Descripción: Índice creado por meta_key para localizar el valor de un metadato rápidamente por identificador del metadato.
- Tipo: Índice.
- Case Sensitive: NO.
- Campos: meta_key.

## 3.2 Guía de buenas prácticas

De cara a garantizar un funcionamiento óptimo de la base de datos, así como facilitar un mantenimiento del mismo. A continuación se detalla una relación de buenas prácticas para aquellos usuarios que tengan la opción de alterar el mismo, ya sea mediante la creación de tablas o volcado de datos:

1. Los nombres de los objetos que se deseen albergar deberán de ir siempre en mayúsculas siguiendo los siguientes estándares:

- TABLAS: El nombre de la tabla irá siempre precedido del prefijo “T_”. Además, en la medida de lo posible, se recomienda utilizar una marca (las iniciales, por ejemplo) que identifique de manera inequívoca para evitar conflicto a la hora de crear tablas con el mismo nombre. En caso de tablas destinadas a fotos periódicas, se indicará la periodicidad de la misma al final del nombre. Ejemplos de lo indicado serían T_MGM_CONTRATO o T_MGM_CONTRATO_MES.
- VISTAS: El nombre de la vista irá siempre precedido del prefijo “V_”. Además, como en el caso anterior, se recomienda utilizar una clave que identifique de manera inequívoca para evitar conflicto a la hora de crear vistas con el mismo nombre. Un ejemplo de lo indicado sería V_MGM_CONTRATO.
- PROCEDIMIENTOS: Siguiendo la tónica de lo expuesto anteriormente, el nombre del procedimiento irá precedido del prefijo “P_”. También se recomienda el uso de una marca, así como de un nombre descriptivo del mismo. Por ejemplo, P_MGM_CONSULTA_CONTRATOS.
- FUNCIONES: Comenzarán por el prefijo “F_”, y de manera similar a los procedimientos, trataremos de que el nombre quede marcado y aporte una descripción. Ejemplo de lo expuesto podría ser F_MGM_CALCULA_SALDOS.

De manera adicional, se recuerda a los usuarios del sistema que, por motivos externos a esta aplicación, el gestor de base de datos MySQL no permite nombres de objetos con más de 30 caracteres por ser un sistema basado en ORACLE.

2. En cuanto a la creación de tablas y vistas (en el caso de que procedan) habrán de tenerse en cuenta los siguientes aspectos:

- Los nombres de las columnas irán en mayúsculas.
- Los campos pertenecientes a la PK irán al principio de la tabla, precedidos del prefijo “ID_”, tratando de ir del más general al más restrictivo, a excepción de tablas periódicas, donde el identificador del tiempo se informará al principio de la tabla. Unos ejemplos sobre las tablas mencionadas en el apartado anterior serían ID_ENTIDAD, ID_CONTRATO para la tabla aperiódica T_MGM_CONTRATO o ID_MES, ID_ENTIDAD, ID_CONTRATO para la periódica T_MGM_CONTRATO_MES.
- La PK se creará siempre en orden ascendente, desde el campo más general al más restrictivo. La PK válida de T_MGM_CONTRATO_MES sería ID_MES ASC, ID_ENTIDAD ASC, ID_CONTRATO ASC.
- De manera predeterminada, el gestor creará los campos de texto bajo la propiedad case insensitive, es decir, no distinguiendo entre mayúsculas y minúsculas. No obstante, se recomienda la inclusión de la colación deseada.

## 3.3 Lista de palabras reservadas

A continuación, se facilita una relación de palabra reservadas por el gestor MySQL y que por lo tanto el mismo nos imposibilitará utilizar para nombrar tablas, campos y objetos en general.

ZEROFILL, YEAR_MONTH, XOR, WRITE, WITH, WHILE, WHERE, WHEN, VARYING, VARCHARACTER, VARCHAR, VARBINARY, VALUES, UTC_TIMESTAMP, UTC_TIME, UTC_DATE, USING, USE, USAGE, UPDATE, UNSIGNED, UNLOCK, UNIQUE, UNION, UNDO, TRUE, TRIGGER, TRAILING, TO, TINYTEXT, TINYINT, TINYBLOB, THEN, TERMINATED, TABLE, STRAIGHT_JOIN, STARTING, SSL, SQL_SMALL_RESULT, SQL_CALC_FOUND_ROWS, SQL_BIG_RESULT, SQLWARNING, SQLSTATE, SQLEXCEPTION, SQL, SPECIFIC, SPATIA, SONAME, SMALLINT, SHOW, SET, SEPARATOR, SENSITIVE, SELECT, SECOND_MICROSECOND, SCHEMAS, SCHEMA, RLIKE, RIGHT, REVOKE, RETURN, RESTRICT, REQUIRE, REPLACE, REPEAT, RENAME, REGEXP, REFERENCES, REAL, READS, READ, PURGE, PROCEDURE, PRIMARY, PRECISION, OUTFILE, OUTER, OUT, ORDER, OR, OPTIONALLY, OPTION, OPTIMIZE, ON, NUMERIC, NULL, NO_WRITE_TO_BINLOG, NOT, NATURAL, MODIFIES, MOD, MINUTE_SECOND, MINUTE_MICROSECOND, MIDDLEINT, MEDIUMTEXT, MEDIUMINT, MEDIUMBLOB, MATCH, LOW_PRIORITY, LOOP, LONGTEXT, LONGBLOB, LONG, LOCK, LOCALTIMESTAMP, LOCALTIME, LOAD, LINES, LIMIT, LIKE, LEFT, LEAVE, LEADING, KILL, KEYS, KEY, JOIN, ITERATE, IS, INTO, INTERVAL, INTEGER, INT, INSERT, INSENSITIVE, INOUT, INNER, INFILE, INDEX, IN, IGNORE, IF, HOUR_SECOND, HOUR_MINUTE, HOUR_MICROSECOND, HIGH_PRIORITY, HAVING, GROUP, GRANT, GOTO, FULLTEXT, FROM, FOREIGN, FORCE, FOR, FLOAT, FETCH, FALSE, EXPLAIN, EXIT, EXISTS, ESCAPED, ENCLOSED, ELSEIF, ELSE, EACH, DUAL, DROP, DOUBLE, DIV, DISTINCTROW, DISTINCT, DETERMINISTIC, DESCRIBE, DESC, DELETE, DELAYED, DEFAULT, DECLARE, DECIMAL, DEC, DAY_SECOND, DAY_MINUTE, DAY_MICROSECOND, DAY_HOUR, DATABASES, DATABASE, CURSOR, CURRENT_USER, CURRENT_TIMESTAMP, CURRENT_TIME, CURRENT_DATE, CROSS, CREATE, CONVERT, CONTINUE, CONSTRAINT, CONNECTION, CONDITION, COLUMN, COLLATE, CHECK, CHARACTER, CHAR, CHANGE, CASE, CASCADE, CALL, BY, BOTH, BLOB, BINARY, BIGINT, BETWEEN, BEFORE, ASENSITIVE, ASC, AS, AND, ANALYZE, ALTER, ALL, ADD.

# 4. Aplicación web

En este capítulo procederemos a presentar todos los cambios que han sido necesarios para que SQLab incluyera las nuevas funcionalidades indicadas en la introducción de esta memoria, así como las correcciones que se han llevado a cabo para garantizar un uso estable de la herramienta.

## 4.1 Análisis de la solución: Obtención automática de los elementos que intervienen en la solución.

En este apartado, procederemos a explicar aquellos archivos PHP que han precisado de modificaciones para poder llevar a cabo esta tarea.

**inserto.php:** es el plugin encargado de dar de alta un ejercicio a través del código html de la página http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-subir-ejercicios/

A continuación, se detallan las funciones que han sido necesarias implementar para abordar este hito en pseudocódigo entendible al lector.

**function eliminaPalabrasReservadas**(string $cadena) outputs string $resultado

{

//función encargada de eliminar las palabras reservadas del gestor de la solución propuesta por el profesor

$resultado = mayúsculas($cadena);

$resultado = remplazarPalabrasReservadasPorNada($resultado);

return $resultado;

}

Como se observa en el código superior, la función eliminaPalabrasReservadas recibe un string como único argumento, devolviendo como resultado un elemento del mismo tipo.

Su función es la de aligerar la comprobación posterior sobre base de datos de los elementos que se consideran susceptibles de recibir consultas.

Para ello, eliminaremos del parámetro $cadena todas aquellas palabras que sabemos a ciencia cierta que no podrán estar dedicadas a objetos, ya que estas se encuentran reservadas por el gestor y por lo tanto no permite el uso de las mismas.

La relación de palabras que esta función elimina la podemos encontrar en el apartado 3.3 de este documento.

**Ejemplo de funcionamiento:**

*eliminaPalabrasReservadas*(“select * from T_CONTRATO”) -> “* T_CONTRATO**”**

**function analizaConsultas**(string $cadena) outputs string $resultado

{

$cadena = remplazarEspaciosPorComas($cadena);

$cadena = *eliminaPalabrasReservadas*($cadena); //elimina las palabras de la lista expuesta en 3.3.1

$array = convierteEnArray($cadena); //tomando la coma como separador

$array = unificaArray($array); //elimina los valores duplicados del array

$paso = 0; //variable para contar el número de elementos localizados

Para cada $elemento del array

{

$comprueba = 'SELECT COUNT(1) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = $elemento';

Si $comprueba = 1 entonces $elemento = serializa($elemento) y $paso++ //convierte la información del elemento en un string y añade 1 al número de elementos tratados

}

$resultado = generaCampo($paso,$elemento);

return $resultado;

}

Es la función encargada de hallar en la solución propuesta por el profesor todos aquellos elementos que son susceptibles de recibir una consulta, es decir, tablas y/o vistas.

El parámetro de entrada que recibe esta función es el resultado de ejecutar la función *eliminaPalabrasReservadas()* sobre la solución del profesor.

En primer lugar, se sustituyen todos los espacios presentes en el parámetro $cadena por comas.

Por otra parte, la función convierte la cadena de entrada en un array tomando el carácter "," como separador de los elementos.

De cara a optimizar la función, y realizar el menor número posible de consultas sobre el gestor, se unifica el array eliminando todos los valores repetidos que pudieran haber mediante.

Seguidamente, se comprueban uno a uno los elementos del array contra el gestor de base de datos, comprobando si el elemento es una tabla o una vista.

Una vez obtenidos todos ellos, solo falta generar la serialización de los objetos para almacenarlo en el campo TablasR de la tabla Ejercicios.

**Ejemplo de funcionamiento:**

*analizaConsultas*("* T_MGM_CONTRATO") -> "a:1:{i:0;s:10:"T_CONTRATO";}"

(Véase 3.1.1)

**edito.php:** es el plugin encargado de editar ejercicios dados ya de alta a través del código html de la página http://antares.sip.ucm.es/gestion_alumnos/profesor/profesor-editar/

A continuación, se detallan las funciones que ha sido necesario implementar para abordar este hito en pseudocódigo entendible al lector.

**function eliminaPalabrasReservadas**(string $cadena) outputs string $resultado

{

//función encargada de eliminar las palabras reservadas del gestor de la solución propuesta por el profesor

$resultado = mayúsculas($cadena);

$resultado = remplazarPalabrasReservadasPorNada($resultado);

return $resultado;

}

El funcionamiento de este método puede verse en el apartado anterior.

**function analizaConsultas**(string $cadena) outputs string $resultado

{

$cadena = remplazarEspaciosPorComas($cadena);

$cadena = eliminaPalabrasReservadas($cadena); //elimina las palabras de la lista expuesta en 3.3

$array = convierteEnArray($cadena); //tomando la coma como separador

$array = unificaArray($array); //elimina los valores duplicados del array

$paso = 0; //variable para contar el número de elementos localizados

Para cada $elemento del array

{

$comprueba = 'SELECT COUNT(1) FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME = $elemento';

Si $comprueba = 1 entonces $elemento = serializa($elemento) y $paso++ //convierte la información del elemento en un string y añade 1 al número de elementos tratados

}

$resultado = generaCampo($paso,$elemento);

return $resultado;

}

El funcionamiento de este método puede verse en el apartado anterior.

## 4.2 Validación de la solución: Comparación de los resultados obtenidos por la consulta propuesta y la solución real

En este apartado, procederemos a explicar aquellos archivos PHP que han precisado de modificaciones para poder llevar a cabo esta tarea.

**compruebo.php:** Es el plugin encargado de validar que la consulta propuesta por el profesor y por el alumno en el campo solución son equivalentes. Se ejecuta en la página http://antares.sip.ucm.es/gestion_alumnos/alumno/alumno-ejercicios/

Para abordar tal tarea, se han precisado de las siguientes funciones cuya lógica se redacta en pseudocódigo entendible al lector.

**function cuentaarraycampos**(string $consulta) outputs integer $resultado

{

$posinicial = posición('select',$consulta) //devuelve la posición de la palabra select

$posfinal = posición('from',$consulta) //devuelve la posición de la palabra from

$campos = subcadena($consulta, $posinicial,$posfinal) //devuelve la relación de campos de la consulta

$array = convierteEnArray($campos) //convierte el string separado por comas en un array

return cuentaElementos($array) // devuelve el número de elementos del array

}

Es la función encargada de comprobar cuantas columnas devuelve una consulta.

Su funcionamiento es sencillo:

1º Obtenemos las posiciones de las palabras "select" y "from" en la consulta.

2º Una vez halladas esas posiciones, nos quedamos con la subcadena resultante desde la posición menor (la del "select") hasta la posición final (la del "from").

3º Convertimos la cadena en un array tomando el carácter "," como separador de los elementos y devolvemos el número de elementos de dicho array.

**Ejemplo de funcionamiento:**

*cuentaarraycampos*("SELECT ID_ENTIDAD, ID_CONTRATO FROM T_CONTRATO") -> 2

**function filasconsultabbdd**(string $consulta) outputs integer $resultado

{

return numeroFilasConsulta($consulta) //devuelve el número de filas de la consulta

}

El método es simple.

Toma una consulta como parámetro de entrada y comprueba el número de filas que devuelve la misma.

**Ejemplos de funcionamiento:**

*filasconsultabbdd*(“select 1 from dual”) -> 1

*filasconsultabbdd*(“select 1 from dual union select 3 from dual”) -> 2

**function reordenacampos**(string $solucion) outputs string $resultado

{

$posinicial = posición(‘select’,$consulta) //devuelve la posición de la palabra select

$posfinal = posición(‘from’,$consulta) //devuelve la posición de la palabra from

$campos = subcadena($consulta, $posinicial,$posfinal) //devuelve la relación de campos de la consulta

$array = convierteEnArray($campos) //convierte el string separado por comas en un array

$array = ordenarArray($array) //devuelve un array ordenador por orden alfabético. En caso de consultas con cruces, se tomará como referencia el literal a partir del punto del alias de la tabla.

$resultado = “select”.convierteEnString($array).subcadena($solución,$posfinal) //genera la consulta nuevamente con los campos ordenados

return $resultado

}

Sin duda, este es el más complicado de los métodos. Debido a que, consultas equivalentes cuyos campos presenten un orden distinto han de darse como válidas, surge la necesidad de ordenar los campos de la solución. Además, como dificultad añadida, se ha de tener en cuenta la existencia de alias.

El método empleado para desarrollar esta función es el siguiente:

1º Obtenemos las posiciones de los términos “select” y “from”.

2º A partir de tales posiciones, obtenemos la subcadena resultante del parámetro $solución.

3º Convertimos la cadena en un array y lo ordenamos, teniendo en cuenta que en caso de existencia de prefijos, estos no han de tomarse en consideración.

4º Transformamos nuevamente el array a un string y generamos la consulta nuevamente con las columnas ordenadas.

**Ejemplo de funcionamiento:**

*reordenacampos*(“select A.ID_ENTIDAD, ID_CONTRATO, TITULAR from T_CONTRATO A” ) -> “select ID_CONTRATO, A.ID_ENTIDAD, TITULAR from T_CONTRATO A”

**function validaSolución**(string $solucionA, $solucionP) outputs integer $veredicto

{

$numColProfe = *cuentaarraycampos*($solucionP) //número de columnas de la consulta del profesor

$numColAlumno = *cuentaarraycampos*($solucionA) //número de columnas de la consulta del alumno

Si $numColProfe = $numColAlumno entonces

{

$numFilProfe = *filasconsultabbdd*($solucionP) //número de filas de la consulta del profesor

$numFilAlumno = *filasconsultabbdd*($solucionA) //número de filas de la consulta del alumno

$consultaFinal = union(reordenacampos($solucionP), reordenacampos($solucionA)) // genera una consulta tipo unión entre la solución del profesor y la del alumno

$numFilFinal = *filasconsultabbdd*($consultaFinal) //número de filas resultante de la unión de ambas consultas

Si numFilProfe = numFilAlumno = numFilFinal entonces return 1 sino 0

}

Sino return 0

}

Es la función principal del plugin y se encarga de comprobar que las soluciones propuestas por profesor y alumno son equivalentes.

Consta de dos partes:

1º Se comprueba, mediante la función *cuentaarraycampos()*, que ambas consultas tengan el mismo número de columnas. En caso afirmativo se realiza el paso 2. Por el contrario, la función devuelve 0.

2º Se comprueba mediante la función *filasconsultabbdd()* que tanto el número de filas de ambas consultas como la unión de las dos coincida. En caso afirmativo la función devuelve 1. Por el contrario, la función devuelve 0.

**Ejemplo de funcionamiento:**

*validasolucion*(“select ID_ENTIDAD, ID_CONTRATO from T_CONTRATO”,

“select ID_CONTRATO, ID_ENTIDAD from T_CONTRATO”) -> 1

*validasolucion*(“select ID_ENTIDAD, ID_CONTRATO from T_CONTRATO”,

“select ID_CONTRATO, TITULAR from T_CONTRATO”) -> 0

## 4.3 Optimización de la aplicación

A continuación, explicaremos brevemente cada una de las modificaciones que se han realizado sobre la aplicación para garantizar un óptimo funcionamiento de la misma:

### 4.3.1 Tratamiento de caracteres especiales

En el transcurso del desarrollo de los apartados 4.1 y 4.2 de esta memoria, se pudo comprobar que el funcionamiento de la aplicación no era el adecuado al manejar caracteres especiales como, por ejemplo, las comillas simples de los campos de texto o los saltos de línea.

Es por ello que mediante la función *stripslashes_deep()* se han corregido todos los campos que son susceptibles de presentar dicho error, como el cuadro destinado para que el profesor introduzca el enunciado del ejercicio o cualquier usuario la solución del mismo.

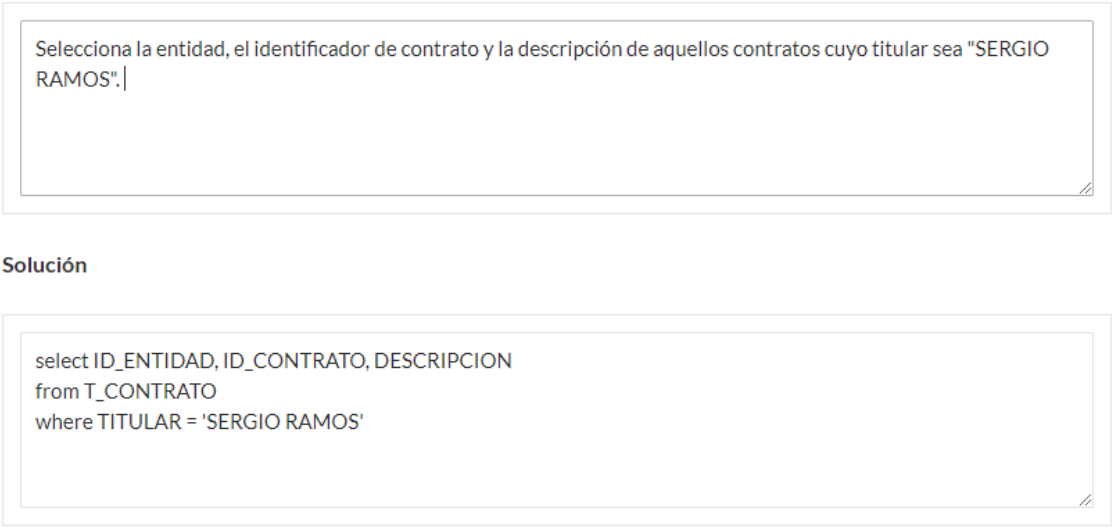
Mediante dicha función, se fuerza la inserción de los caracteres anteriormente mencionados en lugar del valor interpretable por el sistema del mismo (ej.: ‘ -> \’) garantizando la inserción y recuperación de las cadenas de texto tal y como las introdujo el usuario.

A continuación, se adjuntan capturas de pantalla de lo explicado para ilustrar la corrección.

**Antes:**


*Cuadros de Enunciado y solución sin tratamiento de caracteres especiales.*

**Ahora:**




*Cuadros de Enunciado y solución con tratamiento de caracteres especiales.*

Como se puede comprobar, subsanar esta incidencia ha sido necesario por dos motivos.

En primer lugar, a nivel de usuario, para que este no tenga que estar eliminando los caracteres especiales cada vez que decida tratar de resolver el ejercicio nuevamente, ya que los sucesivos intentos por parte del mismo generaban cadenas del tipo \\\…’ debido a la existencia del carácter original en el propio símbolo entendible por el sistema. Además, a nivel visual, resultaba molesto.

Por otra parte, de cara a la comprobación de los resultados, fue necesaria también dicha optimización, ya que, al tratar de lanzar la consulta de la solución sobre la base de datos, esta viajaba con los guiones y provocaba un error de sintaxis en la misma.

## 4.3.2 Comprobación de la solución correcta

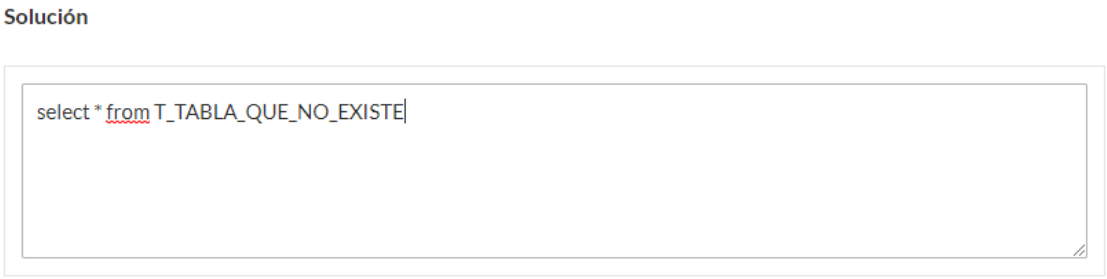
Además de la funcionalidad añadida y explicada en el punto 4.1, se ha añadido una comprobación preliminar que evita que el profesor almacene la solución del ejercicio sin haber comprobado previamente que la sintaxis de la solución, así como el correcto funcionamiento de la misma (es decir, que existan todos los campos y tablas de los que precisa), sea el adecuado.

Dicha comprobación se realiza mediante el uso de la función predefinida *query()* de la clase wpdb, objeto mediante el cual se realizan las conexiones a base de datos y el manejo de la misma.

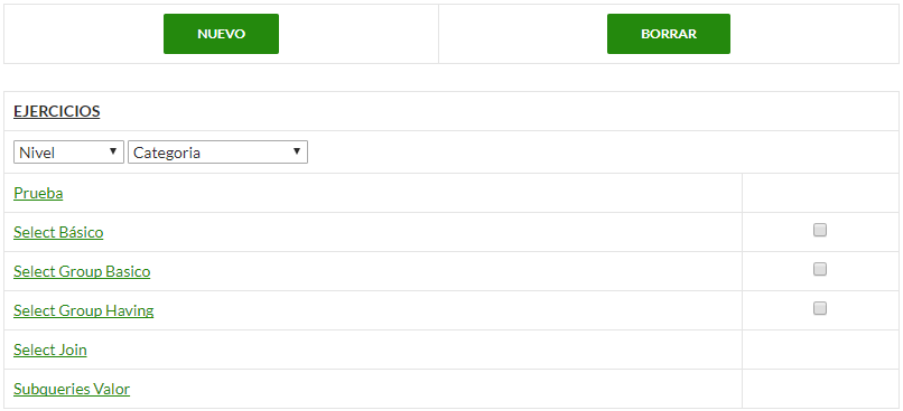
En el caso de que la solución propuesta por el profesor no sea correcta, se le informará con una advertencia en una ventana emergente y, además, permanecerá en la página del ejercicio para evitar la navegación extra por la aplicación.

Para lograr una mejor comprensión de esta mejora, se ilustra a continuación:

**Antes:**



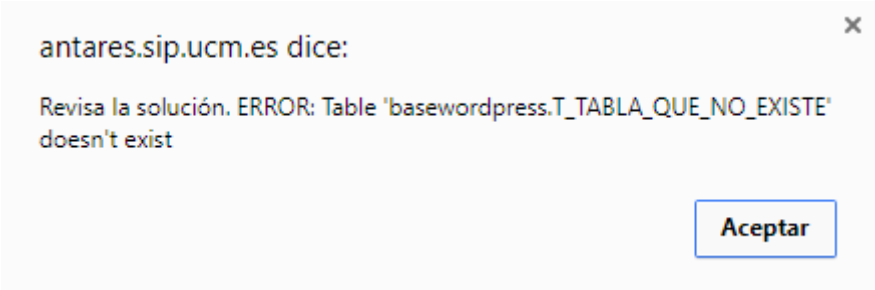

*Al dar de alta/editar un ejercicio.*




*Al haber insertado el ejercicio erróneo, la aplicación nos lleva a la ventana de los ejercicios para operar con ellos.*

**Ahora:**


*Al dar de alta/editar un ejercicio*




*Resultado del ejercicio erróneo.*


*Tras tratar de insertar el ejercicio erróneo, la aplicación nos mantiene en la misma ventana para su edición.*

## 4.3.3 Comprobación de la solución propuesta

De manera similar a lo expuesto en el punto anterior, se ha incluido una comprobación similar para cuando el usuario trata de resolver el ejercicio.

Antes de contrastar la solución propuesta por el usuario con la correcta, se analiza la primera de estas para comprobar la validez de la misma.

En caso de que la solución introducida sea inválida, se informará al usuario de ello, se le mantendrá en la ventana actual y, además, no se le dará veredicto de incorrecto a la consulta propuesta.

Nuevamente, de cara a facilitar la comprensión del lector, se ilustra con un ejemplo la optimización con el ejercicio *Select-Básico.*

**Enunciado**

Selecciona la entidad, el identificador de contrato y la descripción de aquellos contratos cuyo titular sea "SERGIO RAMOS".

La solución es " select ID_ENTIDAD, ID_CONTRATO, DESCRIPCION from T_CONTRATO where TITULAR = 'SERGIO RAMOS' " así como todas las variantes de esta.

| TABLA | COLUMNAS | |
|---|---|---|
| T_CONTRATO | ID_ENTIDAD | int(11) |
| | ID_CONTRATO | int(11) |
| | TITULAR | varchar(100) |
| | COTITULAR | varchar(100) |
| | DESCRIPCION | text |
| | FECHA_MODIFICACION | timestamp |

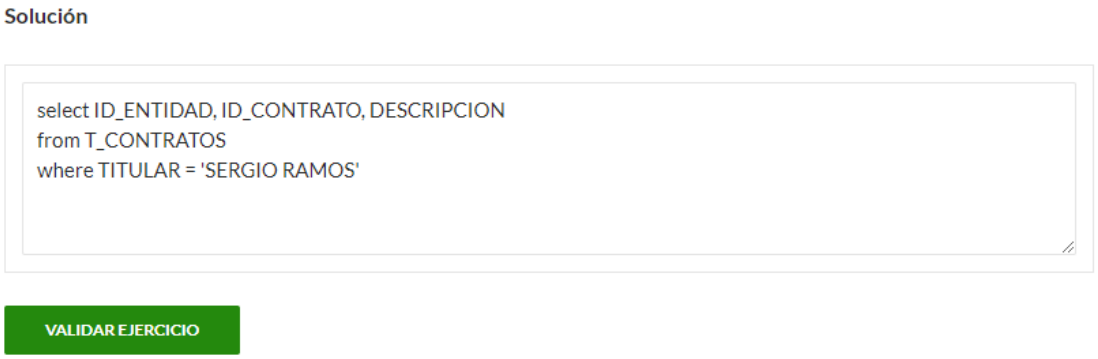
*Enunciado y Tablas del ejercicio. Se incluye solución para facilitar la ilustración.*

**Solución**

```
select ID_ENTIDAD, ID_CONTRATO, DESCRIPCION
from T_DATOS_CONTRATO
where TITULAR = 'SERGIO RAMOS'
```

VALIDAR EJERCICIO

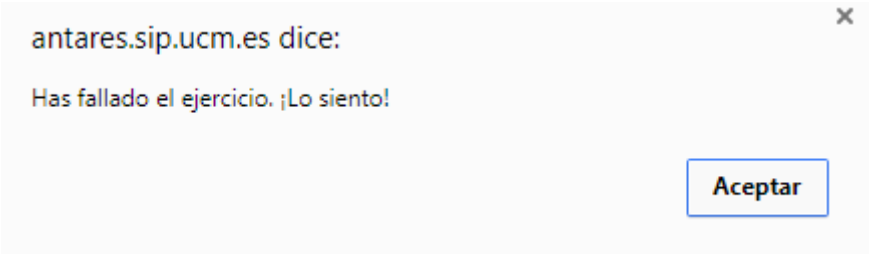
*Solución errónea. La tabla seleccionada no es la correcta, pero existe.*



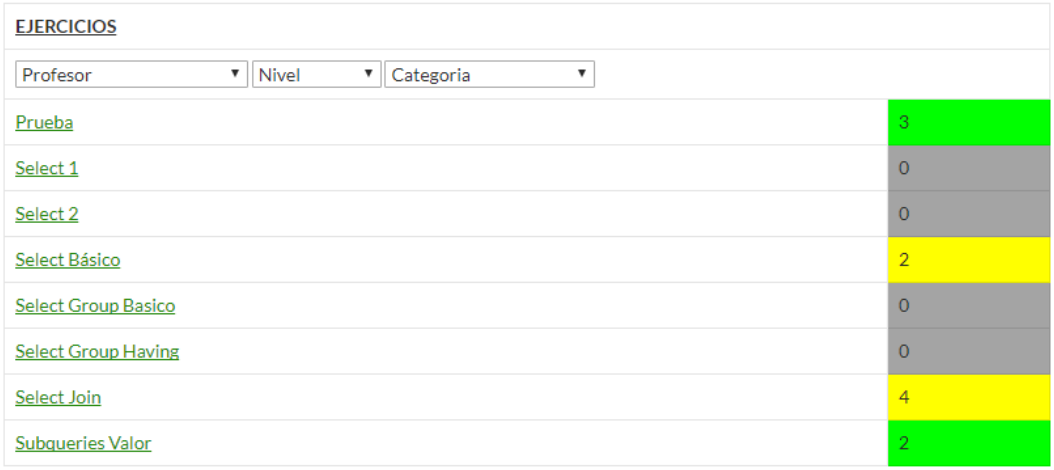

*Solución inválida. La tabla seleccionada no existe.*

Tomando como antecedente las dos soluciones propuestas, se ilustra el funcionamiento antes y después de la optimización.

**Antes:**




*Mensaje indicando que el resultado no es el correcto.*

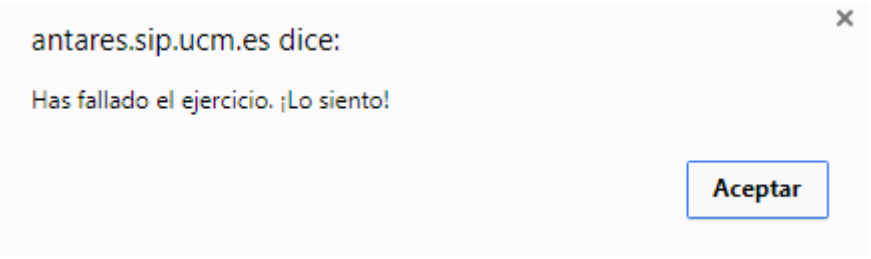



*Página de selección del ejercicio. El número de intentos del ejercicio Select-Básico aumenta una unidad.*

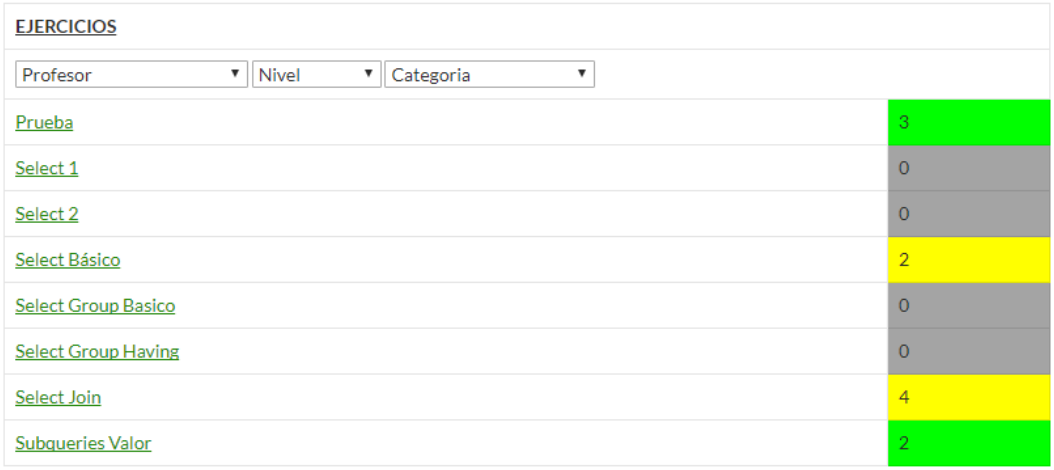
Como se puede comprobar, a efectos del usuario, el resultado tanto para una consulta inválida como de una errónea era el mismo, lo que dificultaba averiguar si el problema venía de una consulta válida incorrecta o de una inválida. Además, se sumaba uno al número de intentos.

**Ahora:**

**a) solución incorrecta**

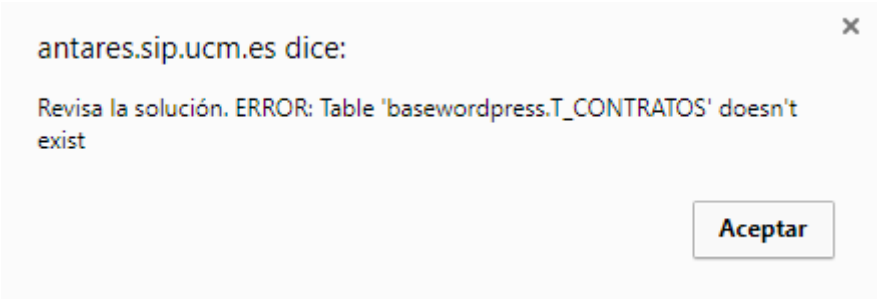



*Mensaje indicando que el resultado no es el correcto. Solución válida pero incorrecta.*




*Página de selección del ejercicio. El número de intentos del ejercicio Select-Básico aumenta una unidad.*

**b) solución inválida**




*Mensaje de error. Solución inválida.*

**Enunciado**

Selecciona la entidad, el identificador de contrato y la descripción de aquellos contratos cuyo titular sea "SERGIO RAMOS".

La solución es " select ID_ENTIDAD, ID_CONTRATO, DESCRIPCION from T_CONTRATO where TITULAR = 'SERGIO RAMOS' " así como todas las variantes de esta.

| TABLA | COLUMNAS | |
|---|---|---|
| T_CONTRATO | ID_ENTIDAD | int(11) |
| | ID_CONTRATO | int(11) |
| | TITULAR | varchar(100) |
| | COTITULAR | varchar(100) |
| | DESCRIPCION | text |
| | FECHA_MODIFICACION | timestamp |

**Solución**

```
select ID_ENTIDAD, ID_CONTRATO
from T_CONTRATOS
where TITULAR = 'SERGIO RAMOS'
```

VALIDAR EJERCICIO

*Página de resolución del ejercicio. Al haber introducido una solución inválida, nos mantenemos en la misma ventana.*

**EJERCICIOS**

Profesor | Nivel | Categoria

| | |
|---|---|
| Prueba | 3 |
| Select 1 | 0 |
| Select 2 | 0 |
| Select Básico | 1 |
| Select Group Basico | 0 |
| Select Group Having | 0 |
| Select Join | 4 |
| Subqueries Valor | 2 |

*Página de selección del ejercicio. El número de intentos del ejercicio Select-Básico no aumenta.*

Como se puede comprobar, tras haber intentado ejecutar una consulta inválida el usuario se mantiene en la misma venta. De manera adicional, si comprobamos la página de selección del ejercicio, veremos que el número de intentos del ejercicio elegido no ha aumentado tras el error.

### 4.3.4 Evitar el uso fraudulento de la aplicación

Se añade una comprobación adicional a la solución propuesta por el alumno, a través de la cual se comprueba que la misma comience por el término *Select* para evitar que el alumno realice sentencias del tipo *Delete, Drop, Update, Truncate,* etc.

Dado que para contrastar las soluciones propuestas estas se han de ejecutar sobre base de datos, se establece esta medida debido a dos aspectos fundamentales.

Por un lado, que el alumno no pueda modificar la base de datos, restringiendo su funcionalidad a la lectura y evitando así prácticas con carácter malicioso en la misma como el borrado o truncamiento de tablas.

Por otro lado, dado que la validación del ejercicio se realiza con una cláusula de conjuntos UNION y esta solo permite cláusulas del tipo SELECT, hemos de asegurarnos de que estas son el único tipo de sentencias que la funcionalidad vaya a validar.

Se ilustra la mejora de la aplicación en este aspecto con ejemplos del tipo DROP TABLE y UPDATE.

**Solución**

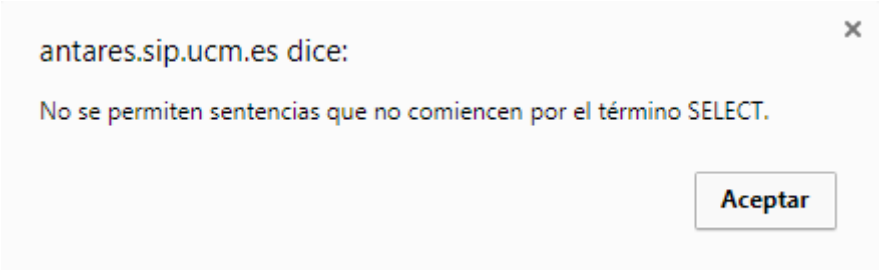
```
UPDATE T_CONTRATO
SET ID_CONTRATO = 5
WHERE ID_CONTRATO = 7
```

*Sentencia del tipo UPDATE*

**Solución**

```
TRUNCATE TABLE T_CONTRATO
```

*Sentencia del tipo TRUNCATE TABLE*




*Resultado*

Como se puede apreciar, se comprueba la sentencia introducida por el usuario antes de intentar ejecutarla para evitar escrituras en la base de datos y en caso de no comenzar por el término *SELECT*, esta se deshecha.

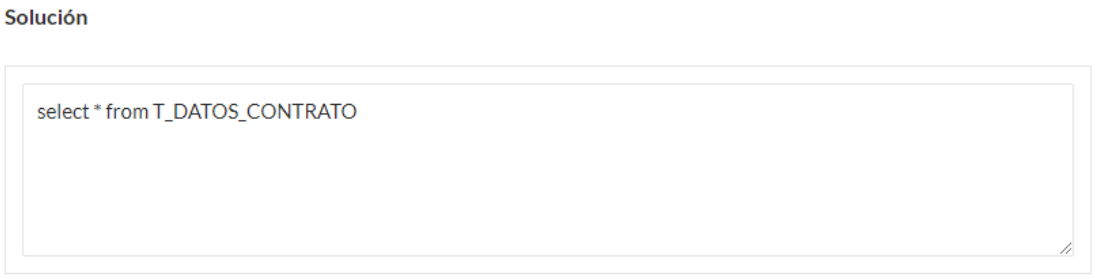
## 4.3.5 Comprobación de consultas en tiempo real

Además de todas las mejoras anteriormente expuesta, se ha añadido una funcionalidad adicional que permite a profesores y alumnos comprobar los resultados de sus consultas. En el caso de los profesores, cuando estos van a dar de alta o editar un ejercicio pueden comprobar los resultados ofrecidos por la solución del mismo en tiempo real, mientras que el alumno deberá validar la solución previamente para que la comparación entre su consulta y la del profesor sea posible, y así evitar veredictos correctos con un número menor de intentos del real.

En el caso del profesor, se han editado las páginas donde se pueden subir o editar ejercicios añadiendo el botón "VISUALIZAR RESULTADO" donde, al presionar, podrán comprobar el resultado de la consulta escrita en pantalla.


*A la izquierda, botón visualizar resultado.*

Este botón muestra los resultados obtenidos con la consulta escrita en el campo Solución, sin necesidad de que el ejercicio haya sido guardado aún en la base de datos:




*Consulta del campo resultado.*


*Resultados de la consulta.*

Como se puede apreciar, se ha incluido además un scrolling para navegar por el resultado en caso de consultas con un gran número de columnas y evitar así el desplazamiento de toda la página en horizontal.

En el caso de los alumnos, podrán comprobar tanto su resultado como el indicado por el profesor en caso de que el veredicto de la solución propuesta sea incorrecto.

Para ello se han creado los botones “RESULTADOS PROFESOR” y “RESULTADOS ALUMNO” que al hacer clic en ellos muestran/ocultan los resultados de las consultas propuestas como solución por cada uno de ellos.


*Botones para ver los resultados del profesor y alumno.*

Al hacer clic en ellos, podremos comprobar la solución real del ejercicio y los resultados de la consulta propuesta por el alumno.

**Resultados Alumno**

| ID_ENTIDAD | ID_CONTRATO | TITULAR | COTITULAR | DESCRIPCION | FECHA_MODIFICACION |
|---|---|---|---|---|---|
| 3058 | 900001 | JUAN BRUGERA MONEDERO | FRANCISCO JAVIER BRUGERA MONEDERO | Contrato firmado por los hermanos Brugera Monedero. | 2017-08-22 22:57:24 |
| 3058 | 900002 | SERGIO RAMOS | PILAR RUBIO | Hipoteca de Sergio Ramos y Pilar Rubio | 2017-08-22 22:57:24 |

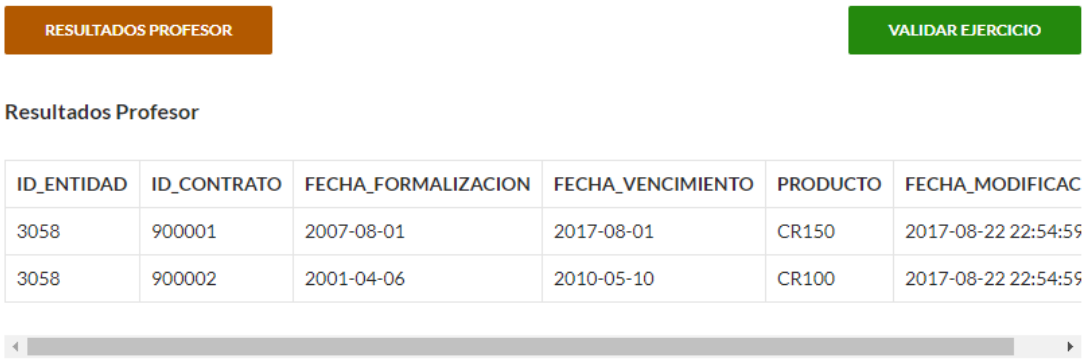
**Resultados Profesor**

| Titular | Descripcion |
|---|---|
| JUAN BRUGERA MONEDERO | Contrato firmado por los hermanos Brugera Monedero. |
| SERGIO RAMOS | Hipoteca de Sergio Ramos y Pilar Rubio |

*Resultados de ambas consultas.*

Como podemos apreciar, en este caso la solución no coincide, por lo que se justifica la presencia de ambos botones.

Por el contrario, en el caso de ejercicios con el veredicto de correcto, únicamente se muestra el botón de la consulta del profesor, ya que coincide con la del alumno:




*Botón mostrar resultados. Veredicto correcto.*

Para mostrar u ocultar los resultados de ambas consultas, basta con hacer clic sobre el botón de los resultados. Las cuatro combinaciones posibles son visibles a través de esta mejora:

1. Sin mostrar resultados


2. Mostrando solo los del profesor

3. Mostrando ambos


**Resultados Alumno**

| ID_ENTIDAD | COTITULAR | IMPORTE_CONCEDIDO | IMPORTE_RESTANTE | IMPORTE_COMISION |
|---|---|---|---|---|
| 3058 | FRANCISCO JAVIER BRUGERA MONEDERO | 5000 | 2340 | 180 |
| 3058 | PILAR RUBIO | 10000000 | 8700000 | 2000000 |

**Resultados Profesor**

| ID_ENTIDAD | TITULAR | IMPORTE_CONCEDIDO | IMPORTE_RESTANTE | IMPORTE_COMISION |
|---|---|---|---|---|
| 3058 | JUAN BRUGERA MONEDERO | 5000 | 2340 | 180 |
| 3058 | SERGIO RAMOS | 10000000 | 8700000 | 2000000 |

4. O solo los del alumno


**Resultados Alumno**

| ID_ENTIDAD | COTITULAR | IMPORTE_CONCEDIDO | IMPORTE_RESTANTE | IMPORTE_COMISION |
|---|---|---|---|---|
| 3058 | FRANCISCO JAVIER BRUGERA MONEDERO | 5000 | 2340 | 180 |
| 3058 | PILAR RUBIO | 10000000 | 8700000 | 2000000 |

# 5. Conclusiones

Tras la finalización de este proyecto puedo decir que me encuentro sorprendido conmigo mismo por varios motivos. Principalmente, la capacidad demostrada de aprender un lenguaje de programación nuevo que nunca había visto, como puede ser PHP, gracias a la ayuda de miles de personas que, indirectamente, exponen dudas similares a las que me han surgido durante el desarrollo de la aplicación. Así mismo, creo que gran parte de esa capacidad es gracias a los conocimientos transmitidos por el profesorado de la Universidad Complutense de Madrid, ya que no solo nos enseñan teoría en las asignaturas, sino que también nos enseñan práctica, a través de la cual conseguimos plantear soluciones para problemas que a priori tomaríamos como inafrontables.

Como consecuencia del párrafo anterior, querría agradecer a todo el profesorado que he tenido durante estos años la formación impartida, sin la cual, la realización de este proyecto no hubiera sido posible. Una especial mención merecen Mercedes Merayo y Manuel Núñez que me han prestado toda la ayuda que he necesitado, sin exigir nada a cambio, en todo momento.

Aprovecho mis circunstancias personales para instar a todo alumno de la Universidad Complutense de Madrid a realizar su proyecto tan pronto como le sea posible. Es un trabajo que conlleva un esfuerzo notable y que es difícil compaginar con una situación laboral estable.

Es mi obligación agradecer a toda persona relacionada con la comunidad informática su participación activa en foros de programadores, sin la cual muchas de las tareas en esta memoria expuestas no habrían sido posibles.

Por otra parte, de cara a hacer esta aplicación aún mejor, se invita a cualquier alumno de la Universidad Complutense de Madrid a que tenga en consideración las siguientes ampliaciones que se podrían llevar a cabo para hacer de SQLab una herramienta más útil y potente así como cualquier otra que considere:

- Habilitar una nueva página donde profesores y alumnos puedan realizar sentencias libres. De esta manera, los alumnos podrán practicar en modo libre y obtener mejores puntuaciones cuando se pongan a prueba, y los profesores podrán diseñar las consultas de sus ejercicios. Estos también podrán crear o modificar tablas, vistas o insertar datos en las mismas para aumentar la complejidad de los ejercicios que propongan.
- Añadir nuevas categorías o niveles al repertorio actual para poder dar de alta nuevos tipos de ejercicio.
- Ampliar el alcance de la herramienta para poder proponer ejercicios no solo del tipo SELECT si no de cualquier tipo de sentencia SQL en general.

Por último, querría agradecer a todas aquellas personas que han estado apoyándome durante la realización de este desarrollo y sobre todo a mis familiares más cercanos: A mi madre Marta, a mi padre Fernando, a mi hermano Francisco y en especial a mi pareja Sara que ha estado apoyándome incondicionalmente día a día y ayudándome en la medida de lo posible sin tener ningún conocimiento de la materia.

# 6. Bibliografía

**Metodologías Ágiles:**

Manifiesto por el Desarrollo Ágil de Software: http://agilemanifesto.org/iso/es/manifesto.html

Desarrollo ágil de software: https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software

Metodologías ágiles de gestión de proyectos: https://www.marblestation.com/?p=661

**Recursos utilizados:**

Los logotipos de los recursos se han obtenido a través del buscador de imágenes del buscador www.google.es

Las descripciones de los recursos utilizados se han obtenido de la enciclopedia electrónica www.wikipedia.org

**Páginas de ayuda a programadores:**

Manual PHP: http://php.net/manual/en/index.php

StackOvwerflow**:** https://stackoverflow.com/

Manual Wordpress: https://codex.wordpress.org/Class_Reference/wpdb

**Otras referencias:**

Lista de palabras reservadas MySQL: https://techtastico.com/post/lista-de-palabras-reservadas-en-mysql/

**Estructuración de la memoria:** Basada en la estructura de la memoria anterior presentada por el compañero José Alfonso Contreras.

El resto de documentación expuesta en esta documentación que no se haya reflejado en este apartado es debido a los conocimientos propios del autor.