UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Elaboración de una plataforma para la realización de prácticas en formato remoto en sistemas embedded

Trabajo de fin de grado

Sales Garcia, Joan

Director: Jimenez Gonzalez, Daniel

Codirector: López Álvarez, David

22-06-2021

Resumen

Programación Consciente de la Arquitectura (PCA) es una asignatura del Grado en Ingeniería Informática que busca el aprendizaje de técnicas de optimización en lenguaje de alto nivel que dependen de la arquitectura del computador.

Hace unos años PCA utilizaba un servidor Intel que permitía a los estudiantes subir código para que fuera analizado automáticamente en base a la experiencia del profesorado. El análisis era para arquitecturas x86 y facilitaba tener un primer *feedback* asíncrono sin la presencia del profesorado. Esto impulsaba una mejor evolución del alumno ya que no se tenía que esperar a recibir respuesta. Tras el análisis se indicaba al alumnado si su código era correcto y se habían aplicado las técnicas de optimización esperadas.

Para añadir al temario la optimización mediante generación de hardware para una FPGA, PCA empezó a utilizar las Zedboard, unas FPGA con procesadores ARM. Este cambio supuso abandonar las funcionalidades del antiguo servidor Intel. También provocó que el profesorado tuviera que configurar sistemas con el software preparado en un disco duro externo para todas las tareas de compilación y soporte de las placas Zedboard.

Otra dificultad añadida era que los alumnos sólo podían disponer de las placas cuando iban al laboratorio, con la gestión de préstamo e instalación de cables y conexión que el proceso conlleva.

Con el objetivo de recuperar las funcionalidades perdidas y adaptarlas a la arquitectura ARM, este proyecto pretende desarrollar una plataforma específica.

Para ello se establece un sistema basado en dos componentes principales: un clúster de Zedboards que permita el trabajo remoto sin necesidad de préstamo, y una página web que posibilite el análisis de código del estudiante. Esta aplicación web también incorpora metodologías docentes basadas en la gamificación, como retos y rankings.

Resum

Programació Conscient de l'Arquitectura (PCA) és una assignatura del Grau en Enginyeria Informàtica que cerca l'aprenentatge de tècniques d'optimització en llenguatge d'alt nivell que depenen de l'arquitectura del computador.

Fa uns anys PCA utilitzava un servidor Intel que permetia als estudiants pujar codi perquè s'analitzés automàticament segons l'experiència del professorat. L'anàlisi era per arquitectures x86 i permetia tenir un primer *feedback* asíncron sense la presència del professorat. Això permetia una millor evolució de l'alumnat, ja que no s'havia d'esperar a rebre resposta.
Un cop finalitzat l'anàlisi s'indicava a l'alumne si el seu codi era correcte i s'havien aplicat les tècniques d'optimització esperades.

Per afegir al temari l'optimització mitjançant la generació de hardware per a una FPGA, PCA va començar a utilitzar les Zedboard, unes FPGA amb processadors ARM. Aquest canvi va suposar oblidar les funcionalitats de l'antic servidor Intel. També va provocar que el professorat hagués de configurar sistemes amb el software preparat en un disc dur extern per a la compilació i suport de les plaques Zedboard.

Una altra dificultat afegida era que els alumnes només podien disposar de les plaques quan eren al laboratori, amb la gestió de préstec i instal·lació de cables i connexió que això comporta.

Amb l'objectiu de recuperar les funcionalitats perdudes i adaptar-les a l'arquitectura ARM, aquest projecte pretén desenvolupar una plataforma específica.

Per aquest motiu es crea un sistema amb dos components principals:
un clúster de Zedboards que permeti el treball remot sense necessitat de préstec i una pàgina web que possibiliti l'anàlisi de codi de l'estudiant. Aquesta aplicació web també incorpora metodologies docents basades en la gamificació, com reptes i rankings.

Abstract

Architecture Conscious Programming (PCA) it’s a subject of the degree in Computer Engineering that aims for the learning of optimization techniques on high level languages that depend on the computer architecture.

Years back PCA made use of an Intel server that allowed students to upload code and be analyzed automatically according to the experience of the professoriate. The analysis was for x86 architectures and allowed students to have a first asyncronous *feedback* without the professoriate.

That let the student have a better progression, since there was no need for waiting for an answer. When the analysis was over students were told about the correctness of the code and if the optimization techniques were properly applied.

To add optimization by FPGA hardware generation, PCA started to use Zedboard, an FPGA with ARM processors. That change caused PCA to forfeit old server functionalities. There was also the need to configure systems with software installed on an extern hard drive to compile and support the Zedboards.

Another hardness was that students were only able to use the Zedboards when in the lab, with all the lending, installation and wire connection process that is implied.

To recover all lost functionalities and to adapt them to the ARM architecture, this project aims to develop a specialized platform.

A system with two main components is built: a Zedboard cluster that allows remote work without any lending, and a web page to analyze students’ code. This web application adds gamification learning methods, like challenges and rankings.

Índice de contenidos

Índice de figuras

Índice de tablas

Glosario

- boada: Nombre del clúster del departamento de Arquitectura de Computadores de la Facultad de Informática de Barcelona.
- Clúster: Agrupación de sistemas hardware.
- Daemon: Proceso en segundo plano que está siempre en funcionamiento.
- Gamificación: Técnica docente que consiste en aplicar conceptos de juego para aumentar la motivación del alumno.
- Query: Consulta a una base de datos.
- Salt: Caracteres generados de forma aleatoria que se añaden a un hash para aumentar la seguridad.
- Script: Programa sencillo escrito en lenguaje interpretado.
- Hash: Función criptográfica que genera un resultado único.

1. Introducción y contextualización

1.1 Definiciones relevantes

1.1.1 Programación Consciente de la Arquitectura (PCA)

PCA es una asignatura del Grado en Ingeniería Informática que pertenece a la especialidad de Ingeniería de Computadores.

El objetivo de la asignatura es enseñar técnicas de optimización en cuanto a tiempo de ejecución de distintas aplicaciones. Estos métodos para reducir el tiempo de ejecución dependen de la arquitectura hardware en la que se vaya a ejecutar dicha aplicación [1].

1.1.2 Zedboard Zynq-7000 ARM/FPGA SoC Development Board

Una Field-Programmable Gate Array (FPGA) es un dispositivo que permite reconfigurar sus bloques lógicos, y por tanto programar su funcionalidad [2].

Actualmente PCA hace uso de estas placas, tanto como para desarrollar hardware específico, como para ser la arquitectura objeto de estudio en que se realizan las optimizaciones. En la figura 1.1 se muestra una Zedboard.

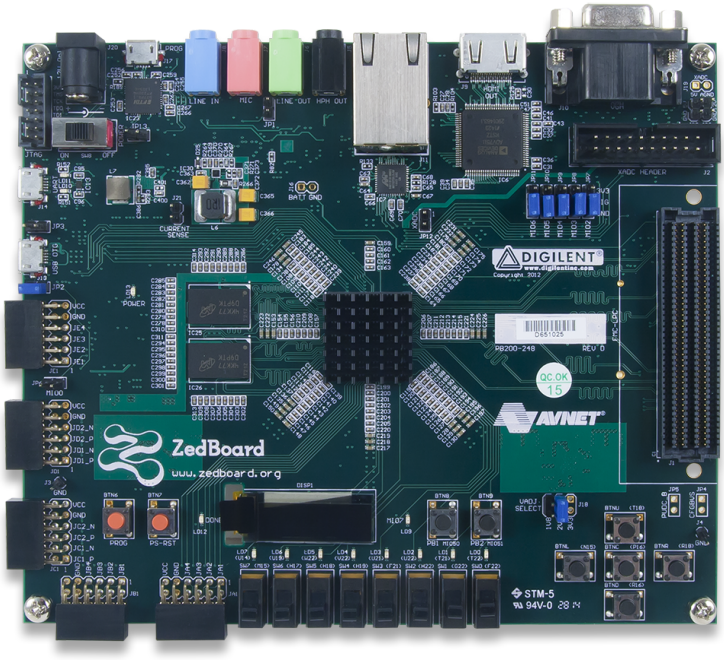


Figura 1.1. Zedboard Zynq-7000 [3].

1.2 Contexto

Antaño en PCA no se disponía de FPGAs, toda optimización era sobre Intel y se utilizaba un servidor al que se le podían enviar códigos para ejecutar. El servidor además implementaba un sistema de hitos.

Toda ejecución que llegaba se comparaba con dos hitos, y según si se superaban ninguno, uno o los dos se consideraba que el estudiante había superado correctamente el ejercicio. Además, había un sistema que analizaba de forma automática el código enviado y daba indicaciones al alumno sobre cómo mejorar la optimización.

Para añadir al temario la parte de desarrollo hardware se quiso cambiar a la arquitectura ARM. Se pidió una donación a través del Xilinx University Program y la UPC recibió 25 Zedboards Zynq 7000

Durante un tiempo, el alumnado utilizó estas placas en las horas de laboratorio

1.3 Problema

En el cuatrimestre de primavera, durante el curso 2019-2020, se decidió debido a la situación del COVID-19 pasar toda actividad docente a formato remoto [4]. Esto supuso un cambio para PCA, pues el estudiantado tuvo que llevarse las placas a casa para poder seguir trabajando con ellas.

Esto provocó una larga lista de problemas, ya que dar soporte técnico a distancia resultó ser una tarea compleja: no solo por la cantidad de horas necesarias, sino por la inviabilidad de resolver algunos de los problemas; discos que no funcionaban, tarjetas SD mal grabadas, cables rotos…

Para poder remediar estas dificultades y recuperar funcionalidades del antiguo servidor de PCA, se pretende crear una plataforma nueva para la asignatura que permita conectarse de forma remota.

1.4 Actores implicados

En este proyecto se verán beneficiados dos grupos.

En primer lugar, el estudiantado. No solo tendrá mayor facilidad para acceder a las placas, sin necesidad de llevárselas a casa o tener que esperar a horas concretas en que esté disponible el aula, sino que además tendrán un sistema enfocado a detectar qué partes de la asignatura no han acabado de interiorizar y cómo pueden mejorar.

En segundo lugar, el profesorado. La reducción de horas dedicadas a soporte técnico, la facilidad de evaluación y la ratificación de si el alumnado está adquiriendo los conocimientos necesarios son ventajas que aportará este proyecto.

2. Justificación

Como se ha comentado anteriormente, el objetivo de este proyecto es crear una prueba de concepto de esta plataforma. La idea es establecer la base y ofrecer soporte para uno de los laboratorios, y así crear unos cimientos para que en un futuro toda la asignatura pueda utilizar esta plataforma.

En concreto se desea dos funcionalidades principales: la posibilidad de conectarse de forma remota a una Zedboard para trabajar desde ella, y tener una aplicación web a la que enviar código, y que de forma automática se compile, realice el timing, compare el tiempo de ejecución con los hitos indicados, se analice el código y se indique al estudiante qué optimizaciones le faltan por hacer y si hay alguna parte del temario que no tenga clara.

Actualmente no existe ningún sistema que cumpla estos requisitos.

Lo más cercano es el antiguo servidor de PCA, que permite realizar envíos y aconsejar al estudiante sobre qué optimizaciones le faltan por realizar. Sin embargo no hace uso de las Zedboard, por lo que no es una alternativa que se desee, ya que se quiere aprovechar las placas para la parte de desarrollo de hardware específico y vectorización.

Por tanto, al tratarse de un problema tan específico, y que existen recursos únicos creados por el profesorado de PCA, se puede afirmar que no es posible encontrar ninguna solución ya existente a este problema.

3. Alcance del proyecto

El proyecto tiene dos objetivos principales.

1) Montar la infraestructura de las Zedboards y servidor.
2) Desarrollar la aplicación web.

3.1 Subobjetivos

Es posible dividir los dos objetivos principales en diversos subobjetivos.

1.1)Conectar físicamente las Zedboard a un servidor y tener conectividad.

1.2) Preparar la configuración inicial del servidor.

1.3) Instalar los requisitos de Slurm en el servidor.

1.4) Instalar Slurm en el servidor.

1.5) Configurar las placas como nodos de cálculo e instalar Slurm.

1.6) Comprobar el funcionamiento correcto de la cola.

1.7) Instalar y configurar un sistema de archivos distribuidos.

2.1) Crear una página web básica.

2.2) Añadir una base de datos.

2.3) Añadir un sistema de login.

2.4) Gestionar la comunicación entre Slurm, la web y la base de datos.

2.5) Implementar la aplicación que permita subir código y compilar.

2.6) Añadir función de análisis de código y *feedback*.

2.7) Añadir un sistema de recompensas a la aplicación web.

2.8) Diseñar un manual de usuario de la plataforma.

3.2 Requisitos funcionales

El objetivo del proyecto es crear una plataforma base que cumpla los siguientes requisitos:

1. Debe ofrecer soporte para trabajar de forma remota.

2. Se debe poder gestionar de forma remota.

3. Debe ofrecer funcionalidades para enviar y ejecutar código de forma controlada y analizarlo de forma automática.

4. Debe ser capaz de aportar *feedback* al estudiante sobre qué optimizaciones puede aplicar para mejorar el resultado.

5. Debe poder enviar *feedback* al estudiante sobre qué conceptos no tiene claro, y aportar ejercicios para repasarlos.

6. Debe dejar ver al docente una lista con todos los envíos de los estudiantes para analizar su progresión.

3.3 Requisitos no funcionales

Además de los requisitos específicos del sistema, también se deben respetar los siguientes requisitos no funcionales:

-Usabilidad: Debe ser intuitiva y fácil de usar, para que tanto el profesorado como los alumnos apenas necesiten tiempo para familiarizarse con ella.

-Disponibilidad: Para ofrecer flexibilidad de horas para que los estudiantes trabajen el servicio debe estar siempre disponible. Además, no se debe permitir que el usuario pueda perder parte del trabajo realizado, por tanto debe tener protección adicional para evitar caídas y errores.

-Adaptabilidad: Al tratarse de una base que permitirá expandir la plataforma en un futuro, se debe tener especial cuidado en que el sistema sea escalable y ampliable.

-Seguridad: Para evitar un posible mal uso, sea intencionado o accidental, el sistema debe cumplir un estándar de seguridad a la vez que limita lo menos posible al estudiante.

-Soporte a la docencia: Al ser una herramienta enfocada a la docencia, debe aplicar conceptos que faciliten la motivación y aprendizaje del alumnado.

3.4 Posibles complicaciones

Al plantear el proyecto se prevén algunas dificultades:
En primer lugar, la falta de experiencia en algunos temas, como el desarrollo de aplicaciones para servidor, o la elaboración de páginas web.
Además, las Zedboard han sido configuradas previamente por el profesorado de la asignatura, por tanto hay que añadir el desconocimiento de la arquitectura a la lista de complicaciones.
Por último, se ha establecido una fecha de entrega inamovible, lo que puede provocar que en algunos casos se tenga que priorizar alguna tarea.

4. Metodología

4.1 Metodología de trabajo

Debido a los problemas comentados anteriormente, como la falta de experiencia en algunos puntos del proyecto o el límite de tiempo, se necesita una metodología que aporte flexibilidad y productividad, y que permita trabajar en proyectos con objetivos cambiantes.
Por ello se utilizará una metodología ágil [5], en concreto Scrum. Esto permitirá organizar cortos periodos de trabajo enfocados a la realización de un objetivo concreto, y si surge alguna dificultad o se ha de replantear el objetivo poder adaptarse rápidamente.

4.2 Herramientas

Para el desarrollo del proyecto se utilizarán diferentes herramientas que facilitan la organización y comunicación.

Respecto a la comunicación, se usa Atenea [6] como intranet del proyecto, y también el servicio de correo electrónico de Google para la comunicación con el director, el codirector del proyecto, y la tutora encargada de la gestión del proyecto.

Para la organización se hace uso de Google Drive [7] para redactar y compartir el documento. Se ha elegido esta opción respecto a otras alternativas ya que es gratuito, es un recurso en línea -lo que permite acceder desde cualquier lugar- y además incluye un sistema de comentarios muy visual. Por último, también se usará trello [8], que permite escribir tareas, organizarlas en diferentes estados (en proceso, hechas…), mantener una *checklist* de los pasos intermedios realizados e incluso añadir deadlines que recuerden para cuando debe estar terminada una tarea.

5. Planificación temporal

Como se ha comentado anteriormente, este proyecto tiene un límite de tiempo estricto, en concreto, tiene su inicio el 22 de febrero de 2021, y debe estar terminado a finales de junio.
El proyecto consiste de 18 ETCS, lo que traduce a unas 450 horas de dedicación.

En un principio, se había calculado que a un ritmo de 8 horas al día, el proyecto estaría terminado el 22 de abril.

Debido a complicaciones en algunas de las tareas y a una estimación errónea de la duración de estas se ha tenido que utilizar el mes libre disponible que se explicita en el apartado de gestión de riesgos.

Finalmente el proyecto se ha terminado el 10 de junio.

5.1 Tareas

Para poder organizar bien el tiempo, se ha dividido el proyecto en tareas.
Se consideran 4 bloques principales:

1) Gestión del proyecto: Corresponde a las tareas de planificación del proyecto, y la redacción de la documentación pertinente.

2) Desarrollo: Incluye las tareas de investigación, desarrollo e implementación de la parte técnica del proyecto.

3) Pruebas: Contiene las tareas que corresponden a las pruebas que se deben realizar para cerciorarse del correcto funcionamiento de la plataforma desarrollada.

4) Documentación: Recoge las tareas relacionadas con la composición de la memoria final del proyecto.

A continuación se describe en más detalle dichas tareas. Todas las tareas de la parte de desarrollo incluyen de forma implícita la investigación de alternativas y posibles métodos de implementación, así como pequeñas comprobaciones de funcionamiento. En aquellas en las que se prevé especial complicación se ha explicitado dicha necesidad.

5.1.1 Gestión del proyecto

- GP1. Definición del alcance y contextualización: Redacción de la documentación que corresponde a la descripción del contexto, alcance, justificación y metodología del proyecto. Duración aproximada de 20 horas.

- GP2. Planificación temporal: Redacción de la documentación que corresponde a la planificación del proyecto. Duración aproximada de 20 horas. Requiere GP1

- GP3. Costes económicos y sostenibilidad medioambiental: Redacción de la documentación que contiene los cálculos relacionados con costes. Duración aproximada de 20 horas. Requiere GP2

- GP4. Elaboración del documento. Repaso, retoques finales y corrección de errores en los anteriores documentos. Duración aproximada 10 horas. Requiere GP1, GP2, GP3

- S1. Reuniones de seguimiento. Reuniones con el director y el codirector del proyecto, para ratificar que se están siguiendo las especificaciones demandadas. Duración aproximada de 20 horas.

- PR1. Preparación de la defensa. Practicar la defensa del proyecto y elaborar soporte visual que facilite la presentación. Duración aproximada de 10 horas. Requiere DC1.

5.1.2 Desarrollo

- D1. Montaje físico de la infraestructura y red. Instalación y configuración de las placas para realizar pruebas de las que se dispone, instalación y configuración de la red. Duración aproximada de 10 horas.

- D2. Sistema de usuarios. Investigación e implementación de un sistema de usuarios para las placas, y las pruebas de funcionamiento y seguridad correspondientes. Duración aproximada de 30 horas. Requiere D1.

- D3. Sistema de ficheros. Creación de un sistema de archivos restringido según el usuario. Duración aproximada de 10 horas. Requiere D2.

- D4. Gestión y resolución de errores. Duración aproximada de 40 horas.

- D5. Sistema de colas. Implementación de sistema que permita enviar a una Zedboard independiente varias peticiones. Duración aproximada de 50 horas.

- D6. Desarrollo de una aplicación para la Zedboard. Desarrollo de la aplicación que permite compilar, hacer timing y calcular métricas en la Zedboard y enviar los resultados al servidor principal. Duración aproximada de 40 horas. Requiere D5.

- D7. Sistema de análisis de código y *feedback*. Añadir las funcionalidades deseadas del antiguo servidor de PCA y añadir el sistema de *feedback*. Duración aproximada de 40 horas. Requiere D5.

- D8. Desarrollo de la aplicación web. Elaboración de una aplicación web que haga uso del sistema de colas implementado. Duración aproximada 10 horas. Requiere D7.

- D9. Sistema base de datos. Creación de la base de datos y la comunicación con las placas, el servidor, y la estructura para hacer envíos y peticiones desde la página web. Duración aproximada 40 horas. Requiere D8, D5.

- D10. Sistemas enfocados a la docencia. Añadir diversas funcionalidades enfocadas a facilitar el trabajo del profesorado y motivar al alumnado. Duración aproximada 40 horas. Requiere D9.

5.1.3 Pruebas

- P1. Comprobación del sistema. Verificación del correcto funcionamiento del sistema, corrección de errores y aplicación de posibles mejoras de usabilidad y rendimiento. Duración aproximada de 30 horas. Requiere D3, D4, D6 y D8.

- P2. Comprobación de la seguridad del sistema. Verificación de permisos de todo el sistema, protección contra inyección de código y otras medidas de seguridad. Duración aproximada de 10 horas. Requiere D5, D10.

5.1.4 Documentación

- DC1. Redacción de la memoria. Documentación que pertenece al proyecto entero. Duración aproximada de 80 horas. Requiere GP4, P1.

- DC2. Preparación del manual de usuario. Documentación enfocada a ayudar al profesorado y alumnado a entender todas las características de la plataforma. Parte de la memoria. Duración aproximada de 20 horas. Requiere D10.

En la tabla 1 se pueden ver las nuevas tareas.

Tarea	Duración aproximada (horas)	Dependencias
Gestión del proyecto		
GP1. Planificación	20	-
GP2. Metodología	20	GP1
GP3. Presupuesto	20	GP2
GP4. Redacción final	10	GP3
S1. Reuniones de seguimiento	20	-
DC1. Redacción de la memoria	80	P1,P2
PR1. Preparación de la defensa	10	DC1
Desarrollo		
D1. Montaje físico	10	-
D2. Sistema de usuarios	30	D1
D3. Sistema de ficheros	10	D2
D4. Sistema de gestión	40	-
D5. Sistema de colas	30	-
D6. Aplicación de servidor	40	D5
D7. Sistema de *feedback*	40	D5
D8. Aplicación web	40	D7
D9. Sistema base datos	40	D8
D10. Funciones docencia	40	D9
P1. Pruebas de funcionamiento	30	D3,D4,D6,D8
P2. Pruebas seguridad	10	D10,D6
DC2. Manual de usuario	20	D10,D6

Tabla 1. Resumen de tareas.

5.1.6. Diagrama de Gantt

Para ayudar en la planificación y poder visualizar con mayor facilidad el reparto de tareas se utiliza un diagrama de Gantt.
Dicho recurso ayuda a tener presente las dependencias entre distintas tareas, el reparto de la carga de trabajo y la existencia de algún posible camino crítico. Es decir, una sucesión de tareas en que si una se ralentiza afectará a gran parte del proyecto.

En la figura 5.1 se puede ver el diagrama.

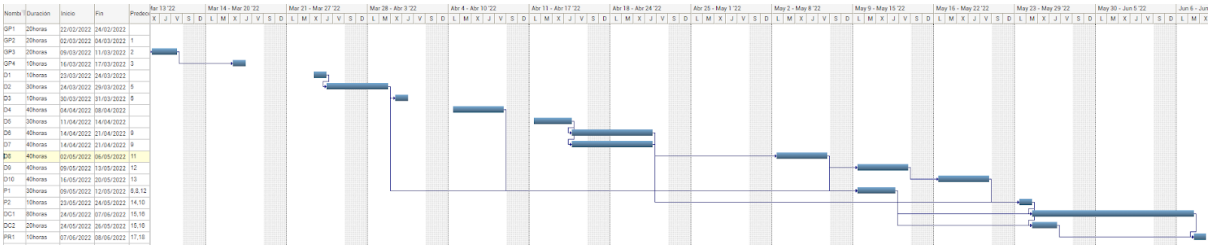


Figura 5.1. Diagrama de Gantt, elaboración propia con Gantter [9].

Como se puede observar, las tareas parecen ser cortas, eso se debe a una gran división del trabajo y a la alta disponibilidad del autor para dedicarle al proyecto (que como se ha comentado anteriormente, es de 8 horas al día). Además, las tareas están bastante alejadas unas de otras, ya que en varias ocasiones se ha decidido no empezarlas consecutivas, y dar un par de días de margen para en el caso de que si surja algún imprevisto tener más margen para poder actuar.

6. Gestión de riesgos

Como se ha mencionado con anterioridad, uno de los principales problemas de este proyecto es el límite de tiempo, lo que provoca que cualquier riesgo añadido pueda afectar no solo a la tarea a la que va asociado, sino también a las que dependen de ella.

Además, otros de los riesgos comentados son el desconocimiento sobre la arquitectura, la configuración inicial de las Zedboard, y varias competencias como la administración de sistemas operativos o el desarrollo de páginas web. Para paliar el efecto que puedan tener estos problemas, se ha decidido sobreestimar el tiempo en horas necesario para realizar algunas de las tareas. También se han separado temporalmente muchas de las tareas, para tener margen y poder solucionar algunos problemas conforme vayan surgiendo, a la par que se intenta reducir al máximo el efecto que tienen dichas complicaciones sobre tareas que dependan de ellas.

Asimismo, se dispone de un mes libre para desarrollar rectificaciones en la planificación original, por si el margen anterior resultara insuficiente.

7. Recursos

Para la correcta realización del proyecto se necesita disponer de una serie de recursos que se detallan a continuación.

7.1 Recursos humanos

En cuanto a los recursos humanos, son necesarios un jefe del proyecto que se encargue de la gestión del proyecto y las tareas relacionadas y un ingeniero informático para todas las tareas técnicas de desarrollo.

7.2 Recursos hardware

Se han utilizado dos placas Zedboard Zynq 7000 para investigación, pruebas de funcionamiento y desarrollo de la plataforma antes de la instalación definitiva.

Además, se requiere de un ordenador personal que pueda cumplir la función de servidor mientras se desarrolla la prueba de concepto. También es necesario disponer de un switch que permita conectar las placas a la misma red de forma simultánea.

7.3 Recursos software

Para todo el desarrollo del proyecto se necesita gran variedad de software. En primer lugar, para la parte de gestión del proyecto se utiliza, como se ha mencionado en el apartado de herramientas de metodología, google drive y trello.
Para la parte técnica se requiere VirtualBox, un software libre de virtualización de sistemas operativos; VisualStudio, para algunas tareas que requieran desarrollo de código y la aplicación de terminal de Ubuntu para un *host* Windows, para hacer pruebas de conexión hacia la máquina virtual.

Por último, la parte de instalación de la plataforma necesita software específico según cómo se decida implementar, entre ellos Slurm, un gestor de clúster.

8. Presupuesto

A continuación se detallan los diversos costes asociados que tiene el proyecto. Como en toda gran empresa, es realmente importante realizar un estudio previo sobre el coste económico, para determinar la viabilidad del proyecto. Se dividen dichos costes en las categorías de personal, genéricos, contingencia e imprevistos.

8.1 Costes de personal

Anteriormente se han detallado las necesidades de personal. Estas consisten en un ingeniero informático y un jefe de proyecto.

El jefe de proyecto tiene la tarea de supervisar el avance del proyecto, y corroborar que se estén cumpliendo las expectativas, asimismo como encargarse de cualquier tarea relacionada con la gestión del proyecto.
El ingeniero informático, en este caso el autor del proyecto, es quién se encarga de todas las tareas de investigación, desarrollo, montaje y certificación del correcto funcionamiento de la plataforma.

La tabla 2 representa los costes económicos de estos roles [10] [11].
También se tiene en cuenta el coste de la seguridad social.

Posición	Sueldo bruto anual	Sueldo bruto (hora)	Coste con SS (hora)
Jefe de proyecto	42.268 €	14€	18€
Ingeniero informático	33.100€	9€	11.5€

Tabla 2. Sueldos de personal, elaboración propia.

Para calcular los costes con mayor exactitud, se ha relacionado cada tarea y las horas de dedicación necesarias con el sueldo bruto del personal, tabla 3, esto se conoce como coste por actividad, CPA.

Código	Tarea	Jefe del proyecto	Programador	Coste
	Gestión del proyecto	**180h**		**3.240€**
GP1	Alcance y contextualización	20h		360€
GP2	Planificación temporal	20h		360€
GP3	Costes económicos y sostenibilidad	20h		360€
GP4	Elaboración de documentación	10h		180€
S1	Reuniones de seguimiento	20h		360€
	Desarrollo		**270h**	**3.105€**
D1	Montaje físico		10h	115€
D2	Sistema de usuarios		30h	345€
D3	Sistema de ficheros		10h	115€
D4	Sistema de resolución errores		40h	460€
D5	Sistema de colas		30h	345€
D6	Aplicación Zedboard		40h	460€
D7	Análisis de código y *feedback*		40h	460€
D8	Desarrollo aplicación web		40h	460€
P1	Comprobación del sistema		30h	345€
DC1	Redacción de la memoria	80h		1.440€
PR1	Preparación de la defensa	10h		180€
	Total CPA			**6.345€**

Tabla 3. Costes de personal según las tareas, elaboración propia.

8.2 Costes genéricos

Existen costes que no están relacionados con la subdivisión en tareas, sino que afectan a todo el proyecto. Se pueden ver a continuación:

8.2.1 Hardware

Para la realización del proyecto se necesitan 2 placas Zedboard Zynq 7000, para las pruebas de configuración. Son la arquitectura para la que se está desarrollando esta plataforma, así que es importante disponer de ellas físicamente. Para la prueba de concepto que se está desarrollando dos serán suficientes. El precio individual de cada placa es de 449€ [3].

También se necesita un ordenador personal, que permitirá tanto hacer las funciones de servidor al que conectar las placas, como las tareas de desarrollo de una página web, elaboración de la documentación, investigación y reuniones de seguimiento. No es necesario disponer de un equipo con gran capacidad de procesado, ni una tarjeta gráfica dedicada, por lo que es posible reducir costes. El coste de un ordenador que cumpla los requisitos básicos sería de 346€ [12].

Por último, para la infraestructura se necesita un switch que permita conectar las Zedboard. Un switch sencillo con 3-4 entradas es suficiente, y cuesta 20€ [13].

8.2.2 Software

Como se ha comentado en el apartado de metodología, no se utiliza ningún tipo de software privativo, y por tanto el coste es de 0€.

8.2.3 Espacio de trabajo

Es vital tener un espacio dedicado a la realización del proyecto.
El precio de alquiler de un piso en Barcelona es de 630€ mensuales [14]. Como se prevé que el proyecto esté terminado en tres meses, el total asciende a 1890€.

8.2.4 Factura de Internet

Es necesario disponer de una conexión a Internet, tanto para las reuniones de seguimiento, como para la investigación. La tarifa más básica de una de las compañías que trabajan en el área de Barcelona en que se ubica el espacio de trabajo es de 40€ al mes [15]. Por tanto se asume que el coste de 3 meses será 120€.

8.2.5 Total costes genéricos

Como conclusión, en la tabla 4 se puede ver el total de los costes mencionados.

Concepto	Coste
Hardware	1.264€
Software	0€
Espacio de trabajo	630€
Factura de internet	120€
Total	2.014€

Tabla 4. Costes genéricos, elaboración propia.

8.3 Contingencias

Se debe disponer de un fondo de reserva para emergencias para poder resolver cualquier coste imprevisto que pueda surgir. No se han valorado algunas necesidades económicas de menor importancia, así que para cubrir esos importes el fondo consistirá en un 30% de los costes calculados. El coste total de personal es de 6345€ y el total de costes genéricos es de 2014€, que suman un total de 8359€.

El 30% de este total, y por tanto la cantidad destinada a contingencias es de 2518€

8.4 Imprevistos

En el apartado de riesgos se menciona la posibilidad de tener que reestructurar parte del proyecto y establecer una solución alternativa.
Si fuera necesaria esta medida, habría que incrementar el coste de personal.

A esta nueva planificación se le asignan 100 horas, 20 horas del jefe de proyecto, y 80 horas del ingeniero informático.
Por último, la vida útil del hardware supera el tiempo necesario para la realización de este proyecto, por lo que no debería haber ningún problema. Sin embargo, solo por prevenir, se tiene en cuenta el coste que implicaría tener que volver a comprar parte del hardware.

En la tabla 5 se pueden apreciar estos costes asociados a posibles imprevistos.

Concepto	Coste
Nuevo ordenador	346€
Nueva Zedboard	449€
Incremento de horas (jefe de proyecto)	360€
Incremento de horas (ingeniero informático)	920€
Total	2.075€

Tabla 5. Costes imprevistos, elaboración propia.

8.5 Coste total

Como conclusión, en la tabla 6 es posible observar el cálculo total de todos los costes mencionados con anterioridad. Esto supondría el coste total del proyecto.

Concepto	Coste
Costes de personal	6.345€
Costes genéricos	2.014€
Contingencias	2.518€
Imprevistos	2.075€
Total	12.952€

Tabla 6. Coste total del proyecto, elaboración propia.

8.6 Control de gestión

Para cerciorarse de que la estimación es correcta, y corregirla en caso de que no resulte serlo, se implementan unos mecanismos de control.
Este proceso de revisión se realizará en dos situaciones, para que sea posible redirigir cualquier desviación y solucionarla cuanto antes.

Se hará una comprobación semanalmente en las reuniones del proyecto, y además al finalizar cada tarea se evaluarán los costes añadidos, las horas de dedicación que han sobrepasado las estimadas, y se anotará cualquier imprevisto que haya surgido durante la realización.
Se utilizarán las siguientes métricas para llevar a cabo la gestión:

1. Desviación del coste de personal
   (coste estimado - coste real)*número de horas

2. Desviación de las tareas en horas
   (horas estimadas - horas reales)*coste real

9. Sostenibilidad

9.1 Autoevaluación

Si bien la sostenibilidad se ha tratado en numerosas ocasiones a lo largo de la carrera, noto la falta de experiencia al tratar el tema. Tras realizar la encuesta de sostenibilidad he sido capaz de identificar mejor aquellos apartados que no domino demasiado, y así poder informarme más para intentar remediarlo.

A lo largo del proyecto se ha tenido bastante presente el aspecto social, pues es una de las motivaciones principales para este trabajo, como se detalla más adelante. Sin embargo, la dimensión económica, y en concreto la ambiental, han estado en un segundo plano.

9.2 Dimensión económica

El coste del proyecto, valorado en 12.952€, parece elevado.
Hay que tener en cuenta sin embargo que se ha hecho un estudio completo de los costes, y en realidad hay muchos conceptos que no supondrán un coste, pues se puede reutilizar hardware.

Como se ha indicado en el apartado de contextualización, no existe una alternativa en el mercado que cumpla todas las funcionalidades que se desean, aunque existen algunos recursos hardware y software similares que la UPC utiliza actualmente, aunque son fruto de muchos años y distintos proyectos, así que no tiene sentido comparar su coste.
Por último, se prevé que sea una plataforma que pueda ir actualizándose a lo largo de los años, por lo que la vida útil del proyecto es elevada.

9.3 Dimensión ambiental

El principal impacto medioambiental que pueda tener este proyecto es el hecho de realizar el cambio de funcionamiento de las placas. Anteriormente solo se conectaban cuando se utilizaban en clase, o en casa de algún alumno. Tras la realización de este proyecto, la idea es que las placas estén siempre disponibles en un clúster.
Esto no es tan grave, pues el consumo de las placas en estado *idle* es muy reducido, y no tiene un gran impacto.

Por último, no es necesario considerar ningún impacto añadido debido al servidor, pues actualmente ya se utiliza uno, y la aplicación de la nueva plataforma funcionará sobre esa máquina.

9.4 Dimensión social

Como se ha comentado anteriormente, este aspecto es una de las principales motivaciones para el proyecto.
Al desarrollar una plataforma que dé soporte a la docencia no solo se está ayudando al profesorado, que está muy interesado en este proyecto y lleva años queriendo llevarlo a cabo, sino también al alumnado.
La posibilidad de conectarse de forma remota en cualquier momento a las Zedboard y trabajar de forma remota reduce las dificultades que se han planteado anteriormente.
Además, el sistema de análisis de código y *feedback* permitirá aclarar con mayor exactitud de qué conocimientos carece el estudiantado y reforzarlos, lo que ayudará a mejorar la calidad de la docencia.

Al tener en cuenta pues, que parte del objetivo del proyecto es mejorar la calidad de vida de las personas, es posible afirmar que se pretende tener un impacto social positivo.

10. Legislación relevante

Con motivo de la necesidad de guardar y utilizar datos de personas reales se plantea si el proyecto cumple la legislación vigente.
Una de las leyes que afectan es la ley orgánica de protección de datos personales.
"en lo que respecta al tratamiento de sus datos de carácter personal deben, cualquiera que sea su nacionalidad o residencia, respetar sus libertades y derechos fundamentales, en particular el derecho a la protección de los datos de carácter personal" [16].

Como en todo momento los datos que se gestionan son los ofrecidos por la universidad, y son únicamente nombres y direcciones electrónicas, se considera que no se inclumple esta ley. Además, por razones de confidencialidad no se mostrará directamente el nombre de los usuarios en la página web, sino un ID asociado.

Otra ley que se debe considerar es el real decreto de la ley de propiedad intelectual [17].
Antes de subir un archivo a la página web se advierte de que los usuarios se comprometen a subir código generado por ellos. Este código solo será utilizado y analizado según los propósitos de la asignatura, y se eliminará del servidor al finalizar el curso.
Por estas razones, tampoco se incumple ningún apartado.

11. Diseño

11.1 Visión general de la plataforma

Debido a los requisitos de alta disponibilidad, la necesidad de permitir conexiones remotas simultáneas y de realizar una ejecución controlada se ha decidido utilizar un clúster.

Slurm es el gestor de clúster que se ha instalado recientemente en boada. Para facilitar el aprendizaje de la herramienta tanto a alumnado como a profesorado se ha considerado buena idea aprovechar y utilizar el mismo sistema, pues ambos grupos están familiarizados con ella.

Esta infraestructura permite a los alumnos conectarse al servidor y desde ahí trabajar en interactivo directamente, enviar *jobs* a la cola del clúster formada por todas las Zedboard o solicitar que se reserve una de las placas de la cola para poder trabajar interactivamente en ella y así realizar otras tareas como *debuggar*, o simplemente ejecutar sin *overheads* añadidos.

Además, para garantizar la persistencia de los datos es necesario disponer de una base de datos.
Slurm está configurado para trabajar con bases de datos relacionales, por lo que se ha decidido utilizar MariaDB, una base de datos que utiliza el lenguaje SQL y además es software gratuito.

Por último, para permitir que el estudiantado suba su código y se analice automáticamente se ha desarrollado una aplicación web. Esta página utiliza un sistema de ranking y medallas, que se explican en profundidad en el apartado 12.2.

En la figura 11.1 se muestra un esquema de toda la plataforma.

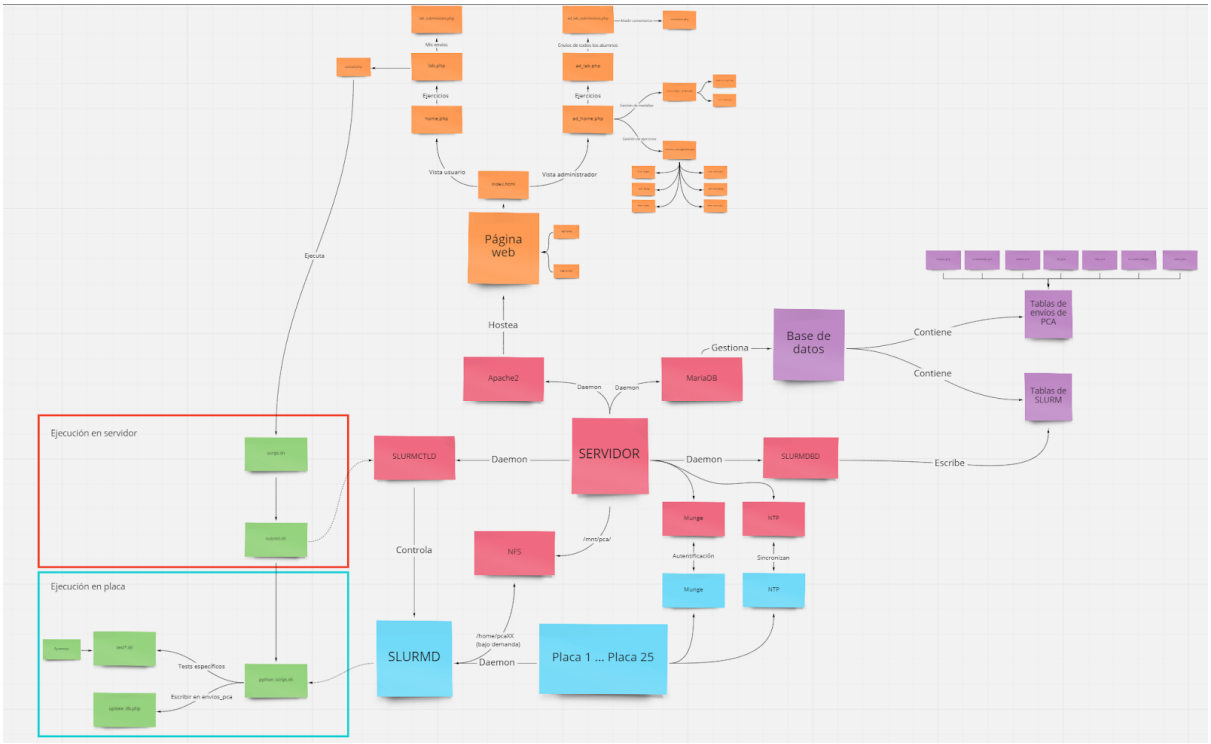


Figura 11.1. Visión global de todo el sistema.

Distribuidos por colores se indican los principales componentes del sistema, posteriormente se explica en detalle cada módulo.

-Rojo, centro de la imagen: Servidor. Aquí están indicados todos los daemon [18] que controlan o se comunican con el resto de módulos.

-Azul, parte inferior de la imagen: Clúster de placas. Se muestra los daemon que corren en cada placa y la comunicación con el servidor.

-Naranja, parte superior de la imagen: Aplicación web. Se enseñan las funcionalidades de la página web y el archivo de código que las gestiona.

-Lila, parte derecha de la imagen: Base de datos. Especifica las tablas necesarias.

-Verde, parte izquierda de la imagen: Proceso de envío desde la web. Se aclara el proceso que se sigue para realizar un envío y analizar el código.

11.2 Infraestructura de la red

La red se ha distribuido como se indica en la figura 11.2.

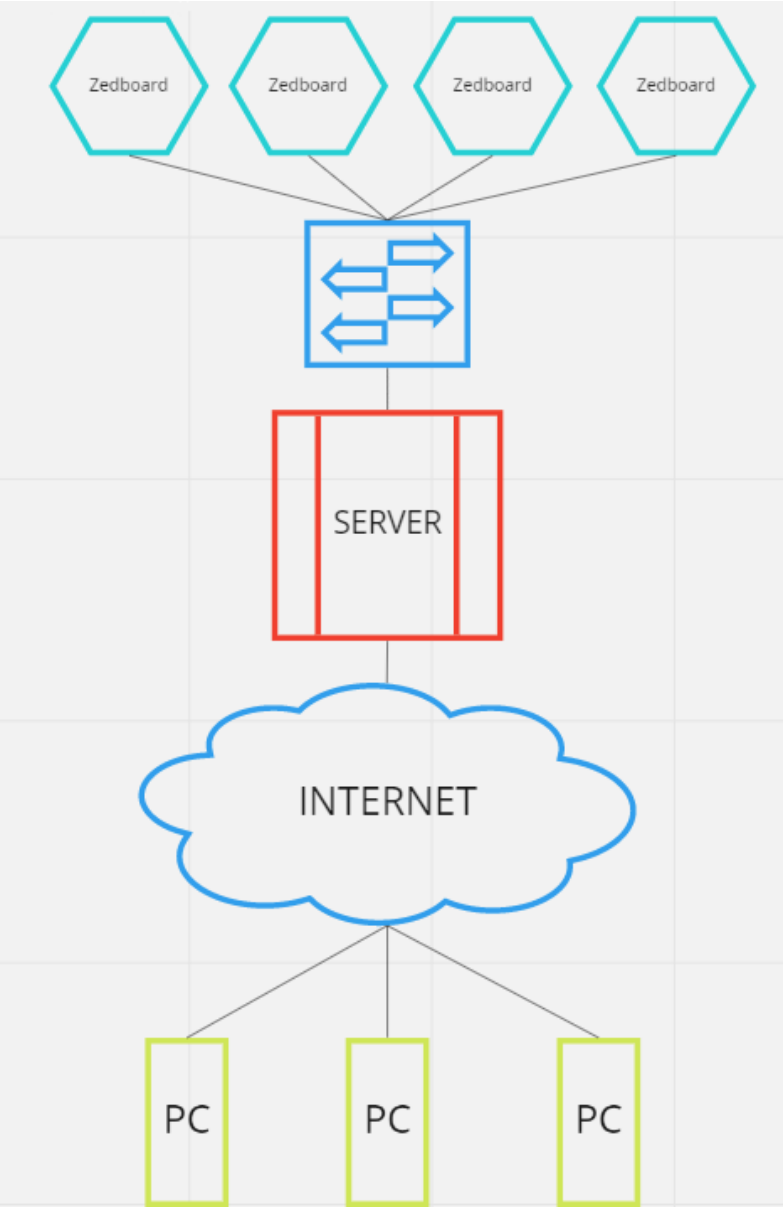


Figura 11.2. Esquema de la red.

Se utiliza un switch para conectar las placas al servidor con la intención de reducir costes y simplificar la configuración de la red.
El servidor y las Zedboard permiten las conexiones por ssh, y se han montado por Network File System (NFS) diversos directorios para facilitar la comunicación entre servidor y nodos.
En el apartado de implementación se explica en más detalle.

12. Implementación

12.1 Clúster

El servidor es el núcleo de toda la plataforma, cumple las funciones de:

-Nodo de entrada al sistema
-Gestión del cluster
-*Hostear* la página web
-Mantener la base de datos

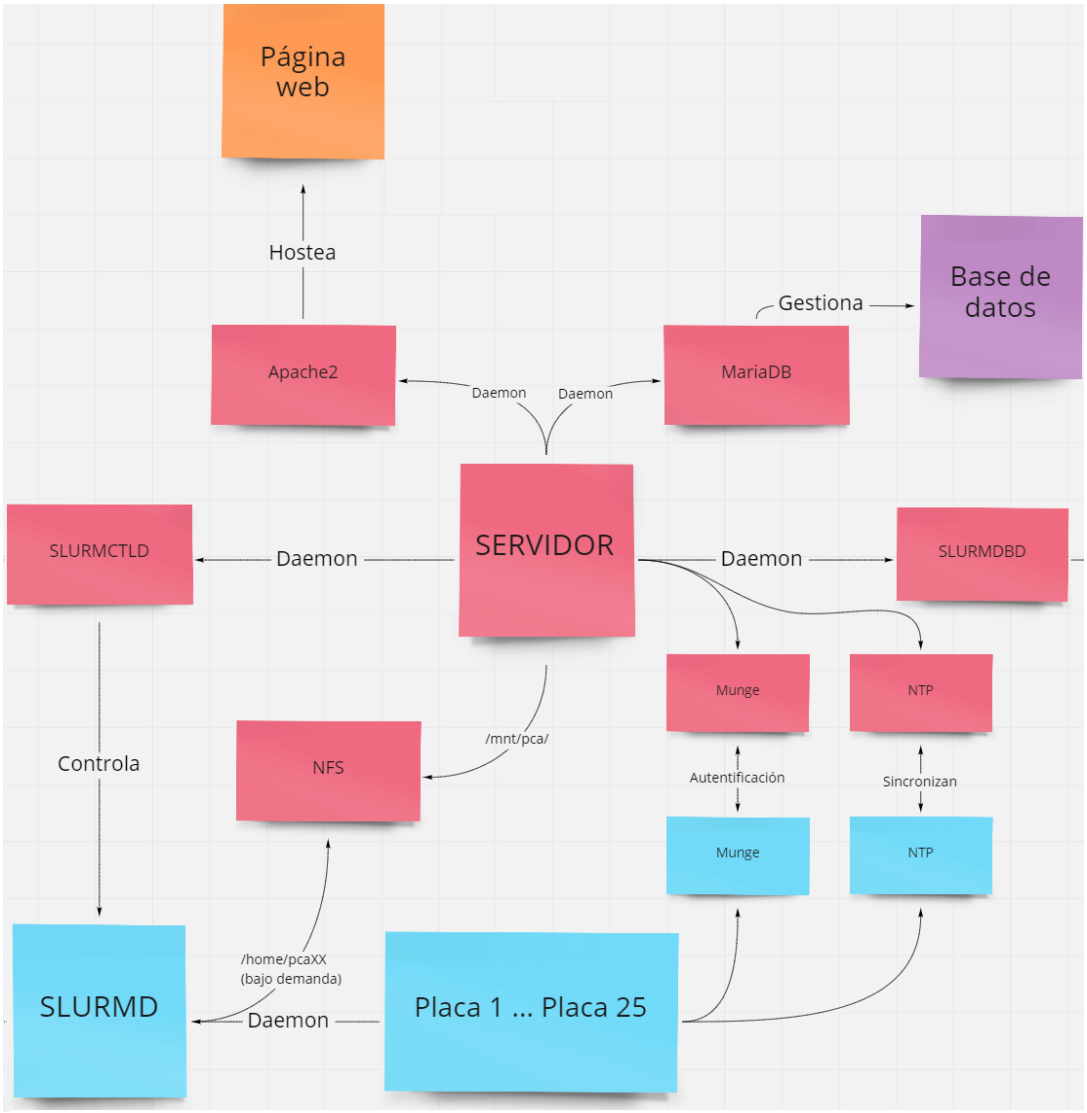


Figura 12.1. Visión del servidor.

En el servidor se ejecutan los daemon de la tabla 7.

Daemon	Función
Apache2	Levantar y hostear la página web.
MariaDB	Gestionar la base de datos del sistema.
Munge	Sistema de autenticación. Requisito de Slurm.
NTP	Sincronizar fecha y hora en todo el clúster. Requisito de Slurm.
NFS	Sistema de archivos compartidos mediante red. Útil para la comunicación entre placas y servidor.
slurmdbd	Controlar accounting de Slurm y escribir en la base de datos.
slurmctld	Controlador principal Slurm. Comunica con slurmd.
slurmd	Controlador individual de cada nodo.

Tabla 7. Daemons del servidor y su función.

12.1.1 Nodo de entrada

Permite conexiones entrantes mediante ssh. Los usuarios que se conectan tiene acceso a un directorio */home* y no se les otorga permisos de administrador. Esto es un cambio respecto al sistema anterior en que a cada alumno se le daba una Zedboard, dónde sí tenían un usuario con permisos de sudo. Para proteger el sistema, tanto las placas como el servidor, se ha decidido que no se les otorgará a los usuarios permisos elevados en ningún momento.

Además de trabajar directamente en sus directorios correspondientes del servidor, los usuarios pueden utilizar directamente el clúster con los comandos que ofrece Slurm, como sbatch, para enviar un script a la cola de ejecución.

Por último, se quería un modo interactivo para que los alumnos pudieran conectarse directamente a las placas mediante el clúster, por lo que se ha definido un alias que realiza esta conexión, así un usuario puede trabajar directamente desde la placa, mirar el código ensamblador, *debuggar*…

El alias es sconnect, y llama a un script que ejecuta el siguiente código:

```
#!/bin/bash
salloc srun --time=240 --pty bash
```

Figura 12.2. Alias sconnect

Este script hace un salloc seguido de un srun cuando encuentra un nodo disponible. El srun abre una terminal de bash para que el usuario pueda trabajar. Se limita el tiempo a 240 minutos (4 horas) para evitar la posibilidad de que algún usuario deje la terminal abierta y esté consumiendo un recurso que no se está utilizando.

Se han pensado alternativas a la forma de conectarse de forma interactiva. Una opción era utilizar el sistema que tiene boada y ejecutar un script que encola al usuario cada vez que se inicia sesión, aunque se descartó ya que entonces todo usuario estaría conectado a alguna placa, lo que dejaría pocas (o ninguna, dependiendo del número de alumnos) libres para poder ejecutar código y aprovechar la página web.
Además, si se quiere poder compilar hardware se debería aprovechar otro componente que se añada posteriormente, ya que en las Zedboard es inviable. El software de compilación sólo está para Intel y se requiere de un sistema con una capacidad de cálculo elevada y una memoria RAM de más de 8GB, mientras que las placas solo tienen 512MB.

Otra alternativa fue la posibilidad de crear dos particiones en Slurm y dividir los nodos. Habría cierta cantidad de placas reservadas para el uso interactivo y otras para la aplicación web. Finalmente se decidió descartar también esta opción y optar por la del alias, ya que así todos los nodos se pueden utilizar para ambas funciones, según se requiera.

12.1.2 Gestión del clúster

Daemons

Hay 4 *daemons* estrictamente relacionados con el correcto funcionamiento del clúster: Munge, Network Time Protocol (NTP), slurmctld y slurmdbd.
Los dos primeros son debidos a requisitos de Slurm, para poder instalar el gestor del clúster es necesario [19] tener todos los usuarios sincronizados en todo el sistema, que la fecha y hora sea consistente en todos los dispositivos y un método de autenticación que garantice que los mensajes que se intercambian slurmctld y slurmd no son fraudulentos.

Para cumplir el requisito de la fecha y hora se utiliza NTP. Este programa accede a una lista de servidores alrededor del mundo que ofrecen una hora precisa. Puesto que todos los nodos y el servidor acceden a la misma lista la fecha que reciban será igual para todos.
También se configura el servidor principal como servidor de NTP, así en caso de fallo de comunicación de las placas con el exterior es posible obtener la hora directamente del servidor.
Además, Slurm está programado específicamente para utilizar Munge como sistema de autenticación, por lo que también es un requisito.

slurmdbd es el daemon que utiliza Slurm para encargarse del accounting y la base de datos de envíos al clúster. Uno de los requisitos de la plataforma es que debe ser de alta disponibilidad, por lo que no se debe permitir que un usuario realice muchos envíos y monopolice el clúster. Para resolver esto Slurm implementa un sistema de prioridades que se pueden configurar, como el uso histórico del clúster por usuario.
Al configurar esta opción se dará diferentes prioridades a los envíos según la cantidad de tiempo que hayan utilizado el clúster recientemente.
Para controlar esto, se necesita guardar la información de todos los envíos según cada usuario. slurmdbd guarda esta información en una base de datos que crea y actualiza automáticamente.

Por último, slurmctld es el daemon que controla todos los envíos. Revisa que los nodos estén activos, se comunica con slurmdbd para pasarle datos sobre los *jobs*, y se comunica con los distintos slurmd que hay instalados en cada nodo de cálculo para saber cuándo están disponibles y se les puede asignar otra tarea.

Particiones

En el clúster se han configurado dos particiones, que se pueden ver en la figura 12.3.

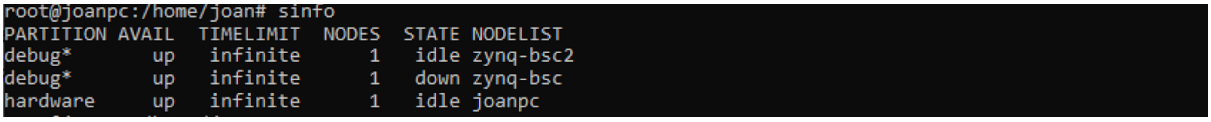


Figura 12.3. Particiones y nodos del clúster.

La partición *debug* es la predeterminada, y todas las placas se configuran como nodos de esta partición. Es la que se usa al realizar tanto envíos desde sbatch como desde la aplicación web.

hardware es la partición que se ha creado para enviar *jobs* relacionados con compilación hardware. Actualmente está configurado que se utilice el propio servidor, aunque no es viable utilizarlo para esta función, se debería adquirir una máquina específica para esta función.

12.1.3 Sistema de ficheros del clúster

Además de los daemon mencionados anteriormente, se ha decidido utilizar Network File System para montar un sistema de archivos compartidos.

Hay tres razones principales para esto:

-La necesidad de que todas las Zedboard tengan acceso a ciertos scripts y programas comunes.

En concreto se requiere acceso a DynamoRIO [20], los scripts necesarios para enviar código desde la página web e *input*, *output* y los tests específicos de cada laboratorio de PCA.
Todo esto está montado en el directorio */mnt/pca* para que todas las placas puedan acceder.

-Al ejecutar un código, se debe devolver al usuario los archivos de *output* y error que genera Slurm al capturar *stdout* y *stderr*. También se desea enviar cualquier archivo generado como resultado de la ejecución del código.
Para ello, se monta el */home* de cada usuario en las placas.

-Por último, el código que se ha subido a la página web también debe estar disponible en el nodo en que se vaya a ejecutar.

En la figura 12.4 se puede ver un esquema de estos directorios.

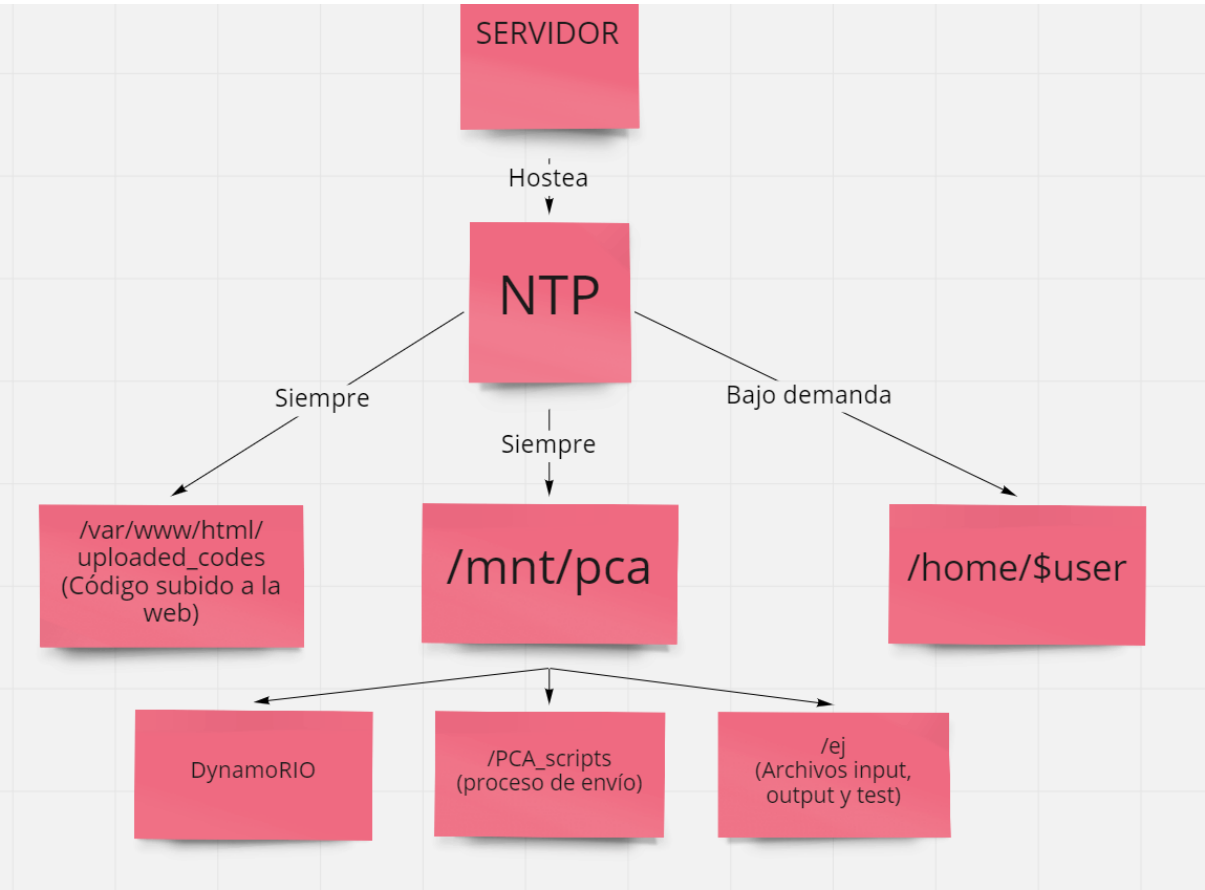


Figura 12.4. Esquema de directorios montados.

Uno de los riesgos de NFS es la sincronización, las operaciones de escritura deben ser mínimas para evitar saturar la red o las placas (el almacenamiento de las Zedboard es mediante una tarjeta SD, por lo que la velocidad de escritura puede ser un cuello de botella).

En el primer caso, el directorio */mnt/pca* solo se usa en lecturas, por lo que la sincronización es mínima.

Para resolver el conflicto del segundo caso, se ha decidido utilizar una función de Slurm, los *prologs* y *epilogs*, que permiten ejecutar cierto script antes y después de cada envío. Gracias a esto es posible montar y desmontar los directorios */home* de cada alumno sólo cuando esté ejecutándose un *job* o estén trabajando en interactivo desde una de las placas, lo que limita la sincronización.

Para finalizar, en */var/www/html/uploaded_codes* solo escribe el servidor cuando se realiza un envío a la página web, por lo que no es un problema.

12.2 Aplicación web

La función básica de la web es permitir al alumnado subir un código para un ejercicio en concreto y que este sea analizado. Tras ser analizado muestra una lista con los mejores envíos (es decir, para todos los envíos que superen todos los tests propuestos, se hace un ranking con los mejores tiempos de ejecución).

Esta es una medida para favorecer la gamificación [21]. Esta técnica consiste en aplicar a una tarea cualquiera elementos extraídos originalmente de los videojuegos para cautivar a los participantes. En el caso concreto de la educación, se busca explotar la motivación intrínseca del alumnado, es decir, el deseo del propio alumno para aprender por su cuenta. En última instancia, al aplicar gamificación de forma correcta -pues una buena gamificación debe venir dada por la experiencia del profesorado- se consigue que el alumno se divierta mientras aprende, lo que le lleva a querer aprender más para seguir pasándoselo bien. De las distintas funciones que se esperaban recuperar con esta plataforma, esta es de las más importancia se le ha dado.

En la aplicación web se utiliza un sistema de ranking y retos similar al utilizado en el antiguo servidor de pca, ya que el profesorado afirma que daba buenos resultados; los alumnos se esforzaban en mejorar el código para competir entre ellos.

También se añade un sistema de medallas: iconos que se otorgan como si fueran premios a cada envío.

Además ofrecen una forma muy visual de ver cómo ha ido el envío: sirven para indicar si la compilación de un código es correcta, si la salida del programa es la esperada…

Por último, para dar *feedback* al estudiante, el profesorado puede utilizar el sistema de medallas mencionado o una función para añadir comentarios que se ha implementado.

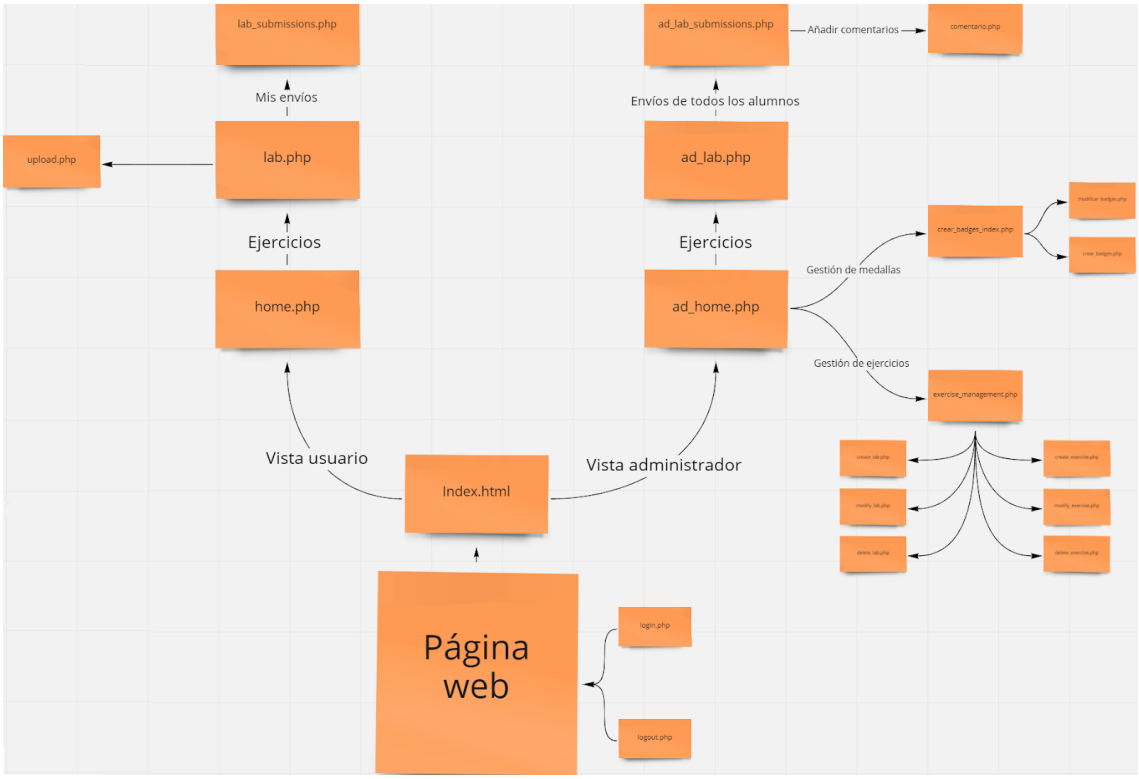


Figura 12.5. Vista general de la aplicación web.

La página tiene dos vistas principales, una para los usuarios y otra para el administrador.

Durante la elaboración de la aplicación web se han tomado precauciones para proteger la página de usos malintencionados, estas medidas se detallan en el Apéndice C: Seguridad.

En la figura 12.6 se puede observar los archivos que componen la parte de usuario, que se explica en detalle en el siguiente apartado.

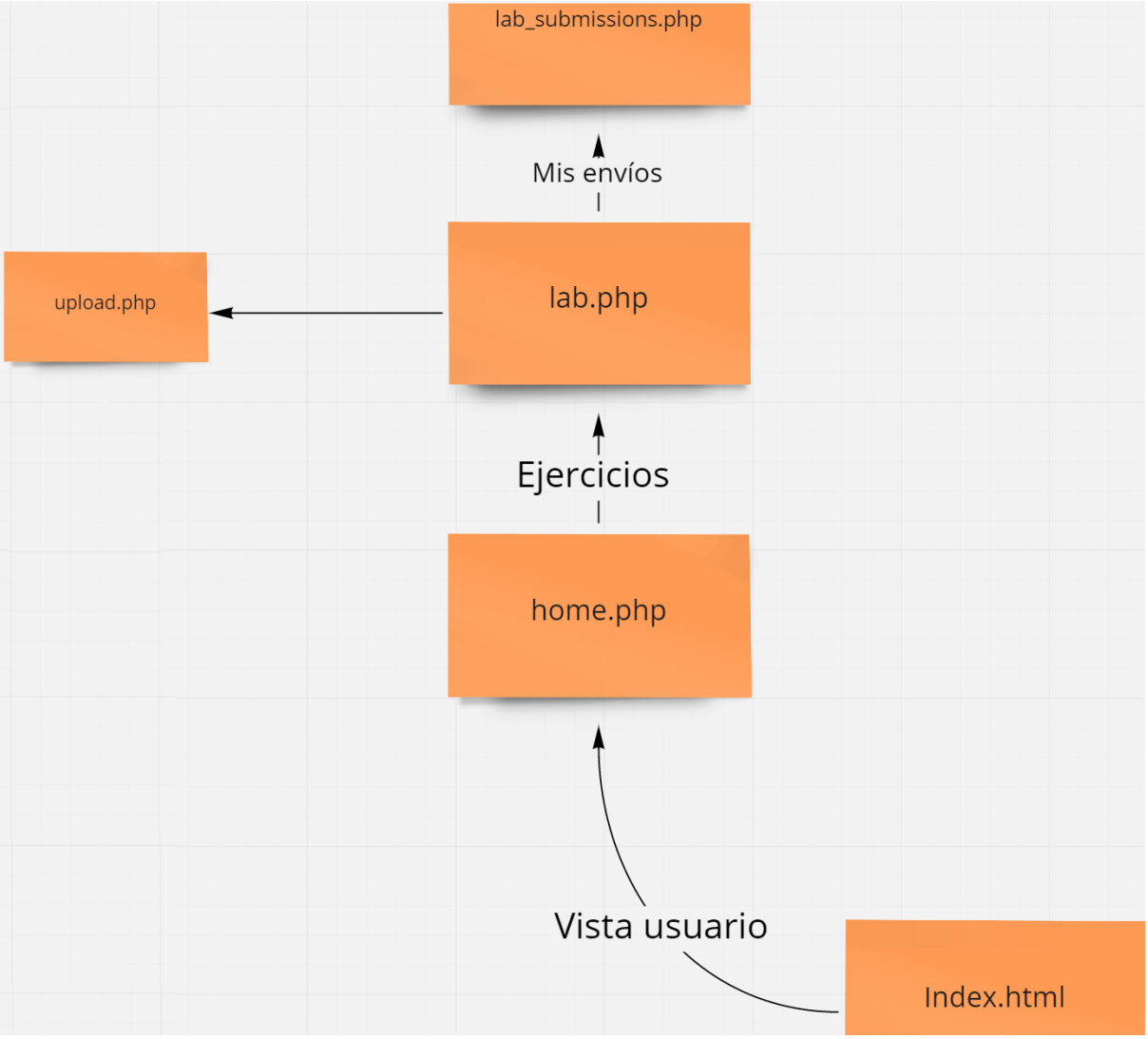


Figura 12.6. Estructura de la parte de usuario.

12.2.1 Vista de usuario

Tras el login, los usuarios son redirigidos a una página como la de la figura 12.7.

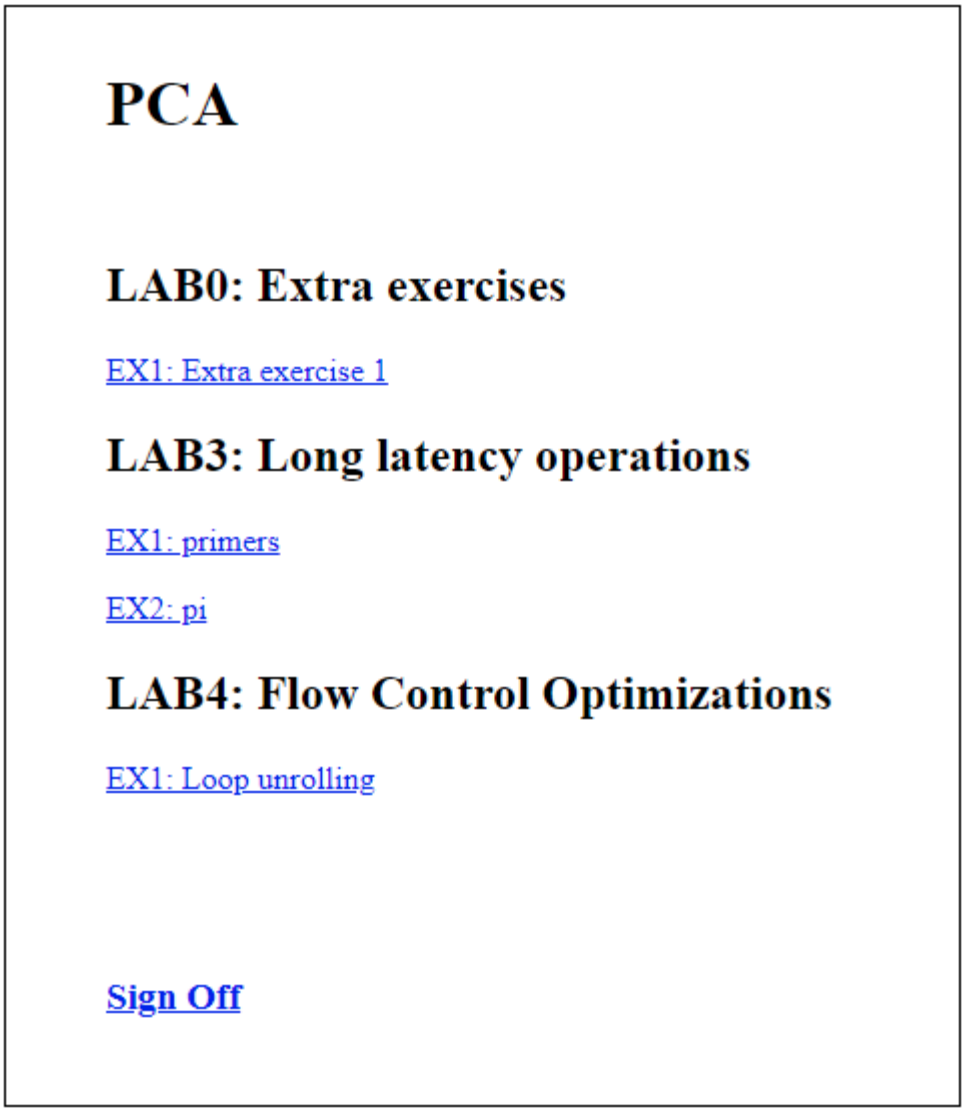
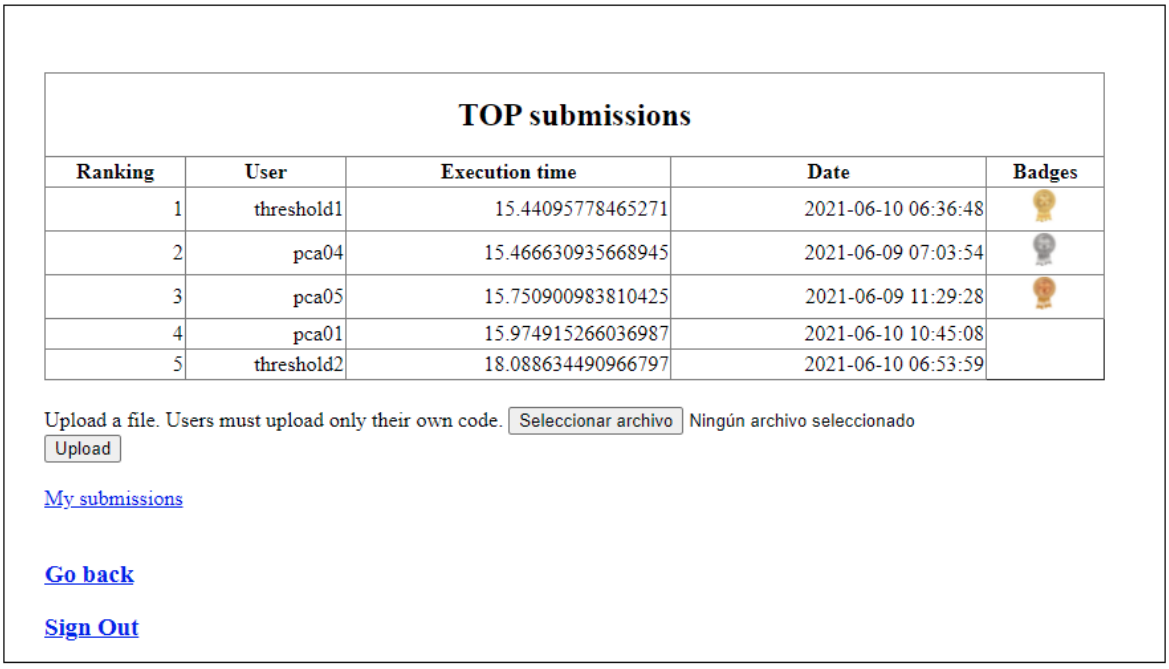


Figura 12.7. Home de los usuarios.

Aquí se generan de forma dinámica las entradas y enlaces a los ejercicios y laboratorios definidos en la base de datos.

Cada ejercicio tiene un apartado como el de la figura 12.8.



TOP submissions

Ranking	User	Execution time	Date	Badges
1	threshold1	15.44095778465271	2021-06-10 06:36:48	
2	pca04	15.466630935668945	2021-06-09 07:03:54	
3	pca05	15.750900983810425	2021-06-09 11:29:28	
4	pca01	15.974915266036987	2021-06-10 10:45:08	
5	threshold2	18.088634490966797	2021-06-10 06:53:59	

Upload a file. Users must upload only their own code. Seleccionar archivo Ningún archivo seleccionado
Upload

My submissions

Go back

Sign Out

Figura 12.8. Vista de un ejercicio.

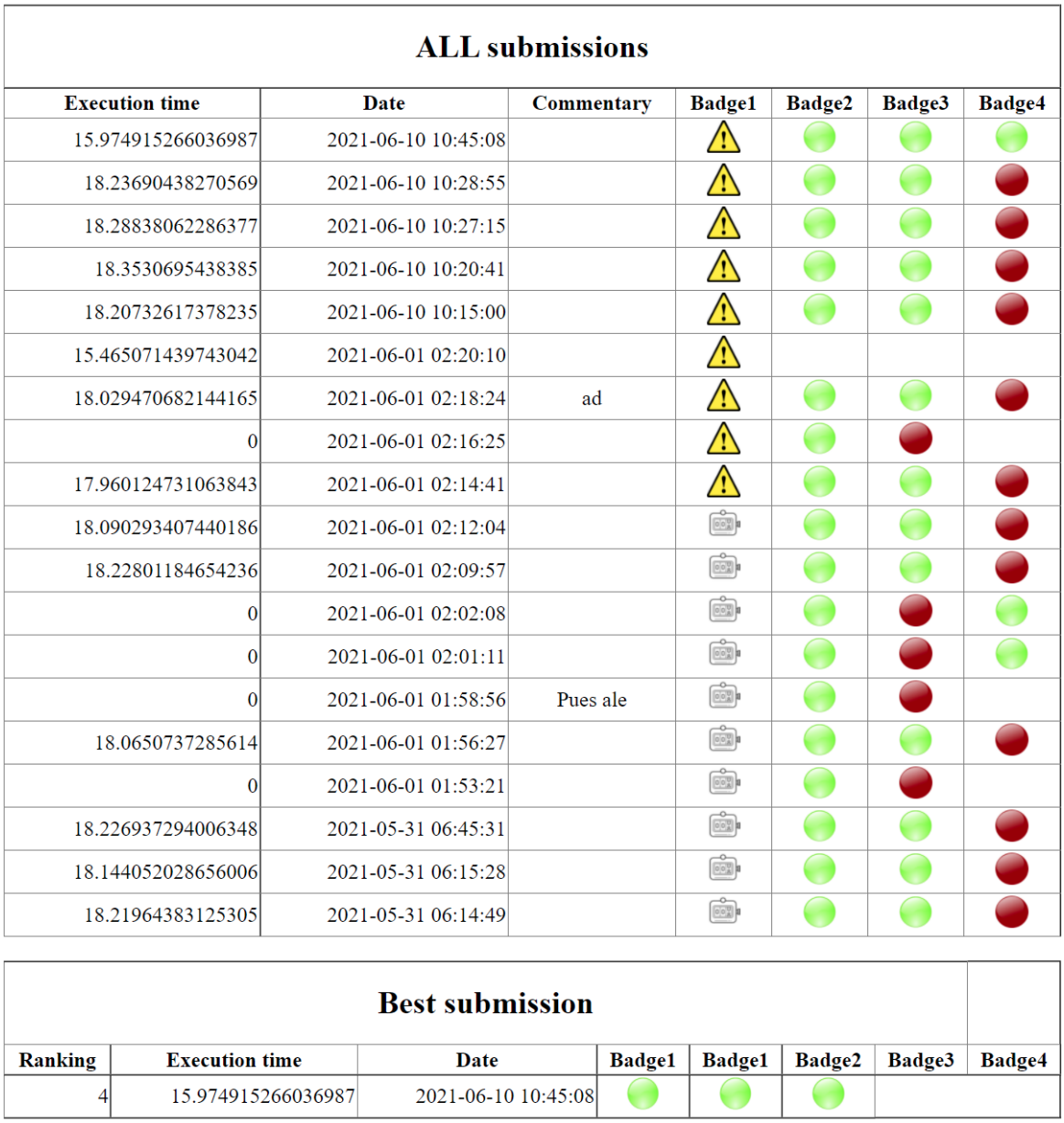
En esta página se muestra el ranking con los mejores envíos de cada usuario. Solo se muestran aquellos que han pasado todos los test específicos de cada laboratorio. De esta forma aunque algún alumno haga alguna optimización al código que no esté prevista si no ha realizado las que se esperaban, es decir, no ha consolidado los conocimientos evaluados, no entra al ranking.

Hay dos usuarios de administrador, threshold1 y threshold2 que se mostrarán siempre que su tiempo de ejecución sea distinto de 0, para que así el profesorado pueda marcar ciertos límites que perseguir. Por ejemplo, puede subir un código con threshold1 que no pase cierto test, así en cuanto los alumnos lo logren se posicionarán por encima.

Además desde esta página es posible subir el código para analizar, el proceso se comenta más adelante.

Todo el contenido de estas páginas se obtiene mediante *querys* a la base de datos. Estas se han optimizado para no saturar la base de datos con muchas *mini-querys*.
Hay más información sobre cómo se realizan en el Apéndice B: Código.

Por último, hay un enlace a *my submissions*, que muestra un contenido como el de la figura 12.9.



ALL submissions

Execution time	Date	Commentary	Badge1	Badge2	Badge3	Badge4
15.974915266036987	2021-06-10 10:45:08					
18.23690438270569	2021-06-10 10:28:55					
18.28838062286377	2021-06-10 10:27:15					
18.3530695438385	2021-06-10 10:20:41					
18.20732617378235	2021-06-10 10:15:00					
15.465071439743042	2021-06-01 02:20:10					
18.029470682144165	2021-06-01 02:18:24	ad				
0	2021-06-01 02:16:25					
17.960124731063843	2021-06-01 02:14:41					
18.090293407440186	2021-06-01 02:12:04					
18.22801184654236	2021-06-01 02:09:57					
0	2021-06-01 02:02:08					
0	2021-06-01 02:01:11					
0	2021-06-01 01:58:56	Pues ale				
18.0650737285614	2021-06-01 01:56:27					
0	2021-06-01 01:53:21					
18.226937294006348	2021-05-31 06:45:31					
18.144052028656006	2021-05-31 06:15:28					
18.21964383125305	2021-05-31 06:14:49					

Best submission

Ranking	Execution time	Date	Badge1	Badge1	Badge2	Badge3	Badge4
4	15.974915266036987	2021-06-10 10:45:08					

Figura 12.9. Vista de my submissions.

En esta página se muestran todos los envíos del alumno, junto con el mejor de ellos (si es que hay alguno apto para que se clasifique).

En la tabla se enseña el tiempo de ejecución de cada envío, la fecha, los comentarios que ha añadido el profesorado y las diferentes medallas asignadas.

12.2.2 Vista de administrador

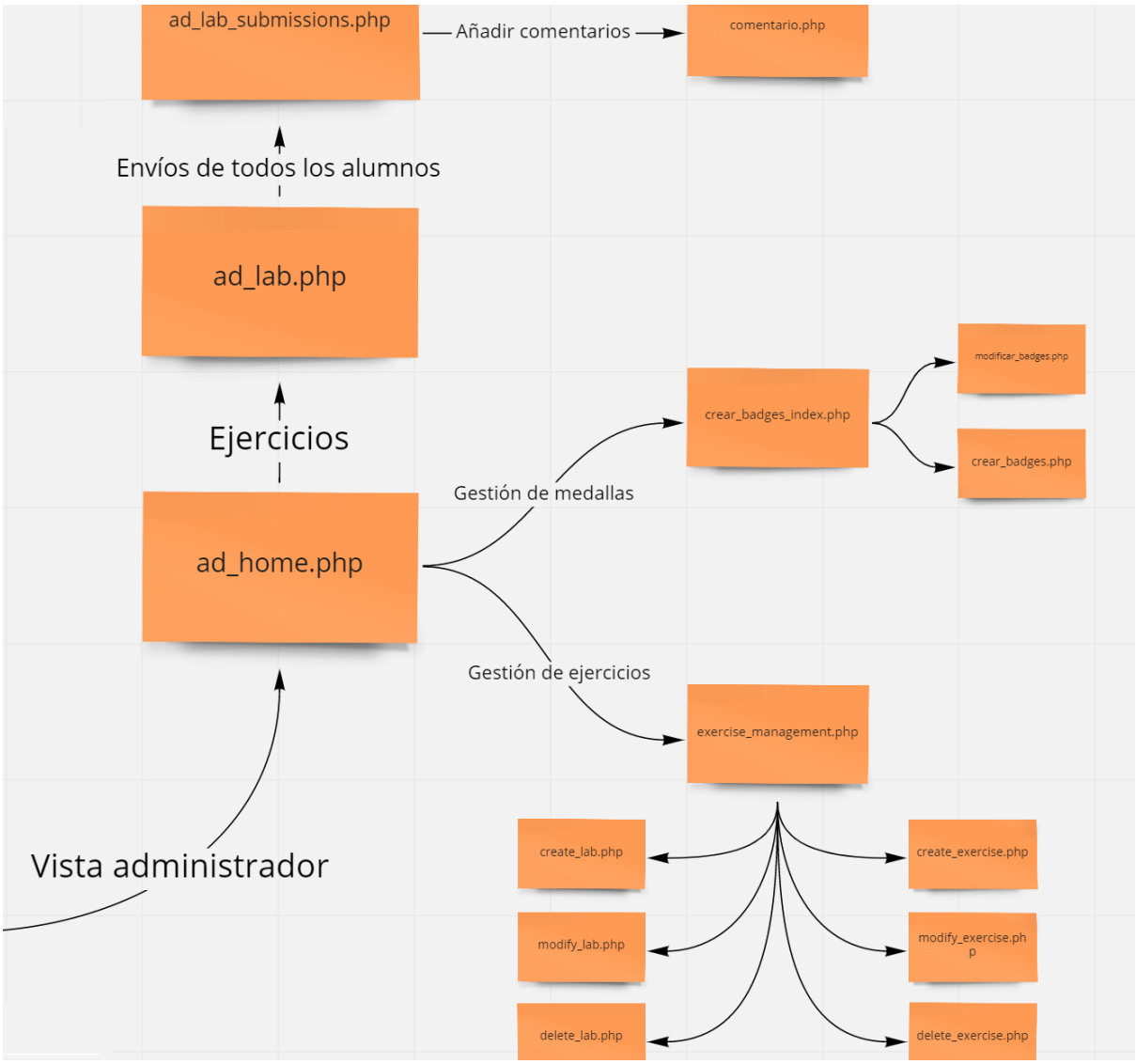


Figura 12.10. Parte de administrador.

Al hacer login como administrador, se redirige a otro home distinto, con enlaces a dos herramientas creadas: la gestión de medallas y de ejercicios.

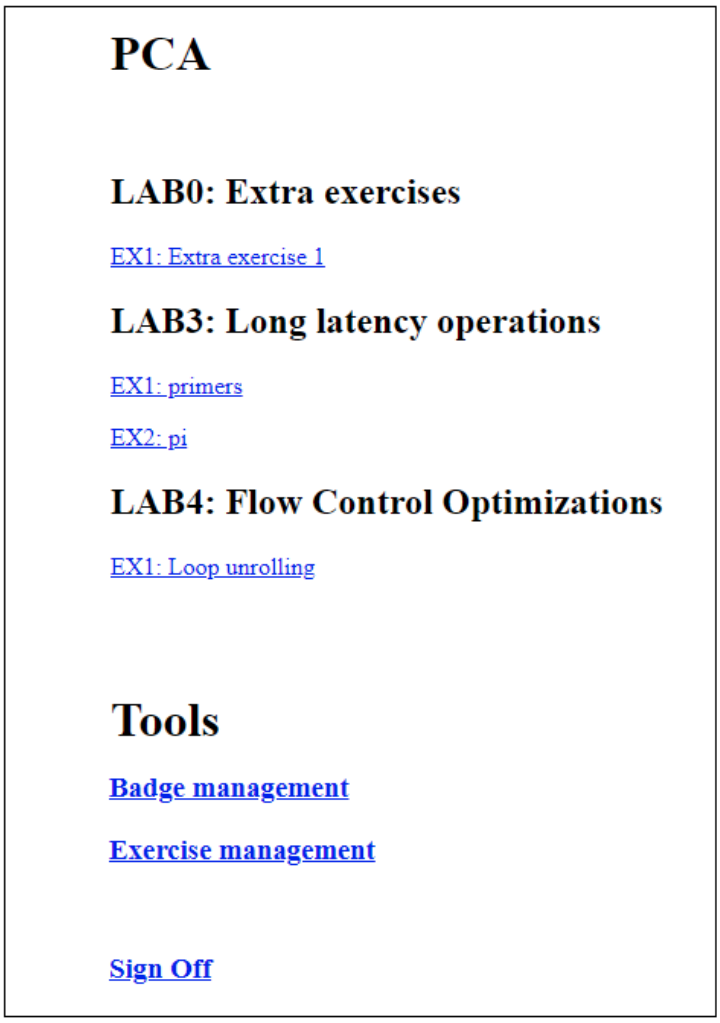


Figura 12.11. Home de administrador.

Si se accede a un laboratorio, la vista es idéntica a la que tendría un alumno, aunque el archivo php es distinto. Esto se ha hecho por si en algún momento se quiere añadir alguna función al administrador que no tengan los usuarios sea más sencillo implementarlo de forma independiente.

La página *my submissions* es reemplazada por *student submissions*, donde el profesorado puede ver todos los envíos que han hecho todos los alumnos, para poder comprobar rápidamente su trabajo.

En la figura 12.12 se puede ver un ejemplo.

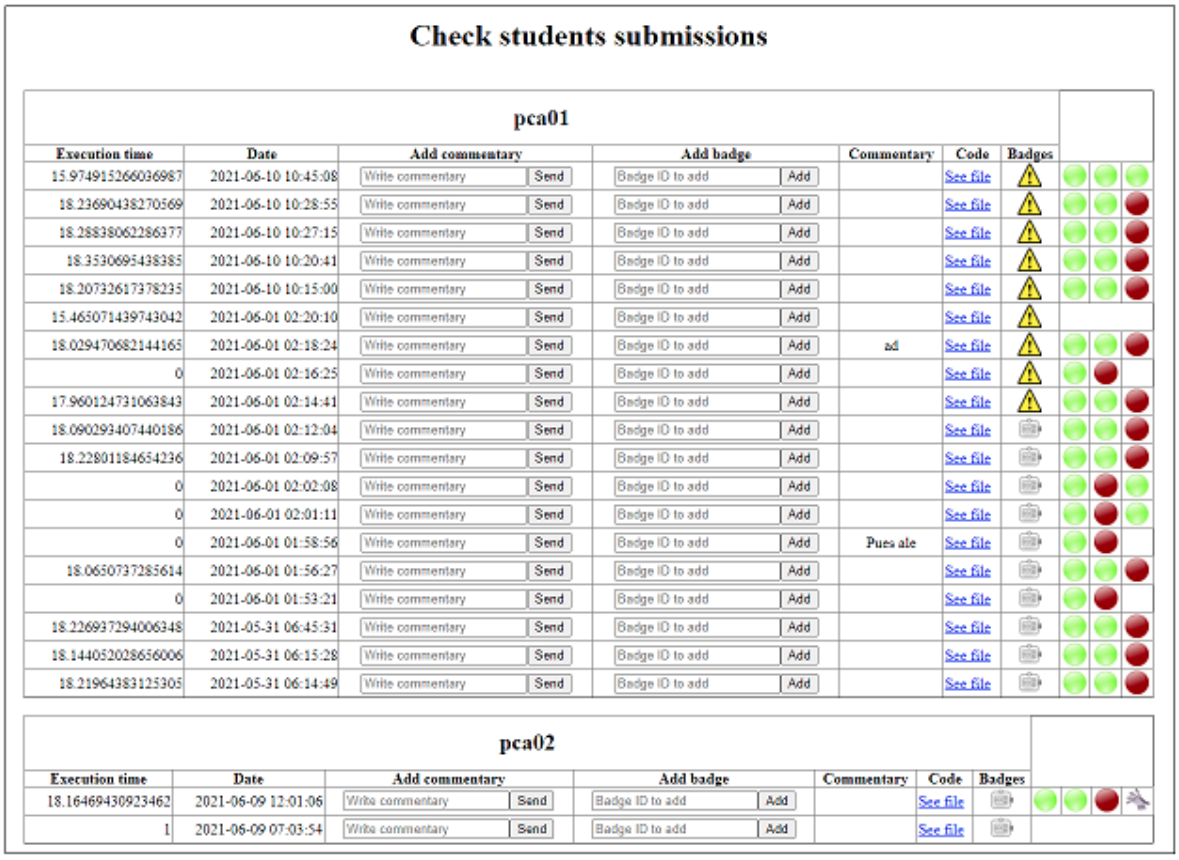


Figura 12.12. Vista de student submissions.

Por último, en las figuras 12.13 y 12.17 se pueden ver las páginas de las herramientas mencionadas.

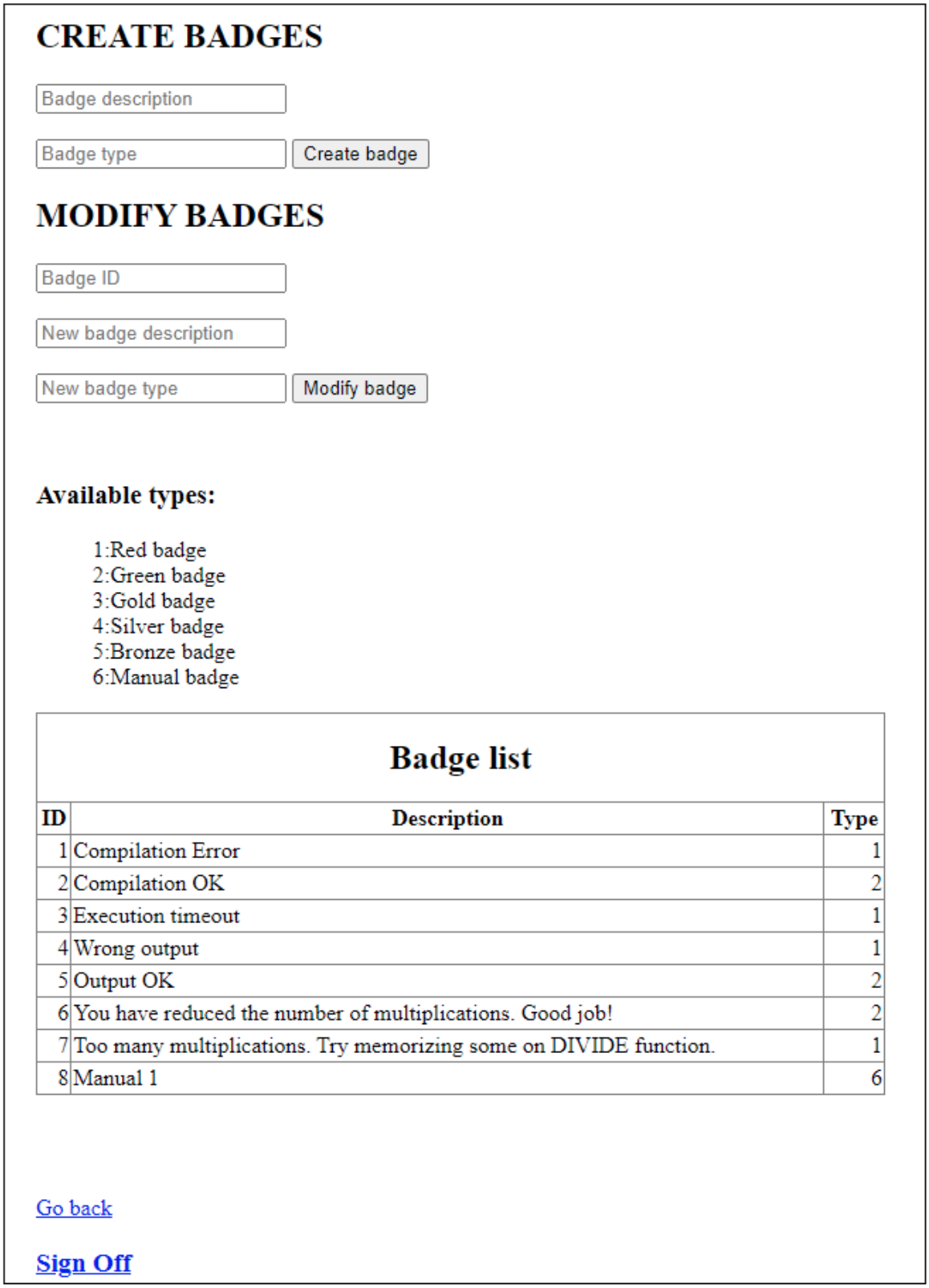


Figura 12.13. Herramienta de gestión de medallas.

Como se ha comentado anteriormente, las medallas son iconos que se asignan a un envío. Están compuestas por una descripción y un tipo.

La descripción es útil para enviar un mensaje generado automáticamente. El mensaje puede ser algo sencillo como un indicativo de cómo ha ido el envío (por ejemplo "output OK" para indicar que la salida del programa era la esperada) o mensaje más complejos como indicar al alumno que haga cierto ejercicio para aclarar mejor los conceptos que no ha aplicado bien.
Al pasar el ratón por encima se muestra su descripción, figura 12.14.

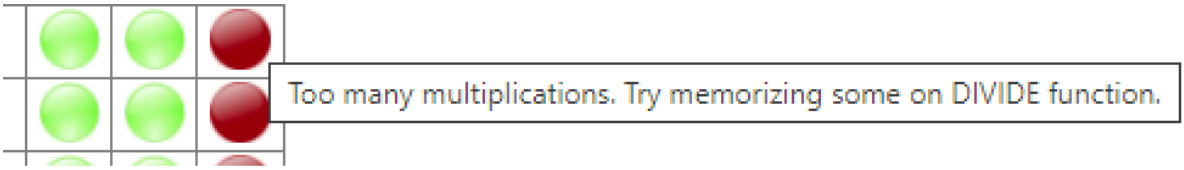


Figura 12.14. Descripción de una medalla.

El tipo sirve para especificar la imagen que tendrá esa medalla asociada. En la figura 12.15 se muestran los tipos definidos.

Figura 12.15. Tipos de medallas.

Conceptualmente hay 3 categorías de medallas:

- Fijas: Están definidas para todos los laboratorios, son las que indican si la compilación es correcta, si la ejecución ha acabado en *timeout* y si el resultado del programa es correcto.
- Variables: Se otorgan según los tests específicos de cada laboratorio. Indican información sobre las optimizaciones que se esperaban para ese ejercicio.
- Manuales: No corresponden a ningún laboratorio, el profesorado es libre de crearlas y otorgarlas según considere. Pueden utilizarse por ejemplo para premiar una optimización que no se pedía.

Dentro de la categoría de medallas fijas está incluida una medalla de penalización. Se ha añadido para evitar que el alumnado intente subir código repetidamente sin haberlo probado antes.

La penalización consiste en añadir al tiempo de ejecución unos segundos, de forma que la posición en el ranking se verá afectada. En la figura 12.16 se puede ver un ejemplo.

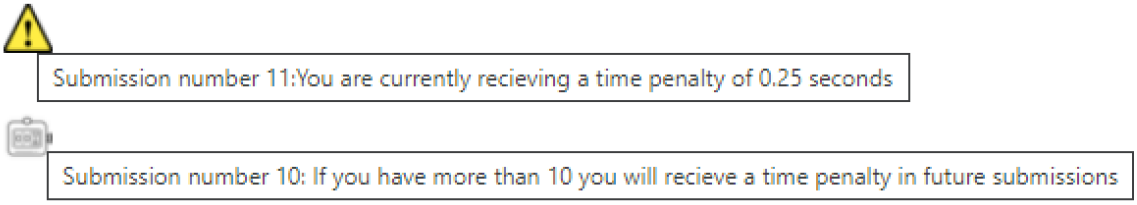


Figura 12.16. Medalla de penalización.

Desde la herramienta de gestión de ejercicios de la figura 12.17 se puede configurar cuántos envíos es posible realizar antes de aplicar la penalización, y la magnitud de la misma.

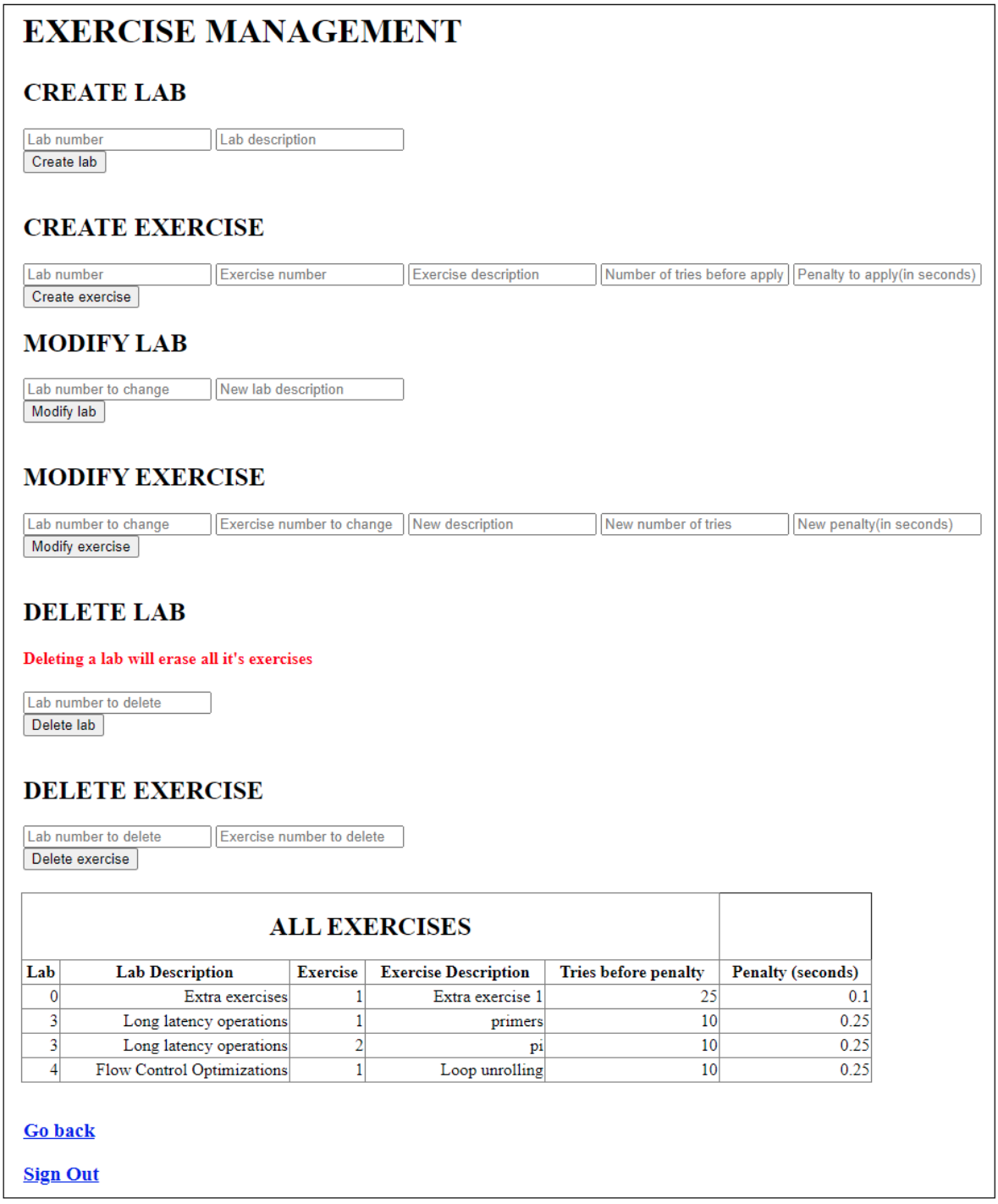


Figura 12.17. Herramienta de gestión de ejercicios.

Estas dos herramientas presentan formularios para añadir, modificar o eliminar elementos de la base de datos directamente.
Cada formulario ejecuta un código en php que comprueba que la entrada del administrador sea correcta y, en caso de que lo sea, realiza la consulta adecuada.

12.3 Base de datos

La base de datos se encarga de guardar toda la información relevante a los envíos y la aplicación web.

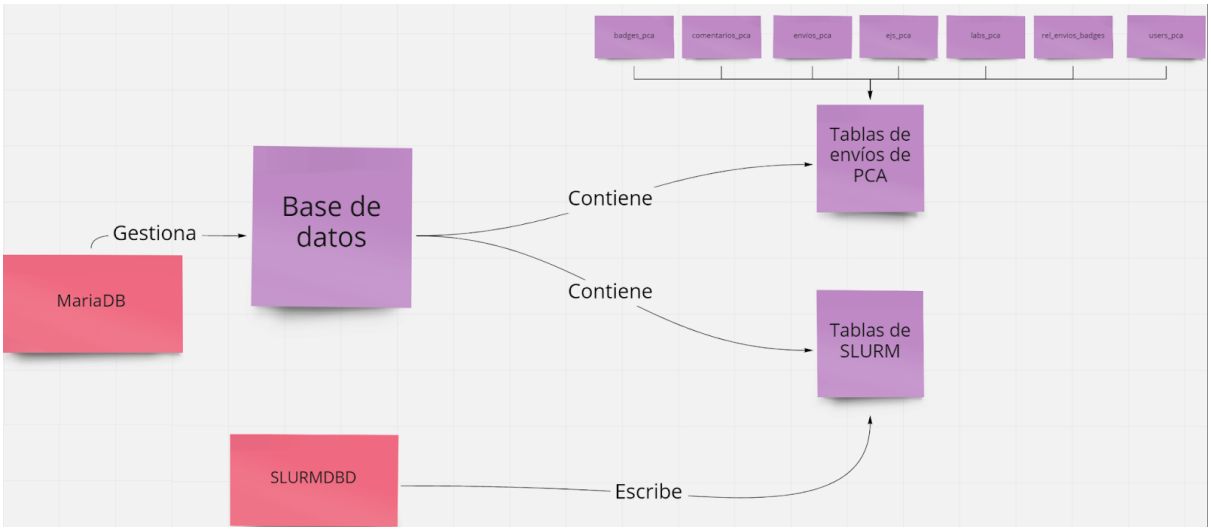


Figura 12.18. Esquema de la base de datos.

En el servidor se utilizan las siguientes bases separadas:

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| envios_pca         |
| information_schema |
| mysql              |
| performance_schema |
| slurm_acct_db      |
+--------------------+
```

Figura 12.19. Distintas bases que se incluyen.

information_schema, mysql y performance_schema son bases que utiliza MariaDB para el propio funcionamiento.

slurm_acct_db es la base que utiliza slurmdbd para guardar la información sobre todos los envíos al clúster y calcular las prioridades.
En la figura 12.20 se pueden observar todas las tablas utilizadas.
Se crean, purgan y actualizan automáticamente.

```
+---------------------------------+
| Tables_in_slurm_acct_db         |
+---------------------------------+
| acct_coord_table                |
| acct_table                      |
| clus_res_table                  |
| cluster_assoc_table             |
| cluster_assoc_usage_day_table   |
| cluster_assoc_usage_hour_table  |
| cluster_assoc_usage_month_table |
| cluster_event_table             |
| cluster_job_table               |
| cluster_last_ran_table          |
| cluster_resv_table              |
| cluster_step_table              |
| cluster_suspend_table           |
| cluster_table                   |
| cluster_usage_day_table         |
| cluster_usage_hour_table        |
| cluster_usage_month_table       |
| cluster_wckey_table             |
| cluster_wckey_usage_day_table   |
| cluster_wckey_usage_hour_table  |
| cluster_wckey_usage_month_table |
| convert_version_table           |
| federation_table                |
| qos_table                       |
| res_table                       |
| table_defs_table                |
| tres_table                      |
| txn_table                       |
| user_table                      |
+---------------------------------+
```

Figura 12.20. Tablas de slurm_acct_db.

La base envios_pca se ha creado específicamente para guardar toda la información de la aplicación web. Contiene las tablas de la figura 12.21.

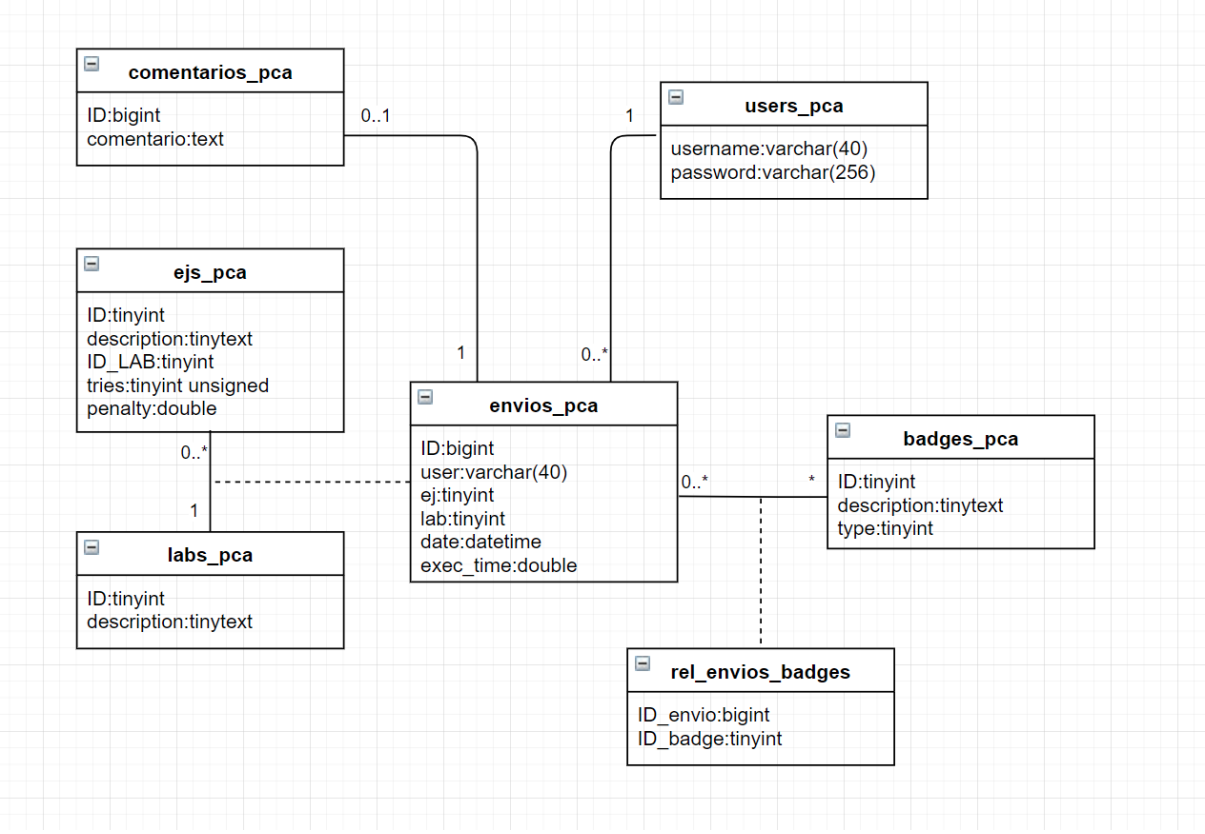


Figura 12.21. Diagrama de especificación de envios_pca.

12.3.1 users_pca

```
MariaDB [envios_pca]> SHOW COLUMNS FROM users_pca;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| username | varchar(40)  | NO   | PRI | NULL    |       |
| password | varchar(256) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
```

Figura 12.22. users_pca.

Tabla que contiene información sobre los usuarios que pueden hacer login a la página web. La contraseña se guarda *hasheada* [22] con la función *crypt_blowfish* de php, que garantiza compatibilidad con el *crypt* que utiliza Linux de forma predeterminada. Esta función añade un *salt* [23], lo que refuerza la seguridad de las contraseñas.

```
+------------+--------------------------------------------------------------+
| username   | password                                                     |
+------------+--------------------------------------------------------------+
| admin      | $2y$10$vv04KByfxjq2r1OHRrM9jOu2/XwytUYHcHUZ54ujqV0EIvarg83fy |
| pca01      | $2y$10$ZTMrhA.tFRkVcYHvamStl.2Pg2WE8RaAbIIegROL1X8KeO8Uiuj7e |
| pca02      | $2y$10$zUQHEoHQbuLnZHPIJNlMWu1tb92h4SLQlbXkCK26GR67siZ6oX17K |
| pca03      | $2y$10$SOMUaRkSboadK9iy.DKdguhnrVEydDf3HKBkfmCPkkxnuYcwzIR0u |
| pca04      | $2y$10$atZ8TdCsMxHps60f7yDLT.tn/mske3/4ge4ehIsfHRMqfoavl39jG |
| pca05      | $2y$10$lktUBHNV2DfHMHdRzxxzAeCO8tu8ySadKjopaiCDm7y2VqPHU2ZQe |
| threshold1 | $2y$10$A1PgxzqIkdOmVIvfu4R0Le6bOP0J2B99xt7eAxG3vHXxw6wMYD5Ee |
| threshold2 | $2y$10$VLujy4/ZQVPgZDjMROMokeBvCgCarGBfQxZu1K9U2fZSVl2uSVZXe |
+------------+--------------------------------------------------------------+
```

Figura 12.23. Ejemplo del contenido de users_pca.

12.3.2 labs_pca

```
MariaDB [envios_pca]> SHOW COLUMNS FROM labs_pca;
+-------------+------------+------+-----+---------+-------+
| Field       | Type       | Null | Key | Default | Extra |
+-------------+------------+------+-----+---------+-------+
| ID          | tinyint(4) | NO   | PRI | NULL    |       |
| description | tinytext   | YES  |     | NULL    |       |
+-------------+------------+------+-----+---------+-------+
```

Figura 12.24. labs_pca.

Esta tabla contiene una lista de los laboratorios de PCA que se hayan definido. Un laboratorio contiene una ID (que es el número del laboratorio) y una descripción, que puede ser el nombre.

```
MariaDB [envios_pca]> SELECT * FROM labs_pca;
+----+----------------------------+
| ID | description                |
+----+----------------------------+
|  0 | Extra exercises            |
|  3 | Long latency operations    |
|  4 | Flow Control Optimizations |
+----+----------------------------+
```

Figura 12.25. Ejemplo del contenido de labs_pca.

Se ha añadido un laboratorio con la ID 0 que representa ejercicios extra. Aquí se pueden introducir ejercicios que sirvan de repaso o ampliación a algunos temas.

12.3.3 ejs_pca

```
MariaDB [envios_pca]> SHOW COLUMNS FROM ejs_pca;
+-------------+---------------------+------+-----+---------+-------+
| Field       | Type                | Null | Key | Default | Extra |
+-------------+---------------------+------+-----+---------+-------+
| ID          | tinyint(4)          | YES  |     | NULL    |       |
| description | tinytext            | YES  |     | NULL    |       |
| ID_LAB      | tinyint(4)          | YES  | MUL | NULL    |       |
| tries       | tinyint(3) unsigned | YES  |     | NULL    |       |
| penalty     | double              | YES  |     | NULL    |       |
+-------------+---------------------+------+-----+---------+-------+
```

Figura 12.26. ejs_pca.

Esta tabla incluye todos los ejercicios de la asignatura. Deben estar relacionados con un laboratorio, y para garantizar la integridad de la base de datos se ha definido una *foreign key* como restricción.
Por cada ejercicio se puede indicar una descripción, una ID (que es el número de ejercicio) y los valores *tries* y *penalty*.
El atributo *tries* especifica el número máximo de intentos que un alumno puede enviar a la página web antes de aplicar una penalización al tiempo de ejecución. En *penalty* se especifica esta penalización en segundos.

```
MariaDB [envios_pca]> SELECT * FROM ejs_pca;
+------+------------------+--------+-------+---------+
| ID   | description      | ID_LAB | tries | penalty |
+------+------------------+--------+-------+---------+
|    1 | primers          |      3 |    10 |    0.25 |
|    2 | pi               |      3 |    10 |    0.25 |
|    1 | Loop unrolling   |      4 |    10 |    0.25 |
|    1 | Extra exercise 1 |      0 |    25 |     0.1 |
+------+------------------+--------+-------+---------+
```

Figura 12.27. Ejemplo del contenido de ejs_pca.

12.3.4 comentarios_pca

```
MariaDB [envios_pca]> SHOW COLUMNS FROM comentarios_pca;
+------------+------------+------+-----+---------+-------+
| Field      | Type       | Null | Key | Default | Extra |
+------------+------------+------+-----+---------+-------+
| ID         | bigint(20) | YES  |     | NULL    |       |
| comentario | text       | YES  |     | NULL    |       |
+------------+------------+------+-----+---------+-------+
```

Figura 12.28. comentarios_pca.

Esta tabla relaciona cada envío a la página web con un comentario (si existe) que haya añadido el profesorado.

```
MariaDB [envios_pca]> SELECT * FROM comentarios_pca;
+------+-----------------------------------------------------+
| ID   | comentario                                          |
+------+-----------------------------------------------------+
|   40 | Prueba exec=18                                      |
|   41 | Good                                                |
```

Figura 12.29. Ejemplo del contenido de comentarios_pca.

12.3.5 badges_pca

```
MariaDB [envios_pca]> SHOW COLUMNS FROM badges_pca;
+-------------+------------+------+-----+---------+----------------+
| Field       | Type       | Null | Key | Default | Extra          |
+-------------+------------+------+-----+---------+----------------+
| ID          | tinyint(4) | NO   | PRI | NULL    | auto_increment |
| description | tinytext   | YES  |     | NULL    |                |
| type        | tinyint(4) | YES  |     | NULL    |                |
+-------------+------------+------+-----+---------+----------------+
```

Figura 12.30. badges_pca.

Esta tabla recoge todas las medallas.

Los campos son ID con *auto_increment*, que sirve para referirse a ellas, una descripción y un tipo.

Un ejemplo del contenido de esta tabla es el de la figura 12.31.

```
MariaDB [envios_pca]> SELECT * FROM badges_pca;
+----+-------------------------------------------------------------------+------+
| ID | description                                                       | type |
+----+-------------------------------------------------------------------+------+
|  1 | Compilation Error                                                 |    1 |
|  2 | Compilation OK                                                    |    2 |
|  3 | Execution timeout                                                 |    1 |
|  4 | Wrong output                                                      |    1 |
|  5 | Output OK                                                         |    2 |
|  6 | You have reduced the number of multiplications. Good job!         |    2 |
|  7 | Too many multiplications. Try memorizing some on DIVIDE function. |    1 |
|  8 | Manual 1                                                          |    6 |
+----+-------------------------------------------------------------------+------+
```

Figura 12.31. Ejemplo del contenido de badges_pca.

12.3.6 envios_pca

```
MariaDB [envios_pca]> SHOW COLUMNS FROM envios_pca;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| ID        | bigint(20)  | NO   | PRI | NULL    | auto_increment |
| user      | varchar(40) | YES  |     | NULL    |                |
| ej        | tinyint(4)  | YES  |     | NULL    |                |
| lab       | tinyint(4)  | YES  |     | NULL    |                |
| date      | datetime    | YES  |     | NULL    |                |
| exec_time | double      | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
```

Figura 12.32. envios_pca.

Esta tabla contiene todos los envíos que se envían a la aplicación web.
Los campos son el usuario que ha realizado el envío, el laboratorio y ejercicio para el que se ha hecho dicho envío, la fecha en la que se ha empezado a ejecutar el código y el tiempo total de ejecución que ha tardado el código (o 0 si el código no compila o el *output* no es el esperado).

La ID de envío se ha definido como un *BIGINT* con *auto_increment*.
Los *BIGINT* en MariaDB permiten representar hasta el número 9223372036854775807, que corresponde con el máximo número que puede representar un entero en php, por lo que este campo es compatible con la interpretación que se hace en la página web.

La decisión de definirlo de esta forma es para cerciorarse de que no se va a llegar al límite de envíos en un período de tiempo aceptable.

ID	user	ej	lab	date	exec_time
16	pca04	2	3	2021-05-31 04:58:19	18.062357425689697
18	pca01	2	3	2021-05-31 06:14:49	18.21964383125305
19	pca01	2	3	2021-05-31 06:15:28	18.144052028656006
20	pca01	2	3	2021-05-31 06:45:31	18.226937294006348
21	pca04	2	3	2021-05-31 06:57:23	18.2834312915802
22	pca05	2	3	2021-05-31 08:21:34	18.054904222488403
23	pca01	2	3	2021-06-01 01:53:21	0

Figura 12.33. Ejemplo del contenido de envios_pca.

12.3.7 rel_envios_badges

```
MariaDB [envios_pca]> SHOW COLUMNS FROM rel_envios_badges;
+----------+------------+------+-----+---------+-------+
| Field    | Type       | Null | Key | Default | Extra |
+----------+------------+------+-----+---------+-------+
| ID_envio | bigint(20) | YES  |     | NULL    |       |
| ID_badge | tinyint(4) | YES  |     | NULL    |       |
+----------+------------+------+-----+---------+-------+
```

Figura 12.34. rel_envios_badges.

Esta última tabla de la base de datos relaciona un envío con todas las medallas otorgadas a esa ejecución.
Se ha decidido utilizar una tabla aparte para hacer esta relación debido a que el número de medallas que puede tener un ejercicio, o incluso un envío, es dinámica, y en cualquier momento puede verse alterada (en parte consecuencia de la existencia de las medallas que se pueden dar de forma manual).

```
MariaDB [envios_pca]> SELECT * FROM rel_envios_badges;
+----------+----------+
| ID_envio | ID_badge |
+----------+----------+
|       16 |        2 |
|       16 |        5 |
|       16 |        7 |
|       17 |        2 |
|       17 |        5 |
|       17 |        7 |
|       18 |        2 |
```

Figura 12.35. Ejemplo del contenido de rel_envios_badges.

12.4 Proceso de envío y análisis de código

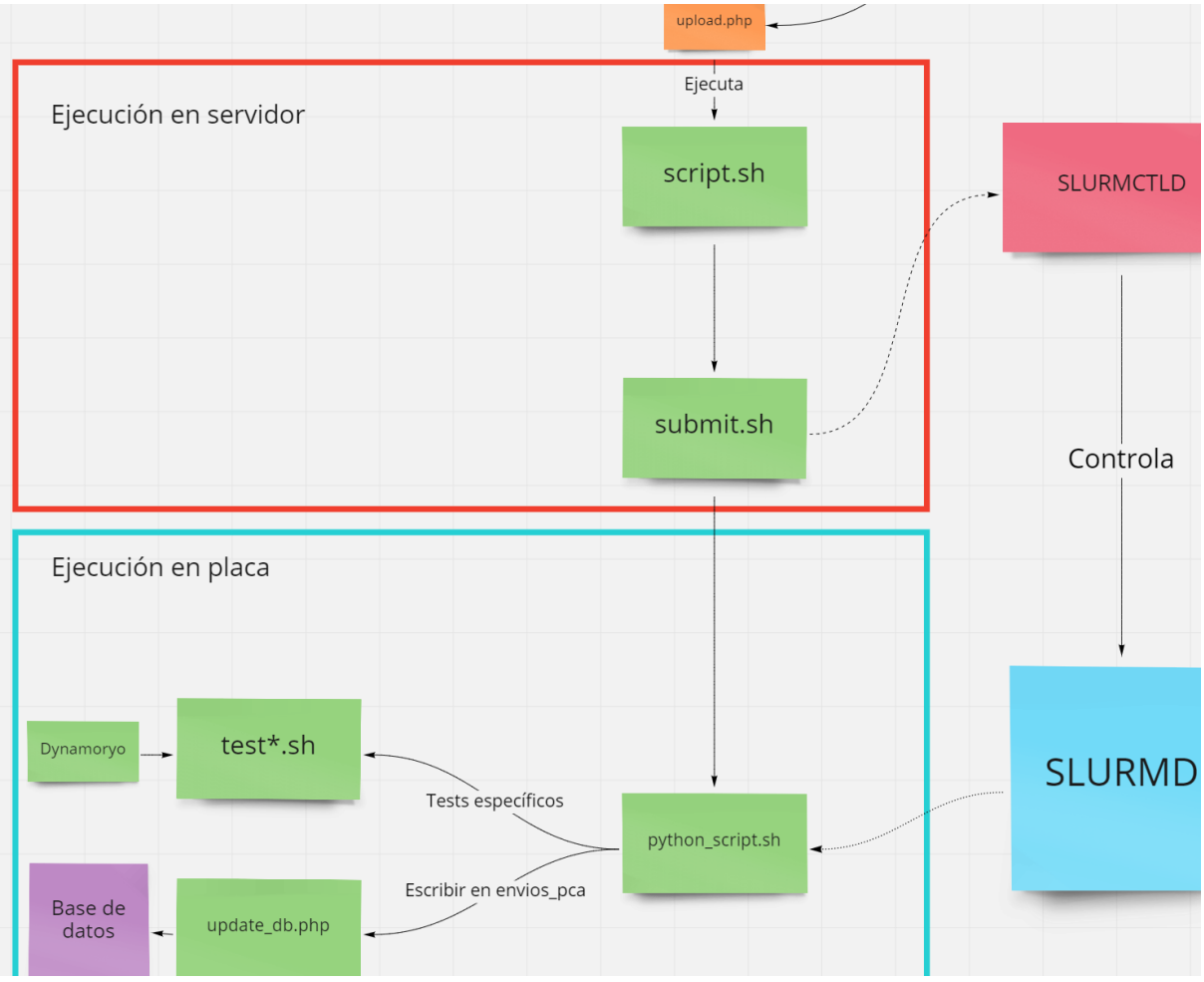


Figura 12.36. Proceso de envío de código desde la web.

Por último, si bien no se considera un módulo del sistema, tiene sentido conceptualmente tratar el proceso de envío aparte, ya que por su complejidad requiere del resto de módulos.

Cuando un alumno sube un código a la página web se ejecuta *upload.php*. Tras comprobar que el archivo es un '.c' se crea una copia del código en el directorio temporal */mnt/pca/tmp*. Al enviar el código al clúster la ejecución de *upload.php* acabará y el código subido será eliminado, independientemente de si se ha llegado a ejecutar en el clúster o no. Por eso se realiza esta copia, para poder acceder al código independientemente del momento en que el clúster esté libre.

Para enviar el código al clúster, desde *upload.php* se ejecuta un script.
El usuario de la aplicación web, www-data, tiene permisos de sudo sólo para ejecutar este script. De esta forma se puede evitar tener que darle permisos de sudo totales al usuario de la aplicación web y crear una vulnerabilidad.

Este último script es importante ejecutarlo como sudo ya que es necesario llamar a *submit.sh* (que es el script que enviará el código al gestor del clúster) como un usuario que tenga permisos para ver */mnt/pca*, directorio donde se guardan los scripts, tests, *input* y *output* esperado de cada ejercicio. El usuario designado para esta tarea es threshold1.

Una vez se ha enviado al clúster, cuando un nodo esté disponible se ejecutará el código enviado, ya en una de las Zedboard.
Esta es una ejecución controlada, pues viene dada por *python_script.sh*.

Un diagrama de este proceso de cambio de usuario se puede ver en la figura 12.37.

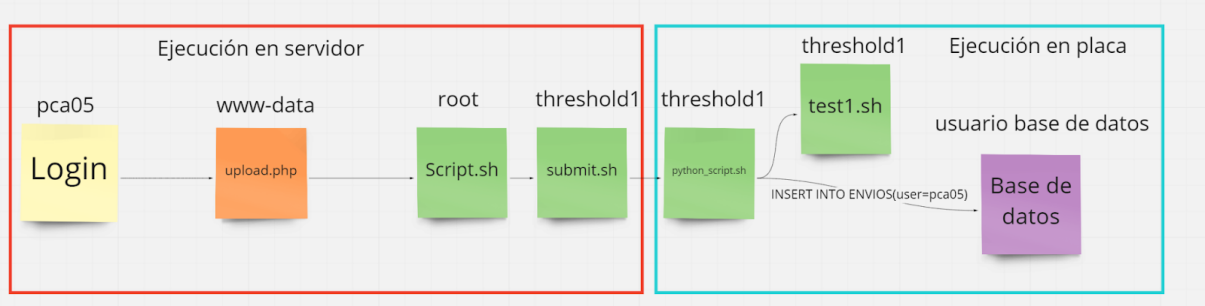


Figura 12.37. Flujo de cambio de usuario en la ejecución.

El funcionamiento resumido de *python_script* se puede ver en pseudocódigo en la figura 12.38.

```
Compilar_código
IF(compilación_erronea){
        Escribir base de datos;
        exit;
}
Ejecutar código
IF(Timeout exceeded OR output_erroneo){
        Escribir base de datos;
        exit;
}
ELSE{
        Ejecutar código con timing;
        Ejecutar todos los test adicionales;
        Escribir base de datos;
}
```

Figura 12.38. Pseudocódigo de python_script.py

Los test adicionales se encuentran en */mnt/pca/ej/labX/ejY/*
El script leerá toda la carpeta y ejecutará todos los archivos '.sh' que encuentre.
Se ha decidido implementarlo de esta manera para que añadir o quitar un test sea fácil y no dependa de tocar el resto del código del script.

Los test.sh deberán hacer un exit(Z), donde Z es la ID de la medalla que se otorgará por llegar a esa condición del script. Un ejemplo está en la figura 12.39.

```
#!/bin/bash
count=0
count=`/bin/cat $1/code_aux.instructions | /bin/grep umull | /usr/bin/awk '{print $1}'`
if [ $count -gt 20000 ]; then
        exit 7
else
        exit 6
fi
```

Figura 12.39. test1.sh del laboratorio 3, ejercicio 2.

En este caso el test devolverá el exit code 7 si se cumple el *if*; *python_script* recibirá su exit code y añadirá al envío la medalla con ID=7

Además, *python_script* realiza una ejecución de DynamoRIO, un programa que permite contar el número de instrucciones ejecutadas en ensamblador.

12.4.1 Análisis de binario

Antaño para implementar los test mencionados se utilizaba pin de Intel.
Esta herramienta permite instrumentar un binario y contar el número y tipo de instrucciones ensamblador que se han ejecutado.

Esta información se guarda temporalmente en un archivo que pueden acceder fácilmente los tests para contar cuantas instrucciones se han ejecutado de determinado tipo y así comprobar si se ha realizado una optimización o no. Aunque pin era compatible con ARM, Intel dejó de darle soporte. Si bien aún es posible descargarla [24] no es compatible con las Zedboard, por lo que no se ha podido utilizar.

Después se pensó en crear una herramienta para valgrind, aunque se descartó la opción, ya que analizando algunas de las herramientas creadas se descubrió que si bien valgrind podía contar el número de instrucciones las separaba en tres tipos: load, store y operaciones de ALU. En este último grupo estaban incluidas sumas, multiplicaciones, saltos…
La razón de esta diferenciación es que valgrind se utiliza principalmente para detectar fallos de accesos a memoria.
Como esta información no era suficiente para aplicar los tests, era necesario buscar otra alternativa.

Finalmente, se decidió utilizar DynamoRIO, un programa de instrumentación de binarios que sí estaba preparado para funcionar en ARM.
DynamoRIO dispone de muchas herramientas para analizar el código. La que se usa en *python_script* es opcodemix.
Para utilizarla primero es necesario generar una traza de la ejecución, como se puede ver en la figura 12.40.

```
root@joanpc:/home/joan# /mnt/pca/DynamoRIO-ARM-Linux-EABIHF-8.0.0-1/bin32/drrun -t drcachesim -offline
-outdir /opt/dynamorio -- pi.x 100 > /dev/null
```

Figura 12.40. Comando para la ejecución de DynamoRIO.

La ejecución lee el binario de la carpeta montada */mnt/pca* y guarda la traza temporalmente en */opt/dynamorio*. Esta carpeta es independiente en cada placa, no está montada en red y se elimina todo su contenido después de la ejecución de todos los tests.

También se reduce el tamaño de la entrada del programa: generar la traza es costoso, tanto en espacio de disco (el almacenamiento de las Zedboard es limitado) como en procesado.

Después de generar la traza se debe leer la información relevante con opcodemix, figura 12.41.

```
root@joanpc:/home/joan# /mnt/pca/DynamoRIO-ARM-Linux-EABIHF-8.0.0-1/bin32/drrun -t drcachesim
-simulator_type opcode_mix -indir /opt/dynamorio/
```

Figura 12.41. Comando para la ejecución con opcodemix.

Este comando cuenta todas las instrucciones ejecutadas de cada tipo, la salida es similar a la de la figura 12.42

```
Opcode mix tool results:
        1298504 : total executed instructions
         150915 :       cmp
         111587 :      strb
          90516 :        it
          89896 :         b
          88088 :     ldrsb
          76013 :       add
          72484 :       mov
          60505 :      subs
          53022 :       sub
          46083 :       ldr
          43986 :       mls
```

Figura 12.42. Salida de opcodemix.

El script creado captura la salida de esta última ejecución y la escribe en un archivo temporal al que los test pueden acceder.

Ejemplo de caso de uso

En este apartado se ejemplifica el proceso descrito anteriormente con uno de los ejercicios de un laboratorio de la asignatura: *pi.c*

El programa calcula los N primeros decimales de pi, el código completo se puede consultar en el Apéndice B: Código.
La función que más ralentiza la ejecución del programa es DIVIDE.
armv7, el procesador de las Zedboard, no tiene módulos especializados en realizar divisiones, por lo que se hace una división software.

Como se puede ver en la figura 12.43, la mayoría de veces se divide entre 25 y 239.

```
void calculate( void )
{
    int j;

    N4 = N + 4;

    SET( a, 0 );
    SET( b, 0 );

    for( j = 2 * N4 + 1; j >= 3; j -= 2 )
    {
        SET( c, 1 );
        LONGDIV( c, j );

        SUBTRACT( a, c, a );
        DIVIDE( a, 25 );

        SUBTRACT( b, c, b );
        DIVIDE( b, 239 );
        DIVIDE( b, 239 );

        progress();
    }

    SET( c, 1 );

    SUBTRACT( a, c, a );
    DIVIDE( a, 5 );

    SUBTRACT( b, c, b );
    DIVIDE( b, 239 );

    MULTIPLY( a, 4 );
    SUBTRACT( a, a, b );
    MULTIPLY( a, 4 );

    progress();
}
```

Figura 12.43. Función calculate de pi.c

Una optimización posible entonces es hacer especialización de las funciones y aplicar *memoization*.
Como se repiten muchos cálculos costosos, se puede precalcular el resultado y utilizarlo directamente en vez de realizar operaciones innecesarias.

En la figura 12.44 se puede ver el código de la función original, y en la figura 12.45 y 12.46 el código de la función de precalcular el resultado, y como quedaría la función DIVIDE25.

```
void DIVIDE( signed char *x, int n )
{
    int j, k;
    unsigned q, r, u;
    long v;

    r = 0;
    for( k = 0; k <= N4; k++ )
    {
        u = r * 10 + x[k];
        q = u/n;
        r = u - q * n;
        x[k] = q;
    }
}
```

Figura 12.44. Función DIVIDE original.

```
void DIVIDE25(signed char *x){
        int k;
    unsigned q,r,u;
    r=0;
    for( k = 0; k <= N4; k++ )
        {
        u = r + x[k];
            q = q25[u];
            r = r25[u];
            x[k] = q;
        }
}
```

Figura 12.45. DIVIDE25.

```
void precalculate(){
    for(int i=0;i<=249;i++){
        q25[i]=i/25;
        r25[i]=((i-(q25[i]*25))*10);
    }
}
```

Figura 12.46. Precalculate.

Al aplicar esta optimización, se necesita dividir y multiplicar menos veces. Por tanto, una forma de comprobar si se ha realizado correctamente esta mejora es mirar el número de multiplicaciones en ensamblador.

En la figura 12.39 se muestra el código del test que comprueba esto.
Si el número de multiplicaciones que se ha ejecutado es mayor que 20.000 se considera que la optimización no se ha realizado.

En la figura 12.47 hay una comparativa del número de multiplicaciones ejecutadas. A la izquierda se muestran las del código original y a la derecha las del código optimizado descrito con anterioridad.

```
Opcode mix tool results:
       1298504 : total executed instructions
        150915 :       cmp
        111587 :      strb
         90516 :        it
         89896 :         b
         88088 :     ldrsb
         76013 :       add
         72484 :       mov
         60505 :      subs
         53022 :       sub
         46083 :       ldr
         43986 :       mls
         43922 :       mla
         37190 :         b
         34175 :      lsrs
         33281 :     umull
```

```
Opcode mix tool results:
       1293454 : total executed instructions
        153311 :       cmp
        111607 :       ldr
        111587 :      strb
        108670 :       add
         92293 :         b
         90516 :        it
         88193 :     ldrsb
         71962 :       mov
         60505 :      subs
         52916 :       sub
         37189 :         b
         29706 :       adc
         29607 :       nop

           311 :     umull
```

Figura 12.47. Comparativa entre número de multiplicaciones.

13. Dificultades encontradas

Como se comenta en el apartado de gestión de riesgos el proyecto presenta posibles dificultades.
Respecto a las complicaciones previstas se incluyen desconocimiento sobre la arquitectura, la configuración inicial de las Zedboard y falta de experiencia en competencias como administración de sistemas operativo o desarrollo de páginas web. Además se añaden otras dificultades encontradas durante la realización del proyecto que no fueron previstas.

El primer problema encontrado fue en relación a la configuración inicial de las Zedboard. Al montar la estructura de red se observó que no era posible comunicarse con las dos placas a la vez, y que la conexión a ellas era muy lenta. Después de escanear la red, se vio que las dos placas tenían la misma MAC, lo que provocaba que el switch estuviera actualizando constantemente sus tablas. La primera idea para solucionar el problema fue aplicar conocimientos de las asignaturas de redes: si las placas estaban en subredes distintas, no hay ningún problema con que tengan la misma MAC. Sin embargo, como no se disponía de un router con más de una tarjeta de red no se pudo probar esta solución, además, tener una subred para cada placa provocaría un problema de escalabilidad. Finalmente se optó por cambiar la MAC que anunciaban las placas desde un archivo de configuración para así evitar el conflicto.

El segundo problema fue de compatibilidad entre versiones al instalar Slurm.
Las Zedboard tienen instalado ubuntu 16.04, mientras que en el servidor principal se decidió instalar 18.04, en parte porque era la versión que se repartía anteriormente en la asignatura.
Al instalar directamente los paquetes distribuidos de Slurm, la versión era distinta para 18.04 y 16.04. Si bien se mantiene la compatibilidad con un máximo de dos versiones anteriores, este no era el caso, y los daemon slurmctld y slurmd no podían comunicarse.
Actualizar las placas a ubuntu 18.04 era imposible, ya que utilizan armv7, de 32 bits, y no existen imágenes de 18.04 para sistemas de 32 bits.
La solución fue instalar manualmente en las placas una versión compatible con la del servidor.

El análisis de binarios generó un último problema, la incompatibilidad entre una función de Slurm y DynamoRIO.

Como se indica en el apartado de proceso de envío de código, la ejecución de los scripts se hace desde otro usuario, threshold1. Para hacer esto es necesario dar permisos de sudo y permitir que haga cambios de usuario sin contraseña. Slurm incluye una opción a la hora de enviar un *job* que permite ejecutarlo como si de otro usuario se tratase, basta con llamar a sbatch con la opción --uid.
DynamoRIO necesita crear unos archivos temporales de configuración para instrumentar el binario, esto se realiza de forma automática y transparente, y al finalizar la ejecución son eliminados.

Si se ejecuta DynamoRIO desde un *job* que se ha invocado con --uid se produce un error que indica que no se ha podido cargar la configuración correctamente.

Las dos posibles razones de este error eran:

a) Slurm no transmite las variables de entorno al nodo de cálculo.

b) Al ejecutar como cierto usuario, se reducen los permisos a los que tiene ese usuario.

Sin embargo, se podía ejecutar DynamoRIO y el resto de scripts directamente desde el usuario de prueba, o incluso hacer un sbatch, siempre que no se llamara con la opción de --uid.

La razón de este error no se llegó a comprender, Slurm debe realizar alguna otra tarea no documentada al utilizar la opción --uid.

En cualquier caso, la solución fue utilizar el comando *su -c* para hacer el cambio de usuario.

14. Conclusiones

Este proyecto tenía el objetivo de crear un prototipo de plataforma para PCA. En el momento de la finalización del trabajo este prototipo permite:

La conexión remota del alumnado para que trabaje, sea desde el servidor, enviando una tarea al clúster, o conectándose directamente a una de las placas. Esto resuelve una de las motivaciones principales del proyecto: permitir el trabajo remoto y no tener que prestar físicamente las Zedboard a los alumnos.

Acceso a un clúster funcional compuesto por todas las placas, lo que garantiza una alta disponibilidad, recogida automática de estadísticas sobre el uso del clúster y ejecuciones controladas en la arquitectura objetivo de estudio de PCA.

Acceso a una aplicación web que implementa las funciones de analizar todo el proceso de compilación y ejecución de código, y que además utiliza técnicas docentes que motiven al alumno y permitan informar y recomendar posibles mejoras fácilmente.

Recuperar y mejorar la gamificación que se había desarrollado hace tiempo para la arquitectura x86.

En definitiva, se cumplen todos los objetivos que se habían planteado y se resuelven los problemas que habían motivado el desarrollo de este proyecto.

14.1 Trabajo futuro

14.1.1 Mejora visual de la página web

La página web ha sido creada con la intención de que sea funcional. El diseño es muy sencillo y aunque es visual y fácil de navegar, no es estéticamente atractiva.
Para facilitar la viabilidad de este cambio se ha creado la página de forma modular, y así poder realizar cambios en la visualización sin necesidad de alterar el funcionamiento de la plataforma.

14.1.2 Compilación para FPGA

Desde las Zedboard no es posible compilar para FPGA; el software de compilación no tiene soporte para ARM.
Desde el servidor es posible, aunque tampoco es viable que todo el alumnado de PCA intente compilar a la vez. Una posible solución a este problema es añadir una nueva máquina preparada para esta tarea a una nueva partición del clúster y crear un script que envíe las tareas de compilación de hardware específicamente a esta máquina. La partición y el script se han elaborado como parte de este proyecto, aunque si se quiere utilizar esta funcionalidad se debe adquirir un nuevo hardware.

14.1.3 Reinicio remoto de las placas

Slurm tiene mecanismos para abortar la ejecución actual si una de las Zedboard se bloquea. Sin embargo, si la placa deja de responder completamente la única solución actual es reiniciar físicamente la placa.
Un posible campo de investigación sería la posibilidad de reiniciar de forma remota las placas, y así mejorar la calidad de vida del profesorado y evitar que alguien tenga que desplazarse hasta la ubicación del clúster.

14.1.4 Sistema de *log*

Si se quiere tener información más detallada sobre la ejecución y el análisis del código es posible implementar un sistema de *log* que registre toda la información del envío. Una opción para ello es aprovechar que Slurm captura la salida de los programas ejecutados en un nodo.

14.2 Impacto personal

Hace dos años llegué a la conclusión de que quería dedicarme a la docencia.

La posibilidad de crear una plataforma para una asignatura que dé soporte a las necesidades del profesorado y además permita aplicar técnicas estudiadas para favorecer la implicación del alumno es algo que no hubiera podido imaginar en ese momento.

Si bien ha habido momentos complicados en los que me he atascado, a lo largo de todo el proceso he descubierto algo de lo que no me había percatado hasta ahora: realmente he aprendido mucho en la carrera.

Desde conocimientos concretos como gestionar una base de datos, a aptitudes personales: la facilidad con la que he sido capaz de enfrentarme y buscar cómo resolver los problemas que me han ido surgiendo.

Este proyecto me ha permitido unir y aplicar todos los conocimientos técnicos adquiridos a lo largo de la ingeniería informática sin perder de vista la vocación profesional que quiero perseguir.

Creo que me hubiera costado encontrar un proyecto más acertado.

Apéndice A: Instalación

En este apartado se detalla el proceso seguido para la instalación de todo el software necesario para el correcto funcionamiento del clúster.
Además se especifica la configuración utilizada.

1. Configuración inicial del servidor

Para el servidor se ha decidido reinstalar el sistema operativo para poder tener una instalación limpia de la que partir.
Se ha instalado ubuntu server 18.04.5 por razones de familiaridad: es el sistema operativo que utilizan las Zedboard y el que se distribuía en los discos duros.

1.1 Configuración de red

El servidor debe estar en todo momento disponible, por tanto es necesario que tenga una IP privada fija.
Primero es necesario averiguar la interfaz que se va a utilizar con ifconfig.

```
joan@joanpc:~$ ifconfig
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.55  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::21d:60ff:fe6f:67d1  prefixlen 64  scopeid 0x20<link>
        ether 00:1d:60:6f:67:d1  txqueuelen 1000  (Ethernet)
        RX packets 71502  bytes 26024776 (26.0 MB)
        RX errors 0  dropped 5  overruns 0  frame 0
        TX packets 39355  bytes 8261557 (8.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 2  collisions 0
        device memory 0xdfec0000-dff00000
```

Figura A.1. Comando ifconfig.

Por defecto ubuntu 18.04.5 utiliza netplan, así que hay que editar el archivo */etc/netplan/50-cloud-init.yaml.* Tras modificarlo es necesario utilizar el comando *>netplan apply* para que los cambios tengan efecto.
La configuración se puede ver en la figura A.2 Es importante desactivar el dhcp, especificar la dirección IP, un *gateway* y unos servidores DNS (en este caso se utilizan los de google).

```
# This file is generated from information provided by
# the datasource.  Changes to it will not persist across an instance.
# To disable cloud-init's network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
    ethernets:
        enp2s0:
            addresses: [192.168.1.55/24]
            dhcp4: false
            gateway4: 192.168.1.1
            nameservers:
                addresses: [8.8.8.8,4.4.4.4]
                search: [google.com]
    version: 2
```

Figura A.2. Configuración de netplan.

2. Instalación del clúster

A continuación se detalla en orden los pasos necesarios para instalar Slurm.

2.1 Previo

Antes de nada, se debe instalar un compilador. Slurm requiere los siguientes paquetes:

-build-essential: Incluye compilador de c, c++ y make.
-python.
-lua50.
-libpam0g-dev: Un módulo PAM para restringir el acceso a los nodos.

2.2 Sincronizar usuarios, grupos y fecha

Tras esto, según la guía de instalación de Slurm oficial es necesario asegurarse de que la fecha, los UID y GID estén sincronizados en todo el clúster.

2.3 Sincronizar UID y GID

Los usuarios y grupos es posible sincronizarlos con LDAP, puppet o algún otro programa que lo permita. Dado que es una prueba de concepto, no

son demasiadas placas y una vez creadas las cuentas no se plantea añadir más (es una limitación dada por el número de alumnos máximos de una asignatura) se ha optado por hacer esta sincronización de forma manual.

La restricción real es que todo usuario que envíe un *job* al cluster debe estar registrado en el nodo en que se vaya a ejecutar y tener un directorio */home* asociado. Para esto se han creado los usuarios en todos los nodos y se han editado los archivos */etc/passwd* y */etc/group* para que tengan el mismo UID y GID.

En la figura A.3 tenemos un ejemplo de esto.

Figura A.3. Contenido de /etc/passwd

Los usuarios definidos para los alumnos de pca son los denominados pcaXX. Para los UID se ha decidido utilizar la nomenclatura de nombrarlos 18XX (donde XX es el número del usuario) y para los GID 19XX.
Es decir, el usuario pc23 tendría UID 1823 y GID 1923

Estos usuarios deben estar creados y con el mismo UID y GID en todos los nodos del cluster. El resto de usuarios que se muestran en la imagen no es relevante, pueden existir en un nodo y no en otro.
La única excepción es el usuario slurm: al instalar Slurm se creará automáticamente con el UID 64030, aunque es conveniente revisar que tenga el mismo UID en todos los nodos, pues también es requisito.

2.4 Sincronización de fecha

Los nodos y el servidor deben tener la misma fecha.
Para esto se ha utilizado Network Time Protocol (NTP).
La fecha real no es importante, la única condición es que tengan la misma. Esto también es requisito para Munge, del que se habla más adelante.

En primer lugar se configuró de forma que el servidor recibía la hora de una de las pools predeterminadas de NTP y que los nodos recibían la hora del servidor. (Esto es, el servidor funcionaría también como un servidor NTP). En ocasiones, sin embargo, la sincronización tardaba unos cinco minutos en efectuarse, lo que podía suponer un problema si uno de los nodos debía reiniciarse. Para solucionar esto se ha optado por utilizar la configuración por defecto de NTP y que todas las máquinas accedan a la *pool* del proyecto NTP.

Para permitir la conexión es necesario abrir el puerto 123 udp.

2.5 Munge

Otro de los requisitos de Slurm es Munge.
Munge es un servicio de autentificación diseñado para funcionar en un clúster. Es posible instalarlo manualmente o de forma directa con
>apt-get install munge

Munge utiliza una clave para realizar la comprobación de credenciales. El servidor y todos los nodos deben tener la misma clave, además como se ha comentado antes todos los relojes de las máquinas deben estar sincronizados.

Es posible generar la clave utilizando el binario que incluye la instalación */usr/sbin/create-munge-key.* Esta clave debe situarse en el directorio */etc/munge/munge.key* y debe pertenecer al usuario munge.

Se recomienda pasar la clave a los otros nodos mediante un método seguro como ssh.
Además, Munge debe estar siempre activo para que Slurm funcione, así que para hacer que se active al encender o reiniciar cualquier máquina:
>systemctl enable munge

2.5.1 Seguridad y permisos

Munge comprueba que algunos permisos estén establecidos correctamente para evitar posibles brechas de seguridad.
Es necesario comprobar los permisos de los siguientes directorios:

/etc/munge/ -> 0700
/etc/munge/munge.key -> 0600
/var/lib/munge -> 0700
/var/log/munge -> 0700
/run/munge -> 0755

2.5.2 Comprobación

Es posible comprobar si Munge se ha instalado y configurado correctamente de la siguiente manera.

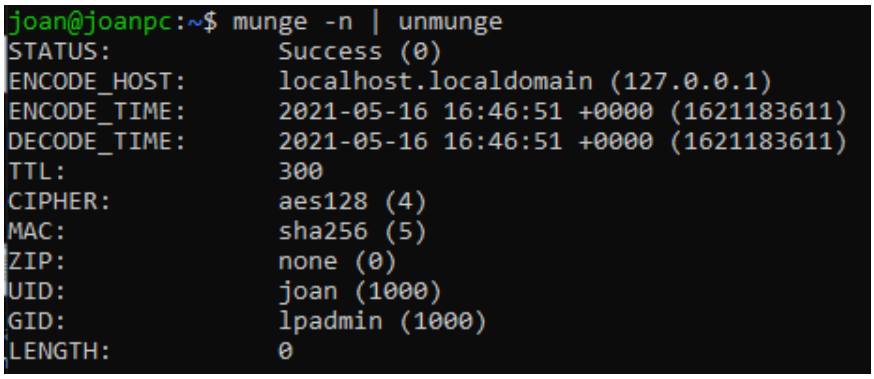


Figura A.4. Comprobación funcionamiento Munge.

El comando *>munge -n | unmunge* codifica y decodifica una credencial localmente. Si el resultado es *succes* significa que la clave funciona y munge está activo.

Para comprobar el funcionamiento entre dos máquinas es posible hacer:

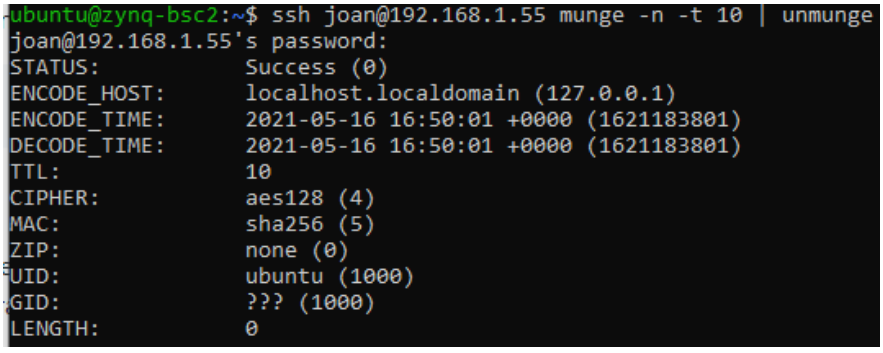


Figura A.5. Comprobación entre dos máquinas.

El comando *>ssh somehost munge -n -t 10 | unmunge* permite comprobar si dos máquinas tienen la misma clave. Si no es el caso o los relojes no están sincronizados este comando fallará.

2.6 Slurm

Una vez se han cumplido todos los requisitos anteriores ya es posible instalar Slurm.

Se ha instalado la versión 17.11.13-2, que se puede descargar de la página oficial.
Después de descargar y descomprimir el archivo hay que crear los directorios en los que se instalará: Slurm no crea directorios automáticamente para comprobar que la instalación es correcta y el administrador ha entendido bien el proceso.

Los directorios a crear son:

Localización del módulo PAM: */lib/security*
Localización de Slurm: */opt/slurm*
Localizaciones de logs y archivos de configuración:

/opt/slurm/var/log/
/opt/slurm/etc/
/opt/slurm/var/
/opt/slurm/var/run
/opt/slurm/var/spool
/opt/slurm/var/spool/slurm

Para empezar a instalar Slurm hay que utilizar el siguiente comando:

```
>./configure --prefix=/opt/slurm --enable-pam --with-munge=/usr --with-pam_dir=/lib/security
```

Las opciones incluidas son la de *--prefix*, que indica donde se instalará, *--with-munge* (que indica que se utilizará munge como medio de autentificación y especifica la ubicación) y *--enable-pam*, *--with-pam_dir* que indican que se utilizará el módulo adicional pam y su ubicación, respectivamente.

Cuando finalice el *./configure* se debe ejecutar *>make* y *>make install.*

2.6.1 Servicios de Slurm

Una vez haya acabado la instalación se debe crear el servicio para poder tenerlo siempre activo (tanto slurmctld en el servidor como slurmd en los nodos). En la propia carpeta de instalación de Slurm hay un ejemplo de este archivo.
>cp slurm-17.11.13-2/etc/slurmctld.service /etc/systemd/system/slurmctld.service

En la figura A.6 se puede ver el contenido del archivo.

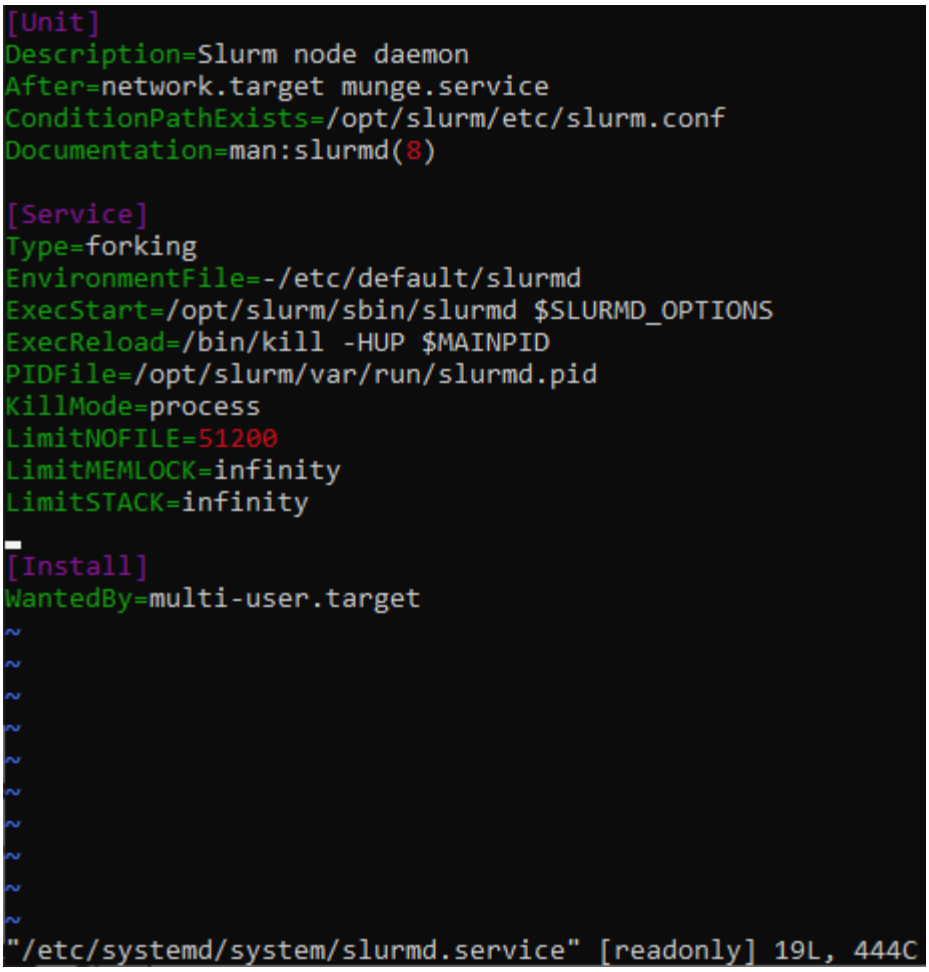


Figura A.6. Contenido de slurmd.service.

Es importante especificar bien las ubicaciones.
En concreto *ConditionPathExists* comprueba que exista un archivo de configuración necesario (que se comentará más adelante).

De forma análoga hay que crear el archivo *slurmctld.service*.

Una vez se hayan creados estos archivos, hay que recargarlos con *>systemctl daemon-reload*

2.6.2 slurm.conf

La configuración entera de Slurm se puede ver con *>scontrol show config*
No toda la información que muestra este comando está definida en el archivo de configuración, pero es posible modificarla si se añade la línea.

Es importante que *slurm.conf* esté ubicado donde se especifica en el servicio, y el archivo debe ser el mismo en todo el clúster. Es decir, en el servidor y en todos los nodos de cálculo.

```
# slurm.conf file generated by configurator easy.html.
# Put this file on all nodes of your cluster.
# See the slurm.conf man page for more information.
#
ControlMachine=joanpc
ControlAddr=192.168.1.55
#
#MailProg=/bin/mail
MpiDefault=none
#MpiParams=ports=#-#
ProctrackType=proctrack/pgid
ReturnToService=1
SlurmctldPidFile=/var/run/slurm-llnl/slurmctld.pid
#SlurmctldPort=6817
SlurmdPidFile=/var/run/slurm-llnl/slurmd.pid
#SlurmdPort=6818
SlurmdSpoolDir=/var/spool/slurm-llnl/slurmd
SlurmUser=slurm
#SlurmdUser=root
StateSaveLocation=/var/spool/slurm-llnl
SwitchType=switch/none
TaskPlugin=task/none
```

Figura A.7. slurm.conf.

Las dos primeras líneas indican el nombre de la máquina en que funcionará slurmctld (definido en */etc/hosts*) y su dirección IP.
Luego se especifican ubicaciones de archivos de PID o directorios temporales, y se explicita el usuario que controlará Slurm.

Figura A.8. Configuración relativa al scheduler.

El scheduler por defecto de Slurm es un FIFO, en este caso se ha configurado, como se puede ver en la figura A.8, para ser FIFO con un sistema de prioridades. Este tipo de configuración es importante para poder calcular el porcentaje de uso del clúster de un usuario, y así asignar distintas prioridades. El objetivo de esto es evitar que un usuario pueda monopolizar el clúster y dar más prioridad de ejecución a usuarios que hayan utilizado menos el clúster. Además, este sistema requiere de *accounting* (figura A.9) y por tanto una base de datos.

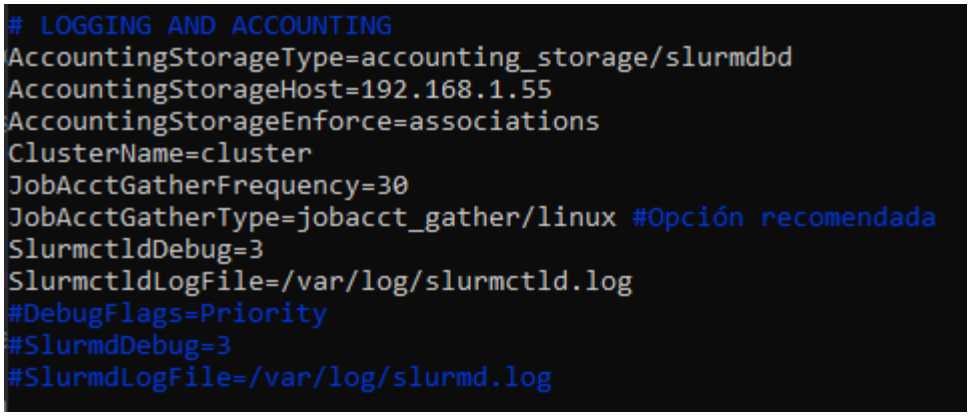


Figura A.9. Accounting y logging.

Respecto al *log*, se puede ver que existen opciones para indicar la ubicación de los logs de slurmctld y slurmd. Es posible desactivar el *log* comentando la línea. Además hay opciones de más o menos *debug* según se quiera, o incluso añadir más *flags* relacionadas con un aspecto en concreto. Se puede ver que hay comentada la opción de añadir más *log* de las prioridades, para comprobar que el sistema funciona correctamente.

Por último, en la figura A.10 podemos ver la partición de Slurm.

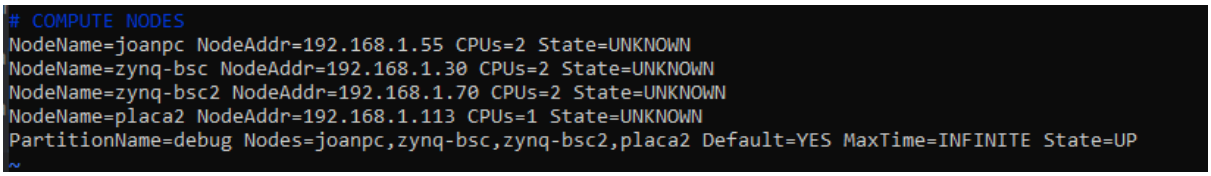


Figura A.10. Especificación de los nodos.

En este apartado se mencionan todos los nodos, también es posible especificar hardware si se quiere restringir (como indicar que una máquina solo tiene 1 CPU).
El parámetro final *MaxTime* indica el tiempo máximo en que un nodo puede estar allocated. Sería posible restringirlo desde aquí, aunque como queremos límites distintos según si un usuario se conecta de forma interactiva o envía un *job* se restringirá desde el propio script de envío.

Una vez este archivo está en todos los nodos del clúster, ya sería posible hacer *>systemctl enable slurmctld* en el servidor y *>systemctl enable slurmd* en los nodos, pero para que las prioridades y la seguridad añadida funcionen correctamente queda instalar y configurar el módulo PAM e instalar la base de datos.

2.6.3 Módulo Slurm PAM.

Los Pluggable Authentication Module (PAM) son mecanismos para gestionar la autenticación en dispositivos Linux.

Slurm incluye un módulo opcional que se puede instalar para bloquear el acceso a las nodos de cálculo si el usuario no tiene ningún *job* activo en él. De esta forma se refuerza la seguridad y se asegura que para que utilice una Zedboard se ha tenido que hacer un *>salloc* antes.

Para instalarlo es necesario hacer *>make contribs* en el directorio de instalación de Slurm. Tras esto se creará un archivo *pam_slurm.so*, que se ha decidido colocar en */opt/slurm/lib/security/* aunque es posible colocarlo en otra ubicación.
A continuación hay que indicar que se utilizará esta nueva librería, para ello se modifica el archivo */etc/pam.d/sshd* (ya que se quiere restringir el acceso por ssh, si se quiere restringir otro tipo de acceso, o todo tipo, hay que modificar otro módulo, aunque como los usuarios no podrán acceder a las placas de forma física con restringir el acceso por ssh es suficiente).

Al final del documento añadimos la línea de la figura A.11.

```
account required /opt/slurm/lib/security/pam_slurm.so
                                                              56,0-1        Bot
```

Figura A.11. Indicación para que se utilice pam_slurm.so

A partir de ahora, al intentar conectarse mediante ssh a un nodo con esta configuración aparecerá el siguiente mensaje:

```
joanubuntu@DESKTOP-DK3CKGQ:~$ ssh -X placauser1@192.168.1.114
placauser1@192.168.1.114's password:
Access denied: user placauser1 (uid=1003) has no active jobs on this node.
Connection closed by 192.168.1.114 port 22
```

Figura A.12. Mensaje de error al intentar conectar.

Sin embargo si hay un problema o es necesario cambiar alguna configuración de la placa a nivel de administrador es importante poder conectarse directamente por ssh y obviar esta restricción.
Para ello modificamos el mismo archivo que antes, y debajo del requisito de *pam_nologin.so* añadimos la línea de la figura A.13.

```
# PAM configuration for the Secure Shell service

# Standard Un*x authentication.
@include common-auth

# Disallow non-root logins when /etc/nologin exists.
account    required     pam_nologin.so

# Uncomment and edit /etc/security/access.conf if you need to set complex
# access limits that are hard to express in sshd_config.
 account  sufficient     pam_access.so
```

Figura A.13. Cambio en la configuración.

Este cambio especifica que si se cumplen las condiciones del módulo *pam_acces.so* será suficiente para permitir el acceso.

Modificamos */etc/security/acces.conf* como en la figura A.14.

```
# All other users should be denied to get access from all sources.
+:joan:ALL
- : ALL : ALL
```

Figura A.14. Configuración de pam_acces.so

Donde es posible ver dos líneas, la primera con un + delante que indica que el siguiente usuario, independientemente de la dirección de red de la que proceda (ALL) tendrá acceso. La siguiente línea especifica que cualquier otro usuario se le denegará y se tendrán que comprobar el resto de requisitos.

De esta forma es posible añadir una cuenta de administrador que no requiera de un *job* activo para poder conectarse por ssh a los nodos.

2.6.4 Base de datos de Slurm y accounting

Como se ha comentado anteriormente para poder establecer un sistema de prioridades en Slurm basado en el uso histórico del clúster es necesario mantener un registro de todos los envíos.
Para ello es necesario una base de datos, aunque según la documentación de Slurm también es posible guardar este registro en un archivo de texto. Sin embargo, esto no ofrece ninguna ventaja más allá de la relativa facilidad de configuración, así que como se necesitaba una base de datos de todas formas se ha optado por utilizarla para esta función también.

Slurm está preparado para funcionar con mysql o MariaDB. Es posible utilizar otra base de datos, aunque for familiaridad tanto del profesorado como del autor con sql se ha optado por MariaDB.

La comunicación entre Slurm y la base de datos escogida se hace mediante otro daemon: slurmdbd.
Por falta de hardware disponible se ha instalado la base de datos en el mismo servidor que se utiliza para gestionar el clúster, aunque es posible tenerla en otra máquina.

Para que slurmdbd funcione se debe crear otro archivo de configuración en la misma ubicación que *slurm.conf* (en este caso */opt/slurm/etc/*).
En la figura A.15 se puede ver la configuración.
Como se puede observar está la posibilidad de definir qué *jobs* se requieren registrar (por ejemplo no se registran los suspendidos manualmente por el usuario), cada cuanto se debe limpiar este registro (si se quiere que las prioridades de un usuario afecten a todo el cuatrimestre o solo a una semana) y la ubicación de la base de datos.

```
ArchiveEvents=yes
ArchiveJobs=yes
ArchiveResvs=yes
ArchiveSteps=no
ArchiveSuspend=no
ArchiveTXN=no
ArchiveUsage=no
#Auth_info
AuthType=auth/munge
DbdHost=joanpc
DebugLevel=info
PurgeEventAfter=1month
PurgeJobAfter=12month
PurgeResvAfter=1month
PurgeStepAfter=1month
PurgeSuspendAfter=1month
PurgeTXNAfter=24month
LogFile=/var/log/slurmdbd.log
SlurmUser=slurm
StorageType=accounting_storage/mysql
StoragePass=1234
StorageUser=slurm
StorageHost=joanpc
```

Figura A.15. Configuración de slurmdbd.

Respecto a Mariadb, cuando se active por primera vez el *daemon* de slurmdbd se creará una base de datos automáticamente (figura A.16).

```
MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| envios_pca         |
| information_schema |
| mysql              |
| performance_schema |
| slurm_acct_db      |
+--------------------+
5 rows in set (0.00 sec)
```

Figura A.16. slurm_acct_db se crea automáticamente.

Y también se crearán una serie de tablas.

```
MariaDB [slurm_acct_db]> SHOW TABLES;
+--------------------------------+
| Tables_in_slurm_acct_db        |
+--------------------------------+
| acct_coord_table               |
| acct_table                     |
| clus_res_table                 |
| cluster_assoc_table            |
| cluster_assoc_usage_day_table  |
| cluster_assoc_usage_hour_table |
| cluster_assoc_usage_month_table |
| cluster_event_table            |
| cluster_job_table              |
| cluster_last_ran_table         |
| cluster_resv_table             |
| cluster_step_table             |
| cluster_suspend_table          |
| cluster_table                  |
| cluster_usage_day_table        |
| cluster_usage_hour_table       |
| cluster_usage_month_table      |
| cluster_wckey_table            |
| cluster_wckey_usage_day_table  |
| cluster_wckey_usage_hour_table |
| cluster_wckey_usage_month_table |
| convert_version_table          |
| federation_table               |
| qos_table                      |
| res_table                      |
| table_defs_table               |
| tres_table                     |
| txn_table                      |
| user_table                     |
+--------------------------------+
29 rows in set (0.00 sec)
```

Figura A.17. Tablas de Slurm creadas.

Sin embargo para que se puedan registrar correctamente los envíos se debe dar permisos al usuario slurm en todas las tablas.

```
MariaDB [(none)]> grant all on slurm_acct_db.* TO 'slurm'@'localhost' identified by '1234';
```

Figura A.18. Comando para dar permisos al usuario.

Con esto el clúster registrará todos los envíos y los guardará en la base de datos.

Pese a la configuración añadida, el sistema de prioridades no funcionará aún. Para que Slurm registre correctamente el usuario que ha enviado un *job* y así poder contabilizar el uso hay que crear esa cuenta dentro del sistema de *accounting*.
Este sistema se configura con el comando *>sacctmgr.*

En primer lugar se ha definir el clúster:

>sacctmgr add cluster cluster

La segunda aparición de cluster representa el nombre que se le ha dado en el archivo *slurm.conf*, el nombre por defecto y el que se ha utilizado es cluster.

Tras esto, hay que definir una cuenta por cada alumno, definir un usuario y asociarlos.

>sacctmgr add account pca01
>sacctmgr add user pca01 DefaultAccount=pca01

Y de forma análoga se definen todos los usuarios.
Es posible comprobar la configuración con el comando de la figura A.19.

```
root@joanpc:/home/joan# sacctmgr show Association
   Cluster    Account       User  Partition     Share GrpJobs       GrpTRES GrpSubmit     GrpWall   GrpTRESMins
MaxJobs       MaxTRES MaxTRESPerNode MaxSubmit     MaxWall   MaxTRESMins                 QOS   Def QOS GrpTRESR
unMin
---------- ---------- ---------- ---------- --------- ------- ------------- --------- ----------- -------------
------- ------------ -------------- --------- ----------- ------------- -------------------- --------- --------
-----
   cluster       root                               1
                                                                                        normal
   cluster       root       root                    1
                                                                                        normal
   cluster      pca05                               1
                                                                                        normal
   cluster      pca05      pca05                    1
                                                                                        normal
```

Figura A.19. cluster, cuentas y usuarios asociados.

2.6.5 Modo interactivo

Como se ha comentado anteriormente, se ha definido un alias para facilitar la conexión a las placas en modo interactivo. Se ha especificado en */etc/bash.bashrc* para que afecte a todos los usuarios.

Figura A.20. Alias declarado.

Este alias ejecuta un script, el de la figura A.21.

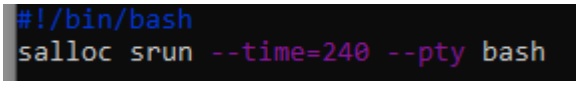


Figura A.21. Script sconnect.

2.7 Network File System (NFS)

Como se especifica en el apartado de diseño del clúster, se utiliza NFS para facilitar la comunicación entre el servidor y los nodos.

En la figura A.22 se muestra la configuración del servidor para exportar los directorios.

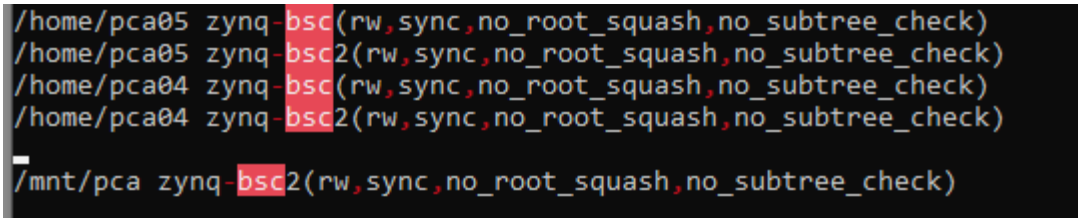


Figura A.22. Configuración de /etc/exports

Por un lado */mnt/pca* debe estar compartido con todas las placas y siempre estará montado.

Por otro lado los diferentes */home* de todos los usuarios también deben estar compartidos con todas las placas, aunque como se ha comentado solo estarán montados bajo demanda.

Y en cada nodo solo es necesario:

```
192.168.1.55:/mnt/pca /mnt/pca nfs auto,nofail,noatime,nolock,intr,tcp,actimeo=1800 0 0
192.168.1.55:/var/www/html/uploaded_codes /var/www/html/uploaded_codes nfs auto,nofail,noatime,nolock,intr,tcp,actimeo=1800 0 0
```

Figura A.23. Configuración de /etc/fstab.

Que montará el directorio */mnt/pca* cada vez que se encienda la placa.

Para los directorios */home* se utiliza los *prolog* y *epilog*. Son funcionalidades de Slurm que permiten ejecutar como root un script al inicio de un *job* y otro al final.

Para configurarlos hay que especificar su ruta en el archivo */opt/slurm/etc/slurm.conf* de cada nodo (figura 26).

```
Prolog=/opt/slurm/prolog
Epilog=/opt/slurm/epilog
```

Figura A.24. slurm.conf prolog y epilog

Se puede ver el *prolog* en la figura A.25.

```
#!/bin/bash
usuario=$(/usr/bin/id -un $SLURM_JOB_UID)
/bin/mount 192.168.1.55:/home/$usuario /home/$usuario
```

Figura A.25. Prolog de slurm.

Lo que hace este código es conseguir el nombre del usuario mediante el parámetro *$SLURM_JOB_UID* que viene de slurmd e indica el UID del usuario que ha ejecutado el *job*. Una vez conseguido el nombre del usuario, monta el directorio que corresponde a su */home*.

De la misma manera, el *epilog* de la figura A.26 desmonta el */home* en cuanto la ejecución del *job* ha acabado.

```
#!/bin/bash
usuario=$(/usr/bin/id -un $SLURM_JOB_UID)
/bin/umount /home/$usuario
```

Figura A.26. Epilog de slurm.

Hay dos aclaraciones relevantes respecto a este funcionamiento:

Si el *prolog* o el *epilog* no se pueden ejecutar (por ejemplo el administrador decide montar manualmente el */home* o se han creado mal los usuarios y el UID no coincide, el *job* quedará en estado cancelado y el nodo pasará a drained, es decir que no estará disponible hasta que se levante manualmente (*>scontrol update nodename="nombre" state=resume*).

Por último, el *prolog* se ejecuta cuando un *job* empieza su ejecución, es decir con un sbatch o un srun; el salloc no ejecuta el *prolog*.
Es posible cambiar este comportamiento, pero debido a que se utiliza un script que ejecuta primero el salloc y el srun solo cuando se ha concedido acceso este comportamiento ya se desea, ya que no provocará problemas de ejecución. Aunque si se cambiara el funcionamiento sería algo a tener en cuenta, ya que no interesa tener un nodo montado si no se va a ejecutar nada en él.

Apéndice B: Código

En este apéndice queda recogido todo el código desarrollado para la plataforma. Se divide en dos secciones, el relevante para la aplicación web, y el relevante para el clúster y el proceso de envío.

1. Código de la aplicación web

index.html

```
<!DOCTYPE html>
<html lang="es">
	<head>
		<meta charset="UTF-8">
		<meta name="viewport" content="width=device-width,initial-scale=1.0">
		<meta http-equiv="X-UA-Compatible" content="ie-edge">
		<title>PCA</title>
	</head>
	<body>
		<form action="login.php" method="post">
			<input name="user" type="text" placeholder="User">
			<br><br>
			<input name="password" type="password" placeholder="Password">
			<br><br>
			<input type="submit" value="Log In">
		</form>
	</body>
</html>
```

login.php

```
<?php
$dbuser='pag_web_admin';
$pass='1234';
$charset='utf8mb4';
$dsn="mysql:host=localhost;dbname=envios_pca;charset=$charset";

$pdo= new PDO($dsn,$dbuser,$pass);
$statement=$pdo->prepare("SELECT * FROM users_pca WHERE username = ?");
$statement->execute([$_POST['user']]);
$user=$statement->fetch();
```

```
if($user && password_verify($_POST["password"], $user["password"])){
	session_start();
	$_SESSION["user"]=$_POST['user'];
	if($_SESSION["user"]=='admin'){
		header("Location: ad_home.php");
	}
	else
		header("Location: home.php");
}
else	header("Location: index.html");
```

logout.php

```
<?php
session_start();
session_destroy();
header("Location: index.html");
```

home.php

```
<?php

session_start();
if(empty($_SESSION["user"])){
	header("Location:index.html");
}

echo "<h1> PCA </h1> <br>";

$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");
$my_query="SELECT labs_pca.ID AS lab_ID, labs_pca.description AS lab_description,
ejs_pca.ID as ej_ID, ejs_pca.description AS ej_description FROM labs_pca LEFT JOIN
ejs_pca ON labs_pca.ID=ejs_pca.ID_LAB ORDER BY lab_ID";
$result=$mysqli->query($my_query);
$row_number=($result->num_rows);
$col_number=($mysqli->field_count);

$labs_ID=[];
$ejs_ID=[];
$labs_description=[];
$ejs_description=[];

while($rows=mysqli_fetch_array($result)){
	$labs_ID[]=$rows['lab_ID'];
	$ejs_ID[]=$rows['ej_ID'];
	$labs_description[]=$rows['lab_description'];
	$ejs_description[]=$rows['ej_description'];
}
```

```
$last_lab=-1;
$i=0;
while($i<$row_number){
	$aux_id_lab=$labs_ID[$i];
	$aux_id_ej=$ejs_ID[$i];
	if($last_lab!=$aux_id_lab){
		echo "<h2> LAB$aux_id_lab: $labs_description[$i] </h2>";
	}
	$display="EX$aux_id_ej: $ejs_description[$i]";
	$location="<p><a href=lab.php?lab=$aux_id_lab&ej=$aux_id_ej";
	echo $location . ">" . $display . "</a></p>";
	$last_lab=$aux_id_lab;
	$i++;

}
?>
<br>
<br>
<br>
<h3> <a href="logout.php">Sign Off</a></h3>
```

ad_home.php

```
<?php

session_start();

if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}

echo "<h1> PCA </h1> <br>";

$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");
$my_query="SELECT labs_pca.ID AS lab_ID, labs_pca.description AS lab_description,
ejs_pca.ID as ej_ID, ejs_pca.description AS ej_description FROM labs_pca LEFT JOIN
ejs_pca ON labs_pca.ID=ejs_pca.ID_LAB ORDER BY lab_ID";
$result=$mysqli->query($my_query);
$row_number=($result->num_rows);
$col_number=($mysqli->field_count);

$labs_ID=[];
$ejs_ID=[];
$labs_description=[];
$ejs_description=[];
```

```
while($rows=mysqli_fetch_array($result)){
	$labs_ID[]=$rows['lab_ID'];
	$ejs_ID[]=$rows['ej_ID'];
	$labs_description[]=$rows['lab_description'];
	$ejs_description[]=$rows['ej_description'];
}

$last_lab=-1;
$i=0;
while($i<$row_number){
	$aux_id_lab=$labs_ID[$i];
	$aux_id_ej=$ejs_ID[$i];
	if($last_lab!=$aux_id_lab){
		echo "<h2> LAB$aux_id_lab: $labs_description[$i] </h2>";
	}
	$display="EX$aux_id_ej: $ejs_description[$i]";
	$location="<p><a href=ad_lab.php?lab=$aux_id_lab&ej=$aux_id_ej";
	echo $location . ">" . $display . "</a></p>";
	$last_lab=$aux_id_lab;
	$i++;

}
?>
<br>
<br>
<h1> Tools </h1>
<h3> <a href="crear_usuarios.php">User management</a></h3>
<h3><a href="crear_badges_index.php">Badge management</a></h3>
<h3><a href="exercise_management.php">Exercise management</a></h3>
<br>
<h3> <a href="logout.php">Sign Off</a></h3>
```

lab.php

```
<!DOCTYPE html>
<?php
session_start();
if(empty($_SESSION["user"])){                                    #Security
check, is user logged?
	header("Location:index.html");
}
$lab=$_GET['lab'];
$ej=$_GET['ej'];
if(!is_numeric($lab) or !is_numeric($ej)){                       #Security
check, if lab or ej are not a number then destroy session
	session_destroy();
	header("Location:index.html");
}
$_SESSION["lab"]=$lab;
```

```
$_SESSION["ej"]=$ej;
$dbuser='pag_web_admin';
$pass='1234';
$charset='utf8mb4';

$dsn="mysql:host=localhost;dbname=envios_pca;charset=$charset";                #Security
check, if lab or ej does not exist then destroy session
$pdo= new PDO($dsn,$dbuser,$pass);
$statement=$pdo->prepare("SELECT * FROM ejs_pca WHERE ID = ? AND ID_LAB = ? ");
$statement->execute([$ej,$lab]);
$result=$statement->fetch();
if(!isset($result[0])){
       session_destroy();
       header("Location:index.html");
}

                                                                    #Get the number of
non_manual badges on this exercise, will be needed to check that every submission
printed has passed all tests
$max_badges_non_manual=2;                                           #First 2 badges do
exist for every exercise(compilation and execution OK)
$files=scandir("/mnt/pca/ej/lab$lab/ej$ej");




foreach($files as $file){
       $filepart=explode(".",$file);
       if($filepart[1]=="sh"){
              $max_badges_non_manual++;
       }
}


$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");
$my_query="SELECT * FROM envios_pca WHERE lab=$lab AND ej=$ej AND exec_time IN (SELECT
MIN(exec_time) FROM envios_pca WHERE exec_time<>0 AND (SELECT count(*) FROM
rel_envios_badges INNER JOIN badges_pca ON badges_pca.ID=rel_envios_badges.ID_badge
WHERE rel_envios_badges.ID_envio=envios_pca.ID AND type NOT IN ('1','6')) >=
$max_badges_non_manual GROUP BY user) OR user IN ('threshold1','threshold2') AND
lab=$lab AND ej=$ej AND exec_time IN (SELECT MIN(exec_time) FROM envios_pca WHERE
exec_time<>0 GROUP BY user) ORDER BY exec_time";
$result=$mysqli->query($my_query);
$row_number=($result->num_rows);
$col_number=($mysqli->field_count);

$users=[];
$exec_times=[];
$datas=[];
```

```
while($rows=mysqli_fetch_array($result)){
	$users[]=$rows['user'];
	$exec_times[]=$rows['exec_time'];
	$datas[]=$rows['date'];
}

##########################################TOP
SUBMISSIONS##########################################
?>

<html>
<body>


	<table align="center" border="1px" style="width:800px; line-height:20px;
border-collapse: collapse">
		<tr>
		<th colspan="5"> <h2> TOP submissions </h2></th>
		</tr>
			<th> Ranking </th>
			<th> User </th>
			<th> Execution time </th>
			<th> Date </th>
			<th> Badges </th>



<?php
for($i=0;$i<$row_number;$i++){
	echo "<tr>";
	echo "<td align='right'>" . ($i+1) . "</td>";
	echo "<td align='right'>" .  $users[$i] . "</td>";
	echo "<td align='right'>" .  $exec_times[$i] . "</td>";
	echo "<td align='right'>" .  $datas[$i] . "</td>";
	if($i<3){
		$type_aux=$i+3;
		echo "<td align='center'> <img src='$type_aux.png' alt=BADGE width='25'
height='25' /> </td>";
	}


}
	echo "</tr>";
?>

	</table>
	<br>
	<form method="POST" action="upload.php" enctype="multipart/form-data">
		<div>
```

```
                    <span> Upload a file. Users must upload only their own code. </span>
                    <input type="file" name="File" />
                </div>
                <input type="submit" name="uploadButton" value="Upload" />
        </form>
        <br>
        <a href='lab_submissions.php'>My submissions</a>
        <br><br><h3><a href='home.php'>Go back</a></h3>

        <h3><a href="logout.php">Sign Out</a></h3>
</body>
</html>
```

upload.php

```
<?php
session_start();

if(empty($_SESSION["user"])){
        header("Location:index.html");
}

$lab=$_SESSION["lab"];
$ej=$_SESSION["ej"];
$user=$_SESSION['user'];
if (isset($_FILES['File']) && $_FILES['File']['error']== 0){
        $file=$_FILES['File']['name'];
        $filepart=explode(".",$file);
        if($filepart[1]!="c"){
                echo "The file extension must be a .c <br> Returning...";
                header("Refresh:3; url=lab.php?lab=$lab&ej=$ej");
                exit();
        }
        else{
        $tmp_name=$_FILES['File']['tmp_name'];
        $move_location= "/mnt/pca" . $tmp_name . ".c";
        move_uploaded_file($tmp_name, $move_location);
        $result=shell_exec("sudo /mnt/pca/PCA_scripts/script.sh $user $move_location
$lab $ej");
        header("Location:lab.php?lab=$lab&ej=$ej");
        }
}
else header("Location:lab.php?lab=$lab&ej=$ej");
```

ad_lab.php

```
<!DOCTYPE html>
<?php
session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}

$lab=$_GET['lab'];
$ej=$_GET['ej'];
if(!is_numeric($lab) or !is_numeric($ej)){
	session_destroy();
	header("Location:index.html");
}
$_SESSION["lab"]=$lab;
$_SESSION["ej"]=$ej;
$dbuser='pag_web_admin';
$pass='1234';
$charset='utf8mb4';

$dsn="mysql:host=localhost;dbname=envios_pca;charset=$charset";
$pdo= new PDO($dsn,$dbuser,$pass);
$statement=$pdo->prepare("SELECT * FROM ejs_pca WHERE ID = ? AND ID_LAB = ? ");
$statement->execute([$ej,$lab]);
$result=$statement->fetch();
if(!isset($result[0])){
	session_destroy();
	header("Location:index.html");
}


                                                        #Get the number of
non_manual badges on this exercise, will be needed to check that every submission
printed has passed all tests
$max_badges_non_manual=2;                               #First 2 badges do
exist for every exercise(compilation and execution OK)
$files=scandir("/mnt/pca/ej/lab$lab/ej$ej");
foreach($files as $file){
	$filepart=explode(".",$file);
	if($filepart[1]=="sh"){
		$max_badges_non_manual++;
	}
}
```

```
$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");
$my_query="SELECT * FROM envios_pca WHERE lab=$lab AND ej=$ej AND exec_time IN (SELECT
MIN(exec_time) FROM envios_pca WHERE exec_time<>0 AND (SELECT count(*) FROM
rel_envios_badges INNER JOIN badges_pca ON badges_pca.ID=rel_envios_badges.ID_badge
WHERE rel_envios_badges.ID_envio=envios_pca.ID AND type NOT IN ('1','6')) >=
$max_badges_non_manual GROUP BY user) OR user IN ('threshold1','threshold2') AND
lab=$lab AND ej=$ej AND exec_time IN (SELECT MIN(exec_time) FROM envios_pca WHERE
exec_time<>0 GROUP BY user) ORDER BY exec_time";
$result=$mysqli->query($my_query);
$row_number=($result->num_rows);
$col_number=($mysqli->field_count);

$users=[];
$exec_times=[];
$datas=[];

while($rows=mysqli_fetch_array($result)){
	$users[]=$rows['user'];
	$exec_times[]=$rows['exec_time'];
	$datas[]=$rows['date'];
}

##############################################TOP
SUBMISSIONS############################################
?>
<html>
<body>


	<table align="center" border="1px" style="width:800px; line-height:20px;
border-collapse: collapse">
		<tr>
		<th colspan="5"> <h2> TOP submissions </h2></th>
		</tr>
			<th> Ranking </th>
			<th> User </th>
			<th> Execution time </th>
			<th> Date </th>
			<th> Badges </th>
<?php
```

```
for($i=0;$i<$row_number;$i++){
	echo "<tr>";
	echo "<td align='right'>" . ($i+1) . "</td>";
	echo "<td align='right'>" .  $users[$i] . "</td>";
	echo "<td align='right'>" .  $exec_times[$i] . "</td>";
	echo "<td align='right'>" .  $datas[$i] . "</td>";
	if($i<3){
		$type_aux=$i+3;
		echo "<td align='center'> <img src='$type_aux.png' alt=BADGE width='25'
height='25' /> </td>";
	}


}
	echo "</tr>";
?>

	</table>
	<br>
	<br>

	<a href="ad_lab_submissions.php">Student submissions</a>
	<h3><a href="ad_home.php">Go back</a></h3>
	<h3><a href="logout.php">Sign Out</a></h3>
</body>
</html>
```

comentario.php

```
<?php
session_start();

if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}

$user='pag_web_admin';
$pass='1234';
$charset='utf8mb4';

$id_envio=$_GET['id'];
$commentary= $_POST["commentary"];

$mysqli=new mysqli("localhost", $user, $pass, "envios_pca");
$mysqli->query("INSERT INTO comentarios_pca (ID, comentario) VALUES
('$id_envio','$commentary');");
header("Location:ad_lab_submissions.php");
```

crear_badges_index.php

```
<!DOCTYPE html>
<?php
session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}
?>
<html lang="es">
	<head>
		<meta charset="UTF-8">
		<meta name="viewport" content="width=device-width,initial-scale=1.0">
		<meta http-equiv="X-UA-Compatible" content="ie-edge">
	</head>
	<body>
	<h2> CREATE BADGES </h2>
		<form action="crear_badges.php" method="post">
			<input name="description" type="text" placeholder="Badge
description">
			<br><br>
			<input name="type" type="text" placeholder="Badge type">
			<input type="submit" value="Create badge">
		</form>
	<h2> MODIFY BADGES </h2>
		<form action="modificar_badges.php" method="post">
			<input name="ID_badge" type="text" placeholder="Badge ID">
			<br><br>
			<input name="description" type="text" placeholder="New badge
description">
			<br><br>
			<input name="type" type="text" placeholder="New badge type">
			<input type="submit" value="Modify badge">
		</form>
			<div>
			<br><br>
			<h3> Available types: </h3>
			<ul style="list-style-type:none;">
				<!--	<li>1:Red badge <img src='1.png'alt BADGE width='20'
height='20'/></li> -->
				<!--	<li>2:Green badge <img src='2.png'alt BADGE width='20'
height='20'/></li> -->
					<li>1:Red badge</li>
					<li>2:Green badge</li>
					<li>3:Gold badge</li>
					<li>4:Silver badge</li>
					<li>5:Bronze badge</li>
					<li>6:Manual badge</li>
			</ul>
			</div>
```

```
	<table align="left" border="1px" style="width:600px; line-height:20px;
border-collapse: collapse">
			<tr>
				<th colspan="3"><h2> Badge list </h2></th>
			</tr>
				<th> ID </th>
				<th> Description </th>
				<th> Type </th>
		<?php
		$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");
		$result=$mysqli->query("SELECT * FROM badges_pca");

		while($rows=mysqli_fetch_array($result)){

		echo "<tr>";
			echo "<td align='right'>" .  $rows['ID'] . "</td>";
			echo "<td align='left'>" .  $rows['description'] . "</td>";
			echo "<td align='right'>" .  $rows['type'] . "</td>";
		echo "</tr>";
		}
		?>
		</table>
	<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
	<div>
	<a href="ad_home.php">Go back</a>
	<h3> <a href="logout.php">Sign Off</a></h3>
	</div>
	</body>
</html>
```

crear_badges.php

```
<?php

session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}

$user='pag_web_admin';
$pass='1234';
$charset='utf8mb4';


$description= $_POST["description"];
$type= $_POST["type"];

$mysqli=new mysqli("localhost", $user, $pass, "envios_pca");
$mysqli->query("INSERT INTO badges_pca (description, type) VALUES
```

```
('$description','$type');");
header("Location:crear_badges_index.php");
```

modificar_badges.php

```
<?php

session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}

$user='pag_web_admin';
$pass='1234';
$charset='utf8mb4';

$description= $_POST["description"];
$type= $_POST["type"];
$ID= $_POST["ID_badge"];

$mysqli=new mysqli("localhost", $user, $pass, "envios_pca");
$mysqli->query("UPDATE badges_pca SET description='$description', type='$type' WHERE
ID='$ID'");

header("Location:crear_badges_index.php");
```

exercise_mangagement.php

```
<!DOCTYPE html>
<?php
session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}
?>

</body>
	<h1 align='center'> EXERCISE MANAGEMENT </h1>
	<h2> CREATE LAB </h2>
		<form action="create_lab.php" method="post">
			<input name="LabID" type="text" placeholder="Lab number">
			<input name="Labdescription" type="text" placeholder="Lab
description">
			<br>
			<input type="submit" value="Create lab">
		</form>
	<br>
	<h2> CREATE EXERCISE </h2>
		<form action="create_exercise.php" method="post">
```

```
                    <input name="LabID" type="text" placeholder="Lab number">
                    <input name="EjID" type="text" placeholder="Exercise number">
                    <input name="Ejdescription" type="text" placeholder="Exercise
description">
                    <input name="tries" type="text" placeholder="Number of tries before
applying a penalty">
                    <input name="penalty" type="text" placeholder="Penalty to apply(in
seconds)">
                    <br>
                    <input type="submit" value="Create exercise">
             </form>
      <h2> MODIFY LAB </h2>
             <form action="modify_lab.php" method="post">
                    <input name="LabID" type="text" placeholder="Lab number to change">
                    <input name="Labdescription" type="text" placeholder="New lab
description">
                    <br>
                    <input type="submit" value="Modify lab">
             </form>
      <br>
      <h2> MODIFY EXERCISE </h2>
             <form action="modify_exercise.php" method="post">
                    <input name="LabID" type="text" placeholder="Lab number to change">
                    <input name="EjID" type="text" placeholder="Exercise number to
change">
                    <input name="Ejdescription" type="text" placeholder="New
description">
                    <input name="tries" type="text" placeholder="New number of tries">
                    <input name="penalty" type="text" placeholder="New penalty(in
seconds)">
                    <br>
                    <input type="submit" value="Modify exercise">
             </form>
      <br>
      <h2> DELETE LAB </h2>

             <form action="delete_lab.php" method="post">
                    <h4 style="color:red"> Deleting a lab will erase all it's
exercises</h4>
                    <input name="LabID" type="text" placeholder="Lab number to delete">
                    <br>
                    <input type="submit" value="Delete lab">
             </form>
      <br>
      <h2> DELETE EXERCISE </h2>
             <form action="delete_exercise.php" method="post">
                    <input name="LabID" type="text" placeholder="Lab number to delete">
                    <input name="EjID" type="text" placeholder="Exercise number to
delete">
                    <br>
```

```
                    <input type="submit" value="Delete exercise">
                </form>



        <br><br>
        <table align="center" border="1px" style="width:800px; line-height:20px;
border-collapse: collapse">
                <tr>
                <th colspan="5"> <h2> ALL EXERCISES </h2></th>
                </tr>
                        <th> Lab </th>
                        <th> Lab Description </th>
                        <th> Exercise </th>
                        <th> Exercise Description </th>
                        <th> Tries before penalty </th>
                        <th> Penalty (seconds) </th>
<?php

$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");
$my_query="SELECT labs_pca.ID AS lab_ID, labs_pca.description AS lab_description,
ejs_pca.ID as ej_ID, ejs_pca.description AS ej_description, tries, penalty FROM
labs_pca LEFT JOIN ejs_pca ON labs_pca.ID=ejs_pca.ID_LAB ORDER BY lab_ID";
$result=$mysqli->query($my_query);
$row_number=($result->num_rows);
$col_number=($mysqli->field_count);

#for($i=0;$i<$row_number;$i++){
while($rows=mysqli_fetch_array($result)){
        echo "<tr>";
        echo "<td align='right'>" . $rows['lab_ID'] . "</td>";
        echo "<td align='right'>" . $rows['lab_description'] . "</td>";
        echo "<td align='right'>" . $rows['ej_ID'] . "</td>";
        echo "<td align='right'>" . $rows['ej_description'] . "</td>";
        echo "<td align='right'>" . $rows['tries'] . "</td>";
        echo "<td align='right'>" . $rows['penalty'] . "</td>";
        echo "</tr>";

}
?>

        </table>
        <h3><a href="ad_home.php">Go back</a></h3>
        <h3><a href="logout.php">Sign Out</a></h3>

</body>

</html>
```

create_lab.php

```
<?php

session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}

$ID= $_POST["LabID"];
$description=$_POST["Labdescription"];
$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");
$my_query="SELECT * FROM labs_pca WHERE ID='$ID'";
$result=$mysqli->query($my_query);
$row=$result->fetch_row();
if(isset($row)){
	echo "That Lab ID already exists <br>Going back...";
	header("Refresh:3;url=exercise_management.php");
	exit();
}
else{
	$my_query2="INSERT INTO labs_pca(ID,description) VALUES ('$ID','$description')";
	$mysqli->query($my_query2);

	header("Location:exercise_management.php");
}
```

modify_lab.php

```
<?php
session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}
$ID= $_POST["LabID"];
$description=$_POST["Labdescription"];
$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");
$my_query="SELECT * FROM labs_pca WHERE ID='$ID'";
$result=$mysqli->query($my_query);
$row_number=($result->num_rows);
$col_number=($mysqli->field_count);
$row=$result->fetch_row();
if(!isset($row)){
	echo "That Lab ID does not exist <br>Going back...";
	header("Refresh:3;url=exercise_management.php");
	exit();
}
else{
```

```
	$my_query2="UPDATE labs_pca SET description='$description' WHERE ID='$ID'";
	$mysqli->query($my_query2);
	header("Location:exercise_management.php");
}
```

delete_lab.php

```
<?php
session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}
$ID= $_POST["LabID"];
$description=$_POST["Labdescription"];
$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");
$my_query="SELECT * FROM labs_pca WHERE ID='$ID'";
$result=$mysqli->query($my_query);
$row=$result->fetch_row();
if(!isset($row)){
	echo "That Lab ID does not exist <br>Going back...";
	header("Refresh:3;url=exercise_management.php");
	exit();
}
else{
	$my_query2="DELETE FROM labs_pca WHERE ID='$ID'";
	$mysqli->query($my_query2);
	header("Location:exercise_management.php");
}
```

create_exercise.php

```
<?php
session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}
$ID_LAB= $_POST["LabID"];
$ID_EJ= $_POST["EjID"];
$description=$_POST["Ejdescription"];
$tries=$_POST['tries'];
$penalty=$_POST['penalty'];
$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");
$my_query="SELECT * FROM labs_pca WHERE ID='$ID_LAB'";
$result=$mysqli->query($my_query);
$row=$result->fetch_row();
if(!isset($row)){
	echo "That Lab ID does not exist <br>Going back...";
	header("Refresh:3;url=exercise_management.php");
```

```
        exit();
}
$my_query2="SELECT * FROM ejs_pca WHERE ID='$ID_EJ' AND ID_LAB='$ID_LAB'";
$result=$mysqli->query($my_query2);
$row=$result->fetch_row();
if(isset($row)){
        echo "There is already an exercise with that ID for that LAB.<br>Going
back...";
        header("Refresh:3;url=exercise_management.php");
        exit();
}
else{
        $my_query3="INSERT INTO ejs_pca(ID,description,ID_LAB,tries,penalty) VALUES
('$ID_EJ','$description','$ID_LAB','$tries','$penalty')";
        $mysqli->query($my_query3);
        header("Location:exercise_management.php");
}
```

modify_exercise.php

```
<?php
session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
        header("Location:index.html");
}
$ID_LAB= $_POST["LabID"];
$ID_EJ= $_POST["EjID"];
$description=$_POST["Ejdescription"];
$tries=$_POST['tries'];
$penalty=$_POST['penalty'];
$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");

$my_query="SELECT * FROM labs_pca WHERE ID='$ID_LAB'";
$result=$mysqli->query($my_query);
$row=$result->fetch_row();
if(!isset($row)){
        echo "That Lab ID does not exist <br>Going back...";
        header("Refresh:3;url=exercise_management.php");
        exit();
}

$my_query2="SELECT * FROM ejs_pca WHERE ID='$ID_EJ' AND ID_LAB='$ID_LAB'";
$result=$mysqli->query($my_query2);
$row=$result->fetch_row();
if(!isset($row)){
        echo "There is no exercise with that ID for that LAB.<br>Going back...";
        header("Refresh:3;url=exercise_management.php");
        exit();
}
```

```
else{
	$my_query3="UPDATE ejs_pca SET description='$description', tries='$tries',
penalty='$penalty' WHERE ID='$ID_EJ' AND ID_LAB=$ID_LAB";
	$mysqli->query($my_query3);
	header("Location:exercise_management.php");
}
```

delete_exercise.php

```
<?php
session_start();
if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
	header("Location:index.html");
}
$ID_LAB= $_POST["LabID"];
$ID_EJ= $_POST["EjID"];
$mysqli=new mysqli("localhost","pag_web_admin","1234","envios_pca");

$my_query="SELECT * FROM labs_pca WHERE ID='$ID_LAB'";
$result=$mysqli->query($my_query);
$row=$result->fetch_row();
if(!isset($row)){
	echo "That Lab ID does not exist <br>Going back...";
	header("Refresh:3;url=exercise_management.php");
	exit();
}

$my_query2="SELECT * FROM ejs_pca WHERE ID='$ID_EJ' AND ID_LAB='$ID_LAB'";
$result=$mysqli->query($my_query2);
$row=$result->fetch_row();



if(!isset($row)){
	echo "There is no exercise with that ID for that LAB.<br>Going back...";
	header("Refresh:3;url=exercise_management.php");
	exit();
}
else{
	$my_query3="DELETE FROM ejs_pca WHERE ID='$ID_EJ' AND ID_LAB='$ID_LAB'";
	$mysqli->query($my_query3);
	header("Location:exercise_management.php");
}
```

2. Código del servidor y proceso de envío

script.sh

```
#!/bin/bash
#$1=User
#$2=.c file
#$3=Lab
#$4=ej

cd /home/threshold1
su threshold1 -c "sbatch /mnt/pca/PCA_scripts/submit.sh $1 $2 $3 $4"
```

submit.sh

```
#!/bin/bash
#SBATCH --time=00:15:00
#SBATCH --output=/dev/null
#SBATCH --error=/dev/null
                          #DEBUG
##SBATCH --output=c%j.o
##SBATCH --error=c%j.e
                          #/DEBUG
#$1=user
#$2=copia codigo a ejecutar
#$3=Lab
#$4=ej



/usr/bin/python3.5 /mnt/pca/PCA_scripts/python_script.py $1 $2 $3 $4
```

python_script_bueno.py

```
import os
import sys
import subprocess
import shutil
import datetime
import time

#sys.argv[1]=$USER
#sys.argv[2]=CODE_LOCATION
#sys.argv[3]=lab
```

```
#sys.argv[4]=ej

def write_to_db():
    subprocess.run(["/usr/bin/php",
"/mnt/pca/PCA_scripts/update_db.php", *arg_list])
    clean_function()

def clean_function():
    shutil.rmtree(WORKING_DIR)
    exit()

var=sys.argv[2]                                  #String manipulation
para generar el nombre de la carpeta según el envío,
var=var.split("/",5)
var=var[5]
var=var.split(".")


user=sys.argv[1]
lab=sys.argv[3]
ej=sys.argv[4]

date=datetime.datetime.now()
date=date.strftime("%Y-%m-%d.%H:%M:%S")        #Fecha en formato
AÑO-MES-DIA.HORA:MINUTO:SEGUNDO

exec_time=0.0

arg_list=[]
arg_list.append(user)
arg_list.append(lab)
arg_list.append(ej)
arg_list.append(str(exec_time))
arg_list.append(date)

HOME_PATH='/opt/PCA_working_dir'
TMP_PATH='/tmp_pca'
WORKING_DIR=HOME_PATH+TMP_PATH+var[0]
#/home/threshold1/tmp_pca$lab$ej$user$envio/
os.mkdir(WORKING_DIR)
os.chdir(WORKING_DIR)
```

```
code_file=shutil.copyfile(sys.argv[2],WORKING_DIR+'/code_aux.c')


############COMPILACIÓN#############

completed_proc=subprocess.run(["/usr/bin/gcc", "-O2", "code_aux.c",
"-o", "code_aux.x"])
exit_code=completed_proc.returncode

if exit_code!=0:
    arg_list.append("1")
    write_to_db()
arg_list.append("2")

############EJECUCIÓN###############

test_dir="/mnt/pca/ej/lab"+lab+"/ej"+ej
inputexists=0
if os.path.exists(test_dir+"/input"):

    file=open(test_dir+"/input",'r')
    program_input=file.read()
    file.close()
    inputexists=1

try:
    if inputexists:

completed_proc=subprocess.run([WORKING_DIR+"/code_aux.x",program_input],
stdout=subprocess.PIPE,universal_newlines=True,timeout=240)
    else:

completed_proc=subprocess.run([WORKING_DIR+"/code_aux.x"],stdout=subproc
ess.PIPE,universal_newlines=True,timeout=240)
except subprocess.TimeoutExpired:
    arg_list.append("3")
    write_to_db()
file=open(WORKING_DIR+"/code_aux.output","w")
file.write(str(completed_proc.stdout))
file.close()
```

```
##########COMPROBAR_OUTPUT########


completed_proc=subprocess.run(["/usr/bin/diff",
WORKING_DIR+"/code_aux.output",test_dir+"/output"],stdout=subprocess.PIP
E)
exit_code=completed_proc.returncode
if exit_code!=0:
    arg_list.append("4")
    write_to_db()

arg_list.append("5")

#############TIMING############

start_time=time.time()
completed_proc=subprocess.run([WORKING_DIR+"/code_aux.x"],stdout=subproc
ess.PIPE)
exec_time=time.time()-start_time
arg_list[3]=str(exec_time)

#############TESTS############

Dyna_PATH="/mnt/pca/DynamoRIO-ARM-Linux-EABIHF-8.0.0-1/bin32/drrun"
Dyna_DIR="/opt/dynamorio/"+var[0]
os.mkdir(Dyna_DIR)

completed_proc=subprocess.run([Dyna_PATH,"-t","drcachesim","-offline","-
outdir",Dyna_DIR,"--",WORKING_DIR+"/code_aux.x","100",">","/dev/null"],s
tdout=subprocess.PIPE)

for test in os.listdir(Dyna_DIR):

completed_proc=subprocess.run([Dyna_PATH,"-t","drcachesim","-simulator_t
ype","opcode_mix","-indir",Dyna_DIR+"/"+test],stderr=subprocess.PIPE,uni
versal_newlines=True)
    file=open(WORKING_DIR+"/code_aux.instructions","w")
    file.write(str(completed_proc.stderr))
    file.close()
```

```
shutil.rmtree(Dyna_DIR+"/")

for test in os.listdir(test_dir):
    if test.endswith(".sh"):

completed_proc=subprocess.run(["/bin/sh",test_dir+"/"+test,WORKING_DIR])
       arg_list.append(str(completed_proc.returncode))

write_to_db()
```

update_db.php

```
<?php
$user='pag_web_admin';
$pass='1234';
$charset='utf8mb4';

#$argv[0]= Nombre del programa
#$argv[1]= Nombre del usuario de pca
#$argv[2]= Lab
#$argv[3]= Ej
#$argv[4]= exec_time
#$argv[5]= date
#$argv[6++]=badges



$mysqli = new mysqli("192.168.1.55:3306", $user, $pass, "envios_pca");
$my_query="INSERT INTO envios_pca (user, ej, lab, date, exec_time) VALUES ('$argv[1]',
'$argv[3]', '$argv[2]', '$argv[5]', '$argv[4]')";
$mysqli->query($my_query);
$last_id=$mysqli->insert_id;

$my_query2="INSERT INTO rel_envios_badges VALUES";
for ($i=6;$i<$argc-1;$i++){
      $my_query2.="($last_id,$argv[$i]), ";
}
$my_query2.="($last_id,$argv[$i])";
$mysqli->query($my_query2);
```

test1.sh

```
#!/bin/bash
count=0
count=`/bin/cat $1/code_aux.instructions | /bin/grep umull | /usr/bin/awk '{print $1}'`
if [ $count -gt 20000 ]; then
      exit 7
else
      exit 6
fi
```

interactive.sh

```
#!/bin/bash
salloc srun --time=240 --pty bash
```

prolog

```
#!/bin/bash
usuario=$(/usr/bin/id -un $SLURM_JOB_UID)
/bin/mount 192.168.1.55:/home/$usuario /home/$usuario
```

epilog

```
#!/bin/bash
usuario=$(/usr/bin/id -un $SLURM_JOB_UID)
/bin/umount /home/$usuario
```

pi.c

```
#include <memory.h>
#include <stdio.h>
#include <stdlib.h>

int N, N4;
signed char a[25480], b[25480], c[25480];

void DIVIDE( signed char *x, int n )
{
    int j, k;
    unsigned q, r, u;
    long v;
```

```
    r = 0;
    for( k = 0; k <= N4; k++ )
    {
        u = r * 10 + x[k];
        q = u/n;
        r = u - q * n;
        x[k] = q;
    }
}



void LONGDIV( signed char *x, int n )
{
    int j, k;
    unsigned q, r, u;
    long v;

    if( n < 6553 )
    {
        r = 0;
        for( k = 0; k <= N4; k++ )
        {
            u = r * 10 + x[k];
            q = u / n;
            r = u - q * n;
            x[k] = q;
        }
    }
    else
    {
        r = 0;
        for( k = 0; k <= N4; k++ )
        {
            if( r < 6553 )
            {
                u = r * 10 + x[k];
                q = u / n;
                r = u - q * n;
            }
            else
```

```
            {
                v = (long) r * 10 + x[k];
                q = v / n;
                r = v - q * n;
            }
            x[k] = q;
        }
    }
}


void MULTIPLY( signed char *x, int n )
{
    int j, k;
    unsigned q, r, u;
    long v;
    r = 0;
    for( k = N4; k >= 0; k-- )
    {
        q = n * x[k] + r;
        r = q / 10;
        x[k] = q - r * 10;
    }
}

void SET( signed char *x, int n )
{
    memset( x, 0, N4 + 1 );
    x[0] = n;
}

void SUBTRACT( signed char *x, signed char *y, signed char *z )
{
    int j, k;
    unsigned q, r, u;
    long v;
    for( k = N4; k >= 1; k-- )
    {
        if( (x[k] = y[k] - z[k]) < 0 )
        {
            x[k] += 10;
```

```
            z[k-1]++;
        }
    }
    if( (x[k] = y[k] - z[k]) < 0 )
    {
        x[k] += 10;
    }
}

void calculate( void );
void progress( void );
void epilog( void );
int main( int argc, char *argv[] )
{
    N = 10000;

    if( argc > 1 )
        N = atoi(argv[1]);

    setbuf(stdout, NULL);

    calculate();

    epilog();

    return 0;
}
void calculate( void )
{
    int j;

    N4 = N + 4;

    SET( a, 0 );
    SET( b, 0 );

    for( j = 2 * N4 + 1; j >= 3; j -= 2 )
    {
        SET( c, 1 );
        LONGDIV( c, j );
```

```
        SUBTRACT( a, c, a );
        DIVIDE( a, 25 );

        SUBTRACT( b, c, b );
        DIVIDE( b, 239 );
        DIVIDE( b, 239 );

        progress();
    }

    SET( c, 1 );
    SUBTRACT( a, c, a );
    DIVIDE( a, 5 );
    SUBTRACT( b, c, b );
    DIVIDE( b, 239 );
    MULTIPLY( a, 4 );
    SUBTRACT( a, a, b );
    MULTIPLY( a, 4 );
    progress();
}
/*
 N = 10000
 A = 0
 B = 0
 J = 2 * (N + 4) + 1
 FOR J = J TO 3 STEP -2
     A = (1 / J - A) / 5 ^ 2
     B = (1 / J - B) / 239 ^ 2
 NEXT J
 A = (1 - A) / 5
 B = (1 - B) / 239
 PI = (A * 4 - B) * 4
*/
void progress( void )
{
    printf(".");
}

void epilog( void )
{
    int j;
```

```
    {
        fprintf( stdout, " \n3.");
        for( j = 1; j <= N; j++ )
        {
            fprintf( stdout, "%d", a[j]);
            if( j % 5  == 0 )
                if( j % 50 == 0 )
                    if( j % 250  == 0 )
                        fprintf( stdout, "    <%d>\n\n   ", j );
                    else
                        fprintf( stdout, "\n   " );
                else
                    fprintf( stdout, " " );
        }
    }
}
```

crear_usuarios.php

```
<?php
#$argv[0]=usuario a añadir
#$argv[1]=contraseña del usuario
$user='pag_web_admin';
$pass='1234';
$charset='utf8mb4';
$usua=$argv[0]
$contra=password_hash($argv[1],PASSWORD_BCRYPT);
$mysqli=new mysqli("localhost", $user, $pass, "envios_pca");
$mysqli->query("INSERT INTO users_pca (username, password) VALUES
('$usua', '$contra')");
```

Apéndice C: Seguridad

A lo largo de la realización del proyecto la seguridad ha tomado una relevancia que no se había previsto. En este apéndice se detallan las medidas de seguridad tomadas.

1. Clúster y servidor

Como se menciona en el Apéndice A: Instalación, se configura un módulo PAM para evitar la conexión directa por ssh a cualquiera de las placas. De esta forma aunque un usuario acceda a leer la configuración de red, no podrá conectarse a ninguna placa sin permiso del gestor del clúster.
Si se quiere restringir aún más el acceso es posible deshabilitar ssh, aunque esto retiraría también acceso al administrador, y permitiría solo conexión mediante el clúster o minicom.

En el servidor solo el administrador tiene permiso de sudo.
Además las carpetas de la aplicación web y */mnt/pca* tienen los permisos restringidos para que solo el usuario de apache y threshold1 puedan acceder.

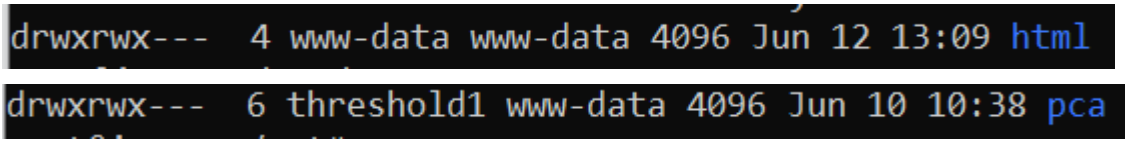


Figura C.1. Permisos de las carpetas de la aplicación web y pca.

En la carpeta de pca además de los scripts de cada laboratorio se guarda temporalmente una copia del código que se sube a la página web, por eso www-data también debe tener permisos de escritura en ese directorio.

Además, el directorio */home/threshold1* se utiliza como directorio temporal para la ejecución de un código enviado a la página web, por lo que también está restringido.

2. Página web

Todos los archivos de código de la página web empiezan comprobando que la sesión esté activa. Esto se hace para evitar que un usuario no registrado pueda acceder a cualquier parte de la web si conoce la dirección.

```php
session_start();
if(empty($_SESSION["user"])){
        header("Location:index.html");
}
```

Figura C.2. Comprobación básica.

Además, aquellas páginas restringidas al administrador hacer una comprobación adicional: que el usuario *loggeado* sea el *admin*.

```php
session_start();

if( (empty($_SESSION["user"])) or ($_SESSION["user"]!="admin")){
        header("Location:index.html");
}
```

Figura C.3. Comprobación de administrador.

Como se hacen consultas a la base de datos también es importante evitar una posible *sql injection* [25].

Todas las consultas a la base de datos están preparadas. Los únicos puntos de entrada son la página de *login* y las herramientas de administrador, que sí recibe parámetros antes de hacer la consulta.

Para las herramientas de administrador no se ha tomado ninguna precaución adicional, para la página de *login* se ha utilizado un mecanismo de php, los PDO que sanitizan la entrada de datos, solo permiten lecturas a la base de datos y están preparados para ser utilizados como medida de protección ante *sql injection*.

Además, todas las contraseñas de la base de datos están *hasheadas* con la función *blowfish* de php, que añade un *salt*, por lo que aunque un atacante pudiera acceder a la base de datos con la lista de contraseñas no le serviría de nada.

Por último, *lab.php* y *ad_lab.php* reciben un POST con el número de laboratorio y ejercicio, para poder generar de forma dinámica las páginas con los ejercicios. En este caso, como se podría hacer un POST con datos incorrectos o maliciosos, se hace una comprobación adicional.

```php
session_start();
if(empty($_SESSION["user"])){                                   #Security check, is user logged?
        header("Location:index.html");
}
$lab=$_GET['lab'];
$ej=$_GET['ej'];
if(!is_numeric($lab) or !is_numeric($ej)){                      #Security check, if lab or ej are not a number then destroy session
        session_destroy();
        header("Location:index.html");
}
$_SESSION["lab"]=$lab;
$_SESSION["ej"]=$ej;
$dbuser='pag_web_admin';
$pass='1234';
$charset='utf8mb4';

$dsn="mysql:host=localhost;dbname=envios_pca;charset=$charset";         #Security check, if lab or ej does not exist then destroy session
$pdo= new PDO($dsn,$dbuser,$pass);
$statement=$pdo->prepare("SELECT * FROM ejs_pca WHERE ID = ? AND ID_LAB = ? ");
$statement->execute([$ej,$lab]);
$result=$statement->fetch();
if(!isset($result[0])){
        session_destroy();
        header("Location:index.html");
}
```

Figura C.4. Comprobación de seguridad de lab.php y ad_lab.php

3. Proceso de envío

Para el proceso de envío de código desde la página web descrito en el apartado de diseño también ha sido necesario tomar ciertas medidas de precaución.

En primer lugar, el script *upload.php* se asegura que el archivo que se ha subido sea un archivo de código acabado en '.c'.

Dada la necesidad de hacer el cambio de usuario antes de ejecutar código, es necesario dar permisos de sudo a www-data. Para evitar dar permisos completos, se ha creado un script que hará de *handler*. www-data solo tendrá permisos de sudo para ejecutar este script, que además está en el directorio de */mnt/pca*, por lo que ningún usuario puede leerlo o modificarlo.

```
www-data ALL=(ALL) NOPASSWD: /mnt/pca/PCA_scripts/script.sh
```

Figura C.5. Permisos de sudo solo para ejecutar un script.

Finalmente, a la hora de ejecutar el código que se ha enviado, se establece un *timeout* configurable, para reducir la posibilidad de que se haya subido código malicioso.

Apéndice D: Manual de uso

En este apéndice se muestra una guía para utilizar todas las funciones de las que dispone la aplicación web.

La primera página a la que accede el administrador es la de *home*.

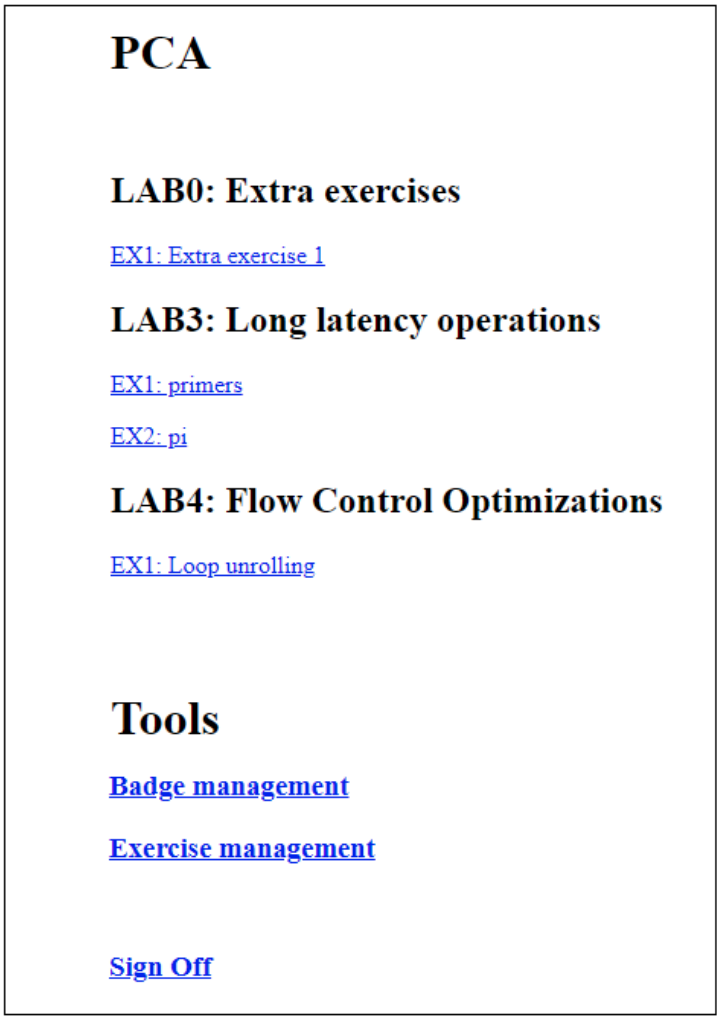


Figura D.1. Página de home de administrador.

Desde aquí se puede acceder a las diferentes herramientas o a cada ejercicio que se haya creado.
Cada página de ejercicio enlaza con la respectiva *student submissions*, que muestra todos los envíos de los alumnos para esa actividad en concreto. Además permite añadir comentarios a determinado envío, o medallas manuales. Para ello es necesario especificar la ID de la medalla.

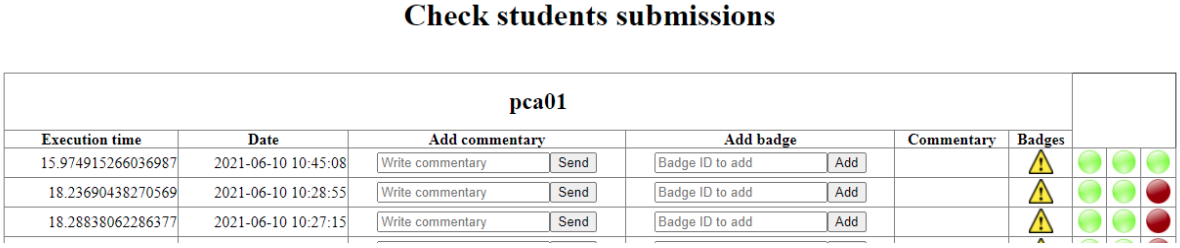


Figura D.2. Funciones de añadir comentarios y medallas.

Para comprobar las ID de las medallas, o crear nuevas se puede utilizar la herramienta *Badge management*.

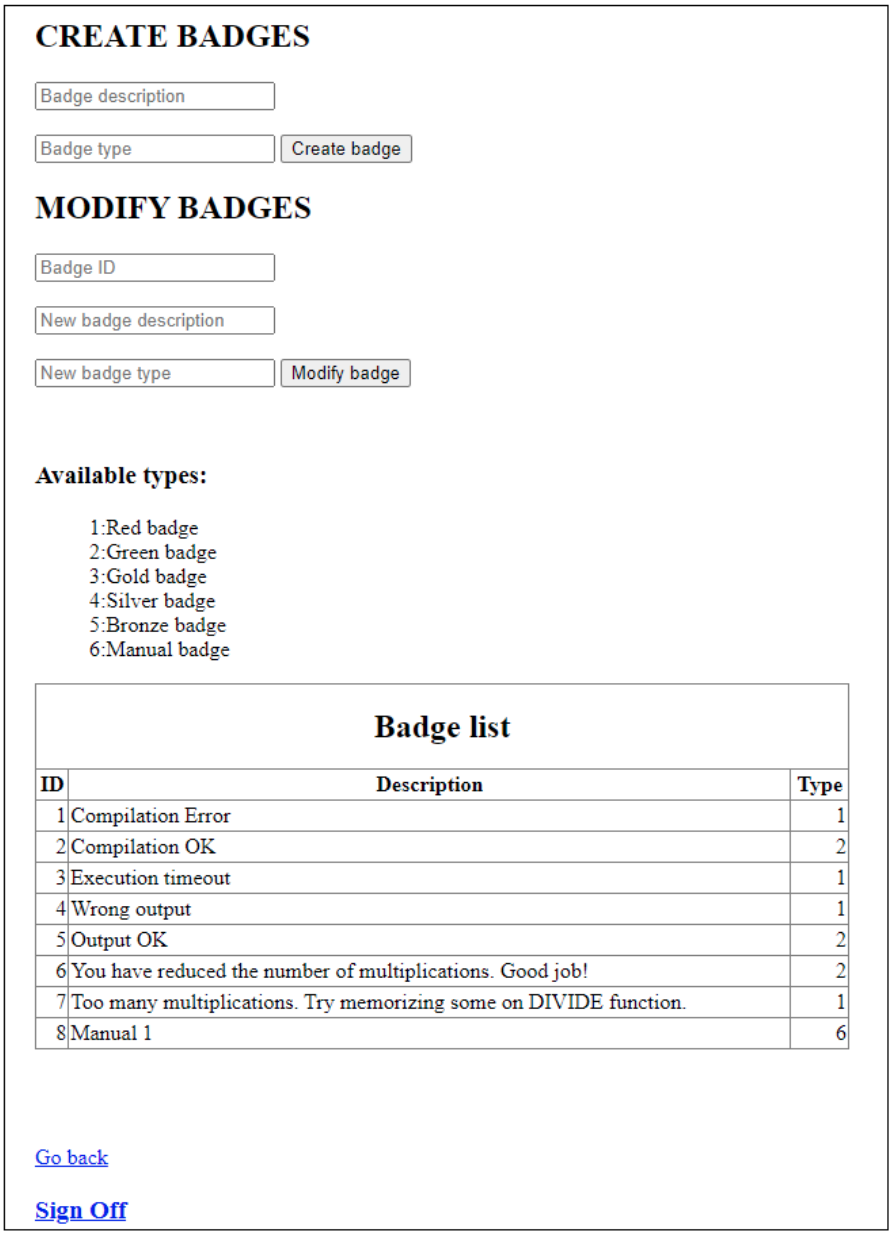


Figura D.3. Herramienta badge management

Desde esta página se pueden crear nuevas o modificar medallas ya existentes.

El tipo de la medalla indica qué icono se verá visualmente.
Además de esto, el tipo es importante para calcular el número de medallas que tiene un ejercicio asociado.
Es decir, un ejercicio puede tener 3 grupos de medallas:

- Medallas fijas: Corresponden a las que son comunes para todos los ejercicios. Son las que tienen ID 1,2,3,4 y 5.
- Medallas variables: Corresponden a aquellas que se otorgan según los test de cada laboratorio. En el ejemplo de la figura D3.3 tienen ID 6 y 7.
- Medallas manuales: Medallas que no corresponden estrictamente a ningún ejercicio ni a ningún test, se pueden otorgar bajo criterio del profesorado desde la página de student submissions.

Para calcular el ranking se comprueba el número de medallas de tipo distinto a 1 y 6, es decir: No se tienen en cuenta las medallas manuales otorgadas ni los tests que han dado un resultado negativo.

Por eso a la hora de crear los test es importante que devuelvan en caso erróneo una medalla de tipo 1.

Para añadir un test hay que crear un archivo `.sh' en el directorio del ejercicio correspondiente. Es decir, para el ejercicio 2 del laboratorio 3 el directorio sería: */mnt/pca/ej/lab3/ej2/*

Se pueden añadir tantos test como se desee.

En este directorio también se pueden añadir el *output* con el que se quiere comparar la ejecución del código para ver si es correcto, o una serie de parámetros de entrada, en un archivo *input*.
El archivo *input* es opcional.

```
#!/bin/bash
count=0
count=`/bin/cat $1/code_aux.instructions | /bin/grep umull | /usr/bin/awk '{print $1}'`
if [ $count -gt 20000 ]; then
        exit 7
else
        exit 6
fi
```

Figura D.4. Ejemplo de test.

Los *exit code* indican la ID de la medalla que se otorgará.
En la figura D.5 hay un esquema del proceso a seguir, según si la medalla a crear es manual o de test.

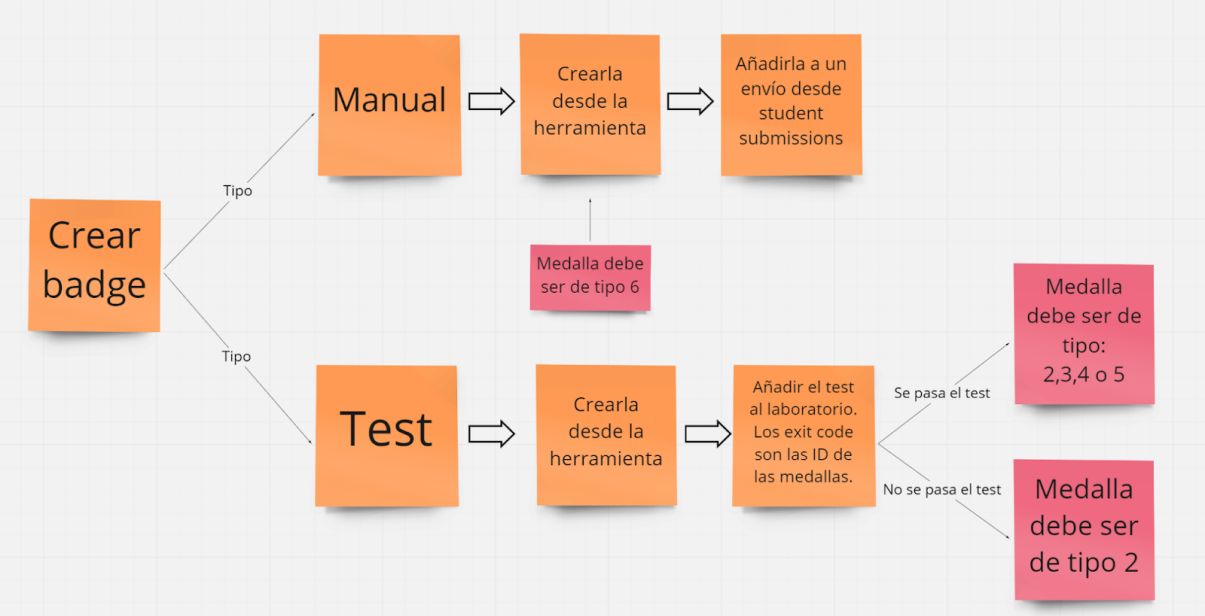


Figura D.5. Proceso de creación de medallas.

Otra herramienta es la de *exercise management*, visible en la figura D.6.

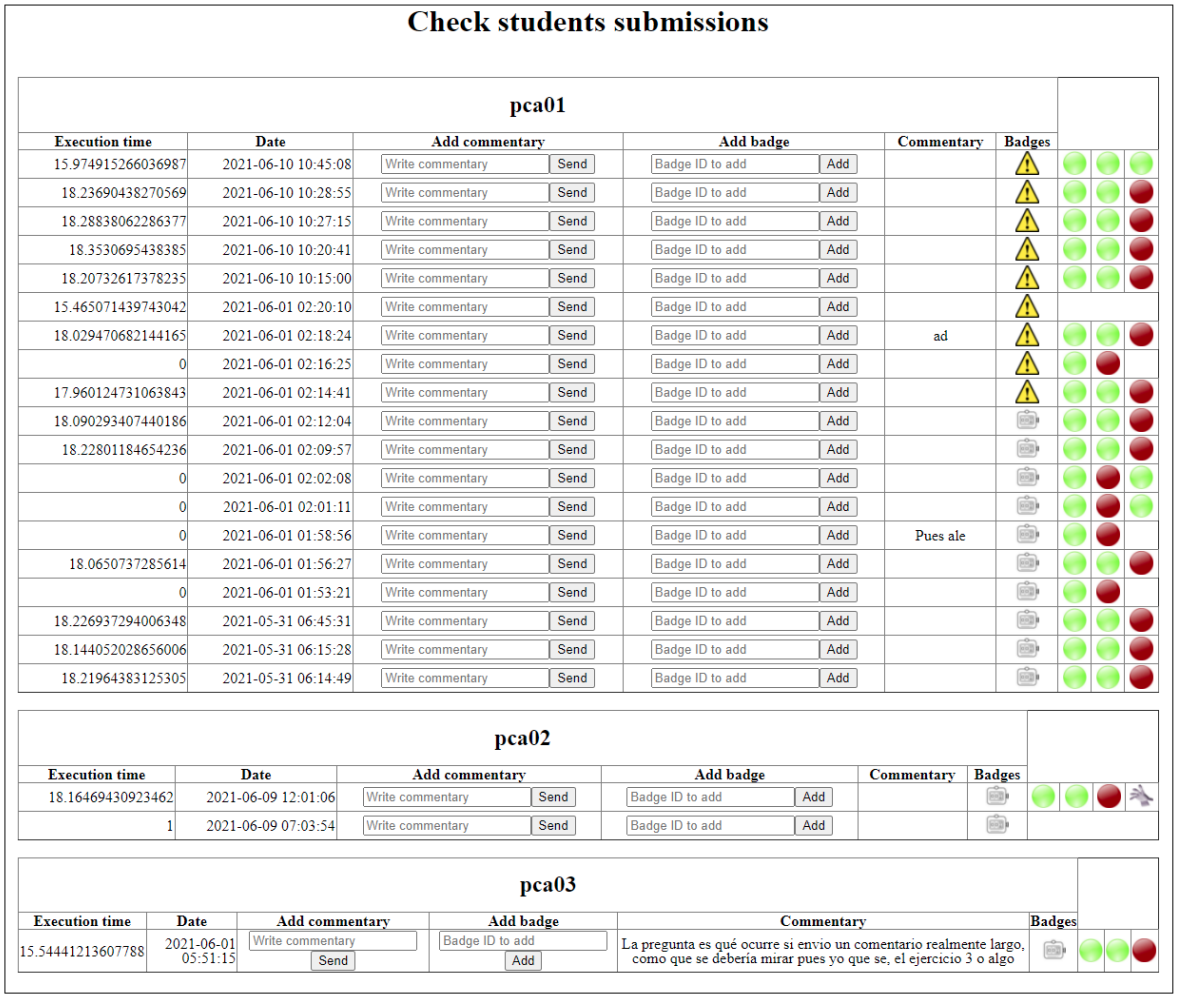


Figura D.6. Herramienta de exercise management.

Desde esta página se pueden añadir, eliminar o modificar los ejercicios y laboratorios especificados.

Aunque se puede alterar el número de intentos máximos antes de aplicar una penalización y el valor de la misma, alterarlos no afecta a la base de datos de forma retroactiva.

Bibliografía

[1] Programació Conscient de l'Arquitectura, guia docent.
https://www.fib.upc.edu/ca/estudis/graus/grau-en-enginyeria-informatica/pla-destudis/assignatures/PCA
[en línea] (consulta: 24/02/2021)

[2] What is an FPGA, Xilinx
https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html
[en línea] (consulta: 24/02/2021)

[3] Zedboard Zynq 7000, imagen.
https://www.xilinx.com/products/boards-and-kits/1-elhabt.html.html
[en línea] (consulta: 24/02/2021)

[4] Noticias UPC
https://www.upc.edu/es/sala-de-prensa/noticias/comunicados-en-relacion-a-la-covid-19
[en línea] (consulta: 24/02/2021)

[5] Scrum
https://proyectosagiles.org/que-es-scrum/
[en línea] (consulta: 1/03/2021)

[6] Atenea UPC
https://atenea.upc.edu/
[en línea] (consulta: 27/02/2021)

[7] Google Drive
https://drive.google.com
[en línea] (consulta: 27/02/2021)

[8] Trello
https://trello.com
[en línea] (consulta: 27/02/2021)

[9] Gantt
https://www.gantter.com/
[en línea] (consulta: 6/03/2021)

[10] Payscale, project manager engineering
https://www.payscale.com/research/ES/Job=Project_Manager%2C_Engineering/Salary
[en línea] (consulta: 13/03/2021)

[11] Payscale, project manager engineering
https://www.payscale.com/research/ES/Job=Hardware_Engineer/Salary
[en línea] (consulta: 13/03/2021)

[12] Pccomponentes
https://www.pccomponentes.com/pccom-basic-essential-intel-core-i3-10100-8gb-480gbssd
[en línea] (consulta:13/03/2021)

[13] Switch
https://www.amazon.es/TP-LINK-TL-SG105-1000Mbps-inoxidable-configuraci%C3%B3n/
[en línea] (consulta:13/03/2021)
[14] Idealista, alquiler en Barcelona
https://www.idealista.com/ca/inmueble/93219413/
[en línea] (consulta:13/03/2021)
[15] Pepephone, ISP
https://www.pepephone.com/
[en línea] (consulta:13/03/2021)
[16] BOE ley orgánica de protección de datos
https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673
[en línea] (consulta:27/05/2021)
[17] BOE ley propiedad intelectual
https://www.boe.es/buscar/act.php?id=BOE-A-1996-8930
[en línea] (consulta:27/05/2021)
[18] Definición daemon
https://en.wikipedia.org/wiki/Daemon_(computing)
[en línea] (consulta:16/05/2021)
[19] Guía de instalación de slurm
https://slurm.schedmd.com/quickstart_admin.html
[en línea] (consulta:16/05/2021)
[20] DynamoRIO
https://dynamorio.org/
[en línea] (consulta:12/06/2021)
[21] Gamificación
https://rua.ua.es/dspace/bitstream/10045/49303/1/Gamificad-insensatos_JENUI2015.pdf
[en línea] (consulta:13/06/2021)
[22] Hash function
https://en.wikipedia.org/wiki/Hash_function
[en línea] (consulta:17/06/2021)
[23] Salt
https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/
[en línea] (consulta:15/06/2021)
[24] Web archive pin ARM
https://web.archive.org/web/20130311084720/http://software.intel.com/sites/landingpage/pintool/downloads/pin-2.0-5567-gcc.3.3.1-softfp-arm-linux.tar.gz
[en línea] (consulta:13/06/2021)
[25] PHP pdo tutorial
https://phpdelusions.net/pdo
[en línea] (consulta:13/06/2021)