

PLATAFORMA PARA EL APOYO A LA EVALUACIÓN DE ESTUDIANTES EN CURSOS DE PROGRAMACIÓN

MARÍA CRISTINA PORTILLA CORTÉS
maria.portilla.cortes@correounivalle.edu.co



Escuela de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Valle
Cali, Colombia

Mayo 2018

PLATAFORMA PARA EL APOYO A LA EVALUACIÓN DE ESTUDIANTES EN CURSOS DE PROGRAMACIÓN

MARÍA CRISTINA PORTILLA CORTÉS
201630799
maria.portilla.cortes@correounivalle.edu.co

Trabajo de grado para optar al título de
Ingeniero de Sistemas

Directores

John Alexander Sanabria Ordóñez
john.sanabria@correounivalle.edu.co

Daniel Barragán Calderón
daniel.barragan@correounivalle.edu.co



Escuela de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Valle
Cali, Colombia

Mayo 2018

PROYECTO DE GRADO

ÍNDICE GENERAL

1. ESTRUCTURA DEL DOCUMENTO	1
2. INTRODUCCIÓN	2
2.1. Descripción del Problema	2
2.2. Descripción de la Solución	2
3. OBJETIVOS	4
3.1. Objetivo General	4
3.1.1. Objetivos Específicos	4
4. MARCO TEÓRICO	5
4.1. Virtualización	5
4.1.1. Técnicas de Virtualización	5
4.1.2. Tipos de virtualización	6
4.2. Software para evaluación automática	7
4.2.1. Dificultades en la enseñanza y aprendizaje de los cursos de programación	7
4.2.2. Herramientas y entornos para el apoyo en los cursos de programación	8
4.3. Arquitectura Orientada a Servicios	9
4.3.1. Servicios Web	11
4.3.2. Especificaciones para la orquestación de <i>Web Services</i>	13
4.3.3. Arquitectura REST	14
4.4. Mensajería	16
4.4.1. Protocolo AMQP	17
4.5. Supervisión de Contenedores	18
5. ESTADO DEL ARTE	20
5.1. Virtualización	20
5.2. Software para evaluación automática	24
5.3. Arquitectura Orientada a Servicios	27
5.4. Mensajería	31
5.5. Supervisión de Contenedores	32
6. TECNOLOGÍAS USADAS	35
6.1. Selección de Tecnología de Virtualización	35
6.1.1. Comparación de <i>hypervisors vs. contenedor</i>	35
6.1.2. Docker	36
6.1.3. CoreOS RKT	39
6.1.4. Docker vs. Coreos RKT	41
6.2. Selección de Software para evaluación automática	42
6.2.1. DOMjudge	42
6.2.2. UVA Online Judge	42
6.2.3. DOMjudge vs. UVA Online Judge	43
7. ARQUITECTURA DE LA SOLUCIÓN	46

7.1. Arquitectura del Front-end	46
7.2. Arquitectura del Back-end	56
8. RESULTADOS	59
8.1. Resultados Obtenidos	59
8.2. Descripción de un conjunto de Ejercicios	59
8.2.1. Ejercicio 1	60
8.2.2. Ejercicio 2	60
8.3. Configuración del Juez Virtual para la caracterización del proceso de aprendizaje	62
8.4. Resultados de un grupo de estudiantes	68
9. CONCLUSIONES	74
10. TRABAJO FUTURO	76
BIBLIOGRAFÍA	77

ÍNDICE DE FIGURAS

1.	Arquitectura SOA	10
2.	Representación y acceso de un recurso	15
3.	Broker AMQP	18
4.	Ejemplo de Petición/Respuesta con GraphQL	30
5.	Arquitectura de Docker	38
6.	Flujo de ejecución de RKT	40
7.	Vista General	46
8.	Página Inicial	47
9.	Inscripción de los estudiantes a los cursos activos	48
10.	Interfaz de Ingreso	49
11.	Perfil de Administrador	50
12.	Perfil de Administrador: Crear usuario	51
13.	Perfil de Administrador: Crear copias de seguridad	52
14.	Perfil de Profesor	53
15.	Perfil de Profesor: Cursos a cargo	54
16.	Perfil de Profesor: Crear curso	55
17.	Diagrama de Despliegue	56
18.	Diagrama de Componentes	57
19.	Diagrama de Casos de Uso	58
20.	Captura de la información de los contenedores de cada grupo.	63
21.	Lenguajes Python y Python Functional habilitados para el grupo 10.	64
22.	Lenguajes Python y Python Functional habilitados para el grupo 13.	64
23.	Equipos registrados para el grupo 10.	65
24.	Resumen de envíos por equipos para el grupo 10.	65
25.	Equipos registrados para el grupo 13.	66
26.	Resumen de envíos por equipos para el grupo 13.	66
27.	Concurso asociado al ejercicio 2 para el grupo 10.	67
28.	Resumen de evaluaciones por equipos del grupo 10 para el ejercicio 2.	67
29.	Concurso asociado al ejercicio 1 para el grupo 13.	68
30.	Resumen de evaluaciones por equipos del grupo 13 para el ejercicio 1.	68
31.	Número de intentos en el ejercicio 1	71
32.	Número de intentos en el ejercicio 2	71
33.	Porcentaje de soluciones subidas para el ejercicio 1	72
34.	Porcentaje de soluciones subidas para el ejercicio 2	72
35.	Porcentaje de soluciones presentadas para el ejercicio 1 vs. ejercicio 2	73

GLOSARIO

- bajo acoplamiento** Permite que los archivos y directorios de sistemas de archivos separados, conocidos como ramas, se superpongan de forma transparente, formando un único sistema de archivos coherente.. [28](#)
- broker de mensajería** Permite que los archivos y directorios de sistemas de archivos separados, conocidos como ramas, se superpongan de forma transparente, formando un único sistema de archivos coherente.. [32](#)
- cgroups** Es parte del subsistema del kernel. Permite asignar recursos de la máquina como CPU, memoria, ancho de banda o combinaciones entre estos a un conjunto de procesos con una configuración de parametros para uno o más subsistemas. De esta manera, los recursos hardware son divididos apropiadamente entre los usuarios y procesos para incrementar su eficiencia. La mayoría de contenedores usan cgroups como mecanismo subyacente para la gestión de recursos. Además, es posible monitorear los cgroups creados, denegar a un cgroup acceso a los recursos y reconfigurar un cgroup que está en ejecución y establecer predefinidos para hacerlos persistentes a los reinicios del sistema [\[72\]](#).. [7](#)
- gamificación** Uso de técnicas y elementos característicos usados en juegos y actividades de ocio con el objetivo de motivar el aprendizaje.. [24](#)
- sistema anfitrión** Permite que los archivos y directorios de sistemas de archivos separados, conocidos como ramas, se superpongan de forma transparente, formando un único sistema de archivos coherente.. [34](#)
- union file systems** Permite que los archivos y directorios de sistemas de archivos separados, conocidos como ramas, se superpongan de forma transparente, formando un único sistema de archivos coherente.. [38](#)

ACRÓNIMOS

ACI Application Container Image. [41](#)
ACS App Container Specification. [39](#)
AMQP Advanced Message Queuing Protocol. [17](#)
API Application Programming Interface. [29](#)
APPC App Container. [41](#)

CRUD Create, Read, Update and Delete. [29](#)

HIBE Hierarchical identity based Encryption. [31](#)
HTTP Hypertext Transfer Protocol. [12](#)

IBE Identity based Encryption. [31](#)
IoT Internet of Things. [16](#)
IPC Inter-Process Communication. [23](#)
ISO International Organization for Standarization. [23](#)

JSON JavaScript Object Notation. [11](#)

LMS Learning Management Systems. [9](#)

MOM Message Oriented Middleware. [17](#)
MOOC Massive Open Online Course. [25](#)
MQTT Message Queuing Telemetry Transport. [32](#)

POO Programación orientada a objetos. [8](#), [26](#)

QoS Quality of Service. [22](#)

SBC Single Board Computer. [34](#)
SO Sistema Operativo. [36](#)
SOAP Simple Object Access Protocol. [11](#)

UDDI Universal Description, Discovery and Integration. [12](#)

VDI Virtual Desktop Infrastructure. [20](#)
VM Virtual Machine. [6](#)

WS-CDL Web Services Choreography Description Language. [13](#)
WS-I Web Service Interoperability. [11](#)
WSDL Web Service Description Language. [12](#), [13](#)
WSFL Web Services Flow Language. [13](#)

XLANG XML Language. [13](#)
XML Extensible Markup Language. [11](#)

ESTRUCTURA DEL DOCUMENTO

En este documento se propone una solución para el problema planteado en el capítulo anterior. Recolecta la base teórica y tecnológica para el desarrollo de la aplicación. Plantea una arquitectura y finalmente, se responde a la pregunta orientadora del trabajo de grado.

Este documento consta de 8 capítulos descritos a continuación:

- **Introducción:** Plantea el problema que da origen a este trabajo de grado y da una corta explicación de la solución propuesta.
- **Objetivos:** Expone los objetivos de este proyecto.
- **Marco Teórico:** Aborda todos los conceptos teóricos que están relacionados con el desarrollo de este proyecto.
- **Estado del Arte:** Cita las herramientas actuales desarrolladas en base a los conceptos del marco teórico.
- **Tecnologías usadas:** Profundiza en las herramientas tecnológicas usadas para la implementación de la solución y la justificación de por qué han sido seleccionadas.
- **Arquitectura de la Solución:** Proporciona bocetos de las interfaces gráficas y diagramas de diferentes vistas arquitectónicas con el objetivo de describir la estructura de la solución implementada.
- **Resultados:** Argumenta cómo los elementos brindados por el desarrollo de la solución pueden apoyar el proceso de aprendizaje durante los cursos. Se anexa un manual de usuario del uso de la aplicación.
- **Conclusiones:** Reúne las ideas finales acerca del proceso y los resultados del desarrollo de este proyecto.

INTRODUCCIÓN

2.1 DESCRIPCIÓN DEL PROBLEMA

El incremento cada año en los cupos para el ingreso a las carreras de ingeniería genera problemas en la capacidad de los docentes para brindar la continua realimentación que necesitan los estudiantes en el proceso de aprendizaje de un lenguaje de programación. Esto se evidencia en los estudiantes que no presentan trabajos, cancelan el curso o que realizan fraude en las entregas programadas. Actualmente, no se cuenta con datos históricos acerca del desempeño de los estudiantes en los cursos de programación como lo son: notas obtenidas en talleres, proyectos y evaluaciones, y si fueron alcanzados los objetivos establecidos para estas actividades. En los sistemas de registro de notas de la Universidad del Valle, el profesor solo debe ingresar una nota final por estudiante. Por tanto, cada semestre se pierde información importante sobre los procesos de aprendizaje, la cual podría ser tenida en cuenta para la toma de decisiones sobre los programas de los cursos.

En la Universidad del Valle se realizó un trabajo de grado[58] para la integración del juez virtual Domjudge con Moodle, pero a la fecha no ha sido puesto en producción debido a que requiere una actualización de Moodle e instalación de plugins y este tipo de integraciones pone en alto riesgo la estabilidad del campus virtual.

Por lo tanto, se propone el desarrollo de una plataforma independiente para la evaluación en cursos de programación, que ofrece elementos para acompañar los procesos de aprendizaje promoviendo una mejoría en la adquisición de competencias y sea un medio para obtener datos estadísticos acerca de los temas y subtemas que presentan más dificultad para los estudiantes, junto a la recopilación de código fuente que puede ser empleado para minería de datos por trabajos futuros.

De esta forma, la pregunta orientadora de este trabajo de grado es ¿Cómo puede una plataforma mejorar el aprendizaje de los estudiantes a través de la realimentación inmediata de sus actividades y el análisis de datos históricos de su desempeño?

2.2 DESCRIPCIÓN DE LA SOLUCIÓN

Se realizó la implementación de una aplicación que permite a los docentes usar una herramienta juez virtual para acompañar los procesos académicos mediante el registro de información básica de sus cursos a través de una interfaz. Este juez está alojado en Internet y los estudiantes pueden consultar los cursos disponibles y registrarse en ellos.

El docente tiene la posibilidad de crear, consultar y eliminar sus cursos, como también descargar un comprimido con la base de datos del juez virtual que contiene los registros recolectados de los talleres propuestos.

La aplicación consta de tres perfiles:

- **Perfil del docente:** En este se presenta la interfaz para la creación de un curso y para ver los cursos a cargo.
- **Perfil Público:** Muestra los cursos disponibles con la opción de inscribirse a ellos. Evitando que el docente tenga que ingresar a sus estudiantes directamente a través del juez.
- **Perfil del administrador:** Realiza la operación de crear los docentes y recolectar la base de datos de los cursos.

OBJETIVOS

3.1 OBJETIVO GENERAL

Desarrollar una plataforma para la creación por demanda de ambientes virtuales de evaluación empleando un juez virtual de código abierto.

3.1.1 *Objetivos Específicos*

1. Seleccionar un juez virtual y una tecnología de virtualización para la creación de los entornos virtuales (Máquinas virtuales ó Contenedores virtuales).
2. Diseñar la arquitectura del software (frontend, backend).
3. Desarrollar la lógica del frontend para la recopilación de información sobre el entorno virtual a crear y la adición de estudiantes a un curso.
4. Desarrollar la lógica del backend para la creación de los entornos virtuales.
5. Caracterizar un proceso de aprendizaje (examen, laboratorio ó actividad de repaso) a gestionar en la plataforma por medio del juez virtual seleccionado y la información a almacenar
6. Realizar pruebas de la aplicación desarrollada sobre el proceso caracterizado en el objetivo anterior y crear un manual de usuario para el manejo de la plataforma.

MARCO TEÓRICO

En este capítulo se exponen los conceptos teóricos necesarios para el desarrollo de este proyecto. Consta de cinco secciones. Iniciando con el concepto de virtualización, tratando los tipos y técnicas. La segunda sección corresponde a software para la evaluación automática y apoyo los estudiantes en su aprendizaje tratando los problemas que se presentan en el docente y sus alumnos, exponiendo cómo las herramientas de este tipo mejoran parcial o complemente estos inconvenientes. En la tercera, se dedica a la arquitectura orientada a servicios, enfatizando en el tipo de arquitectura REST. La cuarta sección realiza una breve investigación acerca de los corredores de mensajes, centrándose en el protocolo AMQP. Finalmente, se aborda el tema de monitoreo de contenedores.

4.1 VIRTUALIZACIÓN

Consiste en la creación de un recurso tecnológico a través de software. Permite que múltiples sistemas operativos se ejecuten en la misma máquina con el objetivo de maximizar el uso de la estructura física. La virtualización se lleva a cabo por medio de un software llamado hipervisor, este se encarga de gestionar los procesos necesarios para crear una o más máquinas virtuales que simulan una plataforma independiente de hardware con su sistema operativo que se ejecuta como si estuviera instalado en la máquina física en donde la ejecución de las aplicaciones es aislada sin afectar a otros procesos.[106]

La virtualización no solo se usa para dividir los recursos. También es posible combinar varios recursos físicos en uno virtual.

4.1.1 Técnicas de Virtualización

■ Full Virtualización

Se realiza mediante la traducción a nivel binario de los comandos que envía el sistema operativo invitado. El hipervisor provee cada característica relevante que ofrece la plataforma física con una completa simulación de hardware. Por lo tanto, este y las aplicaciones en ejecución no necesitan ser modificadas, ya que están completamente desacopladas del hardware por la capa de virtualización. Ofrece el mejor aislamiento y seguridad para máquinas virtuales.[44]

■ Paravirtualización

En esta técnica, el sistema operativo tiene conocimiento que esta siendo virtualizado y es modificado para eliminar la traducción binaria. El hipervisor ofrece interfaces de hiper-llamadas para los procesos como manejo de memoria y gestión de subprocesos

que se comunican directamente con la capa de virtualización. Presenta una interfaz software que es similar pero no idéntica al hardware subyacente.[118]

■ Virtualización Híbrida

Es la combinación de ambas técnicas. En donde partes del sistema operativo usa paravirtualización para ciertos controladores y el sistema anfitrión usa full virtualización para otras características. Esta técnica frecuentemente produce un rendimiento superior en el sistema operativo invitado sin la necesidad de ser completamente paravirtualizado.[65]

4.1.2 Tipos de virtualización

■ Virtualización basada en hipervisor

Un hipervisor, llamado también administrador de máquina virtual. Es una abstracción de hardware físico que permite que múltiples sistemas operativos invitados compartan el hardware de la máquina como red, CPU y unidades de almacenamiento, de esta manera cada sistema operativo invitado obtiene recursos.

El hipervisor interactúa con el sistema operativo anfitrión o hardware para controlar el procesador y los recursos. Asignando lo que necesita para cada máquina virtual y supervisando que las máquinas virtuales no se interrumpan entre sí. Sin embargo, se introduce una sobrecarga ya que replica el comportamiento del hardware. [144]

Virtualizar el hardware entero requiere que cada máquina virtual tenga un sistema operativo completo para su instancia. Además, tiene una considerable sobrecarga en términos de I/O [144].

■ Virtualización a nivel de Sistema Operativo

También llamada *Virtualización basada en contenedores*. Es una arquitectura que promete ser una alternativa ligera a la arquitectura basada en hipervisor. Está caracterizada por usar *userspaces* para aislar y dar permisos a los usuarios de ese contenedor. Es usada exitosamente en aplicaciones donde la máxima eficiencia en la utilización de recursos es requerida.[42].

Un contenedor es un entorno de ejecución aislado y autocontenido. Posee recursos como CPU, memoria, dispositivos I/O y red. Puede compartir archivos y directorios con otros contenedores. Para comunicarse, el anfitrión y los contenedores forman una subred. Se requiere una configuración especial para que el contenedor pueda ser accedido desde afuera. Es percibido por el usuario como si fuera una máquina virtual. Sin embargo, es más liviano y ahorra recursos respecto a una VM. En un mismo equipo físico, por cada máquina virtual se pueden ejecutar de seis a ocho contenedores.[94] El cuadro ?? presenta una comparación entre las características de máquinas virtuales y contenedores.

Los contenedores comparten su kernel mientras el hipervisor trabaja en el nivel de abstracción de hardware. Por esto se considera que los contenedores tienen aislamiento débil respecto a la virtualización basada en hipervisor.

Es posible restringir los recursos usados por cada contenedor, los cuales proveen una justa distribución de todos los recursos disponibles de una sola máquina. Este administrador de recursos es normalmente logrado por el *cgroups* o *control groups*, el cual hace restricciones en el uso de recursos por procesos de grupo. [42].

Para garantizar el aislamiento de recursos entre el sistema anfitrión y los contenedores en ejecución, se implementa el *espacio de nombres*. El espacio de nombres permite a los diferentes procesos tener una diferente vista del sistema. [46]

Las implementaciones más comunes son Linux-Vserver [114], OpenVZ [115] y LXC [131]. Todas están basadas en implementaciones Linux con aspectos similares en cuanto a la seguridad, aislamiento y rendimiento. La principal diferencia entre ellos es la administración de recursos.

4.2 SOFTWARE PARA EVALUACIÓN AUTOMÁTICA

La enseñanza y aprendizaje de la programación es un proceso difícil. Sobre todo, cuando el estudiante no ha tenido ninguna experiencia en el área. Esto se ve reflejado en la alta tasa de estudiantes que han reprobado las asignaturas de introducción a la programación. Debido a esto, es relevante para el docente buscar estrategias y herramientas que intentan facilitar el proceso y disminuir las dificultades en el transcurso de los cursos, durante y fuera del horario de clases. [74] En esta sección se mencionan las dificultades en el proceso de aprendizaje-enseñanza de programación y las herramientas más comunes que intentan solventarlos.

4.2.1 Dificultades en la enseñanza y aprendizaje de los cursos de programación

El objetivo de los cursos de introducción a la programación es iniciar al estudiante en la resolución de problemas algorítmicos usando un paradigma y un lenguaje de programación. Sin embargo, la enseñanza y aprendizaje de programación no es una tarea sencilla, presentando inconvenientes para docentes y estudiantes. [40]

Los inconvenientes más comunes en la enseñanza de programación respecto a los estudiantes son:

- Dificultad con la sintaxis y semántica del lenguaje de programación.
- No han sido entrenados previamente para analizar problemas, desarrollar algoritmos, escribir código, depurar y corregir errores.
- Dificultad al entender conceptos que no corresponden a objetos en el mundo real.
- Uso de lenguaje técnico que dificulta su entendimiento a los estudiantes.

Respecto a los métodos de enseñanza:

- Algunos docentes se enfocan más en la sintaxis del lenguaje de programación en vez de desarrollar habilidades para la solución de problemas en sus estudiantes.

- Usar la lectura como único medio de aprendizaje y no enfatizar en la práctica de la teoría.
- Falta de realimentación constante para los estudiantes durante la clases y elementos de apoyo por fuera del aula.
- Ausencia de atención personalizada cuando el numero de estudiantes sobrepasan la capacidad del docente.

4.2.2 Herramientas y entornos para el apoyo en los cursos de programación

Para facilitar el aprendizaje es adecuado buscar un entorno favorable. Durante el curso, es necesario integrar herramientas que apoyan la enseñanza y el aprendizaje. Estas herramientas tienen la funcionalidad de evaluar y dar la realimentación de los ejercicios subidos, establecer la comunicación entre el docente y sus estudiantes cuando estos tienen inconvenientes en su aprendizaje, promover la comunicación entre los estudiantes para permitir compartir su conocimiento, proveer acceso al material y a las calificaciones, y compartir su conocimiento con otros estudiantes.[139]

Las más comunes son las siguientes:

- **Micromundos de programación o *Programming Microworlds*:** Su objetivo es usar un lenguaje simple con un conjunto limitado de instrucciones. Realiza la ejecución del código paso a paso para ir visualizando su resultado. Ejemplos de estos son: Jeroo [55], objectKarel [121], Code Studio [5]. Ideales para introducir principiantes en la programación. Aunque existe la duda si el conocimiento adquirido en este entorno se puede usar en un lenguaje de programación convencional más adelante.
- **Entornos educativos de programación:** Su objetivo son principiantes. Intuitivos y amigables con el usuario. Incorpora herramientas como editores y uno o más lenguajes de programación para apoyar los estudiantes con sus dificultades a través de interfaces de usuario simplificadas y mejorando el reporte de errores. Por ejemplo BlueJ [91].
- **Entornos profesionales de programación** Su objetivo son programadores avanzados. Ofrecen varias herramientas para programadores profesionales que pueden confundir a programadores principiantes.[121]
- **Herramientas para la visualización de algoritmos y programas**
Muestran la estructura de un programa o el resultado de su ejecución. Herramientas como Jelio [135] permite la visualización de código para apoyar a los estudiantes en la comprensión de POO.
- **Juegos Educativos**
Tienen como objetivo adquirir conocimiento y desarrollar habilidades de una manera implícita. Incluyendo características de gamificación (elementos de juegos como mejores puntajes, promover la competitividad entre los participantes, entre otros.) con impacto positivo en el apoyo a un aprendizaje efectivo.[77]

- **Learning Management Systems (LMS)**

Es una herramienta encargada de gestionar las actividades de formación de forma no presencial. Sus funcionalidades son: gestión de participantes, proveer material educativo, control y seguimiento del proceso de aprendizaje y gestión de la comunicación entre los participantes.[110]

- **Jueces en línea o Online Judge.** Sistema en el cual se pueden evaluar programas en competencias de programación. Dentro de su funcionalidad está compilar, ejecutar y probar código con datos pre-existentes. Tiene la opción de poner restricciones como límite de tiempo, memoria, entre otros. Ofrece realimentación a los ejercicios ingresados y los compara con los puntajes obtenidos de otros participantes destacando el número de soluciones correctas y el menor tiempo de ejecución.[68]

- **Herramientas automáticas de evaluación**

Automatiza la evaluación de tareas y provee a los estudiantes y docentes realimentación inmediata. Estas herramientas realizan completamente la evaluación de los ejercicios propuestos. Dando una evaluación inicial e indicando los problemas en el código de los estudiantes. Además, provee la capacidad de subir los ejercicios propuestos en línea, verifica el estilo del código y el diseño de aspectos del programa, así como el uso de características específicas del lenguaje.[126]

4.3 ARQUITECTURA ORIENTADA A SERVICIOS

Es un estilo de arquitectura fundamentado en el paradigma orientado a servicios. Pertenece a la nueva generación de computación distribuida. Esta arquitectura propone un nuevo modelo de diseño, construyendo servicios basados en procesos de negocio de manera individual. No usa una tecnología en particular y es basado en estándares libres. Esta compuesta de n-servicios con niveles de granularidad bien definidos[111].

La interfaz de un servicio web permite la definición de sí mismo en términos de parámetros de entrada y salida mas no en su implementación lo que permite abstraerse de la tecnología en la que estos se implementan.

Los servicios de SOA consisten en las funcionalidades de alguna aplicación que se comunica mediante una interfaz y es invocada por mensajes.

Elementos de SOA

- **Servicio:** Acceso a la información. Se crea con el objetivo de ser reutilizado. Son la unidad fundamental de esta arquitectura.
- **Contratos:** Interfaz entre el consumidor y el proveedor. Especifica los beneficios que brinda, cómo debe ser enviada una petición, cómo será la respuesta de la solicitud, entre otros detalles técnicos. Expresa la información que se va a intercambiar.
- **Mensajes:** Unidad de comunicación de los servicios entre las unidades de trabajo.

Beneficios del uso de SOA

- Incrementa la interoperabilidad mediante una comunicación sin transformaciones intermedias lo cual baja los costos y el impacto en el rendimiento.
- Facilita la integración entre las diferentes aplicaciones en la organización permitiendo la adaptación a los continuos cambios en los requerimientos.
- Los servicios son independientes. Esto garantiza la reusabilidad de un servicio para diferentes procesos de negocio y facilita su manejo.
- Permite la creación de sistemas de información altamente escalables.
- Los mensajes son estandarizados.

Ciclo de vida de un servicio en SOA

1. El proveedor registra el servicio en los registros de servicios.
2. Un consumidor encuentra los servicios de su interés en los registros.
3. El consumidor envía la petición al proveedor de esos servicios.
4. El proveedor procesa la solicitud y envía el resultado al consumidor.

En la figura 1 se muestra la función de sus elementos de un servicio SOA y las interacciones entre estos.

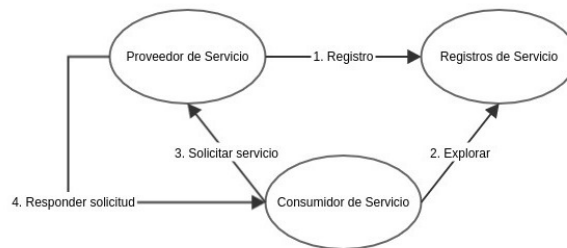


Figura 1: Ciclo de vida de un servicio SOA

1. Proveedor de servicio: Ofrece el servicio, lo hace disponible en internet.

2. Consumidor de servicio: Invoca el servicio.

3. Servicios de registro: Es un directorio de servicios, repositorio centralizado en donde se publica nuevos servicios o se encuentra los existentes.

Source: Autonomous Decentralized Service Oriented Architecture: Concept, Technologies and Application - Figure

1

Para realizar las interacciones entre los elementos de un servicio SOA se requiere:

- Un medio para el flujo de mensajes.
- Un protocolo de comunicación.

- La descripción del servicio.
- Servicios disponibles.
- Servicios de registro.

4.3.1 *Servicios Web*

Es una forma de computación distribuida que extiende la funcionalidad ofrecida por el protocolo HTTP de modo que se incorporen mecanismos que posibilitan la interacción proceso-proceso. Los recursos son ubicados en diferentes lugares de la red e interactúan remotamente usando llamadas a procedimientos remotos con el objetivo de crear, usar e intercambiar grandes cantidades de datos [69]. Es un elemento de software bien definido y autocontenido disponible en internet con una interfaz para ser llamado y proveer su funcionalidad usando un determinado protocolo[102].

Los servicios web pueden ser implementados en una amplia variedad de arquitecturas dependiendo del dominio de la aplicación y co-existir con otras tecnologías y enfoques de diseño de software [45]. Su característica más importante es la interoperabilidad e independencia de la plataforma en la cual son desarrollados. Esto se debe al uso de estándares libres.

Debido a que la interoperabilidad es la característica más importante dentro de la arquitectura SOA, se ha creado [Web Service Interoperability \(WS-I\)](#) con el propósito de promover el funcionamiento y la ejecución de los web services en diversas plataformas software y arquitecturas físicas.[109].

Existen también organizaciones como OASIS [108] y W3C [51], encargadas de la arquitectura y reglamentación de los servicios web.

4.3.1.1 *Estándares Empleados*

- [XML](#): Lenguaje de marcas para la descripción de contenidos de un documento. Permite estructurar y jerarquizar la información sin un límite de etiquetas. Creado con el objetivo de mejorar la compatibilidad entre las aplicaciones. Sus analizadores son estándares y no dependen de las versiones del lenguaje. Los datos en este formato son fácilmente leídos por ordenadores pero difícil para los humanos [116]. Estándar recomendado por la W3C.
- [JSON](#): Formato de codificación de datos de propósito general. Tiene similitud con XML. Consiste en estructuras ramificadas de tipo clave y valor. Los esquemas JSON son opcionales en un documento. Ampliamente usado debido a que la mayoría de lenguajes de programación tienen librerías para procesarlo [49].
- [SOAP](#): Basado en el protocolo creado por Dave Winer en 1998, llamado XML-RPC. Creado por Microsoft, IBM, entre otros y actualmente es patrocinado por la W3C. Provee un framework de mensajería entre pares caracterizado por ser descentralizado, distribuido y sin estado que se basa en XML. Los detalles de los parámetros para entrada y salida del servicio están disponibles en la descripción del servicio web.

SOAP es extensible, multiplataforma e independiente del lenguaje de programación y puede ser usado sobre cualquier protocolo de transporte como SMTP, HTTP, TCP o JMS. Ha sido empleado como bloque de construcción para realizar protocolos más complejos o a necesidad de las aplicaciones que lo implementen. [128].

- **WSDL:** Formato estándar XML que describe un servicio web. Provee una descripción de la interfaz, el método para invocar el servicio y los puntos finales o *endpoints* para el intercambio de datos sobre diversos protocolos [101] .
- **UDDI:**
Es un directorio de empresas y servicios. Para publicar un servicio es necesario realizar un registro con la descripción de este en formato xml. El acceso a estos registros pueden ser públicos o privados.[84].
- **HTTP:** Desarrollado por W3C y es usado desde 1990. Es un protocolo a nivel de aplicación para sistemas distribuidos, colaborativos y de información hipermedia. Basado en la arquitectura cliente-servidor, es un protocolo sin estado que permite el intercambio de datos a través de una conexión TCP/IP. Es Flexible, cualquier dato puede ser transferido y soporta la adición de nuevos métodos para ampliar sus funcionalidades. Diseñado para realizar el mejor esfuerzo de llegada de los paquetes a su destino, no garantiza la llegada de estos, y pueden ocurrir pérdida de paquetes. Si no se cuenta con unas condiciones mínimas de comunicación entre el cliente y el servidor, paquetes pueden ser descartados o no garantizarse el orden de llegada de estos. El protocolo HTTP define métodos que determinan el tipo de interacción entre el cliente y el servidor tales como: *head*, *get* y *post* [143].
- **WS-Security:** Es un estándar industrial que se encarga de los diversos aspectos relacionados con la seguridad en los servicios web. El objetivo primario es asegurar el intercambio de mensajes SOAP entre aplicaciones de extremo a extremo. Sus metas son la especificación de múltiples dominios confiables, seguridad en tokens, tecnologías de encriptación, formatos de firma (*signature formats*) y seguridad a nivel de mensajes. Las organizaciones que más han contribuido en el proceso de WS-S son IBM y Microsoft [88].

Los aspectos más importantes que trata son:

- **Autenticación:** Proceso de identificar a un individuo o una entidad que pretende ser.
- **Integridad:** Proceso de asegurarse que los datos no han sido alterados durante la transacción.
- **Confidencialidad:** Proceso de prevenir que usuarios no autorizados accedan a los datos.

Para implementar la arquitectura SOA sobre un sistema es necesario contar con herramientas para describir y ejecutar los servicios. A continuación, se realiza una descripción de estas herramientas.

4.3.2 Especificaciones para la orquestación de Web Services

A continuación se describen dos especificaciones relacionadas a la forma como se logra la ejecución coordinada de *web services*.

WS-CDL

Desarrollado por W3C en el 2003, en colaboración con compañías como Pi4Tech, Adobe, Oracle, entre otras. Una coreografía en SOA es una composición de servicios en la cual el protocolo de interacción entre estos servicios es definido desde una perspectiva global y describe el orden y las restricciones con que los datos son intercambiados. De esta manera, el lenguaje [WS-CDL](#) está enfocado en la descripción de este protocolo.[117]

[Web Services Choreography Description Language](#) es un lenguaje basado en XML que describe el comportamiento global de la colaboración y comportamiento de un conjunto de servicios para lograr un objetivo. Este lenguaje describe las funciones y los fallos, llamados comportamientos observables que puede presentar un servicio, además los servicios pueden ser validados con su descripción estáticamente y en tiempo de ejecución. Anteriormente se describían los servicios con [WSDL](#). Sin embargo, con [WS-CDL](#) es posible describir los comportamientos no observables, por ejemplo conocer las razones por la cual durante un proceso no se obtuvo el resultado esperado, es decir conocer el flujo interno de los servicios involucrados. De esta manera, [WS-CDL](#) establece restricciones para promover la interacción adecuada entre servicios, determinando un protocolo de negociación y garantizando la interoperabilidad.[90]

WS-BPEL

Fue creado en el 2007, conocido anteriormente como BPEL4WS. Es un lenguaje basado en XML estandarizado por OASIS [108], para la composición, implementación y manejo de servicios web. Nace de la necesidad de integrar diversas tecnologías que aún no lograban interoperabilidad, funcionando como un engranaje. Su objetivo es orquestar servicios web. Toma propiedades de [Web Services Flow Language](#), que modela gráficamente las actividades y conectores que describen las interacciones entre sus componentes y de [XML Language](#), que tiene una notación orientada al comportamiento de intercambio de mensajes a través de los servicios web.[140]

Se diferencia de [WS-CDL](#), en que usa orquestación como método para obtener el objetivo común en vez de coreografía. La orquestación consiste en que un web service lidera un grupo de web services para lograr una función general, por el contrario en la coreografía cada web service es encargado de lograr en conjunto el objetivo común.[18]

4.3.3 Arquitectura REST

El concepto de REST fue propuesto por Roy Fielding en el año 2000 [70] para sistemas distribuidos hipermedia. Definido como un conjunto de principios arquitectónicos usados para el diseño de aplicaciones enfocada en los recursos del sistema. Está basada en la arquitectura cliente-servidor y exhibe un estilo de capas con orden jerárquico donde cada capa inferior solo puede tener contacto con su capa superior, poniendo un límite a la complejidad del sistema global y promoviendo la independencia entre estas. Actualmente, el concepto de REST se usa para describir una interfaz que realiza operaciones sobre los recursos con métodos HTTP.

Principios de la arquitectura REST[138]:

- Todos los recursos deben estar identificados mediante un URI.
- Los recursos deben estar unidos entre sí. Ya que esta arquitectura fue diseñada para sistemas hipermedia, y su concepto clave es el hipermedia, es decir enlaces
- Manipulación de recursos con métodos estándar. Estas operaciones son obtener, crear, modificar y eliminar. En el protocolo HTTP las operaciones correspondientes son GET, POST, PUT y DELETE.
- Múltiple representación de los recursos. Tales como representación en HTML y XML
- Comunicación sin estado.

Los elementos arquitectónicos de REST son los siguientes:

- **Componentes:** Programas que interactúan uno con otro. Existen tres roles: cliente, servidor e intermediario. El cliente, configura la petición y recibe la respuesta. Servidor de origen, es la fuente de los datos. Los componentes intermediarios pueden adquirir cualquiera de los roles anteriores ya que es a través de ellos que se lleva a cabo la comunicación entre clientes y servidores.
- **Conectores:** Mecanismo abstracto que tiene como objetivo el comunicar y coordinar la interacción entre los componentes.
- **Data Elements:** Toda la información necesaria para que los conectores puedan procesar la petición debido a que todas las interacciones en REST son sin estado.

El aspecto clave de la arquitectura REST es el estado actual de los "data elements". Se han definido seis, mostrados a continuación [24]:

- Recurso
- Identificador de recurso
- Meta-datos del Recurso

- Representación del Recurso
- Representation de los meta-datos
- Datos de Control

Se definen los *"data-elements"* usados en la implementación de este proyecto.

- **Recurso:** Es una entidad que representa un concepto, por ejemplo un objeto del mundo real con sus propiedades y es direccionable a través de la web.
- **Identificador de recurso:** Nombre que lo diferencia de los otros recursos. En términos del protocolo HTTP, es llamado URI. Cada recurso puede tener múltiples URIs que proveen diferente información acerca de la ubicación del recurso.
- **Representación del Recurso:** La representación describe y muestra el estado actual de un recurso. Un recurso puede tener múltiples representaciones dependiendo del formato seleccionado por el programador, es decir, puede ser el recurso representado en formato JSON o XML.

En la implementación del proyecto se usa el concepto de curso o asignatura, este objeto corresponde al recurso. Un curso tiene un código, nombre, cantidad de créditos, docente que lo dicta entre otras propiedades, la representación del recurso corresponde al valor de estas propiedades en un formato XML o JSON. En la figura 2 podemos ver la representación del recurso y la URI con la cual es accedido.

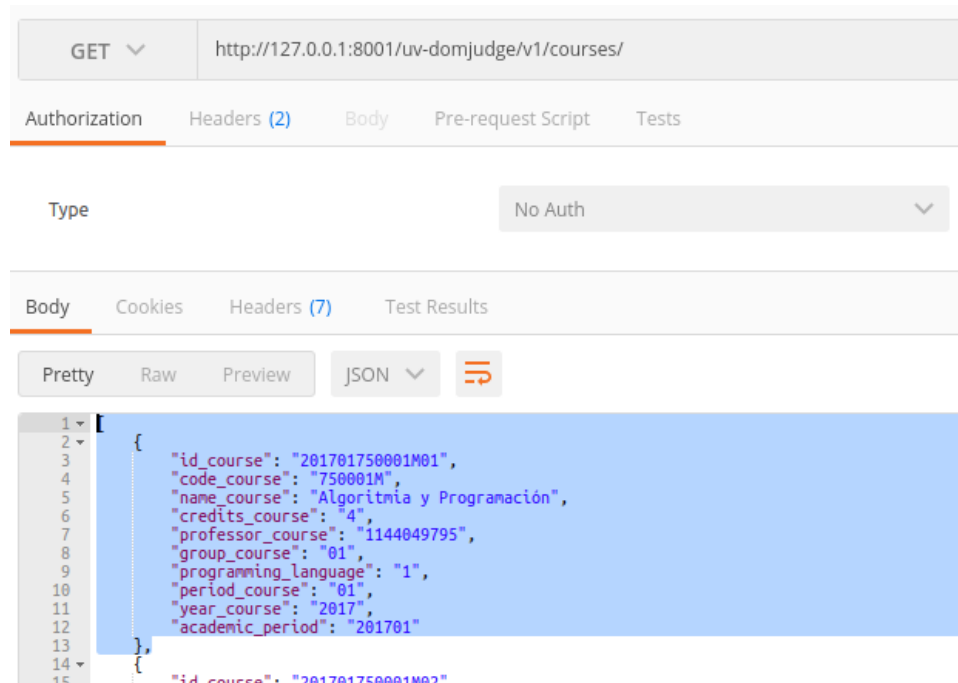


Figura 2: Representación y URI de un recurso

Un servicio que tiene una arquitectura REST es llamado servicio RESTful, la interfaz que tiene este servicio es llamada API REST. A diferencia de SOAP que maneja un conjunto de estándares y documentación (por lo general realizada con WSDL) , Las APIS REST carecen de documentación y estandarización, sin embargo, se dan un conjunto de pautas para intentar realizar las APIS REST de una manera uniforme. A continuación se citan algunas de las buenas practicas en la implementación de APIS REST:[100]

- **Modelado de Recursos:** Las REST APIS manejan diferentes tipos de recursos. Documentos para instancias únicas de recursos denominados mediante un sustantivo en singular, colecciones para un grupo de recursos denominados con un sustantivo en plural y controladores para las acciones que no pueden ser mapeadas a los métodos HTTP nombrados con verbos. Su denominación o nombre les permite ser accedidos mediante la URI indicando qué recurso es.
- **Identificación de Recursos:** En el caso de las REST API, las URI son frecuentemente las URL. Para mejorar la lectura de la URL se recomienda usar guiones medios en vez de guiones bajos, sin letras mayusculas. Usar la palabra API como parte del dominio e incluir la versión de la implementación de la API.
- **Representación de Recursos:** Los recursos deben soportar diferentes representaciones por ejemplo en formato XML o JSON, este formato debe ser acordado en tiempo ejecución.
- **Operaciones:** Para administrar los recursos, REST API deben manejar un conjunto uniforme de operaciones mapeadas a métodos HTTP. Para este proyecto las operaciones más relevantes son: Post, crea un nuevo recurso dentro de una colección, Get, retorna la representación de un recurso, Put, actualiza un recurso, Delete, elimina un recurso.

4.4 MENSAJERÍA

Permite la comunicación entre dos aplicaciones a través de canales limitando el conocimiento entre estas e independizando sus componentes. Considerada como el eje principal de las aplicaciones con una arquitectura en las que sus módulos y componentes solo se comunican entre componentes de la misma aplicación y con el exterior mediante interfaces o mensajes. Así, cada dispositivo conectado al bróker se comunica con los otros indirectamente. Frecuentemente usado para los sistemas los sistemas IoT, ya que garantiza la escalabilidad y confiabilidad de estos. [124]

Existen numerosos patrones de mensajería. A continuación se describen los más usados:

- **Patrón Publish/Suscribe:**

En este patrón las partes se comunican mediante etiquetas. Las etiquetas son categorías que agrupan los mensajes. Cuando un productor publica un mensaje lo categoriza para que sea enviado a los canales que manejan estas categorías, de esta manera los canales envían los mensajes a los suscriptores interesados. [123]

- **Patrón Request/Response:** Un solicitante genera una solicitud a la parte que se encarga de recibir, procesar y enviar la respuesta a esa solicitud. Permite que dos aplicaciones mantengan una comunicación bidireccional a través de un canal.[134]
- **Patrón RPC:** Un solicitante genera una solicitud a la parte que se encarga de recibir, procesar y enviar la respuesta a esa solicitud. Permite que dos aplicaciones mantengan una comunicación bidireccional a través de un canal.[13]

4.4.1 Protocolo AMQP

AMQP aparece en el año 2006 como una propuesta de la industria de los servicios financieros. AMQP es un estándar libre que sigue los lineamientos establecidos en el middleware **Message Oriented Middleware (MOM)** (software que soporta enviar y recibir mensajes en sistemas distribuidos) y fue diseñado para facilitar el diálogo entre los componentes de un sistema, simplificando el intercambio de mensaje independiente de las plataformas subyacentes. Su última versión es la 1.0, liberada en Octubre de 2011.

Su objetivo es habilitar la aplicación para enviar o recibir mensajes. Funciona como mensajería instantánea, pero se diferencia en que comprende un protocolo de red, que permite la interoperación de sus componentes (productor/consumidor, broker), y un protocolo modelo que especifica la representación de los mensajes. También, permite especificar qué mensajes se pueden recibir y cómo se realizan los compromisos con respecto a seguridad, fiabilidad y rendimiento. Los mensajes AMQP son mensajes auto-contenidos, sin límite de tamaño y el contenido de los datos es inmutable.[98]

4.4.1.1 Capas del protocolo AMQP: [142]

- **Capa de modelo:** Define un conjunto de comandos o funciones útiles para la aplicación, agrupados en clases lógicas o por funcionalidad.
- **Capa de sesión:** Proporciona un transporte confiable de comandos desde la aplicación al servidor y respaldos con repetición, sincronización y manejo de errores.
- **Capa de transporte:** Proporciona enmarcado "framming", multiplexación de canales, detección de fallos y representación de datos.

AMQP define una serie de entidades. Desde la perspectiva de la interconexión las más relevantes son: [142]

- **Bróker de Mensajería:** Servidor que establece la conexión con un cliente AMQP y es el encargado de gestionar los mensajes y las colas de mensajes a su cargo.
- **Usuario:** Entidad que mediante unas credenciales puede ser autorizado para conectarse a un bróker.
- **Conexión:** un enlace físico, podría ser TCP/IP o SCTP.

- **Canal:** Conexión lógica que posee un estado. Aquellos clientes que realicen operaciones concurrentes mediante una misma conexión deben mantener un canal distinto para cada una de ellas. Si la concurrencia se realiza usando hilos, se sugiere encapsular la declaración del canal en una variable local de cada hilo.
- **Cola de mensajes:** Es el lugar donde residen los mensajes. Tiene propiedades como privada o pública, duradera o transitoria, permanente o temporal.
- **Intercambio:** Es el proceso mediante el cual mensajes provenientes de un productor son enrutados de acuerdo a un criterio establecido llamado "binding" o enlace.
- **La clave de enrutamiento:** Es una dirección virtual que el intercambio usa para decidir donde enviar un mensaje.

En la figura 2 se da un ejemplo de broker mostrando la interacción entre sus elementos para lograr que un mensaje llegue al consumidor.

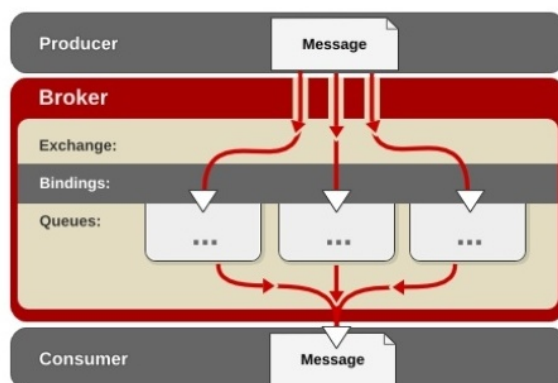


Figura 3: Broker AMQP

Source: "Developing highly scalable applications with Symfony and RabbitMQ" by Alexey Petrov

4.5 SUPERVISIÓN DE CONTENEDORES

Las herramientas para monitorizar contenedores nacieron cuando las herramientas tradicionales no eran validas para supervisar los contenedores. Las soluciones tradicionales fueron diseñadas para conocer el rendimiento de la aplicación, supervisar el uso recursos como cpu, memoria y almacenamiento, y conocer sus errores. Estas soluciones están directamente alojadas en el *host* o equipo anfitrión, o como parte de la funcionalidad de la aplicación en sí. Sin embargo, con el uso de contenedores ya no son efectivas, ya que estos se ubican en una zona entre el host y las aplicaciones, donde no hay alcance por parte de las soluciones tradicionales. [26]

Los objetivos de las herramientas para monitorizar contenedores son:[15]

- Supervisar todas las capas (infraestructura, capa adicionada por los contenedores y aplicaciones) evitando puntos ciegos. Comprobando la integridad, disponibilidad y rendimiento del host y máquinas virtuales con contenedores alojados.
- Ser flexibles y adaptarse a los cambios. Ya que el uso de contenedores ha forzado a las aplicaciones para trabajar como un sistema distribuido. Siendo muy común y conveniente para implementar microservicios, que son detenidos, actualizados y cambiados constantemente.
- Rastrear contenedores en ejecución para obtener información acerca su estado y etiquetarlos para que puedan ser supervisados como conjuntos de consulta en lugar de como individuos
- Mostrar estadísticas del uso de recursos en ejecución para conocer qué recursos deben ser ampliados cuando existe demanda de su uso, picos de tráfico dado un tiempo exacto, volúmenes huérfanos y nodos libres si están en una red.
- Informar acerca de los contenedores que han sido movidos a otro nodo.
- Tener logs centralizados

Ahora, existen al menos dos maneras de ubicar las herramientas para supervisar contenedores. La primera es la inyección de código para monitorear la aplicación dentro del contenedor, generando más complejidad ya que va en contra de las buenas practicas de los contenedores. La segunda consiste en alojar la herramienta en un contenedor. Esta opción trae las siguientes ventajas: [103]

- La herramienta de supervisión será independiente de los fallos del contenedor monitorizado y viceversa.
- Un solo monitor puede supervisar múltiples contenedores.
- Las funciones de supervisión consumirán menos recursos, ya que si fueran alojadas en cada contenedor para ser supervisado competirán por los recursos con la aplicación dentro de este y por ende el consumo de recursos se incrementa.
- Dan más flexibilidad, ya que pueden ser actualizadas o reconfiguradas sin afectar los contenedores monitorizados.
- Puede supervisar monitores que no están en el mismo host o infraestructura.

ESTADO DEL ARTE

En este capítulo se citan los desarrollos, herramientas, técnicas y tendencias actuales encontradas en la investigación que implementan los conceptos teóricos discutidos en el capítulo anterior.

5.1 VIRTUALIZACIÓN

Una de las discusiones constantes acerca de las técnicas de virtualización es si los contenedores deben remplazar a las máquinas virtuales. Actualmente, se están realizando esfuerzos para que los hipervisores y los contenedores trabajen juntos, ya que las máquinas virtuales pueden servir de base para la ejecución de contenedores. Como ejemplo de este esfuerzo la empresa Canonical [4], promueve ampliamente LXD [17], un hipervisor construido con linux containers que permite alojar Docker, logrando el mismo efecto que una máquina virtual.

Por otro lado, debido al creciente uso de contenedores, se ha hecho necesario el uso de herramientas que permitan su orquestación a gran escala. Una de estas herramientas, es Kubernetes, diseñada por Google en el 2014, la cual es una herramienta de código abierto para la automatización del despliegue, orquestación y gestión de grupos de contenedores diseñado para ser un entorno para la creación, funcionamiento y administración de sistemas distribuidos.[16]

Otra tendencia es el uso de máquinas virtuales del tipo [Virtual Desktop Infrastructure](#). Las empresas que ofrecen este servicio intentan proporcionar varias herramientas y servicios como hipervisores para alojar escritorios virtuales, mensajería, conexión a escritorios remotos, seguridad en el *host*, entre otros servicios. Actualmente, el objetivo de los VDI es acomodarse a las necesidades de los usuarios ofreciendo y gestionando servicios, permitiendo a su vez el uso de herramientas de otros proveedores.

En resumen, la virtualización representa ventajas como la optimización de recursos físicos y la independencia en las funciones o aplicaciones que son puestas en ejecución dentro de los ambientes virtualizados. Estas ventajas son aprovechadas tanto a nivel comercial como en la academia. A continuación, se citan algunos desarrollos actuales en relación a virtualización:

- **Herramienta para administración de imágenes Docker con Alta disponibilidad y reducción de tiempo de aprovisionamiento**[105]

Comicon, es un sistema que administra y comparte imágenes Docker en un conjunto de nodos. Ofrece alta disponibilidad de estas ya que son alojadas en un sistema de almacenamiento distribuido para el conjunto sin aumentar el costo de almacenamiento y reduce el tiempo de aprovisionamiento en un 28 %, ya que se mantiene una imagen en un host separado dentro de una red privada.

- **Protocolo para realizar puntos de restauración a máquinas virtuales alojadas en sistemas distribuidos**[76]

UnaCloud snapshot, es un protocolo que se encarga de realizar puntos de restauración de máquinas virtuales en sistemas distribuidos con tolerancia a fallos. Una de las estrategias que ofrecen las máquinas virtuales para recuperarse de un fallo son los puntos de restauración o "snapshot". Sin embargo, obtener un punto de restauración para un conjunto de máquinas virtuales en un sistema distribuido no es trivial, ya que no existe un reloj global para que esta sea realizada. Por lo tanto, se propone un protocolo para que estos puntos de restauración sean realizados. La prueba de este protocolo fue realizada en la Universidad de los Andes (Bogotá - Colombia), mostrando una baja sobrecarga en las máquinas físicas y una alta consistencia de los datos.

- **Algoritmo de reubicación de máquinas virtuales optimizando el uso de energía en los servidores.**[41]

En este trabajo se diseña un algoritmo de reubicación en los servidores para las máquinas virtuales que minimiza el consumo de energía sin afectar la calidad en el servicio a los usuarios. La computación en la nube es usada en casi cualquier campo y los centros de datos se vuelven cada vez más grandes, con esto el costo operacional también se incrementa por lo tanto, el uso eficiente de energía se ha convertido en uno de los principales temas de investigación.

- **Servicio de monitoreo para microservicios alojados en contenedores**[57]

En este estudio se propone un servicio de vigilancia para una red de microservicios ("IoT as a service") que permite monitorear, reparar o reemplazar microservicios. Los microservicios son alojados dentro de un contenedor y estos contenedores se estarán ejecutando en los dispositivos IoT. El servicio consiste en que por cada dispositivo se ejecutan dos contenedores, de estos uno es alternativo en caso de falla. El servicio de vigilancia está alojado en el kernel de Ubuntu (versión de Ubuntu para dispositivos con recursos limitados) y es sincronizado con los contenedores de la red de esta manera, cuando un contenedor falla el problema es detectado por un contenedor vecino. Cuando un problema ocurre el servicio intenta repararlo. De no ser posible, se usa el contenedor alternativo para proveer el microservicio.

- **Comparación en el rendimiento de herramientas de aprendizaje profundo en un contenedor vs. equipo anfitrión**[141]

Se realizó un estudio sobre el rendimiento de herramientas de aprendizaje profundo en un contenedor Docker e instaladas en un equipo anfitrión. Las herramientas para aprendizaje profundo tienen un alto consumo de procesamiento, gráficos, acceso a discos, memoria y dispositivos I/O, también en ocasiones es necesario instalar varias

versiones de las herramientas o herramientas con librerías incompatibles. Debido a esto, al realizar la ejecución de estas herramientas dentro de un contenedor Docker estos conflictos estarían resueltos. Los resultados de las pruebas mostraron que el rendimiento de las herramientas era igual de bueno en un contenedor como en un equipo anfitrión.

- **Plataforma basada en el concepto de cadena de bloques para compartir contenedores** [56]

El objetivo de esta investigación es crear una plataforma en la cual se pueden compartir aplicaciones dentro de contenedores. Para realizar esta tarea se utiliza el concepto de cadena de bloques o "blockchain", permitiendo almacenar todas las versiones y los cambios en el tiempo realizados sobre este contenedor. La plataforma genera una clave hash de la transacción (una transacción son las acciones realizadas en la plataforma como compartir un contenedor con otro individuo o un grupo de ellos) y esta es almacenada junto con el contenedor modificado en una base NoSQL.

- **Creación dinámica de máquinas virtuales para programación de recursos**[97]

La creación de máquinas virtuales es uno de los procesos para la programación de recursos ("resource scheduling") que es clave en computación en la nube. Este proceso es estudiado generalmente en conjunto con un modelo estático en el cual el número y tipo de máquinas virtuales es predefinido antes de la programación de los recursos. Existen muchos modelos de programación de recursos en los cuales se optimiza la calidad del servicio (QoS), costos, energía, balance de carga, ancho de banda, entre otros; que funcionan con un modelo estático y muy pocos estudios relacionados con la creación de máquinas virtuales, aunque este proceso también hace parte de la programación de recursos. Se propone entonces un modelo dinámico para la creación de máquinas virtuales con un requerimiento impreciso de recursos computacionales donde el tipo y el número de máquinas virtuales es calculado de acuerdo al entorno y son creadas dinámicamente. Esta aproximación dinámica de la asignación de recursos muestra mejoras en la calidad de servicio y optimiza de manera general los recursos del sistema.

Finalmente, se citan las herramientas que permiten la virtualización con cualquiera de sus técnicas o están relacionadas con el concepto de alguna manera.

- **VMWare Workstation**

Permite la creación de máquinas virtuales que ejecutan su propio sistema operativo. Con esta herramienta se logra compartir adaptadores de red, intercambiar y simular unidades de almacenamiento, guardar el estado de una máquina virtual en cualquier momento *snapshot* con el objetivo de reanudar posteriormente o evitar daños a la máquina virtual y agrupar varias máquinas virtuales en un "inventario" para usarlas como si fueran un solo objeto. Es desarrollada y vendida por VMware, con una versión para uso no comercial llamado *VMware Workstation Player*. [64]

■ Virtual Box

Es un software de virtualización para arquitecturas x86/amd64. Posibilita la instalación de sistemas operativos invitados dentro del sistema operativo anfitrión. Permite montar imágenes ISO como unidades de almacenamiento. Provee controladores que proporciona la aceleración en 3D, pantalla completa, 8 placas PCI Ethernet, ejecución de máquinas virtuales de forma remota en la versión de pago e integración con teclado y ratón. Los discos duros de los sistemas invitados son almacenados en el sistema anfitrión como archivos, estos archivos son incompatibles con otros software para virtualización. Fue desarrollada por innotek GmbH, actualmente es desarrollada y vendida por Oracle Corporation.[60]

■ Linux VServer

Es una herramienta que particiona un servidor físico en varios servidores virtuales. Cada servidor virtual funciona con su propio sistema operativo y puede ser reiniciado de forma independiente. Ofrece un rendimiento destacado, comparable al de los sistemas operativos instalados directamente en la máquina. Su principal desventaja es que carece de una interfaz gráfica lo que hace que no sea orientado al usuario final.[114]

Implementa su propio mecanismo en el kernel de Linux para aislamiento de procesos, red y CPU. El sistema limita el alcance de los archivos de diferentes procesos a través de chroot (llamada al sistema) y prohíbe comunicaciones indeseadas entre ellos usando la técnica " *global space*". El principal beneficio de esa técnica es la escalabilidad para un gran número de contenedores. Sin embargo, el inconveniente es que el sistema no está disponible para implementar las formas de virtualización habituales, tales como migración en directo, punto de control (*checkpoint*) y reanudar (*resume*), debido a que es imposible instanciar procesos con el mismo .

Linux VServer no virtualiza subsistemas de red, todos los subsistemas de red son compartidos entre los contenedores y el sistema anfitrión.

■ Open Virtuozzo (OpenVZ)

Virtualización basada en contenedores. Trabaja a nivel del sistema operativo con modificaciones a kernel de Linux.[115]

Cada contenedor tiene su propio subconjunto de recursos. El sistema usa IPC y (nombre de espacios) para lograr el aislamiento entre procesos de diferentes contextos. Open VZ también implementa espacios de nombres para la red (*Networking namespaces*), ofreciendo diferentes modos de operación de red.[130]

■ LXC

Está construido sobre *namespaces* del kernel de Linux. Ofrece el despliegue de uno a varios contenedores aislados.[131] Cada contenedor prohíbe que sus procesos en ejecución puedan acceder a recursos fuera de su entorno. Los recursos disponibles son gestionados usando *cgroups*, que agrupa procesos individuales y controla el consumo de recurso agregados.[92]. Estos contenedores comparten el kernel del sistema anfitrión.[119]

- **Xen**

Es un hipervisor código libre. Se sitúa entre el sistema operativo y el hardware.[129] Provee un entorno virtual en el que corre el kernel. Ofrece una forma eficiente de dividir los recursos del hardware de una máquina física en múltiples entornos de ejecución llamados máquinas virtuales. Los tres componentes básicos que involucran Xen son: hipervisor, kernel y espacio de aplicaciones de usuario (*userspace applications*).[137]

- **Qemu**

Es un emulador de máquina y un intérprete binario dinámico multiplataforma.[48] Traduce dinámicamente las instrucciones en código binario para la plataforma de destino en la cual es ejecutado el código generado. Esta operación, más la gestión de caché disminuye su rapidez. Admite la emulación completa del sistema y del modo de usuario. Solo permite instanciar un proceso a la vez.[96]

- **Docker**

Plataforma de código abierto para rápido desarrollo, envío y ejecución de aplicaciones de software. Mediante el uso de una plataforma de virtualización de contenedores ligeros que cuentan con herramientas y flujos de trabajo para ayudar al desarrollador a implementar y administrar sus aplicaciones y de manera aislada. Ofrece un repositorio en el cual se encuentran imágenes de casi cualquier sistema operativo e infraestructura.[81]

5.2 SOFTWARE PARA EVALUACIÓN AUTOMÁTICA

Las formas de aprendizaje están en constante cambio, incorporando nuevas técnicas para facilitar la adquisición de conocimiento y dando al aprendiz varias opciones para su educación. Una de estas formas es el aprendizaje a través de medios electrónicos. En este contexto, una tendencia que ha tomado fuerza es el m-learning o aprendizaje móvil, que cuenta con contenidos e-learning diseñados para dispositivos móviles conocidos como "responsive". El contenido que usa el m-learning es llamado micro-learning, caracterizado por usar pocos recursos físicos en los dispositivos, consta de vídeos y lecciones específicas cortas con el propósito de que la persona lo planifique con horarios flexibles en su agenda diaria.[9]

Respecto al contenido, se ha incorporado el uso de Big Data mediante la recolección de datos por medio de los LMS como realimentación de la calidad y utilidad de sus contenidos con el objetivo de optimizarlos para ajustarse a las necesidades de los usuarios.[1]

Con relación a las técnicas para facilitar el aprendizaje, se ha popularizado el uso de **gamificación**. Esta herramienta se integra para facilitar la interacción del aprendiz con los métodos de aprendizaje promoviendo las habilidades para solucionar problemas y conflictos. De esta manera, un estudiante puede ser capaz de mejorar mediante experiencias prácticas más que memorizando las lecciones.[61] Como ejemplo tenemos el uso de un videojuego llamado *Civilización IV*, enfocado en la construcción de un imperio y economía a escala para apoyar la comprensión de la historia y el desarrollo económico de la humanidad.[80]

En la academia, para la enseñanza de un nuevo lenguaje de programación se usan herramientas para el apoyo en el aprendizaje a estudiantes. Una de estas herramientas son los jueces virtuales, estas herramientas fueron desarrolladas para los concursos de programación como ACM-ICPC[83], con el objetivo de posicionar los equipos respecto a los aciertos en los problemas propuestos y evaluar las soluciones automáticamente. Una solución es evaluada como satisfactoria si la ejecución fue correcta y retorna la salida esperada. El uso de un juez representa un apoyo en el aprendizaje ya que el estudiante recibe una realimentación inmediata de las soluciones que propone. Ejemplos de jueces virtuales son: DOMjudge[86], PC²[19], BOCA[2], y UVA[29].

También se está buscando la incorporación de herramientas que, aunque no son diseñadas para apoyo al aprendizaje proveen características importantes para un estudiante que está iniciando en la programación. Un ejemplo de este es Travis CI [28], una herramienta para integración continua que trabaja con repositorios Github, da realimentación inmediata e indica exactamente dónde están los errores de compilación. Esto es una ventaja en relación a los jueces tradicionales que no hacen diferencia de los errores sintácticos de los errores algorítmicos de las soluciones planteadas.

A continuación, se citan herramientas para apoyar el aprendizaje de los cursos introductorios a la programación:

- **Self-Assessment Web Tool for Java Programming[47]**
Es una herramienta en línea que tiene como objetivo apoyar a los estudiantes de ciencias de la computación a desarrollar sus habilidades para auto-evaluar su aprendizaje en los cursos de programación usando el lenguaje Java. Esta herramienta consiste en un plugin para Moodle que permite a los estudiantes subir archivos de código fuente. La herramienta analiza el código y presenta un informe que incluye errores de consola y una comparación de su solución con los compañeros de curso. De esta forma el estudiante obtiene realimentación antes de la fecha límite de entrega. El docente también puede acceder a esta información, facilitando una visión general del progreso de la clase y también de forma individual.
- **An Interactive Tool to Support Student Assessment in Programming Assignments[132]**
Herramienta interactiva para el análisis de un conjunto de archivos de código fuente. El objetivo es presentar un informe que muestra las diferentes soluciones realizadas por los estudiantes e identifica las soluciones similares a través de relaciones en un grafo.
- **Tests Generation Oriented Web-Based Automatic Assessment of Programming Assignments[133]**
Sistema Web de apoyo a docentes y estudiantes en la evaluación automática de ejercicios de programación. Usa el framework Laravel, sobre un servidor LAMP (Linux, Apache, MySQL and PHP) y diseñado con Twitter Bootstrap. Se basa en funciones que ejecutan el código con diferentes entradas.
- **A Peer Grading Tool for MOOC's on Programming[146]**
Herramienta de calificación por pares para cursos de programación en línea. Ideal para una población grande de estudiantes. Proporciona una evaluación objetiva específica.

Usa balanceadores de carga para equilibrar el trabajo de los revisores. Divide el código en pequeños trozos para facilitar su revisión, además de dar flexibilidad para escalar este proceso.

- **A Framework of an Automatic Assessment System for Learning Programming**[107]
Es un framework para evaluación automática. Su objetivo es apoyar el aprendizaje de [Programación orientada a objetos](#). Sus módulos fueron diseñados para satisfacer las necesidades de la comunidad académica. Tienen la funcionalidad de: mostrar errores y colocarlos en un reporte, realimentación de forma instantánea y un módulo dedicado a las interacciones sociales entre los usuarios.
- **Ask-Elle: an Adaptable Programming Tutor for Haskell Giving Automated Feedback**[71]
Ask-Elle es una herramienta para el apoyo al aprendizaje del lenguaje de programación Haskell. La herramienta provee una interfaz para añadir la descripción de un ejercicio, uno o varios modelos de solución y las propiedades de solución que se deben satisfacer. Soporta el desarrollo paso a paso, verificando la correctitud de los programas y proporcionando sugerencias. La principal funcionalidad se obtiene mediante el seguimiento de modelos basado en estrategias y las pruebas basadas en propiedades.
- **Framework for Evaluation of Programming Language Examinations**[99]
Framework para la evaluación de exámenes prácticos de informática. Implementado en el lenguaje de programación Java, con herramientas y librerías de código abierto. Sus funcionalidades son: Interpretación de la salida del compilador y reporte de errores, realización de pruebas a los programas subidos, cálculo de métricas de software como líneas de código y líneas de comentario, complejidad ciclomática y análisis de similitud semántica del programa del estudiante comparado con el modelo del programa.
- **Adaptive Testing in Programming Courses Based on Educational Data Mining Techniques** [85]
Generación automática de exámenes y pruebas adaptado a cursos de programación basado en la minería de datos educativos. Los puntos de la prueba se evalúan usando técnicas predictivas de regresión y clasificación. Se realiza un mapa conceptual de las competencias del curso y un método de estimación de la dificultad de la prueba. Finalmente, un algoritmo genético selecciona la prueba más adecuada a los criterios del curso.
- **Inginious**[62]
Herramienta para la evaluación y realimentación automática. Todas sus funcionalidades son realizadas como procesos independientes y ejecutados en contenedores aislados. Estos contenedores tienen la opción de crear subcontenedores llamados *Contenedores de estudiantes* para ejecutar código no confiable, asegurando la integridad del servidor de INGIous. Permite la integración con (Open) edX y Moodle. Desarrollado por INGI department.

5.3 ARQUITECTURA ORIENTADA A SERVICIOS

Hoy en día, la arquitectura SOA ha sido ampliamente aceptada debido a que los programadores se centran más en los servicios que en el desarrollo y uso de componentes. Un componente es una unidad de software que es independientemente reemplazable y actualizable que mantiene una conexión con la aplicación o plataforma que lo usa mientras que para usar un servicio solo es necesario saber cómo llamarlo (usando un protocolo de comunicación) y conocer su interfaz. [53]

Existen dos perspectivas para que al implementar arquitectura SOA en un sistema, este pueda evolucionar. La primera es llamada "Diseño para la evolución", consiste en encontrar la modularidad adecuada en el diseño del software con el objetivo de poder actualizar los módulos individualmente. La otra perspectiva es llamada "soporte para la evolución.^{en} donde ya existe un software en uso y se buscan herramientas para que se mantenga actualizado.

Con el objetivo de ampliar la perspectiva "diseño para la evolución", una tendencia reciente en software es el acortamiento de los ciclos de entrega. Es común liberar un software como una aplicación web combinado con un cliente o "app" en su teléfono móvil de esta manera la actualización a la nueva versión del software se realiza de forma automática y se pueden liberar las actualizaciones incluso cada día. Microservicios es llamado a un estilo de arquitectura que maneja estos periodos cortos de entrega y es usada para construir sistemas distribuidos.

Un microservicio es una pequeña pieza comprensible para el programador que es implementado y puesto en ejecución de manera independiente. Usar microservicio trae la ventaja de que puede ser modificado y versionado sin afectar a los otros componentes. [27] Los servicios en una arquitectura de microservicios son procesos que se comunican con otros a través de una red para conseguir el objetivo final. Estos servicios pueden usar protocolos simples (típicamente HTTP con REST o mensajería liviana como RabbitMQ[21] o ZeroMQ[33]). Su objetivo es construir una aplicación como un conjunto de pequeños servicios. En este estilo de arquitectura cada equipo de trabajo se encarga de servicios independientes y las actualizaciones son liberadas a su ritmo y sin coordinar versiones.[136]

A continuación, se mencionan algunos desarrollos relacionados con la mejora e innovación en cuanto al uso de arquitectura de microservicios:

- **Optimización para la reducción de latencia en servicios dependientes.**[145]

Una gran cantidad de aplicaciones interactivas son ahora programadas como una cadena de servicios en la nube buscando mejor escalabilidad y bajos costos de operación. La mayoría de estas aplicaciones consiste en múltiples servicios independientes que se comunican entre ellos. Este paso a paso en la comunicación incluye inevitablemente un retraso que generalmente se debe al manejo de las colas ("*queue*") en los servidores en la nube. Uno de los objetivos es optimizar el retraso de cola general del microservicio. En la prueba se obtuvo un 35 % menos de retraso con un impacto mínimo en los balanceadores de trabajo.

- **Promover el desarrollo de plataformas educativas con arquitectura de microservicios[50]**

Es una necesidad para las entidades educativas proveer un espacio en internet donde los docentes y estudiantes puedan realizar actividades académicas sin embargo, aún proveen desarrollos para aplicaciones académicas con arquitecturas monolíticas desplegadas en la nube, desaprovechando las ventajas que da la arquitectura de microservicios como flexibilidad en la actualización, independencia en los componentes, protección en los accesos no autorizados a datos académicos, optimización de procesos y adición de nuevos componentes.

- **Método para realizar la migración de una arquitectura Monolítica a una de Microservicios[67]**

Easylearn, una aplicación realizada con una arquitectura monolítica fue migrada a microservicios con el objetivo de incorporar los beneficios que esta arquitectura ofrece con técnicas DevOps: integración y despliegue continuo, escalabilidad y facilidad de mantenimiento. La migración fue llevada a cabo de forma gradual con el objetivo de mantener la estabilidad del sistema. Los pasos realizados son: identificar de procesos de negocio para posteriormente encapsularlos en un servicio, extracción del código disponible que ejecutan estos procesos, establecer la comunicación entre estos servicios, finalmente las interfaces actuales son transformadas a interfaces de servicios.

- **Ambientes de Microservicio[78]**

El objetivo en esta propuesta es identificar los límites óptimos del microservicio es decir, lo que es necesario para que la funcionalidad que es encapsulada sea independiente y realice la tarea esperada. La propuesta consiste en identificar los microservicios en la etapa de ejecución, ya que se considera que la etapa de diseño donde los microservicios son identificados generalmente es una fase muy temprana ya que en esta no se logra identificar exactamente el nivel del granularidad y el comportamiento esperado del sistema.

- **Construcción de sistemas IoT con arquitectura de microservicios[95]**

Un sistema IoT es una red de servicios y la infraestructura conectada es sobre lo que se implementan los servicios. Un nodo en un sistema IoT es un objeto inteligente que presta un servicio. De esta manera, estos sistemas pueden ser estructurados con microservicios junto con otros patrones para gateways API, distribución de servicios, descubrimiento de servicios, contención y control de acceso; con el objetivo de tener una visión completa de cómo construir un sistema de IoT en general.

En la actualidad, la combinación de REST y microservicios ofrece la integración de componentes y la posibilidad de una mayor escalabilidad y tolerancia a fallos en aplicaciones en la nube y virtualizadas ya que al implementar una arquitectura REST entre los microservicios se tiene la ventaja de proveer una interfaz uniforme y acceder a servicios con **bajo acoplamiento**, sin estado e independientes de los clientes. En resumen, microservicios es un estilo arquitectural y de diseño mientras que REST se encarga de facilitar la exposición de la funcionalidad de los microservicios.[32]

El estilo arquitectural REST es comúnmente aplicado al diseño de APIs ([Application Programming Interface](#)) para los servicios web, una web API realizada con una arquitectura REST es llamada REST API y cuando un servicio tiene una una API REST este es llamado servicio "RESTful".

Un API es un conjunto de datos y funciones que ofrece una entidad para ser utilizado o llamado por otra entidad con el objetivo de intercambiar información. En un servicio web, una web API es la encargada de escuchar y responder a las peticiones de un cliente.

La cantidad de datos que son intercambiados a través de web APIs demuestra la acogida que han tenido estas tecnologías para interactuar con recursos externos. Sin embargo, en algunos casos al implementar estas APIs no se realiza de forma correcta, es decir siguiendo los principios "Restfull" lo que evita tener un ecosistema de APIs estables y estandarizadas.

Para evaluar el rendimiento de las APIs REST se realizó un estudio[54] en el cual se analizó un conjunto de datos (78GB de tráfico HTTP suministrado por Telecom, Italia). El primer objetivo era conocer el nivel de madurez de las APIS REST implementadas actualmente. Para esto se establecieron 3 niveles:

- **Nivel 0:** Se refiere a la comunicación entre los componentes. En este nivel se evalúa si utiliza RPC o SOAP.
- **Nivel 1:** Se evalúa el uso de múltiples recursos y sus respectivas URL, se verifica si esta url es comprensible para los humanos.
- **Nivel 2:** Se verifica el uso de los métodos HTTP, si han sido mapeados a las funciones correspondientes.
- **Nivel 3:** Se examina si la API ha adoptado el uso de hipermedia. Se verifica que los recursos no solo sean ubicados a través de su URI, sino también poseen relaciones con otros recursos.

Para medir el nivel de las APIs se realizaron una serie de heurísticas. Se obtuvo como resultado que de las APIs analizadas mas de la mitad cumplían con los niveles 1 y 2, sin embargo para los niveles 0 y 3, el porcentaje era mínimo. Lo que significa que las APIS del conjunto de datos están mas enfocadas a proveer operaciones [CRUD](#) a los recursos. Respecto al uso de las buenas practicas, solo algunas heurísticas dieron resultados favorables, lo que indica en general que las APIS aún están lejos de ser una disciplina estable y estandarizada.

Aunque el uso de APIS REST es frecuente, hay que considerar las siguientes desventajas:

- El concepto de CRUD usado en arquitecturas REST está basado en métodos y códigos de respuestas HTTP que dan a la arquitectura limitación e inflexibilidad.
- Cuando es necesario realizar operaciones sobre múltiples recursos a través de sus endpoints se genera sobrecarga en la aplicación, ya que esta arquitectura no posee una forma de realizar operaciones de edición sobre varios recursos.
- Toda la información del recurso es obtenida al realizarse una petición, incluso cuando solo es necesario una o varias de las propiedades del objeto.

- El uso de verbos y códigos de error de HTTP no es estandarizado y al implementar las APIS, los programadores los utilizan a su criterio.

Una tecnología que intenta solventar estas desventajas es GraphQL[10], que ha sido desarrollada por Facebook y se viene optimizando desde el 2012. GraphQL es un lenguaje de consultas del lado del servidor para APIs mediante el uso de un sistema de tipos y funciones que son definidos para cada propiedad de un recurso. [20]

Cuando se realiza una petición a una API REST se usa un método HTTP y este es mapeado a una sola función, en contraste con GraphQL que permite en una sola petición el que se puedan invocar varias funciones. Así mismo, la sintaxis usada en la petición es la misma usada en la respuesta. GraphQL dispone de un sistema de tipado fuerte por el que cualquier mal uso puede ser rápidamente detectado en tiempo de desarrollo [12].

GraphQL al igual que una API REST, trabaja con recursos que son accedidos mediante las URIS, los datos de las peticiones y respuestas son enviados en formato JSON. En la estructura de la petición se especifican las propiedades del recurso deseadas y la respuesta tendrá su misma estructura. En la figura (lado izquierdo) 4 se realiza una petición al objeto "hero" consultando la propiedad de este llamada "friends" que a su vez tiene la propiedad llamada "name" y la respuesta con la misma estructura de la petición (lado derecho) [11].

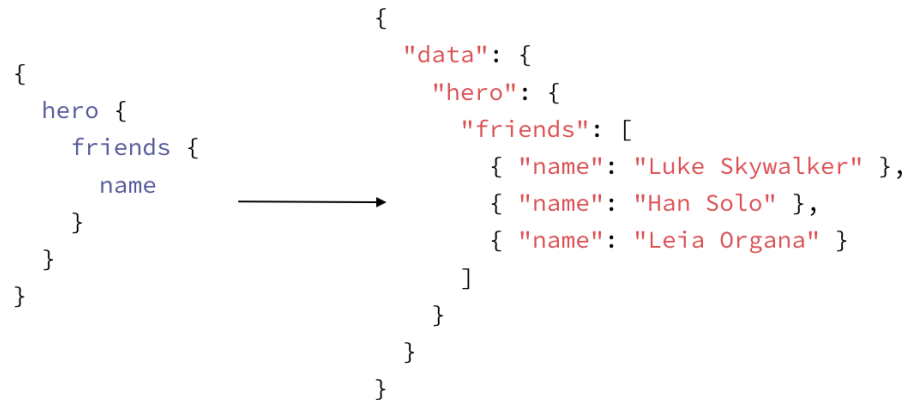


Figura 4: Ejemplo de Petición/Respuesta con GraphQL

Source: Bringing GraphQL to iOS[14]

En seguida, se citan algunos desarrollos relacionados con APIS REST:

- **Integración de APIS REST Y web semántica[122]**

Se realizó un middleware (software que conecta dos aplicaciones independientes pasando datos entre ellas) con el objetivo de incorporar datos enlazados en el intercambio de contenido entre APIs REST. Los datos enlazados son un termino relacionado con web semántica, son representados mediante grafos con el objetivo de que la máquina los entienda y establezca relaciones entre sus nodos.

- **Herramientas de documentación automática de APIS REST[125]**

La documentación para una API debe ser completa y correcta, incluir ejemplo de códigos y explicaciones. Con este objetivo se desarrolla una herramienta llamada SpyREST,

que genera automáticamente la documentación de la API mediante llamadas directas a la API.

- **Herramienta para el análisis de la estructura de un API**[79]

Muchas de las herramientas que intentan aplicar una arquitectura REST no cumplen con todos los requisitos que esta arquitectura implica, por lo tanto no da todos los beneficios que el diseño provee. En este estudio se desarrolla una herramienta para el análisis de la estructura de una REST API basada en su documentación.

- **Uso del algoritmo de camino aleatorio para reducir el numero de uris a los recursos**[93]

Una restricción que tiene la arquitectura REST es que cada recurso debe estar ligado a una URI, cuando esta restricción no se aplica de forma prudente es posible que haya una gran cantidad de URIs que no proveen servicios pero que incrementan la dependencia entre recursos. Para solventar esto se propone un algoritmo de caminos aleatorios para reducir el conjunto de URIs.

5.4 MENSAJERÍA

El uso de brokers de mensajería es indispensable para las arquitecturas con las cuales se están diseñando las aplicaciones actualmente por ejemplo microservicios, servicios o módulos que son desarrollados independientemente y en el desarrollo de plataformas IoTs, conectando dispositivos IoT de todo tipo.

Un patrón que es frecuentemente usado es "publica/suscribe", aunque es posible usarlo sin broker de mensajería, diversos brokers lo implementan como Web Sphere[31], VerneMQ[30], RabbitMQ[21] y ZeroMQ[33]. Este patrón es ampliamente usado para desarrollar sistemas distribuidos de gran escala y varias aplicaciones como cotizaciones de bolsa en línea, juegos, y notificaciones de noticias. En la academia se están realizando investigaciones [127], [75] para implementar autenticación, encriptación y confiabilidad en el envío de los mensajes. Un patrón que sí requiere el uso de un intermediario es el de colas de mensajes, ya que estas estructuras son gestionadas por varios receptores y es necesario que su administración esté centralizada.[112]

A continuación, se citan desarrollos los cuales implementan o mejoran este concepto:

- **HIBE, sistema de encriptación basado en IBE para el patrón publica/suscribe**[127]

Implementar seguridad en un sistema de mensajería como "publish/suscribe" es una tarea desafiante debido a que estos sistemas son de bajo acoplamiento. Un tipo de seguridad implementado ha sido **Identity based Encryption** (basado en pares remitente-destinatario. El remitente cifra un mensaje usando un parámetro público del destinatario como su nombre de usuario o la dirección de correo, el destinatario obtiene la llave para descifrar de una entidad centralizada y confiable que se encarga de generar las llaves para cada usuario). En este estudio se propone el sistema **Hierarchical identity based Encryption**, una generalización de IBE que realiza el cifrado de un mensaje para un grupo de identidades y la decodificación es realizada por parte del cliente de esta

manera, el tiempo de cifrado es mucho menor ya que IBE cifraba el mensaje para cada usuario e HIBE lo realiza para un grupo de ellos.

- **Framework para evaluar brokers de mensajería a gran escala**[89]

Realizar pruebas a sistemas de mensajería a gran escala es una tarea costosa debido a que la entidad encargada de realizar las pruebas debe diseñar y ejecutar los test sobre múltiples conexiones y validar el tráfico entre los clientes y el [broker de mensajería](#). En este estudio se propone un framework que consiste de dos partes: el test manager y el escenario compuesto por el broker y sistemas complementarios. El tester valida los resultados de las pruebas y configura el escenario que puede ser modificado al inicio o durante las pruebas con el objetivo de asegurar que una prueba específica sea repetida o validar la disponibilidad de algún servicio.

- **muMQ: Un broker de MQTT liviano y escalable**[113]

En los sistemas IoT un broker es un elemento indispensable ya que es el medio de comunicación entre los dispositivos IoT y las plataformas de aplicación. Ya que la cantidad de dispositivos IoT está en crecimiento el broker de mensajería puede saturar su capacidad de rendimiento. Se propone un framework basado en [MQTT](#), diseñado para usar CPUs de múltiples núcleos. Se evaluó la efectividad de este desarrollo y se encontró que puede manejar más 512,000 suscriptores sin pérdida de mensajes, publica 930,000 mensajes por segundo, lo cual es 5,38 veces más rápido que el broker de mensajería MQTT.

- **Middleware para plataformas IoT**[75]

Se propone un middleware diseñado para las necesidades de un sistema IoT. Este middleware está ubicado entre la entidad solicitante y el proveedor. Consta de tres partes: Un broker de mensajería (MQTT), DCO (Data and Context Orchestration) que supervisa el tráfico y ayuda a las aplicaciones a intercambiar datos de forma segura y privada, y una base de datos que funciona como repositorio.

- **Sistema de mensajería a gran escala con brokers coordinados y distribuidos**[112]

Se propone un modelo basado en los brokers de mensajería usados en Fog y Cloud Computing para desarrollar sistemas de mensajería escalables y de baja latencia con el patrón publica/suscribe. Los clusters de mensajería que son usados para fog y cloud computing están ubicados en zonas geográficas estratégicas con el objetivo que los mensajes sean enviados de manera eficiente y los participantes (entes que publican o son suscriptores) se comunican con el broker más cercano. Para coordinar los brokers en el sistema se usó ZooKeeper[34] (Servicio para la coordinación de procesos distribuidos. Se encarga de la gestión de grupos y elección de líder. Es altamente confiable).

5.5 SUPERVISIÓN DE CONTENEDORES

Desde su lanzamiento en el 2013, Docker se ha convertido en la tecnología más usada para implementar contenedores ya que ofrece ventajas como ahorro de recursos, ligereza

e independencia. Debido al uso frecuente de esta tecnología se han desarrollado diversas herramientas para su supervisión. Estas herramientas son instaladas en cada *host* (local o en la nube) a supervisar y encapsuladas en un contenedor para facilitar la implementación y la administración con el objetivo de monitorizar un solo contenedor o un conjunto de ellos. La supervisión consiste en obtener datos de los contenedores. Las métricas más comunes son el uso de CPU, I/O, memoria, memoria disponible y transferencia de datos a través de la red. Las herramientas de monitoreo ofrecen características como agregar plugins, librerías o extensiones, notificaciones o alertas cuando una métrica es excedida y visualización mediante gráficas de las estadísticas[22],[23]. A continuación, se da una breve descripción de algunas de ellas:

- **Docker Stats**[8]: Es un comando de consola que provee Docker para realizar inspección de los recursos consumidos por los contenedores. Muestra el porcentaje de consumo del CPU, la memoria usada y la memoria disponible para el contenedor.
- **CAdvisor**[3]: Ofrece interfaz gráfica para la visualización de los resultados estadísticos dados por Docker Stats. Es una herramienta libre.
- **Scout**[7]: Da la opción para agregar métricas, las estadísticas son más detalladas. Una ventaja sobre CAdvisor es que se pueden adicionar plugins. No es una herramienta libre.
- **Data Dog**[6]: Está alojada en internet. Se debe iniciar una sesión de usuario para sincronizar los contenedores. Se pueden adicionar métricas, da la opción de contar contenedores e imágenes docker y agrupar contenedores por imágenes. El usuario recibe alertas cuando un contenedor excede una métrica. No es una herramienta libre.
- **Sensu**[25]: Se instala y ejecuta forma local en el equipo anfitrión. No es una herramienta sencilla de instalar ya que necesita herramientas auxiliares para su funcionamiento. Es libre.

A continuación, se citan desarrollos los cuales implementan o mejoran este concepto:

- **Sistemas distribuido de pruebas automatizadas**[63]
En este estudio se propone una arquitectura de un sistema distribuido de supervisión basado en contenedores. Los sistemas de pruebas centralizados tiene el inconveniente que las pruebas son ejecutadas sobre los nodos de forma remota y sus resultados pueden ser imprecisos ya que el retraso de algunas redes que puede interferir en la transmisión de los datos. Además, el llevar a cabo pruebas de servicios en la nube implica un gran consumo de recursos e incluso en algunos casos pausar el servicio. Para evitar estos inconvenientes se propone un enfoque distribuido de pruebas automatizadas que consiste en desplegar contenedores Docker con scripts de pruebas en cada nodo a supervisar y de esta manera las pruebas serán realizadas de forma local.
- **Most Privileged Container (MPC), Capa de seguridad para supervisión de contenedores**[120]

La virtualización basada en contenedores provee los beneficios de un sistema completamente aislado y ligero sin embargo, los procesos que se ejecutan dentro de los contenedores pueden acceder al kernel del equipo [sistema anfitrión](#), lo que representa un problema de seguridad. En este estudio se propone una capa de seguridad que ofrece las siguientes funcionalidades: Políticas de Acceso, define el acceso al mecanismo y se encarga que darle permisos a los procesos que lo requieren. Lista Negra, es una base de datos que se actualiza continuamente y contiene los contenedores que son marcados con anomalías. Supervisión en tiempo de ejecución, una vez se le da permiso a un contenedor de ser ejecutado, este se monitoriza para validar su comportamiento.

- **ConMon, Sistema de supervisión para el uso de la red en contenedores**[\[104\]](#)

Las herramientas actuales para supervisar contenedores ofrecen estadísticas como uso de CPU, memoria, I/O, y cantidad de bytes transferidos a través de la red. Otro tipo de herramienta involucra la inyección de código para monitorizar dentro del mismo contenedor, lo cual va en contra de las buenas practicas al usar contenedores ya que, incluye procesos extras que incrementan el riesgo de fallo o uso excesivo de recursos físicos. Hasta ahora ninguna de las herramientas existentes proporciona una solución automatizada para la supervisión dinámica de las métricas de rendimiento de la red, como el retraso, la inestabilidad y la pérdida de paquetes entre los contenedores de la aplicación. Para esto se propone un sistema distribuido y automatizado para la supervisión de la red en contenedores que identifica automáticamente nuevas instancias y observa su tráfico para configurar y ejecutar funciones de supervisión dentro de contenedores adyacentes.

- **Pymon, microframework para supervisión de contenedores orientado a SBC**[\[73\]](#)

Debido al uso masivo de dispositivos móviles se ha retomado el uso de máquinas [SBC](#) ya que, estos dispositivos están caracterizados por el uso eficiente de energía y bajo costo. En este estudio se realizó la implementación de Pymon, un microframework orientado a máquinas SBC implementado en Python (Django). Se ejecuta en un contenedor y recibe las estadísticas de los contenedores monitorizados, estos datos son guardados solo por 7 días para disminuir el consumo de almacenamiento. Las métricas incluyen uso de CPU, memoria y trafico de la red de las interfaces.

TECNOLOGÍAS USADAS

En este capítulo se presentan las tecnologías usadas para el desarrollo del proyecto con el objetivo de justificar su elección y mostrar sus principales características.

En la sección 6.1 se hace una descripción alrededor de los conceptos de virtualización y sus tecnologías. En la sección 6.1.1 se realiza un cuadro comparativo para sustentar el uso de contenedores sobre máquinas virtuales. A continuación, en las secciones 6.1.2 y 6.1.3 se describen las 2 tecnologías que actualmente son dominantes cuando se requiere virtualización mediante contenedores y en 6.1.4 se realiza una comparación entre estas.

Posteriormente, en la sección 6.2 se presenta un estudio sobre las herramientas que apoyan la evaluación en el aprendizaje de programación de computadores. Consta de tres subsecciones. En las secciones 6.2.1 y 6.2.2 se describen las tecnologías DOMjudge y UVA online judge respectivamente y finalmente en la sección 6.2.3 se realiza un cuadro comparativo.

6.1 SELECCIÓN DE TECNOLOGÍA DE VIRTUALIZACIÓN

En esta sección se realiza un cuadro comparativo con el objetivo de argumentar la elección de virtualización mediante contenedores sobre hipervisor. Seguidamente, se exponen las principales características de Docker y UVA online judge, tecnologías de virtualización y finalmente, se realiza un cuadro comparativo.

6.1.1 Comparación de *hypervisors* vs. *contenedor*

En el cuadro 1 se presentan algunas características que diferencian a las tecnologías de virtualización desarrolladas alrededor del concepto de *hypervisor* y las que se apoyan en contenedores.

Característica	Máquinas Virtuales	Contenedores
SO Invitado	Cada VM ejecuta su propio SO. Simulando el hardware y el kernel es cargado en su propia región de memoria.	Todos los contenedores comparten el mismo SO y el kernel. La imagen kernel es cargada dentro de la memoria física.
Comunicación	A través de dispositivos Ethernet.	Mecanismos estándar IPC.
Seguridad	Depende de la implementación del hipervisor.	El control de acceso obligatorio puede ser aprovechado.
Rendimiento	Sufre de sobrecarga, ya que las instrucciones deben ser traducidas del SO invitado al SO anfitrión.	El rendimiento es casi el mismo en comparación con el SO del anfitrión.
Aislamiento	Compartir librerías y archivos entre SO invitados y host no es posible.	Los subdirectorios pueden montarse y compartirse.
Tiempo de inicio	Toma unos pocos minutos.	Respecto al tiempo de las máquinas virtuales, toma segundos.
Almacenamiento	Requiere mucho más almacenamiento. Ya que incluye un sistema operativo completo y kernel, además de los programas en ejecución.	Los contenedores toman menor cantidad de almacenamiento cuando el sistema base es compartido.

Cuadro 1: Comparación de Máquinas virtuales vs. Contenedores. Tomado de: *Virtualization vs Containerization to support PaaS*[66]

6.1.2 Docker

Es una tecnología de virtualización mediante contenedores que se ejecuta en un proceso principal llamado *daemon*. Consiste en empaquetar una aplicación y todas sus dependencias en un entorno aislado que se puede ejecutar en cualquier servidor Linux, ofreciendo flexibilidad y portabilidad. Está basado en LXC, para la implementación del contenedor y la administración de imágenes. Proporciona una capa adicional de abstracción sobre el sistema operativo. [66] Cuenta con un repositorio, *Docker Hub* [82], el cual contiene imágenes con casi cualquier sistema operativo¹ y aplicaciones.

La principal característica de los contenedores Docker y los contenedores en general, es que ofrecen un entorno semejante al de una máquina virtual sin la sobrecarga de ejecutar un kernel separado y simular hardware.

¹ Sistema operativo hace referencia al *kernel* y librerías del sistema.

Aspectos claves de Docker

- Cada contenedor tiene asignado un único id de proceso y una IP privada. Se recomienda ejecutar un solo proceso por contenedor.
- Utiliza características de aislamiento de recursos Kernel de Linux como *cgroups* y *namespaces*, que permiten la ejecución de contenedores independientes en la misma instancia de Linux.
- Aislamiento de red y sistemas de archivos, usando las funcionalidades de LXC.
- El ciclo de vida de un contenedor es administrado usando un *daemon* o la línea de comandos
- Provee la capacidad de almacenar y recuperar el estado de un contenedor.

Aspectos de Seguridad en Docker

- *Cgroups* no evita que un contenedor pueda acceder a los datos y procesos de otros. Sin embargo, juega un importante papel para defenderse de algunos ataques de denegación de servicio (DoS). [59]
- Cuando Docker crea un nuevo contenedor, establece una interfaz de Ethernet virtual con un nombre único que es conectada al adaptador puente de red provisto por Docker. La interfaz también es conectada a la *etho interface* del contenedor. Permitiendo enviar paquetes al adaptador puente. El modelo de conectividad de Docker por defecto es vulnerable a ataques *ARP Spoofing* y a *MAC Flooding* desde el adaptador puente en caso que se reciban todos sus paquetes entrantes de red sin ningún tipo de filtro. [59]
- *Docker Daemon Attack Surface*: El *daemon* de Docker se sitúa entre el cliente Docker y los contenedores en ejecución. El usuario no interactúa directamente con el *daemon*, en su lugar lo hace con el cliente Docker. Este *daemon* necesita privilegios de superusuario, y solamente usuarios de confianza permiten el control del *daemon*. Docker permite compartir un directorio entre el anfitrión y un contenedor. Esto permite acceder al contenedor sin ninguna restricción de permisos. El contenedor será capaz de alterar el sistema de archivos del equipo anfitrión. [59]
- Dado el diseño presentado por los sistemas basados en máquinas virtuales, se considera que estos ambientes son más seguros que los basados en contenedores, ya que agregan una capa extra de aislamiento entre las aplicaciones y el equipo anfitrión. Una aplicación que se ejecuta dentro de una máquina virtual solo es capaz de comunicarse con el kernel de la máquina virtual, no con el del anfitrión. Para que pueda atacar al kernel del anfitrión debe traspasar el kernel de la máquina virtual y el hipervisor. Los contenedores pueden comunicarse directamente con el kernel del anfitrión. Así que permite un acceso más directo al sistema anfitrión, [59].

Arquitectura de Docker

Docker usa una arquitectura cliente-servidor. En la figura 5 se muestran los principales componentes y la interacción entre ellos. Los usuarios de Docker manejan un conjunto de comandos desde el componente cliente para interactuar con el *Docker Host* a través del *Docker daemon* que es ejecutado en el sistema anfitrión. El *Docker cliente* puede interactuar con el *Docker daemon* del mismo sistema, o de manera remota mediante una Restful API o socket. Las imágenes creadas pueden ser alojadas en el sistema anfitrión o en el Docker Register en Internet [87].

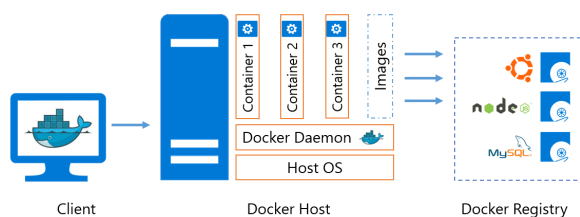


Figura 5: Arquitectura de Docker
Source: <https://docs.docker.com/engine/>

Principales componentes de Docker

■ Imágenes Docker

Una imagen es una plantilla de solo lectura, con la cual el contenedor es inicializado. Consiste en muchas capas y usa [union file systems](#) para combinarlas en una sola. Tiene una imagen base (por lo general es un sistema operativo) en la cual se adicionan los otros componentes que funcionan sobre esta base. Pueden ser construidas localmente o descargadas de un repositorio, [82].

La construcción de imágenes Docker comienza con un archivo llamado *Dockerfile*, donde las instrucciones son almacenadas. Cuando se realiza una solicitud, Docker lee el archivo *Dockerfile*, ejecuta las instrucciones y genera la imagen.

El uso de Docker ayuda a acelerar la portabilidad y disminuye los costos asociados con el almacenamiento de ambientes virtuales. Estas son algunas características relevantes de la tecnología de virtualización por contenedores.[43]

■ Registro Docker

Las imágenes Docker son usualmente almacenadas en repositorios compartidos llamados *Docker Registries*. En los registros se pueden subir y descargar imágenes mediante comandos en el cliente. Los registros facilitan la distribución y junto a otras funcionalidades de la tecnología Docker, contribuyen a crear una nueva modalidad más ágil para distribuir aplicaciones.[87]

Existe un registro público llamado *DockerHub*. [82] Es una gran colección de imágenes existentes listas para usar. La comunidad puede crear nuevas imágenes y agregarlas a la colección o compartirlas a través de repositorios privados.

■ Contenedores Docker

Es un entorno aislado y autónomo. Es una instancia de una imagen Docker. Contiene todo lo necesario para la ejecución de una aplicación. Sus estados más comunes son: en ejecución, iniciado y detenido.

La imagen le indica a Docker qué proceso ejecutar y la configuración para iniciar el contenedor. Las imágenes son de solo lectura, pero cuando se instancia un contenedor, se adiciona una capa de lectura-escritura encima de la imagen usando UFS.[87]

■ Dockerfile

Es un *script* que contiene todos los comandos que deben ejecutarse para construir una imagen Docker dada una sintaxis específica. El Dockerfile empieza cargando la imagen base (sistema operativo), seguido por la lista de comandos para instalar el software deseado. Finalmente, la imagen es construida ejecutando el comando *build*. Este comando ejecuta paso a paso las instrucciones en él. Teniendo como argumento el Dockerfile. [43]

6.1.3 CoreOS RKT

RKT[35] es una tecnología de virtualización mediante contenedores desarrollada para la producción de entornos en la nube orientado a [App Container Specification](#), un nuevo conjunto de especificaciones para la creación de aplicaciones en contenedores. Es la alternativa a Docker más segura, interoperable y de código abierto. Consta de tres partes:

- **ACI:** Es un formato de imagen que contiene todos los archivos necesarios para ejecutar la aplicación.
- **Discovery Mechanism:** Se encarga de ubicar la imagen requerida y convertirla a un archivo descargable.
- **Runtime Environment:** Define cómo la imagen ACI descargada debe ser ejecutada.
- **ACtool:** Es un conjunto de herramientas que se encargan de construir, extraer y validar la imagen descargada.

La unidad central de ejecución de RKT es el *pod*, una colección de una o más aplicaciones que se ejecutan en un contexto compartido. La arquitectura de RKT permite que cada pod sea ejecutado como un proceso de Unix, es decir, sin un *daemon* como es el caso de Docker. Un pod tiene entonces un entorno autónomo y aislado lo que lo hace compatible con systemd, upstart y otros procesos init.

Los pods de RKT se integran con herramientas como Kubernetes y Fleet para realizar orquestación de contenedores y provee un conjunto de instrucciones para realizar operaciones sobre los contenedores.

Arquitectura de RKT

La interfaz principal de RKT es una terminal en donde se introducen los comandos a ejecutar. RKT no requiere un *daemon* principal, esto permite realizar actualizaciones sin afectar los contenedores que están en ejecución y usar diferentes niveles de privilegios para cada operación.

En la figura 6 se muestra el flujo de ejecución de RKT. Este consta de tres escenarios llamados *stages*. Estos escenarios se comunican mediante archivos del sistema, que son bloqueados para garantizar la cooperación y la exclusión mutua entre invocaciones concurrentes del comando rkt.[37]

- **stage0:** Es la ejecución de RKT que se inicia a través de una línea de comandos, posteriormente se convierte en un *child process* de la terminal. En este estado se obtienen las imágenes requeridas por el usuario, se generan el identificador y *pod manifest* (En este archivo se definen las especificaciones de cada aplicación) y se crea los archivos que van a ser pasados a los siguiente escenarios.
- **stage1:** Este escenario obtiene los archivos generados en el paso anterior y se crea el pod o el contenedor configurado de acuerdo al *pod manifest*.
- **stage2:** Es el entorno actual en donde las aplicaciones son ejecutadas.

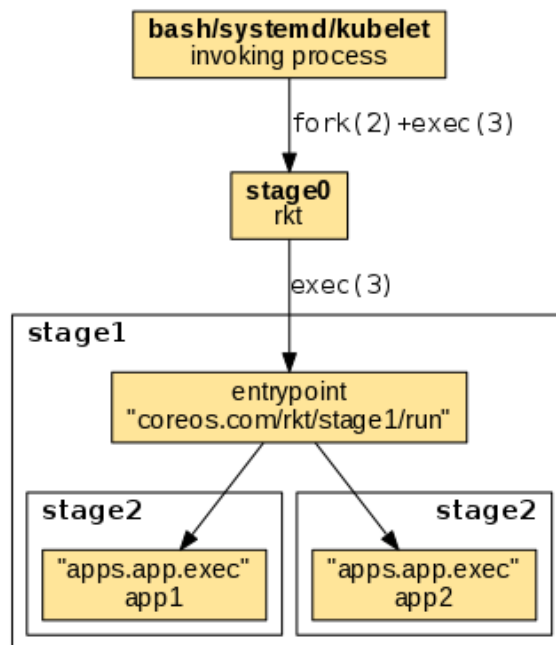


Figura 6: Flujo de ejecución de RKT
Source: <https://coreos.com/rkt/docs/latest/devel/architecture.html>

6.1.4 *Docker vs. Coreos RKT*

A continuación se presenta una tabla comparativa de las características más relevantes de las herramientas Docker y RKT.

Característica	Docker	CoreOS RKT
Servicios	Ofrece <i>Docker Datacenter</i> , solución para orquestar contenedores a nivel empresarial y administración de aplicaciones.	Sus soluciones están más enfocadas a la seguridad. Usa <i>Container Linux</i> , un sistema operativo liviano basado en el kernel de Linux.
Facilidad de uso	Ofrece Kitematic, una solución gráfica para la administración de contenedores	Coreos ofrece Tectonic, una plataforma que permite administrar gráficamente contenedores y clusters.
Fecha de lanzamiento	Versión 1.0 Liberada en Marzo de 2013	Versión 1.0 Liberada en Diciembre de 2014.
Precio	Ambas tecnologías son libres, los servicios de pago incluyen paquetes empresariales y soporte.	
Integración con otras tecnologías	Docker Hub aloja mas de 100.000 aplicaciones libres. Incluyendo repositorios públicos y privados de usuarios y empresas. Provee repositorios oficiales de tecnologías como Nginx, Ubuntu, MongoDB y Redis.	COREOS publica sus proyectos en Github aunque no son tan numerosos como en DockerHub.
Compañías que lo usan	Ambas tecnologías son usadas en importantes empresas como ADP, PayPal, Ebay, BBC News, Spotify, Lyft, Expedia, Groupon, GE Appliances, ING, and Uber, CA Technologies, Verizon, Viacom, Salesforce.com, DigitalOcean, entre otras.	
Curva de aprendizaje	Ninguna de las dos tecnologías fue diseñada para novatos y ambas tecnologías proveen en sus paginas web tutoriales para su aprendizaje.	
Formato de Imagen	Formato de Imagen Docker.	Formato de imagen ACI , sugeridos por APPC .

Cuadro 2: Docker vs CoreOS Rkt.[36]

Debido a que la tecnología Docker es más conocida y lleva más tiempo en el mercado, fue elegida para usarla en el desarrollo de este proyecto ya que en Internet se puede encontrar información, foros de soporte y tutoriales para el uso de esta, ya sea oficiales o de usuarios y cubría las necesidades respecto a virtualización que el proyecto requiere.

6.2 SELECCIÓN DE SOFTWARE PARA EVALUACIÓN AUTOMÁTICA

En esta sección se exponen las principales características de DOMjudge y UVA Online Judge, herramientas para la evaluación automática y se realiza la comparación entre estas.

6.2.1 *DOMjudge*

Es un juez automático en línea usado en los concursos de programación como ACMIPC[83]. Está diseñado para que los equipos envíen la solución a los ejercicios propuestos en un tiempo límite. La evaluación se hace presentando el código fuente de la solución al jurado, el jurado es un proceso o *daemon* en ejecución que compila y ejecuta automáticamente el programa y compara la salida del programa con el resultado esperado. Finalmente, el juez mediante la interfaz gráfica muestra los resultados siendo visible para equipos, jueces y público general. Puede mantener varios concursos en línea simultáneamente.

Su desarrollo comenzó en el 2004 en la en la Universidad de Utrecht. Es de código libre, bajo licencia GNU. Maneja 2 tipos de roles: Juez, encargado de crear el concurso con una configuración específica y Equipos, que son los participantes del concurso.[52]

Para su funcionamiento requiere un servidor con sistema operativo Linux, servidor Apache, PHP 5.3 o posteriores y Mysql para la base de datos.

Características

- Evaluación automática con respuesta inmediata.
- Provee los ejecutables de los lenguajes de programación más comunes.
- Información detallada sobre las soluciones presentadas y la evaluación
- Opciones para realizar una evaluación de nuevo, presentar una solución más de una vez y envío de mensajes a los usuarios
- Interfaz Web portable y simple.
- Sistema escalable

6.2.2 *UVA Online Judge*

UVA Online Judge[29] es una herramienta de evaluación automática alojada en un servidor de la Universidad Valladolid. Usada en concursos de programación y como apoyo al aprendizaje de los lenguajes de programación C++, Pascal, Java y Python. Provee más de 4300 problemas y se puede registrar en la herramienta de forma gratuita disponible para todo el público. Fue creada en 1995 por Miguel Ángel Revilla, dando acceso al público en 1997 y en el 2007 fue completamente remplazada por una nueva versión.

La primera versión de esta herramienta fue enfocada en una audiencia local en la universidad, no tenía funciones para su administración y su interfaz no era amigable para el usuario. Posteriormente fue rediseñada usando Joomla 1.1 CMS, Community Builder Plugin (plugin de Joomla) y un plugin propio, enfocada como una red social usando las siguientes correspondencias[38]:

- Usuarios ->Usuarios.
- Problemas ->Artículos.
- Soluciones ->Comentarios privados en los artículos.
- Evaluación de Soluciones ->Puntuación de los comentarios.
- Veredictos ->Seguimiento de la actividad de un usuario.
- Concursos ->Eventos que integra usuarios y problemas en un intervalo de tiempo.

Finalmente el nuevo diseño ofrece a los usuarios:

- Funcionalidades de una red social como chats, comentarios, foros y seguimiento sobre otros usuarios que han solucionado problemas en los que el usuario está interesado.
- Creación de grupos y subgrupos de usuarios.
- Proponer nuevos problemas y realizar concursos.

6.2.3 *DOMjudge vs. UVA Online Judge*

A continuación se realiza una comparación de las herramientas para evaluación automática mencionadas anteriormente citando sus características más relevantes.

6.2 SELECCIÓN DE SOFTWARE PARA EVALUACIÓN AUTOMÁTICA

Característica	DOMJudge	UVa Online Judge
Alojamiento	En un servidor, con acceso restringido.	En la nube, con acceso al público general.
Facilidad de uso	Interfaz intuitiva, sin embargo tiene opciones e información limitadas para los integrantes de los equipos. Las páginas de soporte y tutoriales son escasos.	Interfaz intuitiva. No maneja perfiles. Al estar disponible al público es más frecuente encontrar tutoriales y foros de soporte.
Lenguajes de programación	Posee ejecutables de los lenguajes de programación más comunes y se pueden integrar otros lenguajes suministrando el ejecutable.	Solo para los lenguajes C, C++, Pascal, Java y Python.
Registro de Usuarios	Los integrantes de los equipos deben ser registrados por el administrador.	El usuario se registra a sí mismo.
Banco de Problemas	El perfil del juez debe crear los concursos y asociar los problemas a estos. No ofrece un banco de problemas general.	Ofrece más de 4000 problemas al público, que pueden ser encontrados por un buscador que ofrece la página y referenciados por un identificador.
Información de ayuda	Ofrece un manual en PDF para los perfiles que maneja el juez.	Ofrece información de cómo subir las soluciones y muestra ejemplos de soluciones por cada lenguaje de programación.
Foros y Soporte	En la página principal publican soluciones a los problemas más comunes con el juez.	Ofrece un foro llamado uHunt[39] dedicado a las soluciones del banco de problemas de UVa OJ.
Estadísticas	Muestra el número de intentos de cada integrante para cada problema por concurso.	Genera gráficas con las estadísticas de las soluciones subidas acertadas, lenguajes de programación más usados, entre otras, con el objetivo de que el usuario realice un seguimiento de su proceso de aprendizaje.

Cuadro 3: DOMjudge vs. UVA Online Judge.

La herramienta para evaluación automática seleccionada para este proyecto ha sido DOMjudge, ya que es una herramienta no alojada en la nube, esta se descarga desde la página

6.2 SELECCIÓN DE SOFTWARE PARA EVALUACIÓN AUTOMÁTICA

oficial y se pone en funcionamiento en un servidor en el cual se puede acceder a la base de datos, su código puede ser modificado y se pueden integrar nuevos lenguajes de programación, características esenciales para la realización de los objetivos de este proyecto.

ARQUITECTURA DE LA SOLUCIÓN

En este capítulo se describe la estructura de la solución propuesta. En la primera sección se muestran los componentes de las interfaces y los datos necesarios para las funciones de la aplicación como son creación de cursos, usuarios e inscripción de los estudiantes a los cursos activos. En la segunda sección se exponen mediante diferentes vistas arquitectónicas la interacción de los módulos necesarios para implementar la aplicación.

La aplicación implementa una arquitectura orientada a servicios. Consta de tres módulos: Una aplicación que se encarga de mostrar la interfaz gráfica al usuario, manejar la autenticación de profesores y administrador y realizar las peticiones a los otros módulos, un api rest que se encarga de gestionar la base de datos de los recursos (imágenes base, periodos académicos, contenedores asociados a cursos y cursos) y un conjunto de funciones que se aplican sobre los contenedores. Estos módulos se comunican mediante una pila de mensajería.



Figura 7: Vista General

7.1 ARQUITECTURA DEL FRONT-END

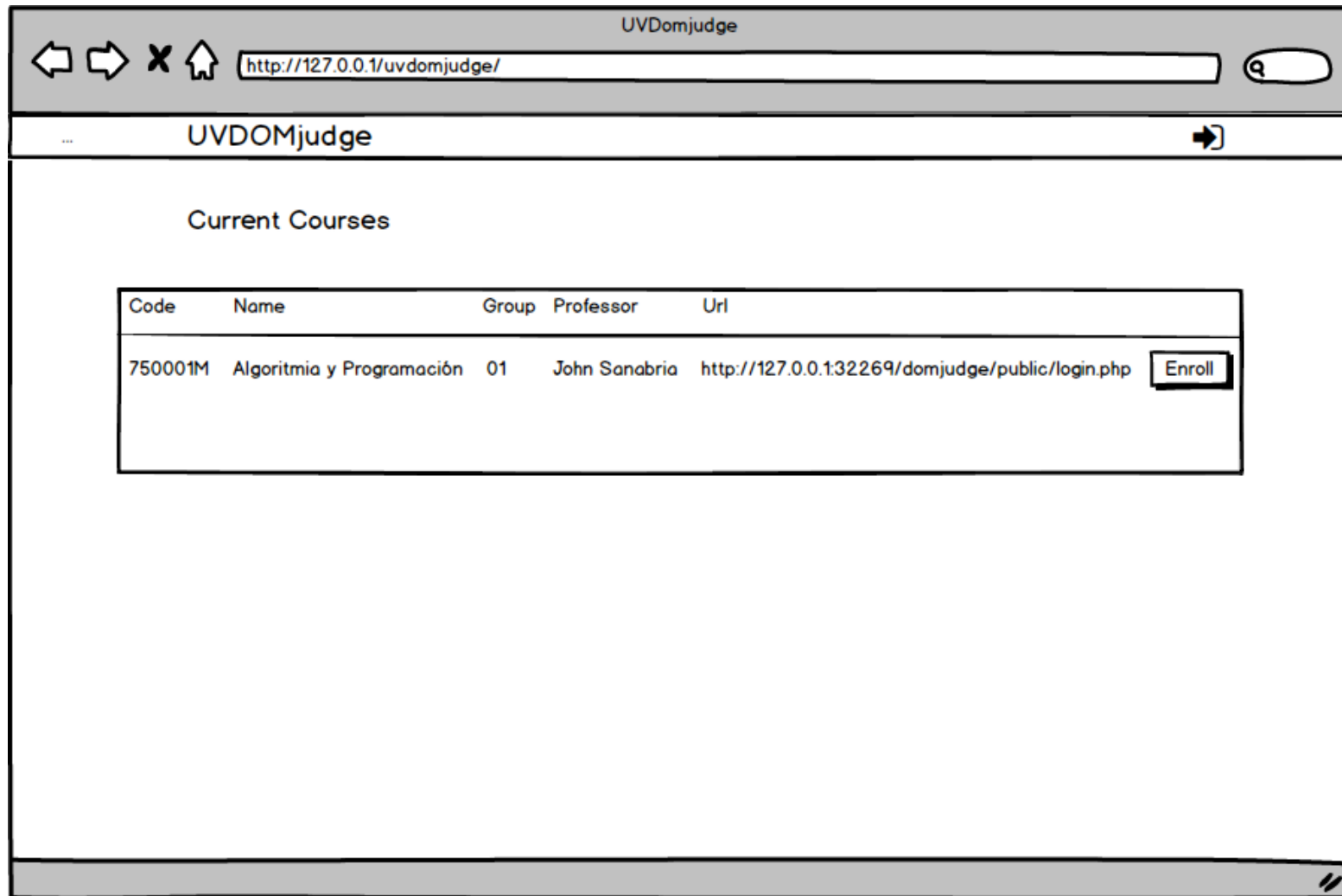


Figura 8: Página Inicial

The image shows a web browser window with the title "UVDomjudge". The address bar contains the URL "http://127.0.0.1/uvdomjudge/201801750001M/enroll". The page content includes a course card for "Algoritmia y Programación" with details: Group: 01, Professor: John Sanabria, and Credits: 4. Below this is a "Register" section with input fields for Code, Name, Last Name, and Email, followed by a "Submit" button.

UVDomjudge

http://127.0.0.1/uvdomjudge/201801750001M/enroll

UVDOMjudge

 **Algoritmia y Programación**
Group: 01
Professor: John Sanabria
Credits: 4

Register

Code

Name

Last Name

Email

Figura 9: Inscripción de los estudiantes a los cursos activos

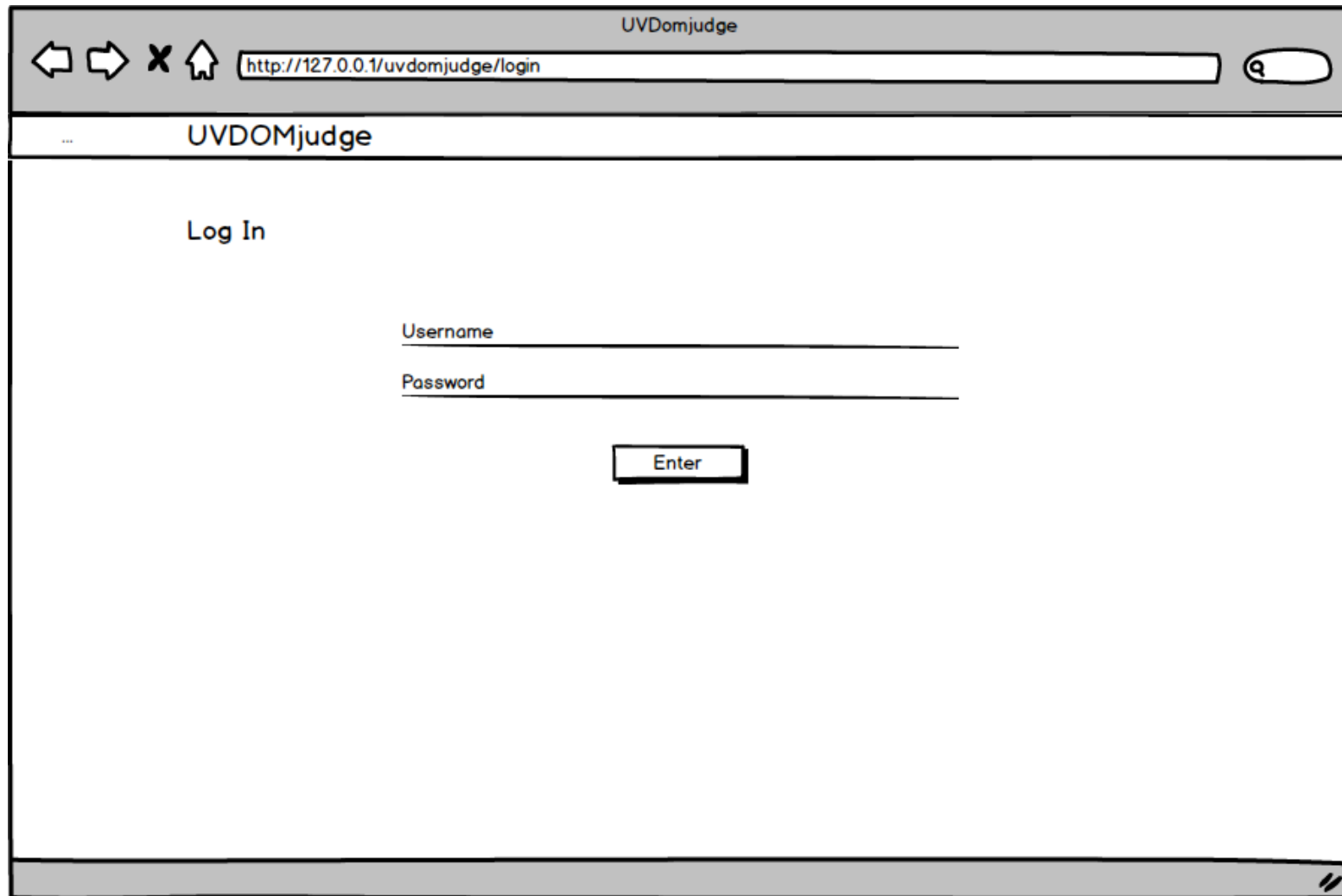


Figura 10: Interfaz de ingreso para administrador y profesores.

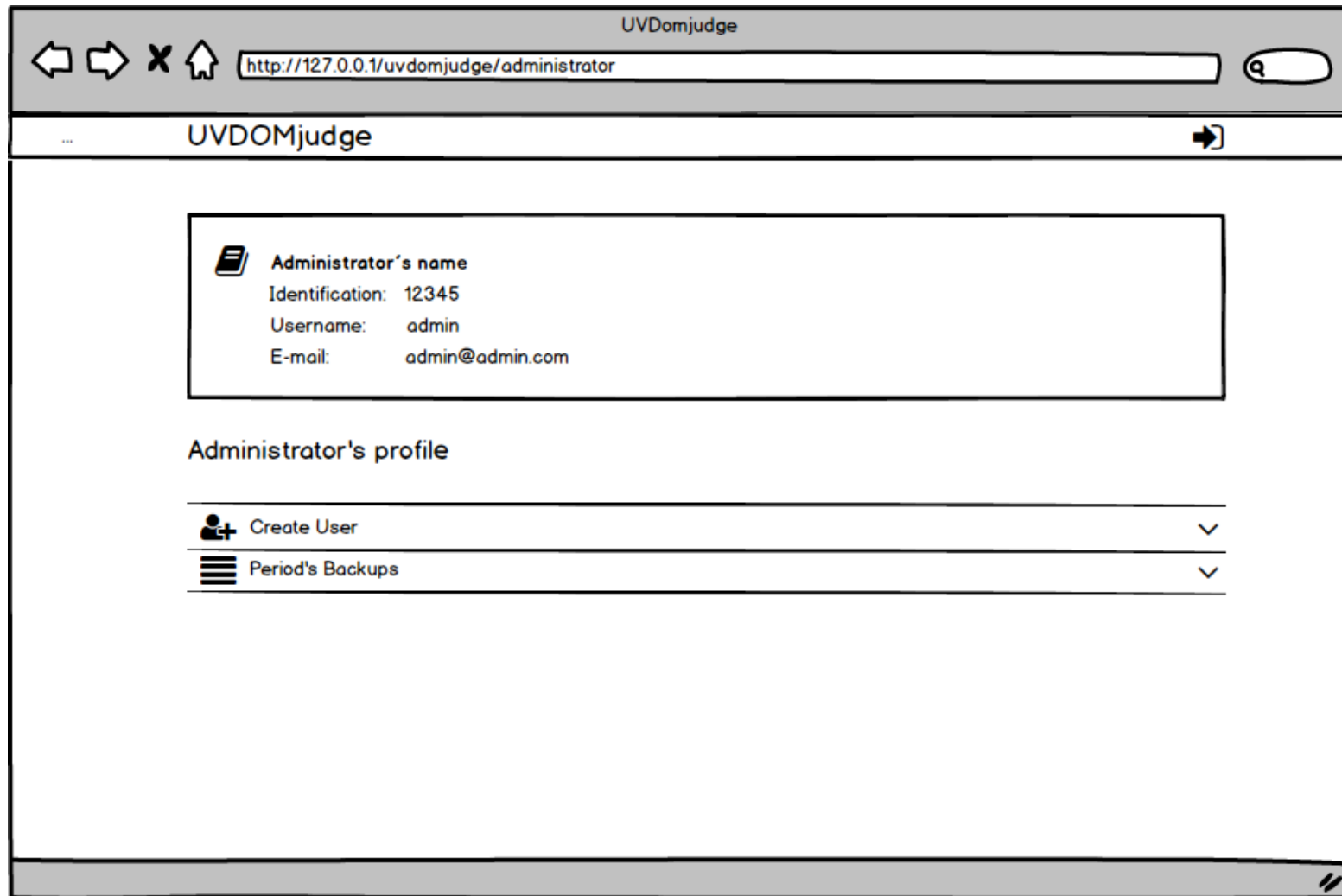


Figura 11: Perfil administrador

The image shows a web browser window with the title "UVDomjudge". The address bar contains the URL "http://127.0.0.1/uvdomjudge/administrator". The browser's tab is labeled "UVDOMjudge". The main content area displays a form titled "Create User" with a user icon and an upward arrow. The form includes a dropdown menu labeled "Type User (Administrator, Professor)", followed by text input fields for "Username", "First", "Last Name", "Email", "Identification", "Password", and "Retype password". A "Create" button is positioned at the bottom of the form. The browser's status bar at the bottom right shows a double-slash icon.

Figura 12: Interfaz para la creación de un nuevo usuario puede ser administrador o profesor.

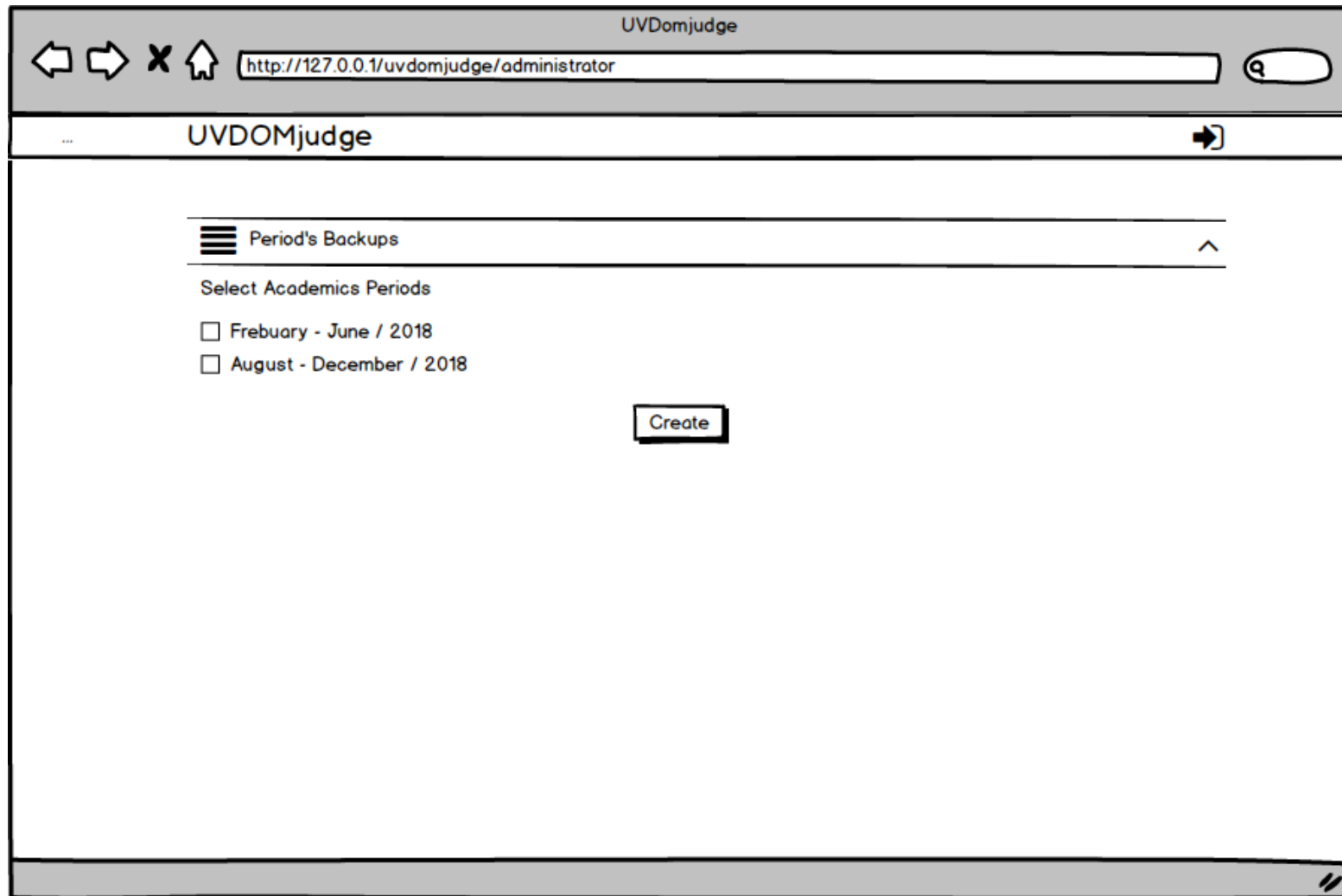


Figura 13: Interfaz para realizar las copias de seguridad los contenedores activos por periodo académico.

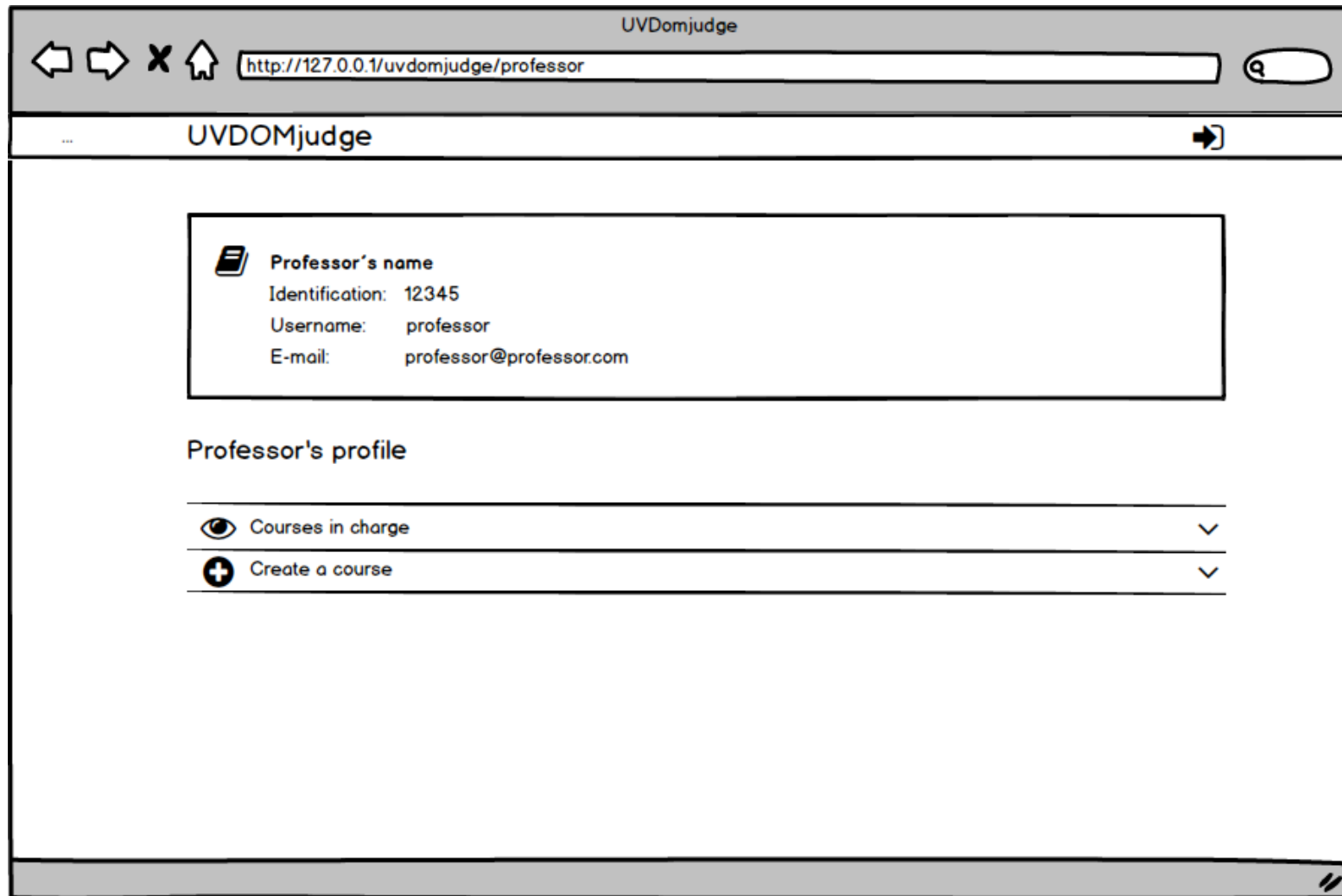


Figura 14: Perfil profesor

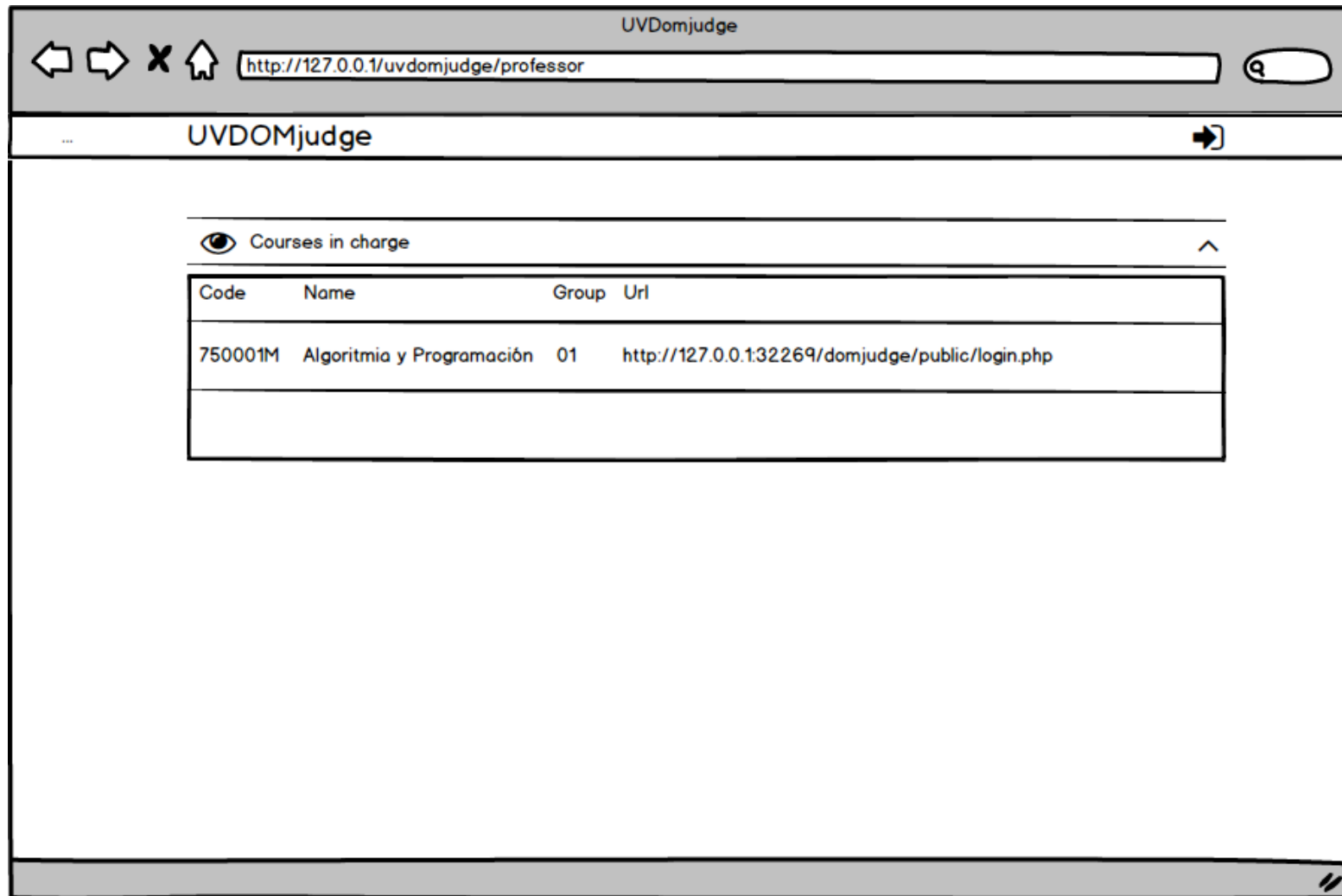


Figura 15: Interfaz que muestra los cursos a cargo del docente.

UVDomjudge

http://127.0.0.1/uvdomjudge/professor

UVDOMjudge

+ Create a course ^

Code (750001M, 750017M) ▼

Name (Algoritmia Y Programación, Métodos Numericos) ▼

Credits (4) ▼

Professor

Professor's identification

Group

Period (February - June , August - December) ▼

Year (current year) ▼

Programming Language (Python, Scilab) ▼

Create

Figura 16: Interfaz para la creación de un curso.

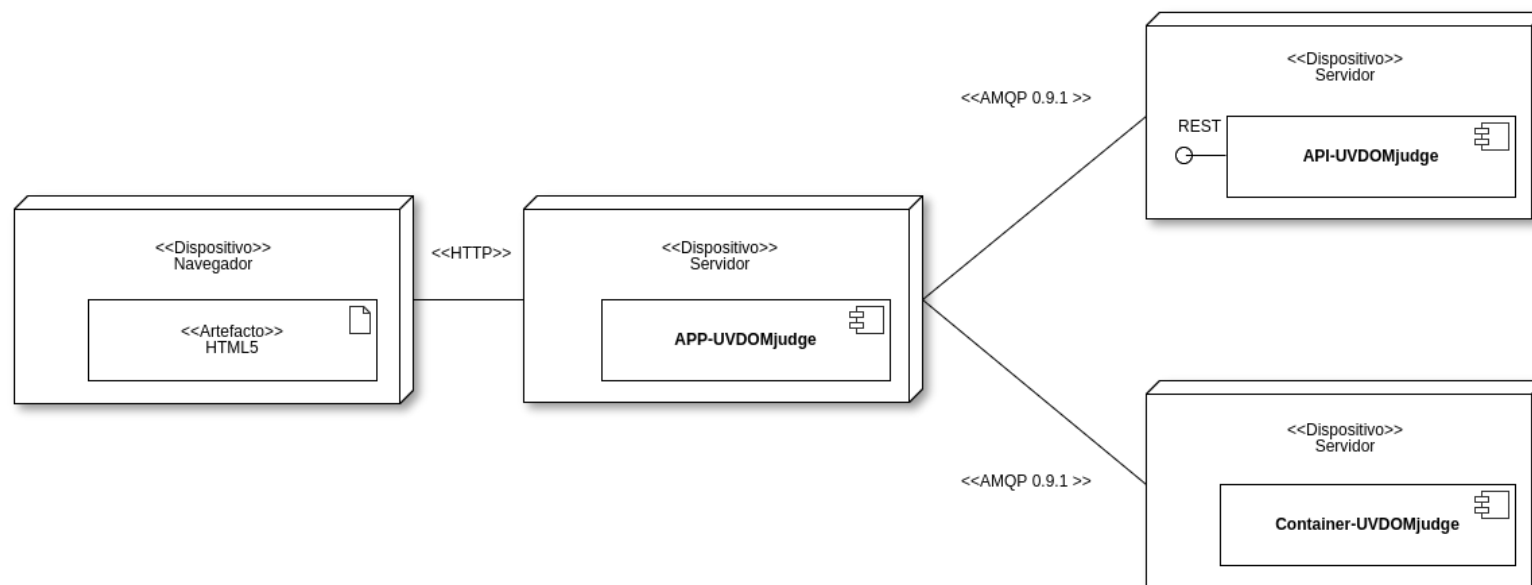


Figura 17: Diagrama de Despliegue

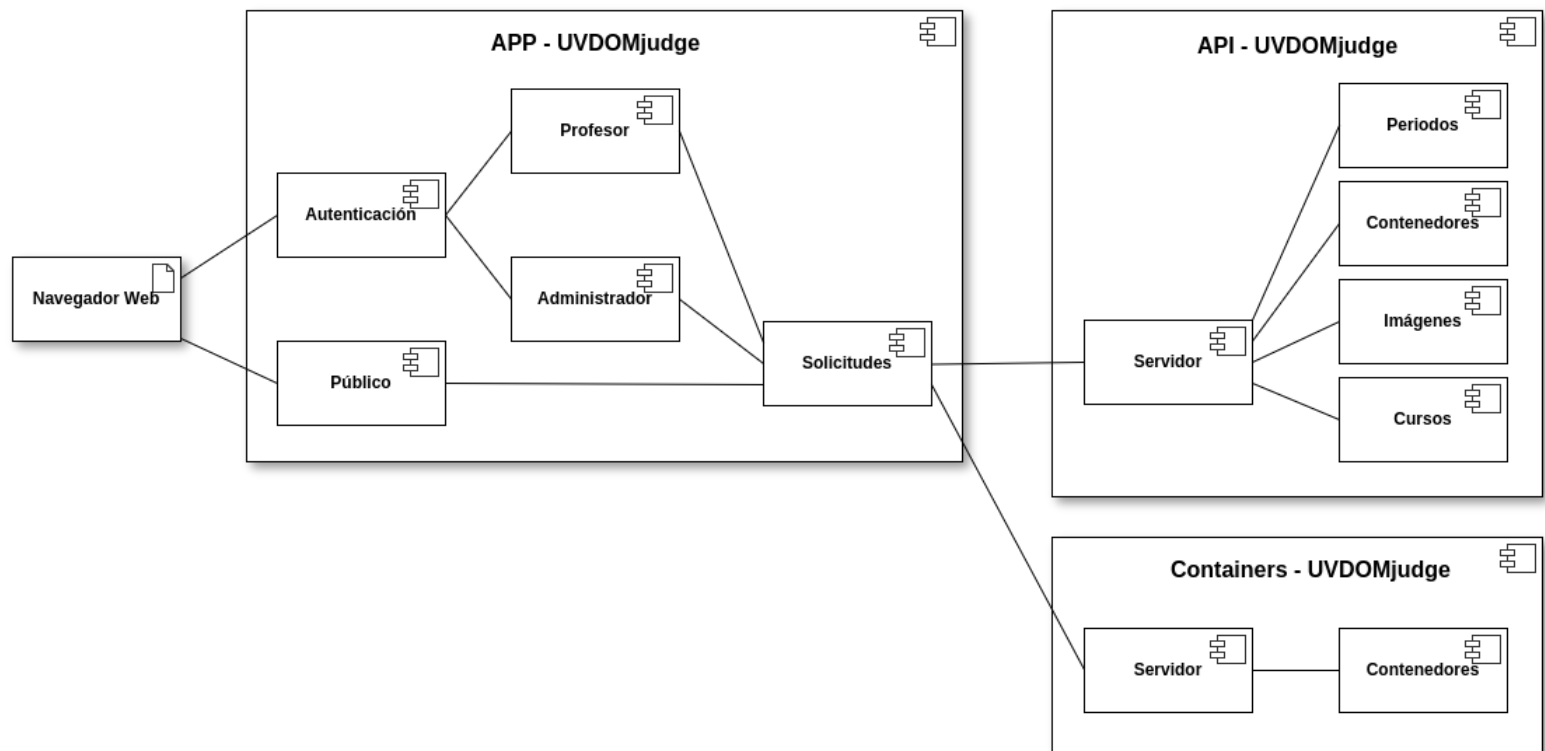


Figura 18: Diagrama de Componentes

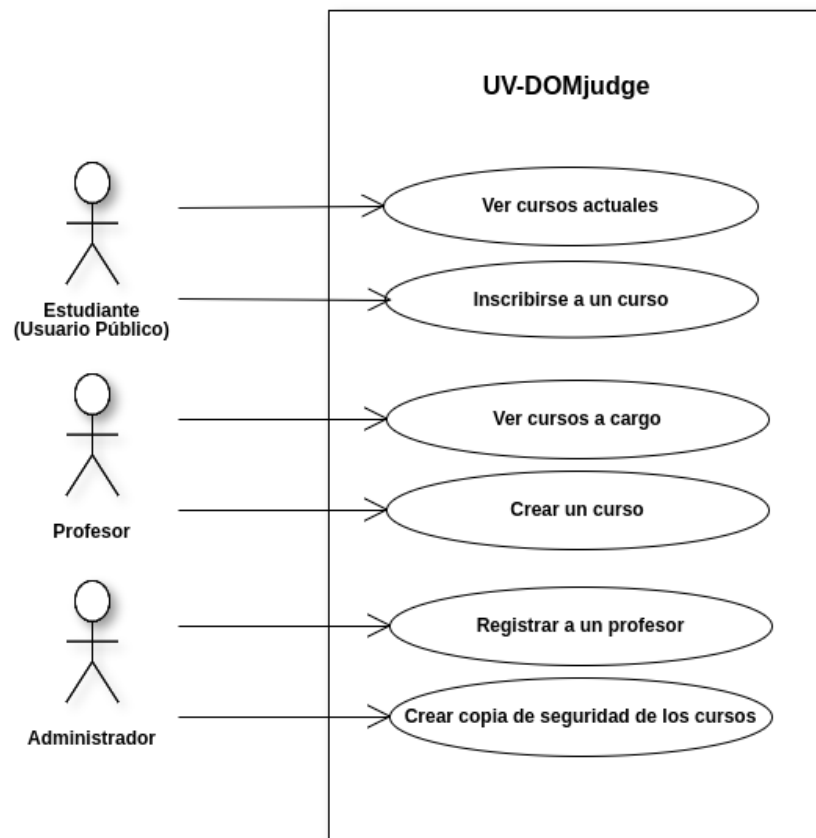


Figura 19: Diagrama de Casos de Uso

RESULTADOS

En este capítulo se realiza la comparación de los objetivos específicos frente a los resultados obtenidos y el análisis de los datos recolectados para este proyecto.

8.1 RESULTADOS OBTENIDOS

Objetivo Específico	Resultado
Seleccionar un juez virtual y una tecnología de virtualización para la creación de los entornos virtuales (Máquinas virtuales ó Contenedores virtuales).	Capítulo 6, Tecnologías usadas.
Diseñar la arquitectura del software (frontend, backend).	Capítulo 7, Arquitectura de la solución.
Desarrollar la lógica del frontend para la recopilación de información sobre el entorno virtual a crear y la adición de estudiantes a un curso.	Aplicación.
Desarrollar la lógica del backend para la creación de los entornos virtuales.	Aplicación.
Caracterizar un proceso de aprendizaje (examen, laboratorio ó actividad de repaso) a gestionar en la plataforma por medio del juez virtual seleccionado y la información a almacenar	Capítulo 8, Resultados de un grupo de estudiantes, sección 8.2 y 8.3. Aplicación.
Realizar pruebas de la aplicación desarrollada sobre el proceso caracterizado en el objetivo anterior y crear un manual de usuario para el manejo de la plataforma.	Capítulo 8, Resultados de un grupo de estudiantes, sección 8.4. Manual de usuario (Archivo PDF).

Cuadro 4: Objetivos específicos vs. resultados.

8.2 DESCRIPCIÓN DE UN CONJUNTO DE EJERCICIOS

A continuación se realiza la descripción del proceso de aprendizaje caracterizado en el juez virtual.

8.2 DESCRIPCIÓN DE UN CONJUNTO DE EJERCICIOS

Se caracterizaron dos ejercicios de repaso voluntario, uno para cada grupo. Estos ejercicios fueron propuestos por los profesores Oscar Bedoya y Juan Manuel Alvarez para los grupos 10 y 13 de Algoritmia y Programación del semestre 2018-01.

La forma de recolección de los datos con el docente Juan Manuel Alvarez fue mediante un taller en la sala de sistemas no calificable, donde los estudiantes subieron sus soluciones. Con el profesor Oscar Bedoya, se recolectaron como un ejercicio opcional para obtener una nota adicional.

A continuación se realiza la descripción de los ejercicios y su competencia asociada.

8.2.1 *Ejercicio 1*

Enunciado: Si la cantidad total de personas es mayor que 5, los precios de las boletas son general a 5000 y primera clase a 10000. Sino, los precios son general a 7000 y primera clase a 12000. En la tabla 5 se muestra la rúbrica asociada a este ejercicio.

8.2.2 *Ejercicio 2*

Enunciado: Diseñe un algoritmo que permita detectar los números pares existentes en una serie de 1 a n. En la tabla 6 se muestra la rúbrica asociada a este ejercicio.

Asignación: Ejercicio 1				
Objetivo Terminal	Actividad	Peso	Nivel 4 - Ejemplar (5.0)	Nivel 1 - Inicial (1.0)
Dar la solución a un problema mediante un programa que implemente estructuras condicionales (if/else) en el lenguaje de programación Python	Realizar la entrega del ejercicio propuesto mediante el juez virtual.	100 %	La evaluación proporcionada por el juez virtual es satisfactoria. Indicando una respuesta correcta al evaluar la solución con las plantillas de entrada del juez.	La evaluación proporcionada por el juez virtual no es satisfactoria. Presenta una respuesta incorrecta o errores de sintaxis/compilación.

Cuadro 5: Rúbrica para el ejercicio 1.

Asignación: Ejercicio 2				
Objetivo Terminal	Actividad	Peso	Nivel 4 - Ejemplar (5.0)	Nivel 1 - Inicial (1.0)
Dar la solución a un problema mediante un programa que implemente estructuras iterativas (for) en el lenguaje de programación Python	Realizar la entrega del ejercicio propuesto mediante el juez virtual.	100 %	La evaluación proporcionada por el juez virtual es satisfactoria. Indicando una respuesta correcta al evaluar la solución con las plantillas de entrada del juez.	La evaluación proporcionada por el juez virtual no es satisfactoria. Presenta una respuesta incorrecta o errores de sintaxis/compilación.

Cuadro 6: Rúbrica para el ejercicio 2.

8.3 CONFIGURACIÓN DEL JUEZ VIRTUAL PARA LA CARACTERIZACIÓN DEL PROCESO DE APRENDIZAJE

8.3 CONFIGURACIÓN DEL JUEZ VIRTUAL PARA LA CARACTERIZACIÓN DEL PROCESO DE APRENDIZAJE

Para caracterizar los ejercicios mencionados anteriormente se configuró el juez virtual DOMjudge de la siguiente manera:

Para cada grupo se desplegó un contenedor con las herramientas necesarias para la ejecución de Domjudge. Los estudiantes podían acceder a los contenedores mediante una dirección IP y un puerto. Los contenedores estaban en un servidor de la EISC y podían ser accedidos desde las salas de sistemas del edificio 331 y la red de la escuela. En la figura 20 se muestra el nombre de cada contenedor y el puerto de acceso a estos. Para el grupo 10 el contenedor asignado es el 201801750001M10 con el puerto 32769 y para el grupo 13 el contenedor con el nombre 201801750001M13 con el puerto 32678.

En ambos contenedores solo se habilitó el lenguaje de programación Python 3 y una modificación para que el juez pueda evaluar funciones llamado Python 3 Functional. En las figuras 21 y 22 se muestran habilitados los lenguajes Python y Python Functional en el juez virtual para ambos grupos.

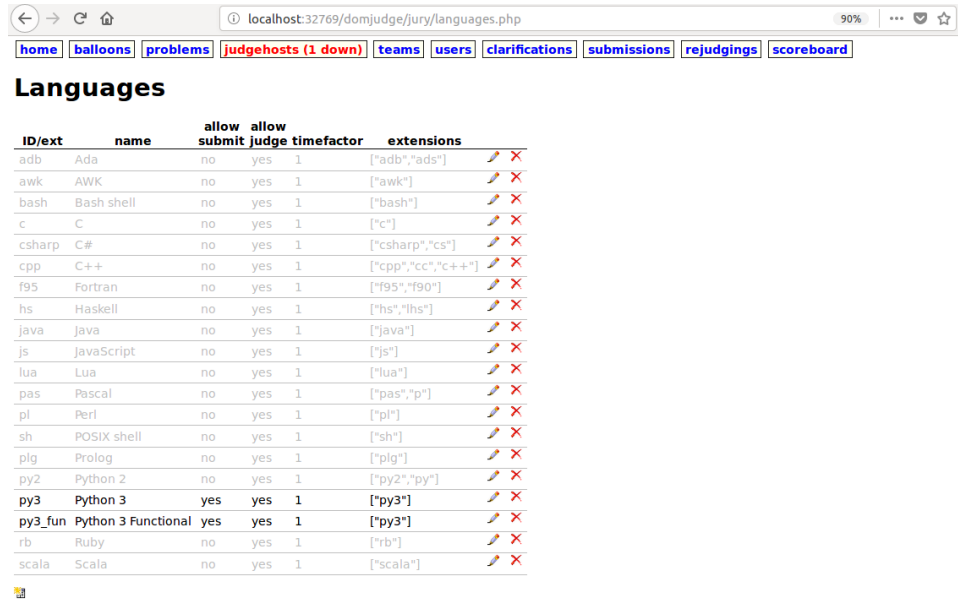
Todos los estudiantes matriculados en los grupos 10 y 13 del curso de algoritmia y programación fueron registrados en la base de datos del juez en el contenedor correspondiente para ingresar a la herramienta y subir sus soluciones. DOMjudge realiza la evaluación de las soluciones sobre un equipo y no sobre un individuo, de esta manera cada estudiante estaba asociado a un equipo de un solo integrante identificado con el código de estudiante de la universidad. En la figura 23 se muestran los equipos registrados y en la figura 24 el resumen de envío de soluciones realizados para el grupo 10. De igual manera en la figura 25 y 26 muestran los equipos y resumen de envíos para el grupo 13.

Domjudge maneja concursos y problemas. Un concurso es el conjunto de uno o varios problemas asociado a este en donde se pueden subir sus soluciones en un tiempo establecido y un problema realiza la asociación de las soluciones subidas a unos casos de prueba. Para los ejercicios propuestos se creó un concurso simulando un taller y un problema con sus casos de prueba para cada ejercicio. Para el grupo 10 se asignó el ejercicio 2 descrito en la sección 8.2.2, en la figura 27 se muestra el concurso realizado para este ejercicio y en la figura 28 su resumen de evaluaciones. De igual manera en las figuras 29 y 30 el concurso y el resumen de envíos para el grupo 13 con el ejercicio 1 descrito en la sección 8.2.1.

```
root@macripco-OptiPlex-780:/home/macripco# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                               NAMES
34874e1a0528        201801750001m13    "/home/domjudge/do...  2 months ago       Up 1 second        0.0.0.0:32769->80/tcp, 0.0.0.0:32768->3306/tcp 201801750001M10
94d843400459        201801750001m13    "/home/domjudge/do...  2 months ago       Up About a minute  0.0.0.0:32678->80/tcp, 0.0.0.0:32679->3306/tcp 201801750001M13
root@macripco-OptiPlex-780:/home/macripco#
```

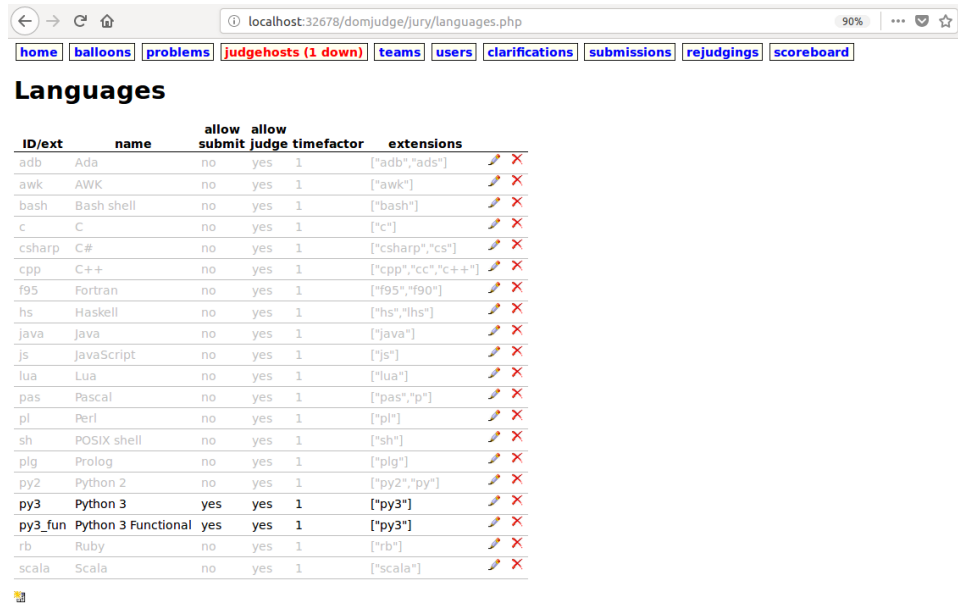
Figura 20: Captura de la información de los contenedores de cada grupo.

8.3 CONFIGURACIÓN DEL JUEZ VIRTUAL PARA LA CARACTERIZACIÓN DEL PROCESO DE APRENDIZAJE



ID/ext	name	allow submit	allow judge	timefactor	extensions
adb	Ada	no	yes	1	["adb","ads"]
awk	AWK	no	yes	1	["awk"]
bash	Bash shell	no	yes	1	["bash"]
c	C	no	yes	1	["c"]
csharp	C#	no	yes	1	["csharp","cs"]
cpp	C++	no	yes	1	["cpp","cc","c++"]
f95	Fortran	no	yes	1	["f95","f90"]
hs	Haskell	no	yes	1	["hs","lhs"]
java	Java	no	yes	1	["java"]
js	JavaScript	no	yes	1	["js"]
lua	Lua	no	yes	1	["lua"]
pas	Pascal	no	yes	1	["pas","p"]
pl	Perl	no	yes	1	["pl"]
sh	POSIX shell	no	yes	1	["sh"]
plg	Prolog	no	yes	1	["plg"]
py2	Python 2	no	yes	1	["py2","py"]
py3	Python 3	yes	yes	1	["py3"]
py3_fun	Python 3 Functional	yes	yes	1	["py3"]
rb	Ruby	no	yes	1	["rb"]
scala	Scala	no	yes	1	["scala"]

Figura 21: Lenguajes Python y Python Functional habilitados para el grupo 10.



ID/ext	name	allow submit	allow judge	timefactor	extensions
adb	Ada	no	yes	1	["adb","ads"]
awk	AWK	no	yes	1	["awk"]
bash	Bash shell	no	yes	1	["bash"]
c	C	no	yes	1	["c"]
csharp	C#	no	yes	1	["csharp","cs"]
cpp	C++	no	yes	1	["cpp","cc","c++"]
f95	Fortran	no	yes	1	["f95","f90"]
hs	Haskell	no	yes	1	["hs","lhs"]
java	Java	no	yes	1	["java"]
js	JavaScript	no	yes	1	["js"]
lua	Lua	no	yes	1	["lua"]
pas	Pascal	no	yes	1	["pas","p"]
pl	Perl	no	yes	1	["pl"]
sh	POSIX shell	no	yes	1	["sh"]
plg	Prolog	no	yes	1	["plg"]
py2	Python 2	no	yes	1	["py2","py"]
py3	Python 3	yes	yes	1	["py3"]
py3_fun	Python 3 Functional	yes	yes	1	["py3"]
rb	Ruby	no	yes	1	["rb"]
scala	Scala	no	yes	1	["scala"]

Figura 22: Lenguajes Python y Python Functional habilitados para el grupo 13.

8.3 CONFIGURACIÓN DEL JUEZ VIRTUAL PARA LA CARACTERIZACIÓN DEL PROCESO DE APRENDIZAJE

ID	teamname	category	affiliation	# contests	host	room	status
t1	DOMjudge	System	1	-	-	-	0/0
t2	Example teamname	Participants	UU	1	-	-	0/0
t201429616	MONTAÑO CHIRIBOGA JAMINSON ANDREY	Participants	1	-	-	-	0/0
t201432956	CHAVES SALAZAR JUAN SEBASTIAN	Participants	1	-	-	-	0/0
t201456088	IZQUIERDO QUINTERO LUISA FERNANDA	Participants	1	-	-	-	0/0
t201526066	RODRIGUEZ CARDOZO IVAN DARIO	Participants	1	-	-	-	0/0
t201630799	MARIA CRISTINA PORTILLA	Participants	1	172.18.0.1	-	-	3/3
t201632812	RIOS LINCE GEOVANNY	Participants	1	-	-	-	0/0
t201727392	MONTENEGRO CASTILLO JUAN ESTEBAN	Participants	1	-	-	-	0/0
t201728036	VERGARA MORENO BRAYAN DAVID	Participants	1	-	-	-	0/0
t201729015	DOLLS SILVA JHONY ESNEYDER	Participants	1	-	-	-	0/0
t201730798	BUENO MARTINEZ JHON EDISON	Participants	1	-	-	-	0/0
t201731646	CASTILLO CHAPARRO KAREN LIZETH	Participants	1	s4pc9	-	-	0/0
t201740617	OROZCO TROCHEZ LUIS DAVID	Participants	1	-	-	-	0/0
t201740801	CARABALI VIAFARA DANIEL ALEJANDRO	Participants	1	-	-	-	0/0
t201741331	ARANGO AGUIRRE ANGIE LORENA	Participants	1	s4pc15	-	-	1/1
t201741398	CEDEÑO AGUAYO JUAN CAMILO	Participants	1	s4pc13	-	-	1/1
t201741602	SUAREZ URREA VALENTINA	Participants	1	s4pc21	-	-	1/3
t201741763	ANGARITA ROMAN KEVIN DAVID	Participants	1	-	-	-	0/0
t201742032	GARCIA CHARRY JUAN DAVID	Participants	1	s4pc11	-	-	1/2
t201742299	ARANGO RODAS DANIEL ESTEBAN	Participants	1	s4pc23	-	-	0/0
t201742937	POSSU VALENCIA MIGUEL ANGEL	Participants	1	s4pc1	-	-	1/1
t201743063	ECHEVERRI SABOGAL JUAN ESTEBAN	Participants	1	sala4	-	-	1/2
t201743076	DIAZ OROBIO DANNY FERNANDO	Participants	1	172.18.0.1	-	-	1/1
t201743611	OSPINA BEJARANO DANIEL FERNANDO	Participants	1	s4pc18	-	-	1/4
t201743754	CASTILLO PATIÑO CARLOS ALFONSO	Participants	1	s4pc14	-	-	1/1
t201744480	PALAU MAFLA JOSE DAVID	Participants	1	-	-	-	0/0
t201744900	PAJOY ROJAS JOHAN DAVID	Participants	1	s4pc22	-	-	1/1
t201744932	JIMENEZ ALVAREZ SERGIO SANTIAGO	Participants	1	-	-	-	0/0
t201745270	MARTINEZ MONCADA MARIA CAMILA	Participants	1	s4pc15	-	-	1/1
t201745385	SUESCUN AGREDO WILMAR	Participants	1	-	-	-	0/0
t201745692	VILLA RODRIGUEZ JULIANA	Participants	1	s4pc20	-	-	1/2
t201745700	GIL ORTEGA LUIS ALEJANDRO	Participants	1	-	-	-	0/0
t201746628	OSPINA RENDON CHRISTIAN ALEXANDER	Participants	1	s4pc16	-	-	1/1
t201770900	SALAZAR BERRIO NATALIA	Participants	1	-	-	-	0/0
t201825102	PERDOMO PORTILLA JOHN JAIRO	Participants	1	s4pc19	-	-	1/2
t201832471	ORDÓÑEZ BOTINA HAROLD ALONSO	Participants	1	-	-	-	0/0

Figura 23: Equipos registrados para el grupo 10.

STARTS: 19:40 - ENDS: 19:40

RANK	TEAM	SCORE	P1	P2	P3
1	MARIA CRISTINA PORTILLA	3 45	1/32	1/6	1/7
2	MARTINEZ MONCADA MAR	1 4446	1/4446	0	0
3	ARANGO AGUIRRE ANGIE	1 4454	1/4454	0	0
4	CASTILLO PATIÑO CARLOS	1 4457	1/4457	0	0
5	CEDEÑO AGUAYO JUAN CA	1 4461	1/4461	0	0
6	PAJOY ROJAS JOHAN DAVI	1 4463	1/4463	0	0
7	OSPINA RENDON CHRISTI	1 4465	1/4465	0	0
8	GARCIA CHARRY JUAN DA	1 4467	2/4447	0	0
9	DIAZ OROBIO DANNY FERI	1 4468	1/4468	0	0
10	PERDOMO PORTILLA JOHN	1 4479	2/4459	0	0
11	POSSU VALENCIA MIGUEL	1 4483	1/4483	0	0
12	VILLA RODRIGUEZ JULIAN	1 4487	2/4467	0	0
13	ECHEVERRI SABOGAL JUA	1 4497	2/4477	0	0
14	SUAREZ URREA VALENTIN	1 4500	3/4460	0	0
15	OSPINA BEJARANO DANIEL	1 4514	4/4454	0	0
16	ANGARITA ROMAN KEVIN	0 0	0	0	0

Figura 24: Resumen de envíos por equipos para el grupo 10.

8.3 CONFIGURACIÓN DEL JUEZ VIRTUAL PARA LA CARACTERIZACIÓN DEL PROCESO DE APRENDIZAJE

ID	teamname	category	affiliation	# contests	host	room	status
t1	DOMjudge	System		3	-		0 / 0
t2	Example teamname	Participants	UU	3	-		0 / 0
t201422911	MATEUS OSSA ALEJANDRO ANDRES	Participants		3	172.17.0.1		1 / 1
t201623406	HENAO MAYOR JOSE EDUARDO	Participants		3	-		0 / 0
t201630799	PORTILLA CORTES MARIA CRISTINA	Participants		3	172.18.0.1		3 / 3
t201631336	CORTES GONZALES DANIEL FERNANDO	Participants		3	-		0 / 0
t201710187	QUINTERO VELEZ ALEJANDRO	Participants		3	-		0 / 0
t201738421	CABRERA HIDALGO SOFIA	Participants		3	-		0 / 0
t201740047	MELO MORA ANGELA CAMILA	Participants		3	ofidhcl1		1 / 3
t201740093	TERAN VALLEJO ANGELA GABRIELA	Participants		3	-		0 / 0
t201740111	ALPALA CANACUAN SANDRA DEL ROCIO	Participants		3	ofidhcl1		1 / 1
t201741067	SARMIENTO GOMEZ JUAN DAVID	Participants		3	ofidhcl1		1 / 2
t201741262	MARTINEZ POLANCO JOHANNA ANDREA	Participants		3	-		0 / 0
t201741865	CARDENAS SAENZ JOSE SAMUEL	Participants		3	-		0 / 0
t201741969	GAVIRIA MARTINEZ ZHARICH ODLET	Participants		3	-		0 / 0
t201742166	MOSQUERA RODRIGUEZ FABIAN STEVEN	Participants		3	-		0 / 0
t201742175	MUNOZ DOMINGUEZ LUISA FERNANDA	Participants		3	ofidhcl1		1 / 2
t201743670	MUNOZ GARCES JUAN DAVID	Participants		3	-		0 / 0
t201743797	VELEZ CASTILLO JUAN SEBASTIAN	Participants		3	-		0 / 0
t201744935	QUENGUAN LOAIZA JUAN PAULO	Participants		3	ofidhcl1		1 / 1
t201745020	PRADA QUINCHUA MARTIN SEBASTIAN	Participants		3	-		0 / 0
t201745083	QUENAN MELO YULISA ALEXANDRA	Participants		3	-		0 / 0
t201745177	PANTOJA ALVARADO CAROLINA	Participants		3	-		0 / 0
t201745510	JIMENEZ PERDOMO DIEGO ALEJANDRO	Participants		3	-		0 / 0
t201745572	CASTAÑO QUINONES SOFIA	Participants		3	ofidhcl1		1 / 6
t201746703	CAMACHO DAZA STEFANIA	Participants		3	-		0 / 0
t201747614	TASCON MUECES LINA MARCELA	Participants		3	-		0 / 0
t201747652	REYES SIERRA SANTIAGO JOSE	Participants		3	-		0 / 0
t201748754	ORTIZ FERNANDEZ NATALIA	Participants		3	-		0 / 0
t201748863	CERON DAZA LAURA DANIELA	Participants		3	ofidhcl1		1 / 1

Figura 25: Equipos registrados para el grupo 13.

RANK	TEAM	SCORE	PS3 [5 POINTS]
1	PORTILLA CORTES MARIA	5 2	1/2
2	CASTAÑO QUINONES SOFI	5 8979	1/8979
3	CERON DAZA LAURA DANI	5 8986	1/8986
4	MATEUS OSSA ALEJANDRO	5 8987	1/8987
5	ALPALA CANACUAN SANDI	5 8989	1/8989
6	QUENGUAN LOAIZA JUAN	5 8992	1/8992
7	MUNOZ DOMINGUEZ LUISA	5 9001	2/8981
8	SARMIENTO GOMEZ JUAN	5 9011	2/8991
9	MELO MORA ANGELA CAM	5 9024	3/8984
10	CABRERA HIDALGO SOFIA	0 0	0
	CAMACHO DAZA STEFANIA	0 0	0
	CARDENAS SAENZ JOSE SA	0 0	0

Figura 26: Resumen de envíos por equipos para el grupo 13.

8.3 CONFIGURACIÓN DEL JUEZ VIRTUAL PARA LA CARACTERIZACIÓN DEL PROCESO DE APRENDIZAJE

The screenshot shows a web browser at localhost:32769/domjudge/jury/contests.php. The navigation bar includes links for home, balloons, problems, judgehosts (1 down), teams, users, clarifications, submissions, rejudgetings, and scoreboard. The main heading is 'Contests'. Under 'CURRENT CONTESTS:', there is a section for 'PROBLEMAS CICLO FOR (PCF - c4)' with the following details:

- Activate time: 2018-03-02 19:40 (UTC)
- Start time: 2018-03-02 19:40 (UTC)
- Freeze time: [freeze now]
- ... End time: 2019-03-02 19:40 (UTC) [end now]
- Unfreeze time:
- Deactivate time: 2019-03-02 19:40 (UTC)

Below this, the 'All available contests' table is shown:

CID	shortcode	activate	start	freeze	end	unfreeze	deactivate	balloons?	public?	# teams	# problems	name
c4	PCF	19:40	19:40	-	19:40	-	19:40	yes	yes	all	3	Problemas Ciclo For

Figura 27: Concurso asociado al ejercicio 2 para el grupo 10.

Contests

contest CID	shortcode	contest name	problem shortcode	allow submit	allow judge	colour
c4	PCF	Problemas Ciclo For	P1	yes	yes	blue

Submissions for Problema1

ID	time	team	problem	lang	result	verified	by
s29	22:23	POSSU VALENCIA MIGUEL ANGEL	P1	PY3	CORRECT	no	claim
s28	22:17	ECHEVERRI SABOGAL JUAN ESTEBAN	P1	PY3	CORRECT	no	claim
s27	22:08	DIAZ OROBIO DANNY FERNANDO	P1	PY3	CORRECT	no	claim
s26	22:07	ECHEVERRI SABOGAL JUAN ESTEBAN	P1	PY3	COMPILER-ERROR	claimed	unclaim
s25	22:07	VILLA RODRIGUEZ JULIANA	P1	PY3	CORRECT	no	claim
s24	22:05	VILLA RODRIGUEZ JULIANA	P1	PY3	WRONG-ANSWER	claimed	unclaim
s23	22:05	OSPINA RENDON CHRISTIAN ALEXAN...	P1	PY3	CORRECT	no	claim
s22	22:03	PAJOY ROJAS JOHAN DAVID	P1	PY3	CORRECT	no	claim
s21	22:01	CEDEÑO AGUAYO JUAN CAMILO	P1	PY3	CORRECT	no	claim
s20	22:00	SUAREZ URREA VALENTINA	P1	PY3	CORRECT	no	claim
s19	21:59	PERDOMO PORTILLA JOHN JAIRO	P1	PY3	CORRECT	no	claim
s18	21:58	SUAREZ URREA VALENTINA	P1	PY3	WRONG-ANSWER	no	claim
s17	21:57	CASTILLO PATIÑO CARLOS ALFONSO	P1	PY3	CORRECT	no	claim
s16	21:54	ARANGO AGUIRRE ANGIE LORENA	P1	PY3	CORRECT	no	claim
s15	21:54	OSPINA BEJARANO DANIEL FERNANDO	P1	PY3	CORRECT	no	claim
s14	21:53	SUAREZ URREA VALENTINA	P1	PY3	WRONG-ANSWER	no	claim
s13	21:52	OSPINA BEJARANO DANIEL FERNANDO	P1	PY3	WRONG-ANSWER	claimed	unclaim
s12	21:51	OSPINA BEJARANO DANIEL FERNANDO	P1	PY3	WRONG-ANSWER	no	claim
s11	21:47	GARCIA CHARRY JUAN DAVID	P1	PY3	CORRECT	no	claim
s10	21:46	GARCIA CHARRY JUAN DAVID	P1	PY3	WRONG-ANSWER	no	claim
s9	21:46	MARTINEZ MONCADA MARIA CAMILA	P1	PY3	CORRECT	no	claim
s8	21:35	PERDOMO PORTILLA JOHN JAIRO	P1	PY3	WRONG-ANSWER	no	claim
s7	21:29	OSPINA BEJARANO DANIEL FERNANDO	P1	PY3	WRONG-ANSWER	no	claim
s6	20:12	MARIA CRISTINA PORTILLA	P1	PY3	CORRECT	no	claim

Figura 28: Resumen de evaluaciones por equipos del grupo 10 para el ejercicio 2.

8.4 RESULTADOS DE UN GRUPO DE ESTUDIANTES

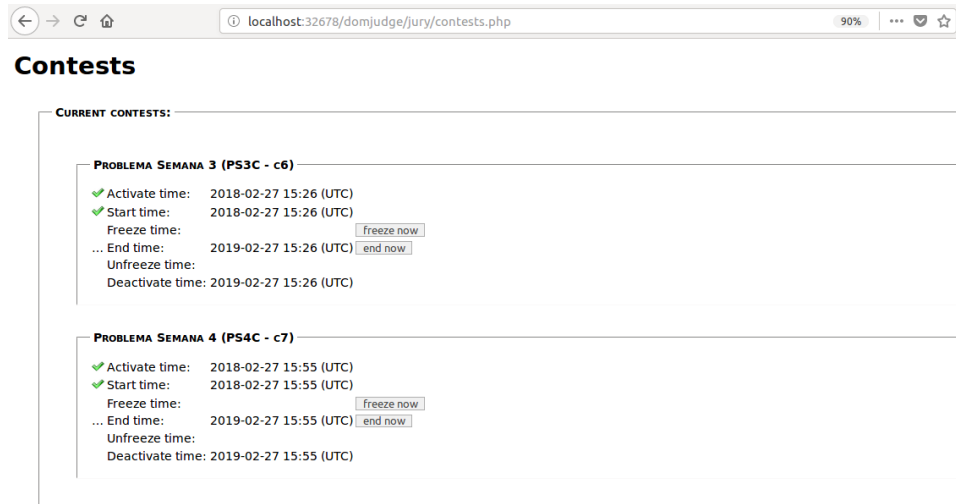


Figura 29: Concurso asociado al ejercicio 1 para el grupo 13.

Contests									
contest CID	contest shortcode	contest name	problem shortcode	allow submit	allow judge	colour			
c6	PS3C	Problema Semana 3	PS3	yes	yes				

Submissions for PS3									
ID	contest	time	team	problem	lang	result	verified	by	
s25	c6	21:18	QUENGUAN LOAIZA JUAN PAULO	PS3	Py3	CORRECT	no	claim	
s24	c6	21:17	SARMIENTO GOMEZ JUAN DAVID	PS3	Py3	CORRECT	no	claim	
s23	c6	21:16	SARMIENTO GOMEZ JUAN DAVID	PS3	Py3	COMPILER-ERROR	no	claim	
s22	c6	21:15	ALPALA CANACUAN SANDRA DEL ROC...	PS3	Py3	CORRECT	no	claim	
s21	c6	21:13	MATEUS OSSA ALEJANDRO ANDRES	PS3	Py3	CORRECT	no	claim	
s20	c6	21:12	CERON DAZA LAURA DANIELA	PS3	Py3	CORRECT	no	claim	
s19	c6	21:10	MELO MORA ANGELA CAMILA	PS3	Py3	CORRECT	no	claim	
s18	c6	21:10	MELO MORA ANGELA CAMILA	PS3	Py3	COMPILER-ERROR	no	claim	
s17	c6	21:09	MELO MORA ANGELA CAMILA	PS3	Py3	COMPILER-ERROR	no	claim	
s16	c6	21:07	MUNOZ DOMINGUEZ LUISA FERNANDA	PS3	Py3	CORRECT	no	claim	
s15	c6	21:06	MUNOZ DOMINGUEZ LUISA FERNANDA	PS3	Py3	COMPILER-ERROR	no	claim	
s14	c6	21:05	CASTAÑO QUINONES SOFIA	PS3	Py3	CORRECT	no	claim	
s7	c6	15:28	PORTILLA CORTES MARIA CRISTINA	PS3	Py3	CORRECT	no	claim	

Total correct: 9, submitted: 13, unverified: 13

Figura 30: Resumen de evaluaciones por equipos del grupo 13 para el ejercicio 1.

8.4 RESULTADOS DE UN GRUPO DE ESTUDIANTES

Se realizaron pruebas con usuarios del juez virtual donde se pedía resolver un ejercicio y subir su solución. Esta prueba se realizó con el objetivo de seleccionar qué datos almacenados en la base de datos del juez usar en la aplicación. Para este proyecto solo se usaran las tablas que indican si la solución del estudiante es correcta o no para realizar la evaluación de la competencia asociada al ejercicio y se almacena la base de datos del juez en un volumen externo al contenedor para usar en trabajos futuros.

8.4 RESULTADOS DE UN GRUPO DE ESTUDIANTES

A continuación se presentan los resultados de la evaluación de los estudiantes que presentaron los ejercicios. La evaluación para el ejercicio 1 corresponde a la tabla 7 y para el ejercicio 2 la tabla 8. Posteriormente, se muestran estadísticas sobre la cantidad de personas evaluadas.

Objetivo Terminal		Dar la solución a un problema mediante un programa que implemente estructuras condicionales (if/else) en el lenguaje de programación Python.
Actividad		Realizar la entrega del ejercicio propuesto mediante el juez virtual.
Peso		100 %
Código	Nombre	
201745572	CASTAÑO QUIÑONES SOFÍA	5.0
201748863	CERON DAZA LAURA DANIELA	5.0
201422911	MATEUS OSSA ALEJANDRO ANDRES	5.0
201740111	ALPALA CANACUAN SANDRA DEL ROCIO	5.0
201744935	QUENGUAN LOAIZA JUAN PAULO	5.0
201742175	MUÑOZ DOMINGUEZ LUISA FERNANDA	5.0
201741067	SARMIENTO GOMEZ JUAN DAVID	5.0
201740047	MELO MORA ANGELA CAMILA	5.0

Cuadro 7: Evaluación para el ejercicio 1.

8.4 RESULTADOS DE UN GRUPO DE ESTUDIANTES

Objetivo Terminal		Dar la solución a un problema mediante un programa que implemente estructuras iterativas (for) en el lenguaje de programación Python.
Actividad		Realizar la entrega del ejercicio propuesto mediante el juez virtual.
Peso		100 %
Código	Nombre	
201745270	MARTINEZ MONCADA MARIA CAMILLA	5.0
201741331	ARANGO AGUIRRE ANGIE LORENA	5.0
201743754	CASTILLO PATIÑO CARLOS ALFONSO	5.0
201741398	CEDEÑO AGUAYO JUAN CAMILO	5.0
201744900	PAJOY ROJAS JOHAN DAVID	5.0
201746628	OSPINA RENDON CHRISTIAN ALEXANDER	5.0
201742032	GARCIA CHARRY JUAN DAVID	5.0
201743076	DIAZ OROBIO DANNY FERNANDO	5.0
201825102	PERDOMO PORTILLA JOHN JAIRO	5.0
201742937	POSSU VALENCIA MIGUEL ANGEL	5.0
201745692	VILLA RODRIGUEZ JULIANA	5.0
201743063	ECHEVERRI SABOGAL JUAN ESTEBAN	5.0
201741602	SUAREZ URREA VALENTINA	5.0
201743611	OSPINA BEJARANO DANIEL FERNANDO	5.0

Cuadro 8: Evaluación para el ejercicio 2.

8.4 RESULTADOS DE UN GRUPO DE ESTUDIANTES

A continuación se muestran los porcentajes obtenidos en la recolección de datos.

En las figuras 31 y 32 se resume el numero de intentos por cada ejercicio, esto es el número de veces que el estudiante subió la solución y esta fue evaluada inmediatamente por el juez.

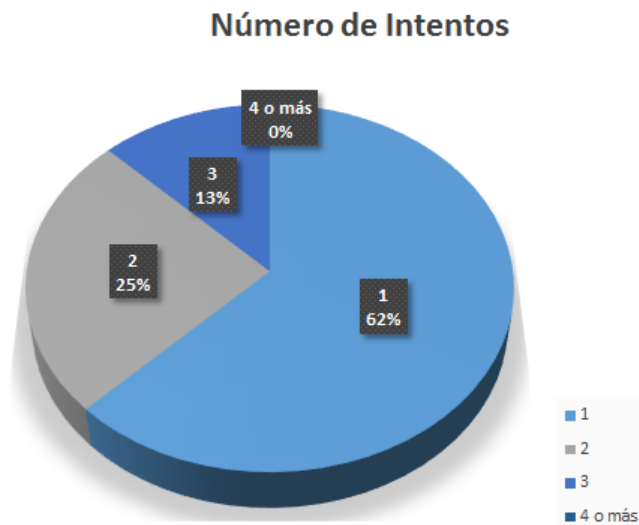


Figura 31: Número de intentos en el ejercicio 1.

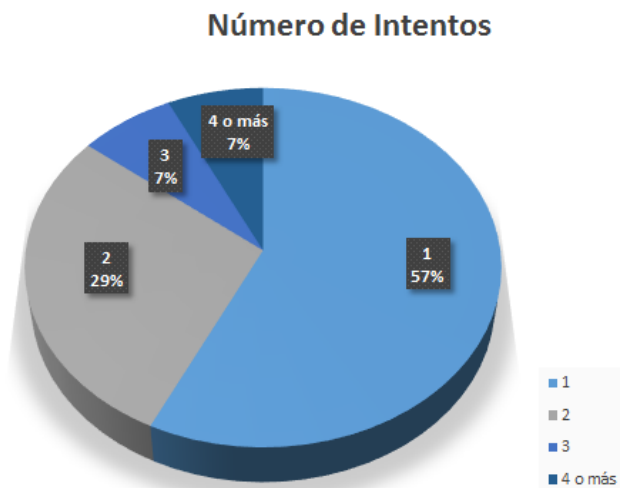


Figura 32: Número de intentos en el ejercicio 2.

En las gráficas 33 y 34 se muestran las personas que decidieron presentar el ejercicio por cada grupo. El ejercicio 1 lo realizó el grupo 13 con 26 personas matriculadas, y el ejercicio 2 lo realizó el grupo 10 con 34 estudiantes.

Personas que subieron el ejercicio

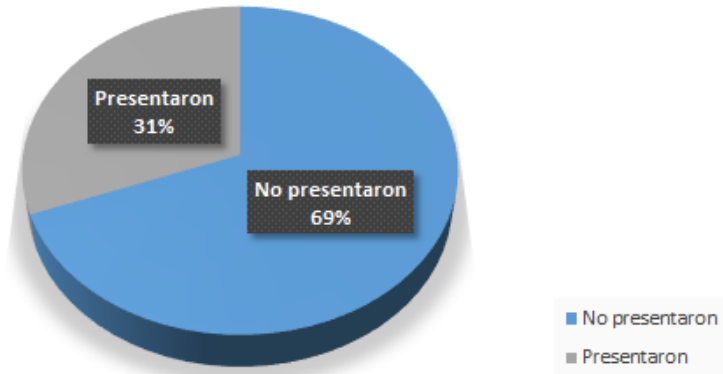


Figura 33: Porcentaje de soluciones subidas para el ejercicio 1.

Personas que subieron el ejercicio

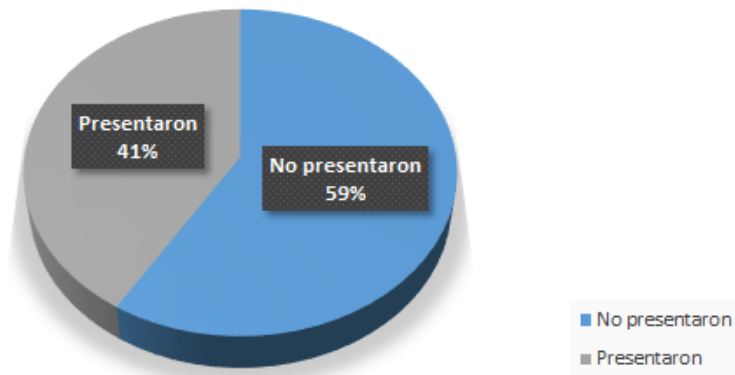


Figura 34: Porcentaje de soluciones subidas para el ejercicio 2.

Finalmente, en la gráfica 35 se comparan las personas que presentaron el ejercicio 1 y el ejercicio 2 contra el total de las personas en los dos grupos.

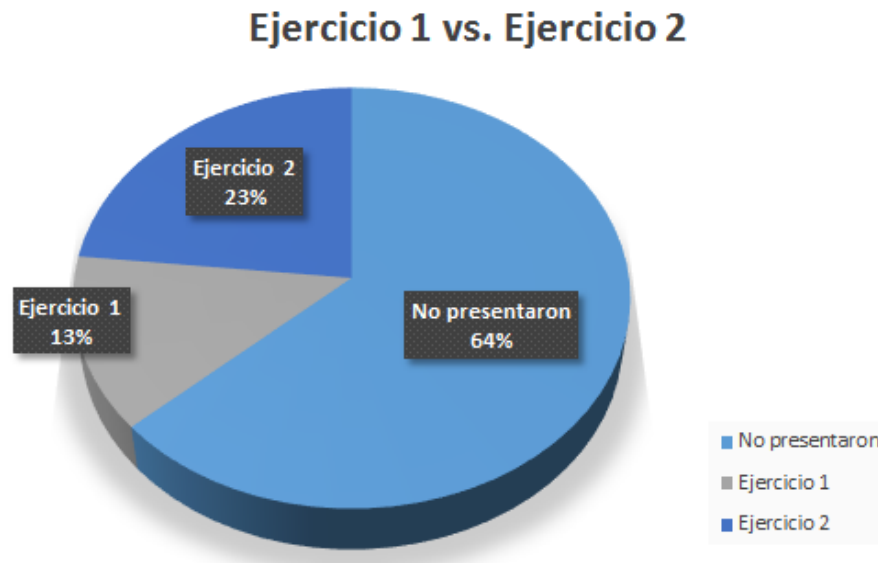


Figura 35: Porcentaje de soluciones presentadas para el ejercicio 1 vs. ejercicio 2.

En conclusión, el principal problema de la primera interacción de los estudiantes con el juez virtual fue el uso de las plantillas para subir la solución, ya que al no seguir cuidadosamente las instrucciones de entradas y salidas para el programa, el juez evaluaba la solución como errónea. Es importante resaltar que para la recolección de estos datos, los estudiantes evaluados nunca habían interactuado con el juez virtual y tampoco se continuó usando la herramienta posteriormente para el apoyo al aprendizaje de estos estudiantes.

Para los datos recolectados en el grupo 13 hubo una mayor cantidad de soluciones subidas, ya que estos fueron recolectados durante la clase en una sala de sistemas y contaban con el acompañamiento del docente y un monitor encargado de solucionar las dudas de los estudiantes para poder subir los ejercicios al juez. Además, el ejercicio era voluntario y sin evaluación, lo que hacía sentir a los estudiantes menos presionados con respecto a un impacto negativo en su nota en caso de que la evaluación del juez no sea satisfactoria (teniendo en cuenta que el juez retorna en la evaluación correcto o incorrecto lo que sería traducido a una nota de 1.0 o 5.0).

CONCLUSIONES

En el presente trabajo se describe la implementación de una plataforma para el apoyo a la evaluación de estudiantes en cursos de programación.

La plataforma desarrollada sigue un diseño e implementación orientada a servicios. Al implementar una arquitectura orientada a servicios, estos son implementados de manera independiente lo que da beneficios como:

- Escalabilidad, poder agregar nuevas funcionalidades.
- Encapsulamiento. Los servicios se desarrollan de modo que estos ofrecen una interfaz que permite el acceso a su funcionalidad. Los usuarios se comunican a través de dicha interfaz y los cambios al interior del servicio son totalmente transparentes al usuario o consumidor del servicio.
- Implementación con diferentes tecnologías.
- Despliegue y Desarrollo independientemente.

Para el desarrollo de este trabajo se incorporó el uso de Docker, una tecnología de virtualización basada en contenedores que permite materializar las características de las arquitecturas orientadas a servicios, tales como escalabilidad, encapsulamiento e integración de tecnologías heterogéneas, entre otros.

El apoyo durante el proceso de aprendizaje de los estudiantes mediante el uso de jueces virtuales en la educación provee realimentación inmediata. Sin embargo, tiene limitaciones en la forma de evaluar las soluciones ya que cuando la solución no es correcta el juez no indica si fue un error de compilación o de sintaxis.

En la primera interacción de los estudiantes con el juez virtual Domjudge se encontraron varias dificultades:

- **El uso de la interfaz de la herramienta:** La interfaz del juez no era lo suficientemente intuitiva y a pesar de la explicación de su uso era importante reiterarles a los estudiantes los pasos para subir sus soluciones.
- **El uso de plantillas para implementar la solución:** La mayoría de intentos y errores en las soluciones evaluadas por el juez fueron debido a que los estudiantes modificaron la plantilla entregada para implementar la solución o a la adición de instrucciones a esta.
- **El uso de editores de texto no apropiados para editar la plantilla:** Al usar un editor de texto no especializado para escribir código, la herramienta usada insertaba caracteres y

espacios que a pesar que la plantilla y la solución eran correctas, la evaluación del juez retornaba error de compilación.

Sin embargo, el apoyo de la herramienta será productivo con su uso continuo, durante todo un semestre. De esta manera los estudiantes aprenderán a usarla correctamente y podrán hacer seguimiento a su aprendizaje.

TRABAJO FUTURO

- Para la función de inscripción de estudiantes a los cursos, realizar la verificación con registro académico para validar si el estudiante está matriculado en el curso, de esta manera evitar que personas ajenas puedan acceder a las actividades que se realizan dentro del juez.
- Indicar errores en el código y realizar sugerencias. La manera en que DOMjudge evalúa una solución es mediante la ejecución del código que el estudiante proporciona. Si este código no tiene errores de sintaxis o ejecución y además retorna las respuestas esperadas la solución es correcta, de lo contrario el juez la puntúa como incorrecta. Este inconveniente para un estudiante principiante no es de mucha ayuda a su aprendizaje ya que es necesario que el estudiante entienda porqué su código no ejecuta o no da la respuesta correcta.
- Integrar una herramienta que indique si el código que el estudiante ha subido puede afectar la integridad de la aplicación.
- Integrar una herramienta de manejo de recursos como memoria y uso de CPU con el objetivo de establecer un límite de uso de recursos cuando el ejercicio así lo requiera.
- Analizar los datos recolectados sobre los cursos con el objetivo de fortalecer los temas en donde los estudiantes tienen más inconvenientes.
- Realizar una función para almacenar la rúbrica relacionada con el(los) problema(s) propuesto y pueda retornar la nota general del taller o examen.
- Aunque la aplicación implementa una arquitectura orientada a servicios, es posible atomizar las funcionalidades con el objetivo de crear microservicios, de esta manera facilitar la integración de futuras funcionalidades.
- Realizar más imágenes bases para los cursos que usen un lenguaje distinto a python o scilab o con otro juez virtual.
- Realizar repositorios o copias de seguridad de los contenedores que se están ejecutando actualmente, ya que si en algún momento el servidor donde estos están alojados falla se pierde la información de los cursos.
- Para apoyar el proceso de aprendizaje con la herramienta DOMjudge, es importante que los docentes tengan un conjunto de ejercicios no calificables además de los que pertenecen a los talleres para que los estudiantes puedan afianzar sus habilidades.

BIBLIOGRAFÍA

- [1] 5 types of big data to extract from your lms and how to use it. <https://elearningindustry.com/types-big-data-extract-lms-how-use>, 2017. Accessed: 2017-12-09.
- [2] Boca online contest administrator. <https://www.ime.usp.br/~cassio/boca/>, 2017. Accessed: 2017-12-10.
- [3] Cadvisor. <https://github.com/google/cadvisor>, 2017. Accessed: 2017-12-14.
- [4] Canonical. <https://www.canonical.com/>, 2017. Accessed: 2017-12-09.
- [5] Learn on code studio. <https://studio.code.org/courses/>, 2017. Accessed: 2017-10-09.
- [6] Data dog. <https://www.datadoghq.com/>, 2017. Accessed: 2017-12-14.
- [7] Docker scout. <https://github.com/pingdomserver/docker-scout>, 2017. Accessed: 2017-12-14.
- [8] Docker stats. <https://docs.docker.com/engine/reference/commandline/stats/>, 2017. Accessed: 2017-12-14.
- [9] Expolearning. <https://www.expolearning.com/tendencias-e-learning-2017/>, 2017. Accessed: 2017-12-09.
- [10] GraphQL. <http://graphql.org/learn/>, 2017. Accessed: 2017-18-09.
- [11] GraphQL explained. <https://dev-blog.apollodata.com/graphql-explained-5844742f195e>, 2017. Accessed: 2017-23-10.
- [12] GraphQL vs. rest. <https://dev-blog.apollodata.com/graphql-vs-rest-5d425123e34b>, 2017. Accessed: 2017-23-10.
- [13] Remote procedure call. https://www.ibm.com/support/knowledgecenter/en/ssw_aix_71/com.ibm.aix.progcom/ch8_rpc.htm, 2017. Accessed: 2017-10-05.
- [14] Bringing graphql to ios. <https://dev-blog.apollodata.com/bringing-graphql-to-ios-fc46423befa1>, 2017. Accessed: 2017-23-10.
- [15] Introduction to container monitoring. <https://www.brianchristner.io/introduction-to-container-monitoring/>, 2017. Accessed: 2017-10-06.
- [16] Kubernetes. <https://kubernetes.io/>, 2017. Accessed: 2017-12-09.

- [17] Linux containers. <https://linuxcontainers.org/lxd/>, 2017. Accessed: 2017-12-09.
- [18] Orquestación de servicios: Bpel. <http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion03-apuntes.html>, 2017. Accessed: 2017-11-09.
- [19] Programming contest control system. <http://pc2.ecs.csus.edu/>, 2017. Accessed: 2017-12-10.
- [20] ¿por qué deberíamos abandonar rest y empezar a usar graphql en nuestras apis? <https://www.genbetadev.com/desarrollo-aplicaciones-moviles/por-que-deberiamos-abandonar-rest-y-empezar-a-usar-graphql-en-nuestras-apis>, 2017. Accessed: 2017-23-10.
- [21] Rabbitmq. <http://www.rabbitmq.com/>, 2017. Accessed: 2017-12-10.
- [22] Comparing 10 container monitoring solutions for rancher. <http://rancher.com/comparing-10-container-monitoring-solutions-rancher/>, 2017. Accessed: 2017-12-14.
- [23] Comparing 7 monitoring options for docker. <http://rancher.com/comparing-monitoring-options-for-docker-deployments/>, 2017. Accessed: 2017-12-14.
- [24] Rest architectural elements and constraints. <http://mrbool.com/rest-architectural-elements-and-constraints/29339>, 2017. Accessed: 2017-10-03.
- [25] Sensu. <https://sensuapp.org/>, 2017. Accessed: 2017-12-14.
- [26] The docker monitoring problem. <https://https://www.datadoghq.com/blog/the-docker-monitoring-problem/>, 2017. Accessed: 2017-10-06.
- [27] The latest trends in the industry in 2017. <https://apiumhub.com/tech-blog-barcelona/tech-trends-software/>, 2017. Accessed: 2017-22-10.
- [28] Travis ci. <https://travis-ci.org/>, 2017. Accessed: 2017-12-09.
- [29] Uva online judge. <https://uva.onlinejudge.org/>, 2017. Accessed: 2017-12-10.
- [30] Vernemq. <https://vernemq.com/>, 2017. Accessed: 2017-12-10.
- [31] Websphere. <https://www.ibm.com/cloud/websphere-application-platform>, 2017. Accessed: 2017-12-12.
- [32] What the future of restful api design holds for developers. <http://http://searchmicroservices.techtarget.com/feature/What-the-future-of-RESTful-API-design-holds-for-developers>, 2017. Accessed: 2017-22-10.
- [33] Zeromq. <http://zeromq.org/>, 2017. Accessed: 2017-12-10.

- [34] Apache zookeeper. <https://zookeeper.apache.org/>, 2017. Accessed: 2017-12-12.
- [35] Coreos rkt. <https://coreos.com/rkt/>, 2018. Accessed: 2018-24-03.
- [36] Docker vs coreos rkt. <https://www.upguard.com/articles/docker-vs-coreos>, 2018. Accessed: 2018-24-03.
- [37] rkt architecture. <https://coreos.com/rkt/docs/latest/devel/architecture.html>, 2018. Accessed: 2018-04-15.
- [38] The new uva online judge platform by miguel revilla rodriguez. <https://www.youtube.com/watch?v=9AhuXXbzqWY&t=50s>, 2018. Accessed: 2018-04-15.
- [39] Uva huntingr. <https://uhunt.onlinejudge.org/>, 2018. Accessed: 2018-24-03.
- [40] S. Alhazbi. *Advances in Engineering Education in the Middle East and North Africa, Part II*, chapter Active Blended Learning to Improve Students' Motivation in Computer Programming Courses: A Case Study, pages 187–204. Springer International Publishing, 2016.
- [41] R. Ali, Y. Shen, X. Huang, J. Zhang, and A. Ali. Vmr: Virtual machine replacement algorithm for qos and energy-awareness in cloud data centers. *Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC)*, 2017 IEEE International Conference on, 2017. doi: <http://ieeexplore.ieee.org/document/8006004/>.
- [42] J. Anderson, H. Hongxin, A. Udit, C. Lowery, H. Li, and A. Apon. Performance considerations of network functions virtualization using containers. *Computing, Networking and Communications (ICNC)*, 2016 International Conference on, pages 1 – 7, 2016. doi: <http://ieeexplore.ieee.org/document/7440668/>.
- [43] A. Azab and D. Domanska. Software provisioning inside a secure environment as docker containers using stroll file-system. *Cluster, Cloud and Grid Computing (CCGrid)*, 2016 16th IEEE/ACM International Symposium on, 2016. doi: <http://ieeexplore.ieee.org/document/7515755/>.
- [44] A. Babu, M. Hareesh, and J. Martin. System performance evaluation of para virtualization, container virtualization, and full virtualization using xen, openvz, and xenserver. *Advances in Computing and Communications (ICACC)*, 2014 Fourth International Conference on, pages 247 – 250, 2014. doi: <http://ieeexplore.ieee.org/document/6906035/>.
- [45] L. Bahri, B. Carminati, and E. Ferrari. Privacy in web service transactions: a tale of more than a decade of work. *IEEE Transactions on Services Computing*, pages 1– 18, 2017. doi: <http://ieeexplore.ieee.org/document/7937912/>.
- [46] R. Barik, R. Lenka, and K. Rao. Performance analysis of virtual machines and containers in cloud computing. *Computing, Communication and Automation (ICCCA)*, 2016 International Conference on, pages 1204–1210, 2016. doi: <http://ieeexplore.ieee.org/document/7813925/>.

- [47] B Baruque and A. Herrero. Self-assessment web tool for java programming. *Advances in Intelligent Systems and Computing, Volume 369*, pages 583–592, 2015. doi: http://link.springer.com/chapter/10.1007/978-3-319-19713-5_51.
- [48] Fabrice Bellard. Qemu. <http://wiki.qemu.org/>, 2017. Accessed: 2017-01-09.
- [49] A. Beresford and M. Kleppmann. A conflict-free replicated json datatype. *IEEE Transactions on Parallel and Distributed Systems Issue: 99*, 2017. doi: <http://ieeexplore.ieee.org/document/7909007/>.
- [50] Y. Berkunskyi, K. Knyrik, T. Farionova, and T. Smykodub. Using microservices in educational applications of it-company . *Electrical and Computer Engineering (UKRCON), 2017 IEEE First Ukraine Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/8100443/>.
- [51] Tim Berners-LeeOasis-Open. World wide web consortium. <https://www.w3.org/>, 2017. Accessed: 2017-06-09.
- [52] J. Boixados, B. Cifuentes, and Y. Zhang. *Generación automática de Test en DomJudge*. Trabajo de grado, universidad Complutense de Madrid, http://eprints.ucm.es/38397/1/TrabajoFin_de_Grado.pdf, 2017.
- [53] Don Box. Xml web services one conference. 2003.
- [54] A. Bozzon, P. Cudre-Mauroux, and C. Pautasso. *Web Engineering, ICWE 2016*, chapter REST APIs: A Large-Scale Analysis of Compliance with Principles and Best Practices, pages 21–39. Springer, 2016.
- [55] Dean Sanders Brian Dorne. Jeroo. <http://home.cc.gatech.edu/dorn/jeroo>, 2017. Accessed: 2017-01-06.
- [56] A. Brinckman, D. Luc, J. Nabrzyski, J. Neidig, T. Puckett, S. Radha, and I. Taylor. A comparative evaluation of blockchain systems for application sharing using containers. *e-Science (e-Science), 2017 IEEE 13th International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/8109185/>.
- [57] A. Celesti, L. Carnevale, A. Galletta, M. Fazio, and M. Villari. A watchdog service making container-based micro-services reliable in iot clouds. *Future Internet of Things and Cloud (FiCloud), 2017 IEEE 5th International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/8114506/>.
- [58] O. Chamat. *Desarrollo de una aplicación web para la automatización de entrega y corrección de trabajos de programación*. Universidad del Valle, 2014.
- [59] J. Chelladhurai, P. Chelliah, and S. Kumar. Securing docker containers from denial of service (dos) attacks. *Services Computing (SCC), 2016 IEEE International Conference on*, pages 856 – 859, 2016. doi: <http://ieeexplore.ieee.org/document/7557545/>.

- [60] Oracle Corporation. Virtual box. <https://www.virtualbox.org/>, 2017. Accessed: 2017-01-09.
- [61] J. Craton. The effect of videogames on student achievement. 2011. doi: <http://www.acsd.org/article/the-effect-of-videogames-on-student-achievement/>.
- [62] INGI department. Inginous. <http://www.inginous.org/>, 2017. Accessed: 2017-01-17.
- [63] S. Dhakate and A. Godbole. Distributed cloud monitoring using docker as next generation container virtualization technology. *India Conference (INDICON), 2015 Annual IEEE*, 2015. doi: <http://ieeexplore.ieee.org/document/7443771/>.
- [64] Edouard Bugnion Scott Devine Edward Wang Diane Greene, Mendel Rosenblum. Vmware workstation. www.vmware.com/co/products/workstation.html, 2017. Accessed: 2017-01-09.
- [65] Y. Dong, X. Zhang, and J Dai. Hyvi: A hybrid virtualization solution balancing performance and manageability. *IEEE Transactions on Parallel and Distributed Systems (Volume: 25, Issue: 9, Sept. 2014)*, pages 2332 – 2341, 2014. doi: <http://ieeexplore.ieee.org/document/6605688/>.
- [66] R. Dua, R. Raja, and D. Kakadia. Virtualization vs containerization to support paas. *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 610 – 614, 2014. doi: <http://ieeexplore.ieee.org/document/6903537/>.
- [67] C. Fan and S. Ma. Migrating monolithic mobile application to microservice architecture: An experiment report. *AI Mobile Services (AIMS), 2017 IEEE International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/8027278/>.
- [68] X. Fang, H. Zhang, and Y. Sun. *Lecture Notes in Electrical Engineering (LNEE), Volume 269*, chapter A Programming Related Courses' E-learning Platform Based on Online Judge, pages 3419–3423. Springer, 2014.
- [69] P. Farzi, R. Akbari, and O Bushehrian. Improving semantic web service discovery method based on qos ontology. *IEEE Conference Publications*, pages 72 – 76, 2017. doi: <http://ieeexplore.ieee.org/document/7940175/>.
- [70] Roy Fielding. Representational state transfer. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, 2017. Accessed: 2017-10-03.
- [71] A. Gerdes, B. Heeren, J. Jeuring, and T. Van Binsbergen. Ask-elle: an adaptable programming tutor for haskell giving automated feedback. *International Journal of Artificial Intelligence in Education*, page 1–36, 2016. doi: <http://link.springer.com/article/10.1007/s40593-015-0080-x>.
- [72] M. Gomes, M. Veiga, and C. FonticIELha De Rose. A performance comparison of container-based virtualization systems for mapreduce clusters. *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 299 – 306, 2014. doi: <http://ieeexplore.ieee.org/document/6787290/>.
- [73] M. Grossmann and C. Klug. Monitoring container services at the network edge. *Teletraffic Congress (ITC 29), 2017 29th International*, 2017. doi: <http://ieeexplore.ieee.org/document/8064348/>.

- [74] M. Guerrero, S. Guaman, and J. Caiza. Revisión de herramientas de apoyo en el proceso de enseñanza-aprendizaje de programación. *Revista Politécnica, Volumen: 35, No. 1*, pages 82–90, 2015.
- [75] A. Guner, K. Kurtel, and U. Celikkan. A message broker based architecture for context aware iot application development. *Computer Science and Engineering (UBMK), 2017 International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/8093381/>.
- [76] C. Gómez, H. Castro, and C. Varela. Global snapshot of a distributed system running on virtual machines. *Computer Architecture and High Performance Computing (SBAC-PAD), 2017 29th International Symposium on*, 2017. doi: <http://ieeexplore.ieee.org/document/8102192/>.
- [77] T. Hainey and T. Connolly. *Advances in Intelligent Systems and Computing, Volume 239*, chapter The Use of Games as an Educational Web2.0 Technology, pages 719–728. Springer, 2014.
- [78] S. Hassan, N. Ali, and R. Bahsoon. Microservice ambients: An architectural meta-modelling approach for microservice granularity. *Software Architecture (ICSA), 2017 IEEE International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/7930193/>.
- [79] F. Haupt, F. Leymann, A. Scherer, and K. Vukojevic-Haupt. A framework for the structural analysis of rest apis. *Software Architecture (ICSA), 2017 IEEE International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/7930199/>.
- [80] L. Hitch and J. Duncan. Games in higher ed: When halo 2, civilization iv, and xbox 360 come to campus. *Educause Evolving Technologies Commitee.*, 2005. doi: <https://www.educause.edu/ir/library/pdf/DECo503.pdf>.
- [81] Solomon Hykes. Docker. <https://www.docker.com/>, 2017. Accessed: 2017-01-17.
- [82] Solomon Hykes. Docker hub. <https://hub.docker.com/>, 2017. Accessed: 2017-01-18.
- [83] IBM. Acm icpc programming contests. <https://icpc.baylor.edu/>, 2017. Accessed: 2017-01-19.
- [84] IBM. Relación entre uddi y wsdl. https://www.ibm.com/support/knowledgecenter/es/SSRTLW_9.6.0/org.eclipse.jst.ws.consumption.ui.doc.user/concepts/cwsdlud.html, 2017. Accessed: 2017-06-10.
- [85] V. Ivancevic, M. Knezevic, B. Pusic, and I. Lukovic. Adaptive testing in programming courses based on educational data mining techniques. *Educational Data Mining, Volume 524*, pages 257–287, 2014. doi: http://link.springer.com.bd.univalle.edu.co/chapter/10.1007/978-3-319-02738-8_10.
- [86] Keith Johnson Thijs Kinkhorst Tobias Werth Jaap Eldering, Nicky Gerritsen. Domjudge. <https://www.domjudge.org/>, 2017. Accessed: 2017-01-18.
- [87] D. Jaramillo, D. Nguyen, and R. Smart. Leveraging microservices architecture by using docker technology. *SoutheastCon, 2016, Conference*, 2016. doi: <http://ieeexplore.ieee.org/document/7506647/o>.

- [88] A. Jesudoss and N. Subramaniam. Attack-tolerant security model for ws-security. *Circuit, Power and Computing Technologies (ICCPCT), 2014 International Conference on*, 2014. doi: <http://ieeexplore.ieee.org/document/7054907/>.
- [89] H. Kawazoe, D. Ajitomi, and K. Minami. A test framework for large-scale message broker system for consumer devices. *Consumer Electronics - Berlin (ICCE-Berlin), 2015 IEEE 5th International Conference on*, 2015. doi: <http://ieeexplore.ieee.org/document/7391245/>.
- [90] A. Khaled and J. Miller. Using pi-calculus for formal modeling and verification of ws-cdl choreographies. *IEEE Transactions on Services Computing, Volume 10, Issue 2, year =*.
- [91] University of Kent Kings' College London. Bluej. <http://www.bluej.org/>, 2017. Accessed: 2017-01-06.
- [92] A. Krylovskiy. Internet of things gateways meet linux containers: Performance evaluation and discussion. *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, 2016. doi: <http://ieeexplore.ieee.org/document/7389056/>.
- [93] L. Li and M. Lou. Connection minimization in rest api with random walks. *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*, 2016. doi: <http://ieeexplore.ieee.org/document/7817076/>.
- [94] D. Linthicum. Moving to autonomous and self-migrating containers for cloud applications. *IEEE Cloud Computing, Volume: 3, Issue: 6*, pages 6–9, 2016. doi: <http://ieeexplore.ieee.org/document/7802520/>.
- [95] D. Lu, D. Huang, A. Walenstein, and D. Medhi. A secure microservice framework for iot. *Service-Oriented System Engineering (SOSE), 2017 IEEE Symposium on*, 2017. doi: <http://ieeexplore.ieee.org/document/7943286/>.
- [96] S. Lu, J. Pang, and Y Pan. A mpi-friendly efficient static binary translator based on qemu. *Computer Science and Mechanical Automation (CSMA), 2015 International Conference on*, pages 234 – 239, 2015. doi: <http://ieeexplore.ieee.org/document/7371618/>.
- [97] F. Luo, I. Scherson, and J. Fuentes. Dynamic creation of virtual machines in cloud computing systems. *Systems Engineering (ICSEng), 2017 25th International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/8121689/>.
- [98] J. Luzuriaga, M. Perez, P. Boronat, C. Cano, C. Calafate, and P. Manzoni. A comparative evaluation of amqp and mqtt protocols over unstable and mobile networks. *Intelligent Systems Design and Engineering Applications (ISDEA), 2015 Fifth International Conference on*, pages 931 – 936, 2015. doi: <http://ieeexplore.ieee.org/document/7158101/>.
- [99] S. Mandalika and H. Mittal. Framework for evaluation of programming language examinations. *Computational Intelligence Techniques for Comparative Genomics*, pages 121–128, 2014. doi: http://link.springer.com/chapter/10.1007/978-981-287-338-5_10.
- [100] M. Masse. *REST API design rulebook*. O'Reilly Media, 2011.

- [101] A. Misbah and A. Ettalbi. Towards a standard wsdl implementation of multiview web services. *Multimedia Computing and Systems (ICMCS), 2016 5th International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/7905542/>.
- [102] T. Mishra and G. Raj. Qos implementation in web services selection and ranking using data analysis. *IEEE Conference Publications*, pages 537 – 542, 2017. doi: <http://ieeexplore.ieee.org/document/7943209/>.
- [103] F. Moradi, C. Flinta, A. Johnsson, and C. Meirosu. On time-stamp accuracy of passive monitoring in a container execution environment. *IFIP Networking Conference (IFIP Networking) and Workshops, 2016*, 2016. doi: <http://ieeexplore.ieee.org/document/7497236/>.
- [104] F. Moradi, C. Flinta, A. Johnsson, and C. Meirosu. Conmon: An automated container based network performance monitoring system. *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, 2016. doi: <http://ieeexplore.ieee.org/document/7987264/>.
- [105] S. Nathan, R. Ghosh, T. Mukherjee, and K. Narayanan. Comicon: A co-operative management system for docker container images. *Cloud Engineering (IC2E), 2017 IEEE International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/7923794/>.
- [106] M. Noorafiza, H. Maeda, and R. Uda. Vulnerability analysis using network timestamps in full virtualization virtual machine. *Information Systems Security and Privacy (ICISSP), 2015 International Conference on*, pages 83–89, 2015.
- [107] A. Nuraini, W. Lau, S. Ooi, H. Tan, Z. Zurinahni, and O. Majid. A framework of an automatic assessment system for learning programming. *Advanced Computer and Communication Engineering Technology, Volume 362*, pages 967–977, 2015. doi: http://link.springer.com/chapter/10.1007/978-3-319-24584-3_82.
- [108] Oasis-Open. Oasis. <https://www.oasis-open.org/>, 2017. Accessed: 2017-06-09.
- [109] Web Services Interoperability Organization. Ws-i. <http://www.ws-i.org/>, 2017. Accessed: 2017-06-09.
- [110] N Partheeban and N. SankarRam. e-learning management system using web services. *Information Communication and Embedded Systems (ICICES), 2014 International Conference on*, pages 1–7, 2014. doi: <http://ieeexplore.ieee.org/10.1109/ICICES.2014.7033900>.
- [111] C. Perez-Leguizamo and G. Godinez-Borja. Autonomous decentralized service oriented architecture: Concept, technologies and application. *2017 IEEE 13th International Symposium on Autonomous Decentralized System*, pages 47 – 54, 2017. doi: <http://ieeexplore.ieee.org/document/7940218/>.
- [112] V. Pham and E. Huh. A fog/cloud based data delivery model for publish-subscribe systems. *Information Networking (ICOIN), 2017 International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/7899539/>.

- [113] W. Pipatsakulroj, V. Visoottiviseth, and R. Takano. mumq: A lightweight and scalable mqtt broker. *Local and Metropolitan Area Networks (LANMAN), 2017 IEEE International Symposium on*, 2017. doi: <http://ieeexplore.ieee.org/document/7972165/>.
- [114] Herbert Potzl. Linux vserver. <http://www.linux-vserver.org/Paper>, 2017. Accessed: 2017-01-09.
- [115] soportado por SWsoft Proyecto comunitario. Open virtuozzo. <https://openvz.org/>, 2017. Accessed: 2017-01-09.
- [116] Z. Qin, Y. Tang, and F. Tang. Efficient xml query and update processing using a novel prime-based middle fraction labeling scheme. *China Communications Vol: 14, Issue: 3*, pages 145 – 157, 2017. doi: <http://ieeexplore.ieee.org/document/7897330/>.
- [117] S. Rebai, H. Kacem, M. Karaa, and S. Pomares. Cdlvt: A formal verification tool of non-functional properties for ws-cdl specification. *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2015 IEEE 24th International Conference on*, 2015. doi: <http://ieeexplore.ieee.org/document/7194358/>.
- [118] P. Reddy and L. Rajamani. Performance evaluation of operating systems in the private cloud with xenserver hypervisor using sigar framework. *Computer Science Education (ICCSE), 2014 9th International Conference on*, pages 183 – 188, 2014. doi: <http://ieeexplore.ieee.org/document/6926451/>.
- [119] R. Rizki, A. Rakhmatsyah, and M. Nugroho. Performance analysis of container-based hadoop cluster: Openvz and lxc. *Information and Communication Technology (ICoICT), 2016 4th International Conference on*, 2016. doi: <http://ieeexplore.ieee.org/document/7571957/>.
- [120] V. Sarkale, P. Rad, and W. Lee. Secure cloud container: Runtime behavior monitoring using most privileged container (mpc). *Cyber Security and Cloud Computing (CSCloud), 2017 IEEE 4th International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/7987222/>.
- [121] M. Satratzemi, S. Xinogalos, and V. Dagdilelis. An environment for teaching object-oriented programming: objectkarel. *Advanced Learning Technologies, 2003. Proceedings. The 3rd IEEE International Conference on*, 2003. doi: <http://ieeexplore.ieee.org/document/1215114/>.
- [122] D. Serrano, E. Stroulia, D. Lau, and T. Ng. Linked rest apis: A middleware for semantic rest api integration. *Web Services (ICWS), 2017 IEEE International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/8029755/>.
- [123] M. Shah and D. Kulkarni. Enabling qos support for multi-core message broker in publish/-subscribe system. *Advance Computing Conference (IACC), 2014 IEEE International*, 2014. doi: <http://ieeexplore.ieee.org/document/6779421/>.
- [124] A. Singh and Y. Viniotis. Resource allocation for iot applications in cloud environments. *Computing, Networking and Communications (ICNC), 2017 International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/7972165/>.

- [125] S. Sohan, C. Anslow, and F. Maurer. Automated example oriented rest api documentation at cisco. *Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), 2017 IEEE/ACM 39th International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/7965445/>.
- [126] D. Souza, K. Felizardo, and E. Barbosa. A systematic literature review of assessment tools for programming assignments. *Software Engineering Education and Training (CSEET), 2016 IEEE 29th International Conference on*, pages 147 – 156, 2016. doi: <http://ieeexplore.ieee.org/document/7474479/>.
- [127] A. Terkhedkar and M. Shah. Providing security mechanisms in broker-less publish/subscribe systems using hierarchical identity based encryption. *Recent Trends in Electronics, Information Communication Technology (RTEICT), IEEE International Conference on*, 2016. doi: <http://ieeexplore.ieee.org/document/7807901/>.
- [128] U. Thakar, A. Tiwari, and S. Varma. On composition of soap based and restful services. *Advanced Computing (IACC), 2016 IEEE 6th International Conference on*, 2016. doi: <http://ieeexplore.ieee.org/document/7544888/>.
- [129] Inc. The Xen Project. XenSource. Xen. <https://www.xenproject.org/>, 2017. Accessed: 2017-01-09.
- [130] P. Vasconcelos and G. de Araujo Freitas. Performance analysis of hadoop mapreduce on an opennebula cloud with kvm and openvz virtualizations. *Internet Technology and Secured Transactions (ICITST), 2014 9th International Conference for*, pages 471 – 476, 2014. doi: <http://ieeexplore.ieee.org/document/7038858/>.
- [131] Google Eric Biederman Virtuozzo, IBM. Linux containers. <https://linuxcontainers.org/>, 2017. Accessed: 2017-01-09.
- [132] L. Vladimir, J. Uskov, and J. Howlett. Tests generation oriented web-based automatic assessment of programming assignments. *Smart Innovation, Systems and Technologies, Volume 41*, pages 117–127, 2015. doi: http://link.springer.com/chapter/10.1007/978-3-319-19875-0_11.
- [133] L. Vladimir, J. Uskov, and J. Howlett. Tests generation oriented web-based automatic assessment of programming assignments. *Smart Innovation, Systems and Technologies, Volume 41*, pages 117–127, 2015. doi: http://link.springer.com/chapter/10.1007/978-3-319-19875-0_11.
- [134] D. Wassim and F. Fethi. Ndn-q: An ndn query mechanism for efficient v2x data collection. *Sensing, Communication, and Networking Workshops (SECON Workshops), 2014 Eleventh Annual IEEE International Conference on*, 2014. doi: <http://ieeexplore.ieee.org/document/6979698/>.
- [135] University of Amsterdam Weizmann Institute of Science. Jeliot. <https://cs.joensuu.fi/jeliot/index.php>, 2017. Accessed: 2017-01-06.
- [136] N. WildeBilal, G. El-Sheikh, and A. Zimmermann. *Emerging Trends in the Evolution of Service-Oriented and Enterprise Architectures*, chapter Approaches to the Evolution of SOA Systems, pages 5–22. Springer, 2016.

- [137] J. Wu, Ch. Wang, and J. Li. La-credit: A load-awareness scheduling algorithm for xen virtualized platforms. *Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2016 IEEE 2nd International Conference on*, 2016. doi: <http://ieeexplore.ieee.org/document/7502295/>.
- [138] Li Xiao-Hong. Research and development of web of things system based on rest architecture. *Intelligent Systems Design and Engineering Applications (ISDEA), 2014 Fifth International Conference on*, pages 1 – 7, 2014. doi: <http://ieeexplore.ieee.org/document/6977704/>.
- [139] S. Xinogalos. Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy. *Education and Information Technologies, Springer US*, 2014. doi: <http://link.springer.com/article/0.1007/s10639-014-9341-9>.
- [140] L. Xu, Q. Sun, B. Xu, and W. Zhang. Statically detect data races for ws-bpel web services by constraint solver. *Web Services (ICWS), 2016 IEEE International Conference on*, 2016. doi: <http://ieeexplore.ieee.org/document/7558037/>.
- [141] P. Xu, S. Shi, and X. Chu. Performance evaluation of deep learning tools in docker containers. *Big Data Computing and Communications (BIGCOM), 2017 3rd International Conference on*, 2017. doi: <http://ieeexplore.ieee.org/document/8113094/>.
- [142] X. Xuandong and Fu. Jiandan. Active status certificate publish and subscribe based on amqp. *Computational and Information Sciences (ICCIS), 2011 International Conference on*, pages 725 – 728, 2011. doi: <http://ieeexplore.ieee.org/document/6086301/>.
- [143] X. Yu and X. Wu, J. Yin. Study on conformance testing of hypertext transfer protocol. *Communication Technology Proceedings, 2003. ICCT 2003. International Conference on*, 2003. doi: <http://ieeexplore.ieee.org/document/1209063/>.
- [144] J. Zhang, L. Xiaoyi, and K. Dhabaleswar. Performance characterization of hypervisor- and container-based virtualization for hpc on sr-ioV enabled infiniband clusters. *Parallel and Distributed Processing Symposium Workshops — IEEE International*, 2016, pages 1777 – 1784, 2016. doi: <http://ieeexplore.ieee.org/document/7530082/>.
- [145] Y. Zhang, K. Xu, H. Wang, Q. Li, T. Li, and X. Cao. Going fast and fair: Latency optimization for cloud-based service chains. *IEEE Network, Volume: PP, Issue: 99*, 2017. doi: <http://ieeexplore.ieee.org/document/8123821/>.
- [146] W. Zhaodong and W. Wenjun. A peer grading tool for moocs on programming. *Intelligent Computation in Big Data Era, Volume 503*, pages 378–385, 2015. doi: http://link.springer.com/chapter/10.1007/978-3-662-46248-5_46.