Facultad de Informática

Aula Virtual de SQL/PSM

UNIVERSIDAD COMPLUTENSE DE MADRID
CURSO ACADÉMICO 2018/2019

Trabajo de fin de grado
Grado en Ingeniería Informática
Grado en Ingeniería del Software

Departamento de Sistemas Informáticos y Computación

Autores:

ALBERTO MÁRQUEZ GÓMEZ

MARTA RODRÍGUEZ COUÑAGO

Director:

JESÚS CORREAS FERNÁNDEZ

Agradecimientos

A todos aquellos que no me dejaron caer al abismo de la desesperación, siempre creyeron en mí y sin los que no habría conseguido llegar hasta el final.

También me gustaría dar las gracias al Dr. Jesús Correas Fernández por toda su ayuda y dedicación y a Marta, mi compañera, la cual a paso de ser una completa desconocida a una de las personas con las que más sufrimiento he compartido los últimos meses.

Gracias.

Alberto Márquez Gómez.

A mi familia, que estuvieron siempre animándome y apoyándome tanto en los suspensos como en los aprobados, en concreto a mis padres que me enseñaron que hay que esforzarse al máximo y nunca rendirse a pesar de las circunstancias.

También a mis amigos que supieron sacarme una sonrisa en los días más estresantes y a mi compañero Alberto, que además de encontrar un buen compañero de trabajo he encontrado un gran amigo.

Además, al Dr. Jesús Correas Fernández por su dedicación y disposición en todo momento a ayudarnos y aconsejarnos a lo largo de todo el proyecto, haciéndolo posible.

Gracias.

Marta Rodríguez Couñago.

Resumen

Este proyecto intenta solucionar el problema que tienen los alumnos actualmente al estudiar cómo hacer procedimientos en PL/SQL.

Para probar un procedimiento y saber si este está bien o no, actualmente no conocemos ningún sistema en el que puedan escoger entre un conjunto de problemas y lanzar sus procedimientos contra uno de estos para ver si su código lo resuelve o no.

Los alumnos en muchos casos no estudian bien la forma de hacer los procedimientos, ya que antes de hacer uno tienes que crear las tablas y rellenarlas con casos clave para ver si su procedimiento cumple lo que el problema exige, lo que muchas veces lleva más tiempo que el desarrollo del propio procedimiento.

Con el sistema que hemos desarrollado, los alumnos podrán probar sus procedimientos sin tener que crear las tablas previamente. Solo tendrán que subir el procedimiento que se les pide hacer en la página y de forma automática esta les informará de los posibles errores que pueda tener, avisos acerca del código o si es correcto.

Estos problemas serán corregidos con una nota por el profesor y en caso de que el alumno lo necesite, se podrán añadir comentarios a su procedimiento para ayudar al alumno a llegar la solución correcta del ejercicio. Hasta llegar al número máximo de intentos permitidos en ese ejercicio.

Palabras clave

DAO, PL/SQL, Node.js, Oracledb, MySQL, JavaScript

Abstract

This project tries to solve the problem that students currently have when studying how to do procedures in PL / SQL.

To test a procedure and know if it is okay or not, we currently do not know any system in which can choose between a set of problems and launch their procedures against one of these to see if their code resolves or not.

Students in many cases do not study well how to do the procedures, because before doing one you have to create the tables and fill them to see if your procedure meets what the problem demands, which often takes more time than the development of the procedure itself.

With the system we have developed, students will be able to test their procedures without having to create the tables previously. They will only have to upload the procedure that they are asked to do on the page and they will automatically inform them of possible errors they may have, notices about the code or if it is correct.

These problems will be corrected with a note by the teacher and in case the student needs it, comments can be added to their procedure to help the student arrive at the correct solution of the exercise. Until you reach the maximum number of attempts allowed in that exercise.

Key words

DAO, PL/SQL, Node js, Oracledb, MySQL, JavaScript

Contenido

1.Introducción

1.1 Motivación

Este proyecto facilita la implicación del alumno a la hora de aprender a desarrollar código en PL/SQL, resolviendo el problema de comprobar si un procedimiento es correcto o no. Se facilita la tarea de tener que instalar un cliente para ejecutar dichos procedimientos como es sql developer y la tediosa tarea de crear y rellenar las tablas de la base de datos.

El profesor podrá poner comentarios a una entrega hecha por el alumno para poder ayudarle y guiarle en los casos en los que no encuentre una solución válida. Además, podrá calificarle y ponerle nota hasta tantas veces como el alumno necesite.

Esta herramienta da apoyo tanto a profesores como a alumnos a la hora de resolver problemas de PL/SQL. Con una sencilla e intuitiva página web que se conecta con MySQL guarda toda la información relacionada con las asignaturas, alumnos, grupos y ejercicios.

Parte de funcionalidad tiene similitud con muchos jueces en línea que se pueden encontrar en internet además de los que se usan en otras asignaturas, como el DomJudge o el juez de la Universidad de Valladolid [19]. Sin embargo, la dificultad de esta herramienta se encuentra en que no compara resultados uno a uno como hacen otros jueces. Esta crea un entorno de prueba en el servidor Oracle con tablas y datos propios de cada ejercicio, almacena el procedimiento del alumno y ejecuta unos scripts de prueba.

1.2 Objetivos

El objetivo general del proyecto consiste en mostrar las ventajas y el potencial que puede tener el corregir automáticamente un procedimiento y que además este pueda tener una realimentación por parte del profesor mediante una corrección del código del alumno.

Otro de los objetivos buscados es que los alumnos puedan probar sus procedimientos desde una aplicación web sin necesidad de instalar otras aplicaciones además poder subir la solución de sus ejercicios y poder generar una corrección automática.

Establece un sistema que le permite al profesor subir ejercicios para que los alumnos puedan resolverlos y obtener una realimentación de sus soluciones.

1.3 Estructura de la memoria

La memoria esta organizada en quince capítulos más un apéndice. A continuación, se presentan brevemente los capítulos mencionados.

- Introducción

  Se describe la motivación por la que decidimos hacer este TFG, los objetivos y su estructura.
- Introduction

  Es la introducción traducida al inglés.
- Estado del arte

  Describiremos las similitudes y diferencias con otros jueces en línea.
- Especificación de requisitos

  Describiremos los requisitos del profesor para realizar el proyecto, saber qué componentes del sistema realiza cada tarea y los servicios que proporciona el sistema.
- Herramientas, metodología y planificación

  En este apartado detallaremos las herramientas que necesita la aplicación para funcionar, tanto en la parte del servidor como para la parte del cliente, además se describe la metodología utilizada para desarrollar el proyecto y la planificación de cada uno de los avances que hemos hecho en la aplicación.
- Proceso de desarrollo

  Describimos el desarrollo del proyecto desde los primeros prototipos y diseños hasta el final de la implementación.
- Arquitectura de la aplicación

  La arquitectura de la aplicación en la cual se describe la parte del cliente y la parte del servidor.

  Este capítulo contiene el diseño de la base de datos MySQL, en la que alojaremos toda información referente a usuarios, ejercicios y el código fuente y respuestas de los alumnos, se incluye el diagrama entidad-relación y el modelo relacional.
- Diseño de la base de datos

  En este trabajo se han utilizado dos bases de datos diferentes, una en MySQL para almacenar los datos de los usuarios tanto profesores como alumnos y otra en Oracle para ejecutar los procedimientos de los usuarios.
- Casos de uso

  Describimos los posibles casos de uso de la aplicación.

- Diagramas de flujo

  Describimos los flujos de las acciones más importantes que los usuarios hacen en la aplicación.

- Implementación

  En este capítulo también se incluye la descripción completa de las interfaces de usuario de la aplicación.

  Cómo se ha desarrollado el cliente y el servidor gestionando las conexiones con las bases de datos y tratando los datos proporcionados por ellas.

- Reparto del trabajo

  Se describe la contribución que ha hecho cada miembro al proyecto y en qué puntos uno ha desarrollado más que el otro, aunque todos los desarrollos han sido comunes.

- Aportaciones individuales

  Se describen las aportaciones únicas de cada miembro del grupo y la descripción de estas.

- Conclusiones y trabajo futuro

  Se explica la forma en la que se ha desarrollado el proyecto y posibles implementaciones futuras para complementarlo y ampliarlo.

- Conclussions and Future work

  Son las conclusiones y trabajo futuro traducido al inglés.

- Repositorio

  En este apartado tratamos como hemos ido guardando los avances hechos en el proyecto desde el comienzo en Google Drive hasta los últimos desarrollos en GitHub.

- Bibliografía

  Se listan las fuentes de información de todas las tecnologías y sistemas utilizadas en este trabajo.

- Apéndice

  Se presenta un apéndice en el que se detalla de qué se compone un ejercicio subido por el profesor.

2. Introduction

2.1 Motivation

This project facilitates the involvement of the student when learning to develop code in PL/SQL, solving the problem of checking whether a procedure is correct or not. It facilitates the task of having to install a client to execute such procedures such as sql developer and the tedious task of creating and filling in the tables in the database.

The teacher can comment on a delivery made by the student to help and guide you in cases where you can not find a valid solution. In addition, you can grade and write it down as many times as the student needs.

This tool supports both teachers and students when solving PL / SQL problems. With a simple and intuitive web page that connects with MySQL, it keeps all the information related to the subjects, students, groups and exercises.

Part of functionality has similarity with many online judges that can be found on the internet in addition to those used in other subjects, such as DomJudge or the judge of the University of Valladolid [19]. However, the difficulty of this tool is that it does not compare results one by one as other judges do. This creates a test environment on the Oracle server with tables and data from each exercise, stores the student's procedure and runs some test scripts.

2.2 Objectives

The general objective of the project is to show the advantages and potential that can automatically correct a procedure and that this can also have a feedback from the teacher by a correction of the student code.

Another of the objectives sought is that students can test their procedures from a web application without installing other applications in addition to uploading the solution of their exercises and generate an automatic correction.

It establishes a system that allows the teacher to upload exercises so that the students can solve them and obtain a feedback of their solutions.

2.3 Estructure

The memory is organized into fifteen chapters plus an appendix. The chapters mentioned are briefly presented below.

- Introduction

  It describes why we decided to do this TFG, the objectives and its structure.

- Introduction

  It is the introduction translated into English.

- Specification of requirements

  We will describe the requirements of the teacher to carry out the project, the requirements of the system to know which components of the system perform each task and the services that It will provide at the end

- Tools, methodology and planning

  In this section we can find what tools the application needs to work, both on the server side and on the client side.

  The methodology used to develop the project and how we have been planning each of the advances we have made in the application

- Development process

  We describe the development of the project from the first prototypes and designs until the end of the implementation.

- Design of the database

  We use the MySQL database to save all the information, We just use Oracle database to correct the procedures.

- Use cases

  We describe the possible processes that we can perform in the application.

- Flow charts

  We describe the flows of the most important actions that users do in the application.

- Implementation

  This chapter also includes the full description of the user interfaces of the application.

  How the client and the server have been developed, managing the connections with the databases and treating the data provided by them.

- Work distribution

  It describes the contribution that each member has made to the project and in which points one has developed more than the other, although all the developments have been common.

- Individual contributions

The unique contributions of each member of the group and their description are described.

- Conclusions and future work

  It explains the way in which the project has been developed and possible future implementations to complement and expand it.

- Conclusions and Future work

  They are the conclusions and future work translated into English.

- Repository

  In this section we discuss how we have kept the progress made in the project from the beginning in Google Drive to the latest developments in GitHub.

- Bibliography

  It lists where we have taken the information of all technologies used

- Appendices

  An appendice is presented in which is detailed what an exercise is made up by the teacher.

3. Estado del arte

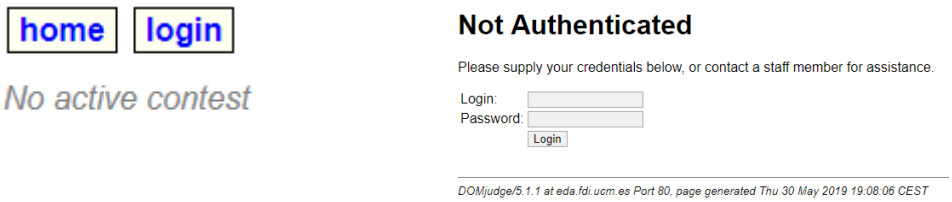
Los sistemas más parecidos al que hemos desarrollado en este trabajo son los jueces online utilizados en programación competitiva. Los distintos jueces que podemos encontrar tienen una única funcionalidad, corregir ejercicios y dar un veredicto, sin embargo, queríamos ir más allá y decidimos implementar que además de corregir ejercicios pudiera crear un entorno para el profesor sencillo e intuitivo, llevar un seguimiento del estado de sus alumnos con sus ejercicios además de poder comunicarse con ellos y transmitirles en caso de que sea necesario una realimentación.

Para tener un conocimiento más extenso de esta área investigamos algunas aplicaciones similares disponibles en las distintas universidades además de la nuestra propia. A continuación, se presentan distintos jueces en línea de los cuales tomamos el concepto de corregir un ejercicio online, sin la necesidad de una persona que confirme si los ejercicios son correctos o no y vea en qué puntos fallan.

Estos jueces tienen el fin de intentar que veas el fallo por ti mismo como si de una competición se tratara sin dar ninguna ayuda adicional.

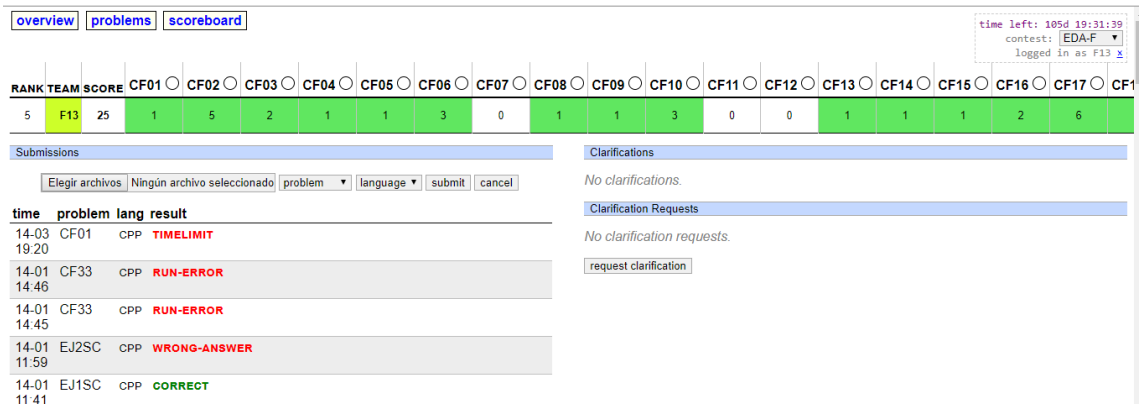
DomJudge

Juez utilizado para la corrección de ejercicios y exámenes en la asignatura de EDA (Estructura de datos y algoritmos) [21].
Este juez tiene una interfaz sencilla y simple enfocándose a la idea de dar un veredicto sobre la corrección de las soluciones a los ejercicios propuestos.
Nada más entrar en el sistema aparecen dos botones, “login” y “home”, para acceder a los ejercicios hay que pinchar en el “login” y después introducir un usuario y contraseña como se muestra a continuación:



Una vez dentro se pueden ver tres menús arriba a la izquierda, “overview”, “problems” y “scoreboard”. El primero de ellos permite visualizar tus propios ejercicios subidos, “problems” muestra una lista de todos los ejercicios disponibles que tiene el alumno y “overview” despliega una tabla con todos los ejercicios que han intentado el resto de los usuarios. En esta aplicación se usa un código de colores para la resolución de ejercicios:

el verde claro se utiliza cuando la solución del ejercicio es correcta, rojo para los erróneos y verde oscuro si el usuario ha sido el primero en resolver el problema.
Aquí se muestra un ejemplo de la vista que tiene un alumno en la aplicación:



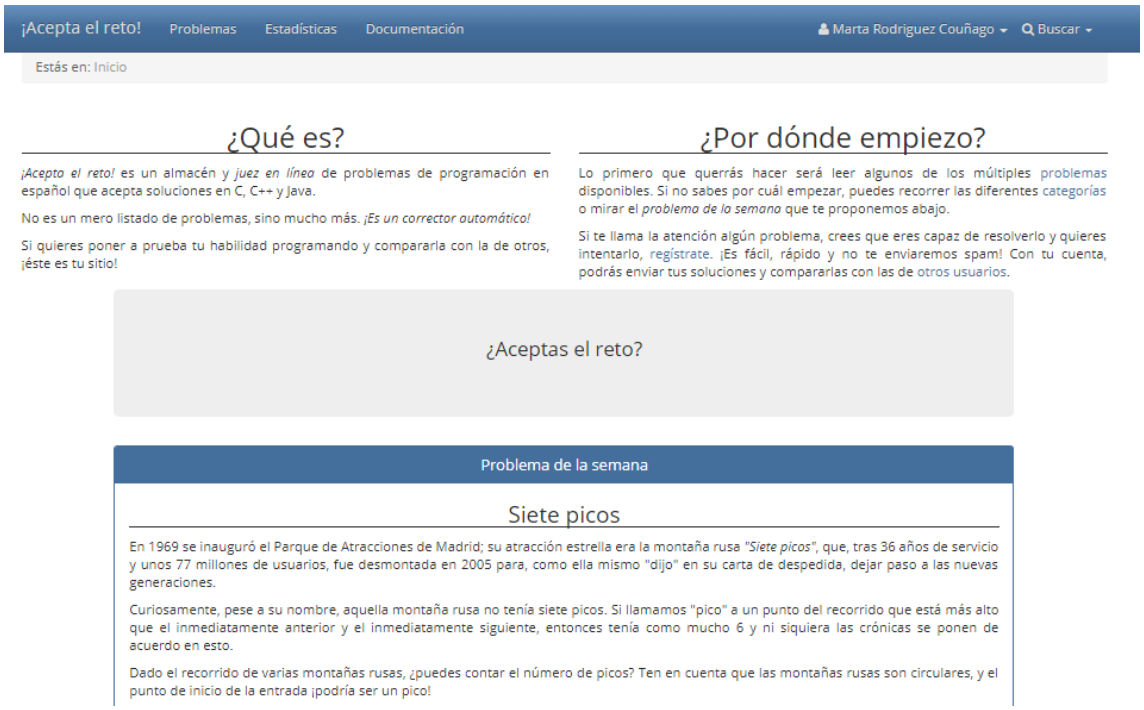
Para subir un problema a la aplicación, que debe ser en lenguaje C++, hay que pulsar el botón “Elegir archivo”, escoger el problema a resolver y finalmente pulsar “Submit”.
Los veredictos más comunes de errores son:

- Timelimit: el problema ha tardado más de lo esperado en resolverse o tiene algún bucle que no termina.
- Run-error: ha habido un error durante la ejecución del programa.
- Wrong-answer: la solución no ha pasado todos los casos de prueba por lo que no es correcta.
- Compiler-error: ha habido un error en la compilación del problema.

Además, existe un manual de usuario [22] en pdf accesible desde la web con información más extensa sobre otros veredictos que puede ofrecer el juez y su uso.

Acepta el Reto

Acepta el reto [23], desarrollado en la Facultad de Informática de la Universidad Complutense de Madrid, presenta una interfaz algo más amigable con además más información describiendo para qué sirve, qué es y a quién va dirigido. Además, acepta soluciones no solo en C++ sino también en C y Java.
Una vez se ha iniciado sesión en el sistema aparece la pantalla principal para escoger el problema que se desee dentro de cada volumen. A continuación se muestra una figura de la pantalla principal:



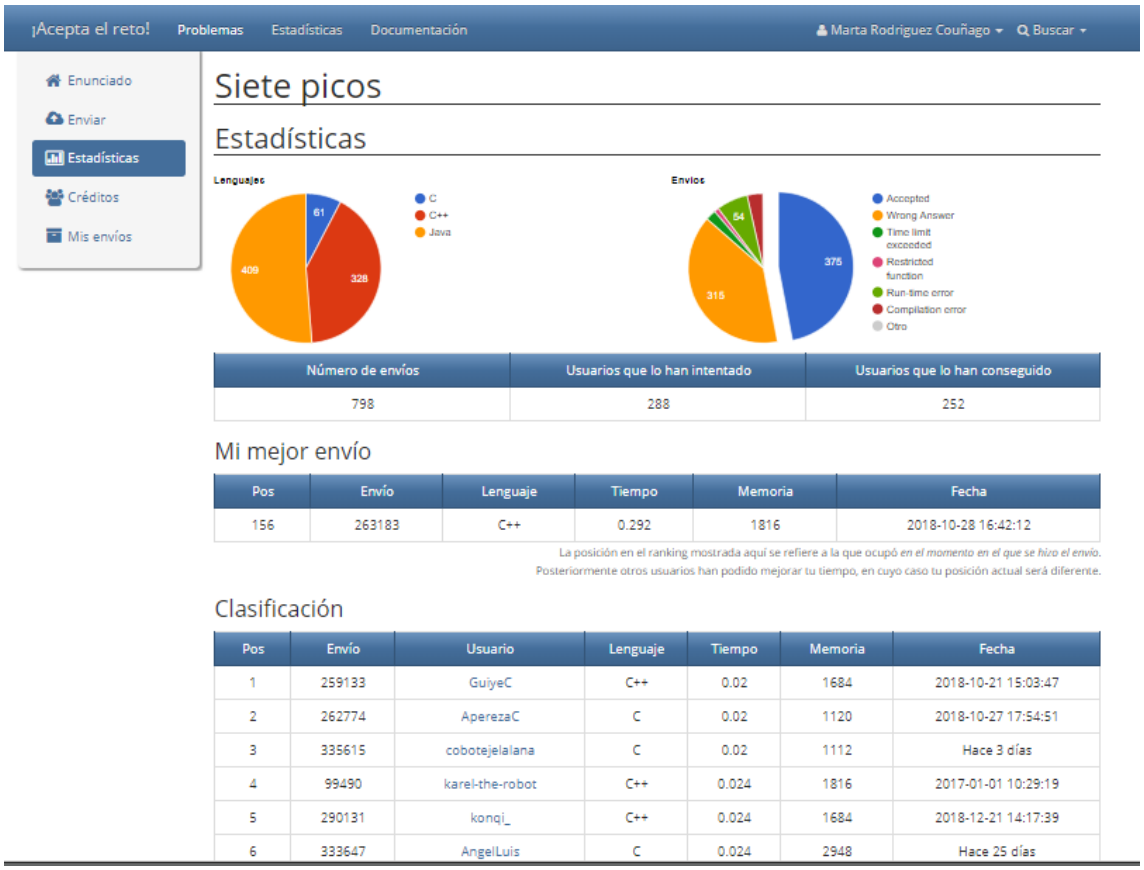
En la pantalla principal da bastante más información de la aplicación y para qué sirve, te sugiere un problema de la semana que puede ser aleatorio entre todos los que hay.

Si no lo hemos escogido, en la ventaba de problemas se recopilan varios de ellos recogidos en volúmenes compuestos por cien problemas en cada volumen.

En la siguiente figura se puede ver el aspecto de la aplicación una vez que se ha seleccionado la pestaña de “Problemas”.

Además de resolver el problema, este juez en vez de recopilar los datos en una tabla los reagrupa en estadísticas por problema mostrando los usuarios que lo han intentado, los que lo han conseguido y además muestra un ranking con puntuación con los mejores

envíos teniendo en cuenta el tiempo que ha tardado en ejecutarse y el espacio que ha usado en memoria tal y como se ve en la siguiente figura:

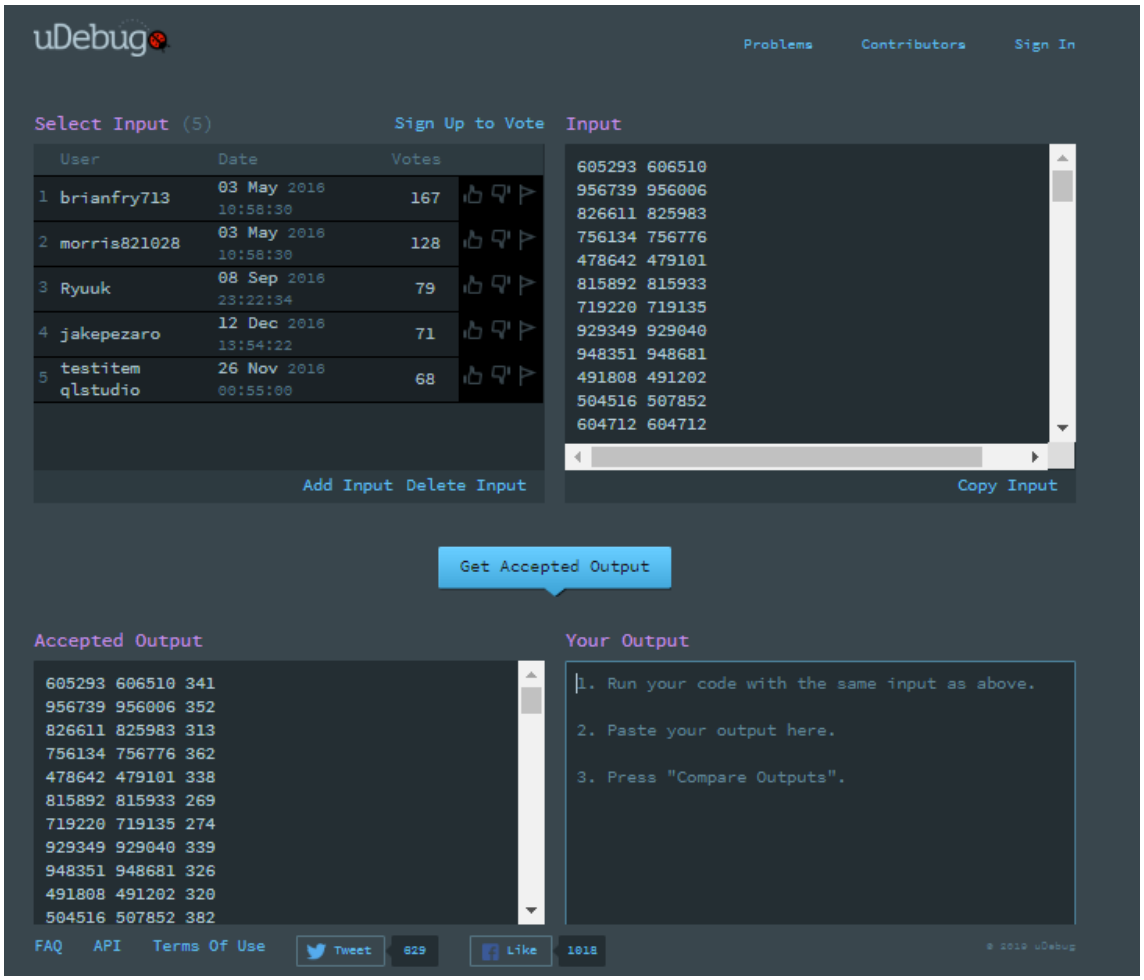


UVA Online Judge

Es un juez automático desarrollado por la Universidad de Valladolid [20]. Provee de una herramienta complementaria llamada uHunt que mantiene estadísticas, proporciona selecciones de problemas a resolver y expone una API web para que otros desarrolladores web puedan desarrollarla. Además incluye un chat donde todos los usuarios registrados pueden dejar su comentario o dudas. Un ejemplo de esta herramienta se muestra a continuación:

Además, permite evaluar los problemas sin tener que subirlos al juez, esta herramienta se llama uDebug donde puedes probar casos de prueba de otras personas o los tuyos propios, ver la salida que devuelve y además si lo necesitas puedes subir tu código y ejecutar esos casos de prueba seleccionados.

La siguiente figura muestra el aspecto general de esta herramienta:

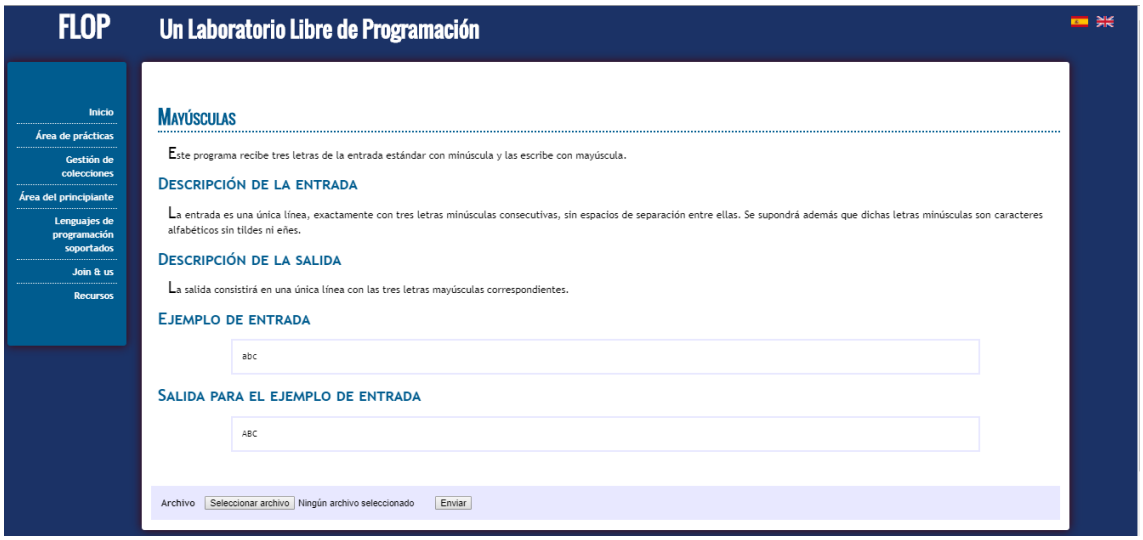


FLOP

Laboratorio virtual para practicar la programación [24] desarrollado en las universidades Rey Juan Carlos y Complutense. Contiene una colección de problemas de programación y un corrector automático de programas (juez) para dichos problemas. Acepta C++, Java, Python, Pascal o Haskell.

Flop está programado en C con licencia GPL por lo que a quien quiera que le interese se lo puede descargar y hacerle cualquier mejora.

La pantalla para subir un problema es bastante sencilla. se muestra un ejemplo en la siguiente imagen de un problema para escribir en minúsculas cualquier palabra:



Jutge.org

Es un programa educativo en línea gratuito [19] en el que los estudiantes pueden tratar de aprender sobre 1500 problemas calificados utilizando 20 lenguajes de programación diferentes. El juez calcula el veredicto de sus soluciones utilizando conjuntos de pruebas exhaustivos ejecutados con restricciones de tiempo, memoria y seguridad.
Ha sido desarrollado por la Universidad Politécnica de Cataluña, su repositorio está organizado por temas y grado de dificultad.

4. Especificación de requisitos

El Aula Virtual de SQL/PSM debe cumplir unos requisitos mínimos de funcionalidad y de accesibilidad. Para ello hemos hecho en las fases iniciales de desarrollo una especificación de requisitos, la cual permite dejar claro cómo se tendrá que hacer el desarrollo de la aplicación.

Requisitos funcionales del sistema

Estos requisitos describen lo que el sistema debe hacer. Estos requerimientos dependen del tipo de software que se desarrolle y de los posibles usuarios del software. Estos son:

- Debe almacenar información de usuarios de al menos dos tipos: profesores y alumnos con distintos accesos a las funciones del sistema.

- Debe incluir un interfaz web de usuario con seguridad básica de acceso (id de usuario y contraseña).

- El sistema debe contener información de los ejercicios que ha intentado resolver el alumno, si los ha conseguido resolver, el número de intentos hasta el momento, etc.

- El sistema debe incluir un mecanismo para que el profesor pueda poner comentarios sobre una entrega del alumno.

- El profesor debe permitir definir problemas en el sistema, inspeccionar la solución propuesta por un alumno e introducir una calificación y comentarios sobre esa solución.

- En la lista de ejercicios se debe poder editar un ejercicio existente, eliminar un ejercicio si no está añadido a ningún grupo, o crear uno nuevo (mediante un botón).

- El sistema debe permitir al alumno introducir una solución a un problema en forma de script SQL/PSM que contenga el procedimiento almacenado.

- La corrección debe ejecutar el procedimiento modelo y el procedimiento del alumno en entornos separados para que no interfieran entre sí.

- El alumno podrá visualizar ejercicios de otros años para poder practicar, estos ejercicios se podrán ejecutar en el juez. Sin embargo, no se registrarán las soluciones del alumno en la base de datos.

- Se deben poder crear asignaturas y asociarlas a un profesor.

- La aplicación debe poder crear cursos y grupos y asignarlos a asignaturas existentes.

- En la opción Asignaturas, al eliminar una asignatura con grupos asociados se debe mostrar un mensaje de error tipo: "La asignatura tiene grupos asociados. Elimine primero los grupos asociados a esta asignatura."

- Por cada intento de resolución, debe almacenar el código fuente propuesto por el alumno, el resultado de la prueba (éxito, fracaso, y posiblemente algún otro nivel intermedio), fecha y hora de la corrección, y los mensajes producidos por la corrección automática.

- También se debe poder almacenar en cada intento de resolución una calificación asignada manualmente por el profesor y comentarios adicionales que pueda introducir el profesor sobre el código fuente del alumno (como texto).

- La definición de un problema debe estar formada al menos por los siguientes elementos (además de un código identificador, denominación, número máximo de intentos para resolverlo, etc.):

    1) Script SQL de creación de tablas e inserción de datos previos para las pruebas.

    2) Enunciado del problema. Puede ser un texto multilínea o bien un documento pdf.

    3) Script SQL con el código SQL/PSM con la solución correcta del problema.

    4) N scripts SQL con el código SQL/PSM con las pruebas a realizar. Cada prueba estará en un script diferente y tendrá dos partes: primero un bloque anónimo con la llamada al procedimiento y a continuación las comprobaciones a realizar.

- La corrección de un ejercicio debe realizar lo siguiente:

    1. Crear el entorno vacío para el profesor (sin tablas ni ningún otro objeto). Por ejemplo, en Oracle debe crear un usuario nuevo que acceda a los objetos de su schema (con los permisos adecuados). Ejecutar los scripts de creación de tablas (1) y de creación del procedimiento almacenado correcto (3).

    2. Crear otro entorno vacío como en el paso 1 para ejecutar el procedimiento propuesto por el alumno. Ejecutar el script de creación de tablas (1) y el script que ha proporcionado el alumno.

    3. Por cada uno de los scripts de (4), se debe hacer lo siguiente:

        - Ejecutar el bloque anónimo de (4) con la llamada al procedimiento almacenado correcto en el schema del profesor.
        - Ejecutar el script completo de (4) con la llamada al procedimiento almacenado del alumno en el schema del alumno. Se debe ejecutar tanto el bloque anónimo como las comprobaciones de (4).
        - Comparar la salida por consola de cada una de las dos ejecuciones y, si son diferentes, mostrar al usuario la diferencia entre ambas.

    4. Una vez ejecutadas las comprobaciones del apartado cuatro en la definición de un problema, se deben recoger los errores almacenados en la tabla “resultados_correccion" para mostrarlos al usuario junto al bloque anónimo de cada prueba.

- Al final, el sistema debe proporcionar al alumno los resultados de 3 y 4 (el contenido de la tabla).

- El sistema debe utilizar un SGBD para almacenar su base de datos interna (usuarios, problemas, soluciones, puntuaciones, mensajes, etc.). Este SGBD puede ser el SGBD que se utilice para ejecutar las soluciones de los problemas propuestos, o bien otro, por ejemplo, MySQL si se utiliza LAMP.

Aspecto general de la aplicación.

Al tener dos tipos de usuarios bien definidos a continuación se pasará a explicar algunas restricciones propuestas por cada tipo de usuario.

1. Usuario profesor.

1.1 Menú Ejercicios.

- El orden de las operaciones para los ejercicios debe ser la siguiente: listar ejercicios, crear ejercicio, añadir un ejercicio a un grupo, quitar un ejercicio de un grupo y eliminar un ejercicio.

- En la lista de ejercicios se debe poder seleccionar una asignatura y grupo para no mostrar todos los ejercicios, incluyendo una opción (o un botón) para poder listar los ejercicios no añadidos a ningún grupo.

1.2 Menú Alumnos.

- El orden de las operaciones para los alumnos debe ser: cargar lista de alumnos, evaluar alumnos y mostrar calificaciones.

- Mostrar calificaciones debe filtrar por asignatura y grupo y debe mostrar los datos de cada alumno. Cuando se pulse sobre un alumno debe mostrar una ventana con las calificaciones de todos los ejercicios y exámenes indicando el tipo de cada uno. En el mismo formato que en el menú del alumno.

- Mostrar calificaciones debe permitir filtrar por asignatura y grupo.

- Cargar lista de alumnos debe permitir seleccionar asignatura y grupo. Debe añadirse un texto que explique el formato que debe tener el fichero.

- En la opción de evaluación de alumnos se debe poder seleccionar la asignatura grupo y ejercicio para mostrar la lista.

- Cuando se evalúa un ejercicio de un alumno, se puede modificar el resultado de las pruebas.

- Cuando se vuelve a evaluar un ejercicio de un alumno que ya se ha evaluado, debería aparecer en algún sitio el código corregido por el profesor, si no fuese así, el profesor no tiene acceso a sus propias anotaciones de la corrección anterior.

- En la pantalla de evaluación de un ejercicio, la nota debe ser un número entero de 0 a 100.

1.3 Menú Asignaturas y Grupos

- Debe tener dos opciones: "Asignaturas" y "Grupos".

1.4 Menú Profesores

- Debe haber una opción para que un profesor pueda añadir a otro profesor. Que tenga la opción "Añadir Profesor" (pero no se puede eliminar un profesor si tiene ejercicios asociados).

2. Usuario alumno.

- Si el alumno está en más de una asignatura/grupo, debe tener la posibilidad de seccionar el grupo que quiera.

- La asignatura/grupo seleccionado debe aparecer en la parte superior de la pantalla.

- Debe tener cuatro menús: listado de ejercicios, ejercicios, exámenes y calificaciones.

- El menú ejercicios debe tener dos opciones: "Ejercicios del curso actual" y "Ejercicios de cursos anteriores".

- Cuando seleccione cualquiera de los menús anteriores, el alumno debe poder seleccionar asignatura y grupo (o todos).

5. Herramientas, metodología y planificación

Herramientas utilizadas para el desarrollo del proyecto

A continuación, se presentan las herramientas principales en el desarrollo y planificación que hemos usado para el desarrollo de este proyecto.

Visual Studio Code

Editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS, incluye soporte para la depuración y control integrado de Git. Se ha utilizado para implementar la aplicación en Node js.
Decidimos utilizar esta herramienta para el desarrollo del código porque uno de nosotros ya la conocía.

XAMPP

XAMPP es un servidor independiente de la plataforma de código libre que permite instalar de forma sencilla el servidor web Apache. También incluye un servidor de bases de datos MySQL con su respectivo gestor, phpMyAdmin.

Al ser un proyecto pequeño, nos decantamos por usar dicho servidor además de que ya lo habíamos usado con anterioridad para otras asignaturas.

Google Drive

Al principio del proyecto utilizamos Google Drive como repositorio para almacenar nuestras versiones pensando que sería más sencillo almacenarlas y bajarlas de ahí, pero al poco tiempo nos dimos cuenta de que las versiones cada vez iban creciendo más y se hacía inviable integrar los cambios, por lo que cambiamos a un método de control de versiones algo más sofisticado como Git.

GitHub

GitHub es un repositorio que permite almacenar las versiones de manera ordenada y si ha habido algún problema, poder revertirlo fácilmente. Además, también permite hacer

varias ramas, poder trabajar en paralelo en varias funcionalidades y tener una copia local del código generado.

Nuestra experiencia utilizando esta herramienta ha sido bastante satisfactoria ya que durante en la carrera se nos ha explicado cómo se usa y algunos comandos, pero no en un proyecto real con varias ramas. Gracias a este proyecto hemos tenido la oportunidad de experimentar y poder utilizar esta herramienta.

El repositorio donde hemos hecho el trabajo se encuentra en [26].

Oracle

Oracle es una herramienta cliente/servidor para la gestión de Bases de Datos. Se basa en la tecnología cliente/servidor, para su utilización primero es necesario la instalación de la herramienta servidor y posteriormente podríamos atacar a la base de datos desde otros equipos con herramientas de desarrollo.

Es posible atacar a la base de datos a través del SQL plus incorporado en el paquete de programas Oracle para poder realizar consultas, utilizando el lenguaje SQL.

En nuestra aplicación debíamos implementar Oracle como un servicio RESTful a través de Node js donde un servicio RESTful es una interfaz capaz de interactuar con cualquier dispositivo capaz de comunicarse vía HTTP con la finalidad de obtener o generar operaciones en un repositorio de datos principalmente

Node Js

Node.js es un lenguaje de programación basado en JavaScript el cual se ejecuta en el lado del Servidor y la programación está orientada a eventos permitiendo generar aplicaciones más livianas y eficientes. Este lenguaje se encuentra construido en el motor de JavaScript de Chrome.

Oracledb

La librería oracledb es el controlador de conexión a la base de datos Oracle para Node.js.

Los módulos específicos encargados de conectar con las bases de datos y ejecutar sentencias se llamarán DAO.

Jquery

JQuery es una biblioteca multiplataforma de JavaScript de software libre y de código abierto posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2 permitiendo su uso en proyectos libres y privados. Permite interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

Bootrstrap

Es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.

Metodología y Planificación

La metodología con la que hemos trabajado es ágil porque es la más conveniente para adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.

Al decidir utilizar una metodología ágil decidimos planificarnos acorde a esto, por lo que hicimos reuniones cortas diarias para tener más control de lo que hacía cada uno, qué es lo que faltaba y qué es lo que haríamos los próximos días.

Además, dividimos las tareas según prioridad y dificultad utilizando una herramienta sencilla, Wunderlist, en la que creamos varias listas y cada uno escogía una funcionalidad que le parecía más cómoda y sencilla.

Así cada semana o dos semanas nos reuníamos con nuestro director para explicarle los avances y algunas dudas que surgieron en el proceso de desarrollo.

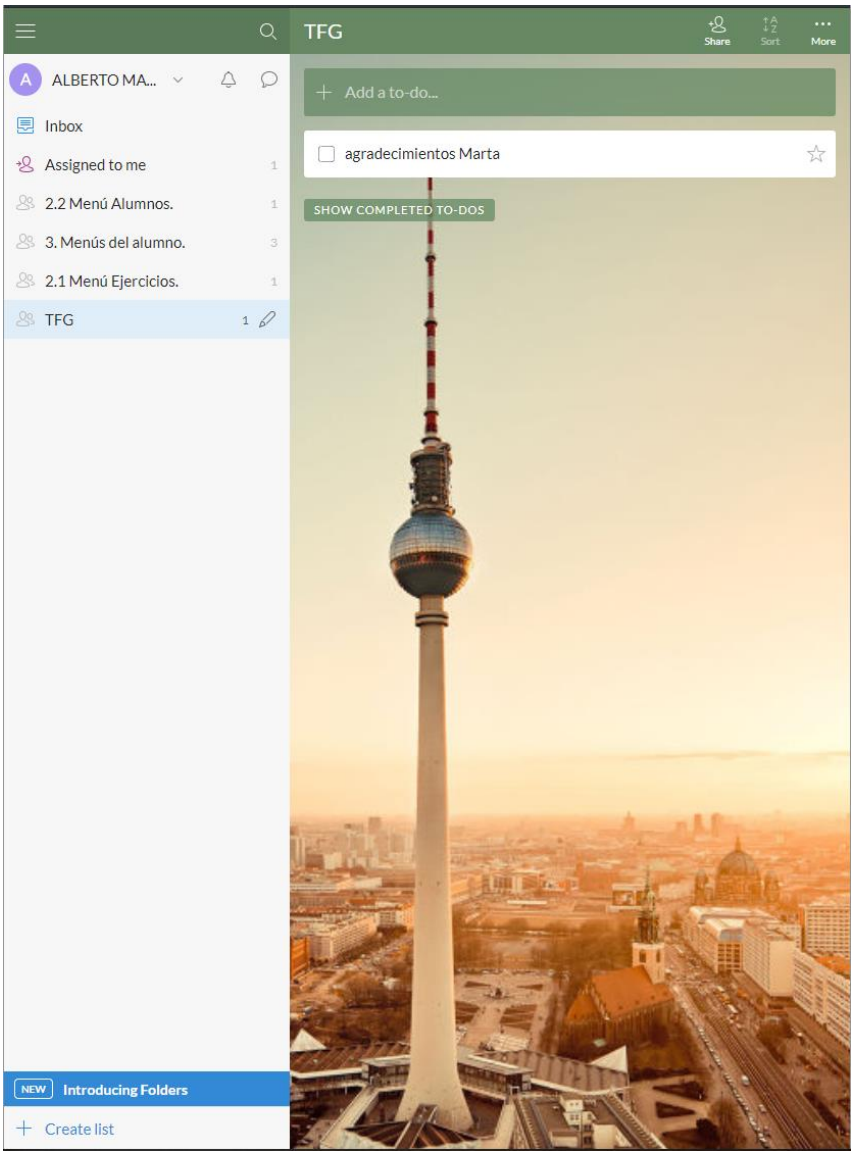


Imagen de la aplicación Wunderlist

6. Proceso de desarrollo

Para garantizar la calidad y gestionar los cambios del código empezamos escogiendo JavaScript como lenguaje único en toda la aplicación y descartamos la posibilidad de utilizar PHP. Así intentamos simplificar la cantidad de tecnologías conviviendo en la página web que íbamos a tener, luego escogimos el diseño de carpetas que tendría la aplicación, el núcleo de la aplicación estaría en la raíz.

Todo lo referente al diseño visual de la aplicación lo colocamos en la carpeta public, donde decidimos separar el grueso de cada tipo de tecnología diferente.
Todo el código en JavaScript se encuentra en la carpeta js. El estilo de la página lo pusimos en la carpeta CSS y el HTML estaría en la raíz de la carpeta public.

Los dos habíamos trabajado ya con MySQL, con lo que escogimos esta base de datos para almacenar los datos de la aplicación. No queríamos juntar los datos de la aplicación y a los usuarios, junto con las tablas, procedimientos y pruebas que iba a almacenar este alumno para comprobar que su procedimiento y ejecutar las pruebas sobre el mismo estuviese bien.

Decidimos usar Oracle únicamente para la evaluación de ejercicios, el alumno se conectaría para almacenar el procedimiento y el resto de información que se le mostrase estaría en MySQL.

Al principio empezamos utilizando los módulos que ya conocíamos para realizar el Login, pensamos que todo sería así de fácil he intentamos meter elementos más innovadores en la página, pero vimos que conocer y saber usar algunos elementos puede llevar más tiempo del que suponíamos.
Esto hizo que subir un ejercicio en la página del profesor fuese complicado.

Cuando habíamos conseguido hacer la parte del frontend descubrimos que almacenar procedimientos no sería tan sencillo como las consultas que hacíamos a la base de datos de MySQL. Y que la tecnología que tenía Oracle para conectar Node.js con Oracle tendría algunas limitaciones.

Por eso utilizamos algunos procedimientos almacenados para realizar ciertas funciones.

Sin duda alguna lo que más tiempo nos llevó fue realizar todas las operaciones que teníamos que hacer con oracledb ya que ninguno de los dos había trabajado antes con esta tecnología.

Cuando terminamos nos dimos cuenta de que aún faltaba mucha funcionalidad por hacer y que la página no tenía un aspecto muy cuidado con lo que fue necesario darle nuestro propio estilo. La mayor parte de este estilo nos la facilitó Bootstrap, sin el cual no habría sido tan sencillo diferenciar nuestra página de otros portales.

Finalmente distribuimos bien los menús para que el profesor pueda acceder a cada funcionalidad de la forma más fácil posible y siempre siguiendo los requisitos de la aplicación.

7. Arquitectura de la aplicación

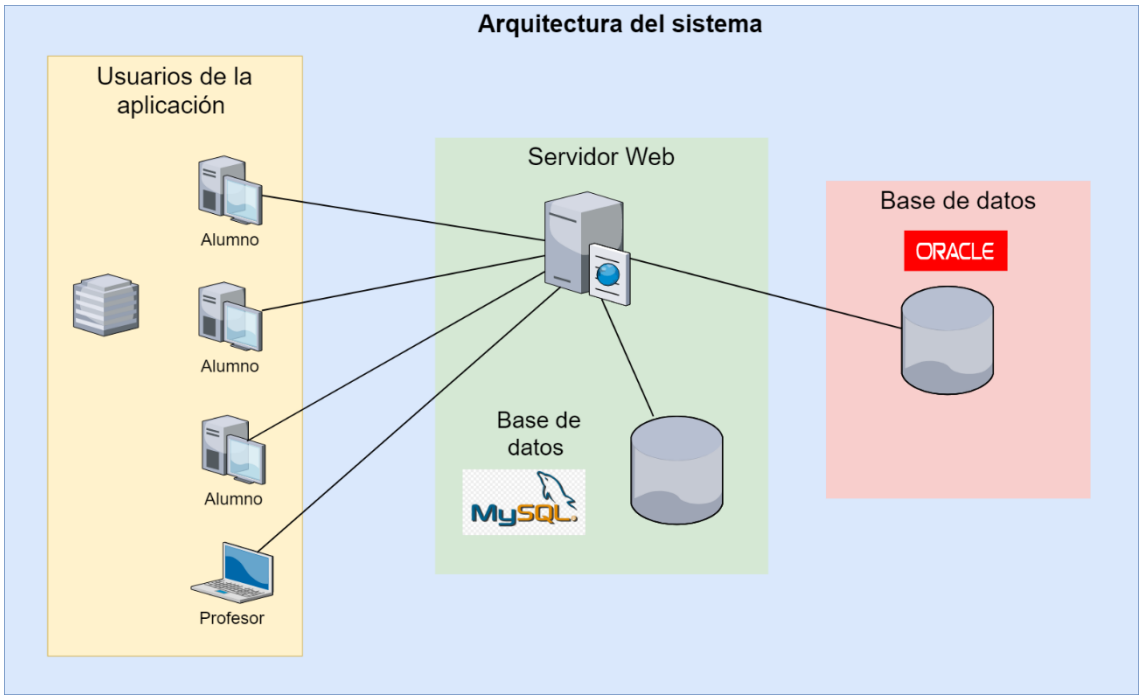
La versión de desarrollo con la que nosotros hemos trabajado aloja todas las tecnologías en un mismo equipo, cliente, servidor y bases de datos.

Pero estos componentes son completamente independientes unos de otros.

La arquitectura de la aplicación se divide en tres partes diferenciadas, por una parte, todos los usuarios disponen de una aplicación web que les sirve de cliente para la aplicación. El servidor web se encarga de gestionar las peticiones de los clientes y estará alojado en la misma máquina que la base de datos de MySQL.

La base de datos de Oracle se encontrará alojada en otro equipo.

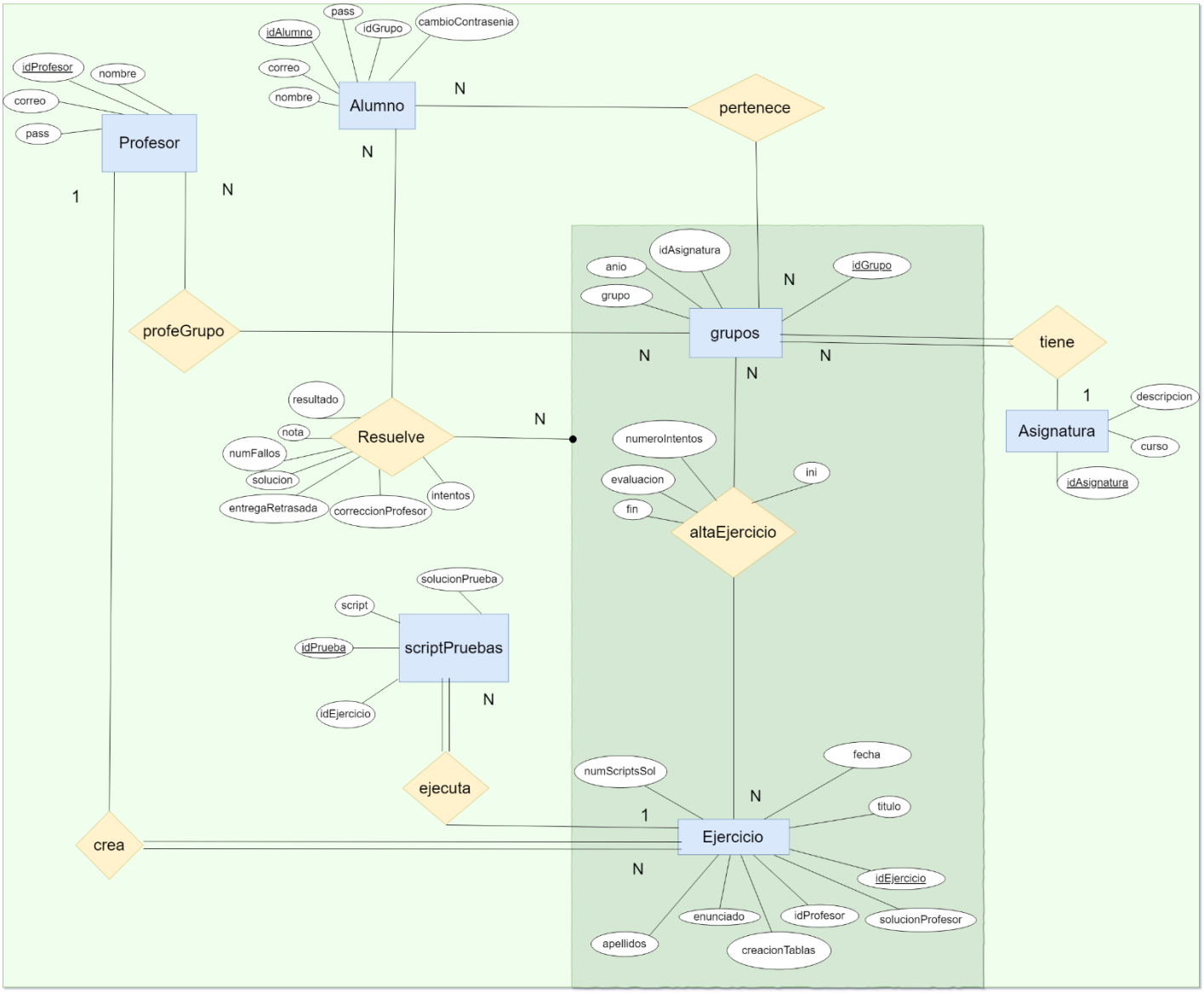
El siguiente esquema describe la arquitectura de la aplicación.



Se puede apreciar que la base de datos MySQL al ser la que más uso tendrá y al almacenar los datos de la aplicación se encontrará en la misma máquina que el servidor web, esta decisión permite simplificar la arquitectura y la cantidad de equipos necesarios.
Mientras que la base de datos Oracle utilizada para la corrección de los procedimientos estará en otra máquina aparte.

8. Diseño de la base de datos

La siguiente imagen se corresponde con el diagrama Entidad Relación de la base de datos MySQL de la aplicación.



Este diagrama muestra el diseño final de la base de datos. Se partió de un diseño inicial de base de datos muy básico y en varias interacciones de diseño casi todas las tablas han sido ampliadas o añadidas para proporcionar más funcionalidades a la aplicación.

A continuación, explicaremos los principales elementos del diagrama.

El profesor está conectado con los grupos de clase, de tal forma que un grupo puede tener más de un profesor y un profesor puede estar en varios grupos. En el esquema aparece representado como profeGrupo.

La entidad asignaturas permite que los grupos pertenezcan a una asignatura. De esta forma, una asignatura puede tener muchos grupos, pero un grupo solo pertenece a una asignatura.

El profesor es el encargado de crear los ejercicios, un profesor puede crear todos los ejercicios que quiera. Un ejercicio podrá ejecutar un número determinado de scripts de prueba, el campo de la tabla se llama numScriptsSol.

La clave primaria es idEjercicio, el campo fecha indica la fecha en la que se creó el ejercicio, solucionProfesor guarda el resultado de la corrección del ejercicio del profesor por parte de los scripts.

El campo creacionTablas guarda el script que genera todas las tablas relacionadas con el ejercicio y las rellena.

El enunciado guarda un PDF con el enunciado del problema en la base de datos.

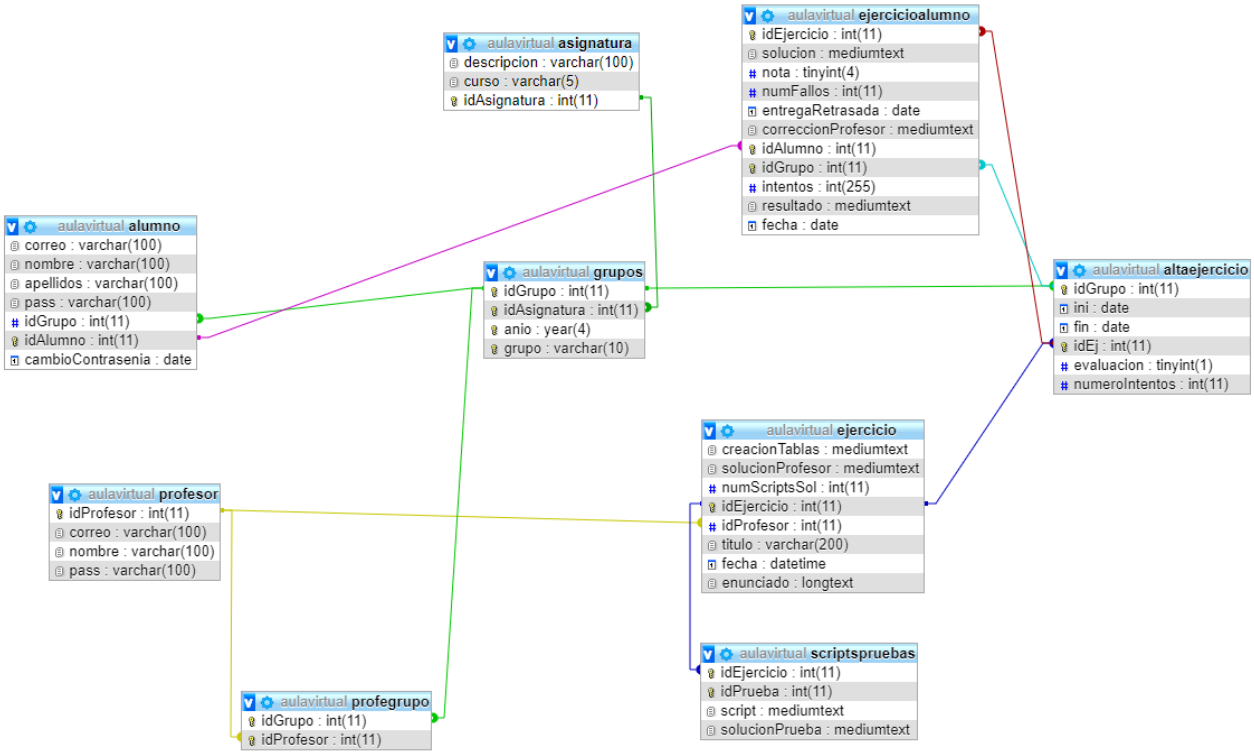
La relación alta ejercicio es la encargada de hacer que los problemas creados por el profesor se asocien a un grupo de clase, de tal forma que un ejercicio puede pertenecer a cualquier grupo y un grupo puede tener asignados los ejercicios que los profesores hayan seleccionado.

Un alumno puede estar dado de alta en más de un grupo y un grupo tendrá asignados los alumnos que añada el profesor a este grupo.

Los alumnos resuelven los ejercicios dados de alta para sus grupos mediante la relación resuelve que conecta con esta agregación.

En la entidad Alumno el campo cambioContrasenia guarda la fecha en la que se cambió la última vez la contraseña, esto se hace para que el alumno cambie la contraseña de un año para otro. Es una forma de proporcionar más seguridad a las contraseñas del alumno.

El alumno entra en la aplicación escribiendo su correo y contraseña los cuales se almacenan en la base de datos junto con su nombre y apellidos.



Diseño final de la base de datos

La tabla asignaturas fue de las últimas en añadirse, ya que en un principio la aplicación solo se iba a utilizar para la asignatura de bases de datos. Pero se introdujo para una mejor escalabilidad a la hora de utilizar la aplicación para cualquier asignatura relacionada.

Los scripts que utilizaremos para corregir los ejercicios los incluimos en la tabla scriptspruebas. Cada ejercicio tendrá un conjunto de scripts de prueba.

La solución producida por estos scripts en el procedimiento del alumno sin embargo se recogerá y almacenará como un único bloque de texto en el campo resultado, de la tabla ejercicioalumno. Los comentarios del profesor sobre el código del alumno, junto con el código del alumno, se almacenan en esta tabla en el campo correccionProfesor.

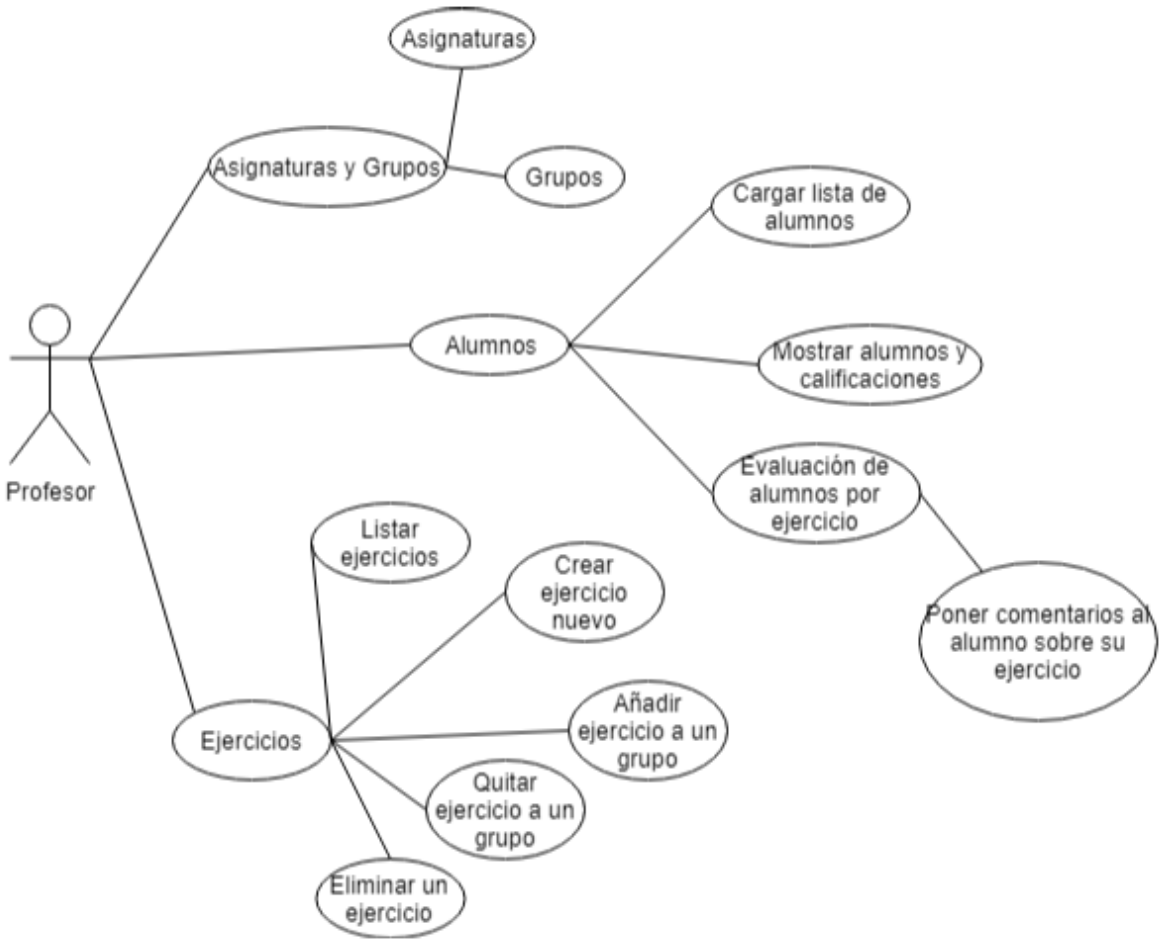
Los ejercicios y exámenes serán evaluables por el alumno cuando la fecha actual este entre ini y fin. Una vez pasada la fecha de fin el alumno subir a la base de datos las soluciones de los ejercicios y exámenes.

9. Casos de uso

Un caso de uso es la descripción de las actividades que deberá realizar alguien o algo para llevar a cabo un proceso. los personas o entidades que participarán en un diagrama de caso de uso se llaman actores. En este contexto, los actores en nuestra aplicación serán el profesor y el alumno.

Profesor

Las posibilidades que tiene el profesor dentro de la aplicación se muestran en la siguiente figura. Se puede ver que tiene acceso a los módulos de asignatura, ejercicios y alumnos. Estos módulos contienen distintas operaciones como listar ejercicios que da la posibilidad además de editar un ejercicio concreto. Otras operaciones que puede hacer el profesor es evaluar el ejercicio de un alumno poniéndole una nota y/o un comentario sobre el propio código fuente del alumno.



Alumno

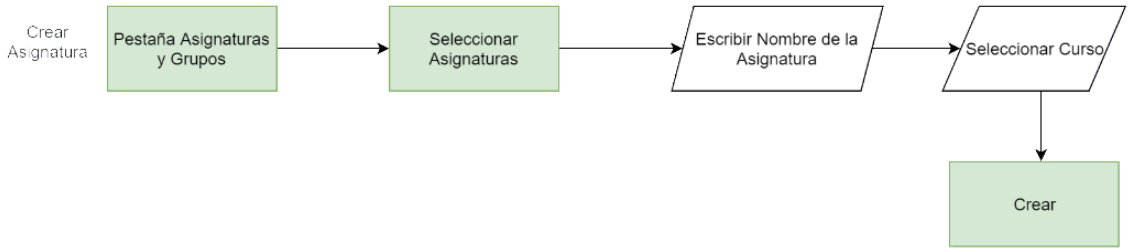
El alumno tiene la posibilidad de ver la lista de ejercicios de otros años, mostrar los ejercicios y exámenes de este año y poder subir su solución y probarla con los scripts de prueba, aunque correspondan a ejercicios de otros años o cuyo plazo de entrega ha terminado. Además hemos incluido un apartado de calificaciones para que también pueda tener un espacio para visualizar sus notas y tener una visión más general del curso.

10. Diagramas de flujo

Para mostrar el flujo de ciertos procesos en la aplicación hemos hecho los diagramas más relevantes para clarificar el recorrido de las acciones de un profesor, desde dar de alta una asignatura hasta poder corregir los ejercicios de sus alumnos.

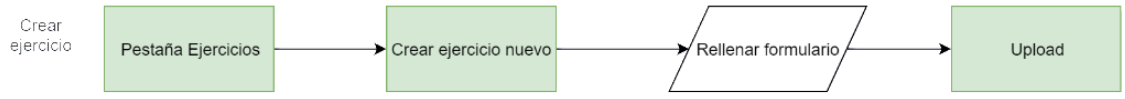
Crear asignatura

Para que un profesor pueda crear ejercicios y asignarlos a un curso y grupo debe tener asignaturas y grupos creados, el flujo de creación tanto de asignaturas como grupos es el siguiente.



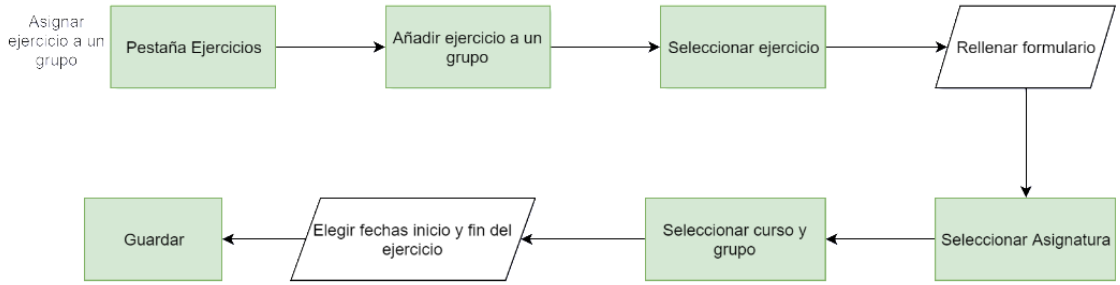
Crear ejercicio

independientemente de la asignatura y grupo el profesor puede crear ejercicios y listar los que no se han asignado a ningún grupo.



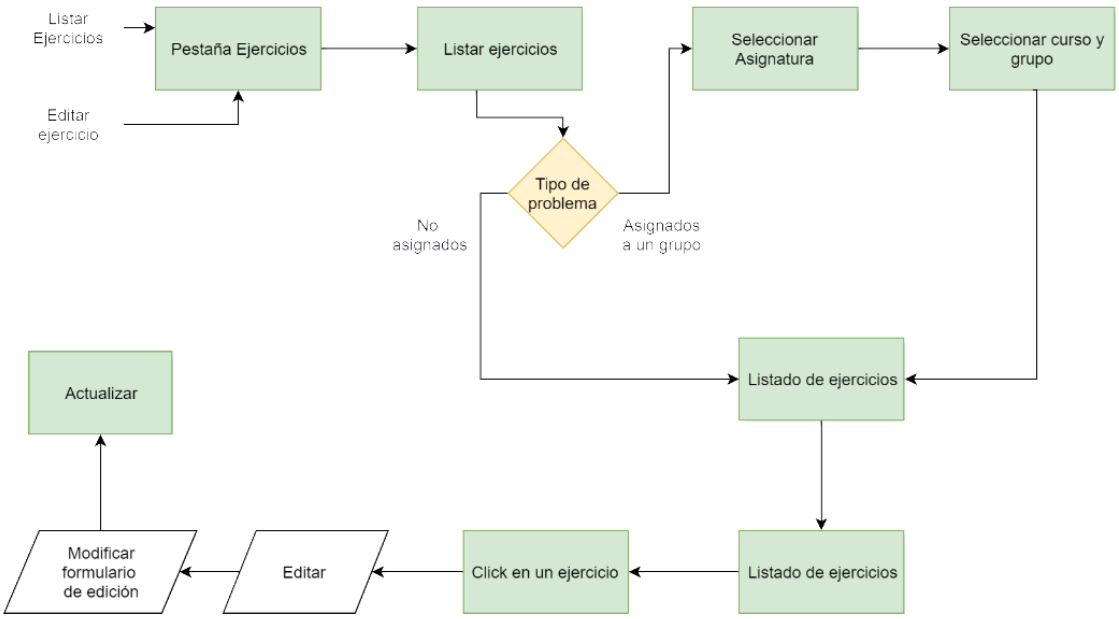
Asignar ejercicio a un grupo

Para poder asignar un ejercicio a un grupo y que los alumnos puedan resolverlo el profesor debe realizar las acciones que se indican a continuación, escogiendo el ejercicio, después indicando el número de intentos, el tipo de ejercicio si es de evaluación o no y las fechas en las que estará disponible el ejercicio.



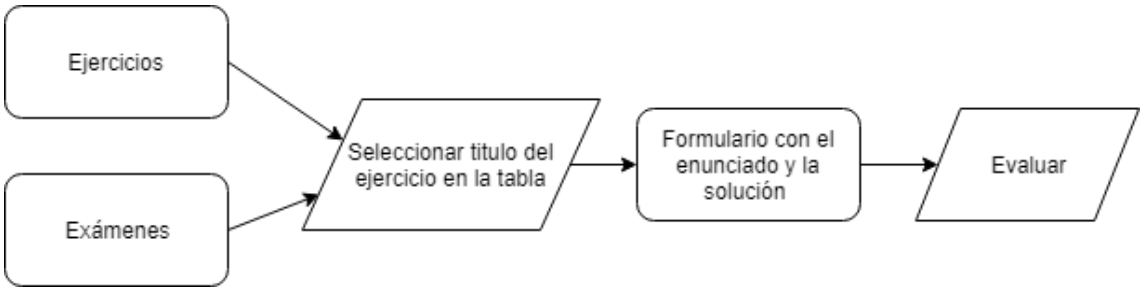
Listar y editar ejercicios.

Además de listar ejercicios que no se han asignado, el profesor puede ver y editar los ejercicios que se han asignado a alguna asignatura y grupo a los que él pertenece. En la edición de ejercicios el profesor podrá eliminar los scripts de prueba que desee además de añadir texto y visualizar los demás scripts, adicionalmente podrá cambiar el título del ejercicio y el enunciado.



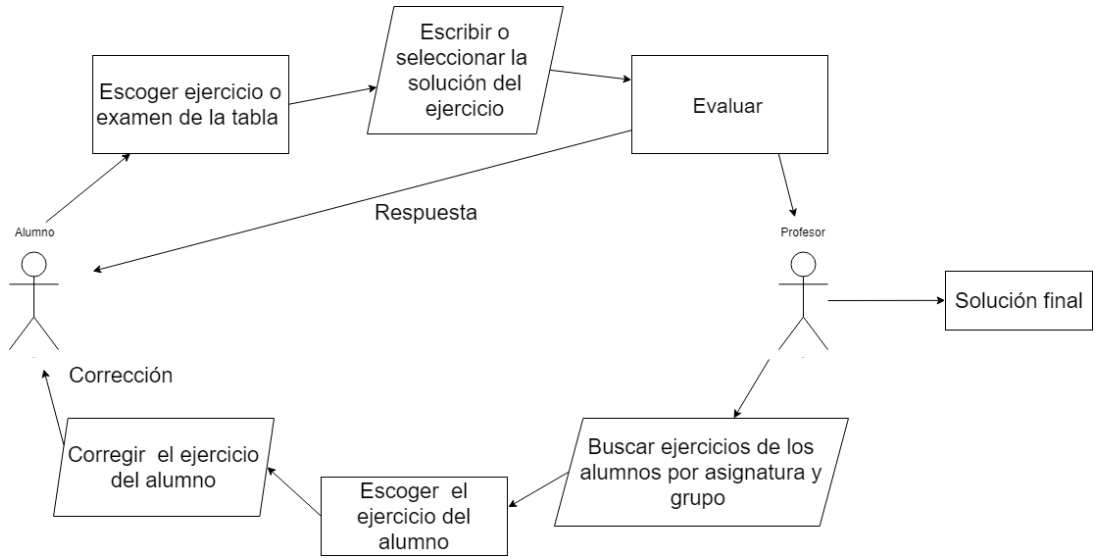
Evaluar un ejercicio por parte de un alumno

La operación principal que realiza el alumno es la de ejecutar un procedimiento en la aplicación, para ello tendrá que listar un tipo de problema, ejercicios o exámenes elegir el problema a ejecutar de la lista que se le muestra subir su solución y ejecutarla.
El flujo para listar tanto ejercicios como exámenes para después ejecutarlos por parte del alumno es el siguiente.



Este diagrama describe como un alumno selecciona el ejercicio o examen de una tabla y tras rellenar el campo solución podrá evaluar el ejercicio.

El siguiente diagrama muestra como un profesor interacciona con el alumno para hacer una mejor corrección del ejercicio.



El alumno realiza a la acción de escoger un ejercicio, escribir o subir su solución y solicitar a evaluarlo. El profesor será el encargado de escoger su ejercicio y ver los resultados del alumno.

Aparte de ver las correcciones automáticas, el profesor puede añadir un comentario o una corrección en el código del alumno, y cuando el alumno entre de nuevo en su ejercicio, el comentario del profesor aparecerá resaltado en su anterior entrega.

Esto permite que el alumno pueda tener un ciclo de realimentaciones por parte del profesor, para entender mejor como se tiene que realizar el procedimiento PL/SQL.

11. Implementación

A continuación, describimos como hemos desarrollado la parte del cliente y del servidor, detallando las tecnologías usadas tanto en el lado del cliente como en el lado del servidor. Además, se describe todos los elementos del interfaz de usuario de la aplicación.

Aplicación cliente

Para la parte del cliente decidimos utilizar todo lo que habíamos aprendido en la carrera sobre desarrollo de aplicaciones web, por lo que después de hablarlo entre los miembros del equipo decidimos utilizar JavaScript tanto en el cliente como en el servidor.

Para la ejecución en el cliente separamos el código JavaScript por cada HTML creado en nuestra aplicación. En estos ficheros utilizamos JQuery y llamadas Ajax para entablar comunicación con el servidor. Además, implementamos un plugin en jQuery con el que se le agrega funcionalidad extra al parámetro "$", con esto implementamos el uso de cookies en el cliente.

El aspecto de un plugin es el siguiente:

```
(function ( $ ) {
  ---- funciones internas --------
}( jQuery ));
```

Donde "funciones internas" se deben implementar los métodos que se quiera.

Gracias a esto podemos guardar las variables del usuario que se ha iniciado sesión y las podemos ver y editar en cualquier punto de la aplicación en la que el plugin esté activo.

Para poder recoger los valores del HTML se utilizó el Modelo de Objetos de Documento (DOM) un modelo estándar sobre cómo pueden combinarse los objetos, y una interfaz estándar para acceder a ellos y manipularlos.

El DOM permite el acceso dinámico programable para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos.

La estructura básica de nuestros ficheros del lado del cliente es la siguiente:

$(document).ready(function(){…}) es el punto de partida donde comienza la aplicación.

$.galleta().getc("usuario") es la operación para coger el usuario almacenado en la cookie. Una vez tenemos el usuario hay que comprobar que existe o será redireccionado a la página de inicio. Una vez sabemos que el usuario es válido se carga el tipo de cabecera que le corresponde, la del profesor o la del alumno.
Después de todas las comprobaciones se implementan las funciones a las que corresponde este fichero, dar de alta, listar o ejecutar un procedimiento.

En la siguiente figura se muestra un fragmento de código correspondiente a inicio de todos los códigos escritos en JavaScript en el lado del cliente.

```
var user;
$(document).ready(function() {
    var options={}
    $.galleta(options);
    user = $.galleta().getc("usuario");

    if(user != "undefined"){
        user = JSON.parse(user);
        $("#cabecera").load("cabecera.html",function(responseTxt, statusTxt, xhr){
            if(statusTxt == "success"){
                $("#nombre_usuario").text(user.nombre);
                $("#roll_usuario").text(user.user + " :");
                if( user.user.localeCompare("profesor")===0){
                    $("#menu").load("menuProfesor.html");
                    $(".ejs_ex").addClass("hidden");
                    $("#aYG").addClass("hidden");
                }else if(user.user.localeCompare("alumno")===0){
                    $("#menu").load("menuAlumno.html");
                    $("#aYG_usuario").text(user.descripcion +" "+user.curso+"º"+user.grupo);
                }
                $("#desconectar").click(function(event) {
                    $.galleta().setc("usuario", "undefined", "Thu, 01 Jan 1970 00:00:01 GMT");
                    var link = window.location.href;
                    var res = link.split("/");
                    window.location = res[1] + "/";
                });
            }
        });
```

Aplicación servidor

Antes de describir cómo funciona la parte del servidor nos gustaría introducir al entorno de ejecución que hace posible todas nuestras conexiones con las bases de datos.

Concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node está diseñado para construir aplicaciones en red escalables.

Esto contrasta con el modelo de concurrencia más común hoy en día, donde se usan hilos del Sistema Operativo. Las operaciones de redes basadas en hilos son relativamente ineficientes y son muy difíciles de usar. Además, los usuarios de Node están libres de preocupaciones sobre el bloqueo del proceso, ya que no existe. Casi ninguna función en Node realiza I/O directamente, así que el proceso nunca se bloquea. Debido a que no hay bloqueo es muy razonable desarrollar sistemas escalables en Node.

Node tiene un diseño similar y está influenciado por sistemas como Event Machine de Ruby o Twisted de Python. Node lleva el modelo de eventos un poco más allá, este presenta un bucle de eventos como un entorno en vez de una librería.

Node ingresa el bucle de eventos después de ejecutar el script de entrada. Node sale del bucle de eventos cuando no hay más callbacks que ejecutar, comportándose de una forma similar a JavaScript en el navegador(el bucle de eventos está oculto al usuario).

HTTP es ciudadano de primera clase en Node, diseñado con operaciones de streaming y baja latencia en mente. Esto hace a Node candidato para ser la base de una librería o un framework web.

Solo porque Node esté diseñado sin hilos, no significa que no pueda aprovechar los múltiples núcleos de su sistema, los procesos hijos pueden ser lanzados usando la API child_process.fork(), la cual está diseñada para comunicarse fácilmente con el proceso principal. Construida sobre la misma interfaz está el módulo cluster, el cual permite compartir sockets entre procesos para activar el balanceo de cargas en sus múltiples núcleos.

La parte de la aplicación en el lado del servidor está estructurada en estos 6 módulos:

- oracleProfesor.js
- oracleAlumno.js
- DAOS
    - daoAsignatura.js
    - daoEjercicio.js
    - daoUser.js
- app.js

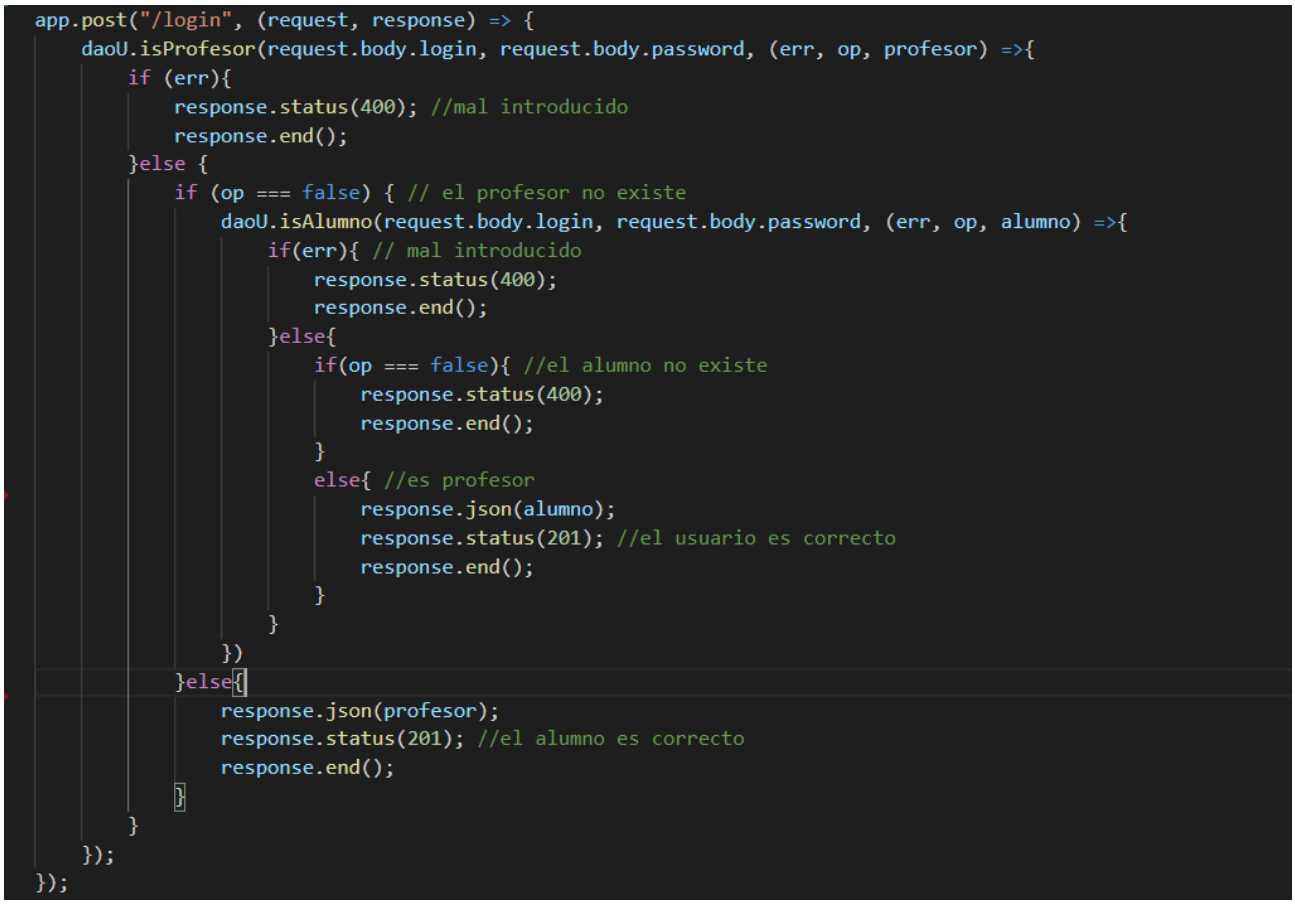
Los tres módulos de acceso a datos se encuentran en la carpeta daos, mientras que los otros tres están directamente en la raíz del proyecto.

App

Como punto central de la aplicación tenemos el app.js, que recibe, envía y ejecuta las peticiones que le llegan. Al ser un sistema de tipo REST las peticiones son de tipo get, post, put y delete.
Los objetos se manipulan a través de la URI, por lo que se usa como identificador único del recurso de nuestro sistema.
Para el manejo de dichas peticiones usamos el paquete express que además proporciona librerías para trabajar con cookies, sesiones y datos POST entre otras.

Además, implementa el manejo de la base de datos de mysql mediante el módulo "mysql", por el que se crea y define el pool de conexiones que después se pasará a cada DAO para realizar consultas.

Aquí se muestra un ejemplo de cómo se ha usado el módulo express para capturar las peticiones del cliente, en este caso es la función que inicia la sesión en la aplicación, como hemos explicado previamente, puede iniciar sesión tanto un profesor como un alumno por lo que esta función lanza una consulta contra Mysql y comprueba si el usuario no existe, si existe y lo ha introducido mal y, si es alumno o profesor; el resultado de esta consulta se la envía al cliente con la operación response.json(profesor) en caso de ser profesor o response.json(alumno) en caso de ser alumno, además también se envía el status para saber si el usuario es incorrecto o qué tipo de usuario se ha iniciado sesión en la aplicación, si profesor o alumno.



```
app.post("/login", (request, response) => {
    daoU.isProfesor(request.body.login, request.body.password, (err, op, profesor) =>{
        if (err){
            response.status(400); //mal introducido
            response.end();
        }else {
            if (op === false) { // el profesor no existe
                daoU.isAlumno(request.body.login, request.body.password, (err, op, alumno) =>{
                    if(err){ // mal introducido
                        response.status(400);
                        response.end();
                    }else{
                        if(op === false){ //el alumno no existe
                            response.status(400);
                            response.end();
                        }
                        else{ //es profesor
                            response.json(alumno);
                            response.status(201); //el usuario es correcto
                            response.end();
                        }
                    }
                })
            }else{
                response.json(profesor);
                response.status(201); //el alumno es correcto
                response.end();
            }
        }
    });
});
```

Imagen operación iniciar sesión en el servidor

Oracle Profesor

El archivo dbconfig es el .js encargado de cargar la configuración por defecto de la conexión de la aplicación con la base de datos de Oracle, este permite definir los parámetros de la configuración para que toda la conexiones se hagan con el mismo usuario, contraseña y dirección.

```
//https://github.com/oracle/node-oracledb/blob/master/examples/dbconfig.js
module.exports = {
    user          : process.env.NODE_ORACLEDB_USER || "SYS",

    // Instead of hard coding the password, consider prompting for it,
    // passing it in an environment variable via process.env, or using
    // External Authentication.
    password      : process.env.NODE_ORACLEDB_PASSWORD || "SYS",

    // For information on connection strings see:
    // https://oracle.github.io/node-oracledb/doc/api.html#connectionstrings
    connectString : process.env.NODE_ORACLEDB_CONNECTIONSTRING || "localhost",

    // Setting externalAuth is optional.  It defaults to false.  See:
    // https://oracle.github.io/node-oracledb/doc/api.html#extauth
    externalAuth  : process.env.NODE_ORACLEDB_EXTERNALAUTH ? true : false
};
```

Esta clase se encarga de la creación de los usuarios en la base de datos de Oracle y de la corrección del ejercicio de Oracle del alumno.

Los permisos de creación de los usuarios son los mismos en caso del alumno y del profesor ya que el procedimiento que los crea es el mismo.

Para poder crear un usuario se instala y ejecuta desde el servidor web el siguiente procedimiento almacenado de PL/SQL en la base de datos de Oracle:

```
create or replace PROCEDURE ALTA_USUARIO(uid VARCHAR2) AS
        num_alumnos number;
    BEGIN
        SELECT COUNT(*) INTO num_alumnos FROM all_users where USERNAME = uid;
        DBMS_OUTPUT.PUT_LINE(uid);
        IF num_alumnos=1 THEN

            EXECUTE IMMEDIATE 'DROP USER '||uid||' CASCADE';
        END IF;

        EXECUTE IMMEDIATE 'CREATE USER '||uid||' IDENTIFIED BY '||uid;
        EXECUTE IMMEDIATE 'GRANT RESOURCE TO '||uid;
        EXECUTE IMMEDIATE 'GRANT CONNECT TO '||uid;
        EXECUTE IMMEDIATE 'GRANT CREATE TRIGGER TO '||uid;
        EXECUTE IMMEDIATE 'GRANT CREATE SEQUENCE TO '||uid;
        EXECUTE IMMEDIATE 'GRANT CREATE TABLE TO '||uid;
        EXECUTE IMMEDIATE 'GRANT CREATE SYNONYM TO '||uid;
        EXECUTE IMMEDIATE 'GRANT CREATE VIEW TO '||uid;
        EXECUTE IMMEDIATE 'GRANT debug any procedure, debug connect session TO '||uid;
        EXECUTE IMMEDIATE 'GRANT EXECUTE ON sys.write_log TO '||uid;
    END;
```

El usuario y contraseña de Oracle para el usuario creado serán los mismos ya que la información que se almacene en esta base de datos será de forma temporal y sólo se utilizará para corregir los procedimientos del alumno.

El servidor de la aplicación que ejecuta el procedimiento de ALTA_USUARIO debe tener permisos de SYSDBA para poder crear los usuarios. Por esto, es necesario que la creación de usuarios se haga desde el usuario SYS de Oracle.

Este procedimiento da permiso a los usuarios para poder crear tablas, procedimientos, triggers, tablas y vistas y además ejecutar el procedimiento write_log almacenado en el usuario SYS de Oracle.

EL procedimiento write_log almacena en un fichero toda la salida por pantalla de tipo DBMS_OUTPUT.

Se muestra en la siguiente figura:

```
create or replace PROCEDURE write_log(nombreFic VARCHAR2) AS
  l_line VARCHAR2(255);
  l_done NUMBER;
  l_file utl_file.file_type;
  BEGIN
      l_file := utl_file.fopen('TMP', nombreFic, 'W');
      LOOP
          EXIT WHEN l_done = 1;
        dbms_output.get_line(l_line, l_done);
        utl_file.put_line(l_file, l_line);
      END LOOP;
      utl_file.fflush(l_file);
      utl_file.fclose(l_file);
  END write_log;
```

Los resultados de las pruebas escogidas por el profesor se guardarán por defecto en una carpeta temporal creada desde Oracle y cada uno de los ficheros que se creen o editen en esta carpeta llevarán el identificador de cada usuario de Oracle para que varios usuarios de Oracle puedan ejecutar pruebas concurrentemente.

Estos identificadores los hemos creado usando el id del alumno o del profesor junto con el nombre del alumno o del profesor. Tanto el id como el nombre son los que se encuentran en la base de datos de MySQL.

Los archivos de los profesores llevarán el identificador con una p al principio para distinguirlos de los alumnos y que no se dé el posible caso de que un profesor o un alumno tuviesen el mismo id y nombre.

Los archivos que no existen se crean y si el archivo ya existía de haber grabado una prueba anterior sobrescribimos este archivo. Esto lo hacemos para recoger individualmente el resultado de cada una de las pruebas que lanzamos al procedimiento del alumno o del profesor.

Para realizar estas operaciones con las carpetas y los ficheros hemos utilizado una librería de node.js llamada fs, que es la que más se utiliza para tratar archivos [1].

Oracle Alumno

En este archivo se encuentra el código encargado de corregir los procedimientos de los alumnos y recoger las soluciones y errores que se produzcan.

Este código se llama cuando el alumno manda a corregir un procedimiento y devuelve los resultados de la corrección.

La creación del alumno se realiza desde Oracle profesor. Esta clase solo utiliza el nombre del usuario creado anteriormente para corregir el procedimiento.

La clase empieza creando las tablas en la base de datos Oracle igual que se hace en el caso del profesor, esto consiste en ejecutar primero el código de creación de las tablas y después insertar los datos en estas.
Oracledb no permite ejecutar un script de SQL, con lo que dividimos cada consulta en un bloque y las ejecutamos en la base de datos de una en una.
La creación e inserción debe de hacerse de forma ordenada ya que se ejecutará de arriba abajo el script de las tablas introducido por el profesor.

A continuación, se almacena el procedimiento del alumno, la diferencia del alumno con el profesor es que aquí se recogen los errores al corregir el procedimiento derivados de la compilación del ejercicio por parte de Oracle. Este procedimiento consta únicamente de un bloque de código.

Por último, se corregirá el procedimiento almacenado anteriormente pasando cada uno de los scripts de prueba del profesor. Los resultados se recogerán del fichero guardado en la carpeta temporal y se almacenarán en un array para ser mostradas.
En cada uno de estos scripts se sustituye PROC_alumno por el usuario formado anteriormente y se deja la extensión .log .
El resultado puede ser correcto, tener algún aviso o ser incorrecto.
En caso de que el problema haya tenido el un error de compilación al corregirlo en Oracle este no se tratará como un error generado de las pruebas, pero también se le mostrará al alumno.
Los cuatro casos posibles se almacenan en arrays independientes, para facilitar el trabajo a la hora de tratarlos en el cliente.

Solo se guardan en la base de datos la solución de los ejercicios que están clasificados como exámenes.

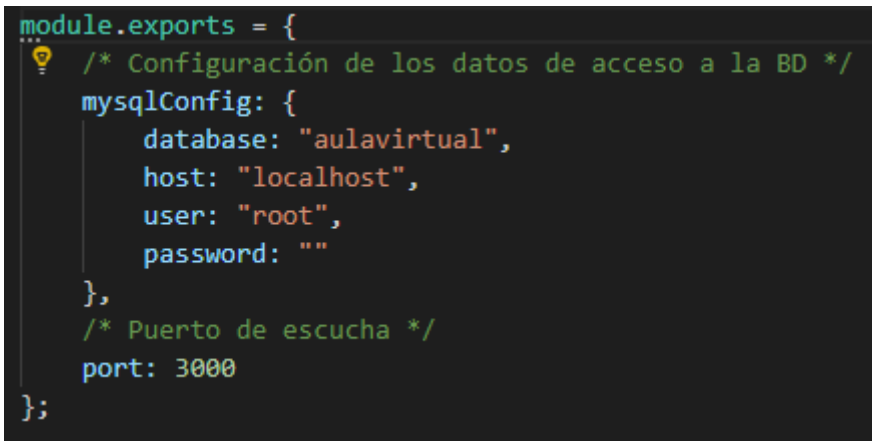
DAOS (Data Access Object)

El DAO es un Patrón de diseño utilizado para acceder a la capa de datos. Se suele utilizar en Java para aislar a una aplicación de la tecnología de persistencia Java subyacente.

En nuestro proyecto hemos usado este patrón para realizar las querys lanzadas a MySQL y Oracle, las operaciones principales que suelen hacer estos objetos son las de crear, listar, actualizar y borrar, CRUD (Create, Read, Update y Delete).

Todas las clases de los daos tiene en común un constructor para utilizar el pool de conexiones creado en el app.js el cual será encargado de conectar con la base de datos MySQL.

Nuestra configuración del servidor se encuentra en el archivo config.js el cual se encarga de definir la conexión. Estos datos los hemos separado para desacoplar el código y abstraerlo de las distintas capas. En la siguiente imagen se muestra un ejemplo del código para la configuración en local:



Todas las clases definidas a continuación comparten el mismo constructor definido de la siguiente forma:

Dao Asignatura

En este dao se encuentran todas las consultas hecha a la base datos para obtener o guardar información de una asignatura.

A parte de crear, eliminar y cargar las asignaturas también hay algunas hechas para ciertos menús como:

- Listar Asignaturas:

  select *
  from asignatura a join grupos g join profegrupo pg
  on a.idAsignatura = g.idAsignatura
  and pg.idGrupo = g.idGrupo
  and pg.idProfesor = ?
  and g.anio >= ?
  group by g.idAsignatura

  Dependiendo del profesor y del año cargará una lista con los datos de las asignaturas.

- Listar Asignaturas por ejercicio y grupo

  select *
  from asignatura a join grupos g join profegrupo pg
  on a.idAsignatura = g.idAsignatura
  and pg.idGrupo = g.idGrupo
  and pg.idProfesor = ?
  and g.anio = ?
  and pg.idGrupo in (select ae.idGrupo from altaEjercicio ae where idEj = ?)
  group by g.idAsignatura

  Esta consulta trae los datos de una asignatura, un profesor y grupo asignado a esta asignatura.

Dao Ejercicio

Es el dao encargado de manejar todas las operaciones en las que se utiliza la tabla de ejercicios, por lo que en él se encuentran la mayoría de las consultas de nuestra aplicación.

A parte de subir, modificar y borrar ejercicios también se encarga de las operaciones relacionadas con los scripts de estos ejercicios.

Como `subir scripts`:

```
var sql = "INSERT INTO scriptspruebas (idEjercicio, idPrueba, script)
VALUES"
for(var i = 0; i < datos.length;i++){
	sql += " (?, ?, ?)"
	if(i < datos.length - 1){
		sql +=","
	}else{
		sql += ";"
```

```
	}
}
var info = [];
for(var j = 0; j < datos.length; j++) {
	info.push(idEjercicio);
	info.push(datos[j].id);
	info.push(datos[j].archivo);
}
```

La cual carga en la base de datos los scripts del ejercicio.

Dao User

Todas las consultas para comprobar las credenciales de los usuarios y la creación de estos se encuentran en este dao.

Evaluar alumno

```
SELECT *
from
	(SELECT a.nombre, a.apellidos, a.idAlumno, a.idGrupo,
	ea.idEjercicio,ea.solucion, ea.nota, ea.numFallos,
	ea.entregaRetrasada,ea.intentos,ea.resultado
	from ejercicioalumno ea join alumno a
	ON ea.idAlumno = a.idAlumno and a.idGrupo = ?)
a JOIN
	(SELECT e.idEjercicio,
	e.numScriptsSol,e.titulo,ae.evaluacion,ae.numeroIntentos
	from ejercicio e join altaejercicio ae
	ON ae.idEj = e.idEjercicio and ae.evaluacion = ?)
b ON a.idEjercicio = b.idEjercicio`
```

Dependiendo del grupo y de si es un examen o un ejercicio se trae de la base de datos, los datos y resultados del alumno para ese problema.

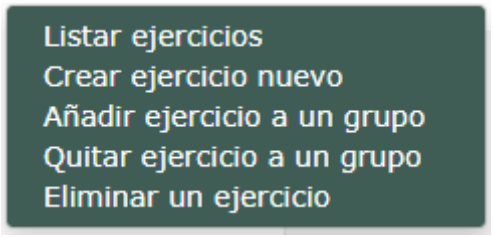
Interfaces del usuario

Vistas del alumno y del profesor con las distintas operaciones que puede hacer cada uno y cómo están distribuidas en la pantalla principal.

Interfaz del profesor.

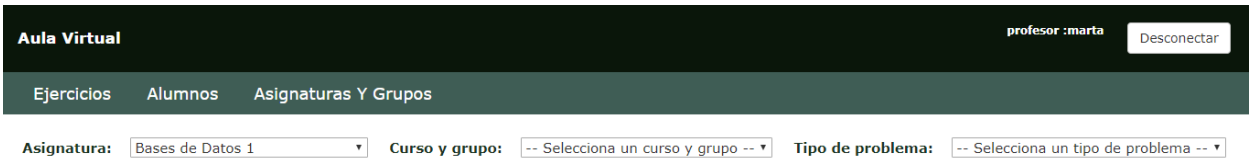
Menú Ejercicios

En este menú el profesor tiene la posibilidad de listar los ejercicios disponibles y poder editarlos, crear unos nuevos, asignarlos a grupos, desasignarlos y eliminarlos.

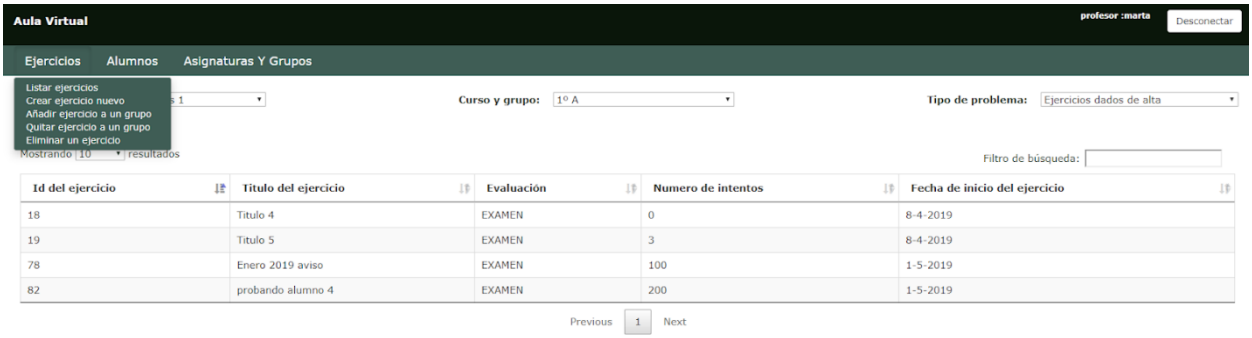


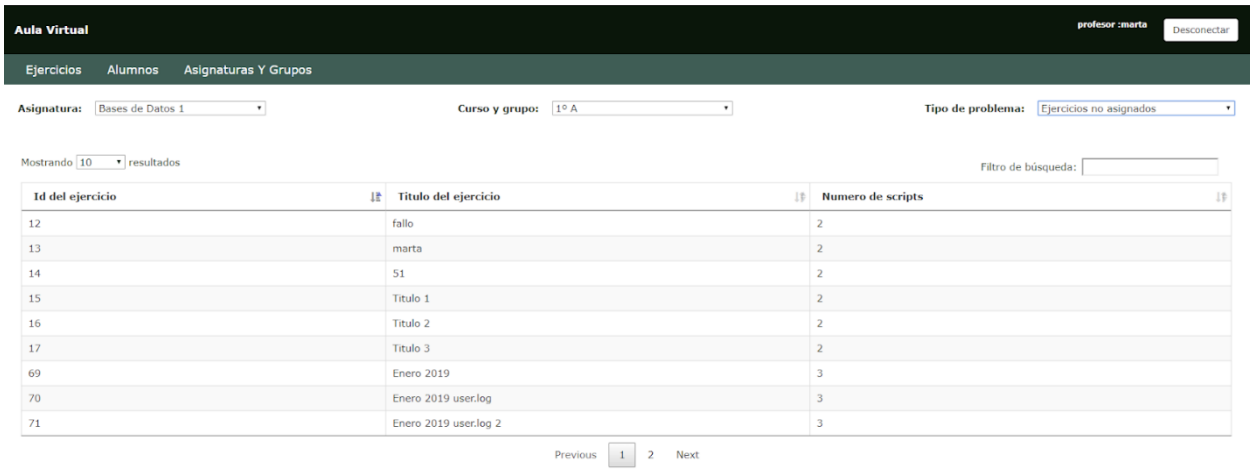
Listar ejercicios

Esta opción lo que permite es listar los ejercicios que se han dado de alta para una asignatura curso y grupo del año actual y en la que pertenece este profesor.
Primero se debe elegir la asignatura, y después el curso y grupo.

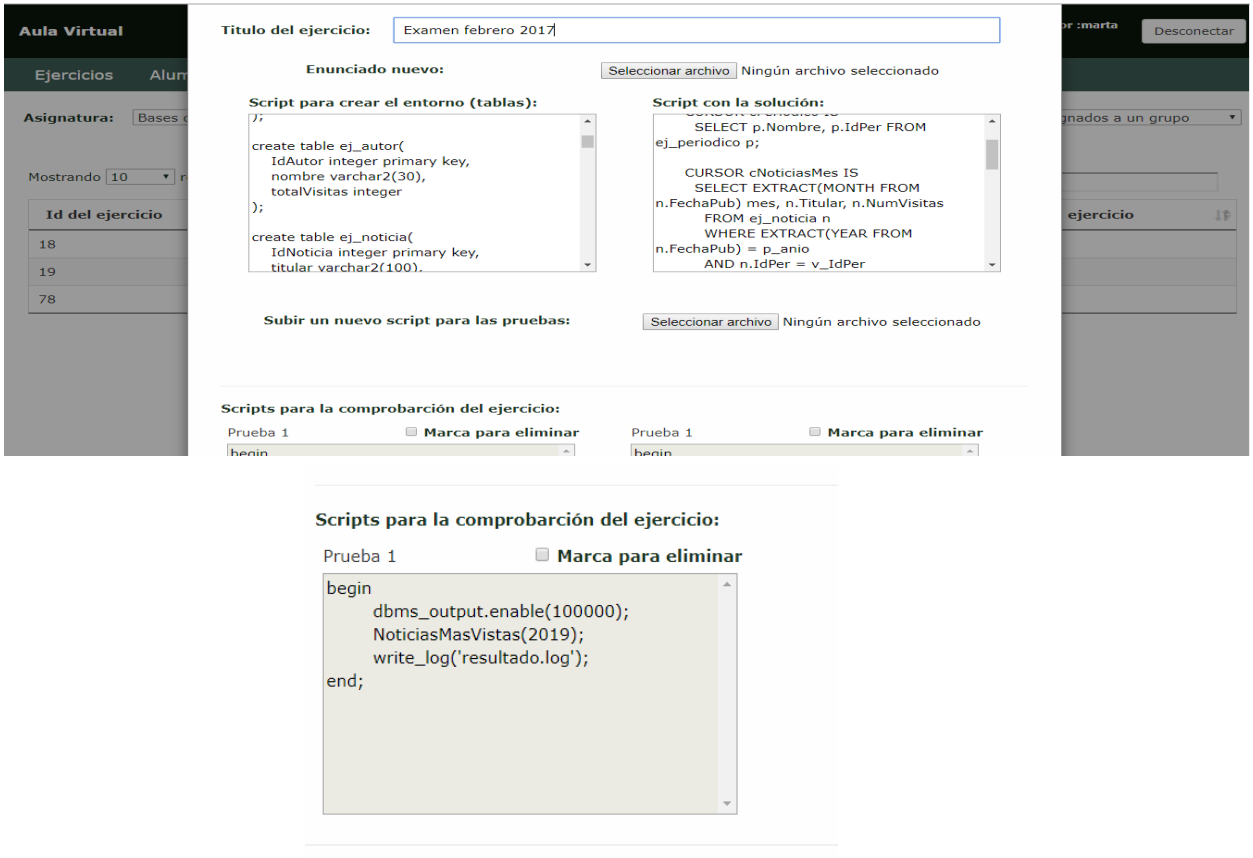


Después hay un último selector que nos indica qué tipo de problema queremos mostrar, puede ser ejercicios dados asignados a esa asignatura, curso y grupo especificados o ejercicios que todavía no se han asignado a ninguna asignatura.
En estos últimos no hace falta especificar asignatura ni curso y grupo.





Si se pasa el cursor por encima de cualquiera de ellos, se puede ver que son elementos seleccionables. Al pinchar en cualquiera de ellos, sale un resumen del ejercicio, indicando el título, el script para crear las tablas, el script con la solución, y los scripts con las pruebas.



Se da la posibilidad de poder visualizar dichos elementos y poder editarlos o sobrescribirlos, como es el caso del enunciado.

Además, está la opción de eliminar el script que no se quiera seleccionando el checkbox asociado a cada script. También se podrá añadir uno nuevo usando el botón para subir un nuevo script.

Crear ejercicio nuevo

En la vista encontramos un formulario que será el encargado de recoger los datos para poder subir un ejercicio, los campos que tendremos que rellenar son:

- El título del problema, este se muestra en todas las listas relacionadas con los ejercicios.

- El Enunciado del problema, tiene que ser un PDF, que se le mostrará al alumno en su pantalla de subir el ejercicio.

- El script con la creación de tablas, encargado de crearlas y rellenarlas. Este es el único documento que permite ejecutar más de una consulta simultáneamente.

- El script con el procedimiento a almacenar del profesor.

- Por último, encontramos un área para poder arrastrar los scripts de las pruebas que sube el profesor junto con el ejercicio, este elemento ofrece también la posibilidad de borrar los archivos subidos o subirlos mediante la selección con un explorador de carpetas común; una vez seleccionados se mostrará una previsualización de los archivos para saber si son correctos.

El botón Upload que se muestra junto con Remove y Browse es el encargado de almacenar el ejercicio completo en la base de datos.

Titulo:

Enero 2019

Enunciado:

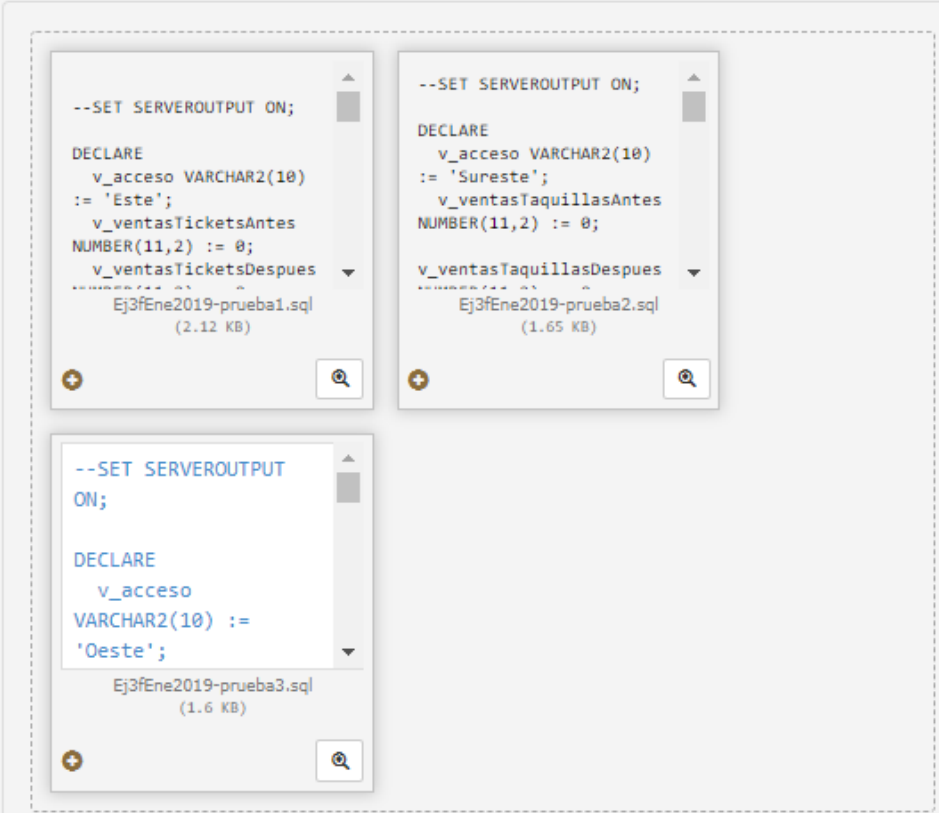
Enero2019.pdf Reset

Script para crear las tablas:

Ej3fEne2019-tablas.sql Reset

Script con la solucion:

Ej3fEne2019-solucion.sql Reset



3 files selected Remove Upload Browse ...

Las imágenes en miniatura de los scripts subidos coinciden con los scripts creados para el ejercicio. Se muestra un ejemplo en el apéndice A.

La corrección en caso del profesor será guardada en la base de datos de MySQL para evitar tener que corregir el ejercicio del profesor cada vez que un alumno mande una solución a corregir.

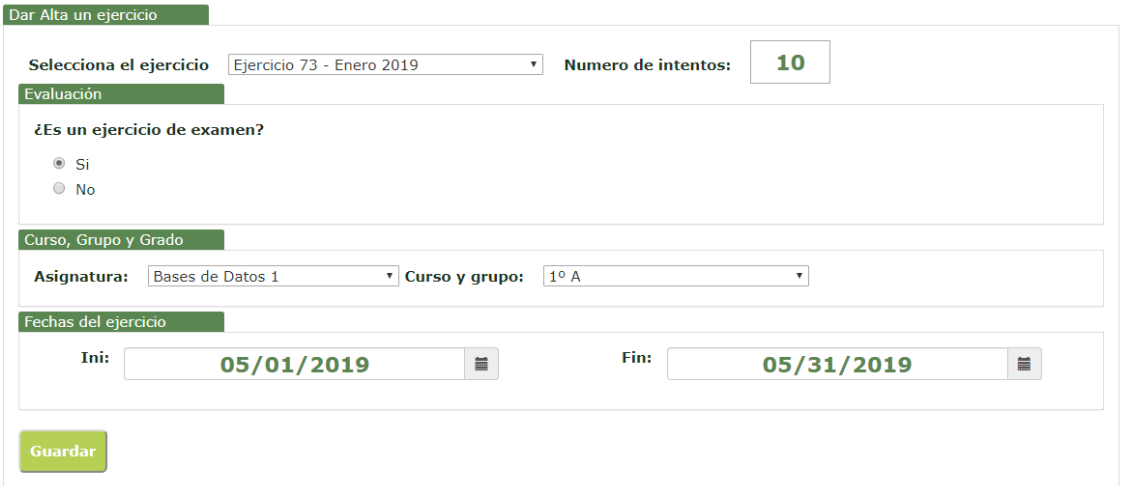
Añadir ejercicio a un grupo

Este formulario permite asignar los problemas que encontrará el alumno en la aplicación, ejercicios y exámenes.

El profesor tiene que seleccionar el ejercicio que quiere asignar a un grupo, poner el número de intentos máximo que tiene el alumno para resolverlo, en caso de que sea un ejercicio se supone que tendrá un alto número de intentos, aunque esto queda a criterio del profesor; mientras que los exámenes tendrán intentos limitados.

Además, se pide que seleccione una asignatura y un curso y grupo. Solo aparecen las asignaturas son en las que un profesor está asignado.

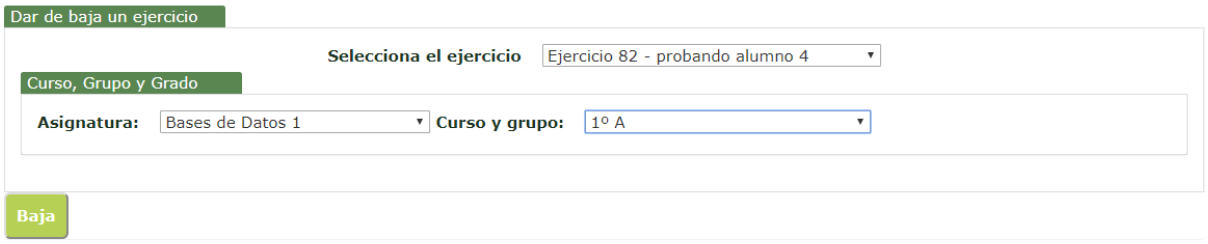
La fecha de inicio marca a partir de qué fecha un alumno podrá empezar a ver un ejercicio en su listado de ejercicios o exámenes y la fecha de fin marca hasta cuando un alumno puede subir ejercicios a la base de datos para que se puedan evaluar, pasada esa fecha se podrán ejecutar en Oracle y comprobar si es correcto o no, pero no se actualizará la base de datos.



Quitar ejercicio de un grupo

En el selector del ejercicio aparecerá una lista con todos los ejercicios que se han dado de alta, el profesor escogerá uno de ellos, después se cargarán las asignaturas en la que este ejercicio se ha dado de alta, al elegir la asignatura se podrá elegir el curso y grupo donde este ejercicio se ha dado de alta, finalmente para ejecutar esta operación se le da al botón de Baja.

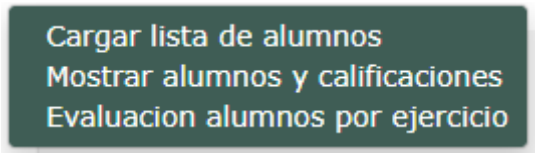
Puede pasar que este ejercicio lo hayan intentado alumnos de ese grupo, si es así aparecerá un mensaje de confirmación al profesor indicando que se va a borrar todo lo asociado con este ejercicio, al igual que los intentos de esos alumnos.



Eliminar un ejercicio

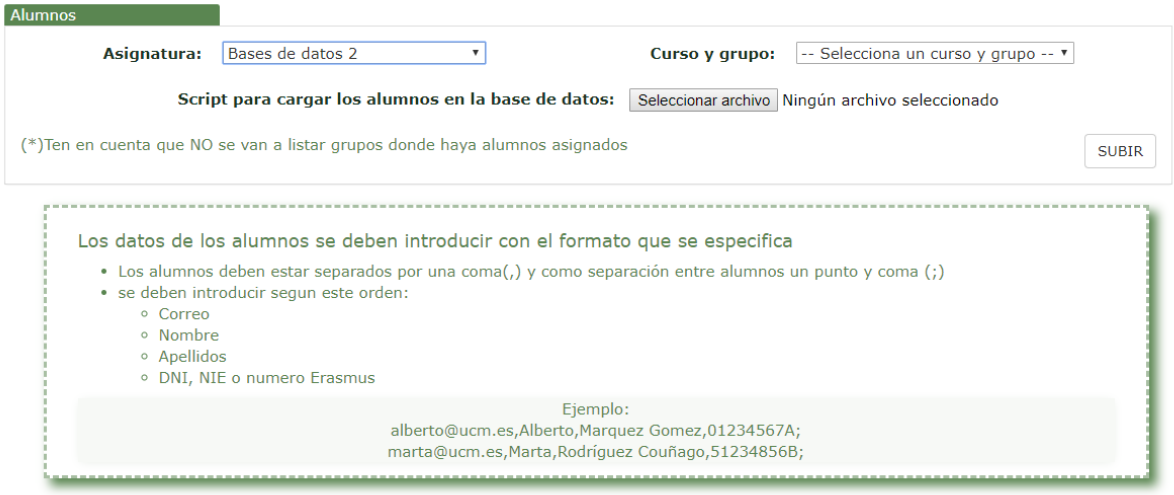
En este caso se puede seleccionar un ejercicio de la lista de los no para eliminarlo de la base de datos.
Por motivos de seguridad y consistencia de la aplicación, no será posible eliminar un ejercicio que esté dado de alta en un grupo y ya haya empezado a ser resuelto por los alumnos. Solo será posible eliminarlo dándolo de baja de todos los grupos en los que esté asignado desde la opción de menú de modificar un ejercicio y finalmente borrarlo.

Menú Alumnos



Cargar lista de alumnos

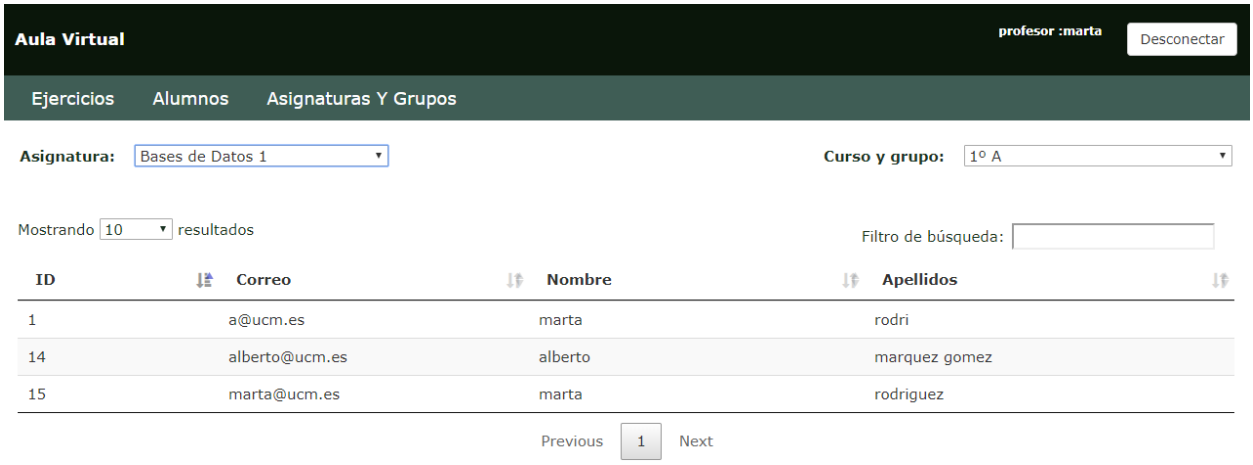
Para cargar una lista de alumnos hay que seleccionar la asignatura y el curso y grupo al que pertenecen esos alumnos, después se carga en el selector un fichero con el formato especificado y finalmente se pulsa el botón “SUBIR” para crear una lista de alumnos nuevos que puedan acceder a la aplicación.
Estos alumnos se crean en la base de datos con una contraseña por defecto, en este caso es su DNI o en los casos de estudiantes erasmus, el número que les identifique. La aplicación les pedirá que cambien la contraseña la primera vez que inicie sesión por otra que ellos elijan.



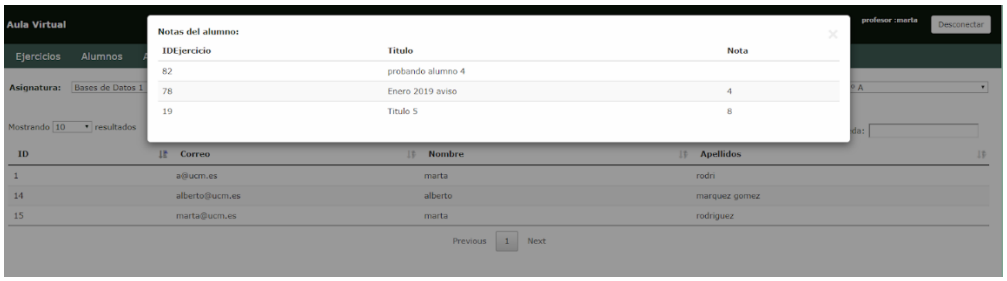
Mostrar alumnos y calificaciones

La lista de alumnos se muestra por asignatura curso y grupo ya que un profesor puede tener varias asignaturas cursos y grupos dados de alta en este mismo año.

Primero hay que seleccionar la asignatura seguido del curso y grupo.

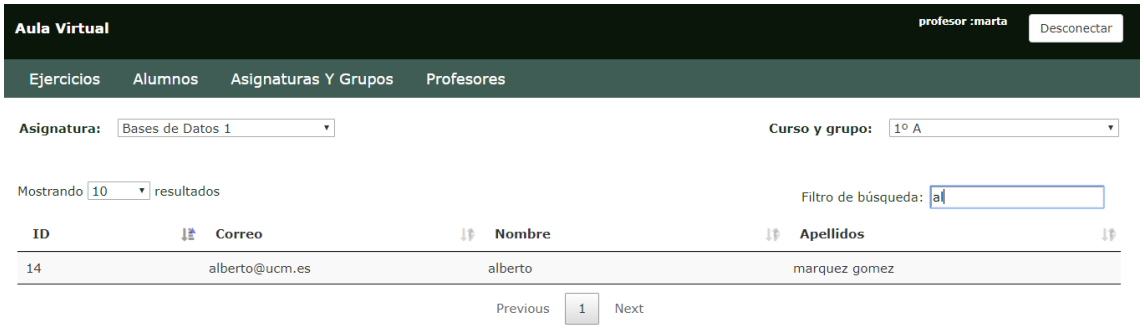


Aparecerá la lista de alumnos que dicho profesor ha dado de alta. Pulsando sobre cualquier alumno aparecerá una pantalla donde se mostrará un resumen de los ejercicios que ha intentado el alumno junto con su calificación, si no se le ha calificado aparecerá el blanco.



Esto es para que el profesor tenga más control sobre sus alumnos y pueda tener toda la información que necesite a mano sin tener que desplazarse por varias pantallas.

Además, con esta tabla el profesor podrá filtrar y ordenar por campo, ID, Correo, Nombre o Apellidos.



En siguiente figura se ha filtrado por el id del alumno.

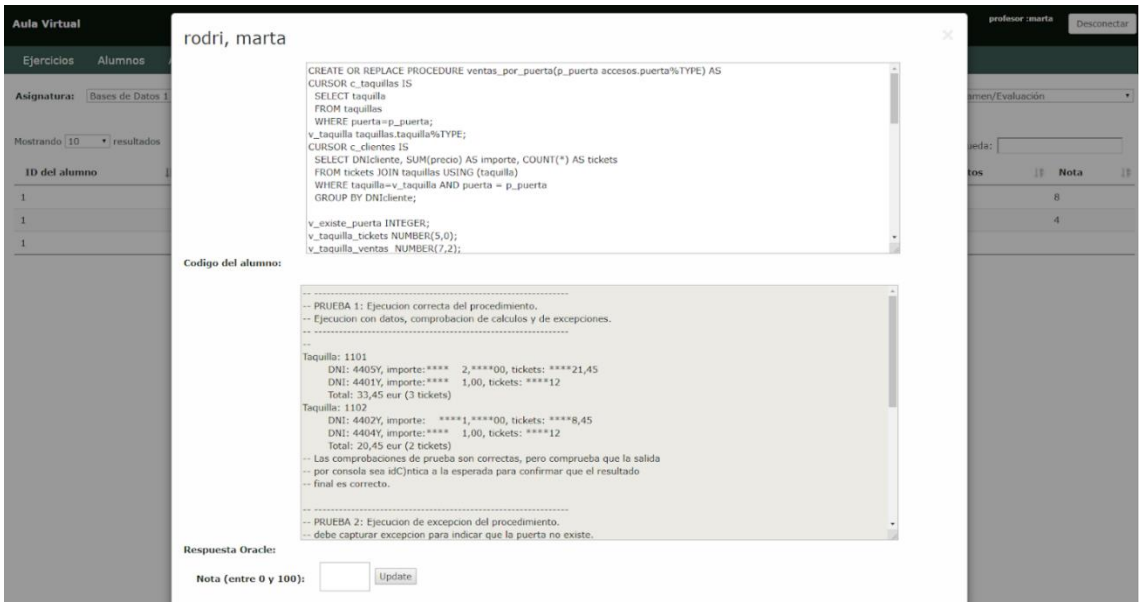
Evaluación alumnos por ejercicio

En esta opción de menú el profesor puede corregir los ejercicios entregados por los alumnos.

Primero debemos seleccionar asignaturas, curso y grupo.

Los cursos y grupos que se muestran están asignados a esa asignatura.
Después de escoger el curso y el grupo será necesario escoger entre exámenes o todos.

Se mostrará una lista con todos los alumnos asignados y al pinchar encima de uno de ellos se abrirá una ventana en la que se mostrará lo siguiente:



Primero encontramos el texto con la solución del alumno. Este se puede modificar para añadirle una corrección al alumno o un comentario. Todo lo que modifiquemos en este texto le saldrá subrayado al alumno en su solución.

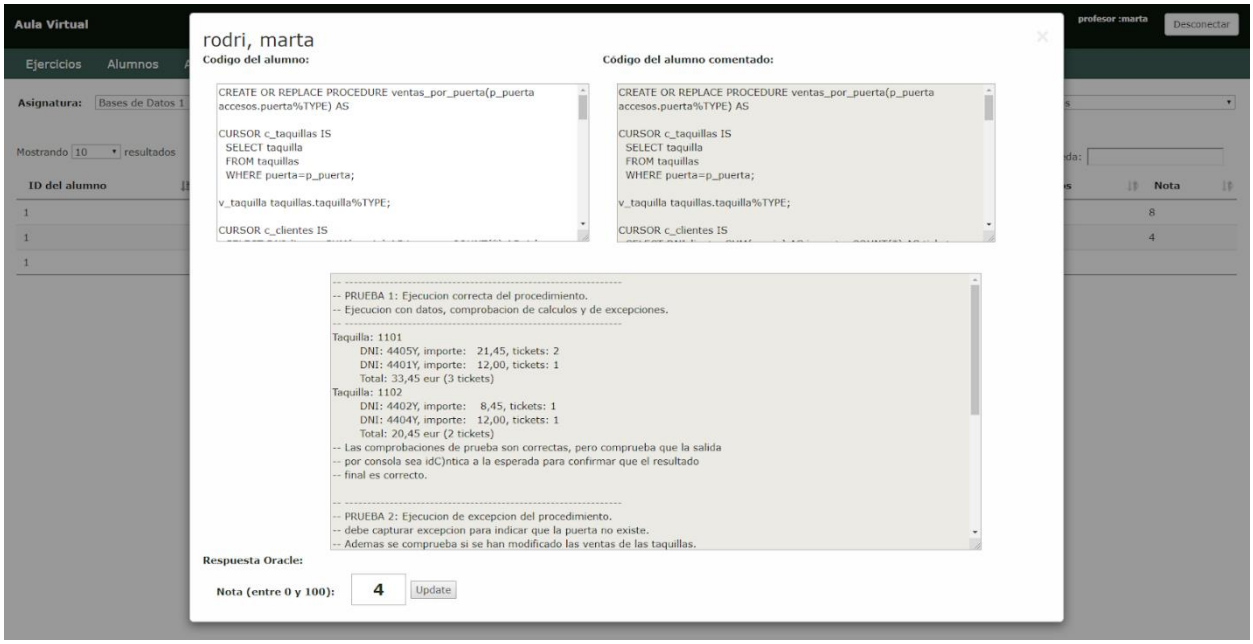
Debajo encontramos la salida que ha obtenido el alumno en las pruebas del script, se podrá ver si ha habido alguna diferencia entre la salida por pantalla del resultado del alumno y la del profesor.
Esta salida aparece con asteriscos si hubo alguna diferencia con el resultado que obtuvo el profesor al ejecutar la misma prueba.

Esto principalmente es para resaltar que los resultados no se están mostrando de la forma correcta, aunque realmente el ejercicio haya pasado la prueba.
Será entonces tarea del profesor el decidir si ese ejercicio es válido o no para esa prueba, aunque el resultado sea correcto.

Por último, encontramos un apartado para añadir una calificación al alumno este número estará comprendido entre 0 y 100.
La nota aparecerá actualizada en la tabla cuando se cierre la ventana emergente.

Adicionalmente un profesor podrá escribir un comentario o corregir lo que necesite del código que se le muestra del alumno. Cuando el alumno accede a este ejercicio se le mostrará el código modificado por el profesor, en el lado izquierdo de la pantalla.

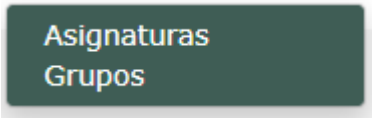
Si el profesor ha modificado el código del alumno en una corrección anterior, le aparecerá como se muestra en la siguiente figura.



En un primer texto a la izquierda el procedimiento que ha subido el alumno y en la ventana de texto de la derecha la anterior corrección del profesor si es que ha habido alguna. El resto de campos son los mismos que la primera vez que entra al ejercicio.

Menú de Asignaturas y Grupos

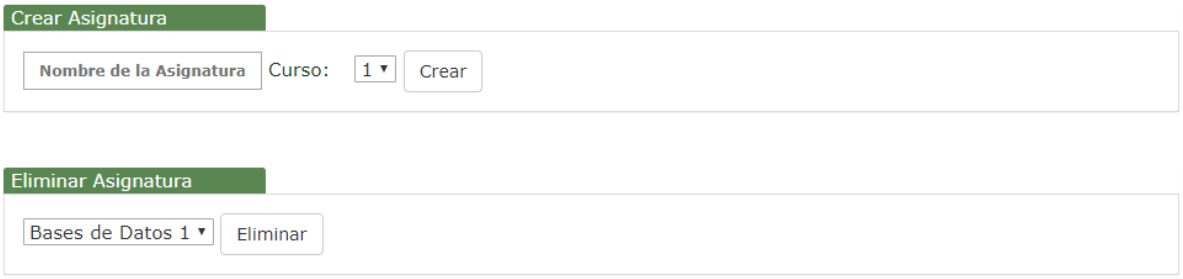
Podemos encontrar las opciones de las asignaturas en una página y la de grupos en otro, pudiendo hacer las operaciones de creación y borrado para ambas.



Asignaturas

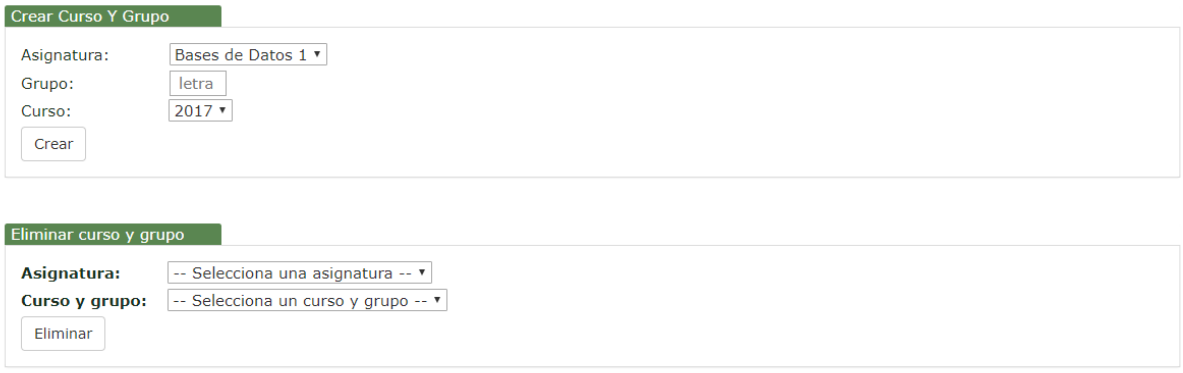
En la página de asignaturas será posible crear una nueva asignatura y eliminarla.

Para la creación de una asignatura no será necesario relacionarla con un grupo, simplemente se escribirá el nombre y esta quedará creada.
Una asignatura puede ser asignada a tantos grupos como sea necesario.
Para poder eliminar una asignatura esta tendrá que no tener ningún grupo asignado. Si tiene algún grupo asignado se mostrará un mensaje avisando de que se deben de eliminar los grupos asignados a esta asignatura antes de su eliminación.



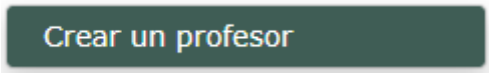
Grupos

En la página de grupos será posible crear un nuevo grupo y eliminarlo.
Si un grupo ha sido creado y ya tiene alumnos asignados se necesitará una doble confirmación para su eliminación, cuando pulse el botón de eliminar se le mostrará un mensaje de aviso para confirmar que realmente desea eliminar el grupo y a todos los alumnos asignados a ese grupo con sus ejercicios y exámenes.
Si pulsa el botón aceptar no habrá posible vuelta atrás, todos los usuarios serán eliminados, junto con su información generada. Y todos estos datos dejarán de estar accesibles tanto desde la página web como desde la base de datos.



Menú profesores

En el siguiente menú sirve para crear profesores.



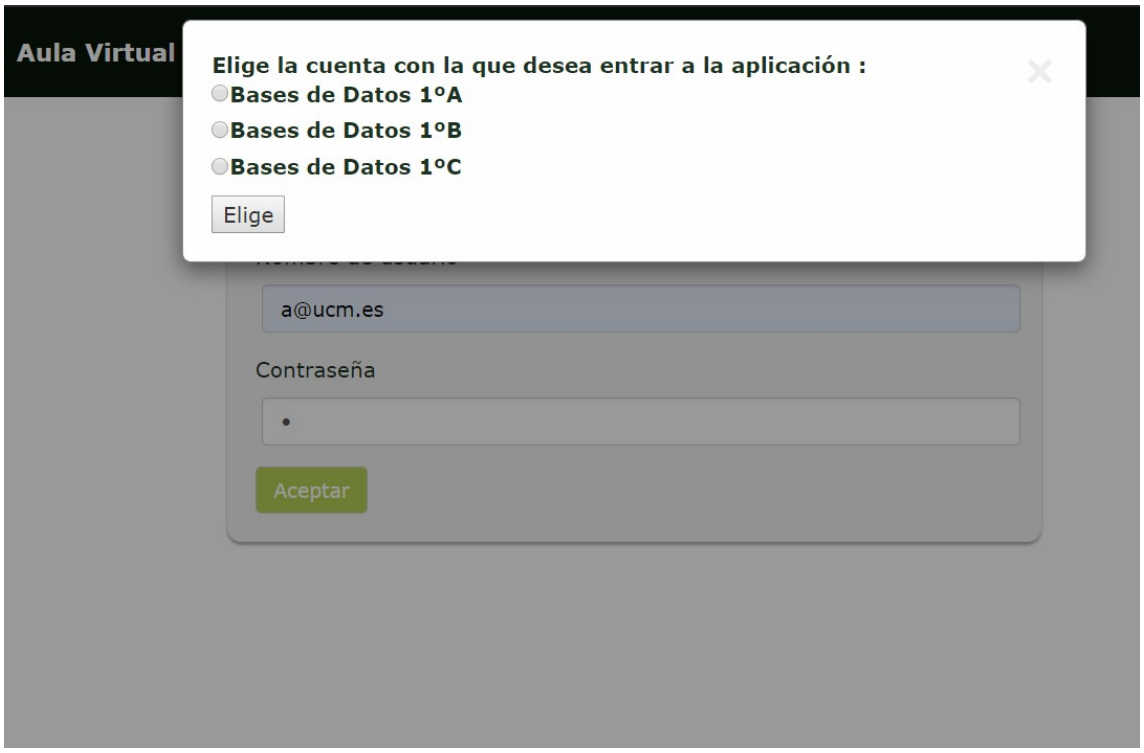
Crear un profesor

El siguiente formulario sirve para crea un usuario profesor para la base de datos.

Interfaz del alumno.

Antes de entrar en la aplicación en caso de que el usuario tenga mas de una cuenta dada de alta en la aplicación se le pedirá escoger una de ellas.

En la siguiente imagen se muestra la lista de asignaturas en las que el usuario esta dado de alta para que entre a la aplicación con una de ellas.



Al seleccionar una de estas cuentas o entrar directamente si solo tiene una cuenta en la aplicación, encontramos 4 menús en la barra horizontal bien diferenciados.

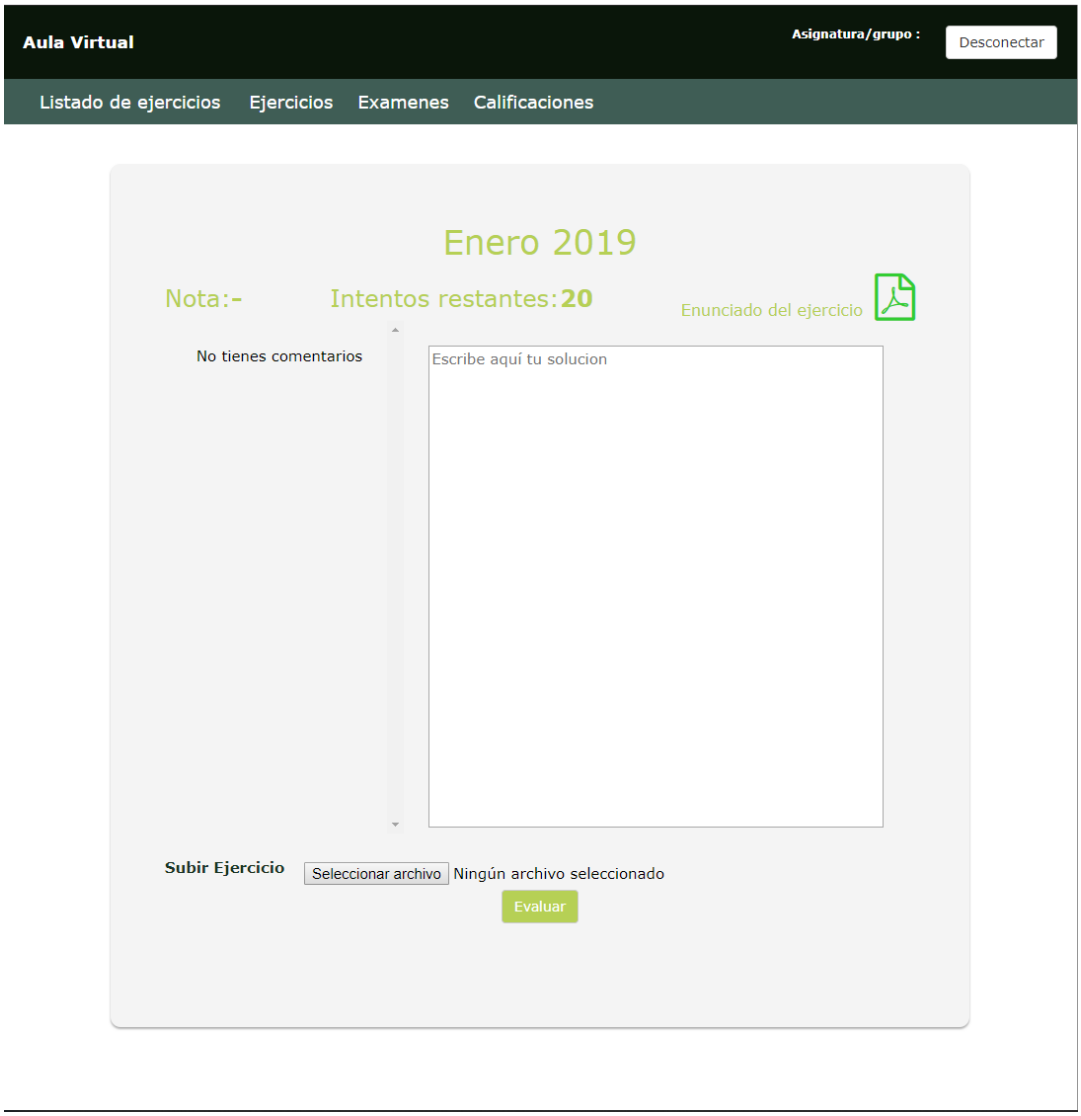
Listado de ejercicios

La idea principal de esta pantalla es que los alumnos que quieran realizar más ejercicios de los propuestos por el profesor tengan la oportunidad de hacerlo gracias a los ejercicios de otros años.

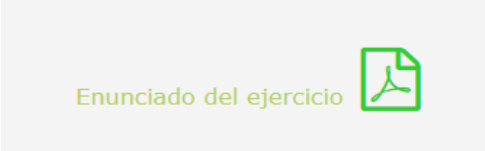
En el apartado de listado de ejercicios se mostrarán los ejercicios de años anteriores para que el alumno pueda realizarlos y obtener una corrección de ellos.
Es necesario escoger asignatura y año, después al pinchar sobre el ejercicio será redirigido al formulario para consultar el enunciado del ejercicio y escribir su solución.
Estos ejercicios no le llegarán al profesor y no podrá ver los resultados de los alumnos.

Formulario para la subida de un ejercicio

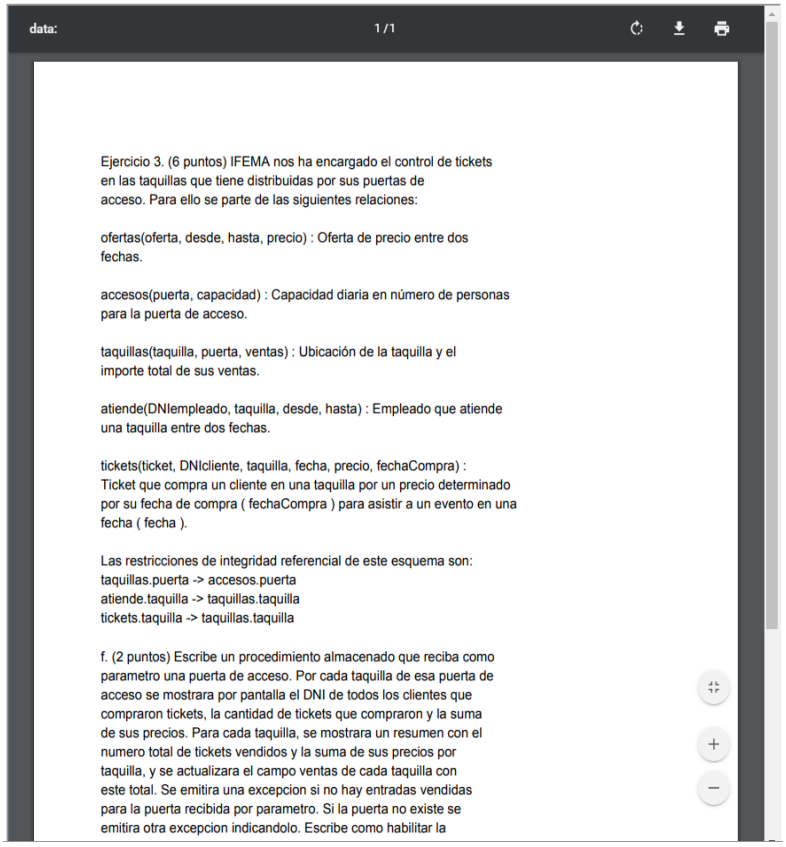
El aspecto del formulario cuando todavía no ha sido rellenado y mandado a corregir será el siguiente:



Esta página es la encargada de mostrar la nota del ejercicio arriba a la izquierda, los intentos restantes en el medio y el icono del pdf que encontramos arriba a la derecha que es el encargado de abrir en una página nueva el enunciado del ejercicio.

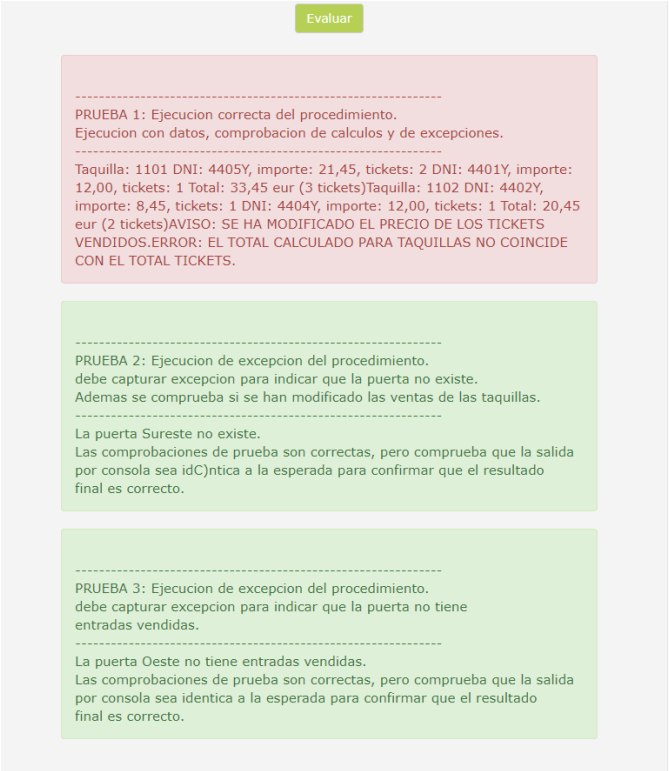


Tras pulsar sobre el icono del PDF este nos llevara al enunciado del ejercicio. Que se mostrara como en la siguiente ventana:



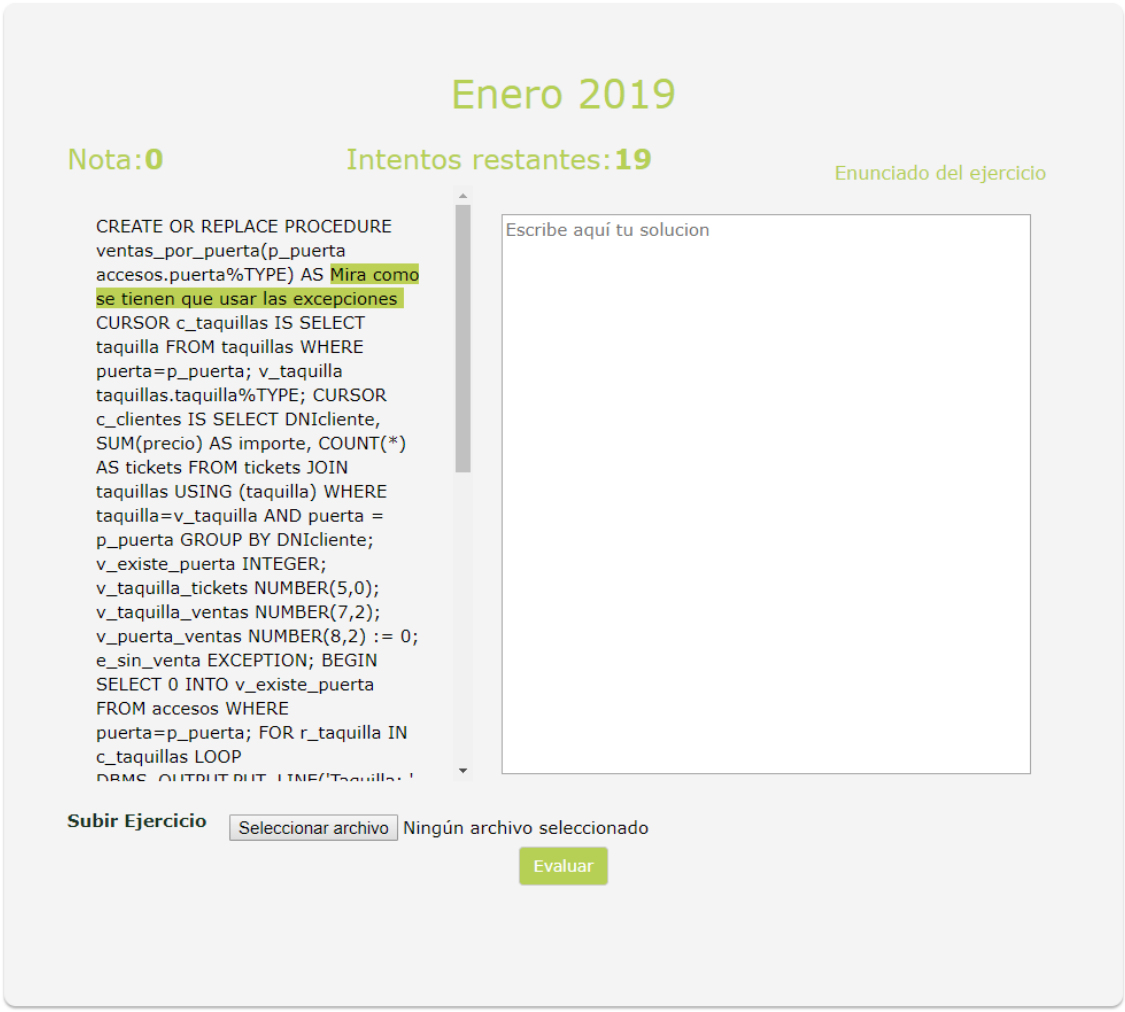
En el medio encontramos el título del ejercicio y más abajo, dos paneles, el texto de la izquierda será para mostrar los comentarios que ha hecho el profesor sobre el ejercicio ejecutado del alumno, y el de la derecha para escribir el procedimiento que se quiera evaluar. También será posible subir un archivo con el código del procedimiento sin tener que escribirlo.

Los resultados de las pruebas se mostrarán cuando se pulse evaluar, se ejecutará el procedimiento del alumno con las pruebas diseñadas por el profesor, en cuadros de colores.
Si el error que se muestra es de color gris será un fallo de compilación por parte de Oracle, si el color es rojo el fallo vendrá de uno de los scripts de corrección del profesor, donde pondrá el motivo del fallo y el número de la prueba en la que el procedimiento fallo.
Si en el recuadro se muestra en amarillo la prueba se mostrará como un aviso de algo que podría estar mejor hecho o con lo que hay que tener cuidado, no se contara como un fallo, y principalmente tendrá un uso informativo para que el alumno aprenda como no se tienen que hacer los procedimientos y cuál es la forma correcta, aunque el resultado haya sido el mismo.
El color verde indica que el ejercicio ha pasado esa prueba y es correcto.



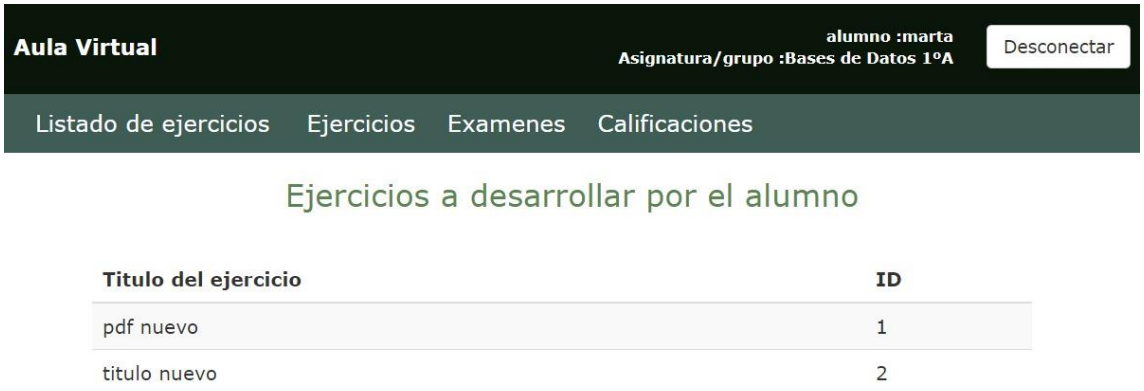
Los ejercicios y exámenes tendrán un número limitado de intentos por lo que será posible repetirlos tantas veces como se quiera hasta agotar los intentos o tener todas las pruebas en verde, sin embargo, solo se guardará en la base de datos el último intento realizado.

En la siguiente imagen se muestra un ejemplo de cómo se le mostrará el comentario al alumno sobre su código:



Ejercicios

En este segundo apartado se le mostrarán todos los ejercicios del curso actual que ha subido el profesor para que el alumno los codifique, solo se le mostrará la pantalla para resolver el ejercicio cuando el alumno pinche en el ejercicio y le redirija a la pantalla para ejecutar el procedimiento.



Exámenes

El tercer apartado muestra los exámenes, estos se diferencian con los ejercicios en que estos serán evaluados por el profesor, además tendrán un número de intentos máximo, este número de intentos será definido por el profesor y tras llegar al límite, no se podrá probar más este ejercicio.

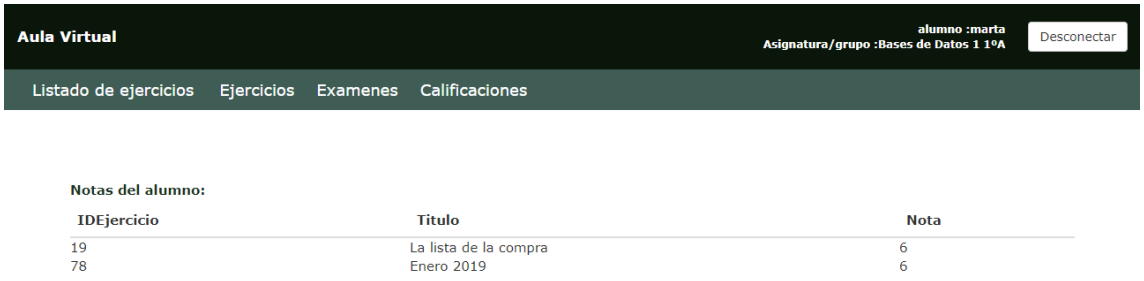
Es importante destacar que solo se guardará el último intento tanto en el caso de ejercicios como de exámenes. Esto quiere decir que si no ha llegado a terminar el número de intentos o su último intento es fallido, aunque haya tenido un intento anterior en el que todas las pruebas habían sido correctas, se tendrá en cuenta este último intento fallido y no el correcto.

El botón de enviar será el encargado de enviar a corregir el procedimiento, en caso de no haber subido ningún ejercicio o haber dado un problema en Oracle se mostrará un error y no se sumará al número de intentos.

Al seleccionar en un ejercicio de examen para ejecutarlo, todas las pantallas redirigirán a la vista de ejecutar un procedimiento explicado en el apartado de Listado de ejercicios.

Calificaciones

Por último, encontramos un menú de calificaciones, en el que se le mostrará una tabla con el id del ejercicio, el nombre del ejercicio y la nota que ha obtenido en este examen. El alumno podrá comprobar cuáles han sido las notas de todos los exámenes que ha realizado en el curso para que no tenga que entrar en cada examen ejecutado para comprobar su nota.



12. Conclusiones y trabajo futuro

Conclusiones

Nuestro principal objetivo era realizar una aplicación web que realmente llegase a ser útil en la universidad y solventara o facilitara alguno de los problemas que tenían los alumnos a la hora de aprender o de estudiar una determinada tecnología. Esta aplicación no solo ha hecho posible una mayor facilidad a la hora de estudiar una asignatura, sino que también ha facilitado la labor de los profesores con los alumnos cuando tienen que corregir y evaluar los procedimientos, teniendo a su disposición todos los procedimientos de sus ejercicios resueltos, a los cuales podrán añadir comentarios para que el alumno pueda entender el problema y escribir una mejor solución de este. Con una interfaz hecha a medida para sus necesidades.

La aplicación ha permitido a los alumnos tener una herramienta fácil de utilizar, sin instalaciones previas y que resuelve la dificultad de aprender un nuevo lenguaje de programación como es PL/SQL.

Gracias a las correcciones automáticas de sus procedimientos, podrán practicar este lenguaje de forma autónoma, y además pude tener los comentarios del profesor como guía para llegar a la solución del ejercicio.

Trabajo futuro

Después de haber estado desarrollando durante 9 meses la aplicación, nos parece que está bastante completa. Sin embargo, una gran ampliación que no se incluyó en el plan de proyecto fue añadir la posibilidad de corregir triggers hechos por el alumno. Si hubiésemos tenido más tiempo nos habría gustado hacerlo, pero desplegar toda la funcionalidad y hacer que funcionase correctamente ya ocupó todo el tiempo que teníamos disponible para desarrollar la aplicación. Sin embargo, esta ampliación junto con alguna que otra mejora descrita a continuación podrían formar parte de un nuevo TFG que completara el nuestro y sirviese de un mantenimiento de este.

Otra ampliación es la comunicación directa entre alumno y profesor, un chat que por falta de tiempo no se pudo hacer.

Además, como todos los jueces se podría añadir un timeout para la eficiencia de las consultas, aunque Oracle ya ofrece uno para que ninguna consulta entre en un bucle infinito, se podría añadir la funcionalidad de poder cambiar este tiempo según el ejercicio definido por cada profesor.

Otra de las mejoras es tener una aplicación estable y segura usando el protocolo https y comprobando que la vulnerabilidad Cross Site Scripting no se produce junto con algunas otras propias de seguridad en la web.

13. Conclusions and Future work

Conclusions

Our main objective was to make a web application that really became useful in the university and solve or facilitate some of the problems that students had when learning or studying a certain technology. This application has not only made it easier to study a subject, but has also facilitated the work of teachers with students when they have to correct and evaluate the procedures, having at their disposal all the procedures of their exercises solved, to which may add comments so that the student can understand the problem and write a better solution of this. With an interface tailored to your needs.

The application has allowed students to have an easy-to-use tool, without previous facilities and that solves the difficulty of learning a new programming language such as PL / SQL.

Thanks to the automatic corrections of their procedures, they will be able to practice this language autonomously, and I could also have the teacher's comments as a guide to arrive at the solution of the exercise.

Future Work

After 9 months developing this application we think that it has a lot of functionality, however an extension that was not included was to implement the possibility of evaluate triggers in Oracle database. Moreover, other extension that we contemplate from the beginning is direct communication between teacher and student through a chat regardless of the exercise.

Additionally, as all online judge it could be possible to implement a timeout, although Oracle gives one, the possibility is set this timeout depending on what teachers want.

14. Reparto del trabajo

Al comenzar el trabajo los dos coincidimos en que haríamos el esfuerzo de intentar tocar todas las funcionalidades de la aplicación y aunque al principio fue muy costoso debido a la falta de un buen control de versiones, conseguimos implementar más o menos los dos toda la funcionalidad de la aplicación.

Las tareas se han repartido acorde a la carga de trabajo, las más pesadas están distribuidas para que no se encargue uno de nosotros de todas y el resto las hemos puesto en una lista y según hemos ido terminando unas hemos empezado las siguientes

Marta Rodríguez Couñago

El proyecto en un principio estaba pensado para 3 personas, extendiendo su funcionalidad a triggers y disparadores, sin embargo, una de nuestras compañeras decidió abandonarlo, eso nos supuso tener que volver a pensar cómo teníamos que hacerlo y recortar algunas partes del proyecto.

Eso no nos supuso ningún contratiempo a nivel de proyecto ya que ocurrió antes de empezar el curso, pero eso llevó a que tenía que trabajar con alguien que no conocía, sin embargo, con el paso del tiempo hemos llegado a ser más que compañeros, amigos, formando un gran equipo.

Al principio del proyecto mi compañero no conocía casi ninguna herramienta de programación web y yo solo conocía la que se da en la asignatura de Aplicaciones Web en la carrera de Ingeniería del Software, le expliqué que con la herramienta y el lenguaje que conocía se podía hacer muchas cosas, además de que tenía bastante práctica gracias a la asignatura, además que en las prácticas en empresa estuve aprendiendo un poco de programación web. Con esto convencí a mi compañero de hacer el código en Node js usando JavaScript tanto en el cliente como en el servidor.
Mientras que mi compañero aprendía a programar en JavaScript, decidimos ir haciendo los HTML para cada pantalla del proyecto teniendo en cuenta la funcionalidad que el profesor nos había pedido.

Mientras que hacíamos los HTML de cada apartado que el profesor quería le expliqué a mi compañero cómo hacer algunas funcionalidades básicas como el login o logout de la aplicación.

Una vez teníamos las pantallas y alguna funcionalidad hecha, empezamos a realizar la base de datos.

Al principio el profesor nos pedía mucha funcionalidad, como la existencia de un chat para la comunicación entre alumno y profesor de manera directa, esto se implementó en la base de datos, sin embargo, lo tuvimos que descartar por falta de tiempo.
La base de datos no fue complicada de montar ya que el profesor había expuesto unos requisitos bien definidos y gracias a su disponibilidad y ayuda en entender algunos apartados no nos costó tener lista una primera versión.
Con esto, nos pusimos a hacer el login de la aplicación para que los alumnos como los profesores pudieran tener acceso a ella.

Una vez terminado el login, el siguiente paso fue diferenciar entre alumno y profesor en las cabeceras y en la cookie, que fue lo que utilizamos para guardar las credenciales del usuario, una vez terminado esto nos pusimos con la funcionalidad de crear un nuevo ejercicio, con esto necesitábamos ejemplos y datos reales para poderlo más tarde ejecutar, por lo que le pedimos al profesor que nos proporcionara algunos datos sencillos.

Mientras que el profesor buscaba estos ejemplos nosotros íbamos avanzando en funcionalidad; decidimos implementar un elemento "Drag and Drop" para subir ficheros al servidor, que tanto habíamos visto en páginas modernas. Pero nos resultó tremendamente complicado, por la falta de documentos y errores de los componentes utilizados, mientras mi compañero se puso a investigar cómo hacer la funcionalidad de este elemento, me puse a convertir todos los scripts de soluciones y creación de tablas a base64 para subirlas a la base de datos. Esta decisión se tomó así para aumentar la seguridad la base de datos y que no estuviera tan accesible, además resultaba más sencillo poder subir un enunciado en pdf a la base de datos si estaba en este formato.

Después de esto, hice la plantilla para la ejecución de procedimientos por parte del alumno. En esta plantilla añadí la funcionalidad de poder pulsar un botón y abrir el enunciado en el buscador directamente sin tener que descargar ningún documento extra en el directorio.

Una vez terminé esta funcionalidad mi compañero consiguió hacer la funcionalidad "Drag and Drop" sin embargo esto nos llevó más tiempo del previsto.
En los últimos 2 meses dividimos el trabajo diferenciando la parte que el profesor tenía acceso y la del alumno.

Mientras que mi compañero hacía la conexión con Oracle para la ejecución de ejercicios me puse con las funcionalidades del profesor, dar de alta un ejercicio, dar de baja. Sin embargo, nos íbamos intercambiando un poco el trabajo para ayudarnos y que no fuera tan pesado para ninguno. Así mientras que mi compañero hizo crear y eliminar asignatura y grupos yo estuve ayudando en la ejecución de procedimientos en Oracle.

La conexión con Oracle fue bastante complicada ya que usamos una librería que proporcionaba el mismo Oracle para realizar consultas con JavaScript, sin embargo, en este punto no éramos capaces de hacerlo como nosotros queríamos por lo que tuvimos que abrir un hilo en la cuenta de GitHub donde sacamos el código para la conexión.
En las últimas semanas los dos, tanto mi compañero como yo hicimos funcionalidades tanto en la interfaz del alumno como en la del profesor.

Alberto Márquez Gómez

Antes de contar mi colaboración en la aplicación me gustaría poner en contexto cómo llegamos a desarrollarla en este lenguaje y por todos los problemas por los que hemos ido pasando.

Empezamos las reuniones del proyecto antes de empezar el curso, en principio el proyecto estaría formado por 3 personas contando conmigo, de las cuales yo ya conocía a una de ellas. Lamentablemente esta persona tuvo que dejar el proyecto, con lo que terminé haciendo el proyecto con una persona que no conocía. Pero en la que encontraría más tarde no solo una compañera sino también una buena amiga.

Tras ver varios ejemplos de aplicaciones parecidas a la que tendríamos que desarrollar y saber en qué nos queríamos diferenciar de ellas.

Lo primero que decidimos fue el lenguaje en el que haríamos la aplicación, lo cual fue algo más duro para mí al principio. Decidimos utilizar JavaScript en toda la aplicación, del cual yo casi no había visto nada en la carrera. Ella sí había hecho algunas prácticas con este lenguaje y conocía un poco algunas librerías utilizadas.
Tanto ella como yo pensábamos que sería más sencillo y desconocíamos la magnitud que tendría finalmente el proyecto.

Mis conocimientos de JavaScript pasaban por la poca teoría que había visto en clase y algún ejemplo, pensaba que solo se utilizaba en la parte del cliente y desconocía la existencia de node.js.

Lo primero que tuve que hacer fue familiarizarme con el lenguaje, gracias a mi compañera que me ayudó en todo momento y me dejó sus apuntes para que aprendiese todo lo que ella sabía en un par de meses ya sabía lo básico para poder conectarme con la base de datos de MySQL y mostrar los datos de las consultas en la aplicación del cliente.

Empecé encargándome de que los ejercicios del alumno que se muestran en la página principal se cargan dinámicamente, para lo cual tuve que estudiar jQuery y ver cómo se podrían añadir y eliminar elementos al DOM.

Lo complicado vino después al intentar darle más estilo a la aplicación y querer usar librerías determinadas. De lo cual aprendimos una gran lección y es que no siempre puedes conectar sencillamente una librería con tu aplicación y esperar que esta funcione perfectamente. A veces es mejor escoger otras aplicaciones que, aunque no tengan el estilo que uno desea originalmente, te acaban dando la misma funcionalidad de una forma muy parecida y te permiten ahorrar mucho tiempo.

La única librería que encontramos fiable para realizar las conexiones con la base de datos de Oracle en la parte del servidor fue oracledb, pues están disponibles el código y ejemplos en su GitHub.
Hemos visto que para realizar consultas a la base de datos es igual de sencillo que las consultas que realizamos en MySQL. El problema llega a la hora de almacenar procedimientos, sobre todo si se quieren hacer más cosas a parte de almacenarlo ya que será necesario ejecutar cada bloque de código independiente del resto y respetando el orden de ejecución.
Esta parte ha sido una de las que más tiempo he dedicado a la hora de desarrollar el proyecto, ya que ninguno de los dos la habíamos probado antes y el código cambia respecto al JavaScript que estábamos escribiendo.

Aparece el término async, el cual nos indica que la llamada de la función será asíncrona, lo cual hace que las funciones se ejecuten en orden consecutivo, sin necesidad de callbacks.
Si había sido costoso aprender cómo funcionaban los callback, ahora tendría que aprender a hacer que el código que ya teníamos fuese compatible con las llamadas asíncronas a parte de la configuración que tenía que tener la propia base de datos para funcionar.
Para lo cual fue necesario estudiar los privilegios de los roles de los usuarios y ver como entrar en oracledb con un rol determinado.
No todas las pruebas que haces en sqldeveloper luego se pueden hacer en oracledb este tiene más limitaciones.

Aunque no permite hacer las mismas cosas que el sqldeveloper nos ha permitido crear todo lo que necesitábamos por partes, aunque ha sido necesario almacenar procedimientos para realizar algunas funciones que no podíamos realizar fácilmente con este. Cómo almacenar la salida por pantalla de los bloques anónimos que nos corrigen los procedimientos del alumno o la creación de usuarios.

Mi compañera también me ayudó con la compatibilidad de lo asíncrono con lo síncrono e incluso llegamos a poner un issue en el proyecto de GitHub de oracledb para ver si nos podían ayudar con un problema que estábamos teniendo. Es impresionante cómo la gente está dispuesta a ayudarte de una forma completamente altruista e intenta comprender qué es lo que quieres hacer con ese fragmento de código para poder ayudarte mejor. Sinceramente pensábamos que no nos iban a contestar o no iba a servir para nada, pero sí que consiguieron hacernos ver por donde teníamos que ir más o menos y lo que podría fallar.

Los últimos desarrollos en las interfaces y funcionalidades de la aplicación los hicimos los dos, distribuyéndonos el trabajo tanto en la parte del alumno como del profesor y la memoria también fue distribuida de la misma forma.

15. Repositorio en GitHub

Todo el código de nuestro proyecto se puede encontrar en GitHub actualizado y estable en la siguiente dirección [26].
Para hacer una instalación de la aplicación sería necesario descargar el siguiente software:

- Descargar el código de GitHub
- Descargar el software de Visual Studio Code [27]
- Descargar una versión de Oracle para Windows
- Descargar XAMPP

Los requisitos para hacer una instalación del software de forma local son los siguientes:

- En la carpeta prueba, situada en la raíz del proyecto en GitHub encontramos los procedimientos almacenados que usaremos para la creación de usuarios y tratar los resultados de la corrección de los procedimientos de los alumnos. Estos se deben almacenar en el usuario SYS de la base de datos Oracle.
- Será necesario abrir en Visual Code Studio Code la carpeta con todo el código que nos hemos descargado.
- Lanzaremos el siguiente comando para la ejecución de código desde el terminal de Visual Estudio Code: node app.js
- Sera necesario haber cargado la base de datos aulavirtual.sql que encontramos en la raíz de la carpeta prueba en una nueva base de datos llamada aulavitual en XAMPP.
- Después hay ejecutar en XAMPP: Apache y MySQL , dándole al botón de aceptar en el panel de control.
- Tras seguir estos pasos al poner la siguiente dirección en el navegador se mostrará la página de inicio de la aplicación: http://localhost:3000/

En el archivo REEDME de GitHub se encuentra una instalación más detallada de la aplicación.

16. Bibliografía

[1] Node JS:
- https://nodejs.org/es/

[2] MySQL instalando mysql:
- https://github.com/mysqljs/mysql

[3] Base de datos de Oracle mediante oracledb:
- https://github.com/oracle/node-oracledb

[4] Documentación Oracledb:
- https://www.oracle.com/technetwork/es/articles/database-performance/restful-node-3813084-esa.html

[5] Módulo fs utilizado para recoger la salida por consola de Oracle:
- https://nodejs.org/api/fs.html

[6] Definición de DOM :
- https://es.wikipedia.org/wiki/Document_Object_Model

[7] Biblioteca de librerías para JavaScript:
- https://www.npmjs.com/

[8] Tabla de Bootrstrap para mostrar los resultados:
- https://mdbootstrap.com/docs/jquery/tables/sort/

[9] Modulo Diff para la comparación de textos:
- https://www.npmjs.com/package/diff

[10] Para arrastrar los documentos utilizamos una funcionalidad de bootstrap llamada fileinput:
- https://cdnjs.com/libraries/bootstrap-fileinput

[11] Versión de JQuery:
- https://code.jquery.com/jquery-3.3.1.min.js

[12] definición DAO:
- https://es.wikipedia.org/wiki/Objeto_de_acceso_a_datos

[13] definición caso de uso:
- https://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso

[14] Modelo Entidad-Relación
- https://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n

[15] Bootstrap:
- https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.js

[16] Bootstrap Datepicker:
- https://bootstrap-datepicker.readthedocs.io/en/latest/

[17] Plantillas EJS:
- https://ejs.co/

[18] Código para crear una cookie en el lado del cliente:
- https://www.w3schools.com/js/js_cookies.asp

[19] Como hacer plugin de JQuery:
- https://learn.jquery.com/plugins/basic-plugin-creation/

[20] Jutge:
- https://jutge.org

[21] Juez de la Universidad de Valladolid:
- https://uva.onlinejudge.org

[22] Juez de la Facultad de Informática:

- http://eda.fdi.ucm.es/domjudge/public/

[23] Documentación de domJudge:
- https://www.domjudge.org/docs/team-manual.pdf

[24] Juez Acepta el reto:
- https://www.aceptaelreto.com/

[25] FLOP:
- http://problem-g.estad.ucm.es/FLOP/index.jsp

[26] Repositorio de código:
- https://github.com/albertoMarquez/AulaVirtual.git

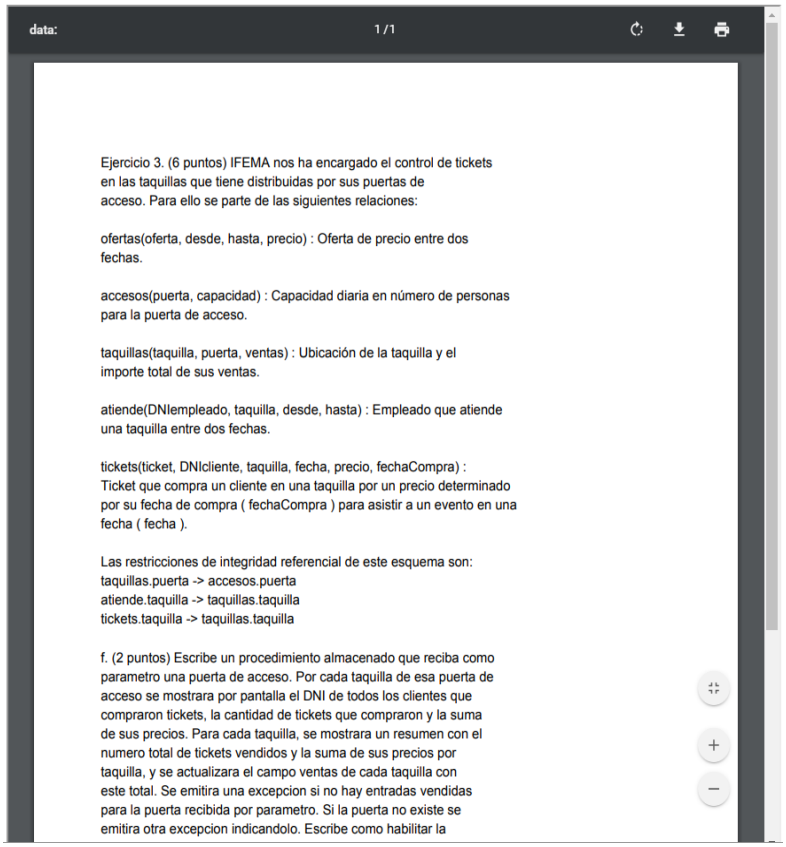
[27] Visual Studio Code:
- https://code.visualstudio.com/

Apéndice

Ejemplo de scripts de un ejercicio de PL/SQL:

Enunciado

El archivo del enunciado debe ser un PDF, los enunciados de los ejercicios se mostrarán de la siguiente forma para el alumno.



Script de creación y carga de datos

Para hacer correctamente el script y que el programa lo entienda, se empiezan creando las tablas en el orden correcto según las dependencias y continuación se cargan los datos.

```
CREATE TABLE ofertas (
 oferta INTEGER PRIMARY KEY,
 desde DATE,
 hasta DATE,
 precio NUMBER(11,2));

CREATE TABLE accesos (
 puerta VARCHAR2(10) PRIMARY KEY,
 capacidad INTEGER);

CREATE TABLE taquillas (
 taquilla INTEGER PRIMARY KEY,
 puerta REFERENCES accesos,
 ventas NUMBER(11,2));

CREATE TABLE atiende (
 DNIempleado VARCHAR2(10),
 taquilla REFERENCES taquillas,
 desde DATE,
 hasta DATE,
 PRIMARY KEY (DNIempleado, taquilla, desde));

CREATE TABLE tickets (
 ticket INTEGER PRIMARY KEY,
 DNIcliente VARCHAR2(10),
 taquilla REFERENCES taquillas,
 precio NUMBER(11,2),
 fechaCompra DATE
);

INSERT INTO ofertas VALUES (1, '01/12/2018','08/12/2018',11.32);
INSERT INTO ofertas VALUES (2, '25/12/2018','31/12/2018',7.50);

INSERT INTO accesos VALUES ('Oeste', 1);
INSERT INTO accesos VALUES ('Este', 2);
INSERT INTO accesos VALUES ('Sur', 3);
INSERT INTO accesos VALUES ('Norte', 4);

INSERT INTO taquillas VALUES (1101, 'Este', 0.00);
INSERT INTO taquillas VALUES (1102, 'Este', 0.00);
INSERT INTO taquillas VALUES (1201, 'Sur', 0.00);
INSERT INTO taquillas VALUES (1202, 'Sur', 0.00);
INSERT INTO taquillas VALUES (1301, 'Norte', 0.00);

INSERT INTO atiende VALUES ('3701X', 1101, '18/12/2018',
'20/12/2018');
INSERT INTO atiende VALUES ('3701X', 1201, '21/12/2018',
'24/12/2018');
INSERT INTO atiende VALUES ('3702X', 1101, '21/12/2018',
'26/12/2018');
INSERT INTO atiende VALUES ('3702X', 1301, '27/12/2018',
'30/12/2018');
INSERT INTO atiende VALUES ('3703X', 1201, '18/12/2018',
'20/12/2018');
INSERT INTO atiende VALUES ('3703X', 1301, '21/12/2018',
'30/12/2018');
```

```
INSERT INTO tickets VALUES (21, '4401Y', 1101, 12.00,
'20/12/2018');
INSERT INTO tickets VALUES (22, '4402Y', 1102, 8.45,
'21/12/2018');
INSERT INTO tickets VALUES (23, '4405Y', 1101, 9.45,
'22/12/2018');
INSERT INTO tickets VALUES (24, '4404Y', 1202, 6.25,
'23/12/2018');
INSERT INTO tickets VALUES (25, '4405Y', 1301, 12.00,
'24/12/2018');
INSERT INTO tickets VALUES (26, '4405Y', 1101, 12.00,
'25/12/2018');
INSERT INTO tickets VALUES (27, '4404Y', 1102, 12.00,
'26/12/2018');
INSERT INTO tickets VALUES (28, '4406Y', 1201, 12.00,
'27/12/2018');
INSERT INTO tickets VALUES (29, '4407Y', 1301, 12.00,
'28/12/2018');

COMMIT;
```

Script con la solución del profesor

El profesor sube un script con solución correcta del ejercicio para comparar la solución del script del alumno con la suya y ver si el orden o los datos que ha tenido son los mismos. El script debería de tener el siguiente formato.

```
CREATE OR REPLACE PROCEDURE ventas_por_puerta(p_puerta
accesos.puerta%TYPE) AS

CURSOR c_taquillas IS
 SELECT taquilla
 FROM taquillas
 WHERE puerta=p_puerta;

v_taquilla taquillas.taquilla%TYPE;

CURSOR c_clientes IS
 SELECT DNIcliente, SUM(precio) AS importe, COUNT(*) AS tickets
 FROM tickets JOIN taquillas USING (taquilla)
 WHERE taquilla=v_taquilla AND puerta = p_puerta
 GROUP BY DNIcliente;
v_existe_puerta INTEGER;
v_taquilla_tickets NUMBER(5,0);
v_taquilla_ventas  NUMBER(7,2);
v_puerta_ventas NUMBER(8,2) := 0;
e_sin_venta EXCEPTION;
BEGIN
 SELECT 0 INTO v_existe_puerta FROM accesos WHERE puerta=p_puerta;
 FOR r_taquilla IN c_taquillas LOOP
```

```
   DBMS_OUTPUT.PUT_LINE('Taquilla: ' || r_taquilla.taquilla);
   v_taquilla_tickets := 0;
   v_taquilla_ventas := 0.0;
   v_taquilla := r_taquilla.taquilla;
    FOR r_clientes IN c_clientes LOOP
     DBMS_OUTPUT.PUT_LINE('  DNI: ' || r_clientes.DNIcliente ||
', importe: ' ||
          TO_CHAR(r_clientes.importe,'999D99') || ', tickets: '
|| r_clientes.tickets);
      v_taquilla_ventas      :=      v_taquilla_ventas      +
r_clientes.importe;
      v_taquilla_tickets     :=     v_taquilla_tickets     +
r_clientes.tickets;
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('  Total: ' || v_taquilla_ventas || ' eur
(' || v_taquilla_tickets || ' tickets)');
   v_puerta_ventas := v_puerta_ventas + v_taquilla_ventas;

   UPDATE   taquillas   SET   ventas=v_taquilla_ventas   WHERE
taquilla=r_taquilla.taquilla;
 END LOOP;
 IF v_puerta_ventas=0 THEN RAISE e_sin_venta; END IF;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
   DBMS_OUTPUT.PUT_LINE('La  puerta  '  ||  p_puerta  ||  '  no
existe.');
 WHEN e_sin_venta THEN
   DBMS_OUTPUT.PUT_LINE('La puerta ' || p_puerta || ' no tiene
entradas vendidas.');
END;
```

Scripts de pruebas del ejercicio

Estos tres ejemplos muestran como debe de ser el formato para evaluar el ejercicio del alumno:

Primero:

```
DECLARE
 v_acceso VARCHAR2(10) := 'Este';
 v_ventasTicketsAntes NUMBER(11,2) := 0;
 v_ventasTicketsDespues NUMBER(11,2) := 0;
 v_ventasTaquillas NUMBER(11,2) := 0;
 v_numAvisos INTEGER := 0;

 -- cursor para recorrer las taquillas de la puerta p_puerta.
 CURSOR c_taquillas IS
    SELECT taquilla
   FROM taquillas
   WHERE puerta=v_acceso;
BEGIN
 dbms_output.enable(100000);
```

```
 DBMS_OUTPUT.PUT_LINE('-- ----------------------------------------
----------------------');
 DBMS_OUTPUT.PUT_LINE('-- PRUEBA 1: Ejecucion correcta del
procedimiento.');
 DBMS_OUTPUT.PUT_LINE('-- Ejecucion con datos, comprobacion de
calculos y de excepciones.');
 DBMS_OUTPUT.PUT_LINE('-- ----------------------------------------
----------------------');
 DBMS_OUTPUT.PUT_LINE('-- ');
  UPDATE taquillas SET ventas = 0;
  SELECT SUM(precio) INTO v_ventasTicketsAntes
 FROM tickets JOIN taquillas USING (taquilla)
 WHERE puerta = v_acceso;

 ventas_por_puerta('Este');

 SELECT SUM(precio) INTO v_ventasTicketsDespues
 FROM tickets JOIN taquillas USING (taquilla)
 WHERE puerta = v_acceso;

 SELECT SUM(ventas) INTO v_ventasTaquillas
 FROM taquillas WHERE puerta = v_acceso;
  IF v_ventasTicketsAntes != v_ventasTicketsDespues THEN
   DBMS_OUTPUT.PUT_LINE('AVISO: SE HA MODIFICADO EL PRECIO DE LOS
TICKETS VENDIDOS.');
   v_numAvisos := v_numAvisos + 1;
 END IF;
 IF v_ventasTicketsAntes != v_ventasTaquillas THEN
   DBMS_OUTPUT.PUT_LINE('ERROR: EL TOTAL CALCULADO PARA TAQUILLAS
NO COINCIDE CON EL TOTAL TICKETS.');
   v_numAvisos := v_numAvisos + 1;
 END IF;
 IF v_numAvisos = 0 THEN
   DBMS_OUTPUT.PUT_LINE('-- Las comprobaciones de prueba son
correctas, pero comprueba que la salida ');
   DBMS_OUTPUT.PUT_LINE('-- por consola sea idéntica a la esperada
para confirmar que el resultado');
   DBMS_OUTPUT.PUT_LINE('-- final es correcto.');
 END IF;
 SYS.write_log('PROC_alumno.log');
EXCEPTION
 WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('ERROR: SE HA PRODUCIDO UNA EXCEPCION NO
CAPTURADA: ' || SQLCODE || ' - ' || SQLERRM);
   SYS.write_log('PROC_alumno.log');
END;
```

Segundo:

```
DECLARE
 v_acceso VARCHAR2(10) := 'Sureste';
 v_ventasTaquillasAntes NUMBER(11,2) := 0;
 v_ventasTaquillasDespues NUMBER(11,2) := 0;
 v_numAvisos INTEGER := 0;
BEGIN
 dbms_output.enable(100000);
```

```
 DBMS_OUTPUT.PUT_LINE('-- ----------------------------------------
----------------------');
 DBMS_OUTPUT.PUT_LINE('-- PRUEBA 2: Ejecucion de excepcion del
procedimiento.');
 DBMS_OUTPUT.PUT_LINE('-- debe capturar excepcion para indicar que
la puerta no existe.');
 DBMS_OUTPUT.PUT_LINE('-- Ademas se comprueba si se han modificado
las ventas de las taquillas.');
 DBMS_OUTPUT.PUT_LINE('-- ----------------------------------------
----------------------');
 DBMS_OUTPUT.PUT_LINE('-- ');
  SELECT SUM(ventas) INTO v_ventasTaquillasAntes
 FROM taquillas WHERE puerta = v_acceso;
  ventas_por_puerta(v_acceso);

 SELECT SUM(ventas) INTO v_ventasTaquillasDespues
 FROM taquillas WHERE puerta = v_acceso;
  IF v_ventasTaquillasAntes != v_ventasTaquillasDespues THEN
   DBMS_OUTPUT.PUT_LINE('ERROR: SE HAN MODIFICADO LAS VENTAS DE
TAQUILLAS.');
   v_numAvisos := v_numAvisos + 1;
 END IF;

 IF v_numAvisos = 0 THEN
   DBMS_OUTPUT.PUT_LINE('-- Las comprobaciones de prueba son
correctas, pero comprueba que la salida ');
   DBMS_OUTPUT.PUT_LINE('-- por consola sea idéntica a la esperada
para confirmar que el resultado');
   DBMS_OUTPUT.PUT_LINE('-- final es correcto.');
 END IF;
 SYS.write_log('PROC_alumno.log');
EXCEPTION
 WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('ERROR: SE HA PRODUCIDO UNA EXCEPCION NO
CAPTURADA: ' || SQLCODE || ' - ' || SQLERRM);
   SYS.write_log('PROC_alumno.log');
END;
```

Tercero:

```
DECLARE
 v_acceso VARCHAR2(10) := 'Oeste';
 v_ventasTaquillasAntes NUMBER(11,2) := 0;
 v_ventasTaquillasDespues NUMBER(11,2) := 0;
 v_numAvisos INTEGER := 0;
BEGIN
 dbms_output.enable(100000);
 DBMS_OUTPUT.PUT_LINE('-- ----------------------------------------
----------------------');
 DBMS_OUTPUT.PUT_LINE('-- PRUEBA 3: Ejecucion de excepcion del
procedimiento.');
 DBMS_OUTPUT.PUT_LINE('-- debe capturar excepcion para indicar que
la puerta no tiene');
 DBMS_OUTPUT.PUT_LINE('-- entradas vendidas.');
```

```
 DBMS_OUTPUT.PUT_LINE('-- ----------------------------------------
----------------------');
 DBMS_OUTPUT.PUT_LINE('-- ');
  SELECT SUM(ventas) INTO v_ventasTaquillasAntes
 FROM taquillas WHERE puerta = v_acceso;
  ventas_por_puerta(v_acceso);

 SELECT SUM(ventas) INTO v_ventasTaquillasDespues
 FROM taquillas WHERE puerta = v_acceso;
  IF v_ventasTaquillasAntes != v_ventasTaquillasDespues THEN
   DBMS_OUTPUT.PUT_LINE('ERROR: SE HAN MODIFICADO DATOS DE VENTAS
DE TAQUILLAS.');
   v_numAvisos := v_numAvisos + 1;
 END IF;

 IF v_numAvisos = 0 THEN
   DBMS_OUTPUT.PUT_LINE('-- Las comprobaciones de prueba son
correctas, pero comprueba que la salida ');
   DBMS_OUTPUT.PUT_LINE('-- por consola sea identica a la esperada
para confirmar que el resultado');
   DBMS_OUTPUT.PUT_LINE('-- final es correcto.');
 END IF;
 SYS.write_log('PROC_alumno.log');
EXCEPTION
 WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('ERROR: SE HA PRODUCIDO UNA EXCEPCION NO
CAPTURADA: ' || SQLCODE || ' - ' || SQLERRM);
   SYS.write_log('PROC_alumno.log');
END;
```