UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

TESIS DE INGENERÍA

GitEDU y EduNube: Un Doble Sistema de Integración de LTI, Git y Virtualización

---

Autor:
Nicholas SPALDING
EARLEY-DOLENC

Supervisor:
Dr. Jorge LÓPEZ

Tesis entregado para cumplir con los requisitos del titulo de Ingenería en Sistemas Informáticos y Computación

en el

Laboratorio de Tecnologías Web
Escuela de Ciencias de la Computación

16 de febrero de 2018

Declaracion of Autoria

Yo, Nicholas SPALDING EARLEY-DOLENC, declaro que esta tesis titulada, «GitEDU y EduNube:
Un Doble Sistema de Integración de LTI, Git y Virtualización» y que el trabajo presentado aquí es p. Yo afirmo que:

- Este trabajo fue realizado principalmente mientras que fui candidato de un titulo de investigacion en la Universidad.
- Partes de esta tesis que se han sido utilizados por algun otro titulo o cualificacion, se lo ha señalado de una forma clara e transparente.
- Partes de esta tesis donde he consultado el trabajo publicado de terceros, se lo ha citado de la forma correcta.
- Donde he citado el trabajo de tercersos, la fuente siempre esta dada. Con la excepcion de aquellos citas, esta thesis es totalmente mi propio trabajo.
- He documentado mis principales fuentes de ayuda.
- En lugares donde esta tesis esta basada en trabajo realizado por terceros, he documentado claramente cuales partes realizaron ellos y cuales partes yo realicé.

Firmado:

Fecha:

«Elegimos… hacer las… cosas, no porque sea fácil, sino porque es difícil… Porque esta meta, servirá para organizar y probar lo mejor de nuestras energías y habilidades.»

John F. Kennedy

Universidad Técnica Particular de Loja

Resumen

Departamento de Ciencias de la Computación y Electrónica
Escuela de Ciencias de la Computación

Ingenería en Sistemas Informáticos y Computación

GitEDU y EduNube:
Un Doble Sistema de Integración de LTI, Git y Virtualización

by Nicholas SPALDING EARLEY-DOLENC

Para mejorar el entorno educativo para los estudiantes de carreras que involucran programación, es necesario el desarrollo de un sistema que integra los LMS de ahora a través del estándar LTI con Git, un editor de código en línea y un entorno para la ejeccución del mismo conlleva varias fases. En una fase inicial, es importante analizar la problemática, el entorno existente y características de sistemas similares para poder formular una solución facible y novedoso desde sus requisitos funcionales y diseño. Con un marco de diseño, se puede proceder a definir una arquiectura que asegurará que se cumple con los requisitos no funcionales. Con el entorno bien estudiado y un diseño robusto, se puede inicializar con varias fases ágiles e incrementales de desarrollo, pruebas y desplegue hasta llegar a un prototipo final que representa las funcionalidades basicas de la rafaga inicial de desarrollo.

Agradecimientos

Quiero agradecer a la Universidad Técnica Particular de Loja por darme un espacio para crecer y aprender lo justo y necesario para ser un buen profesional y persona que está al servicio de la sociedad.

Quiero agradecer a mi Director de Tesis, el Doctor Jorge López, quien, a pesar de mis limitaciones y amor para mi tesis ha logrado guiarme y controlar mi alcanze para que el reto de este trabajo de titulación sea alcanzable en el tiempo dado. Para mi siempre ha sido una gran voz de razón, me ha encaminado tanto a nivel profesional como personal. Además le quiero agradecer por brindarme la confianza y libertad para resolver los problemas planteados en mi trabajo de titulación.

Quiero agradecer al Ingeniero Rene Elizalde por darme aportarme las bases necesarias para entrar al hermoso mundo de Python y Django, que sin estos conocimientos que me dio, no hubiera tenido factibilidad técnica un proyecto de este alcance en el tiempo dado.

Quiero agradecer al Ingeniero Santiago Quiñones por motivarme a seguir estudiando, programando y por calificar de una forma muy justa el código de cada estudiante, no tanto por funcionalidad, si no, por calidad de lógica.

Quiero agradecer a mi profesor de historia mundial de colegio, Robert Peoples, quien me enseñó que lo más importante saber es el porqué de las cosas.

Quiero agradecer a mis maestros de matemáticas en especial a Ms. Pearman quien me brindó las bases necesarias para poder analizar problemas desde un punto analítico y computacional, una de los habilidades más importantes para cualquier ingeniero en sistemas.

Quiero agradecer a la Ingeniera Maria del Carmen Cabrera por su paciencia y disposición en la resolución oportuna de mis dudas y revisión de la tesis a lo largo de su desarrollo.

Quiero agradecer a la Ingeniera Pamela Guamán quien siempre ha sido una buena amiga, compañera y fuente de apoyo para acabar mi tesis a tiempo.

Quiero agradecer a Marcelo Bravo por su amistad y apoyo en la revisión de mi tesis.

Quiero agradecer a la Ingeniera Leidy Yaguachi por su compañerismo y amistad, tanto en los buenos como en los malos momentos de la carrera.

Quiero agradecer a Galo Celly, Anita Cardenas y Dickson Armijos por su la amistad y apoyo a lo largo de la carrera.

Quiero agradecer a mi abuelita que siempre, a pesar de tener dificultades económicas, siempre hizo el ahorro y sacrificio necesario para que podríamos ser compañeros de viaje y así conocer culturas y experiencias nuevas para tratar de formarme como una persona de mente abierta.

Quiero agradecer a mi abuelito por enseñarme a utilizar la computadora, a disfrutar usar la computadora, y de una manera muy inconsciente mostrarme la potencia de Linux para resolver problemas.

Quiero agradecer a mi papá por enseñarme que el éxito profesional está en la felicidad con lo que uno hace y eso solo se puede proveer de una fuerte ética profesional y conciencia social que busca el bien de la sociedad y todos sus miembros.

Quiero agradecer a mi mamá por darme la vida, nunca dejar mi lado y mostrarme paciencia y enseñarme el significado del amor incondicional.

Finalmente quiero agradecer a mi esposa Pao e hija Valerie por ser mi motivo de todos mis días para nunca dar nada menos que lo mejor de mi.

Índice general

Índice de figuras

Índice de cuadros

Lista de Abbreveaturas

CSS	**C**ascading **S**yle **S**heets (Hojas de Estilo en Cascada)
FOSS	**F**ree and **O**pen **S**ource **S**oftware (Software Abierto/Gratuito/Libre)[1]
FLOSS	**F**ree/**L**ibre and **O**pen **S**ource **S**oftware (Software Abierto y Libre)
GNU	**G**NU is **N**ot **U**NIX (GNU no es UNIX)
GPL	**G**NU **P**ublic **L**icence (Licencia Publica de GNU)
GPLv3	**GPL v**ersion **3** (Version 3 de la GPL)
HTML	**H**yper**T**ext **M**arkup **L**anguage (Lenguaje de Marcas de HiperTexto)
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol (Protocolo de Transferencia de HiperTexto)
HTTPS	**H**yper**T**ext **T**ransfer **P**rotocol **S**ec ure (Protocolo Seguro de Transferencia de HiperTexto)
IaaS	**I**nfrastructure **a**s **a** **S**ervice (Infrastructura como un Servicio)
IAENG	**I**nternational **A**ssociation of **Eng**ineers
IDE	**I**ntegrated **D**evelopment **E**nvironment (Entorno de Desarrollo Integrado)
IMAP	**I**nternet **M**essage **A**ccess **P**rotocol (Protocolo de Accesso de Mensajes de Internet)
IMAPS	**IMAP** over **S**SL (IMAP atravez de SSL)
IPSec	**I**nternet **P**rotocol **Sec**urity (Seguridad de Protocolos de Internet)
JWT	**J**SON **W**eb **T**okens, un estandar para validar clientes en entornos desconfiables, definido e
KLOC	**K**ilo-**LOC** (mil lineas de codigo)
L2F	**L**ayer **2** **F**orwarding (Renvio de Capa 2)
L2TP	**L**ayer **2** **T**unneling **P**rotocol (Protocolo de Tunel de Capa 2)
LA	**L**earning **A**sset (Activo de Aprendizaje)
LAN	**L**ocal **A**rea **N**etwork (Red de Area Local)
LaTeX	**La**mport **TeX** (sistema de tipografia TeXpor Lamport)
LDAP	**L**ightweight **D**irectory **A**ccess **P**rotocol (Protocolo Ligero de Acceso a Directorios)
LMS	**L**earning **M**anagement **S**ystem (Sistema de Gestion de Aprendizaje)
LO	**L**earning **O**bject (Objeto de Aprendizaje)
LOC	**L**ine **o**f **C**ode [measurement] (medida, linea de codigo)
LOR	**L**earning **O**bjects **R**epository (Repositorio de Objetos de Aprendizaje)
LTI	**L**earning **T**ools **I**nteroperability (Interoperabilidad entre Harramientas de Aprendizaje)
MOOC	**M**assive **O**nline **O**pen **C**ourse (Curso Online Masivo Abierto)
OAS	**O**pen **A**doption **S**oftware (Software de Adoptacion Abierta)
PPTP	**P**oint to **P**oint **T**unneling **P**rotocol (Protocolo de Tunel de Punto a Punto)
RHEL	**R**ed**H**at **E**nterprise **L**inux
SSH	**S**ecure **SH**ell (Intepreter Seguro de Commandos)
SSL	**S**ecure **S**ocket **L**ayer (Capa de Puertos Seguros)
TeX	Sistema de **T**ipografia por Donald Knuth
TLS	**T**ransport **L**evel **S**ecurity (Seguridad en la Capa de Transporte)
SMTP	**S**imple **M**ail **T**ransfer **P**rotocol (Protocolo para Transferencia Simple de Correo)
SMTPS	**SMTP** over **S**SL (SMTP atravez de SSL)
UTPL	**U**niversidad **T**ecnica **P**articular de **L**oja
VCS	**V**ersion **C**ontrol **S**ystem (Sistema de Control de Versiones)
VPN	**V**irtual **P**rivate **N**etwork (Red Privada Virtual)

[1] Significado abierto a debate

WYSIWYG	**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et (Lo Que Ves Es Lo Que Obtienes)
WWW	**W**orld **W**ide **W**eb (Red Informatica Mundial)

Dedicado a las tres mujeres más importantes en mi vida, mi hija Valerie, mi esposa Paola y mi madre Diana…

Capítulo 1

Introducción

Hoy en día se encuentra una necesidad creciente para programadores, informáticos y personas que pueden leer y entender código debido a una tendencia a tratar de siempre automatizar a un mayor grado las tareas humanas con la tecnología. Actualmente la Universidad Técnica Particular de Loja cuenta con métodos clásicos y manuales para enseñar y calificar código que estudiantes programan lo cual no es beneficioso para los docentes, quienes pierden mucho tiempo realizando tareas que se podría realizar de forma automática, ni estudiantes quienes tengan menos atención de sus docentes debido a que tienen mucho código que calificar.

Actualmente, en el mercado existen varios sistemas que permiten escribir, ejecutar y probar código en línea. Uno de estos es Repl.it que puede integrase con sistemas que disponen del protocolo de integración entre sistemas de aprendizaje LTI (Repl.it y Neoreason, Inc., 2017) y puede calificar el código de estudiantes en base a pruebas unitarias pero el mismo tiene un alto coste que no se escala bien y no está orientando tanto al aprendizaje o colaboración si no a evaluación (López, 2017). Por otro lado, Io.livecode.ch es un sistema libre que utiliza contenedores de Docker para permitir la ejecución de código en línea, pero el problema de este sistema es que solo está orientado a enseñanza de programación mas no a realizar pruebas en línea y además no implementa nada de seguridad para proteger el servidor contra usuarios maliciosos (Amin, 2017). Cloud9 IDE es otra herramienta popular para programar en línea, éste integra máquinas virtuales de Ubuntu con el mismo para permitir hacer pruebas en tiempo real (*Your development environment, in the cloud*), pero el mismo tampoco es factible por su falta de interoperabilidad con LTI y falta de enfoque ni en enseñanza ni en evaluación y calificación.

Como cualquier otro tipo de institución con fines comerciales, las universidades e instituciones educativas siempre deben estar buscando la manera en que pueden mantener su competitividad para dar la mejor educación posible y al menor costo. La tecnología de hoy en día ha alcanzado una madurez en donde puede apoyar en temas de automatizar algunos procesos dentro de la enseñanza, evaluación y calificación de estudiantes y de la misma forma liberar el tiempo de professores para que estos puedan ayudar a aquellos estudiantes que realmente necesitan un poco más de apoyo.

Frente a este problema, se propone un sistema para editar código en línea que a su vez integra LMS externos (para autenticación y notas), un servidor de control de versiones externo (para la persistencia de código), y un servicio web de ejecución de código en línea de una forma segura, eficaz y eficiente (para dar un ambiente de ejecución y pruebas tanto para los usuarios del sistema como para calificar de forma automática). A continuación se presenta la problemática para definir el contexto del tema frente al cual se espera resolver dentro de este trabajo de titulación.

1.1. Problemática

Dentro de la universidad, existe la necesidad de contar con una mejor forma de enseñar y evaluar conocimientos de programación en los estudiantes de las titulaciones de Sistemas Informáticos y Computación y Electrónica, especialmente quienes estudian a distancia. Para solucionar este problema, existen sistemas alternativos que se analizan en el capítulo 2, pero estos resultan demasiado costosos para su implementación completa con los recursos actualmente disponibles.

Se considera que cualquier sistema implementado como solución debe ser capaz de captar información acerca de su propio uso con un fin de ayudar con la administración del mismo, debe guardar de manera segura y confiable el código escrito dentro del mismo. Así mismo, debe proporcionar de las herramientas que requieren los estudiantes y sus profesores para desarrollar, probar y calificar de manera eficaz, segura y eficiente el código escrito en línea. Además como plataforma en línea se considera que se podría aprovechar para temas de colaboración como: programación en pares y otras técnicas que apliquen los equipos de desarrollo para cumplir con sus responsabilidades de una forma paralela.

1.2. Metodología

La fase de investigación consiste en realizar una breve recopilación de lo que hay dentro del entorno de despliegue para realizar un estudio del mismo. Además se realiza un estudio de aquellas tecnologías que se utiliza dentro de la solución planteada. Para la investigación de todos los puntos anteriores, se basa en fuentes confiables como: fuentes académicos o comerciales con un fin de explicar alguna tecnología para vender la misma.

Después de la fase de investigación, donde se formula buenas bases y entendimiento del contexto del ambiente en el que se encuentra el presente trabajo, se empieza una fase de análisis el cual se basa en un documento de visión que desglosa las necesidades existentes para el sistema, seguidamente un ERS o especificación de requerimientos demuestra de forma más exacta las capacidades que debe tener el sistema.

Una vez que se tiene entendidas las capacidades que debe tener el sistema, se procede a una fase de diseño donde se realiza diagramas para definir la arquitectura, despliegue e interacción que el sistema tendrá con los usuarios.

Con la definición de los requerimientos y el diseño del sistema como tal, lo que sigue es una fase de desarrollo la misma que se plantea llevar bajo una metodología de desarrollo iterativo incremental que se caracteriza por la revisión contínua con los stakeholders principales del proyecto. De esta forma se puede mantener un poco de agilidad en el desarrollo y consecuentemente desarrollar un producto que cumpla con las necesidades de los usuarios mientras se mantiene dentro de los límites de tiempo y recursos preestablecidos.

Además se propone tener solo iteraciones (basados en entregables) que duren máximo un mes, esto se lo propone para los dos entregables más grandes; el subsistema de ejecución de código por su complejidad y el capítulo seis que se trata del despliegue de la aplicación final. Las otras iteraciones son de dos semanas para tratar de mantener el proyecto en un estado fluido de actividad y presentación de entregables de forma contínua.

Las pruebas se llevaran igual en fases, donde primero en una prueba alfa se integra con sistemas reales para probar temas de integración. En base a los resultados

de estas pruebas se realiza recomendaciones y mejoras previo a la próxima fase de pruebas que son constituidas por pruebas beta donde se prueba la aplicación en un entorno real con usuarios finales, tanto estudiantes como profesores para que puedan dar retroalimentación previo al despliegue de una versión final.

Finalmente se trabaja juntamente con los que serán encargados de mantener la aplicación para el despliegue del mismo. Esto se lleva en fases en donde se levantan primero las dependencias necesarias, como bases de datos y servicios externos entre otros previo al levantamiento y configuración del sistema. La última fase del despliegue consiste en levantar los proxies y servicios auxiliares que ayuden a mejorar temas de seguridad y rendimiento a la aplicación desplegado.

1.3. Objetivos

El objetivo principal es mejorar la enseñanza y evaluación de estudiantes de programación de la Universidad Técnica Particular de Loja. Para cumplir este objetivo se propone un sistema de edición y ejecución de código en línea que ayude a los docentes y estudiantes a interactuar e integrar funcionalidades de sistemas LMS institucionales adicionalmente de servidores de control de versiones institucionales. Para alcanzar este fin, también se plantea cumplir con algunos objetivos específicos en vía al principal.

- Facilitar la autenticación de los estudiantes para que puedan entrar directamente desde sistemas LMS institucionales a través del protocolo LTI. Esto es necesario con la finalidad de que los estudiantes puedan tener una transición suave entre sistemas y no tener mayor interrupción en su aprendizaje.
- Proveer la capacidad de poder ejecutar código en línea. Con esta funcionalidad, el sistema se acerca más a un entorno completo de desarrollo donde estudiantes requieren un mínimo de programas instalados en sus equipos personales.
- Utilizar servidores de sistemas de control de versiones para persistir el código creado por estudiantes de forma externa. Eso permitirá, que de una forma transparente y sin mayor interacción del usuario, ver un historial de código escrito por sus usuarios en la plataforma y también servir de respaldo contínuo de los trabajos que realizan los estudiantes.

1.4. Resultados Esperados

En el presente trabajo de titulación pretende obtener los siguientes resultados:

Sistema Prototipo para Editar Código en Línea que permita a los estudiantes y sus docentes tener una herramienta para mejorar el proceso de enseñanza, aprendizaje y evaluación de la programación.

Sistema Prototipo para Ejecutar Código en Línea que se integró con lo anterior y facilita las pruebas que requieren los estudiantes o profesores con respeto a algún código al cual tengan acceso.

1.5. Organización del Documento

Dentro del capítulo uno, se introduce el tema y problemática, se organiza el documento y se plantea la metodología y objetivos que guían el proyecto desde su inicio hasta su fin.

En el capítulo dos, se ve el estado del arte que contenga un análisis de trabajos relacionados y sistemas similares a la que se plantea desarrollar. También dentro del capítulo dos se encuentra un subsistema del estado del arte en la forma de marco teórico, el cual desglosa la situación actual y permite ver la teoría de las tecnologías de apoyo para el sistema final.

En el capítulo tres, se da un análisis profundo del problema para entenderlo de forma completa previa a las fases siguientes. Dentro del mismo capítulo, en base al análisis, se plantea un diseño para el producto final, el cual se refleja más adelante en el desarrollo durante el curso del capítulo cuatro.

A lo largo del capítulo cuatro, se lleva al lector por el proceso de desarrollo. Por lo tanto el mismo se divide en los varios módulos que se necesitan desarrollar para el proyecto; un módulo de autenticación LTI que permite a los estudiantes autenticarse contra sistemas LMS ya existentes, un módulo de editar código en línea, un módulo de persistencia de código en servidores de control de versiones institucionales, un subsistema de ejecución de código en línea, un módulo de pruebas unitarias para calificaciones automáticas y un módulo para sincronizar calificaciones con sistemas LMS institucionales.

El quinto capítulo se basa en la fase anterior de desarrollo para planificar y realizar pruebas de funcionamiento e integración. De esta manera se encuentran las fallas existentes para resolver los mismos antes de seguir con el despliegue en el sexto capítulo.

A continuación, en el capítulo sexto se planifica y se despliega la aplicación final para que pueda empezar su vida útil dentro de la universidad.

Dentro del séptimo capítulo se resume los resultados y se concluye el trabajo realizado en base al cumplimiento de los objetivos planteados dentro del capítulo uno. Además se da recomendaciones e ideas para trabajos futuros en base al proyecto realizado y para la evolución continua del prototipo final.

Al final de todos los capítulos anteriores se encuentran los anexos con documentos de apoyo como: el documento de vision (Anexo A) y especificacion de requisitos (Anexo B), implementacion tecnicas llevadas a lo largo del desarollo (Anexos C, D, E), pruebas realizadas para validar funcionalidades (Anexos F, G, H), y ubicacion y licencias de codigo llevado en el curso de la tesis (Anexo I).

Capítulo 2

Estado del Arte

Sin la ciencia, no podría existir la ingeniería. Sin investigación y curiosidad acerca del entorno, no podría existir innovación y creatividad frente los problemas que se da. Es por tal motivo que el siguiente capítulo trata de un estudio del entorno en cuanto tecnologías y a trabajos e investigaciones previos que han sido realizados por terceros para en base a ello dar contexto a la investigación actual y aprender de lo bueno y malo de cada antecendente.

En la sección 2.1 de este capítulo, se presenta el marco teórico donde se realiza una investigación y estudio a las tecnologías y otros aspectos de entorno previo al planteamiento de una solución al problema introducido anteriormente. En la sección 2.2, se encuentra con un estudio de trabajos relacionados a los temas abarcados en este trabajo de titulación, mientras que en la sección 2.3 se realiza un estudio de sistemas similares en funcionalidad. Y finalmente se cierra el capítulo con la sección 2.4 donde se de una discusión y conclusión a todo lo analizado y presentado del capítulo con comparaciones de los mismos.

2.1. Marco Teorico

Para llevar a cabo de forma exitosa el trabajo de titulación y entender el contexto del ambiente en que sea implementado es importante establecer una línea base de conocimiento que se ve a continuación y dividido de la siguiente forma:

Aspectos de Propiedad Intelectual que se topa con las licencias y culturas de código abierto y libre que se encuentra actualmente en el entorno.

Aspectos Ambientales del Entorno de Desarrollo se trata de dar mejor contexto al entorno en el cual se encuentra la aplicación planteada. Esto incluye pero no está limitado a conceptos teóricos, aplicaciones y protocolos.

2.1.1. Aspectos de Propiedad Intelectual

Hoy en día, en un mundo cada vez más conectado y con cada vez más informacion compartida entre personas distintas, es importante conocer bien temas de derechos de autor y propiedad intelectual, no sólo para desarrollar y despues dar licencia a un trabajo de titulación, si no también para entender lo que se puede y no se puede hacer dentro de un entorno de desarrollo que cada vez involucra más algún codigo o trabajo que fue desarrollado alguna licencia abierta o libre.

Software Libre

Software Libre es un movimiento sociopolítico que busca liberar códigos fuente. Se divide en dos campos que son: los de Open Source (Fuentes Libres) que quieren liberar código fuente sólo en base a los méritos de desarrollo colaborativo y los de Free/Libre Software (Software Libre) que buscan liberar código fuente en base a ciertos derechos de compartir, colaborar y tener control de lo que hacen con sus dispositivos (su propiedad) para todos los usuarios que ocupen el software (Stallman, 2016a) (Stallman, 2016c). En el contexto de esta tesis, Software Libre se refiere al segundo campo, no al primero. Para el mismo, la fundación GNU con su fundador Richard M. Stallman define 4 libertades mínimas que deben ser cumplidas para constituir Software Libre y los cuales se los enumera desde el 0 hasta el 3 (Stallman, 2017a) (Stallman, 2016b):

0. La libertad de ejecutar el programa para cualquier propósito que desee
1. La libertad de estudiar cómo funciona el programa y poderlo modificar como desee. Un requisito para esta libertad es que el usuario tenga acceso al código fuente original.
2. La libertad de distribuir copias del programa original con terceros.
3. La libertad de distribuir copias de sus versiones modificadas a terceros. Un requisito para esta libertad es que el usuario tenga acceso al código fuente original.

Donde el software no cumpla con uno de los anteriores, ya deja de ser considerado libre. Sólo por ser software libre no significa que no puede ser comercializado, únicamente se requiere nuevos modelos de negocio los cuales si se los puede encontrar en uso diario (Stallman, 2016b). Como sociedad, Stallman argumenta que es importante proteger estas libertades porque con ellas avanza la humanidad, como la libertad de expresión, ya que las libertades que se proponen proteger, buscan sostener una sociedad que trabaja por el bien de todos y no solo de unos pocos que saben más de la funcionalidad de ciertos aspectos tecnológicos (Stallman, 2016c). En la época actual de gobiernos y empresas que espían sin vergüenza, se ha revivido el movimiento político y se ha convertido en algo más necesario debido a que todos tenemos y queremos nuestro derecho a la privacidad (Stallman, 2017a).

OAS OAS o Adopción de Software Libre es la tendencia que hay en el mundo empresarial de adoptar soluciones de tecnologías abiertas como los que ofrecen el mundo de software libre ya que las mismas pueden llegar a ser superiores en cuanto su eficacia y costo, que lo que ofrece su competencia comercial. Se estima que más del 78 % de instituciones usan software de Fuentes Abiertas y menos del 3 % indican que no utilizan nada de software de fuentes abiertas. Eso demuestra un enorme mercado creciente y emergente a nivel mundial (*The Next Wave in Software: Open Adoption Software (OAS)*).

GNU El proyecto GNU fue lanzado por el señor Richard Mathew Stallman en el año 1983 con el fin de desarrollar en comunidad un sistema operativo que podría reemplazar los sistemas operativos UNIX de aquella época con algo que sería un 100 % software libre (Stallman, 2014). Pero nunca terminaron todo el proyecto, específicamente el núcleo del sistema operativo, conocido como GNU Hurd, porque

antes llegó otro núcleo desarrollado por un joven estudiante de sistemas llamado Linus Torvalds quien llamó su clon libre de UNIX, Linux, y de esta forma, unido con todo lo que se había desarrollado el proyecto GNU, se formó un sistema operativo completamente libre (Stallman, 2017b).

GPL Aunque es la licencia más común en proyectos de software libre y usada para una mayoría de ellos, especialmente dentro del proyecto GNU, la GPL o GNU Public License no es la única licencia libre que existe (Free Software Foundation, 2016). Fue diseñada específicamente en la década de los 1980s para proteger la libertad de los usuarios de software (Stallman, 2016c). La última versión, GPLv3 fue publicada en Junio de 2007 y construye encima de versiones anteriores para tratar de asegurar de mejor manera las 4 libertades que tiene que garantizar cualquier licencia que quiere ser software libre (Smith, 2014) (Free Software Foundation, 2007).

2.1.2. Aspectos Ambientales del Entorno de Desarrollo

El contexto dentro del cual se desarrolle cualquier proyecto, conocer las tecnologías existentes dentro de un ambiente en los proyectos tecnológicos, son puntos importantes que se debe tomar en cuenta previo a la ejecución del mismo. Es por tal motivo que a continuación se presenta algunas de las tecnologías con que se va a encontrar el proyecto y la teoría tras de ellos.

Tipos de recurso de aprendizaje

CUADRO 2.1: Tipos de Recurso de Aprendizaje.

Concepto	Definición
LA	LA es un activo de aprendizaje el cual representa un archivo que no llega al nivel de un objeto de aprendizaje. También se los guarda en repositorios de objetos de aprendizaje (*About Learning Repository*).
LO	LO es un objeto de aprendizaje y es un recurso que ayuda a los estudiantes con su aprendizaje, por ejemplo un tema (que se puede enseñar con diapositivas, enlaces, documentos, etc), pruebas, exámenes, etc (*About Learning Repository*).
LOR	LOR es un repositorio de objetos de aprendizaje unidos por docentes y otros profesionales educativos para enseñar a sus alumnos de la mejor forma posible (*About Learning Repository*).

MOOC

MOOCs o Cursos Online Masivos Abiertos, son cursos abiertos en línea que guían a los estudiantes paso a paso en su aprendizaje sin límites como: costos, fronteras o culturas. Los mismos usan expertos del área en cuestión para dar la mejor educación posible a través de videos, pruebas, foros, redes sociales, charlas y artículos que promuevan debate y reflexión (*What is a MOOC? Massive Open Online Courses Explained*).

CMS

Un sistema de gestión de cursos es una colección de herramientas que ayudan proveer un ambiente en línea para interacciones entre integrantes de una clase. Esto puede incluir espacio para publicar anuncios, recolección de deberes, gestión de

notas, tanto para que los docentes puedan calificar a sus alumnos y los mismos alumnos puedan ver sus notas en tiempo real, integración con sistemas de correos institucionales, chat en vivo y foros que permitan responder a entradas previas (*Course Management Systems*).

LMS

Según Mindflash, un vendedor de sistemas educativos (*Mindflash Learning Management*), los LMS, o Sistemas de Gestión de Aprendizaje, son herramientas muy útiles para instituciones educativas y otras entidades económicas que lo apoyen para organizar y gestionar recursos educativos, estudiantes, docentes y cursos. Las mismas son un aspecto crítico de la tendencia que tiene el sector educativo a digitalizarse que se puede ver en la aproximadamente 600 LMS que existen para elegir. Características comunes entre ellos incluyen: (*What is an LMS?*)

- Listas de estudiantes para tomar asistencia entre otras interacciones que puede haber entre estudiantes y sus profesores.
- Matrículas
- Gestión de documentos
- Soporte para educación distribuida; por ejemplo alumnos y/o docentes a distancia
- Calendarios de curso para dar a conocer cronogramas, fechas para pruebas, exámenes y deberes.
- Interacción estudiantil a través de correo electrónico, foros y chat.
- Comprobación de conocimiento a alumnos con pruebas y exámenes.
- Sistemas de calificación que pueden ser automatizados (en casos objetivos), manuales por parte del docente (para casos más subjetivos) e híbridos cuando un profesor ofrece ambos casos en un curso.

Moodle Moodle es un LMS de software libre, licenciado bajo el GPLv3 (Dougiamas y Free Software Foundation, 2016), que ofrece las características típicas de un LMS como herramientas de colaboración (wikis, foros, glosarios, etc), gestión e interacción con estudiantes (notificaciones, gestión de progreso, notas, etc), en adición a la gestión de usuarios y matrículas (Moodle Community, 2016). Actualmente es lo que usa la Universidad Técnica Particular de Loja (UTPL) con sus docentes y alumnos a través del Entorno Virtual de Aprendizaje (López, 2017).

OpenCampus OpenCampus es un sistema para instituciones educativas con el objetivo de ayudar a gestionar sus estudiantes y procesos a través de una combinación de OAS y software propietario con el fin de proveer a todos el mejor acceso a recursos educativos (*Technology*), el mismo integra sistema de gestión de campus (CMS), gestión de escuelas de postgrado (GSM), sistema de gestión de aprendizaje (LMS), gestión de escuelas de medicina, sistema de aprendizaje electrónico, gestión de la salud en el lugar de trabajo, gestión de objetivos, sistema de gestión de información estudiantil, gestión de clases, sistema de gestión de dato de investigación, sistema

de gestión de ensayos clínicos, gestión de becas de investigación y gestión de laboratorios (*OpenCampus*). De esta manera ofrece características comunes de los LMS como exámenes, cursos en línea, rastreo académico para docentes, alumnos y administradores (*Technology*) (*For Universities*). La UTPL está ocupando esta tecnología actualmente para ofrecer "cursos en línea... de forma abierta y libre" (UTPL, 2017c).

Open edX Una organización sin fines de lucro famosa por sus cursos abiertos masivos (MOOCs), EdX, se ha liberado con una licencia libre su plataforma bajo el nombre Open edX. La misma ofrece:

- CMS para gestionar varios cursos
- LMS para presentar contenido a estudiantes y profesores
- Open edX Studio para el diseño de cursos desde su contenido y políticas de calificación hasta el horario que seguirá y el equipo de docentes.
- Un plataforma para MOOCs
- XBlock para construir arquitecturas de componentes donde los docentes puedan agregar funcionalidad a su curso sin meterse mucho al resto del sistema.
- Foros
- Recolección de Datos para la toma de decisiones estratégicas

Según la documentación del mismo, Open edX utiliza una arquitectura de NGinX con Gunicorn y Django. Para el despliegue del mismo en entornos de desarrollo se ofrece una máquina virtual preconfigurada denominado "DevStack" (*About Open edX*). Como tecnología de backend, el OpenCampus de la UTPL utiliza Open edX (López, 2017).

LTI

LTI es un estándar para asegurar interoperabilidad entre sistemas educativos. Fue diseñado y gestionado por IMS Global Learning Consortium (*Learning Tools Interoperability® Background*), un conjunto de instituciones y organizaciones que desean mejorar procesos educativos (*Contributing Members, Affiliates, and Alliance Participants*), con el fin de simplificar el proceso de interconección de aplicaciones de aprendizaje (conocidos en el estándar como herramientas [de aprendizaje]), como los que saben ofrecer terceros, con ambientes educativos (conocidos en el estándar como Consumidores de Herramientas [de aprendizaje]) que pueden ser sistemas de gestión de aprendizaje (LMS), portales de aprendizaje, repositorios de objetos de aprendizaje (LOR), u otros ambientes educativos (*Learning Tools Interoperability® Background*). Moodle ofrece una implementación estándar para comunicarse con sistemas de terceros (Moodle Community, 2016).

Sistema de Control de Versionamiento

Un sistema de control de versiones mantiene y organiza un historial de versiones de uno o más archivos con el fin de poder intercambiar versiones rápidamente. Son muy usados para quienes trabajan con código fuente y también artistas (Chacon y Straub, 2014a) que necesitan llevar un historial de sus obras de arte, pero su utilidad no termina allí ya que son diversos los grupos que pueden aprovechar los beneficios que ofrecen estos sistemas. El sistema común que se suele usar actualmente es

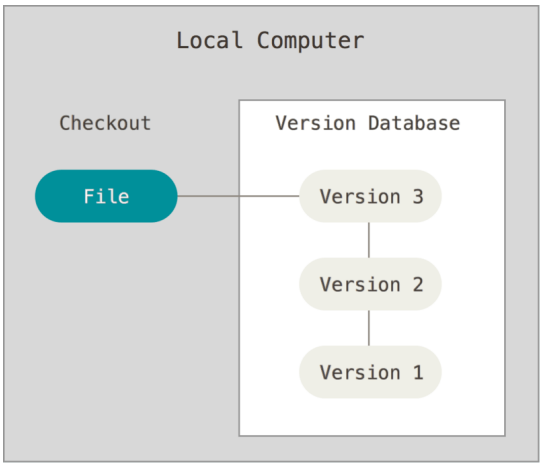

FIGURA 2.1: Sistema de Control de Versionamiento Local. (Chacon y Straub, 2014a).

copiar archivos que deseen versionar, típicamente dándoles una nueva ubicación o nombre dado que es una solución muy simple para lograr versionar, pero eso puede ser peligroso ya que es muy fácil trabajar sobre una versión incorrecta, por lo que se inventaron los sistemas de control de versionamiento, programas especializados en ayudar a los seres humanos llevar una historial de versiones en uno o más archivos.

Inicialmente esos eran sumamente locales, como RCS, que mantiene una base de datos con todas las versiones (almacenadas como los cambios que hubieron en comparación con la versión anterior, una práctica conocida como respaldo incremental) y en base a eso permitir acceso a la versión actual o cualquiera de los otros guardadas previamente, mira figura 2.1. (Chacon y Straub, 2014a).

Luego se dieron cuenta que seria bueno poder colaborar sobre las mismas versiones, entonces se inventó lo que se conoce como Sistemas de Control de Versionamiento Centralizado los cuales permiten un alto grado de control/seguridad sobre quién y cuando puede editar dentro de la base de datos de contenido, ya que las mismas siguen un modelo arquitectónico cliente-servidor (Chacon y Straub, 2014a). Eso ha resultado que empresas grandes que necesitan desarrollar sistemas grandes en paralelo con un cierto grado de seguridad, quieran usar un sistema de control de versionamiento de esta clase, aunque esto está cambiando ya que las empresas están dándose cuenta que pueden optar por una solución híbrida (Rose, 2015). Ejemplos de estos CVCS son CVS, Subversión y Perforce. Pero el problema viene a ser que todo el repositorio tiene un solo punto de falla porque si cae el servidor o le pasa algo, se pone en riesgo la productividad contínua de quién colabora y en el peor de los casos, la integridad del repositorio. Esta propiedad se puede ver en la figura 2.2 (Chacon y Straub, 2014a).

Con la explosión de desarrollo de software libre, hubo una necesidad de poder trabajar grandes cantidades de personas desconocidas en paralelo (Raymond, 1999), y para resolver este problema y también la de integridad de datos que presenta el único punto de falla de los Sistemas de Control de Versionamiento Centralizado, se inventó Sistemas de Control de Versionamiento Distribuidos como Git, Mercurial, Bazaar y Darcs. Cada repositorio es un espejo de los demás en una arquitectura distribuida o red de iguales. Con esta arquitectura (mira figura 2.3), ningún nodo viene a ser un punto de falla ya que se puede espejar entre cualquiera de ellos debido al hecho de que todos tienen una copia local de todo lo que tienen los demás (Chacon

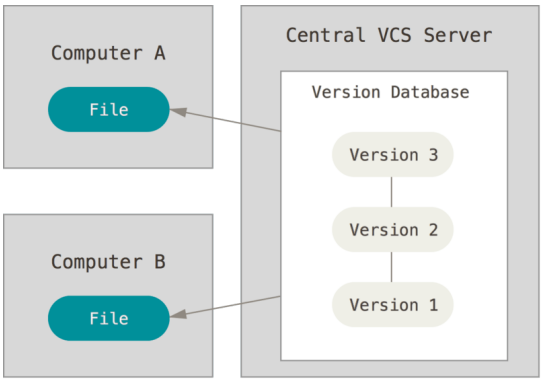

FIGURA 2.2: Sistema de Control de Versionamiento Centralizado. (Chacon y Straub, 2014a).

y Straub, 2014a).

Git

Git es un sistema de control de versiones distribuido que a diferencia de otros VCS guarda con cada versión una copia entera de los archivos modificados (otros sistemas de control de versionamiento suelen guardar solo las diferencias entre versiones de forma incremental, mira la diferencia entre la figura 2.4 y la figura 2.5) (Chacon y Straub, 2014b).

Esto hace que Git se convierta más en un pseudo sistema de ficheros que un simple VCS. Además, significa mayor integridad de los datos guardados a costo de mayor consumo de almacenamiento. Trabaja de esta manera distribuida, es decir el repositorio existe en esta forma en todo lado donde se encuentra y por lo tanto la mayoría de operaciones de Git se puede realizar de manera local sin ninguna conexión de red (Chacon y Straub, 2014b). Volviendo al tema de integridad de datos, Git utiliza SHA-1 como algoritmo criptográfico para calcular los hash de cada cambio que se realiza en base a una combinación de qué información contiene el cambio, su metadata como autor, fecha, hora y fecha en adición a los SHA-1 de su(s) cambio(s) padre(s)[1] de tal forma que es difícil si no prácticamente imposible[2] para un atacante modificar el historial sin dejar evidencia de lo que se ha hecho. Por defecto funciona así (Chacon y Straub, 2014b), pero si uno desea asegurar la integridad del historial aún más, se integra Git con GPG para ofrecer mayor seguridad ya que con eso se puede firmar digitalmente cambios realizados para demostrar su autenticidad (Chacon y Straub, 2014c).

La mayoría de operaciones normales que se hacen en Git no eliminan datos ya guardados en el repositorio previamente lo cual lo hace una buena herramienta para personas que recién está entrando al mundo de Sistemas de Control de Versionamiento y permite que se pueda recuperar de la mayoría de errores que se puede cometer y una recuperación casi garantizado de datos “perdidos”. Eso es gracias en parte a la filosofía que se refleja en el diseño del mismo ya que una prioridad primordial era la integridad de los datos (Chacon y Straub, 2014b).

[1] La mayoría de commits solo tienen un padre, pero aquellos que unen varios historiales, tienen dos padres

[2] Un equipo de investigación de Google y CWI Amsterdam ha encontrado una vulnerabilidad que recién publicaron y les permite atacar SHA-1 y encontrar colisiones hash (Stevens y col., 2017)

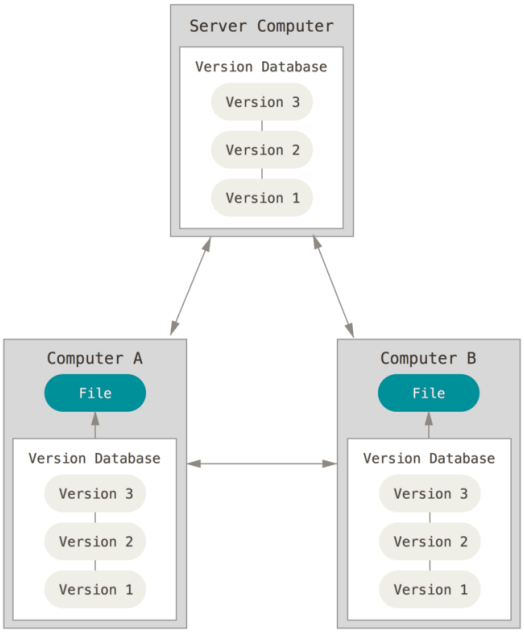

FIGURA 2.3: Sistema de Control de Versionamiento Distribuido. (Chacon y Straub, 2014a).

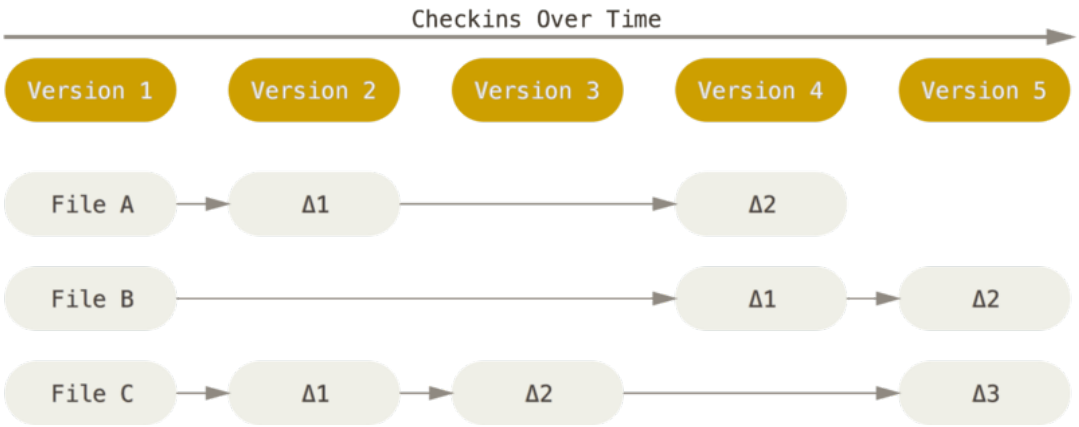

FIGURA 2.4: Sistema de Control de Versiones llevado de forma incremental. (Chacon y Straub, 2014b).

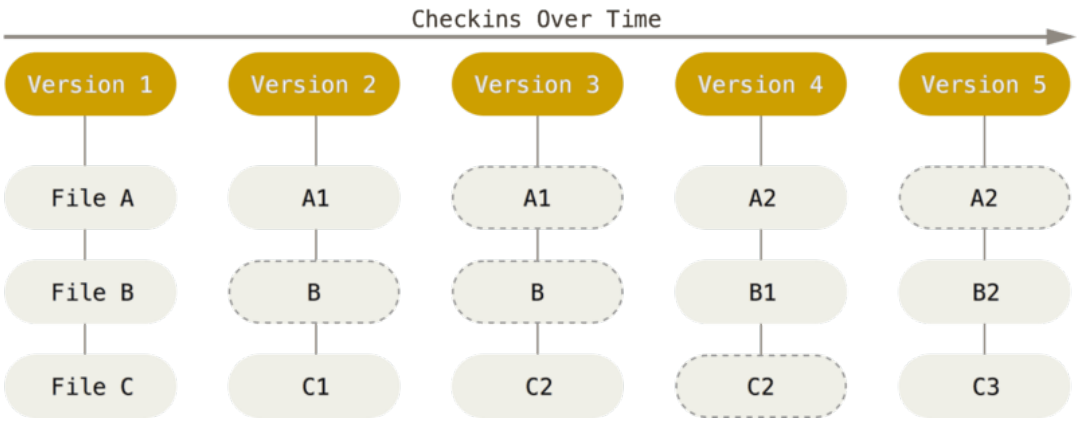

FIGURA 2.5: Sistema de Control de Versiones llevado con archivos enteros. (Chacon y Straub, 2014b).

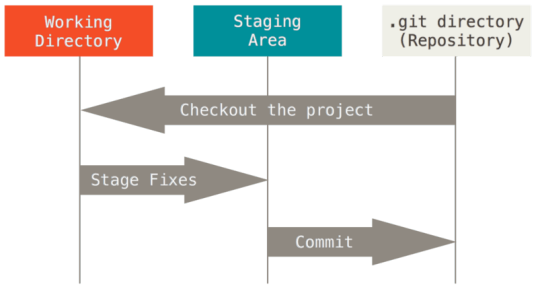

FIGURA 2.6: Los tres fases de Git y sus relaciones. (Chacon y Straub, 2014b)

Cada repositorio de Git trabaja con tres fases que son el directorio de trabajo donde se realiza cambios, una área de preparación donde se marca cambios listos para guardar y una base de datos junto un sistema de ficheros donde se guardan los cambios. Al saltarse entre versiones, se sobreescribe los contenidos del directorio de trabajo con lo que se extrae del sistema de ficheros que tiene los cambios guardados. Esas tres fases y su relación se puede ver en la figura 2.6 (Chacon y Straub, 2014b).

GitLab GitLab es un servidor de Git ofrecido por GitLab Inc. Se ofrece en dos versiones: una versión gratis (Freyd y col., 2017) de código abierto liberado bajo el nombre GitLab Community Edition (GitLab CE) y una versión pagada que forma la línea principal de negocio de GitLab Inc. que lo venden bajo el nombre GitLab Enterprise Edition (GitLab EE) (Sijbrandij y col., 2017). También proporcionan hosting de las mismas versiones alojados por ellos en GitLab.com (allí se puede usar GitLab EE gratis (Nierop y col., 2017) (Freyd y col., 2017)) y en GitHost.io (donde se puede arrendar servidores para alojar instancias de GitLab CE o EE y su respectivo infraestructura de apoyo (GitLab Inc., 2017)) (Sijbrandij y col., 2017). Actualmente la Universidad Técnica Particular de Loja utiliza un servidor de GitLab CE como su instancia institucional de Git (UTPL, 2017d).

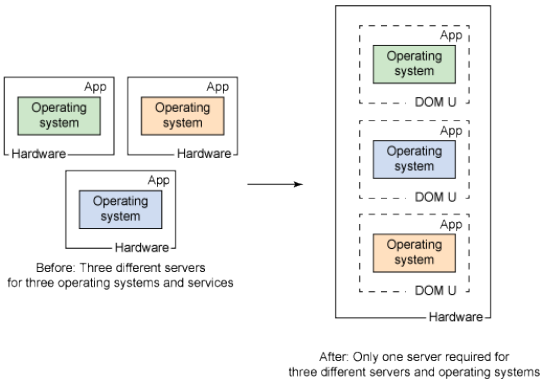

FIGURA 2.7: Antes y después de virtualizar aplicaciones. (Tholeti, 2011).

Virtualización

Virtualización es la creación de recursos virtuales en software para usar los mismos en lugar de un recurso físico. Este puede ser usado en áreas diversas como aplicaciones, servidores, almacenamiento y redes. Según la empresa VMware, que se dedica a vender productos de virtualización, la misma es la estrategia más efectiva para reducir costos de TI mientras que aumenta la eficiencia y agilidad de un negocio de cualquier tamaño. Llegan a esta conclusión porque es tan común que se ha convertido en estándar industrial que solo se ocupa en entre 5 % y 15 % de la capacidad de los servidores debido a que se les pone solo un sistema operativo y aplicación a la vez. Virtualización permite resolver estas deficiencias en permitir la división lógica de equipos y distribuir estas partes donde más se los necesitan para aumentar la tasa de eficiencia y reducir la cantidad de recursos físicos necesarios. Con esta clase de tecnología, se puede reducir y en algunos casos eliminar tiempo fuera de servicio. Las facilidades de portabilidad y agilidad de ajustar recursos hace que se puede reducir los tiempos y costos de administración. Se domina hipervisor un software que gestiona virtualización (*What is Virtualization?*). Tholeti y IBM tienen un concepto similar de reducción de recursos físicos y gastos de operación, como se indica en la figura 2.7, la optimización de recursos físicos después de la virtualización.

Las Máquinas Virtuales Una Máquina Virtual es una sistema computacional entero, con sistema operativo y aplicación corriendo sobre recursos virtuales. Por lo tanto cada máquina virtual es completamente aislada e independiente de los demás de tal forma que se puede tener varios sistemas operativos y aplicaciones corriendo sobre un solo servidor físico. Eso permite tener varias características importantes:

División de Recursos Se puede dividir recursos físicos entre varios sistemas operativos corriendo sobre la misma máquina.

Aislamiento Proveer mayor seguridad en prevenir o restringir acceso entre las varias máquinas virtuales.

Encapsulación de Persistencia Todo el estado de la máquina virtual se lo puede guardar en archivos, los mismos que pueden ser sincronizados entre varios equipos o ubicaciones.

Independencia de Hardware Se puede usar la máquina virtual en cualquier hipervisor que lo soporte.

Todas estas características, permiten la consolidación de servidores para utilizar menos y acortar costos (*What is Virtualization?*).

Virtualización de Servidores Con la virtualización de servidores, se puede reducir el número de servidores necesarios dentro de una granja, especialmente cuando los mismos son miembros de un cluster que realiza balanceo de recursos frente su carga para optimizar la eficiencia de toda la granja (*What is Virtualization?*).

Virtualización de Redes Con virtualización de redes se puede simular todo el tráfico y funcionamiento de una red sin mayor retardo. Se puede virtualizar interfaces de red, switch, routers, cortafuegos, balanceadores de carga, VPNs y más. Como los otros tipos de virtualización, esta ofrece los mismos beneficios de independencia, costos y escalabilidad (*What is Virtualization?*).

Virtualización de Escritorios Virtualización de Escritorios se trata de usar recursos centralizados para dar escritorios de trabajo a usuarios remotos. Esto permite a cualquier organización reducir sus gastos tecnológicos debido a que el mismo permite menor inversión en tecnología para la misma cantidad de usuarios mientras al mismo tiempo facilita temas de administración de los equipos de los empleados en una organización y seguridad de la información de la misma (*What is Virtualization?*).

Hipervisores

Un hipervisor es una capa intermedia de software que gestiona la interacción entre lo virtualizado y hardware u otro software que ayuda realizar las peticiones de lo virtualizado (*What is Virtualization?*). De la misma manera, el mismo hipervisor se encargan de la seguridad para asegurar que lo que sea virtualizado no pueda salir de su ambiente y atacar otras partes virtualizadas o no virtualizados (VMWare, Inc., 2014) (Wilson, Day y Taylor, 2011). La figura 2.8 muestra los tipos clásicos de hipervisor que se presentan a continuación, los hipervisores de tipo 1 que son a bajo nivel en lugar de un sistema operativo y los hipervisores de tipo 2 que son una aplicación montada encima de un sistema operativo.

Hipervisores de Tipo 1 Un Hipervisor de Tipo 1 se ejecuta directamente sobre hardware físico (Tholeti, 2011) y por lo tanto puede llegar a conseguir mejor rendimiento y seguridad pero con el costo de requerir en muchos casos mayor configuración y administración para poder alcanzar su rendimiento óptimo.

Xen Xen es un hipervisor de tipo 1, que corre directamente sobre hardware. Es el único hipervisor de su clase que es completamente disponible bajo una licencia abierta y forma una base por muchas aplicaciones comerciales y no comerciales como virtualización de servidores, IaaS, virtualización de escritorios, aplicaciones de seguridad, y dispositivos embebidos. Dentro del mercado, es la tecnología detrás de las nubes más grandes (Xen Community, 2016). En la figura 2.9 se puede encontrar la arquitectura Xen física/lógica para entender mejor como funciona la misma.

Para el mismo, se considera las siguientes características:

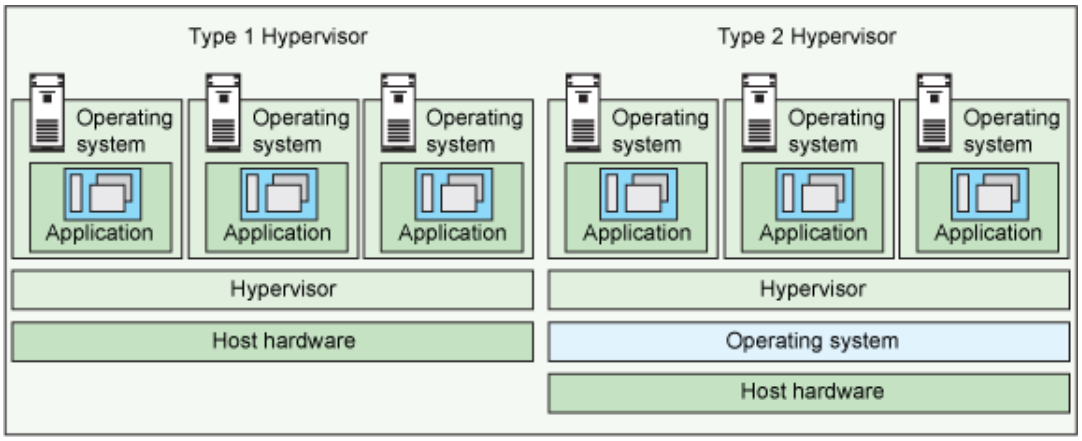

FIGURA 2.8: Tipos Clásicos de Hipervisor (Tholeti, 2011).

- La implementación y uso de un micronúcleo que ocupa un mínimo de memoria y tiene interfaz externa limitada, el cual ayuda proporcionar mayor seguridad y estabilidad en comparación con otros hipervisores. El hipervisor en si esta escrito en menos de 150 KLOCs que para un sistema operativo es muy liviano. Puede ser tan liviano porque no necesita tener conocimientos de operaciones de entrada/salida como redes o almacenamiento debido a que Dom0 se encargará de estos tareas. Al mismo tiempo, las demás máquinas virtuales no son privilegiados y siempre tienen que comunicarse con Dom0 para realizar cualquier operación con respecto al hardware físico. Por lo tanto la muerte del Dom0 es fatal para todas las máquinas que están en el Hipervisor.
- La arquitectura de Xen ocupa una maquina virtual que se domina Dom0 el cual contiene todos los drivers para el hardware y también la herramientas necesarias para ejercer control sobre todo el hipervisor. Estás funcionalidades son independientes del sistema operativo motivo por el cual el sistema operativo residente del Dom0 puede ser Linux, una BSD, OpenSolaris, etc...
- Como los drivers que controlan hardware físico existen dentro de una máquina virtual, se los consideran drivers aislados. En caso de que pase algo, solo hay necesidad de reiniciar la máquina afectada y el resto del sistema puede seguir funcionando normalmente.
- Con una tecnología llamada paravirtualización, en lugar de optimizar al nivel de hipervisor se optimiza los sistemas operativos que son virtualizados para de esta manera mejorar su rendimiento incluso en hardware que no tiene soporte para virtualización. (Xen Community, 2016).

Hipervisores de Tipo 2 Un hipervisor de Tipo 2 se ejecuta por encima de un sistema operativo que le ayuda con su gestión interna como si fuera una aplicación más (Tholeti, 2011), lo que hace este tipo de hipervisor más lento que hipervisores de tipo 1 (por tener un mayor número de capas y también por virtualizar partes del hardware) pero a su vez más fáciles de configurar y administrar. El mismo hecho de ejecutar por encima de un sistema operativo también vulnera la seguridad de todo el sistema en proveer un superficie de ataque más grande para actores maliciosos.

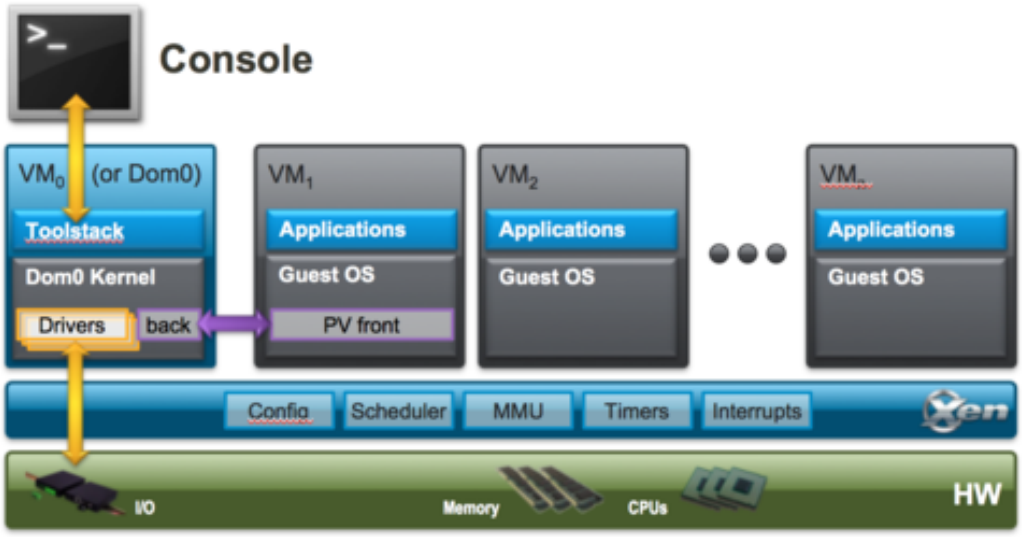

FIGURA 2.9: La arquitectura de Xen. (Xen Community, 2016).

Qemu-KVM QEMU es un proyecto de software abierto con el fin de crear un emulador y virtualizador de procesadores de varias arquitecturas para una variedad de sistemas operativos. Permite el uso de backends como KVM o Xen para dar rendimiento casi nativo (*QEMU*). Aunque su despliegue normal requiere un sistema operativo, basado en Linux, intermedio debido a que es un módulo que se integra directamente en el núcleo del mismo sistema operativo, Wilson et. al. consideran KVM como un hipervisor de tipo 1 debido a sus características y implementación a bajo nivel cerca al nivel hardware física, mientras que su arquitectura lo hace aparecer como un hipervisor de tipo 2, que es como se lo está considerando dentro del contexto de este trabajo (mientras que otros autores lo consideran un hipervisor de tipo 1.5, ni uno ni el otro si no algo híbrido dentro de la taxonomía de hipervisores). KVM es el "hipervisor estratégica para IBM" debido a (1) su bajo costo por ser tecnología completamente abierta, (2) su alta evolución y madurez, (3) alta evolución y madurez de Linux (su tecnología fundamental atras), (4) eficiencia y rendimiento alto, (5) una comunidad de desarrollo muy activa y responsable frente fallos de seguridad, (6) la idea de que muchos ojos viendo código lo hace más seguro[3], (7) varias tecnologías de seguridad que integra que falta la competencia, (8) mayor flexibilidad que ofrece frente otros hipervisores competidores y (7) control que IBM puede ejercer sobre el proyecto de KVM (Wilson, Day y Taylor, 2011). En la figura 2.10, se puede encontrar la arquitectura lógica para entender visualmente la funcionalidad de la misma.

VirtualBox VirtualBox es un hipervisor para virtualización de arquitecturas estándar x86 y x86_64 tanto para uso casero como uso empresarial. Para ello ofrece muchas características, rendimiento optimizado, y un gran número de sistemas operativos que permite virtualizar como para ser virtualizados todo bajo una licencia de software libre, la GPL versión 2 (*VirtualBox*). Su arquitectura de ser montado encima de cualquier sistema operativo y apoyarse en ello para realizar operaciones por parte de un sistema operativo y hardware virtualizado hacen este hipervisor un ejemplo clásico de un hipervisor de tipo 2.

[3]Uno de las características principales de Código Abierto

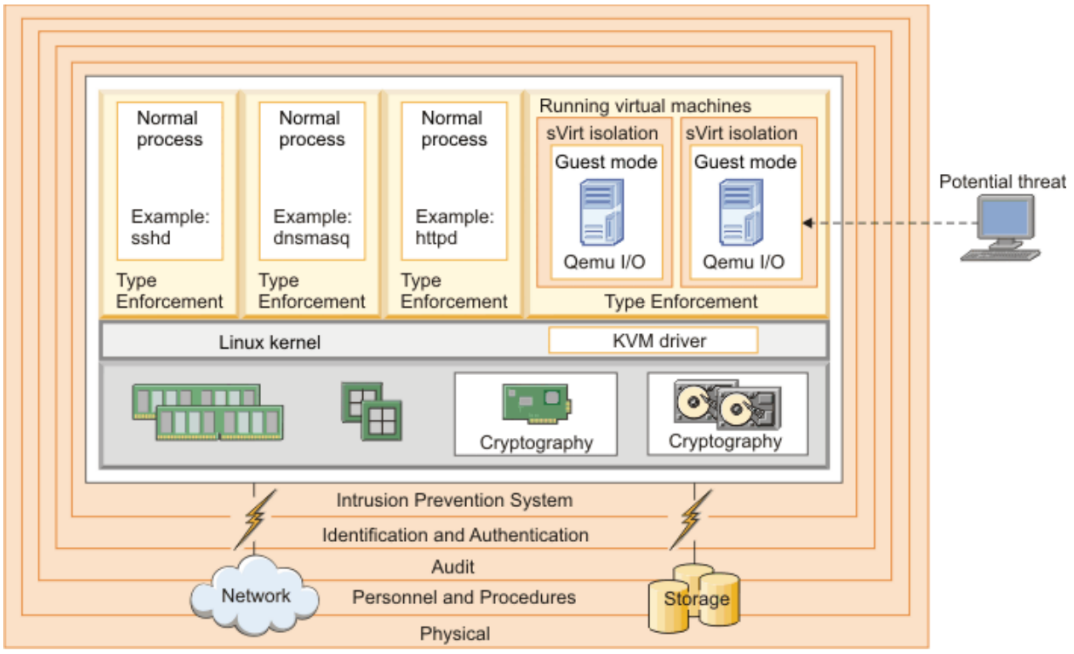

FIGURA 2.10: Arquitectura de KVM/QEMU. (Wilson, Day y Taylor, 2011).

Virtualización a Nivel de Sistema Operativo (Contenerización) Desde antes de la incepción de los sistemas Linux, los sistemas de la familia Unix han tenido un mecanismo de aislamiento conocido como chroot el cual genera un espacio aislado de usuario para contener alguna aplicacion o varias aplicaciones mientras que todos siguen compartiendo el mismo sistema operativo o kernel por detrás. La tendencia de contenerización que existe hoy en día sigue por la misma línea pero lo implementa de una forma aún más avanzada. Eso permite a muchos usuarios que se desconfían mutuamente unos en otros utilizar el mismo hardware mientras que al mismo tiempo quedan completamente aislados unos de otros con menor impacto al rendimiento o una necesidad de hardware que soporta la carga adicional que virtualizacion genera (Teimouri, 2016), mirar la figura 2.11. Algunos sistemas de este tipo de aislamiento que son populares hoy en día son:

- chroot
- Docker
- LXC
- LXD
- Linux-VServer
- OpenVZ
- Solaris Containers
- FreeBSD jail
- Hyper-V containers (Microsoft)
- Photon (VMware)
- vSphere Integrated Container (VMware)
- Windows Containers (a partir de Windows Server 2016)

Debido al hecho de que muchas veces cada máquina virtual necesita su propio hardware virtual y sistema operativo, se termina consumiendo mucho más RAM y ciclos de CPU de tal manera que un servidor típicamente puede soportar dos a tres veces más servicios si están en contenedores en lugar de si son virtualizados (Teimouri, 2016). La diferencia se puede ver en la figura 2.12.

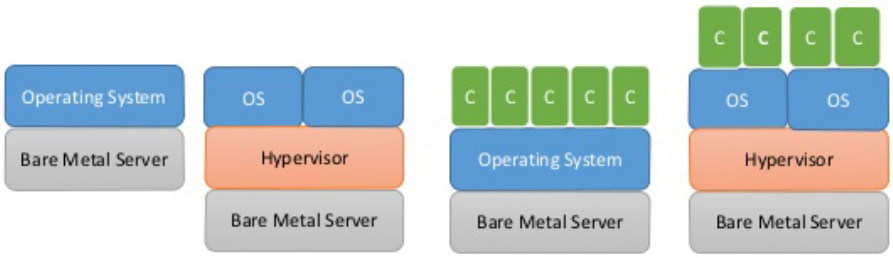

FIGURA 2.11: Un concepto básico de que son los contenedores (Teimouri, 2016).

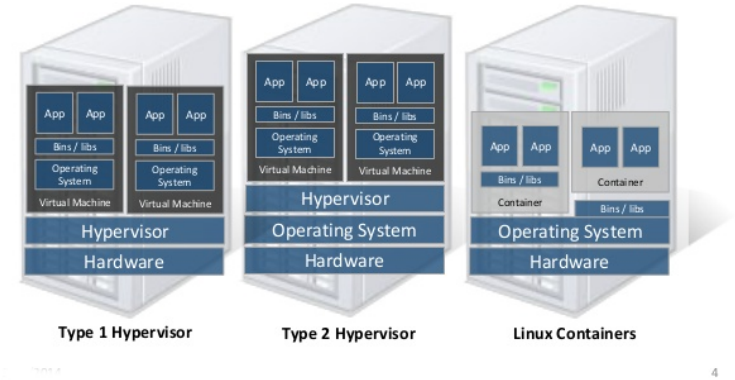

FIGURA 2.12: La diferencia que abarca contenedores de los tipos de hipervisores (Teimouri, 2016).

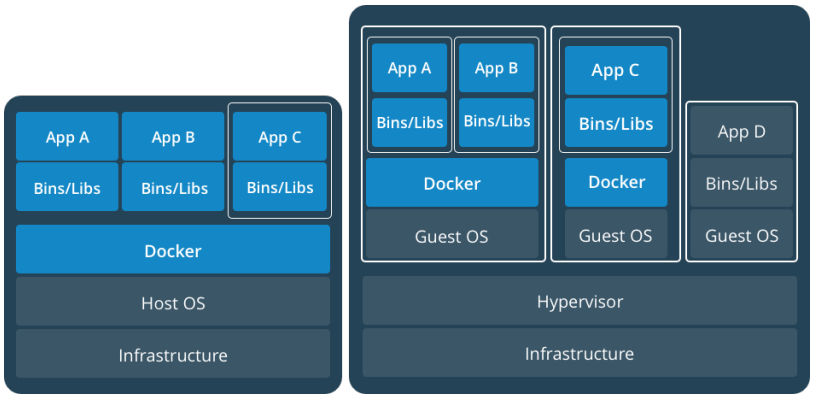

FIGURA 2.13: La diferencia entre contenedores nativas y dentro de una máquina virtual. (Docker Inc., 2017b)

Entre las ventajas de contenedores está que permiten tener ambientes aislados livianos para prevenir conflictos de dependencias y estandarizar versiones utilizadas en ambientes de desarrollo y producción mientras que al mismo tiempo requieren mucho menos recursos que una máquina virtual. Las mismas se pueden levantar casi instantáneamente a diferencia de máquinas virtuales que requieren tiempo para arrancarse. Como se puede ver en la figura (contenedores-vms), también se puede tener un motor de contenedores dentro de una máquina virtual, el cual ofrece beneficios como mayor aislamiento (seguridad) y flexibilidad al momento de migrar aplicaciones entre servidores (Docker Inc., 2017b).

La estandarización que ofrece la contenerización ayuda abstraer sistemas operativos que están en los servidores (permitiendo mayor flexibilidad de despliegue), permite mayor escalabilidad si la aplicación está diseñada para eso, es fácil de versionar dado que los archivos de configuración que describen los contenedores muchas veces son de texto plano y es muy orientado a arquitecturas compuestas por servicios como SOA o las arquitecturas de Microservicios debidos a que estas ya tienen componentes donde es fácil saber dónde poner la frontera de cada contenedor (Ellingwood, 2015).

Para tener seguridad adicional, también se puede ocupar contendores dentro de máquinas virtuales aisladas como se puede observar en la figura 2.13 (Docker Inc., 2017b).

Docker Docker es la plataforma más popular hoy en día para contenedores. Se puede usar para eliminar problemas de dependencias entre equipos de desarrolladores, para aumentar la capacidad computacional de equipos y también para desplegar de una forma más ágil, rápida y segura. Con contenedores de Docker, toda la complejidad está dentro del contenedor, facilitando su compilación, y la forma en que son compartidos y ejecutados, lo que hace fácil que cualquier persona que tenga los archivos de configuración necesarios pueda levantar la aplicación en minutos en lugar de en horas (Docker Inc., 2017c).

La arquitectura de Docker permite funcionar con cualquier tecnología. La estandarización y facilidad que ofrece permite mejor colaboración entre miembros de un equipo. Por defecto Docker se trata de ser lo más seguro, flexible y extensible posible mientras que al mismo tiempo no necesitar cambios para que un proveedor de software no necesita realizar cambios ni casarse con la tecnología (Docker Inc., 2017a).

Frente la percepcion comun de que Docker es insegura, McKendrick y Gallagher ofrecen buenas prácticas para el uso de Docker en ambientes de producción como:

- Utilizar solo contenedores confiables que se pueda entender sus contenidos en base a un Dockerfile
- Ejecutar el mismo contenedor con y sin un sistema ficheros y despues realizar una comparación entre los dos para auditar el comportamiento.
- Auditorías de Contenedores con:
  - Docker Bench Security
  - El uso de servicios de terceros:
    - Quay
    - Clair
- Una aplicación en cada contenedor

- Instalaciones minimalistas de lo mínimo necesario.
- Control de quién puede enviar comandos al servicio de Docker.
- Utilizar la version mas actual de Docker
- Utilizar restriciones de recursos.
- Restringir el nivel de permisos con que se ejecuta el contenedor.
- Reducir la superficie de ataque dentro del contenedor al no montar archivos críticos del sistema host dentro del contenedor.

Aunque los autores son muy claros en especificar que Docker no es un sistema de virtualizacion dan estos consejos, listados previamente, entre otros para tener mayor seguridad (McKendrick y Gallagher, 2017).

Tecnologías de gestión de virtualización

Hoy en día los sistemas son cada vez más grandes, y para que sean escalabes en tiempo real, seguros y a su vez tolerantes a fallos, se utiliza nubes tanto públicas como privadas. Tendencias de nube como IaaS, PaaS y SaaS ayudaron optimizar el despliegue de sistemas con la automatización de virtualización. Pero las tecnologías de hoy toman eso y lo llevan mas allá para integrar culturas como el famoso DevOps con nube para llegar a lo que es orquestación de nube o en las palabras de Suchitra Venugopal, "automatizar la automatización" de la generación anterior de nube. Sistemas de orquestración toman como entradas el estado actual y estado deseado en cuanto sistemas e infrastructura y ellos realizan todo lo posible para establecer y mantener el estado deseada. Esto incluye mantener alta la disponabilidad de servicios, auto escalarlos en base a su utilización, recuperación después de fallos entre otras operaciones (Venugopal, 2016).

OpenStack es una herramienta de codigo abierto para gestión de nubes. Tiene una comunidad grande que lo desarrolla en conjunto para que sea una plataforma robusta y segura para todos sus usuarios. Su arquitectura se forma por componentes o servicios, de los cuales cualquier persona pude agregar adicionales. Los nueve servicios que forman el nucleo del proyecto son:

Nova , que gestiona instancias de computación como máquinas virtuales.

Swift , que gestiona almacenamiento como una abstracción donde el usuario final no le importa como esta gestionado el almacenamiento de verdad.

Cinder , que gestiona almacenamiento como el model clásico de bloques en un disco para aplicaciones que necesitan mayor rendimiento de disco.

Neutron , que gestiona redes.

Horizon , que provee toda una nube de OpenStack con su interfaz de usuario web (WUI).

Keystone , que es un servicio de identificación/autenticación para todos los servicios de un nube de OpenStack.

Glance , que es un servicio de imágenes que pueden servir como plantillas para la creación de nuevas máquinas virtuales.

Ceilometer , que es un servicio de monitoreo que recolecta datos del consumo de servicios por usuarios para generar reportes y facturas de consumo.

Heat , es un servicio de orquestración de nube que permite a usuarios definir su despliegue deseado y después el servicio negocia con los demás servicios de la nube para establecer y mantener dicha configuración.

(*What is OpenStack?*)

A traves del trabajo de la comunidad y proyectos dentro de OpenStack como Nova, el mismo suporta una gran variedad de hipervisores como Xen, KVM y Docker entre otros y por lo tanto, aplicaciones que utilizan el API de OpenStack pueden independizarse de tecnologias de virtualizacion que existen por detras (OpenStack Community, 2018) (*Nova Docker*).

Kubernetes es una herramienta de código abierto para gestión de contenedores que fue el resultado de 15 años de experiencia de mantener sistemas de producción en Google. Esta diseñado para permitir escalar de forma horizontal sin restricciones para orquestrar nubes de contenedores a cualquier escala. Soporta autorecuperación, escala de servicios de forma automatizada en base a parámetros como uso de CPU o bajo petición de un operador, lleva DNS interno para que los servicios puedan encontrarse entre ellos a cualquier escala y balanceo de carga para los mismos servicios. Permite una gran número de tipos de backend de almacenamiento, en adicion a control de versiones de despliegue y despliegues en cascada para nunca dejar usuarios sin servicio mientras que se esta actualizando un servicio de versión o en caso de que una versión actual no funcione es fácil volver a alguna version anterior. Pero no solo es para ejecución de servicios, si no tambien soporta el trabajo por lotes (The Kubernetes Authors, 2018c).

Protocolo de Túnel

Un túnel es una técnica para permitir el acceso remoto a recursos en una red a los cuales normalmente no se tendría acceso desde afuera. Se puede realizar esto en la capa 2 de redes con los protocolos como L2TP, PPTP y L2F quienes buscan encapsular paquetes previo a su viaje entre redes y desencapsularlos en ambos lados del túnel. Puede ser una solución económica para todos los involucrados ya que permite montar varios VPNs en la misma infraestructura física (*Tunneling*). Los protocolos de túnel saben transmitir dos tipos de mensajes, aquellos paquetes que encapsulan datos y otros paquetes auxiliares que definen las reglas de reenvío (*What is a Tunneling Protocol?*). Por esta razón y también la tendencia de estos protocolos de proteger los datos que transmiten con encriptación, resulta en una velocidad de transferencia de datos menor que un simple envío directo. Algunos otros protocolos de encapsulación buscan reemplazar TCP y UDP en la capa 4 pero su funcionalidad de transportar paquetes entre redes, que de ninguna otra forma serían conectados, es igual y forma un tipo de VPN. Los protocolos SSH y IPSec también tienen esas funcionalidades. Así se puede lograr reenviar puertos para utilizar servicios internos como portales internos de una institución o convertir una máquina con Linux/UNIX en un servidor gráfico para "terminales tontos" (Schluting, 2006) como los días clásicas de la infancia de UNIX (Geerling, 2014). Aunque su poder y seguridad la ha hecho una herramienta común entre profesionales que trabajan con redes (Schluting, 2006), también por su misma naturaleza se puede ocultar tráfico y por lo tanto ahora es común que malware y actores maliciosos utilizan esos mismos protocolos

para esconder su actividad y realizar actos malvados con un menor riesgo de ser detectados en tiempo real. Por lo tanto, para usuarios normales, se recomienda limitar el uso de estos protocolos (*What is a Tunneling Protocol?*).

Criptografía

Criptografía es la ciencia que protege datos comunicados que han sido utilizado por miles de años para comunicarse de tal forma que el emisor y receptor pueden enviar y recibir mensajes pero personas intermedias no pueden entender los mensajes en tránsito. Tiene aplicaciones militares, financieros y también en el ámbito tecnológico, sobre todo para el internet, entre otras aplicaciones para los cuales ha sido usado a lo largo de la historia. Cifrar es el acto de ocultar datos y decifrar es la acción opuesta para devolverlos a su estado original (Cruise, 2012b).

Criptografía Asimétrica Con la introducción de redes y la necesidad de encriptar comunicaciones entre personas quienes nunca se ha habían reunido para llegar a un acuerdo de que usar de llave criptográfica para comunicaciones, se necesitaba una manera de llegar a este acuerdo sin que ningún agente intermedio también sepa la llave criptográfica resultante del acuerdo y por lo tanto se introdujo la necesidad de la criptografía asimétrica (Cruise, 2012a). Para cumplir con este requisito, el receptor genera una llave privada y llave pública, los cuales son operaciones inversas y pública su llave pública para todos que deseen comunicarse con el mismo. Ahora quienes quieren comunicar con el dueño de la par de llaves ocupan su llave pública para cifrar el mensaje y lo envían. Solo con la llave privada puede descifrar el mensaje así como el dueño único que tiene esta llave, sólo él puede leer los mensajes enviados (Cruise, 2012a).

SSH

En la infancia de la computación, solo habían computadores centralizados en instituciones grandes. Aún no existía el internet y las redes estaban en su infancia. Poco a poco se introdujo la idea del terminal tonto y con ello el acceso por varios usuarios a estas mismas computadoras centralizados sobre un LAN institucional. De aquí proviene el concepto de control remoto y en aquella época de redes bien controlados y cableados, no hubo mucho riesgos de seguridad entonces por lo tanto el protocolo de esos tiempos fue Telnet, donde todos los datos, incluyendo usuarios y contraseñas son enviados en texto plano. Con el tiempo se volvió más económico y ubicó a la tecnología de tal forma que, junto al internet, se estaba empezando a usar Telnet en redes de desconocidos y generando situaciones potencialmente maduras para ataques clásicos de análisis o incluso modificación de paquetes en tránsito. Fue frente esta realidad que un Finlandés diseñó e implementó el protocolo SSH (Geerling, 2014). OpenSSH es una implementación abierta y libre (*OpenSSH Features*) del protocolo SSH, que como se define el protocolo, cifrar su tráfico para proteger contra espionaje, robo de conexión y otros ataques que afectan otros protocolos de una naturaleza similar. El proyecto OpenSSH, para ofrecer la mayor funcionalidad posible tiene integrado funcionalidades de túneles seguros, varios métodos de autenticación, y opciones avanzadas de configuración. Adicional a esto ofrece una variedad de herramientas para la gestión de sus funcionalidades con clientes como los comandos ssh, scp y sftp, gestión de llaves con comandos como ssh-add, ssh-keysign, ssh-keyscan, y ssh-keygen, en adición a servicios en segundo plano como

sshd, sftp-server y ssh-agent (*OpenSSH*). Hoy en día se puede utilizar SSH para cifrar conexiones en redes que de ninguna otra forma serían cifradas, reenviar tráfico de un servidor gráfico X11 (para tener varios terminales gráficos en una sola máquina servidor (Schluting, 2006)), reenvío de puertos sobre un canal seguro, fuerte mecanismos de autenticación como claves de único uso y llaves de criptografía asimétrica, reenvío de agentes de autenticación, interoperabilidad con otras versiones y implementaciones del protocolo SSH, y la opción de activar compresión y descompresión de datos en tiempo real para optimizar el consumo de recursos en red (*OpenSSH Features*) (OpenBSD, 2016).

Túnel de SSH SSH es un protocolo de túnel muy popular (*What is a Tunneling Protocol?*) ya que ofrece mucha flexibilidad (*OpenSSH*) y alta seguridad desde la autenticación para iniciar la conexión como para los datos transferidos en dentro de la conexión (*OpenSSH Features*). Su facilidad de uso permite con rapidez levantar mini VPNs de un puerto para acceder a puertos de otros equipos que normalmente no estan disponibles, por ejemplo acceder servicios que solo estan disponibles en un intranet (Schluting, 2006).

SSL

SSL es un estándar tecnológico para cifrar tráfico (SSL.com Corp, 2016) (*What is an SSL Certificate?*) de protocolos que normalmente no son cifrados por ejemplo HTTP, SMTP e IMAP. Esto se hace con el fin de proteger datos sensibles que se transfieren (en adición a asegurar su integridad (SSL.com Corp, 2016)) entre un cliente y servidor, los cuales normalmente se transfieren en texto plano y podrían ser leídos por cualquier agente intermedio. El proceso de comunicación se puede encontrar en la figura 2.14. Para iniciarse la conexión cifrada, primero el cliente pide que el servidor se identifique (1) con la llave pública que forma parte de su certificado (2). Una vez que lo tiene, el cliente procede a validar el certificado para ver si confía en su origen (a través de firmas digitales por autoridades centrales) y en caso de que haya confianza, se procede a generar una llave criptográfica para la sesión, lo cual se lo transmite cifrada al servidor con la llave pública del mismo (3). El servidor descifra la clave de sesión con la llave privada de su certificado, avisa al cliente que la transacción ha sido exitosa (4) y ambos proceden a transferir los datos, cifrados con la llave de sesión que tanto cliente como servidor comparten (5) (*What is an SSL certificate?*).

Para ayudar con el proceso de validación, el certificado SSL contiene detalles como la llave pública del servidor, el nombre de dominio, nombre de empresa, dirección, ciudad, estado o provincia y país. Es firmado por un autoridad de certificados para dar confianza a los clientes que revisarán el certificado (SSL.com Corp, 2016). Antes de firmar un certificado, una autoridad de certificados primero valida la identidad del servidor que ha realizado la petición para que el certificado luego pueda confirmar a los clientes que es el mismo servidor (*What is an SSL Certificate?*).

TLS Cuando llegó la versión 4.0 de SSL, se cambió de nombre a TLS versión 1.0 (*What is an SSL certificate?*) para señalar un fuerte cambio de seguridad en cómo se manejan los casos de uso, ya que estos pueden causar fallas de seguridad en antiguas versiones de SSL como POODLE, DROWN y BEAST que dejan la seguridad que promete SSL inútil y peligrosa (ya que las mismas prometen seguridad falsa) (Kangas, 2016) (GlobalSign, 2016).

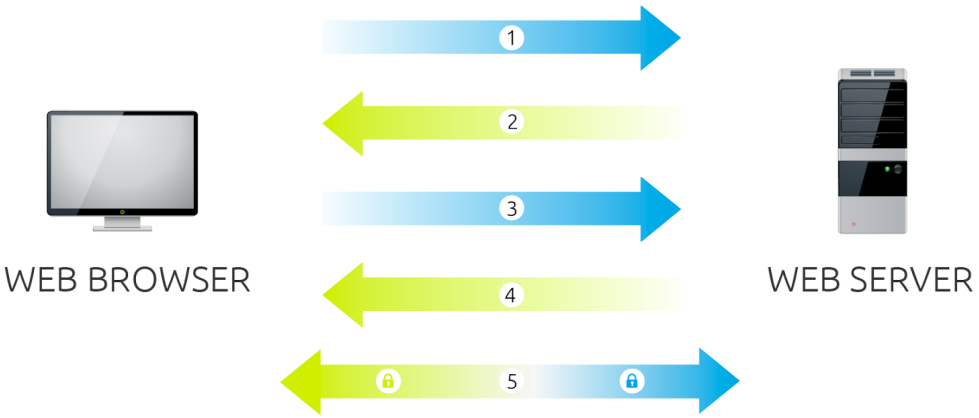

FIGURA 2.14: El proceso para comunicación con el protocolo SSL.
(*What is an SSL certificate?*)

2.1.3. Tecnologías de la Web

El cuadro 2.2 presenta conceptos de varias tecnologías que existen en la web hoy en dia.

CUADRO 2.2: Algunos de las tecnologías de la web.

Concepto	Definición
URI	Una URI identifica de forma única un recurso que existe en la red como en el Internet (*URI - Uniform Resource Identifier*). Como se define en el RFC 3986, debe consistir de forma genérica en un: [protocolo de acceso][usuario][clave][dominio][ruta de recurso][parámetros de acceso al recurso] Donde usuario, clave son opcionales y la ruta del recurso y sus parámetros también pueden ser opcionales o innecesarios en casos de protocolos que no requieren el mismo (Berners-Lee, Fielding y Masinter, 2005).
HTTP	HTTP es el protocolo con el cual se intercambia información como peticiones y datos con páginas web en el World Wide Web. Usa un esquema de códigos de error para comunicar tipos de sucesos que pueden ocurrir con el uso del protocolo (ComputerHope, 2017). Dentro del protocolo HTTP, se usa verbos que son comandos que dicen que tipo de operación quiere hacer un cliente con relación a un servidor web. HTTP es conocido como un protocolo sin estados ya que dentro del protocolo se desconoce acciones anteriores que se ha tomado en la interacción cliente/servidor (*HTTP - HyperText Transfer Protocol*). Algunas tecnologías adicionales de la web se muestran en la tabla (tecnologias-web).
HTTPS	HTTPS utiliza el estándar SSL y ahora TLS para cifrar conexiones HTTP entre un cliente y servidor (ComputerHope, 2017).
API REST	Un API REST es un estilo arquitectónico donde se construye encima del protocolo HTTP para orientar el mismo protocolo a manejar estados. Este mismo estilo arquitectónico se orienta a escalabilidad y sistemas de software grandes (*REST - Representational State Transfer*).
HTML	HTML es el lenguaje de autoría para páginas en la web. Usa una combinación de cientos de etiquetas predefinidas para definir estructura, formato y posición de contenido (*HTML - HyperText Markup Language*).
CSS	CSS es un lenguaje que permite definir estilos de cómo se presentará el HTML. Se dice que son hojas de estilo en cascada porque se puede aplicar varios de ellos "en cascada" a una página de HTML (*CSS - Cascading Style Sheets*).
JavaScript	JavaScript es un lenguaje de programación que tiene un fin de permitir que páginas de HTML sean dinámicas y pueden interactuar con el usuario en tiempo real (*JavaScript*)..

2.2. Trabajos Relacionados

Dentro de la temática de este trabajo de titulación es importante entender otros trabajos relacionados que reqieren ser analizados para entender investigaciones previas y de esta forma aprender de ellos. A continuación se mencionan los siguientes:

2.2.1. GitEduERP

En un trabajo reciente del autor con colegas (Marcelo Bravo y Priscila Vargas), en la Universidad Técnica Particular de Loja, y frente el problema de ofrecer un ambiente de programación a estudiantes en línea, se realizó un editor de código en línea con sistema de permisos y la capacidad de compartir el mismo entre usuarios a través de un backend con Django y utilizando una librería ACE liberado por Cloud9 IDE que guarda el código editado en una instancia de GitLab CE. Además ofrece un chat en línea con una liberia TogetherJS. El resultado es un prototipo que guarda snippets en GitLab y permite compartir código de una manera muy primitiva entre usuarios. Se llevó el codigo editado en un sistema de control de versiones interno, usando MongoDB. Entre los problemas que se dieron fue el corto tiempo para el desasrrollo del proyecto y el alto nivel de complejidad para el mismo lo cual no permitío terminar con el alcance propuesto (Bravo, Earley y Vargas, 2017).

2.2.2. Sistema de Encuestas Online

En la Universidad Técnica Particular de Loja, para mejorar temas de business analytics, dos autores, Diana Ortega y Juan Carlos Sánchez, trataron de diseñar procesos de negocio para realizar recolección de datos en tiempo real a través de encuestas y analizar los mismos con el fin de ayudar a la toma de decisiones estratégicas de negocio en tiempo real. Plantearon soluciones con el fin de optimizar tiempos y recursos a través del uso de soluciones tecnológicas (Salinas y Castillo, 2007). Aunque no se trató de enseñar ni de programar en línea, se ha considerado importante este trabajo, debido a que llevaron a cabo temas de business analytics, y son una de las finalidades de este trabajo de titulación.

2.2.3. Metodología de Enseñanza con la Web 2.0

En la Universidad Técnica Particular de Loja, Mauricio Castillo, buscó utilizar la manera en que la lectura-escritura en la Web 2.0 podría crear cursos interactivos con estudiantes online como base de una nueva metodología de enseñanza. Destaca estos temas desde el punto de vista de ingeniería en sistemas para proporcionar soluciones netamente técnicas y estratégicas para dar el mejor aporte posible a quienes deseen implementar un sistema de este tipo. Estos conocimientos fueron aplicados en una materia "Diseño de Paginas Web dinámicas con PHP" en la modalidad abierta y a distancia de la misma universidad, para comprobar la validez de la metodología desarrollada. Se encontró que al aplicar la metodología de integrar tecnologías de la web 2.0 y aumentar la interacción estudiantil dentro del curso, se pudo aumentar con un grado significativo el porcentaje de estudiantes que aprobaron la materia (Torres, 2008).

2.2.4. Xen Web-based Terminal for Learning

Los autores, Abdullah Almurayh y Sudhanshu Semwal de IAENG, propusieron e implementaron una arquitectura de cliente-servidor para la distribución de recursos educativos que coadyuve al proceso enseñanza a los estudiantes sobre Linux, programación orientada a la nube y gestión de nubes/servidores. Su aplicación web ofrece terminales SSH donde cada estudiante y docente disponía de una máquina virtual de tal forma que tenían su propio ambiente con permisos de superusuario pero en ambientes aislados de la infraestructura real y de los demás usuarios, lo que permitía brindar una solución flexible y a su vez segura.

Como hipervisor ocuparon un sistema de Xen que controlan remotamente con su servidor de aplicación (servidor web). El enfoque del trabajo era garantizar seguridad y al mismo tiempo generar ambientes con permisos completos (de root) donde los estudiantes podrían experimentar y aprender sin restricciones mientras que al mismo tiempo ser aislados y que la institucion educativa podria prevenir, sin mayor preocupacion, el mal uso o de alguna forma maliciosa estos recursos dedicados al aprendizaje de los estudiantes.

La parte de seguridad se implementó con una validación de comandos y eliminación de caracteres especiales que se podrían usar para escapar de las restricciones establecidas previamente. De esta forma se logró realizar un sistema útil para administradores, docentes y estudiantes dentro del ámbito educativo. El sistema solo se implemento un comando, xe, para la gestión del hipervisor Xen por atrás. Los autores dejaron como trabajo futuro mejorar la seguridad debido a que actualmente no esta completo en esta parte (Almurayh y Semwal, 2014).

2.3. Sistemas Similares

De la misma forma y a la vez que se analiza los trabajos relacionados, es importante conocer también el contexto actual del mercado para ver las alternativas que ofrece la competencia y saber si ya hay alguna solución que sea de mayor beneficio a la universidad que el mismo sistema que se plantea (y por tal razón sería más factible implementar dicha solución en lugar de desarrollar algo nuevo). Se presenta a continuación las iniciativas investigadas:

2.3.1. Repl.it

Repl.it es una plataforma en línea, desarrollado por Neoreason Inc., para escribir y probar código en tiempo real como un IDE en línea. Su producto tiene un componente educativo que permite integración con otros sistemas de aprendizaje, organización por aulas, texto que guía las tareas, calificación automática de tareas a través de pruebas unitarias entre otras características que mejoran la interacción entre docentes y sus alumnos en su aprendizaje de programación. Actualmente usa ”máquinas completas de linux” para soportar ”más de 30 lenguajes” de una manera ”fiable y segura”. Como sistema completo ofrece muchas de las funcionalidades que se esperan de un sistema de este tipo, como integración LTI, ejecucion de codigo en linea, soporte para muchos lenguajes, calificación basada en pruebas unitarias, entre otros. Pero tanta funcionalidad se relaciona con un precio alto que reduce la accesibilidad al mismo para las instituciones educativas (Repl.it y Neoreason, Inc., 2017).

2.3.2. io.livecode.ch

Io.livecode.ch, desarrollado por Nada Amin, es una plataforma prototipo que convierte repositorios públicos de GitHub en tutoriales interactivos y documentados de programación. Permite la ejecución de código en contenedores de docker en tiempo real. El mismo utiliza máquinas virtuales alojadas en DigitalOcean y permite la extensión por terceras personas con más tutoriales (Amin, 2017). El problema del mismo es que no lleva estados, como ambiente de programación en línea estática, solo lleva los datos del momento sin persistirlos y por lo tanto no ofrece factibilidad

dentro del alcance del proyecto actual. El mismo hecho de llevar Docker como su tecnología de virtualización, según Amin, también baja la seguridad de la aplicación y genera graves vectores de ataque dentro del mismo.

2.3.3. Cloud9 IDE

Cloud9 IDE, desarrollado por una empresa incorporada con el mismo nombre, ofrece un espacio de trabajo personal rápido y escalable (pero administrado por la misma empresa) basado en contenedores de Ubuntu encima de Docker para cada usuario, con soporte para 40 lenguajes de programación. Permite ver aplicaciones web en tiempo real con una variedad de navegadores y sistemas operativos. También permite conectarse a servidores privados del usuario por SSH para utilizar estos en lugar de los servidores que ellos proveen. También tiene la capacidad de compartir código entre varios usuarios bajo permisos de lectura o lectura y escritura, permitiendo que los mismos editen en tiempo real (visible a todos) y que puedan comunicarse a través de chat. Se puede compartir el mismo código (una vista previa de ello o su versión completa) públicamente con usuarios que no pertenecen a la plataforma. Todo esto con el fin de reemplazar todo el ambiente de desarrollo local con un entorno completamente en la nube; ofrece un terminal, editores avanzados de código, división de pantalla, un debugger, temas, personalización de atajos, comandos communes, modo vim y modo sublime para el editor y un editor de imágenes. Cloud9 IDE está orientado más a profesionales y por lo tanto aún no ofrece tan buena integración con sistemas educativos (*Your development environment, in the cloud*).

2.3.4. GitLab

GitLab, desarrollado por GitLab Inc. con un conjunto de miembros de una comunidad de software abierto, es una adición a ser un servidor de Git, y por lo tanto permite llevar un control de versiones de los datos que aloja, también ofrece otras características asociados con entornos profesionales de desarrollo como seguimiento de incidentes, revisión de código, un IDE para editar código en línea, la capacidad de tener wikis asociado con proyectos, un sistema de integración continua para probar, compilar y desplegar código en una variedad de ambientes. Su versión de pago también soporta el consumo de un servidor de LDAP para autenticar usuarios, hooks de Git para tomar acciones personalizadas en respuesta a eventos de Git y capacidades para auditoria por parte de administradores (Hughes y col., 2017). Además, GitLab puede importar proyectos de otras plataformas y servidores de Git a través de una URI, gestión de snippets, ramas protegidas, un API que permite controlar al servidor de GitLab y un registro de contenedores de Docker asociados con cada proyecto. Como su enfoque principal es guardar código, no ofrece características fuertes para el desarrollo ni ejecución en línea, aunque si soporta otros plataformas de integración contínua (Pipinellis y col., 2017).

2.3.5. OverLeaf

Overleaf es una plataforma en línea, desarrollado por Writelatex Limited en Londres del Reino Unido, para editar y publicar de forma colaborativa documentos de LaTeX. Permite editar LaTeX directamente o usar un editor WYSIWG para quienes no conozcan bien el sistema TeX. La plataforma compila el documento de LaTeX en tiempo real a lo que se lo va editando para disponer de una vista previa con los últimos

cambios y así identificar los errores que se generan al compilar el documento. Como es una plataforma en línea, la edición de un documento se lo puede relizar entre varias personas al mismo tiempo desde distintas dispositivos. El sistema que respalda los documentos es un motor completo de LaTeX/TeX, permite una gran variedad de tipos de documentos y contenido en ellos (*Collaborative Writing and Publishing*). Overleaf se integra con Git ya que dispone de un servidor interno de Git para interactuar con los proyectos a través de este sistema de control de versiones (Lees-Miller, 2015). Obviamente aunque es un plataforma muy potente está completamente especializada con el fin de realizar documentos de LaTeX.

2.3.6. Google Drive

Google Drive como servicio de la misma empresa Google ofrece 15 Gigabytes de almacenamiento gratis para guardar cualquier tipo de archivos, los mismos que pueden ser compartidos con distintas personas para facilitar la colaboración entre estas. Entre sus funcionalidades esta el editar documentos, hojas de cálculo, presentaciones, formularios (para hacer encuestas), dibujos en línea. Es una API que permite mayor expansión por terceros. También controla las versiones de forma automática para que se pueda volver a revisar versiones anteriores e identificar quienes han introducido cambios y cuales han sido. Además en caso de requerir acceder a ciertos archivos sin internet se puede indicar a la plataforma que esté sincronizandose en segundo plano para poder tener disponible y actualizado los archivos cada vez que se encuentre sin conexión. Está orientado completamente al desarrollo de documentos, como paquete de ofimática y por lo tanto ofrece poca utilidad para escribir código (*Explore the Storage Features of Drive*).

2.4. Discusión

A continuación, se recapitula los trabajos relacionados y sistemas similares previo al fase de análisis y diseño de una solución en el próximo capítulo.

2.4.1. Comparación de Trabajos Relacionados

En el cuadro 2.3 se ve las diferencias entre los trabajos relacionados y los campos del trabajo actual en donde se aplica cada uno de ellos. Tanto GitEduERP como Xen Web-based Terminal for Learning Virtualization permiten a estudiantes editar y persistir el código en línea a diferencia del Sistema de Encuestas Online donde solo se recolecta, analiza datos y la metodología de enseñanza online con la web 2.0 que no implementa la parte práctica de programación en línea. Para enseñar a estudiantes en línea solo dos trabajos relacionados se tratan de eso; Metodología de Enseñanza con la Web 2.0 que buscaba aplicar tecnologías como los blogs para aumentar la participación estudiantil dentro de las materias a distancia y con ello aumentar su aprendizaje, esto es medido a través de sus notas en una materia de programación, por otro lado con Xen Web-based Terminal for Learning Virtualization a diferencia del resto de trabajos relacionados, ocupó máquinas virtuales (de Xen) para brindar ambientes completos y no tan restringidos a estudiantes para permitir el aprendizaje libre en línea con aspectos de virtualización, programación y Linux.

CUADRO 2.3: Comparación de Trabajos Relacionados.

	Editar Código en Línea	Persistir Código en Línea	Recolección y Análisis de Datos para decisiones estratégicas en tiempo real	Metodología de Enseñanza Online	Ambientes Virtualizados
GitEduERP	x	x			
Sistema de Encuestas Online			x		
Metodología de Enseñanza con la Web 2.0				x	
Xen Web-based Terminal for Learning Virtualization	x	x		x	x

2.4.2. Comparación de Sistemas Similares

En el cuadro 2.4 se ve las características básicas que se considera para un sistema de este tipo. Dentro de esta comparación, se ve que Repl.it es la mejor alternativa debido a que soporta todas las características menos la compilación en tiempo real (que podría venir a ser una desventaja que empeora el rendimiento del sistema).

En el cuadro 2.5 se observa aspectos sociales de cada sistema, entre ellosÑ cómo funcionan las dinámicas de compartir código e interacción entre usuarios. Aqui Repl.it sale perdiendo por el pobre soporte para compartir entre varios usuarios. Los sistemas que son mejores en estos aspectos es por ejemplo GitLab, Overleaf y Google

CUADRO 2.4: Comparación de Características Generales entre Sistemas Similares.

	Escribir Código Online	Probar Código Online	Integración LTI	Sistema de Autocalificación	Compilación en Tiempo Real
Repl.it	Si	Si	Si	Si	No
io.livecode.ch	Si	Si	No	No	No, pero si permite ejeccución de código en linea
Cloud9 IDE	Si	Si	No	No	Si para ciertos lenguajes
GitLab	Si	No	No	No	No
Overleaf	Solo LaTeX	Solo LaTeX	No	No	Si, LaTeX
Google Drive	No, solo se lo considera como texto	No	No	No	No

CUADRO 2.5: Comparación de Características Sociales entre Sistemas Similares.

	Control Interno de Versiones	Integración de algún sistema de control de versiones	Sistema de Permisos	Sistema de Compartir	Compartir / Editar en Tiempo Real
Repl.it	No	No	Si, basado en docente / alumno	No	No
io.livecode.ch	No	No	No	Todo es Público	No
Cloud9 IDE	Si	No	Si, basado en usuarios	Si, basado en usuarios	Si
GitLab	Si	Si, Git	Si, basado en usuarios y grupos	Si, basado en usuarios y grupos	No
Overleaf	Si	Si, Git	Si, basado en usuarios	Si, basado en usuarios	Si
Google Drive	Si	No	Si, basado en usuarios	Si, basado en usuarios	Si

Drive sin embargo no son tan optimizados para escribir código en línea o a ejecutar el mismo.

En el cuadro 2.6, se compara las diferencias entre los aspectos más avanzados de cada uno de los sistemas similares. Aquí no hay ni ganadores ni perdedores. Repl.it ofrece una solución robusta pero con la falta de no tener acceso fuera de línea. io.livecode.ch ofrece un sistema básico de documentación y ejecución basado en HTML, pero no maneja estados y no ofrece interactividad en tiempo real.

Adicionalmente, Cloud9 tiene los mismos problemas de falta de usabilidad sin conexión y de no disponer de una solución robusta de documentación. GitLab y Overleaf ofrecen acceso a través de Git de sus contenidos, teniendo la posibilidad de trabajar sin conexión, pero Overleaf no es para código exactamente. GitLab ofrece el

CUADRO 2.6: Comparación de Características Avanzadas entre Sistemas Similares.

	Control Completo sobre ambiente de ejecución de código	Acceso Remoto al ambiente de Ejecución de Código	Sistema de Documentación	API sobre HTTP(S)	Acceso fuera de línea
Repl.it	Si para instalación de dependencias	No	Si, documentación del ejercicio	Si, pero está cerrado a nuevos clientes	No
io.livecode.ch	Si, script de Bash	No en tiempo real	Si, HTML	No mantiene estados	No
Cloud9 IDE	Si, Terminal	Si, SSH	No, fuera de documentos en el servidor	Si	No
GitLab	No	No	Si, Wikis con GitHub Markdown	Si	Si, si es que se ha bajado todo anteriormente con Git
Overleaf	No	No	No, todo el sistema es de documentación	Si	Si, si es que se ha bajado todo anteriormente con Git
Google Drive	No	No	No, todo el sistema es de documentación	Si	Si

mejor solución de documentación a través de sus Wikis, pero estos no se visualizan al mismo tiempo que el código. Y Google Drive ofrece la mejor solución para trabajar sin conexion, pero la plataforma es para documentos, no para código.

Al final, ninguno de los sistemas similares alcanza el nivel de funcionalidad que se requiere. Por ello, a continuacion en el siguiente capítulo es necesario analizar, diseñar e implementar una solución nueva que satisfaga las necesidades institucionales que actualmente se requiere.

Capítulo 3

Análisis y diseño

3.1. Análisis

La resolución de los problemas siempre debe empezar con un análisis acerca de su alcance y características. El análisis surge con las ideas de soluciones del problema para en base a ellas diseñar una resolución del mismo. Para la fase de análisis se ha realizado un Documento de Visión (Apéndice A) y una Especificación de Requerimientos (Apéndice B).

3.1.1. Visión

El resto del documento de visión se puede encontrar en el Apéndice A. Aquí se ha incluido la parte de mayor consideración.

Propósito

Ayudar a mejorar los métodos de enseñanza que ofrece la Universidad Técnica Particular de Loja en cuanto a la programación y uso de base de datos para las carreras de Sistemas y Electrónica.

Alcance

Se propone un sistema de editar código en línea que a su vez integra LMS externos (para autenticación y notas), un servidor de control de versiones externo (para la persistencia de código), y un servicio web de ejecución de código en línea de una forma segura, eficaz y eficiente (para dar un ambiente de ejecución y pruebas tanto para los usuarios del sistema como para calificar de una forma automática).

Oportunidad de Negocio

Con el avance continuo de la tecnología y su introducción en varios aspectos de la vida cotidiana, hay una necesidad creciente de ingenieros en sistemas e electrónica que puedan programar y entender el software funcional en adición a las bases de datos que están por detrás de estos mismos sistemas.

Por la razon antes mencionada, está de moda ofrecer plataformas en línea para la enseñanza dinámica de la programación. GitEDU espera ofrecer las mismas funcionalidades a un costo institucional menor a través de integración con sistemas existentes e innovación para proveer una mejor experiencia de usuario, tanto estudiantes como profesores para llevarse a cabo un mejor proceso de aprendizaje.

Demográfia del mercado

En el año 2011, la Universidad Técnica Particular de Loja contaba con aproximadamente 4000 estudiantes presenciales y 24000 estudiantes a distancia con una tendencia creciente (UTPL, 2017a). En la experiencia personal del autor, las carreras de Sistemas y Electrónica, por lo menos en la modalidad presencial, juntas representan aproximadamente un 10 % de todos los estudiantes en la universidad, lo cual daría aproximadamente un mínimo 2800 estudiantes afectados por un nuevo sistema. Según el directorio de docentes de la universidad, son 60 profesores en el departamento de Ciencias de la Computación y Electrónica (UTPL, 2017b). Con eso se puede estimar un mínimo de 2860 usuarios que el sistema propuesto podría llegar a afectar.

Es precisamente la parte de la población y de usuarios potenciales mencionados anteriormente, que están involcrados en el proceso de enseñar, evaluar y aprender habilidades de programación/consulta a bases de datos, que forman la base de usuarios de la aplicación.

3.1.2. Especificacion de Requerimientos

El resto del documento de especificación de requerimientos se puede encontrar en el Apéndice B. A continuación se ha incluido las partes mas importantes.

Propósito

El siguiente documento espera dar una descripción detallada de los requisitos del Sistema Git Education (GitEDU) con la finalidad de definir la intención de la misma, declarar de forma completa todos los componentes a ser desarrollados del sistema, y también explicar limitaciones del mismo además de sus interacciones e interfaces con otros sistemas. Se destina el siguiente documento para revisión por el equipo de asesores y como una referencia de alcance para el equipo de desarrollo.

Alcance

GitEDU es un sistema de integración para promover la programación estudiantil y el sistema educativo que soporta el mismo. El sistema final debe disponer de buena documentación y alta calidad de código para sostener su futuro desarrollo y mantenimiento por parte de una comunidad de profesionales y profesionales en formación.

Docentes de las carreras involucrados en la enseñanza de programación, tanto en la carrera de Sistemas Informáticos y Computación como en la carrera de Electrónica y Telecomunicaciones, actualmente pueden agergar en la plataforma, deberes, talleres, pruebas y exámenes, donde a través de los LMS institucionales pueden compartir los mismos con sus alumnos. Los alumnos pueden entrar a través de un enlace que su docente comparta dentro del LMS institucional y con un canal de comunicación LTI, la plataforma GitEDU les identifica y autentica sin interacción del usuario para que el estudiante pueda empezar directamente con el trabajo que tiene que realizar sin digitar sus credenciales. En el curso de su actividad, se va guardando su progreso periódicamente contra un servidor de control de versiones institucional (GitLab CE) y en cualquier momento el estudiante puede probar e interactuar con su código a través de un terminal virtual de Linux que se encuentra dentro de la misma interfaz. Una vez que termina su trabajo y desee enviarlo, se lo envía y el código escrito se auto califica contra un conjunto de pruebas unitarias ocultas que el docente

agregó a la actividad al crearlo. La nota que se genera se lo envía al LMS institucional a través del mismo canal LTI. En caso de que el docente decide no utilizar pruebas unitarias o calificar cada trabajo a mano o realizar una calificación híbrida entre las pruebas unitarias y la parte a mano, no se reflejarán las notas en el LMS hasta que el docente haya terminado de calificar el código en la plataforma de GitEDU.

Perspectiva de Producto

La plataforma GitEDU se compone de dos subsistemas: el subsistema de editar código y el subsistema de ejecutar código. El subsistema de editar código consiste en la integración de autenticación y notas por el lado del LMS, el uso de persistencia del código por el servidor de control de versiones por otro lado, el consumo de un servicio de ejecución de código por un tercer lado y un editor de código en línea. El subsistema de ejecutar código consiste en esperar llamadas del subsistema anterior, validar la autenticidad, establecer permisos, limitaciones y parámetros de ejecución, gestionar recursos para ejecución, la ejecución interactiva y devolución de resultados al otro subsistema.

Debido a la necesidad de mantener la seguridad y prevenir la ejecución indebida de código, el subsistema debe ser aislado de todos los usuarios y solo accedido desde el otro subsistema a través de un API que permite su interoperabilidad. La solución para la ejecución de código debe ser liviana debido al gran número de usuarios concurrentes que se pueda tener a la vez.

Funcionalidades de Producto

Para entrar al subsistema de editar código en línea, se debe autenticar contra el LMS institucional. El editor de código en línea debe soportar una gran variedad de lenguajes y estar abierto a la agregación de más lenguajes en un futuro. La persistencia de código en un servidor de control de versiones debe ser transparente y no requerir ningún interacción del usuario. La interfaz para interactuar con el ambiente virtual debe ser simple y fácil de manejar sin mayor conocimiento de las tecnologías que le respaldan. Se debe poder acompañar al editor de código con documentación que guíe al estudiante en explicar o resolver el problema actual. La manera en que los docentes pueden generar nuevos ejercicios, su documentación y pruebas unitarias para la calificación debe ser intuitivo.

Características de usuario

Los tres tipos de usuarios que interactúan con el sistema son Docentes, Alumnos y Administradores. Cada tipo de usuario tiene sus propias necesidades y requerimientos.

Docentes, necesitan un sistema que sea fácil de manejar y que simplifique y automatice el proceso de enseñar y evaluar a los estudiantes. Eso significa tener una interfaz intuitiva para los docentes puedan escribir, documentar y generar pruebas unitarias para probar la funcionalidad de código que escriben los estudiantes para los ejercicios.

Alumnos, necesitan un sistema que les brinde una buena experiencia de aprendizaje y les proporcione todas las herramientas que requieren para programar en línea. al igual que sus profesores, los elumnos deben disponer de una interfaz intuitiva que les ayude a cumplir con las responsabilidades de estudiar, aprender y demostrar conocimientos en una cantidad de tiempo óptimo.

Administradores, necesitan un sistema que sea seguro y cuyos componentes sean fáciles de mantener y extender a lo largo del tiempo. Para ello se necesita tener una buena documentación del funcionamiento del sistema y todos sus componentes para tener la misma como referencia.

Limitaciones

Se requiere de hardware con alta capacidad de virtualización y conectividad al internet y la intranet institucional para poder brindar la mejor experiencia al usuario.

Suposiciones y Dependencias

Se supone que siempre habrá la conectividad necesaria para poder usar todos los componentes del sistema.

División de Requerimientos

En caso de que falte recursos de tiempo o algún otro recurso, se limitará el alcance del proyecto para enfocarse en los requerimientos de prioridad más alta y dejar los demás requerimientos para trabajos futuros.

3.2. Diseño

Previo a la implementación de cualquier sistema es importante analizar su funcionamiento y en base a esto realizar un diseño que provee de las funcionalidades que se requiere. Por lo tanto, a continuación se demuestra las fases del diseño realizado desde la definición de sistemas, subsistemas y módulos así como la interacción entre ellos, hasta la arquitectura interna y de despliegue.

3.2.1. Sistemas, Subsistemas y Módulos

Debido a la naturaleza del alcance de este trabajo de titulación, se considera que se debe dividir la plataforma en dos sistemas que sean capaces de interactuar entre sí. El primer sistema es el que ofrece el editor de código en línea, integración con los LMS institucionales y a su vez integración con un servidor de control de versiones. El segundo sistema solo se encarga de ejecución de código en línea a través de algún tipo de virtualización y aislamiento de procesos. De esta forma se puede aislar el sistema que está bajo mayor riesgo de ser atacado (el de ejecución de código arbitrariamente) del sistema con mayor riesgo de impacto negativo (el sistema de escribir código en línea, el cual también se encarga de las calificaciones). A continuación se presenta la interacción entre los sistemas y actores en el Diagrama de Contexto de Sistemas.

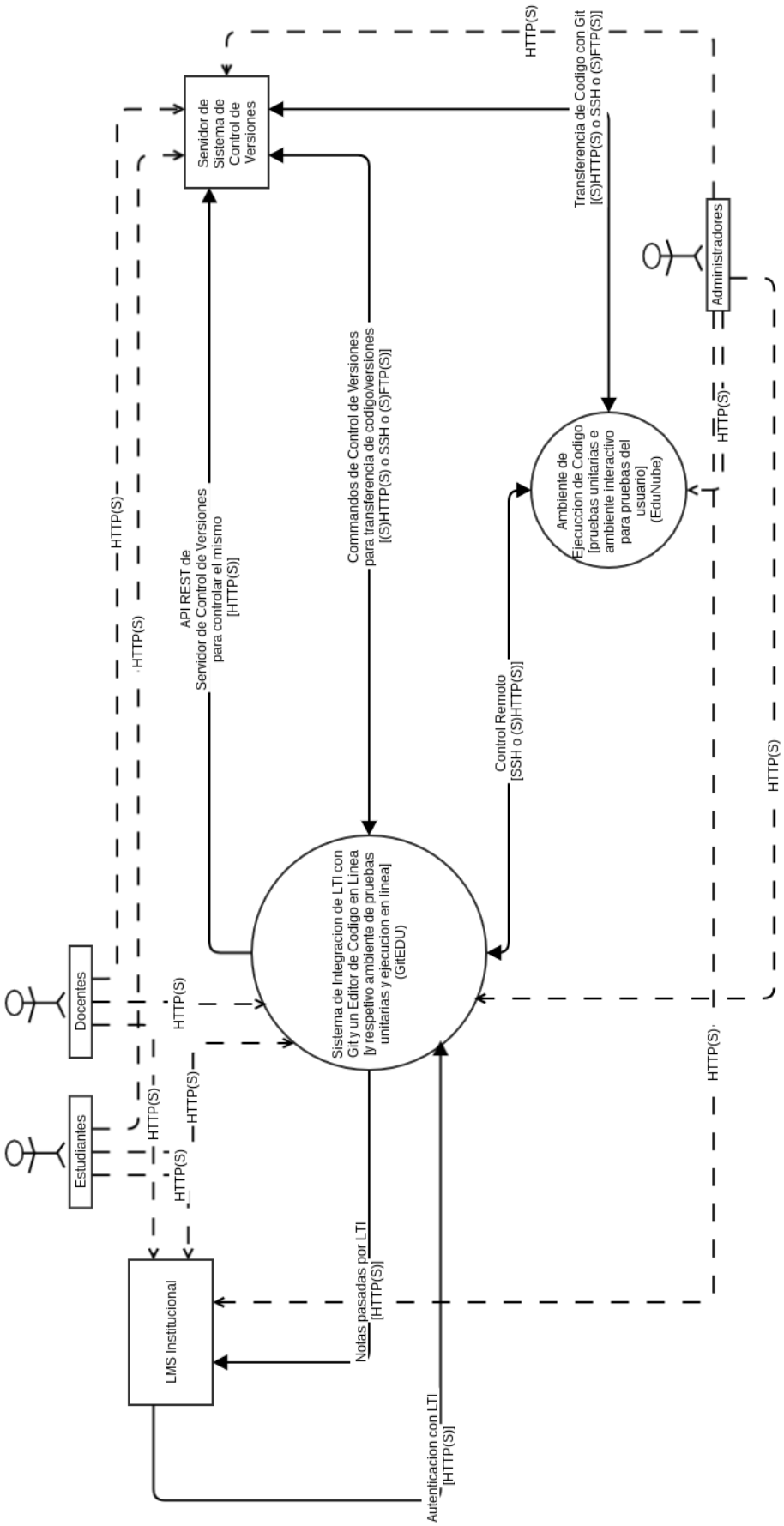

FIGURA 3.1: Diagrama de Contexto para GitEdu y EduNube.

El diagrama de contexto visualizado en la figura 3.1 demuestra la relación entre las tres clases de usuario, sistemas existentes, los sistemas que se plantea desarrollar y los protocolos de interacción entre sistemas distintos y también entre humanos y sistemas.

Los tres tipos de usuarios que se considera para el sistema son administradores, quienes administran y mantienen todo el sistema en su ambiente de despliegue; docentes, quienes generan contenido didáctico para sus estudiantes y califican los mismos motivo por el cual tienen acceso a los LMS, el sistema de editar codigo (que les permite generar nuevas tareas / exámenes / pruebas / talleres para sus estudiantes y ver el progreso y calificaciones de los mismos) y el sistema de control de versiones (donde puede ver y bajar código que escriben los estudiantes y ver a un nivel más profundo el historial y evolución del mismo), y estudiantes, quienes tienen acceso a los mismos tres sistemas que tienen los docentes. Los estudiantes acceden al LMS donde ven las tareas / exámenes / pruebas / talleres nuevos que hay y al abrirlos les lleva a identificase y autenticarse en el sistema de editar código en línea mediante LTI. Los estudiantes realizan sus tareas de código, mientras que los datos se respaldan de forma automática en un servidor de control de versiones independiente de tal forma que lo pueden revisar tanto los estudiantes involucrados como su docente a futuro. Aquel control adicional podría llegar a ser una evidencia importante para la resolución de conflictos entre estudiantes y/o docentes a futuro.

El sistema de editar codigo, que en este documento se conoce simplemente bajo el nombre GitEDU, controla remotamente por medio de comandos, y una mezcla de protocolos (SSH, HTTP sobre SSH [SHTTP] y HTTPS), dependiendo de cómo sea el caso, un sistema de ejecucion de código, conocido a partir de aquí en adelante como EduNube, señala acciones que debe tomar el segundo y/o datos que necesita sincronizar con el primer sistema. Al mismo tiempo, GitEDU comunica con el servidor de control de versiones mediante su API externa (una API REST) sobre HTTP o HTTPS para controlar el mismo y prepararle para flujos continuos de código editado que puede viajar por una variedad de protocolos como SHTTP, HTTPS, HTTP, SFTP, FTP o FTPS como requiera la situación y ambiente de despliegue. Cuando EduNube requiere de codigo de algún usuario para realizar una calificación o algún otro tipo de interacción con un usuario que requiere probar el mismo, primero se procede a descargar el código involucrado desde el servidor de control de versiones sobre uno de los protocolos mencionado previamente para el mismo. En base a la naturaleza de las operaciones que se está haciendo, comunica sus resultados con GitEDU para que pueda manejar las mismas de la forma adecuada. En el caso de notas, GitEDU se encarga de sincronizar las mismas con los LMS institucionales con los cuales está integrado mediante el protocolo LTI.

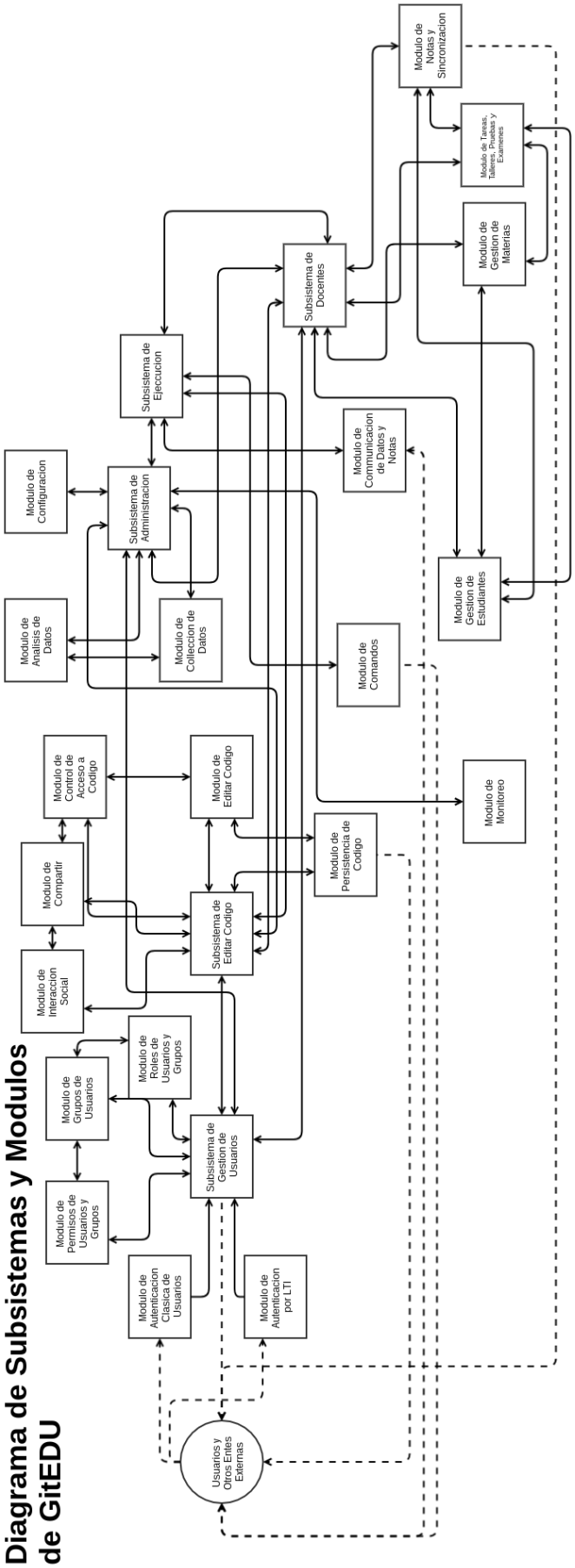

FIGURA 3.2: Diagrama de Subsistemas y Módulos de GitEdu.

GitEdu

La figura 3.2 presenta todos los componentes en forma de subsistemas y módulos que forman parte del diseño propuesto para el sistema GitEdu.

Subsistema de Gestión de Usuarios El subsistema de gestión de usuarios se encarga de temas de seguridad y acceso por usuarios a varios componentes de la aplicación. Para esto autentica usuarios mediante dos módulos (uno clásico y el otro mediante el protocolo LTI) y lleva un sistema de permisos orientado a usuarios, grupos y roles.

Módulo de Autenticación Clásica de Usuarios El módulo de autenticación clásica de usuarios permite que los usuarios finales puedan iniciar sesión mediante un usuario y contraseña.

Módulo de Autenticación por LTI El módulo de autenticación por LTI permite que los usuarios finales puedan iniciar su sesión mediante una existente en algún sistema externo que soporta LTI.

Módulo de Grupos de Usuarios El módulo de grupos de usuarios permite definir grupos específicos de usuarios cuya membresía les cambia el nivel y naturaleza del acceso que tengan en la aplicación.

Módulo de Permisos de Usuarios y Grupos El módulo de permisos de usuarios y grupos, define la naturaleza de acceso que tienen los usuarios autorizados y los grupos a los cuales pertenecen dentro de la aplicación.

Módulo de Roles de Usuarios y Grupos El módulo de roles de usuarios y grupos define las responsabilidades de varios grupos de usuarios y usuarios específicos, en base a los mismos les asigna permisos adecuados tanto para proteger la aplicación contra el uso y acceso indebido y al mismo tiempo permitir a los usuarios poder realizar sus actividades normalmente sin impedimentos.

Subsistema de Editar Código El subsistema de editar código forma el núcleo de la funcionalidad del sistema GitEdu. Esto lo realiza a través de funcionalidades de editar codigo, persistirlo y permitir la interacción adecuada y social entre usuarios.

Modulo de Editar Codigo El módulo de editar código en línea ofrece funcionalidades básicas de un editor de código (IDE) en línea.

Módulo de Persistencia de Código El módulo de persistencia de código, persiste el código versionado en servidores de control de versiones externos. Al momento de necesitar abrir un código anterior cualquiera, recupera el mismo de los servidores de control de versiones externos para que el usuario pueda seguir editando lo mismo.

Módulo de Control de Acceso de Código El módulo de control de acceso de código permite a los usuarios definir y modificar los permisos que tiene su código para controlar la naturaleza del acceso que tienen otros usuarios, grupos y roles.

Modulo de Compartir El módulo de compartir permite compartir el acceso al código con otros usuarios y grupos.

Módulo de Interacción Social El módulo de interacción social permite que los usuarios puedan interactuar en tiempo real mientras editan juntos el código.

Subsistema de Docentes El subsistema de docentes ofrece muchas de las funcionalidades que requieren los docentes para llevar exitosamente su materia de programación. Estas funcionalidades incluyen gestión de estudiantes, materias y notas. La gestión de materias y notas también involucra una gestión y calificación de talleres, deberes, pruebas y exámenes. Además por el hecho de que se genera nuevas notas y calificaciones a través de GitEdu, también se encarga de sincronizar notas con otros sistemas externos como los LMS a través de LTI.

Módulo de Gestión de Estudiantes El módulo de gestión de estudiantes permite que los docentes puedan ver notas de sus estudiantes. Además de que los estudiantes puedan revisar sus notas. Los administradores como docentes pueden asignar estudiantes a aulas o cambiar o eliminar asignaciones previas.

Módulo de Gestión de Materias El módulo de gestión de materias permite a los docentes y administradores ver estadísticas de una materia, creación de grupos de estudiantes dentro de una materia (por ejemplo para un proyecto grupal) tanto para un docente o para un estudiante, modificar grupos de estudiantes (por ejemplo eliminar un integrante), y asignar talleres, deberes, pruebas o exámenes a un estudiante, grupo o materia. Además dentro de la gestión de materias, docentes se puede establecer el peso de cada taller, deber, prueba y examen para en base a ello calcular los promedios de un estudiante.

Módulo de Talleres, Deberes, Pruebas y Exámenes El módulo de talleres, deberes, pruebas y exámenes permite a los docentes definir nuevos talleres, deberes, pruebas, exámenes, etc y los criterios de evaluación de los mismos. En ello se incluye las pruebas unitarias (si desea calificación automatizada) y documentación que explica lo que hay que hacer al estudiante.

Módulo de Notas y Sincronización El módulo de notas y sincronización gestiona las notas que mantiene GitEdu y se encarga de sincronizar cambios en las mismas. Aunque realiza este trabajo por detrás normalmente, también proporciona una interfaz gráfica para interacción con los docentes respectivos y adecuados. Además por el tema de seguridad e integridad de las notas, se lleva versionado y con un registro preciso de cuando fueron modificados y por quién.

Subsistema de Ejecución de Código El subsistema de ejecución de código se encarga de comunicar con el sistema externo EduNube con el fin de aprovechar las funcionalidades del mismo y proveer de funcionalidades de ejecución en línea, aplicación de pruebas unitarias y calificación automática de código sin exponer al sistema EduNube a usuarios finales. Para aumentar la seguridad que logra llevar EduNube, GitEdu se comunica con los mismos a través de canales seguros como HTTPS y SSH, utilizando métodos de autenticación orientados a una esquema de llaves públicas y privadas (criptografía asimétrica) para asegurarse de la identidad de ambos previo al envío de comandos y comunicación de datos y notas.

Módulo de Comandos El módulo de comandos se encarga de controlar el sistema de EduNube a través de su API externa para indicar al mismo cuando debe bajar, ejecutar o calificar algún código.

Módulo de Comunicación de Datos y Notas El módulo de comunicación de datos y notas permite la sincronización de los mismos cuando sea necesario entre EduNube y GitEdu.

Subsistema de Administración El subsistema de administración ayuda a los administradores de GitEdu a recolectar datos acerca del uso del mismo y analizar los datos, además de ser capaces de configurar y monitorear el rendimiento del sistema.

Módulo de Configuración El módulo de configuración permite la configuración en tiempo real del sistema GitEdu, por ejemplo conexiones con sistemas externos y gestión de usuarios.

Módulo de Monitoreo El módulo de monitoreo permite ver en tiempo real el consumo del sistema.

Módulo de Recolección de Datos El módulo de recolección de datos lleva un registro contínuo del uso del sistema para su revisión y análisis por parte de operadores humanos, para después realizar temas de auditoría como la toma de decisiones estratégicas.

Módulo de Análisis de Datos El módulo de análisis de datos proporciona las herramientas necesarias para que un operador humano pueda interpretar los mismos y sacar sus conclusiones.

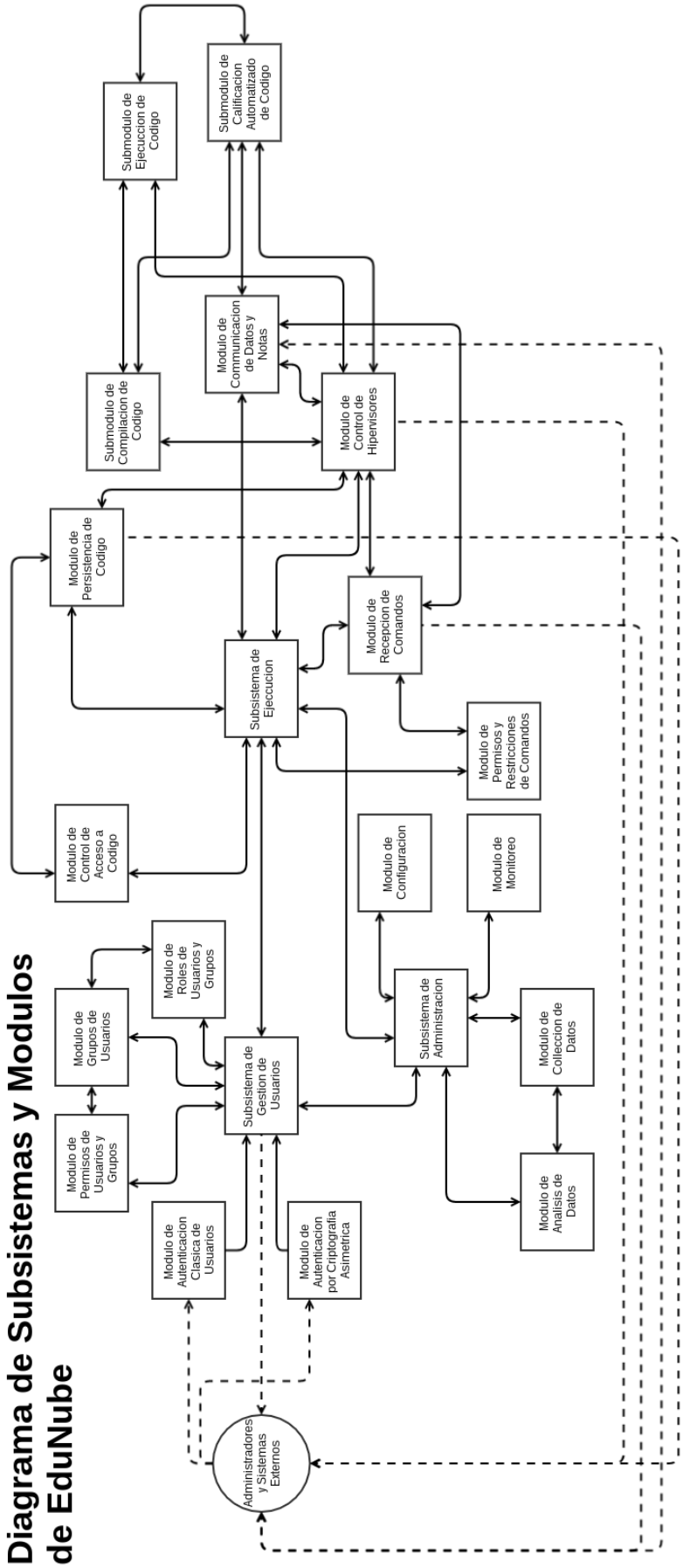

FIGURA 3.3: Diagrama de Subsistemas y Módulos de EduNube.

EduNube

La figura 3.3 presenta todos los componentes en forma de subsistemas y módulos que forman parte del diseño propuesto para el sistema EduNube.

Subsistema de Gestión de Usuarios El subsistema de gestión de usuarios de encarga de temas de seguridad y acceso indirecto por usuarios (a través de GitEdu) y acceso directo por parte de administradores a varios componentes de la aplicación. Para lo mismo autentica usuarios mediante dos módulos (uno clásico solo para administración del mismo y el otro mediante llaves públicas/privadas [criptografía asimétrica] para aplicaciones externas) y lleva un sistema de permisos orientado a usuarios, grupos y roles.

Módulo de Autenticación Clásica de Usuarios El módulo de autenticación clásica de usuarios permite que administradores puedan iniciar sesión mediante un usuario y contraseña.

Módulo de Autenticación por Criptografía Asimétrica El módulo de autenticación por criptografía asimétrica permite a los usuarios finales, a través de GitEdu iniciar sesión para la ejecución controlada y remota de comandos y código.

Módulo de Grupos de Usuarios El módulo de grupos de usuarios permite definir grupos específicos de usuarios cuyo membresía les cambia el nivel y naturaleza del acceso que tengan en la aplicación.

Módulo de Permisos de Usuarios y Grupos El módulo de permisos de usuarios y grupos define la naturaleza de acceso que tienen los usuarios autorizados y los grupos a los cuales pertenecen dentro de la aplicación.

Módulo de Roles de Usuarios y Grupos El módulo de roles de usuarios y grupos define las responsabilidades de varios grupos de usuarios y usuarios específicos, en base a esto les asigna permisos adecuados tanto para proteger la aplicación contra uso o accesos indebidos y al mismo tiempo permitir a los usuarios poder realizar sus actividades normalmente sin impedimentos.

Subsistema de Ejecución El subsistema de ejecución de código se encarga de recibir y procesar comandos del sistema externo GitEdu con el fin de aprovechar los hipervisores que tiene a su cargo y proveer de las funcionalidades de ejecución en línea, aplicación de pruebas unitarias y calificación automática de código sin exponer dichos hipervisores a usuarios finales. Para aumentar la seguridad que logra llevar EduNube, GitEdu se comunica con lo mismo a través de canales seguros como HTTPS y SSH utilizando métodos de autenticación orientados a un esquema de llaves públicas y privadas (criptografía asimétrica) para asegurar de la identidad de ambos previo al envío de comandos y comunicación de datos y notas.

Módulo de Recepción de Comandos El módulo de recepción de comandos se encarga de recibir comandos externos enviados a través del API externa que provee para que otras aplicaciones puedan indicar cuando debe bajar, ejecutar o calificar algún código.

Módulo de Permisos y Restricciones de Comandos El módulo de permisos y restricciones de comandos se encarga de limitar o permitir comandos externos recibidos por el sistema EduNube. Se basa en políticas establecidas previamente que definen los horarios de trabajo de ciertos procesos y dependencias de los mismos (por ejemplo no se debe calificar los exámenes de un paralelo hasta que todos terminen). Junto al módulo de control de acceso a código, permite a los docentes y administradores tener control refinado sobre cuando y que está permitido hacer por parte de los usuarios que están usando EduNube a través de GitEdu.

Módulo de Control de Hipervisores El módulo de control de hipervisores se encarga de cumplir con funcionalidades críticas del sistema EduNube de compilar, ejecutar y calificar código. Esto lo hace a través de tres respectivos submódulos para cada fase. Este módulo también controla los hipervisores conectados al sistema EduNube con el fin de poder realizar las operaciones listadas previamente.

Submódulo de Compilación de Código El submódulo de compilación de código se encarga de compilar código que se entrega en base a procedimientos que define el profesor anteriormente, y reporta resultados del mismo: de éxito o errores de compilación. Para la implementación del mismo submódulo, se aprovecha de las máquinas virtuales que provee el hipervisor con el fin de realizar la compilación en un ambiente controlado y aislado.

Submódulo de Ejecución de Código El submódulo de ejecución de código recibe de entradas de código compilado y comandos que se desea ejecutar en el ambiente de ejecución y pruebas con el fin de proveer, a través de las máquinas virtuales controladas y aisladas, un ambiente interactivo donde los estudiantes y docentes pueden trabajar de una forma dinámica con código desarrollado dentro de la plataforma.

Submódulo de Calificación Automatizado de Código El submódulo de calificación automatizado de código se basa en pruebas unitarias escritas previamente por un docente y que van contra la funcionalidad que se pide en el código de los estudiantes, con el fin de calificar el código de una forma autónoma. Una vez que termina de calificar un código, reporta los resultados a gestores de los mismos.

Módulo de Persistencia de Código El módulo de persistencia de código recupera el código versionado en servidores de control de versiones externos que los usuarios han escrito y editado con el fin de permitir la compilación, ejecución y calificación del mismo dentro del sistema EduNube.

Módulo de Control de Acceso a Código El módulo de control de acceso a código se encarga de restringir y permitir en base a reglas las operaciones de compilación, ejecución y calificación automatizada de códigos que los usuarios editan. De la misma forma permite definir flujos de trabajo y dependencias para el procesamiento de proyectos en adición a establecer horarios y restricciones de procesamiento para los mismos.

Módulo de Comunicación de Datos y Notas El módulo de comunicación de datos y notas permite la sincronización de los mismos cuando sea necesario entre EduNube y GitEdu.

Subsistema de Administración El subsistema de administración ayuda a los administradores de EduNube recolectar datos acerca del uso del mismo y analizarlos, además de ser capaces de configurar y monitorear el rendimiento del sistema.

Módulo de Configuración El módulo de configuración permite la configuración en tiempo real del sistema EduNube, por ejemplo conexiones con sistemas externos, gestión de llaves públicas y privadas, y gestión de usuarios.

Módulo de Monitoreo El módulo de monitoreo permite ver en tiempo real el consumo del sistema y sus hipervisores respectivos.

Módulo de Recolección de Datos El módulo de recolección de datos lleva un registro contínuo del uso del sistema para su revisión y análisis por parte de los operadores humanos, para después poder hacer temas de auditoría como toma de decisiones estratégicas.

Módulo de Análisis de Datos El módulo de análisis de datos proporciona las herramientas necesarias para que un operador humano pueda interpretar los mismos y sacar sus conclusiones.

Ventajas del Diseño de Sistemas, Subsistemas y Módulos

La ventaja de un diseño orientado a módulos como se lo presenta aquí es la división de la funcionalidad para ayudar en la extensibilidad y mantenibilidad de los sistemas al largo plazo. Por ejemplo este diseño permite agregar funcionalidad adicional a futuro como autenticación por otros medios (por ejemplo LDAP) o la integración de mecanismos anti-plagio. Esta propiedad puede dar una vida útil mayor a los sistemas desarrollados porque muchas veces no se conoce todo lo que puede requerir un software a futuro para el diseño e implementación inicial. A continuación se presenta la arquitectura de estos dos sistemas, tanto GitEdu como su ambiente de ejecución de código virtualizado, EduNube.

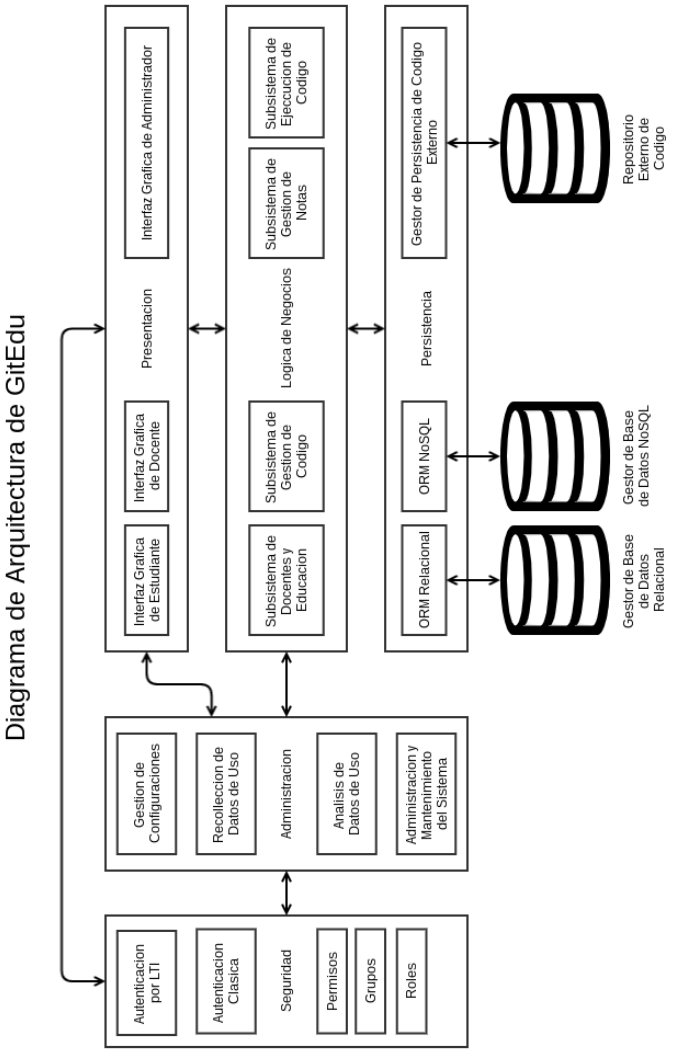

FIGURA 3.4: Diagrama de Arquitectura para el Sistema GitEdu.

3.2.2. Arquitectura

Como se puede observar en la figura 3.4, a nivel global, el sistema GitEdu consiste en 5 capas que son:

- **Seguridad**, auténtica usuarios y controla el acceso que tiene cada uno de ellos. Esto se realiza con los siguientes componentes:
  - **Autenticación por LTI** usa interfaces LTI de aplicaciones externas como LMS institucionales para autenticar usuarios.
  - **Autenticación Clásica**, permite el login (autenticación) de usuarios mediante la forma clásica, de un usuario y contraseña.
  - **Permisos**, permite otorgar o revocar partes específicas del sistema a usuarios específicos. Ejemplos de estos podrían ser: acceso a la administración de una materia, por ejemplo el control que se da a un docente o grupo de docentes sobre una materia.
  - **Grupos**, permite aplicar el sistema de permisos a grupos de usuarios creados bajo la discreción de usuarios suficientemente privilegiados dentro del sistema. Ejemplos de estos podrían ser miembros de una cierta materia o tipo de materia, por ejemplo "Base de Datos".
  - **Roles**, permite definir usuarios por su propósito del sistema con el fin de aplicarles un conjunto global de permisos asociados por el tipo de usuario que es. Ejemplos de roles podrían ser Estudiante, Docente y Administrador.
- **Administración** permite la administración, mantenimiento y recolección de datos por usuarios dentro del sistema. Para cumplir con estas responsabilidades se divide en:
  - **Subsistema de Gestión de Configuraciones**, que permite a administradores cambiar de forma rápida y fácil las configuraciones internas del sistema.
  - **Subsistema de Recolección de Datos de Uso**, que colecta automáticamente datos del uso del sistema.
  - **Subsistema de Análisis de Datos de Uso**, que permite a administradores visualizar y analizar los datos que ha recolectado el Subsistema de Recolección de Datos de Uso.
  - **Subsistema de Administración y Mantenimiento**, que permite a administradores ver en tiempo real y también registros históricos de consumo de recursos. Este subsistema también proporciona a administradores las herramientas que necesita para gestionar y controlar problemas como usuarios maliciosos, sistemas caídos entre otros problemas detectados.
- **Presentación**, que se encarga de presentar interfaces web a varias tipos de usuario, incluyendo:
  - **Estudiantes**
  - **Docentes**
  - **Administradores**

- **Lógica de Negocios**, que se encarga de llevar a cabo los procesos de negocio para cumplir con el propósito de la aplicación. Esto se subdivide en cuatro subsistemas:
  - **Subsistema de Docentes y Educación**, que se lleva a cabo procesos de negocio que tienen que ver con enseñanza y evaluación. Dentro de este subsistema también se lleva las tareas de administración asociadas con un docente y sus aulas.
  - **Subsistema de Gestión de Código**, lleva el control del código y los procesos asociados.
  - **Subsistema de Gestión de Notas**, maneja los procesos de negocio asociados con la sincronización y auditoría de notas generadas dentro del sistema.
  - **Subsistema de Ejecución de Código**, que lleva a cabo procesos de negocio que tienen que ver con la ejecución arbitraria de código y por lo tanto requiere un cierto nivel de seguridad y aislamiento elevado a través de entornos virtualizados. Dentro de los procesos que se incluyen aquí: compilación, ejecución y calificación de código automatizado mediante pruebas unitarias.
- **Persistencia**, que se encarga de asegurar que persisten datos importantes para su uso a futuro y también la recuperación de los mismos. Esto se subdivide en tres clases de persistencia:
  - Persistencia a través de un **ORM Relacional**, que guarda y recupera datos estructurados de la aplicación en bases de datos relacionales y orienta los mismos datos a objetos dentro de la aplicación para permitir su fácil manipulación.
  - Persistencia a través de un **ORM NoSQL**, que guarda y recupera datos no estructurados de la aplicación en bases de datos no relacionales y orienta los mismos datos a objetos dentro de la aplicación para permitir su facil manipulación.
  - Persistencia de código a través de un **Gestor de Persistencia de Código Externo** que guarda y recupera código de usuarios para su manipulación dentro de la aplicación y persistencia fuera del mismo.

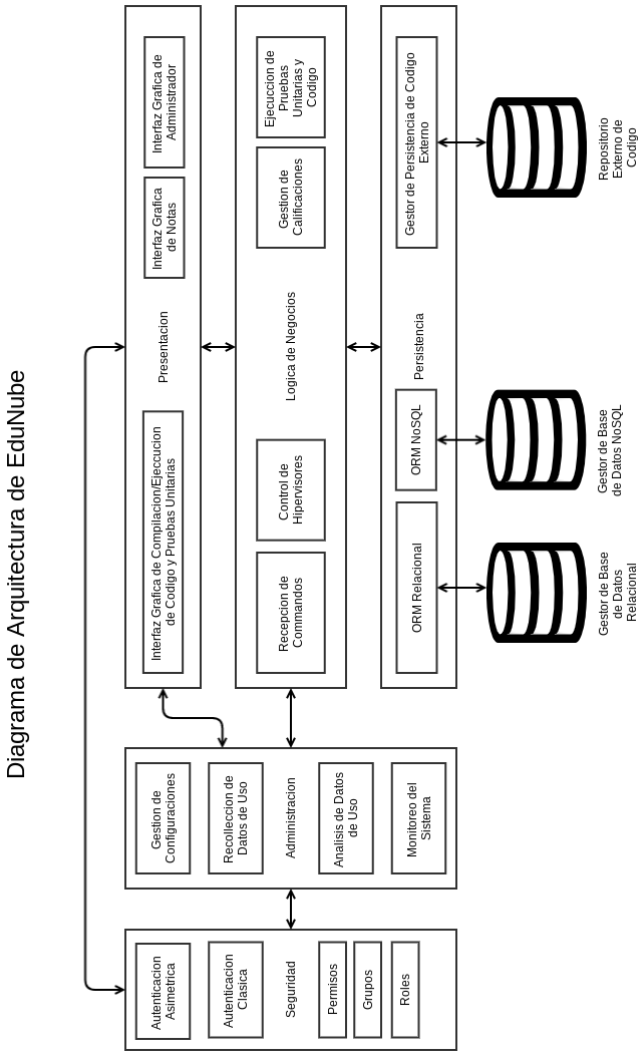

FIGURA 3.5: Diagrama de Arquitectura para el Sistema EduNube.

Como se puede observar en la figura 3.5, a un nivel global, el sistema EduNube consiste en 5 capas que son:

- **Seguridad** auténtica administradores y sistemas externos y controla el acceso que tiene cada uno de ellos. Esto se realiza con los siguientes componentes:
  - **Autenticación por Criptografía Asimétrica**, usa pares de llaves públicas y privadas suficientemente grandes para ser difíciles de falsificar matemáticamente para autenticar sistemas externos.
  - **Autenticación Clásica**, permite el login (autenticación) de administradores mediante la forma clásica de un usuario y contraseña.
  - **Permisos**, permite otorgar o revocar partes específicas del sistema a usuarios específicos. Ejemplos de permisos podrían ser quién puede crear máquinas virtuales o ver el consumo de recursos del sistema.
  - **Grupos**, permite aplicar el sistema de permisos a grupos de usuarios creados bajo la discreción de administradores suficientemente privilegiados dentro del sistema. Ejemplos de estos podrían ser ciertos tipos de usuario que representa algún tipo de estudiante en algún tipo de materia, por ejemplo para la materia de ”Base de Datos”.
  - **Roles**, permite definir usuarios por su propósito en el sistema, con el fin de aplicarles un conjunto global de permisos asociados por el tipo de usuario que es. Ejemplos de roles podrían ser Sistema Externo y Administrador.
- **Administración**, permite la administración, mantenimiento y recolección de datos por usuarios dentro del sistema. Para cumplir con estas responsabilidades se divide en:
  - **Subsistema de Gestión de Configuraciones**, que permite a administradores cambiar de forma rápida y fácil de las configuraciones internas del sistema.
  - **Subsistema de Recolección de Datos de Uso**, que colecta automáticamente datos del uso del sistema.
  - **Subsistema de Análisis de Datos de Uso**, que permite a administradores visualizar y analizar los datos que ha recolectado el Subsistema de Recolección de Datos de Uso.
  - **Subsistema de Monitoreo**, que permite a administradores ver en tiempo real y también registros históricos de consumo de recursos. Este subsistema también proporciona a administradores las herramientas que necesita para gestionar y controlar problemas como usuarios maliciosos, sistemas caídos entre otros problemas detectados.
- **Presentación**, que se encarga de presentar interfaces web a varias tipos de usuario, incluyendo:
  - **Compilación/Ejecución de Código y Pruebas Unitarias**, proporciona las interfaces gráficas utilizadas durante la compilación y ejecución de código en la aplicación de pruebas unitarias contra un código para ayudar en la calificación del mismo.
  - **Notas**, permite a docentes y sistemas externos ver las notas asignadas a un código.

    - **Administradores**, proporciona las interfaces gráficas necesarias para ayudar un administrador a cumplir con sus responsabilidades de monitoreo, control y mantenimiento del sistema en ambientes de producción.

- **Lógica de Negocios**, que se encarga de llevar a cabo los procesos de negocio para cumplir con el propósito de la aplicación. Esto se subdivide en cuatro subsistemas:

    - **Subsistema de Recepción de Comandos**, que se lleva a cabo procesos de negocio que tienen que ver con recibir comandos de sistemas externas y procesar los mismas.
    - **Subsistema de Control de Hipervisores**, lleva el control de los hipervisores conectados y les guía en el proceso de virtualizar y ejecutar código dentro de máquinas virtuales de los mismos hipervisores.
    - **Subsistema de Calificaciones**, maneja los procesos de negocio asociados con la calificación, sincronización y auditoría de notas generadas dentro del sistema.
    - **Subsistema de Ejecución de Código y Pruebas Unitarias**, que lleva a cabo procesos de negocio que tienen que ver con ejecución arbitraria de código en entornos virtualizados. Dentro de los procesos que se incluyen aquí, hay compilación, ejecución y calificación de código automatizado mediante pruebas unitarias.

- **Persistencia** que se encarga de asegurar que se persisten los datos importantes para su uso futuro y también la recuperación de los mismos. Esto se subdivide en tres clases de persistencia:

    - Persistencia a través de un **ORM Relacional** que guarda y recupera datos estructurados de la aplicación en bases de datos relacionales y orienta los mismos datos a objetos dentro de la aplicación para permitir su fácil manipulación.
    - Persistencia a través de un **ORM NoSQL** que guarda y recupera datos no estructurados de la aplicación en bases de datos no relacionales y orienta los mismos datos a objetos dentro de la aplicación para permitir su fácil manipulación.
    - Persistencia de código a través de un **Gestor de Persistencia de Código Externo** que guarda y recupera código de usuarios para su manipulación dentro de la aplicación y persistencia fuera del mismo.

Ventajas del Diseño Arquitectónico

Se ha considerado para ambos sistemas una arquitectura de n-capas debido a su capacidad de escalabilidad sobre varios niveles físicos. Además la separación lógica entre capas permite mayor mantenibilidad al largo plazo donde solo se necesitan reemplazar los componentes que requieren cambios en lugar de todo el sistema. El nivel de aislamiento entre estas mismas capas brinda mejor seguridad contra usuarios externos maliciosos o que quieran dar mal uso (un uso inadecuado) al sistema. A continuación se presenta la arquitectura de despliegue propuesta dentro de este trabajo de titulación.

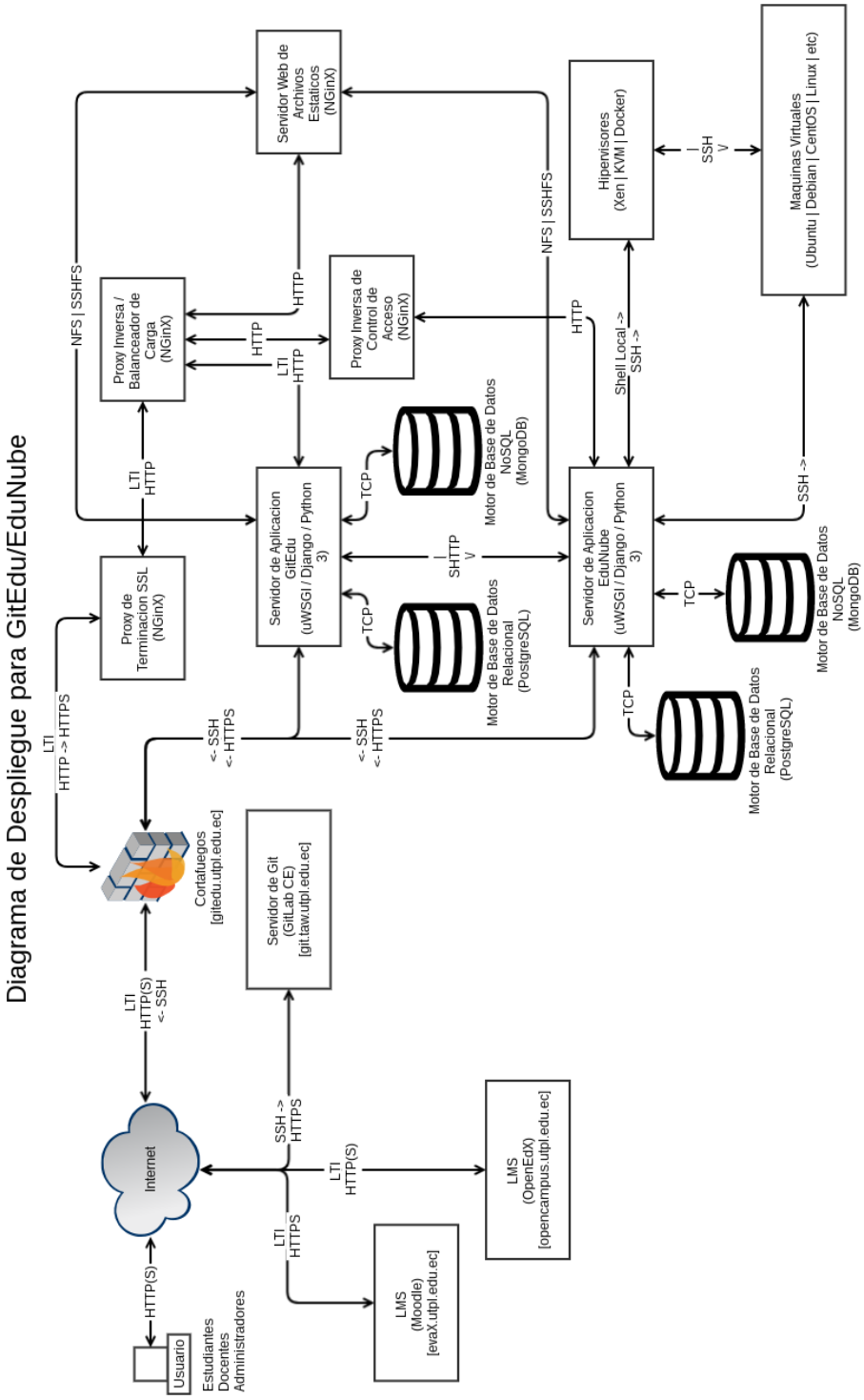

FIGURA 3.6: Diagrama de Despliegue para los Sistemas GitEdu y EduNube.

3.2.3. Despliegue

Como se puede observar en la figura 3.6, el despliegue del sistema GitEdu junto al sistema EduNube, consiste en dos redes, una externa que contiene los usuarios finales (Estudiantes, Docentes, Administradores) en adición a sistemas externas como los dos LMS institucionales (Moodle y OpenEdX) y el servidor de Git (GitLab Community Edition), mientras que la red interna contiene varios componentes que forman el conjunto de sistemas:

- **Cortafuegos**, divide la red interna de la red externa y permite únicamente que pase tráfico HTTP y HTTPS de manera bidireccional (con posibilidad de llamadas LTI sobre estos protocolos) y la salida de SSH al servidor de Git (para ofrecer mayor seguridad que transferencias sobre HTTP(S)).
- **Proxy de Terminación SSL**, accepta HTTP y HTTPS, pero para asegurar la seguridad de usuarios finales y aplicaciones externas redirecciona al uso del protocolo HTTP a HTTPS. Además descifra conexiones HTTPS entrantes para manejarlos por HTTP dentro de la red interna (reduciendo la carga en otros servidores sin abandonar la seguridad en la red externa) y cifra conexiones salientes. Se utiliza NGinX por su bajo consumo de recursos en comparación con sus competidores como Apache.
- **Proxy Inversa / Balanceador de Carga**, acepta todo el tráfico HTTPS (ahora por el protocolo HTTP dentro de la red interna) entrante y elige en base a la ruta cual servidor se lo puede responder. A futuro este componente permite que se pueda escalar el sistema en lo que permite dividir tráfico entre varios servidores del mismo tipo (por ejemplo dos servidores de aplicación: GitEdu1 y GitEdu2). Se considera que NGinX es la solución más óptima para realizar este rol.
- **Servidor Web de Archivos Estáticos**, se responsabiliza por peticiones de archivos estáticos que no conviene enviar a los servidores de aplicación (quienes los procesaría con mayor lentitud). Estos mismos tienen una conexión NFS o SSHFS con los mismos servidores de aplicación para que estos sean capaces de actualizar remotamente los archivos estáticos en caso de que los mismos cambien (por ejemplo cuando un usuario actualiza su foto de perfil). Estudios de rendimiento demuestran que NGinX cumple este rol de ser servidor web de archivos estáticos mejor que Apache.
- **Servidor de Aplicación "GitEdu"**, se aloja el sistema GitEdu, desarrollado bajo el marco Django y el lenguaje de programación Python (version 3.x). Se comunica con el servidor externo de Git sobre SSH y dispone tanto de un base de datos relacional como de una no relacional para persistir los distintos tipos de datos que tiene que manejar. Se conecta directamente al Servidor de Aplicación "EduNube" mediante SHTTP (HTTP sobre un túnel SSH) utilizando un par de llaves (pública y privada) como esquema de criptografía asimétrica. Se considera uWSGI como mejor servidor de aplicación para el sistema debido a que, a diferencia de su competencia (Gunicorn), esta suporta Websockets.
- **Motores de Base de Datos Relacional**, se alojan datos estructurados para ambos sistemas, tanto GitEdu como NubeEdu.
- **Motores de Base de Datos NoSQL**, se alojan datos no estructurados para ambos sistemas, tanto GitEdu como NubeEdu.

- **Proxy Inversa de Control de Acceso**, restringe acceso al servidor de aplicaciones de EduNube cuando se accede al mismo desde la red externa (a través del proxy inversa).
- **Servidor de Aplicación "EduNube"**, se aloja el sistema EduNube, desarrollado bajo el marco Django y el lenguaje de programación Python (version 3.x). Se comunica con el servidor externo de Git sobre SSH y dispone tanto de un base de datos relacional como una no relacional para persistir los distintos tipos de datos que tiene que manejar. Utiliza un shell local (dentro del mismo servidor) para manejo de un hipervisor local o en el caso de un hipervisor remoto, maneja el mismo sobre SSH. Las máquinas virtuales donde se levanta el hipervisor, también se los controla con SSH. Se considera uWSGI como mejor servidor de aplicación para el sistema debido a que, a diferencia de su competencia (Gunicorn), esta suporta Websockets.
- **Hipervisores**, gestionan máquinas virtuales por parte del sistema EduNube el mismo que envía comandos por SSH o un shell local. A su vez, controla sus máquinas virtuales mediantes SSH. Dentro de las tecnologías de hipervisores se considera Xen, KVM y Docker como buenas opciones para diferentes necesidades. Cada sistema de hipervisores soporta el uso de plantillas para la creación masiva de máquinas virtuales iguales para cada estudiante de una materia.
- **Máquinas Virtuales**, ejecutan código de usuarios y reportan los resultados.

Ventajas de la Arquitectura de Despliegue

De la misma manera, el diseño de sistemas, subsistemas y módulos en adición al diseño arquitectónico, el diseño de arquitectura de despliegue está orientado a la extensibilidad a través de modularidad, mantenibilidad por la naturaleza de sus capas y módulos, además de escalabilidad y seguridad mantenida a través de mecanismos de aislamiento de componentes que son libres de escalar independientemente entre ellos, la arquitectura de despliegue se orienta a los mismos principios.

La separación de tareas dentro del sistema permite que cada servicio dentro de la arquitectura de despliegue se pueda dedicar a hacer bien su parte correspondiente de la manera más eficiente posible sin responsabilidades que no pueda sostener de forma inadecuada, que es algo importante para empezar a poder garantizar el rendimiento óptimo del sistema. El uso de un cortafuegos y varios proxies inversos al frente de cada sistema también ayuda a aislar los mismos del mundo externo a un nivel de protocolos que reduce las probabilidades de ataques contra los mismos, ofrece seguridad sin sacrificar rendimiento de manera similar al punto anterior de separación de trabajo en roles y asignación al actor más adecuado para el mismo.

Este mismo uso de procesos, especializado cada uno con sus responsabilidades y posible carga de trabajo también sigue ofreciendo escalabilidad a futuro. Esto se realiza con comunicación entre servicios que está sobre protocolos de red para que los mismos puedan estar en un solo recurso (sea servidor físico o virtual) o en varios apartados lo cual permite a futuro la asignación de recursos físicos o virtuales de acuerdo a la necesidad. Esta capacidad para escalar se puede ofrecer gracias al hecho de que los servicios son semi independientes unos de otros, permitiendo n-capas sobre m-niveles físicos/lógicos.

Donde haya dependencias, por ejemplo, sistema de ficheros compartidos para dos capas distintas se puede implementar con protocolos que permitan acceso remoto para las operaciones necesarias, en este caso que los servidores de aplicación sean capaces de actualizar los archivos estáticos en el servidor para este mismo (se lo está considerando esta parte con un NFS o SSHFS). Obviamente en el caso de que estén en el mismo servidor, no se da la necesidad de establecer esta conexión.

En casos donde se necesite mayor seguridad para proteger canales de comunicación, se ha considerado que la mejor forma de controlar esto es con tuneles SSH que encapsulan el tráfico que de por sí solo ocupa un protocolo adecuado. Por ejemplo para consumir el API de EduNube, GitEdu utiliza HTTP sobre SSH (también conocido como SHTTP). De esta forma se puede aprovechar lo que ya existe y que está comprobado como seguro y eficiente (SSH como protocolo, especialmente en su segunda versión, tiene fuertes mecanismos de autenticación y seguridad mientras que al mismo tiempo ofrece menor sacrificio de rendimiento y alta funcionalidad).

A continuación se presenta el alcance del desarrollo, pruebas y despliegue para este trabajo de titulación.

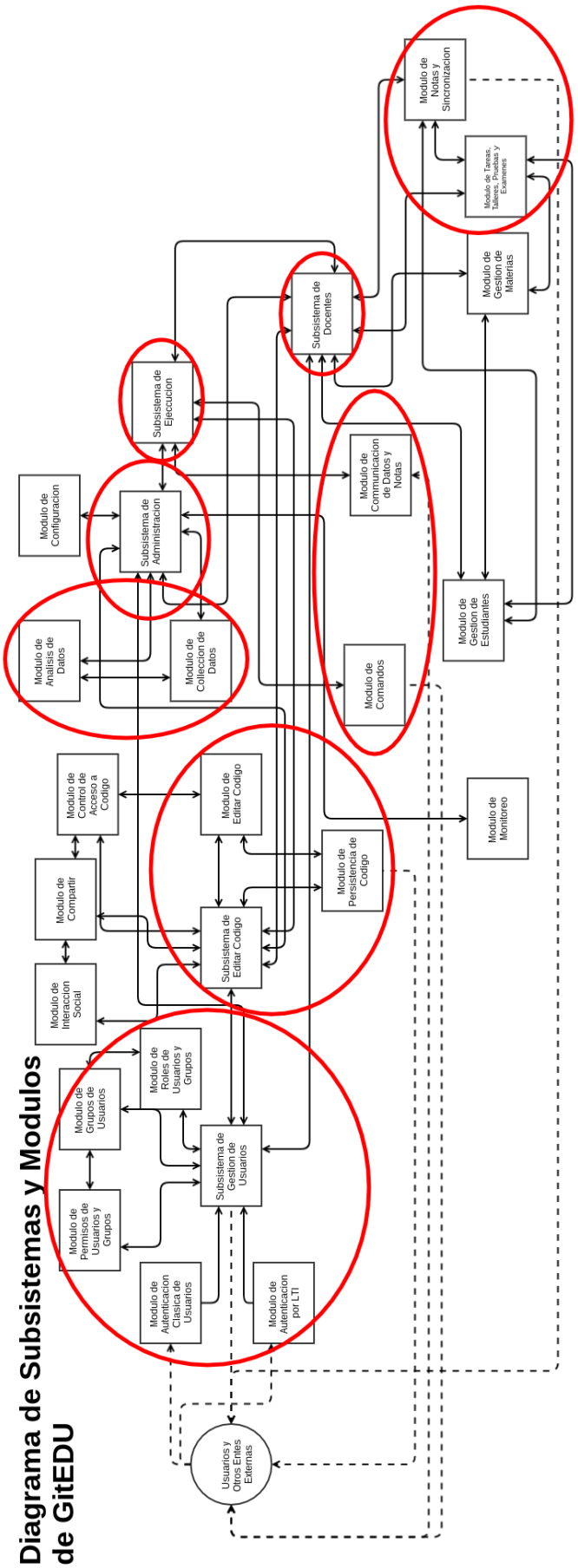

FIGURA 3.7: Alcance de Módulos para GitEdu.

3.2.4. Alcance

Debido al hecho de que se cuenta con tiempo y recursos limitados para realizar y acabar el trabajo de titulación, se considera un alcance limitado en ciertos aspectos con el fin de terminar con los componentes críticos del sistema dentro del tiempo dado.

Para el sistema GitEdu se considera los siguientes subsistemas y módulos críticos como se puede observar en la figura 3.7:

- **Todo el subsistema de Gestión de Usuarios.** Todos los módulos de este sistema son críticos porque se encargan de cómo los usuarios se autentican y el control de acceso que tiene cada uno. Por lo tanto el subsistema en sí es una parte importante para garantizar seguridad en el sistema final.
- **Parte del subsistema de Editar Código.** El subsistema de editar código se considera crítico así como los módulos de editar y de persistir código. El módulo de editar código es crítico porque es la funcionalidad principal de que se trata el sistema mientras que el modulo de persistir codigo viene a ser critico para sincronizar código entre los dos sistemas. No se toma en consideración como críticos los módulos de interacción social, compartir código ni control de acceso a código porque las mismas se tratan de llevar el sistema más allá de su propósito original y son más adecuados para un futuro trabajo.
- **Parte del subsistema de Docentes.** Dentro del subsistema de docentes, se considera crítico elmódulo de crear talleres, pruebas y exámenes en adición a un módulo de notas y su sincronización. Entre estos dos se forma el núcleo de funcionalidad que requieren los docentes dentro de la versión inicial. Dentro de trabajos futuros, se podría extender esta funcionalidad crítica con temas más administrativos como los módulos de gestión de materias y gestión de estudiantes que por el momento no se los considera como funcionalidades críticas para una primera versión estable de este trabajo de titulación.
- **Parte del subsistema de Administración.** Dentro del subsistema de administración, solo se considera que los componentes críticos son la colección de datos y el análisis de los mismos debido a que estos dos módulos respectivamente ayudan en la toma de decisiones que tienen que ver con la escuela de Ciencias de Computación y Electrónica. Los módulos de monitoreo y configuración, aunque podrían ser muy útiles en un ambiente de producción no caen dentro del alcance inicial de esta tesis y por lo tanto no se los considera críticos bajo los parámetros del proyecto. Sin embargo, su implementación podría ser una funcionalidad adicional interesante para un futuro proyecto.
- **Todo el subsistema de Ejecución.** Todos los módulos de este sistema son críticos porque se encargan de la comunicación con el sistema EduNube. El módulo de comandos prepara comandos para ejecutar a través del módulo de comunicación de datos y notas se los envía en adición a conseguir resultados y sincronizar notas y calificaciones generadas por EduNube.

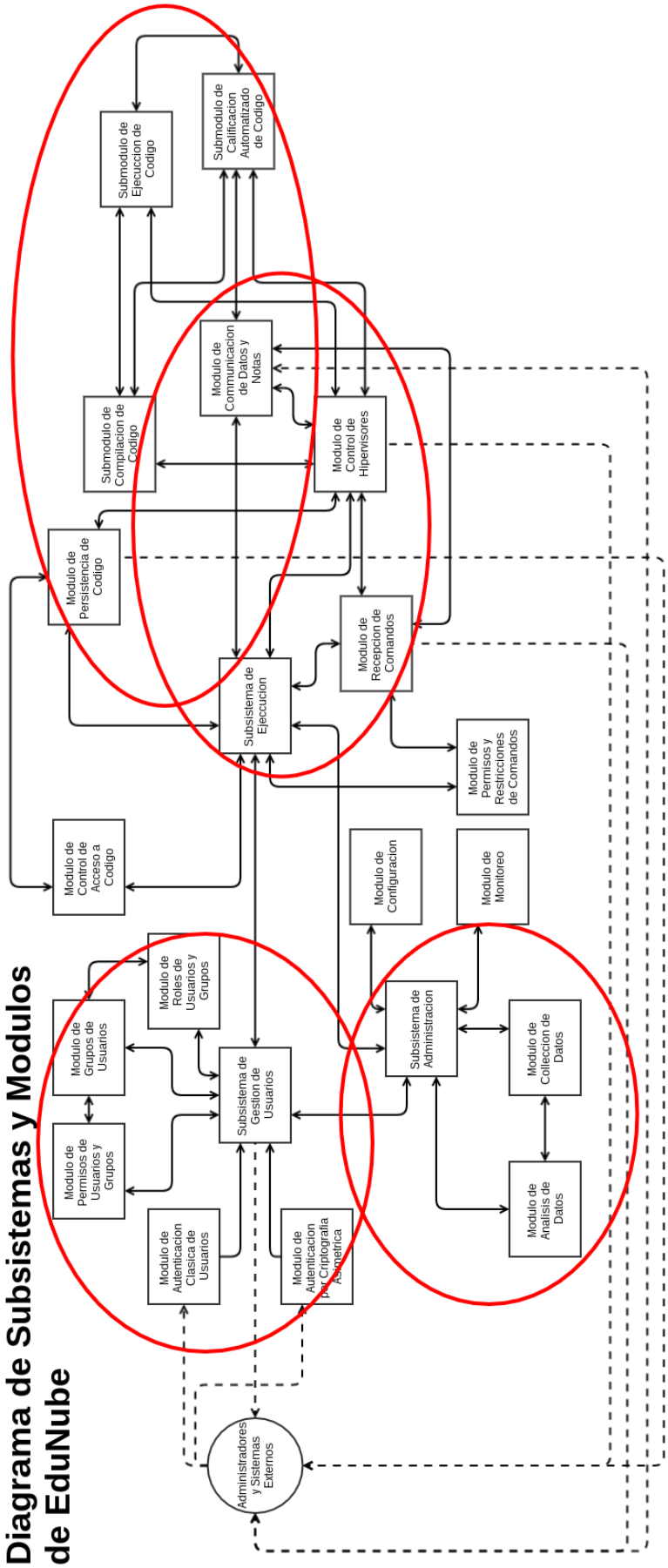

FIGURA 3.8: Alcance de Módulos para EduNube.

Para el sistema EduNube, se considera los siguientes subsistemas y módulos como críticos, como se muestra en la figura 3.8:

- **Todo el subsistema de Gestión de Usuarios.** Todos los módulos de este sistema son críticos porque se encargan de cómo los usuarios se autentican contra el mismo sistema y el control de acceso que tienen cada uno en él. Por lo tanto el subsistema en sí es una parte importante para garantizar la seguridad del sistema final.
- **Parte del subsistema de Ejecución.** Dentro del subsistema de ejecución se consideran críticos los módulos de comandos (para procesar comandos entrantes), de control de hipervisores (para gestionar recursos virtuales), y de persistencia de código (para recuperar código para su compilación, compilación y ejecución). Además dentro de los componentes del módulo de control de hipervisores, se considera crítico los submódulos de compilación de código (para aquellos lenguajes que lo requieren), de ejecución (para ejecutar código), y de calificación automatizada de código (para calificar el código de estudiantes). Se considera que los módulos de seguridad adicionales dentro de este subsistema no son necesarios por el momento debido a que el único sistema que tendrá acceso a esta será la de GitEdu. A futuro, donde más sistemas y servicios empiecen a necesitar consumir el API de EduNube, llegará a ser necesario que estos módulos cuenten con seguridad para proveer protección adicional.
- **Parte del subsistema de Administración.** Dentro del subsistema de administración, solo se considera que los componentes críticos se tratan de colección de datos y el análisis de los mismos debido a que estos dos módulos respectivamente ayudan en la toma de decisiones que tienen que ver con la escuela de Ciencias de Computación y Electrónica y el uso del sistema. Los módulos de monitoreo y configuración, aunque podrían ser muy útiles en un ambiente de producción no se involucrab dentro del alcance inicial de esta tesis y por lo tanto no se los considera críticos bajo los parámetros del proyecto. De todas formas su implementación podría ser una funcionalidad adicional e interesante para un futuro proyecto.

En cuanto el despliegue (figura 3.9), los sistemas externos tomados en consideración para el alcance son Moodle como LMS, y GitLab Community Edition como servidor de control de versiones. En relación a los ambientes virtuales, solo se considera un hipervisor y un sistema operativo para las máquinas virtuales dentro de la versión final (durante el desarrollo se presenta los candidatos, comparación y selección de los mismos). También se incluye dentro del alcance el desarrollo y despliegue de los dos sistemas GitEdu y EduNube con sus respectivas bases de datos utilizando uWSGI como servidor de aplicación. No se incluye dentro del alcance OpenEdX o ningún otro LMS externo fuera del mencionado previamente, ni las configuraciones del cortafuego y servidores web, estos quedan como responsabilidades institucionales si se desean implementar en el despliegue. Lo que se presenta aquí es una sugerencia del autor basado en su experiencia práctica.

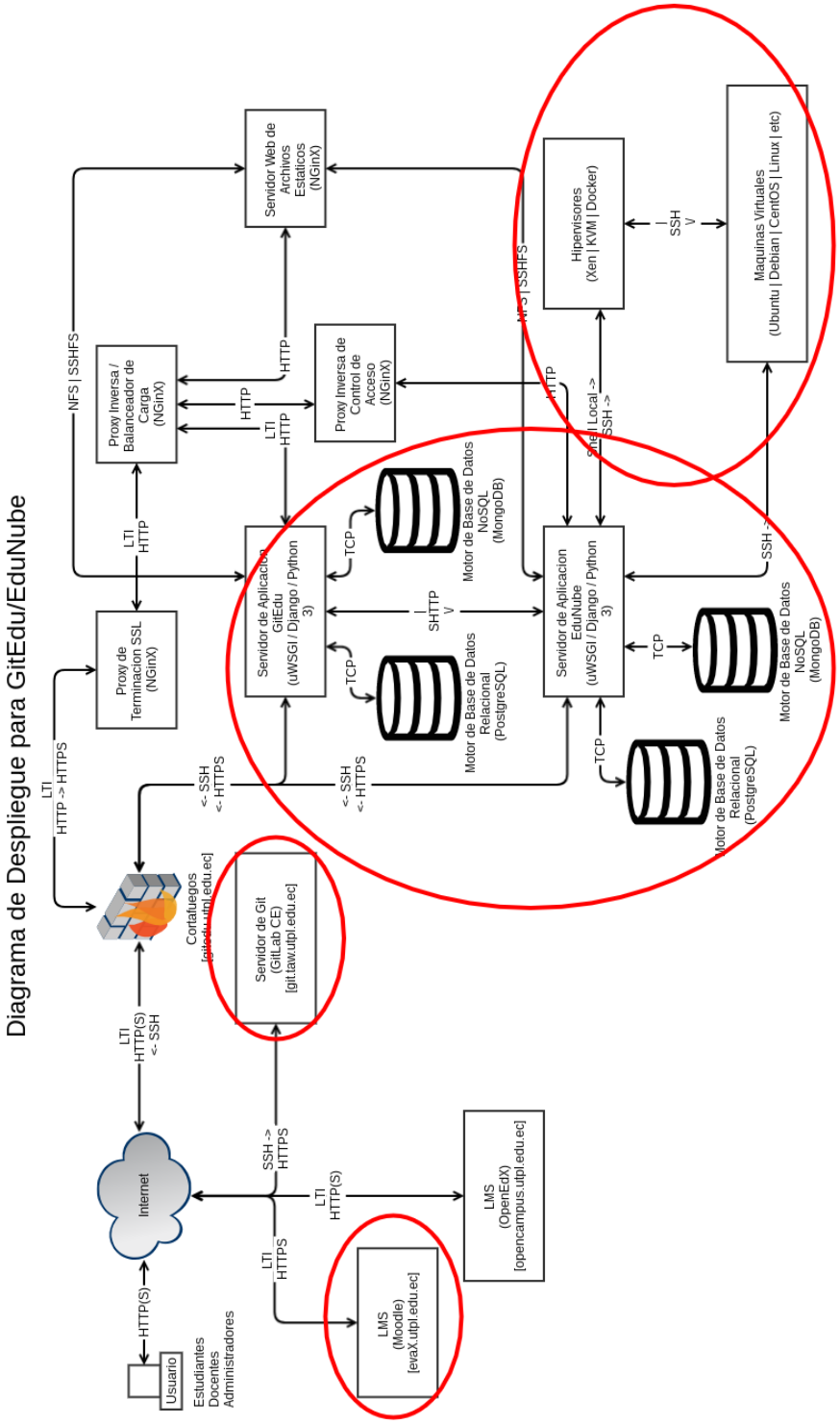

FIGURA 3.9: Alcance de Despliegue.

Capítulo 4

Desarrollo

4.1. Preparaciones del Ambiente de Desarrollo

Previo a realizar el desarrollo, es necesario que se prepare el ambiente con:

- un sistema base (Debian Stretch con el Xen Hipervisor)
- un servidor local de Git (Git + Gitweb + NGinX),
- un servicio para recibir peticiones HTTP y convertirlos en sus respectivos objetos en un repositorio de Git (Git Server HTTP Endpoint)
- un LMS (Moodle),
- un servidor de Git (GitLab CE)
- un servidor de Virtualización (Docker)
- plantillas de máquinas virtuales (en Docker)

4.1.1. Sistema Base de Desarrollo

Como se presenta en la figura 4.1, el sistema base para el desarrollo de este trabajo de titulación utiliza Xen instalado con Debian Stretch (9.1)[1] instalado en un LVM con el nombre del grupo de volúmenes "Xephyr-VG". Originalmente se tenía planificado trabajar con Debian Jessie (8.x) debido que esa era la versión estable al momento de la instalación, sin embargo luego se optó por la actualización a la versión beta de Debian del momento (Debian Stretch). En resumen los pasos realizados fueron:

1. Instalación Limpia de Debian 8 con un LVM.

[1] Como el sistema de Dom0

FIGURA 4.1: Información del Sistema Base.

Volumen Físico: /dev/sda8 310g

Grupo de Volúmenes: Xephyr-VG 310g

Volúmenes Lógicos: Originalmente 4 (se agrega 2 por cada nueva máquina virtual – uno para su disco y otro para su área de intercambio).

Xephyr-Dom0 Xephyr-VG 30g

Xephyr-IMG-Repo Xephyr-VG 20g

Xephyr-ISO-Repo Xephyr-VG 20g

Xephyr-Swap Xephyr-VG 4g

2. Actualizar la instalación de Debian 8.

```
apt update
apt upgrade
apt dist-upgrade
reboot
```

3. Actualizar Debian 8 a Debian 9.

```
sed -i 's/jessie/stretch/g' /etc/apt/sources.list
apt update
apt upgrade
reboot
```

4. Instalación de las Herramientas de Trabajo

```
apt install tmux vim zsh
```

5. Instalación y Configuración de Hipervisor Xen

```
apt install xen-hypervisor
dpkg-divert --divert /etc/grub.d/08_linux_xen --
   rename /etc/grub.d/20_linux_xen
update-grub
cat > /etc/network/interfaces.d/xenbr << EOF

auto xenbr0
iface xenbr0 inet static
        address 10.10.10.1
        netmask 255.255.255.0
        bridge_ports wlan0

#other possibly useful options in a
#       virtualized environment
        #bridge_stp off         # disable
        #               Spanning Tree Protocol
        #bridge_waitport 0      # no delay
        #               before a port becomes
        #               available
        #bridge_fd 0            # no forwarding
        #               delay

## configure a (separate) bridge for
```

```
#         the DomUs without giving Dom0 an
#         IP on it
#auto xenbr1
#iface xenbr1 inet manual
#    bridge_ports eth1

EOF

reboot
```

6. Instalación de Herramientas de Xen

```
apt install xen-tools xen-utils
```

7. Instalación de Herramientas de Desarrollo para Python 3

```
apt install python3 python3-virtualenv python3-pip
```

8. Instalación de Servidores

```
apt install nginx-full postgresql mongodb redis-
    server gitweb fcgiwrap
```

9. Instalación de otras dependencias

```
apt install libffi-dev
```

10. Instalación de IDEs en /opt. Se descargaron los respectivos .tar.*xx* y se los descomprimió en /opt con un comando similar al siguiente:

```
tar -axvf nombre.tar.xx -C /opt/ruta/raiz/donde/
    descomprimir
```

    a) PyCharm (Community o Professional Edition[2])
    b) DBeaver (Community o Enterprise Edition[3])

Estos pasos de instalación se basaron en la guía de instalación de Xen publicado en el wiki del proyecto de Debian (Debian Community, 2017).

Para dar conexiones hacia el exterior de las maquinas virtuales, se necesita activar una regla NAT en el cortafuego IPTables:

```
iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

Servidor Local de Git

Para gestionar versiones de código escrito y facilitar la integración con sistemas externos y el sistema de virtualización, se levantó un servidor sencillo de Git. La instalacion del mismo se documenta en el Anexo C.

[2]Se ocupó la versión profesional, en parte por elbuen soporte para el desarrollo con Django) con una licencia estudiantil que actualmente que permite solicitar y renovar año tras año con un correo institucional

[3]Históricamente siempre han sido gratis ambas versiones con la diferencia que Enterprise Edition no es completamente software libre y agrega soporte para bases de datos no relacionales. Se ocupó la versión Enterprise, de las últimas ofertas gratis antes de que se convierta en un producto de pago, por el mismo motivo de requerir soporte para bases de datos NoSQL como MongoDB y Redis.

Git Server HTTP Endpoint

Para recibir llamadas HTTP y convertirlas en sus respectivos objetos, en un repositorio de Git se creó un miniservicio que realiza esta tarea, basado en la misma aplicación de autenticación que utiliza EduNube:

```
pipenv --three
pipenv install django six psycopg2 PyJWT bcrypt ipython \
       requests
pipenv run django-admin startproject GitServerHTTPEndpoint .
# Configurar Base de Datos, Reutilizar authApp de EduNube, \
#       migrar la base de datos, Crear un superusuario
rsync -av --progress ~/EduNube/authApp \
       ~/GitServerHTTPEndpoint/
vim GitServerHTTPEndpoint/settings.py
pipenv run python manage.py makemigrations authApp
pipenv run python manage.py migrate
pipenv run python manage.py createsuperuser

pipenv run python manage.py runserver 8002
# crear un api token para GitEDU

pipenv run python manage.py startapp apiApp

pipenv lock --requirements | awk '{print $1}' > \
       requirements.txt
```

Se puede encontrar el repositorio para este servicio en `https://gitlab.com/nishedcob/GitServerHTTPEndpoint`.

Tokens de Autenticación Para el uso del API del servicio de GitServerHTTPEndpoint, se emite tokens mediante una interfaz CRUD, construida con formularios de Django. Estos API tokens son construidos bajo el estandar JWT o JSON Web Tokens donde el servidor (servicio de Django) generar una llave secreta (mediante la función generar sal de bcrypt) y con ello se cifra los datos sobre el cliente con el fin de poder validar la identidad del mismo mediante el API token sin que el cliente pueda modificar los datos de su API token. El mismo esquema de autenticación para el API se utiliza con el servicio de EduNube. El acceso al mismo se puede encontrar en la url `http://10.10.10.1:8020/auth/tokens` y acceder con cualquier cuenta de superusuario para la creación, lectura, actualización y eliminación de los API Tokens.

API externa Los URLs del API de GitServerHTTPEndpoint toman la forma general de: `/api/<tipo_de_objeto>/<accion>/<ruta/de/objeto>`
donde los tipos de objetos posibles son:

- `ns` para los Namespace
- `repo` para Repositorios y
- `file` para Archivos

acciones posibles son:

- `create` para crear,
- `edit` para editar y
- Para el caso de los archivos, no existe un solo `edit` si no una subclasificacion de:
  - `edit/mv` para alterar la ruta de un archivo y
  - `edit/contents` para alterar los contenidos de un archivo.

Los URIs del API que operan sobre Namespace, solo aceptan un namespace de parámetro con un slash de delimitador final por ejemplo:
`/api/ns/create/nearley/`

Los URIs del API que operan sobre los repositorios requieren de los parámetros: un namespace y un repositorio igual con un slash de delimitador final por ejemplo:
`/api/repo/create/nearley/GitEDU/`

Los URIs del API que operan sobre archivos toman de parámetros un namespace, un repositorio y una ruta de archivos que puede llevar cualquier número de slash para indicar los directorios dentro del repositorio selecionado, pero la misma ruta no puede terminar en un slash, por ejemplo:
`/api/file/create/nearley/GitEDU/despliegue/Readme.md`

Un cliente ejemplar de todo el API, se puede encontrar con el nombre `example_client.py` dentro de la raíz del proyecto.

4.1.2. Ambiente Virtual de LMS (Moodle)

Originalmente se planteó trabajar con un LMS como una máquina virtual de Xen, pero con el tiempo, se ha vio la necesidad de consumir todos los servicios posibles de una forma mas liviana y por lo tanto aunque se incluye por este medio la instalación de Moodle en una máquina virtual de Xen, detallado en el Anexo C, al final se optó por un levantamiento del mismo en Docker. El levantamento de Moodle en Docker se documenta en el capítulo 6 de Despliegue.

4.1.3. Servidor de Git (GitLab CE)

Originalmente se planteó trabajar con GitLab Community Edition como una máquina virtual de Xen, pero con el tiempo, se ha vio la necesidad de consumir todos los servicios posibles de una forma más liviana al final se optó por un levantamiento del mismo en Docker. Sin embargo en este documento se incluye la instalación de la primera opción en una máquina virtual de Xen, detallado en el Anexo C, . El levantamento de GitLab en Docker se documenta en el capítulo 6 de Despliegue.

4.1.4. Servidor de Virtualización

Para la ejecución de código se consideró necesario la virtualización. La ejecución puramente nativa de código daría mayor grado de ataques al servidor físico además de no brindar de una flexibilidad necesaria para sostener una gran variedad de usuarios y necesidades.

CUADRO 4.1: Pesos de las Características de Comparación para Hipervisores Considerados

Característica	Peso
Rendimiento	10
Seguridad	10
Facilidad de Uso (CLI)	10
Facilidad de Uso (GUI)	5
Facilidad de Uso (WUI)	5
Facilidad de Instalación	5
Facilidad de Mantenimiento	5
Experiencia Personal del Autor	10
Total de Pesos	60

Comparación de Hipervisores

Para la selección de una tecnología de virtualización, se ha considerado conveniente realizar el siguiente cuadro comparativo, en base a una valoración de características de cada ecosistema de hipervisor generando una calificación ponderada de las mismas y poder llegar objetivamente al mejor hipervisor a ser implementado en el presente proyecto.

El cuadro 4.1 presenta los pesos que se asignaron a cada categoría de la comparación, para poder realizar un promedio ponderado de las valoraciones con mayor peso en las categorías que se consideraban más importantes. Las características consideradas son:

Rendimiento , indica la máxima eficiencia que puede llegar a tener la virtualización con esta tecnología.

Seguridad , indica el nivel de aislamiento y mitigación de riesgo que se considera los usuarios virtualizados frente la máquina host.

Facilidad de Uso (CLI) , la facilidad en utilizar el hipervisor desde línea de comandos (terminal).

Facilidad de Uso (GUI) , la facilidad en utilizar el hipervisor desde una aplicación de escritorio gráfico.

Facilidad de Uso (WUI) , la facilidad en utilizar el hipervisor desde la plataforma web.

Facilidad de Instalación , la facilidad para instalar el hipervisor y realizar la configuración inicial para que empiece a funcionar.

Facilidad de Mantenimiento , la facilidad para actualizar, utilizar y mantener el hipervisor.

Experiencia Personal del Autor , la opinión que tiene el autor del hipervisor después de años de uso y experimentación en estas tecnologías.

Se ha considerado con mayor importancia el Rendimiento y la Seguridad como necesidades del presente proyecto. La Facilidad de Uso desde la Línea de Comandos y la Experiencia Personal del Autor también se ha considerado importante para que se pueda implementar de forma mas concisa, rápida y precisa la tecnología de virtualización elegida. La Facilidad de Uso tanto Gráfica como Web en adición a la

CUADRO 4.2: Características de Comparación I: Hipervisores Considerados

Hipervisor	Rendimiento	Seguridad	Facilidad de Uso (CLI)	Facilidad de Uso (GUI)	Facilidad de Uso (WUI)
Xen	4	5	4	3	2
KVM	3	5	4	3	3
Docker	5	3	5	3	4

CUADRO 4.3: Características de Comparación II: Hipervisores Considerados

Hipervisor	Facilidad de Instalación	Facilidad de Mantenimiento	Experiencia Personal del Autor
Xen	3	2	4
KVM	4	2	2
Docker	5	4	4

complejidad de la instalación y mantenimiento se considera importantes pero menos relevantes, debido a que son características de menor impacto para el desarrollo actual y a largo plazo es donde tienen mayor impacto se puede implementar otra solución más adecuada frente a estos temas si es necesario.

En los cuadros 4.2 y 4.3 se presenta las características consideras y un puntaje relativo (sobre 5) dado a cada una de estos de acuerdo a investigaciones, experiencias y opiniones del autor. A continuación se explica el porqué de las valoraciones:

Xen como un hipervisor de tipo 1 cuenta con un rendimiento muy alto, especialmente cuando se lo considera la paravirtualización que también soporta sistemas operativos modificados, casi a un nivel nativo. Este hipervisor logra virtualizar todas las capas de un sistema operativo, desde su núcleo hasta el espacio de los usuarios, pero al mismo tiempo dispone de un modelo de seguridad muy bueno. Tiene falla en las herramientas de administración, de los cuales hay muy pocas que están actualizadas. La mejor herramienta de administración es en la Línea de Comandos. Sus cambios frecuentes y documentación desactualizada son un problema para la instalación y mantenimiento. Sin embargo, la experiencia del autor con esta tecnología ha sido muy positiva.

KVM como un hipervisor de tipo uno y medio tiene un nivel más alto que Xen pero no dispone del mismo nivel de rendimiento que ofrece ni la paravirtualización. Aunque es de más alto nivel, se ha xonsiderado otros pasos adicionales para alcanzar un nivel de seguridad equivalente al de Xen. Para su administración cuenta con herramientas muy similares a los de Xen, pero su alto nivel de integración con el API de Libvirt hace que cuente con mejores interfaces para su manejo web. Posiblemente por desconocimiento de las posibilidades para mejorar el rendimiento que se puede aplicar, el autor no ha tenido una experiencia muy positiva con la aplicación de esta tecnología de virtualización.

Docker como un hipervisor de "tipo tres" es de muy alto nivel y dispone de una seguridad mucho menor al tener menor aislamiento entre el sistema que provee la virtualización y la virtualizada. Docker tiene mejor rendimiento ya que se

$$Promedio = \sum_{c=[Categoras]} CalificacinCatagoria_c * \left(\frac{PesoCatagoria_c}{TotalPesos}\right)$$

FIGURA 4.2: Ecuación para calcular los promedios ponderados en base a pesos para los hipervisores comparados

CUADRO 4.4: Calificaciones Promediados Ponderadas para la Comparación de los Hipervisores

Hipervisor	**Calificación Promedio Ponderado**
Xen	3.667
KVM	3.333
Docker	4.167

virtualiza solo los niveles más altos de un sistema operativo y las aplicaciones que se encuentran por encima. Su técnica de contenerización como un tipo de virtualización, puede dar un mejor rendimiento que la paravirtualización. Se considera muy buena las interfaces para utilizar Docker, tanto a nivel de terminal como a nivel Web, por el hecho de que Docker es una tecnología que aún es nueva y que esta en auge. Esto promueve a una gran cantidad de personas en contribuir continuamente con mejoras. La instalación y mantenimiento de Docker es muy sencilla en cualquier plataforma. Debido a su sencillez, alto rendimiento, funcionalidad y utilidad, el autor ha tenido una buena experiencia con el uso de esta tecnología de virtualización.

Para calcular los pesos promedio de forma ponderada se aplica la fórmula que se encuentra en la figura 4.2. Las calificaciones promedias se reflejan en el cuadro 4.4. Como se puede observar en el mismo cuadro, frente a los criterios previamente establecidos, Docker se puede considerar la mejor tecnología de virtualización en el proyecto actual con un puntaje de $4{,}167$, con medio punto más que su competencia mas cercana Xen, $3{,}667$ y $\frac{5}{6}$ de un punto más que KVM, $3{,}333$, que está en último lugar.

Servidor de Virtualización Virtualizado

Frente las desventajas/problemas que tiene Docker, se ha elegido a Kubernetes como una tecnología complementaria para la orquestación con la nube, y brindar a futuro una mayor independencia a la plataforma EduNube frente a cualquier hipervisor en particular. El otro candidato considerado para esta capa intermedia de orquestación de nube, entre EduNube y los respectivos hipervisores que se puede utilizar, fue OpenStack debido a que es una solución muy completa para el problema actual con más características que de los Kubernetes, pero el mismo, por su gran lista de características es muy pesado y no entraría adecuadamente en la infrastructura actual de la Universidad Técnica Particular de Loja, entonces para ofrecer una solución que cumpla con el contexto actual de la universidad, los Kubernetes ofrecen una buena alternativa que, aunque sea relativamente liviana, ofrece las características necesarias para implementar una solución de ejecucion que sea eficaz y eficiente en la infrastructura actual. Kubernetes ocupa una arquitectura cliente-servidor, motivo por el cual se lo instala en dos fases; primero el cliente y después de forma

virtualizada, para ayudar a mitigar algunos de los riesgos asociados con Docker, el servidor.

Kubernetes La instalación y uso de Kubernetes se encuentra en el Anexo C.

4.1.5. Plantillas de Máquinas Virtuales (Docker)

Para la ejecución de código en Docker, es necesario tener un contenedor de Docker que cumpla con las dependencias requeridas al momento de la ejecución. Los contenedores están basados en Debian y en Alpine Linux.

Sistema Operativo: Debian GNU/Linux

Se considera Debian como una buena base para los contenedores debido a que es bastante conocido, estable, seguro y tiene una amplitd de paquetes que se pueden instalar directamente desde sus repositorios oficiales.

Sistema Operativo: Alpine GNU/Linux

Alpine Linux es muy utilizado en el mundo de los contenedores (por ejemplo con Docker) y en la nube en general porque es un sistema operativo minimalista que lleva menor peso. Aquí tiene el mismo fin, aunque aveces es mas complicado de manejar y configurar, puede que valga la pena para entornos de producción que necesitan optimizar el uso de sus recursos.

Tipo de Contenedor: Shell-Executor

El tipo de contenedor ”Shell-Executor” es para ejecutar comandos dentro de un contenedor con el shell ”sh” y forma la base para los otros tipos de contenedores. Tanto en Debian como en Alpine no hay necesidad de instalar más paquetes, ya que la imagen por defecto viene con todo lo necesario. Lo que si se tiene que definir para estos contenedores es un usuario menos privilegiado con el cual se ejecuta los comandos/código dentro del contenedor y una ubicación, un volumen desde el punto de vista de Docker, donde se puede enviar el código a ser ejecutado. La diferencia entre la implementación tanto en Debian y Alpine es la creación de un usuario con menos privilegios en adición a la imagen base que se toma. La implementacion del mismo se puede encontrar en el Anexo D.

Tipo de Contenedor: Python3-Executor

El tipo de contenedor ”Python3-Executor” es para ejecutar comandos de shell en adición a código de Python 3. Se extiende de los contenedores definidos para el ”Shell-Executor” para proveer de la funcionalidad de ejecutar comandos en un shell. El punto de entrada de ejecución es el archivo ”main.py” a nivel raíz del repositorio. Tanto Debian como Alpine requieren de la instalación de Python 3, ya que la versión de Python por defecto es 2.7 que actualmente debería considerarsela prácticamente obsoleta. Docker requiere que se vuelva a definir los volúmenes y para la adecuada ejecución del código también se define la ubicación inicial de ejecución como el mismo volumen que se monta desde un sistema de ficheros externo al contenedor. La implementacion del mismo se puede encontrar en el Anexo D.

Tipo de Contenedor: PostgreSQL-Executor

El tipo de contenedor "PostgreSQL-Executor" es para ejecutar comandos de shell en referencia a SQL de PostgreSQL. Se extiende de los contenedores definidos para el "Shell-Executor" para proveer la funcionalidad de ejecutar comandos en un shell. El archivo "init.sql" se ejecuta primero y por lo tanto puede ser utilizado para generar y llenar la base de datos. El archivo "main.sql" se ejecuta después y por lo tanto podría ser útil para ejecutar comandos SQL definidos por el usuario. Tanto Debian como Alpine requieren de la instalación de PostgreSQL, tanto cliente como servidor. Docker requiere que se vuelva a definir los volúmenes y para la adecuada ejecución del código, también se define la ubicación inicial de ejecución como el mismo volumen que se monta desde un sistema de ficheros externo al contenedor. La implementacion del mismo se puede encontrar en el Anexo D.

4.2. GitEDU

La gestión de dependencias del sistema GitEdu se maneja con un entorno virtual y el gestor de paquetes de python pip. El siguiente es un script de bash diseñado para detectar si es que existe el entorno virtual (lo crea en caso que no exista) y después instala los requerimientos faltantes como son especificados en un archivo aparte "requirements.txt" en el que se encuentran las librerías necesarias y sus versiones.

```
# run with 'source activate.sh'
if [ ! -d env ]; then
        virtualenv --python=python3 env
fi
source env/bin/activate
pip3 install -r requirements.txt
```

El script anterior se ejecuta desde el raíz del repositorio con el comando:

```
source activate.sh
```

Con el entorno creado y activado se puede proceder a instalar cualquier dependencia necesaria que no ha sido instalado previamente:

```
pip install <nombre_dependencia>
```

Para el presente proyecto se instalaron las siguientes dependencias:

- Django (1.10.6) como framework para el desarrollo del backend.
- Psycopg2 (2.7.1) como librería para conectarse a bases de datos PostgreSQL.
- Python-GitLab (0.20) como librería para consumir el API de GitLab.
- ipython (6.1.0) como interprete interactivo de Python para hacer pruebas en el curso del desarrollo.

Con las dependencias instaladas o con cada cambio que se realice de las dependencias, se ejecuta el siguiente comando para actualizar la lista de las mismas:

```
pip freeze > requirements.txt
```

Se inicia el proyecto de Django con el comando:

```
django-admin startproject GitEdu
```

También antes de iniciar el desarrollo debe estar configurada la base de datos para proceder a ocuparla. Esto se documenta en el Anexo D.

Ahora se puede migrar las tablas iniciales de Django:

```
cd GitEDU
python manage.py makemigrations
python manage.py migrate
```

También creamos un superusuario de Django para temas administrativos:

```
python manage.py createsuperuser
```

También se puede crear el app inicial (para lógica del editor de código):

```
python manage.py startapp ideApp
```

Y de paso se lo agrega a INSTALLED_APPS una linea 'ideApp' en el settings.py para que las tablas definidos a futuro en models.py también se migren con los demás migraciones.

Para resolver un problema de zonas de tiempo, se ha desactivado esta característica de Django con la siguiente línea en el settings.py:

```
USE_TZ = False
```

La creacioón de la base de datos NoSQL y su configuración con Django se encuentra en el Anexo D.

Automatización del Levantamiento del Servicios / DevStack

Para el manejo de dependencias, inicialmente se lo manejó unicamente con Pip y Virtualenv hasta que se descubrió otra herramienta que combina la funcionalidad de ambos en un solo commando: `pipenv`. El mismo agrega funcionalidades adicionales como validacion del ambiente (sistema operativo / arquitectura / versión exacta de Python) y validación de paquetes (mediante sus hash SHA256). Es por ello que se vio la importancia en cambiar la modalidad con que se llevaba las dependencias del proyecto, aunque también se sigue apoyando el método anterior para quienes no realizan la instalacion de pipenv con pip como root.

Para el fácil levantamiento del DevStack, se realizó dos scripts, de los cuales cada uno detecta la presencia de pipenv para utilizarlo si existe o no existe ocupar la modalidad clásica de virtualenv con pip para gestionar las dependencias. Lo que diferencia los dos scripts es que uno, después de validar/instalar y activar el entorno, ejecuta el servidor en el puerto designado para producción (`runserver`) mientras que el otro (`runshell`) abre el shell de Django después de adecuar el entorno.

4.2.1. Autenticación Clásica

Primero se crea una nueva aplicación para autenticación clásica:

```
python manage.py startapp authApp
```

Y de paso se lo agrega a INSTALLED_APPS en una línea 'authApp' en el settings.py para que las tablas definidos a futuro en models.py también se migren con las demás migraciones.

Para la implementación de autenticación clásica, se reutilizó el módulo con el mismo propósito del proyecto anterior GitEduERP. A esta se aumentó la funcionalidad adicional, que es que desde el settings se pueda activar o desactivar la funcionalidad de permitir a los usuarios registrarse a través del atributo ENABLE_REGISTRATION.

Para el registro de usuarios, se considera que solo debe permitirse (en caso de que sea habilitado por la administración) el registro de estudiantes y docentes pero no de administradores, motivo por el cual se ha considerado duplicar los formularios de registro y ofrecer uno para estudiantes y otro para docentes. Los mismos se pueden habilitar y deshabilitar con los atributos ENABLE_STUDENT_REGISTRATION y ENABLE_TEACHER_REGISTRATION respectivamente.

4.2.2. Autenticación por LTI

Se documenta el proceso de levantamiento de autenticación por LTI en el Anexo D.

Validación

Se levanta la aplicación (en `http://localhost:8000/` con:

```
python manage.py runserver
```

La máquina virtual de Moodle también debe estar levantada para navegar a `http://10.10.10.10/`, iniciar sesión como administrador, e ir a la parte administrativa, a la pestaña de ”development”, seleccionar ”debugging”, y poner ”debug messages” a nivel ”developer”. Ahora podemos volver a la parte de administración, volver a seleccionar la pestaña de ”development” y crear un curso de prueba. Vamos a crear un curso pequeño (10 MB) con el nombre corto ”Prog_I” y nombre largo / descripción ”Introducción a la Programación”. Una vez que se carga el curso, activamos el modo de editar y agregamos una actividad externa.

Se da un nombre a la actividad, en este caso ”Prueba 1 Programación”. La mejor forma de poder reutilizar la herramienta externa en varias actividades es de crear una herramienta ”Preconfigurada”. Las opciones elegidas para la misma fueron los siguientes:

Nombre GitEdu

URL `http://localhost:8000/lti/launch`

Descripción Edit Code Online

Llave de Consumidor GitEduLMS_Playground

Secreto Compartido b2e0158c3cb4ddb0202d

Parámetros Adicionales lo dejamos en blanco

Contenedor de Lanzamiento Nueva ventana

Compartir Nombre Siempre

Compartir Correo Electrónico Siempre

Aceptar Notas Siempre

Con la herramienta preconfigurada realizada, se puede seleccionarlo así no mas para completar la integración. Donde se creó la actividad debe haber un enlace con el nombre del mismo, en este caso ”Prueba 1 Programación”. Al abrir el enlace se debe llevar a la aplicación con los siguiente puntos interesantes:

1. se encuentra autenticado con el mismo usuario de Moodle (incluyendo sus roles como ”administrador”, ”instructor”, ”estudiante”, etc. . .)
2. hay contexto de:
   a) el curso de origen de Moodle
   b) la actividad de origen de Moodle (no es contexto completo, un problema que se tendrá que resolver en el curso del desarrollo)
   c) la llave de consumidor ocupada

Si, dentro del mismo curso creamos una segunda actividad con todo lo mismo, pero solo cambiando el nombre a ”Prueba 2 Programación”, se identifica que sí se logra distinguir entre actividades.

4.2.3. Integración de dos esquemas de autenticación

Los dos schemas de autenticación implementados, sea la forma clásica con nombre de usuario y contraseña correspondiente o por LTI donde una aplicación externa autentica un usuario autenticado previamente por el mismo, tienen el mismo fin de permitir que el sistema conozca quién esta accediendo al sistema para actuar con el comportamiento adecuado y en base a ello proveer seguridad y funcionalidad a todos los usuarios según su rol.

Se propone tres tablas (EquivalentUser, AuthenticationType, UserAuthentication) en la base de datos expresados como modelos de Django, los mismos que se encargaran de generar las tablas adecuadas en la base de datos a través de su ORM interno.

La tabla EquivalentUser relaciona a los usuarios autenticados de forma clásica con usuarios autenticados por LTI LTI (porque por cada método de autenticación se crea un nuevo usuario para su uso interno) con el fin de permitir relacionar distintos usuarios dentro del sistema quienes representan el mismo individuo natural. La idea es por ejemplo, si un docente se autentica por LTI y también por usuario y contraseña, para el sistema esta ocupando dos usuarios diferentes pero para el docente esta accediendo el mismo sistema y por lo tanto el comportamiento del sistema, sin tomar en cuenta como se autentica, debe ser igual y debe tener acceso a los mismos contenidos. La tabla AuthenticationType es un catálogo de las formas de autenticación que suporta el sistema y por su naturaleza de catálogo se llena con las migraciones. Y finalmente la tabla UserAuthentication documenta el tipo de autenticación asociada con un usuario para dar el contexto al sistema que necesita para manejar distintos datos de los diferentes tipos de usuarios. La migracion escrita a mano para llenar la tabla AuthenticationType se puede encontrar en el Anexo D.

4.2.4. Accesibilidad a Datos de Autenticación de LTI

Las vistas y settings definidas para exponer datos de la autenticacion por LTI se presenta en el AnexoD.

4.2.5. Establecer un Modelo de Base de Datos

Para empezar con el desarrollo de la parte fuerte del sistema GitEDU, es importante considerar su modelo de base de datos preliminar. La parte más importante de ello son las tablas o entidades y las relaciones entre las mismas. Para esto se ha considerado 3 grupos de funcionalidades importantes:

1. El aspecto social y con ello varios roles con los que se puede manejar a los usuarios dentro del sistema. Esto se representa con el app 'socialApp'.

2. El aspecto académico, se refiere a cómo se llevan las materias académicas dentro del sistema. Esto se representa con el app 'academicApp'.

3. El aspecto de las calificaciones y con ello la forma en que se llevan las calificaciones que los estudiantes obtienen en las materias. Este aspecto es muy relacionado con el tema anterior de la parte académica y la línea que las divide es un poco subjetiva ya que por lo mismo hecho no se ha optado por la creación de tablas adicionales para separar estos dos aspectos, de todas maneras donde ha sido posible, se ha realizado la división. Este aspecto se representa con el app 'gradesApp'.

```
python manage.py startapp socialApp
python manage.py startapp academicApp
python manage.py startapp gradesApp
```

Cada una de las anteriores aplicaciones han sido agregadas al INSTALLED_APPS del settings para que Django reconozca la necesidad de tomar en cuenta los modelos de cada uno.

Los modelos iniciales del socialApp solo definen distintos roles sociales dentro del sistema y ocupan cuatro tablas:

1. Person

2. Student

3. Teacher

4. Administrator

Se esta tomando en cuenta que la tabla de usuarios de Django ya lleva por defecto muchos detalles de los usuarios como sus nombres y apellidos, por lo tanto no hay necesidad de replicar estos datos. Esta jerarquía de clases viene a ser la infraestructura necesaria para definir a cada uno de los usuarios en caso de que sea necesario.

El modelo inicial de academicApp permite a los docentes y administradores definir las materias ofrecidas (con paralelos y metadatos como categoría del componente), ya sean los mismos docentes o sus estudiantes, adicionalmente de proveer de la infraestructura de datos necesaria para que un docente pueda definir su libreta de calificaciones a una materia, en base a un sistema relativo o un sistema de promedios ponderados (el sistema puede calcular en base al sistema relativo). Esto involucra nueve tablas en la base de datos:

1. ClassSubject

2. Course

3. Section

4. Classroom

5. ClassroomTeacher

6. ClassroomStudent

7. AcademicCategory

8. AcademicSubCategory
9. AcademicAssignment

ClassSubject define el tipo de materia que es, por ejemplo ”Programación” o ”Base de Datos”. Course define una materia ofrecida, por ejemplo ”Introducción a la Programación”. Section define paralelos ofrecidos de forma general, por ejemplo ”A”, ”B”, ”C”, etc... Classroom une las dos tablas anteriores de materia de oferta con paralelo para definir aulas de una materia. ClassroomTeacher define el docente asignado a una aula y el peso que tiene el docente en la nota final de los estudiantes de esta aula. Esto es con un fin de permitir varios docentes con varios pesos dentro de la misma aula emitiendo calificaciones distintas para estudiantes previo a un promedio ponderado del estudiante. ClassroomStudent relaciona estudiantes y aulas. AcademicCategory provee docentes con la oportunidad de disponer de una herramienta para organizar las calificaciones de sus estudiantes en categorías con distintos pesos, por ejemplo ”Exámenes”, ”Deberes”, ”Talleres”, etc... AcademicSubCategory viene de la misma linea para permitir subdivisiones de las categorías. AcademicAssignment es la abstracción que se da a los itemes calificadas dentro de una subcatagoría.

El modelo inicial de gradesApp permite persistir notas para cada estudiante bajo el modelo de tres niveles definido previamente de categorías, subcatagorías y itemes de subcatagorías con tres tablas:

1. StudentCategoryGrade
2. StudentSubCategoryGrade
3. StudentAssignmentGrade

4.2.6. Editor de Código en Linea

Una de las funcionalidades principales del sistema GitEDU es la capacidad de editar código en linea, la lógica del mismo se realiza en la app 'ideApp'.

Con relación al editor de código en linea, inicialmente se empezó con el trabajo realizado anteriormente en el proyecto GitEduERP y en base al mismo se fue expandiendo las características necesarias en el nuevo sistema. Durante el proceso del desarrollo, se observó una necesidad de desarrollar la mayoría de los componentes desde cero y con el mismo se pudo dar mejoras a la propuesta anterior. A diferencia de GitEduERP, GitEDU espera ofrecer:

- Un modelo de permisos más robusto y flexible basado en grupos y usuarios.
- Un sistema de persistencia de código el cual ofrece la siguientes características:
  - Ser extensible con una programación orientada a objetos.
  - Manejar versiones de código con un sistema de control de versiones interna llevada por archivos completos
  - Manejo de jerarquías de sistemas de persistencia para llevar varias sistemas de persistencia en paralelo
- Integración con sistemas externas que permiten la ejecución de código.

Modelo de Base de Dato para Editor de Código

Para el editor de código en linea, es necesaria extender la funcionalidad del modelo de base de datos introducido previamente. Primero se agrega dos entidades a socialApp que representan grupos de personas (usuarios) y membresía dentro de estos mismos grupos con dos tablas:

1. Group
2. GroupMembership

Dentro de ideApp se crea cuatro entidades para representar repositorios (una colección de archivos asociados con un proyecto), el mismo tiene capacidad de un usuario y, de forma opcional, un grupo dueño quien esta encargado de la administración del mismo, una representacion de archivos, misma que entre sus metadatos esta un lenguaje asociado para ayudar con el manejo del mismo a nivel de editor y a nivel de otros tipo de gestión (como ejecución y compilación), y finalmente membresía de personas y grupos en repositorios con una finalidad de llevar mas adelante un sistema de permisos para restringir acceso y/u otras acciones sobre código para usuarios no autorizados. Estas entidades como tablas en la base de datos son:

1. Repository
2. File
3. RepositoryPersonMembership
4. RepositoryGroupMembership

4.2.7. Persistencia de Código

Para la Persistencia de Código se considera dos backends con una posibilidad de introducir más a futuro:

- MongoDB como base de datos NoSQL para acceso rápido y recuperación de código persistido de una manera similar a GitEduERP.
- GitLab CE, como servidor de Git, manejado con una combinación de la API del mismo y comandos de Git para el manejo de cambios en repositorios.

Para el mismo es necesario manejar una API interna estandarizada. La solución propuesta utiliza programación orientado a objetos y un modelo estándar de namespace (como usuario o grupo) que contiene repositorios (que representan proyectos o conjuntos lógicos de archivos) y a su vez contienen dichos archivos. La clase base para estandarización se llama CodePersistenceBackend y se encuentra en generics.py del paquete ideApp.CodePersistenceBackends. Para el manejo de datos dentro del API interna, se estandariza los objetos de datos con una serie de clases genéricas, las cuales pueden ser sobrescritas en cualquier backend con una finalidad de agregar o cambiar funcionalidad existente. Los mismos también se encuentran en el mismo archivo de generics.py y tambien dentro del Apéndice F.

En el settings, hay una necesidad de definir los backends de código y su prioridad:

```
CODE_PERSISTENCE_BACKENDS = {
    'mongodb': {
        'use': True,
        'backend': 'ideApp.CodePersistenceBackends.MongoDB.
            backend.MongoDBCodePersistenceBackend',
        'connection_profiles': NOSQL_DATABASES,
        'connection_profile': 'nosql',
    },
    'gitlab': {
        'use': False,
        'backend': 'ideApp.CodePersistenceBackends.GitLab.
            backend.GitLabCodePersistenceBackend',
        'connection_profiles': GITLAB_SERVERS,
        'connection_profile': GITLAB_DEFAULT_SERVER,
    }
}

CODE_PERSISTENCE_BACKEND_READ_PREFERENCE = ['mongodb']
CODE_PERSISTENCE_BACKEND_WRITE_OUT = ['mongodb']

MONGODB_CONNECT_TO = 'mongodb'
GITLAB_CONNECT_TO = 'gitlab'
```

Por el momento se ha optado por dejar desactivado el backend de GitLab ya que el mismo no esta implementado para ser utilizado.

El gestor de persistencia de código carga todos los backends de persistencia de código que encuentra en el settings para gestionar las conexiones a las mismas. El mismo esta implementado asi mismo como un CodePersistenceBackend para permitir que se puede remplazar fácilmente a futuro si se da el caso. Implementa dos tipos de persistencia:

- Lectura (read)
- Escritura (write)

Y permite para toda operación que se especifique sobre cual grupo se debe realizar la operación. Estos grupos se definen en el settings con los variables:

- CODE_PERSISTENCE_BACKEND_READ_PREFERENCE (que también da preferencia del orden en que se debe leer de los backends de persistencia de código).
- CODE_PERSISTENCE_BACKEND_WRITE_OUT que define cuales son y da el orden en que se debe escribir a los backends de persistencia de código.

El código completo de este backend se encuentra en el paquete ideApp.CodePersistenceBackends en el archivo backend_mánager.py con el nombre de clase CodePersistenceBackendManager que se puede encontrar en el Apéndice F.

También se define el gestor del backend de persistencia de código en el Settings y el código necesario para cargarlo en el momento adecuado:

```
CODE_PERSISTENCE_BACKEND_MANAGER_CLASS = 'ideApp.
    CodePersistenceBackends.backend_manager.
    CodePersistenceBackendManager'
```

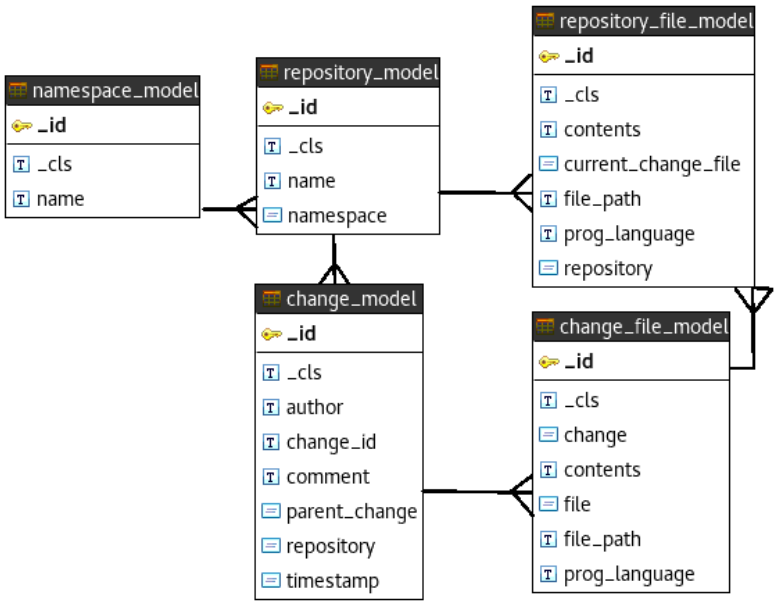

FIGURA 4.3: Modelo de Base de Datos en MongoDB.

```
def load_code_persistence_backend_manager(load_class=
    CODE_PERSISTENCE_BACKEND_MANAGER_CLASS):
    try:
        module_path, class_name = load_class.rsplit('.', 1)
    except ValueError:
        msg = "%s doesn't look like a module path" %
            load_class
        six.reraise(ImportError, ImportError(msg), sys.
            exc_info()[2])
    mod = importlib.import_module(module_path)
    backend_manager_class = None
    try:
        backend_manager_class = getattr(mod, class_name)
    except AttributeError:
        msg = 'Module "%s" does not define a "%s" attribute/
            class' %(
             module_path, class_name)
        six.reraise(ImportError, ImportError(msg), sys.
            exc_info()[2])
    return backend_manager_class()
```

MongoDB

Para la persistencia de código en MongoDB se define unos modelos de base de datos en Python para su utilización con el ORM PyModm. Se presenta el modelo de base de datos para MongoDB en la figura 4.3.

El NamespaceModel solo consiste de:

- el nombre del Namespace
- el ID que se genera por defecto

Esto representa los espacios donde usuarios y grupos pueden crear repositorios. Se considera que los Namespace deben existir de forma única.

El RepositoryModel representa un repositorio y tiene:

- un nombre
- permanece a un Namespace
- el ID que se genera por defecto

Cada par de Namespace y Repositorio debe ser una combinación única.

El próximo modelo, RepositoryFileModel, representa un archivo ubicado dentro de un repositorio. Sus atributos son:

- contenidos del archivo
- el repositorio a que permanece
- el lenguaje de programación para que el editor puede cambiar su comportamiento para el lenguaje específico
- la ubicación del archivo dentro del repositorio incluyendo su nombre
- un archivo de cambio que representa el estado actual y la última modificación realizada en el mismo, de lo cual se presenta más adelante.

El ChangeModel representa un cambio realizado en el repositorio, como los commits de Git. Tiene atributos como:

- una ID que se lo maneja la aplicación (que viene a ser un hash SHA1, inspirado por Git, de los contenidos y metadatos del mismo cambio)
- un comentario
- un autor
- una estampa de tiempo que tiene tanto la hora como la fecha en que se realizó el cambio
- el repositorio en que se hizo el cambio
- una referencia al cambio anterior para que se puede construir una historial de cambios.

ChangeFileModel se asemeja a RepositoryFileModel ya que representa archivos, pero en este caso son archivos modificados dentro de un cambio. Por lo tanto lleva los mismos atributos que RepositoryFile solo que en lugar de tener referencias a un repositorio directamente se referencia a un cambio realizado (el cual tiene metadatos como un comentario, autor, cambio previo y momento de modificación) y al archivo del repositorio que modifica (de esta forma puede modificar también la dirección de los archivos quienes no habría otra forma de buscarlos).

Finalmente el TemporaryChangeFileModel no se utiliza actualmente pero como los dos FileModels vistos anteriormente, esta lleva atributos básicos de un archivo. Originalmente se lo planteo para realizar una operación de intermedio como la fase

del Git Stage en donde se selecciona previo a un commit los cambios que van a estar incluidos dentro del mismo. Pero al final, aunque el modelo de datos esta para suportar varios archivos cambiados dentro de un solo cambio, no esta implementado así. Por temas de tiempo, retraso y alcance, se ha optado por llevar un mini control de versiones de cada archivo dentro de un repositorio, así que en realidad cada cambio en el repositorio solo refleja un archivo cambiado de forma independiente y no hay la necesidad de aumentar la complejidad con dicho paso intermedio; lo que este modelo representaba.

GitLab

Al settings agregamos las siguientes lineas para configurar la conexión con GitLab:

```
GITLAB_DEFAULT_SERVER = '1'

GITLAB_SERVERS = {
    '1': {
        'WITH_TOKEN': True,
        'WITH_CRED': False,
        'API_PROTOCOL': 'http://',
        'API_PORT': '',  # por defecto:
                         # :22 para SSH,
                         # :443 para HTTPS,
                         # :80 para HTTP
        'HOST': '10.10.10.11',
        'SSH_PORT': 22,
        'HTTP_PORT': 80,
        'HTTPS_PORT': 443,
        'USER': "GitEDU",
        'PASSWORD': 'GitEDU2017',
        # expira el 31 de marzo 2018:
        'TOKEN': 'JqMzkgDNvhZ7ofdPa5z5',
        # nunca expira, pero el nivel de
        # acceso es menor:
        # 'TOKEN': 'TrCfvrdsXzpLFETyc7Q5',
    }
}
```

La implementacion de la conexion al API de GitLab se presenta en el Anexo D.

Aunque originalmente se planifico el uso de GitLab como backend de persistencia de codigo (en repositorios, a diferencia de GitEduERP que ocupó snippets), finalmente no se ocupó el mismo por cinco razones:

1. La complejidad del API frente la documentación no es tan clara para el community edition (esta mucho mejor documentado el enterprise edition al cual no se tuvo acceso, ni en el ambiente de desarollo, ni de produccion), y eso genera un factor limitante frente al tiempo disponible para la implementacion de la misma.

2. Un gran inestabilidad del servicio, Gitlab CE.

3. La falta de recursos en los ambientes de produccion y de desarollo que empeora problemas de estabilidad.

4. La simplicidad de utilizar el sistema de ficheros local directamente con GitWeb para estar disponible por HTTP.

5. Frente a las necesidades actual del proyecto, mientras no sea necesario exponer el servidor de Git a los usuarios finales, no hay necesidad de un servidor de Git de tanta complejidad como el Gitlab.

4.2.8. Aspectos Sociales

Para temas de organización, se ha considerado que aquellos componentes de la aplicación que tienen que ver con interacción social deben existir de forma independiente de las funcionalidades principales de la misma aplicación. Es con este fin que los componentes del mismo se desarrollaron en un "app" distinto, "socialApp", como fue creada previamente. A futuro se considera el desarollo del mismo para promover interaccion entre estudiantes y tambien otra herramienta de comunicacion e colaboracion entre estudiantes y sus professores que viene a ser un elemento critico para el proceso educativo.

4.3. EduNube

La gestión de dependencias del sistema EduNube se maneja con un entorno virtual y el gestor de paquetes de python pip. El siguiente es un script de bash diseñado a detectar si es que existe el entorno virtual (lo crea en caso de que no existe) y después instala los requerimientos faltantes como son especificados en un archivo aparte "requirements.en.txt" que da librerías y sus versiones.

```
#! /usr/bin/head -n 2
# run with 'source activate-en.sh'
PROJECT=en
ENV_DIR=env-$PROJECT
if [ ! -d $ENV_DIR ]; then
        virtualenv --python=python3 $ENV_DIR
fi
source $ENV_DIR/bin/activate
pip3 install -r requirements.$PROJECT.txt
```

El script anterior se ejecuta desde el raiz del repositorio con el comando:

```
source activate-en.sh
```

Con el entorno creado y activado se puede proceder a instalar cualquier dependencia necesaria que no ha sido instalado previamente:

```
pip3 install django==1.10.6 psycopg2==2.7.1 pymodm==0.4.0
    ipython
```

Para el presente proyecto se instalaron las siguientes dependencias:

- Django (1.10.6) como framework para el desarrollo del backend.
- Psycopg2 (2.7.1) como librería para conectarse a bases de datos PostgreSQL.
- ipython (6.1.0) como interprete interactivo de Python para hacer pruebas en el curso del desarrollo.

Con las dependencias instaladas o con cada cambio que se realiza de dependencias se ejecuta el siguiente comando para actualizar la lista de los mismos:

```
pip freeze > requirements.txt
```

Se inicia el proyecto de Django con el comando:

```
django-admin startproject EduNube
```

También antes de iniciar el desarrollo debe estar configurado el base de datos para ser usada. Esta configuracion lo puede encontrar en el Anexo E.

Ahora se puede migrar las tablas iniciales de Django:

```
cd EduNube
python manage.py makemigrations
python manage.py migrate
```

También creamos un superusuario de Django para temas administrativos:

```
python manage.py createsuperuser
```

También se puede crear el app inicial (para lógica del editor de código):

```
python manage.py startapp apiApp
```

Y de paso se lo agrega a INSTALLED_APPS una linea 'apiApp' en el settings.py para que sus tablas definidos a futuro en models.py también se migran con las demás migraciones.

Para resolver un problema de zonas de tiempo, se ha desactivado esta característica de Django con el siguiente linea en el settings.py:

```
USE_TZ = False
```

La configuracion de base de datos no relacional se encuentra en el Anexo E.

4.3.1. Autenticación

El diseño planteado para el sistema EduNube propone dos tipos de autenticación:

- Clasica para usuarios humanas, orientada solo a la administracion del mismo sistema.
- Basada en Criptografía Asimétrica para ofrecer mayor seguridad y permitir integración con otros sistemas como GitEDU.

Autenticación Clasica

Para la administración del sistema por operador humano se considera necesario la autenticación clásica mediante usuario y contraseña. Para la misma se uso la implementación del framework Django y código reutilizado de GitEDU que tiene el mismo propósito.

Autenticación basado en Criptografía Asimétrica

Para ofrecer mayor seguridad al sistema de ejecución de código, EduNube, se considera más óptimo utilizar una sistema de autenticación basado en criptografía asimétrica de una manera similar a las llaves de SSH, donde el servidor tiene la

oportunidad de conocer bien la identidad del cliente. Aunque no se ha logrado encontrar una implementación factible para la solución planteada, se considera como una solución utilizar una especie de API token similar a otros sistemas. Para generar el token, se ha logrado encontrar un estándar, JSON Web Tokens, que tiene la finalidad de ayudar al servidor identificar y verificar la validez de un cliente desconfiable mediante datos que el servidor mismo cifra con un tipo de llave privada.

La logica para gestionar los API tokens se encuentra en el Anexo E.

4.3.2. Ejecución de Código en Linea

La ejecucion de codigo se divide en dos componentes principales que se complementan para dar funcionalidad completa frente a las necesidades requeridas. Primero en el sistema de plantillas se define y se controla el ambiente en el cual se ejecuta el codigo y despues en el curso del ciclo de vida del codigo, antes, durante y despues de su ejecuccion se maneja mediante una API externa.

4.3.3. Metadatos de Plantillas

Para ofrecer mayor seguridad y funcionalidad a la ejecucion de codigo en el sistema, se basa en lo que son plantillas de ejecucion diseñados especifcamente para suportar herencia por parte de los usuarios. Estos utilizan varios mecanismos para llegar a este fin, incluyiendo listas negras de archivos que no se puede sobre escribir plantillas que extienden la plantillas (`.edunubeignore` y `.edunubeignore.children`), listas blancas de que se debe incluir de archivos cuando recien se crea una nueva plantilla o proyecto en base a la plantilla (.templateinclude), y tokens cifrados para validar repositorios frente un registro centralizado en la base de datos que pertenece al servicio de ejecuccion y tambien la ubicacion del repositorio padre. La herrencia se basa en la creacion de un nuevo repositorio donde se copia uno por uno los contenidos (archivos) de cada repositorio, desde el inicio de la cadena de herrencia (una plantilla hecha por la administracion para un lenguaje en especifico) hasta el final con el repositorio del usuario, mediante llamadas de `rsync` y el uso de las listas negras para controlar lo que no se puede copiar en cada fase. De esta forma no se toca la historia original del repositorio del usuario y se puede controlar con mayor facilidad el nivel de acceso que tiene el usuario final sobre lo que esta ejecutando.

.edunubeignore

El `.edunubeignore` de un repositorio especifica los archivos de un mismo repositorio que no se deben copiar a sus repositorios hijos, y que tampoco se puede heredar de ningun otro repositorio que le sigue en la herencia de repositorios. Un ejemplo de los mismos serian archivos propios de Git.

.edunubeignore.children El `.edunubeignore.children` tiene una funcionalidad similar al `.edunubeignore` pero permite el copia de archivos desde la repositorio actual, pero no de sus repositorios hereditarios. Un ejemplo de los mismos que se involucran en este conjunto de metadatos serian los shell scripts base que ocupa el backend de virtualizacion para iniciar la ejecuccion.

.repospec

El `.repospec` es un mecanismo que utiliza JWT para dar mayor seguridad de emitir un token que solo puede decifrar el servidor para determinar el repositorio padre de un repositorio. Esto ayuda prevenir ataques contra plantillas de ejecucion en base a la inyeccion de plantillas maliciosas porque realizar un ataque de este tipo requiere que se rompe la cadena de confianza. Este mecanismo todavia se puede mejorar, el cual se discuta en trabajos futuros.

.templateinclude

El `.templateinclude` es un archivo de metadatos exclusivo a las plantillas de ejecuccion que determine cual archivos de la plantilla se debe copiar al nuevo repositorio. Esto archivo, combinado con la manera en que el profesor realiza el repositorio de la plantilla determina tanto el entorno de ejecucion (donde atraves del mecanismo del `.edunube/.edunubeignore.children` los archivos que un estudiante no puede sobreescribir de la plantilla) y del estado inicial del proyecto/archivos expuestos al estudiante (atraves del `.templateinclude`). Un ejemplo del mismo, serian los archivos de un cierto lenguaje de programacion que deben ser expuestas al usuario final para que el mismo los puede editar.

4.3.4. API Externa

Similar al servicio de GitServerHTTPEndpoint, EduNube tambien ofrece una API externa que requiere un token de autenticacion para su funcionamiento como se lo detalle a continuacion. Ademas se considera critico tres componentes del API:

1. API de RepoSpec donde se implementa la creacion y actualizacion de los RepoSpecs de tal forma que al momento de enviar un repositorio a ser ejecutado, se puede validar que el repositorio esta autorizado para ejecucion y de donde previene la herencia del mismo de tal forma que el usuario final no tenga mucho libertad a modificar el mismo y realizar cambios indebidos al entorno de ejecucion.

2. API de Ejecuccion donde se solita ejecucion de codigo. Esto agrega la solicitud a la cola de trabajo para que el mismo puede ser ejecutado cuando hay disponibilidad de los recursos limitados.

3. API de Trabajos (por Lotes) es para soliticar el estado actual de un trabajo y a lo que finaliza, los resultados del mismo.

Autenticacion para el API

Igual que el sistema de GitServerHTTPEndpoint, EduNube utiliza JWT tokens para autorizar y validar llamadas a su API. Estos tokens los pueden administrar usuarios con los permisos adecuados, o por defecto (si no se crea los permisos), solo por superusuarios, en el URI `http://10.10.10.1:8010/auth/tokens`. Dentro de aquello interfaz existe la posibilidad de crear tokens, visualizar los mismos (`http://10.10.10.1:8010/auth/token/<app_name>.json`), actualizar un token (`http://10.10.10.1:8010/auth/token/edit/<app_name>/`), cambiar el secreto (llave privado) de un token (`http://10.10.10.1:8010/auth/token/secret/<app_name>/`), y eliminar un token (`http://10.10.10.1:8010/auth/token/delete/<app_name>/`).

Para dar conectividad a este servicio desde otro servicio, se copia las respectivas clientes del mismo, llamadas `execute_status_example_client.py` y `repospec_example_client.py` respectivamente y se agrega las siguientes lineas al settings:

```
EDUNUBE_CONFIG = {
    "protocol": "http",
    "host": "10.10.10.1",
    "port": 8010,
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhcHBf"
             "bmFtZSI6IkdpdEVEVSIsImV4cGlyZXMiOmZhbHNlLCJjc"
             "mVhdGVkX2RhdGUiOiIyMDE3LTExLTEyIDE3OjMzOjE3Lj"
             "Y0MTY4MSIsImVkaXRfZGF0ZSI6IjIwMTctMTEtMTIgMjA"
             "6MzY6NDAuMjc5NDAyIn0.825oh2rZUlIPZFaP_UbYPDpd"
             "sXTE0XCaNsia-3NnGuc"
}
```

RepoSpec API

Los URIs del API de RepoSpecs de EduNube toma la forma general de:
`POST /api/repospec/<operacion>`
donde las operaciones posibles son los siguientes:

- `create` para crear un nuevo RepoSpec. Parametros del `POST` son:

  token con el API token que se genera previamente.

  repo con el nombre del repositorio, actualmente no se utiliza este valor, solo se lo persiste internamente, pero a futuro se deberia agregarlos a las validaciones de los RepoSpec para prevenir ataques de robo de RepoSpec de un repositorio para otro.

  parent *de forma opcional* con el URI a traves del cual se puede acceder al repositorio padre del actual, eso forma la base para la herencia de los repositorios. Solo en el caso de un repositorio plantilla de ejecuccion que no hereda de ninguna otra plantilla,es permitido que este vacio, en cualquier otro caso, incluyendo todos los casos estudiantiles, debe ser obligatorio un valor valido. Pero dicha validacion queda para ser implementada a futuro.

- `get` para leer un RepoSpec existente. Parametros del `POST` son:

  token con el API token que se genera previamente.

  repo *de forma opcional* con el nombre del repositorio actual para la busqueda. Dentro de los parametros del `POST` debe estar minimo este parametro o el `repospec_token`.

  repospec_token *de forma opcional* con el token del repospec actual para la busqueda. Dentro de los parametros del `POST` debe estar minimo este parametro o el `repo`.

- `get_or_create` para leer un RepoSpec existente o si no existe, crearla. Parametros del `POST` son:

  token con el API token que se genera previamente.

repo con el nombre del repositorio actual para la busqueda/creacion.

parent *de forma opcional* con el URI con el cual se puede acceder al repositorio padre del repositorio actual, eso forma la base para la herencia de los repositorios. Solo en el caso de una repositorio plantilla de ejecuccion que no hereda de ningun otra plantilla, debe ser permitido que este vacio, en cualquier otro caso, incluyendo todos los casos estudiantils, debe ser obligado un valor valido aqui. Pero dicho validacion queda para ser implementado a futuro.

repospec_token *de forma opcional* con el token del repospec actual para la busqueda.

- `edit` para modificar un RepoSpec existente. Parametros del `POST` son:

  token con el API token que se genera previamente.

  repo *de forma opcional* con el nombre del repositorio actual para la busqueda. Dentro de los parametros del `POST` debe estar minimo este parametro o el `repospec_token`.

  repospec_token *de forma opcional* con el token del repospec actual para la busqueda. Dentro de los parametros del `POST` debe estar minimo este parametro o el `repo`.

  parent *de forma opcional* con el URI con el cual se puede acceder al repositorio padre del repositorio actual, eso forma la base para la herencia de los repositorios. Solo en el caso de una repositorio plantilla de ejecuccion que no hereda de ningun otra plantilla, debe ser permitido que este vacio, en cualquier otro caso, incluyendo todos los casos estudiantils, debe ser obligado un valor valido aqui. Pero dicho validacion queda para ser implementado a futuro. Si se deja este sin valor (considerado como Null o en Python None), el comportamiento por defecto es de sobreescribir el valor anterior.

  new_repo *de forma opcional* con el nuevo nombre del repositorio actual para cambiar la misma, no incluir este parametro deja el RepoSpec con el mismo nombre de repositorio que ya tiene.

  regen_secret_key *de forma opcional* con una bandera booleano que indica si se debe generar un nuevo secreto (llave privada) para el RepoSpec. Si no se incluye el parametro o lo deja el valor en false, se lo interpeta como mantener el mismo secreto actual para cifrar el token actualizado, caso contrario se interpreta cualquier otro valor como true, y procede a cambiar el secreto.

Ejecuccion API

Los URIs del API de Ejecuccion de EduNube toma la forma general de:
`POST /api/execute/create/<lenguage_ejecutor>/<namespace>/<repositorio>/`
donde los ejecutores de lenguajes posibles son los siguientes:

shell que ejecuta comandos de terminal ocupando el interprete `/bin/sh` (puede ver el imagen de Docker `shell-code-executor` para entender su funcionamiento) dentro del cluster de Kubernetes.

python3 que ejecuta codigo de Python 3 utilizando `virtualenv` y `pip` para manejo de dependencias (puede ver el imagen de Docker `python3-code-executor` para entender su funcionamiento) dentro del cluster de Kubernetes.

postgresql que ejecuta sentencias de SQL de PostgreSQL (puede ver el imagen de Docker `postgresql-code-executor` para entender su funcionamiento) dentro del cluster de Kubernetes.

El unico parametro que toma esta parte del API es el token definida previamente ya que con el mismo se valida que la peticion de ejecuccion viene de un fuente autorizado para el mismo. El mismo API devuelva un JSON que dentro de sus parametros contiene el `id` con el cual se puede realizar consultas del estado de trabajo y ver sus resultados cuando el mismo termina.

Jobs API

Una vez que se realiza una peticion de un trabajo, se puede consultar el estatus y al finalizar se puede observar resultados del mismo con su identificador unico devuelto por el API de Ejeccucion. Estas llamadas toman una forma general de: `POST /api/execute/<operacion_de_consulta>/<lenguaje_ejecutor>/<id>/` Donde operaciones de la consulta son los siguientes:

status devuelve en formato JSON si un trabajo existe o no y en caso de que existe, si el mismo ha terminado.

result devuelve en formato JSON la salida del trabajo con la finalidad de que se puede presentar el mismo al usuario final.

y lenguaje ejecutores son los mismos que se definen para el API anterior. Actualmente el valor de lenguaje no tiene ningun efecto sobre la consulta en el backend, pero seria mala practica de programacion que se solicita el trabajo con un lenguaje y despues consulta su estado o resultado con otro ya que, a futuro, si se implementa otros backends de virtualizacion para distintas lenguaje ejecutores, puede que este parametro se convierte en lo unico que les distingue para realizar la consulta en el backend adecuado. Todo operacion contra este API debe llevar su respectivo `token` definido anteriormente para validar que realmente puede realizar estas consultas.

Passthrough de API de EduNube en GitEDU para su consumo via AJAX

Con la finalidad de que el usuario final de GitEDU tenga acceso a los APIs de Ejecuccion y de Trabajos de EduNube, se realiza un mapeo de URIs entre los dos sistemas:

Ejeccucion *GitEDU*: `POST /ide/execute/<namespace>/<repositorio>` → *EduNube*: `POST /api/execute/create/<lenguaje_ejecutor>/<namespace>/<repositorio>/` donde `<lenguaje_ejecutor>` se extrae GitEDU de su metadata (definido con el ayuda del usuario, especificamente en este caso el lenguaje/ejecutor del proyecto) para el repositorio.

Trabajos *GitEDU*: `POST /ide/execute/<op>/<namespace>/<repositorio>` → *EduNube*: `POST /api/execute/<op>/<lenguaje_ejecutor>/<namespace>/<repositorio>/` donde `<lenguaje_ejecutor>` se extrae GitEDU de su metadata (definido con el ayuda del usuario, especificamente en este caso el lenguaje/ejecutor del proyecto) para el repositorio y `<op>` es la operacion `status` o `result` como se lo ha definido anteriormente.

Actualmente estas vistas quieren como minimo que el usuario se encuentra logueado, pero a futuro se podria integrar con un sistema de permisos con la finalidad de restringir cuales usuarios y/o grupos pueden ejecutar un proyecto.

Capítulo 5

Pruebas

5.1. Preparaciones del Ambiente de Pruebas

El ambiente utilizado para pruebas es el mismo utilizado para el despliegue y se documenta en el capitulo 6 de Despliegue.

5.2. Plan de Pruebas

La validación del presente trabajo de titulación se lo ha planteado en tres fases:

1. Flujos completos de funcionalidad
2. Pruebas funcionales
3. Pruebas de integración

Donde se considera que estas mismas pruebas deben ser automatizados en el mayor parte posible con la finalidad de reducir variaciones que puede introducir un operador humano en interacciones distintas de una prueba.

5.2.1. Flujos completos de funcionalidad

Estas pruebas son de alto nivel para revisar funcionalidad desde el punto del usuario en distintos roles de la aplicación y llevar cada uno de esos roles por las distintas fases de su ciclo de vida dentro de la aplicación. Se considera Selenium como buena opción para la automatización de esta fase de pruebas. Aunque por falta de tiempo, no se ha podido implementar las pruebas de forma automatizada, ni en su totalidad, a continuacion se explica la manera en que en teoria deben ser implementados.

Administrador

Dentro del ciclo de vida del administrador se debe probar:

1. Creación/Lectura/Actualización/Eliminación de API Tokens en los sistemas respectivos que son:
   - EduNube.
   - GitServerHTTPEndpoint.
2. Registros de plantillas nuevas y actualización de plantillas existentes de virtualización.
   - La capacidad de establecer herencia de plantillas con el .repospec.

- La capacidad de proteger archivos a través del .edunubeignore y el .edunubeignore.children.
- La capacidad de incluir archivos de forma automática a lo que se extiende una plantilla mediante el mecanismo .templateinclude.

Profesor

Dentro del ciclo de vida del profesor se debe probar:

1. Extensión de plantillas de virtualización.
2. Ejecución de plantillas de virtualización.
3. Validación de las protecciones y herencia dado por los archivos de metadata en cada plantilla.

Estudiante

1. Autenticación por LTI.
2. Editar código en línea.
3. Ejecutar código en línea.

5.2.2. Pruebas funcionales

Estas pruebas son de baja nivel y van directamente contra funcionalidades del código. Se propone llevar las mismas con las pruebas unitarias que llevan Django como marco de desarrollo pero de esta misma forma se requiere que la pruebas no se llevan en más de un sistema a la vez y entonces en algunos casos no se podrá realizar pruebas completas de APIs de integración con otros sistemas, si no para las mismas se podría utilizar clientes que simulan la presencia del sistema consumidor externo o incluso utilizar referencias cruzadas (como enlaces simbólicas de UNIX y submódulos de Git) para probar estas funcionalidades.

GitEDU

Para las pruebas de GitEDU, se necesita deshabilitar ciertas funcionalidades de consumo de los dos servicios externos EduNube y GitServerHTTPEndpoint los cuales no se encuentran activos y levantados al momento de ejecutar estas pruebas con la necesidad de que estas pruebas de integración se los extrae a ser ejecutadas directamente con las pruebas unitarias de las respectivas sistemas.

EduNube

Las pruebas de EduNube deben llevar acabo validación de los entornos distintos de virtualización/ejecución de código a trevés del API externa, el cliente estándar que utiliza GitEDU y el manejo adecuado de los API Tokens y .RepoSpec.

GitServerHTTPEndpoint

Las pruebas de GitServerHTTPEndpoint deben validar la API externa, manejo de API Tokens, cliente estándar que utiliza GitEDU y distintas operaciones internas de Git que realiza el mismo sistema.

5.2.3. Pruebas de Integración

Como parte de las pruebas unitarias del fase anterior, se debe validar que los mismos clientes que se han probado son los que realmente esta utilizando cada sistema para con ello asegurar el cumplimiento de la integración esperada. En realizar esta fase esta forma para permitir que se desarrolla en el mismo momento que las pruebas unitarias de Django del fase anterior.

5.3. Resultados

De las pruebas aplicadas, detalladas en los Anexos F, G y H respectivamente, la plataforma desarrollada cumple con su especificacion, pero solo por dos motivos:

1. Las pruebas aplicadas son incompletas para probar todo las especficaciones originales del sistema.

2. La correcciones de errores en los servicios fue dirigido en base a las pruebas aplicadas.

Capítulo 6

Despliegue

6.1. Plan de Despliegue

En el curso del desarrollo, depuración y ejecución de pruebas llegó a ser evidente el enorme consumo de recursos para levantar todos los servicios desarrollados y auxiliares a la vez, razón por el cual, se ha visto la necesidad de un rediseño de forma mas liviana de la manera en que se despliega todos los componentes integrados a la aplicación para con ello terminar las fases mencionadas anteriormente y dar paso a ejecución en ambientes de pocos recursos, tanto en el ámbito de desarrollo como de producción. En la implementación original de la arquitectura física para el desarrollo, se había planteado una colección de maquinas virtuales quienes representarían servidores distintas dentro de una red, pero para temas de optimizar recursos, se ha realizado los siguientes cambios los cuales se documentan a lo largo de este capitulo:

- Kubernetes (backend de virtualización) se ubica en una maquina virtual de Xen (para virtualización activa con el fin de aumentar el rendimiento del sistema virtualizado) dado que el mismo, por temas de seguridad, debe seguir de una forma aislada. De esta forma se simula que esta en un servidor aparte, sea físico o virtual (se recomienda virtualizar todo el cluster final de Kubernetes en algún hipervisor de tipo 1 para garantizar mayor seguridad y aislamiento) pero sin la mayor parte de las perdidas de rendimiento que se dio con MiniKube (alojado en VirtualBox).
- Los servicios auxiliares en lugar de ser maquinas virtuales de Xen ahora son contenedores de Docker.
- Los servicios desarrollados se los han convertido en servicios (Systemd) del sistema operativo, no tanto por temas de rendimiento (aunque se podría mejorar el rendimiento de esta forma, asignándoles a un usuario con mayor prioridad de ejecución como el usuario root[1][2]) si no por temas de facilitar la administración del mismo.
- La integración de los servicios con NGinX para que el mismo puede protegerlos y operar en la capacidad de proxy inverso, proxy de terminación SSL/TLS, servidor de archivos estáticos y validador de peticiones sin mayor perdida de rendimiento.

[1] Pero realizarlo se encuentra fuera del alcance de este tesis, una solución así de software tampoco puede cambiar los recursos físicos de la maquina.

[2] Nunca se debe asignar un servicio que se consume en la red externa a algún usuario con privilegios de superusuario, como root, ya que el mismo abre todo el sistema operativo a un nuevo vector de ataque por el mismo servicio. En este caso debe ser un nuevo usuario, preferiblemente uno por cada servicio, con acceso restringido que tiene mayor prioridad de ejecución para sus procesos.

6.2. Preparaciones del Ambiente de Despliegue

Para preparar el ambiente de despliegue se lo ve pertinente planificar direcciones IP, dominios y puertos para los varios servicios que requieren ser levantados:

Maquina Física con Debian 9.x para Servicios y Docker: *10.10.10.1*

NGinX *http://0.0.0.0:80*

- *http://10.10.10.1:80* & *http://git.localhost:80* ->GitWeb (HTTP)
- *http://gitedu.localhost:80* ->GitEDU (HTTP)
- *http://edunube.localhost:80* ->EduNube (HTTP)
- *http://gitsrvendpoint.localhost:80* ->GitServerHTTPEndpoint (HTTP)
- *http://gitlab.localhost:80* ->GitLab (HTTP)
- *http://moodle.localhost:80* ->Moodle (HTTP)
- *http://registry.localhost:80* ->Registry (HTTP)

GitEDU *http://0.0.0.0:8000*

EduNube *http://0.0.0.0:8010*[3]

GitServerHTTPEndpoint *http://0.0.0.0:8020*[4]

Kubernetes Proxy *http://0.0.0.0:8001*

Docker *10.10.10.1*

MySQL *0.0.0.0:3306 ->moodledb:3306* (TCP/MySQL)

Moodle *0.0.0.0:8201 ->moodle:80* (HTTP)

GitLab CE *10.10.10.1*

- *0.0.0.0:8143 ->gitlab:443* (HTTPS)
- *0.0.0.0:8101 ->gitlab:80* (HTTP)
- *0.0.0.0:8122 ->gitlab:22* (SSH)

Registry *0.0.0.0:5000 ->registry:5000* (HTTP)

Maquina Virtual (Xen) con Debian 9.x para el Cluster (solo 1 nodo maestro) de Kubernetes: *10.10.10.12*

Se resume la configuracion en la figura 6.1.

Moodle en Docker

La instalación de Moodle en Docker consiste de los siguientes pasos:

1. Bajar el imagen de Docker oficial de MySQL:

   ```
   docker pull mysql
   ```

2. Bajar un imagen de Docker de Moodle:

   ```
   docker pull jauer/moodle
   ```

[3] Previamente el puerto asignado fue :8001, pero esto estaba en conflicto con el Dashboard/Proxy de Kubernetes, por lo tanto se ha cambiado este servicio al puerto :8010

[4] Para temas de consistencia con el cambio de puerto de EduNube, se ha cambiado el numero de puerto de :8002 a :8020

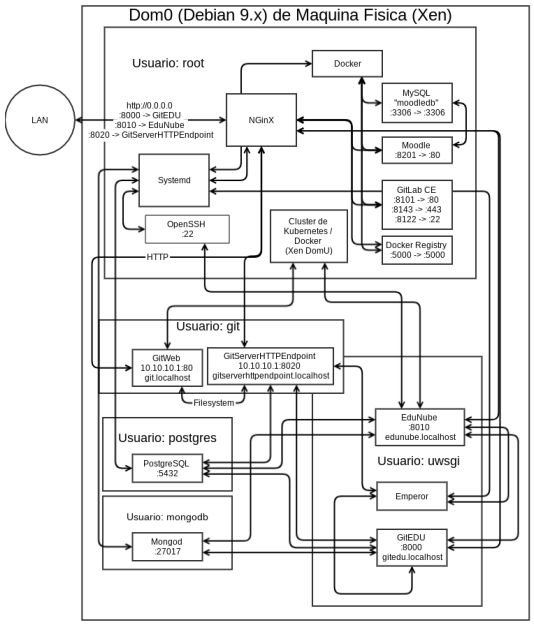

FIGURA 6.1: Resumen de Arquitectura Despliegada

3. Ejecutar el imagen de MySQL en el fondo (-d) con un nombre posible de identificar (moodledb), paso de puertos (10.10.10.1:3306 ->moodledb:3306), un volumen de persistencia para el motor de base de datos, nombres de bases de datos, usuarios y contraseñas de MySQL y una petición de que siempre se reinicia el contenedor a lo que muere con algún error:

```
docker run -d --name moodledb -p 3306:3306 -v \
/srv/moodle/mysql:/var/lib/mysql -e \
MYSQL_DATABASE=moodle -e MYSQL_ROOT_PASSWORD=moodle \
-e MYSQL_USER=moodle -e MYSQL_PASSWORD=moodle \
--restart always mysql
```

4. Ejecutar el imagen de Moodle con opciones similares al anterior, con un enlace a la base de datos con un dominio local de DB, metadatos del url en que debe responder y ocupar el puerto 8201 local como puerto 80 del contenedor (un passthrough):

```
docker run -d -P --name moodle --link moodledb:DB\
-e MOODLE_URL=http://10.10.10.1:8201 -p 8201:80 \
-v /srv/moodle/data:/var/moodledata \
--restart always jhardison/moodle
```

5. Y finalmente visitamos http://10.10.10.1:8201/ para terminar la instalación inicial de Moodle y realizar su configuración inicial.

GitLab en Docker

Para instalar GitLab en Docker(Baum y col., 2017), se sigue pasos similares a los vistos previamente con Moodle:

1. Bajar el imagen de Docker:

```
docker pull gitlab/gitlab-ce
```

2. Ejecutar el imagen de Docker:

```
docker run --detach --hostname 10.10.10.1 \
--publish 8143:443 --publish 8101:80 \
--publish 8122:22 --name gitlab --restart always \
--volume /srv/gitlab/config:/etc/gitlab \
--volume /srv/gitlab/logs:/var/log/gitlab \
--volume /srv/gitlab/data:/var/opt/gitlab \
gitlab/gitlab-ce:latest
```

3. Visitar http://10.10.10.1:8101/ para cambiar la contraseña de root

Registry en Docker

A diferencia de el ambiente de desarrollo, que ocupa el Docker Registry de GitLab[5], para el ambiente de despliegue se plantea replicar el servicio de forma local para reducir el consumo de red. Para ello, se utiliza un contenedor de Docker:

1. Bajar el imagen de Docker:

```
docker pull registry
```

2. Ejecutar el imagen de Docker:

```
docker run -d -p 5000:5000 --restart always \
--name registry registry
```

El proyecto de Docker, en parte por su mala fama de ser insegura, toma muy enserio la seguridad. Por lo tanto, aunque se puede consumir normalmente un Docker Registry por HTTP con el dominio localhost, al ocupar cualquier otro dominio o direccion IP, el cliente de Docker por defecto exige que las llamadas al API en el servidor sean sobre HTTPS. Con la finalidad de no complicar la configuracion más y tambien por el hecho de que la arquitectura fisica del despliegue realmente no sale del localhost, aunque para Docker le parece que si (obviamente no tiene conocimiento de son maquinas virtuales a los cuales el acceso es por adaptadores de red virtuales, etc), se ha optado para realizar la adecuada configuracion al archivo `/etc/docker/daemon.json` en cada host que consume el Registry sobre HTTP para que los mismos no exigen HTTPS donde solo existe HTTP. La configuracion es lo siguiente (en este caso no existia el archivo JSON, entonces se lo creaba con los contenidos a continuacion, en otros casos se agregaria solo los valores abajo a los valores existentes):

```
{
        "insecure-registries" : [
                "10.10.10.1:5000"
        ]
}
```

Despues de realizar aquello configuracion es necesario reiniciar Docker para que tome en cuenta los cambios con el comando `systemctl restart docker`. Para popular el registry con los contenedores necesarias para ejecuccion de codigo con el

[5]Se puede encontrar en `https://gitlab.com/nishedcob/GitEDU/container_registry`

sistema EduNube, puede referirse a los commandos encontrados en `docker/commands-local-doc.sh` del repositorio principal. Este script toma los contenedores compilados previamente durante el desarrollo, los prepara para subirse y se los sube al Registry para que otros consumidores del mismo Registry tengan acceso a ellos.

Máquina Virtual de Xen

Con el fin de proteger la maquina física contra usuarios finales, sean maliciosos o solo sin conocimientos adecuados, hay la necesidad de que el ambiente que ejecuta código sea aislado con virtualización. Pero el rendimiento de esta maquina virtual necesite ser maximizado para poder permitir su uso con un mínimo de recursos. Es, por lo tanto, que se ha propuesta utilizar una maquina virtual de Xen de baja nivel con mejores de rendimiento con la tecnología de paravirtualización con la finalidad de que esta misma logra ofrecer alta aislamiento, y por lo tanto seguridad, frente una alta rendimiento.

Construcción de Servidor Virtualizado para Cluster de Kubernetes Las características mínimas que pide Kubernetes para su nodo maestro son 2 núcleos y 2 GiB de RAM, es por aquello razón que se crea una maquina virtual para virtualización con estas mismas características. En el curso del año que se ha trabajado este trabajo de titulación la comunidad de Debian se ha logrado arreglar el bug que antes causó problemas la ultima vez que se realizó una maquina virtual nueva de Debian 9 y es con este motivo que se puede crear una maquina virtual nueva sin ninguna necesidad para pasos adicionales, los resultados de la creacion del mismo, se indica en la figura 6.2:

```
xen-create-image --hostname=debian-k8s-master \
--ip=10.10.10.12 --netmask=255.255.255.0 \
--gateway=10.10.10.1 --memory=2048mb --vcpus=2 \
--lvm=Xephyr-VG --pygrub --dist=stretch --force \
--size=10240mb --swap=1024mb
```

Instalación de Docker Para utilizar Kubernetes dentro de esta maquina virtual es necesario primero contarnos con una instalación de Docker. Kubernetes no garantiza que va a funcionar con las ultimas versiones de Docker ya que el API de Docker cambia constantemente, especialmente con actualizaciones de seguridad, pero de esta misma forma para la seguridad de la misma, queremos contar con la misma forma estable. Se instala de la siguiente manera en Debian (para otros sistemas operativos, a lo mucho solo se tendría que cambiar el gestor de paquetes apt por el respectivo de su sistema), los resultados del mismo se encuentran en la figura 6.3:

1. Actualizar el sistema operativo

   ```
   apt update
   apt upgrade
   ```

2. Instalar curl en caso de no estar instalado previamente

   ```
   apt install curl
   ```

3. Bajar el instalador actual de Docker

FIGURA 6.2: Resultados de Crear la Maquina Virtual de Xen para el Cluster de Kubernetes de un solo Nodo

```
curl -fsSL get.docker.com -o get-docker.sh
```

4. Ejecutar el instalador de Docker

```
sh get-docker.sh
```

Para utilizar el Registry creado anteriormente, se agrega el siguiente configuracion a las maquinas del cluster (seguido por el reinicio del servicio de Docker con `systemctl restart docker`):

```
{
        "insecure-registries" : [
                "10.10.10.1:5000"
        ]
}
```

Instalación de Cluster de Kubernetes Para instalar el cluster de Kubernetes, el proceso es relativamente sencillo con una herramienta que se llama `kubeadm` que se encarga de levantar, gestionar y bajar nodos del cluster. Primero para instalar el mismo:

1. Debemos tener suporte en el gestor de paquetes APT para el protocolo HTTPS:

```
apt update && apt install -y apt-transport-https
```

2. Agregamos el repositorio de Kubernetes para Debian y Ubuntu:

```
Server:
 Engine:
  Version:      17.12.0-ce
  API version:  1.35 (minimum version 1.12)
  Go version:   go1.9.2
  Git commit:   c97c6d6
  Built:        Wed Dec 27 20:09:54 2017
  OS/Arch:      linux/amd64
  Experimental: false
If you would like to use Docker as a non-root user, you should now consider
adding your user to the "docker" group with something like:

  sudo usermod -aG docker your-user

Remember that you will have to log out and back in for this to take effect!

WARNING: Adding a user to the "docker" group will grant the ability to run
         containers which can be used to obtain root privileges on the
         docker host.
         Refer to https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface
         for more information.
root@debian-k8s-master:~#
```

FIGURA 6.3: Resultados de la Instalacion de Docker en el Cluster de Kubernetes de un Solo Nodo

```
curl -s \
        https://packages.cloud.google.com/apt/doc/apt-key.gpg \
        | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
```

3. Actualizamos los índices de APT y procedemos a instalar:

```
apt-get update
apt-get install -y kubelet kubeadm kubectl
```

(The Kubernetes Authors, 2018b)

El levantamiento de un nodo maestro básica se puede hacer con el comando:

```
kubeadm init
```

Que utiliza todos los valores por defecto, pero Kubernetes requiere para su funcionamiento algún driver de red de los cuales se ha elegido Flannel ya que es el más sencillo y no necesitamos mayor funcionalidad como Switches programables, ni enrutamiento o ACLs en las redes de nuestro cluster debido a que no se busca levantar sistemas de producción aquí, solo contenedores independientes y obviamente otros drivers de red con mayor funcionalidad tienen un costo de rendimiento mayor. Si es que se levanto el cluster con el comando anterior, se lo puede destruir con el siguiente comando:

```
kubeadm reset
```

Ya que Flannel requiere que se define el rango de IPs con que se trabajara el cluster como 10.244.0.0/16[6], por lo tanto en realidad, para trabajar con Flannel es necesario agregar un argumento al comando de levantamiento del cluster como se lo indica a continuación:

```
kubeadm init --pod-network-cidr=10.244.0.0/16
```

[6]Si, parece que tiene que ser exactamente este rango y no suporta mas que $65,536$ contenedores al mismo tiempo, pero eso debe ser suficiente para el propósito actual.

Esto se genera una salida como la que se demuestra a continuación:

```
To start using your cluster, you need to run the following (...)
as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl␣apply␣-f␣[podnetwork].yaml" with one of the (...)
options listed at:
  https://kubernetes.io/docs/concepts/cluster-(...)
  administration/addons/

You can now join any number of machines by running the (...)
following on each node
as root:

  kubeadm join --token 655cb5.2275aa7df206fe69 \
  10.10.10.12:6443 --discovery-token-ca-cert-hash \
  sha256:4919df120063c4535fd03e909ce11dfe9e6448f8a7\
  67be914e86b16660d267c8
```

(The Kubernetes Authors, 2018d)

Por defecto Kubernetes no permite que el nodo maestro aloja contenedores como una política de seguridad, pero en este caso se quiere levantar un cluster de un solo nodo y por lo tanto se requiere cambiar esta política con el siguiente comando:

```
KUBECONFIG=/etc/kubernetes/admin.conf kubectl taint nodes \
--all node-role.kubernetes.io/master-
```

El Variable de Entorno de KUBECONFIG da el archivo que se ve a continuación que autentica el cliente con el cluster. Una vez que se configura bien el cliente, este mismo deja de ser necesario. Entonces a continuación para configurar el cliente:

```
mkdir -p $HOME/.kube
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config
```

(The Kubernetes Authors, 2018d)

Se puede probar la conexión pidiendo la versión del cliente y del servidor:

```
kubectl version
```

(The Kubernetes Authors, 2018d)

Para instalar el driver de red Flannel (que es necesaria cualquier driver de red para Kubernetes previo a funcionamiento – no se instala con ningún por defecto para que sea al elección del administrador quien instala ya que un cluster de Kubernetes solo puede ocupar un driver de red a la vez):

```
sysctl net.bridge.bridge-nf-call-iptables=1
cat sysctl.conf
cat >> /etc/sysctl.conf << EOF

# For Kubernetes Flannel
```

```
net.bridge.bridge-nf-call-iptables = 1

EOF
kubectl apply -f \
https://raw.githubusercontent.com/coreos/flannel/v0.9.1/\
Documentation/kube-flannel.yml
```

(The Kubernetes Authors, 2018a)

A continuación se puede validar que Flannel o cualquier driver de red para Kubernetes esta funcionando con el comando:

```
kubectl get pods --all-namespaces
```

Y que se revisa en la salida del mismo si se logra levantarse `kube-dns` con una linea similar a:

```
kube-system   kube-dns-...-...   3/3   Running   ...   ...
```

(The Kubernetes Authors, 2018d)

Para utilizar volúmenes de Git dentro del cluster, es necesario instalar Git en cada nodo:

```
apt install git
```

Para consumir el cluster desde el servicio de EduNube, es necesario copiar el archivo /etc/kubernetes/admin.conf al otro equipo. Para realizar la copia, lo que mas conviene, especialmente en redes no tan confiables (ya que el admin.conf contiene los credenciales de superusuario de Kubernetes), es realizar la copia sobre SSH. Primero para validar la existencia de sshd:

```
apt install net-tools
netstat -tupln
```

Se puede buscar un servicio llamado sshd que normalmente escucha en el puerto 22 (pero puede ser configurado en otro puerto para ofrecer un poco mas de seguridad[7]). Si es que no existe, hay que instalar el servicio:

```
apt install openssh-server
```

Pero en la instalación que realiza xen-tools se instala sshd por defecto. Dentro del servidor de Kubernetes no se tiene acceso al admin.conf cualquier usuario, entonces se requiere conectar por SSH como root, algo que normalmente es muy peligroso y se deshabilita por defecto. Para permitirlo, hay que editar `/etc/ssh/sshd_config` y reiniciar el servicio:

```
vim.tiny /etc/ssh/sshd_config

# Editar la linea que diga PermitRootLogin para que diga:
PermitRootLogin yes

systemctl restart sshd
```

Desde el equipo que tiene EduNube como servicio se realiza la copia por SSH:

```
scp root@10.10.10.12:/etc/kubernetes/admin.conf .
```

[7]Normalmente un cambio de puerto no ofrece mayor seguridad y es mal consejo, pero en la experiencia del autor, la mayoría de ataques de fuerza bruta de SSH no utilizan ningún escaneo de puertos ya que son script kiddies o bots bien básicos

Se puede probar que funcionan los credenciales copiados con una petición a Kubernetes de una lista de las maquinas que componen el cluster:

```
kubectl --kubeconfig ./admin.conf get nodes
```

Si es que funcionan los credenciales y el sistema que aloja el nodo maestro de Kubernetes esta expuesto a alguna red donde no hay suficiente confianza para dejar que se conecta root por SSH, ya se puede deshabilitar la configuración que se realizo anteriormente y reiniciar el servicio. (The Kubernetes Authors, 2018d)

Para instalar los credenciales de Kubernetes para que EduNube tiene acceso (pero sin perderse los credenciales al MiniKube instalado anteriormente) se realiza los siguientes comandos:

```
mv admin.conf ~/.kube/config.xen
cp ~/.kube/config ~/.kube/config.minikube
cp ~/.kube/config.xen ~/.kube/config
```

(The Kubernetes Authors, 2018d)

Para validar que los credenciales se instalaron correctamente:

```
kubectl version
```

(The Kubernetes Authors, 2018d)

Validación de Cluster de Kubernetes Para validar el cluster de Kubernetes se puede entrar a la carpeta kubernetes dentro del repositorio de GitEDU/EduNube y realizar las siguientes validaciones del nuevo cluster:

```
cd kubernetes/
kubectl create -f debian-pod.yaml
kubectl get pods/utility
kubectl describe pods/utility
kubectl create -f debian-pod-2.yaml
for manifest in 'ls *.json'; do
    kubectl create -f $manifest;
done
kubectl get jobs
# no todos tendran exito al ejecutarse, algunos
# trabajos requieren configuraciones especiales
# que solo se encontraron en el entorno de
# experimentos, y algunos de los experimentos
# no fueron exitosos
watch -n 15 "kubectl␣get␣jobs"
# en este caso el primero en terminar:
kubectl describe jobs/pi
# El pod que fue creado (sera diferente):
# pi-kf8zv
kubectl describe pods/pi-kf8zv
# Ver la salida del trabajo
kubectl logs jobs/pi
```

Interfaz de Administracion Para instalar la interfaz web de administracion para Kuberntes, se sigue los siguientes pasos:

1. Para evitar problemas, se connecta mediante SSH al nodo maestro del cluster y se instala Dashboard:

```
ssh root@10.10.10.12
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/m
```

2. Desde el mismo servidor o otro que tiene conexion al API del nodo maestro (desde donde se quiere tener acceso en su navegador), se ejecuta:

```
kubectl proxy
cat > dashboard-admin.yml << EOF
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: kubernetes-dashboard
  labels:
    k8s-app: kubernetes-dashboard
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: kubernetes-dashboard
  namespace: kube-system
EOF
kubectl create -f dashboard-admin.yml
```

Para ver el Dashboard se visita en su navegador: y en lugar de autenticarse, se pone "skip".

6.3. Despliegue

Para el despliegue de los servicios y su infrastructura de apoyo se propone utilizar NGinX como proxy inversa, con uWSGI como servidor de aplicacion para los servicios de Django/Python y todo gestionado por el sistema operativo de despliegue con Systemd que ahora es un estandar por defecto en casi todos los distribuciones de Linux. La segmentacion de servicios en el mismo puerto (80/HTTP) de NGinX se realiza con subdominios como se indica al inicio de este capitulo.

La configuracion de `http://10.10.10.1` y `http://git.localhost` se encuentra en el capitulo 4 de Desarrollo.

La configuracion de los tres servicios de Django, GitEDU, EduNube y GitServerHTTPEndpoint se detallan a continuacion por seperado por el hecho de que la configuracion de cada uno es mas extenso.

Para la configuracion de NGinX para Gitlab se escribe las siguientes lineas en el archivo `/etc/nginx/sites-available/gitlab`:

```
server {
        listen 80;
        listen [::]:80;
        server_name gitlab.localhost;
        location / {
```

```
                proxy_pass http://127.0.0.1:8101;
        }
}
```

Se agrega la nueva configuracion de NGinX, valida y renicia el servicio de NGinX:

```
ln -s -r /etc/nginx/sites-available/gitlab \
    /etc/nginx/sites-enabled/21-gitlab
nginx -t
systemctl restart nginx
```

Para probar, puede vistar: `http://gitlab.localhost/users/sign_in`

Para la configuracion de NGinX para Moodle se escribe las siguientes lineas en el archivo `/etc/nginx/sites-available/moodle`:

```
server {
        listen 80;
        listen [::]:80;
        server_name moodle.localhost;
        location / {
                proxy_pass http://127.0.0.1:8201;
        }
}
```

Se agrega la nueva configuracion de NGinX, valida y renicia el servicio de NGinX:

```
ln -s -r /etc/nginx/sites-available/moodle \
    /etc/nginx/sites-enabled/22-moodle
nginx -t
systemctl restart nginx
```

Para probar, puede vistar: `http://moodle.localhost/`

Para la configuracion de NGinX para Docker Registry se escribe las siguientes lineas en el archivo `/etc/nginx/sites-available/docker-registry`:

```
server {
        listen 80;
        listen [::]:80;
        server_name registry.localhost;
        location / {
                proxy_pass http://127.0.0.1:5000;
        }
}
```

Se agrega la nueva configuracion de NGinX, valida y renicia el servicio de NGinX:

```
ln -s -r /etc/nginx/sites-available/docker-registry \
    /etc/nginx/sites-enabled/23-docker-registry
nginx -t
systemctl restart nginx
```

Para probar, puede vistar: `http://registry.localhost/v2/_catalog`

Para instalar uWSGI, se lo instala como root a nivel de sistema operativo para Python 3:

```
pip3 install uwsgi
```

El usuario y grupo que se utiliza para los servidores de aplicaciones (menos la de GitServerHTTPEndpoint que utiliza el usuario Git) es uWSGI, por lo tanto hay la necesidad de crear dicho usuario:

```
adduser uwsgi
```

El servicio de SystemD para gestionar cada servidor de aplicacion (uWSGI) mediante un archivo de configuracion `.ini` se define de la siguiente manera:

```
[Unit]
Description=uWSGI Emperor service

[Service]
ExecStart=/usr/local/bin/uwsgi --emperor /etc/uwsgi/sites
Restart=always
KillSignal=SIGQUIT
Type=notify
NotifyAccess=all
User=uwsgi
Group=uwsgi

[Install]
WantedBy=multi-user.target
```

El servicio de uWSGI se empieza con el siguiente manera:

```
systemctl start uwsgi
```

Se puede revisar el estado del mismo con:

```
systemctl status uwsgi
```

Cuando se ha confirmado su correcto funcionamiento, se puede activar el servicio para iniciar con el sistema operativo:

```
systemctl enable uwsgi
```

Finalmente se edita el archivo sudoers para que cualquier usuario que es miembro del grupo uwsgi puede reiniciar el servicio como root:

```
visudo
```

y se agrega la siguiente linea:

```
%uwsgi  ALL= NOPASSWD: /bin/systemctl restart uwsgi
```

Donde `%uwsgi` indica el grupo `uwsgi`, `ALL` indica desde cualquier host, local o remoto, `NOPASSWD` indica que no hay necesidad de pedir la clave del usuario, eso permite utilizar el comando de forma autonoma (no interactiva) dentro de scripts y finalmente se define el commando exacto para el cual se aplica la regla (cualquier modificacion commando y ya no se aplica la regla).

Para facilitar el acceso al usuario `uwsgi` por SSH, se genera un par de llaves:

```
ssh-keygen -f ~/.ssh/id_uwsgi
```

Esta llave se lo copia al usuario `uwsgi`:

```
ssh-copy-id -i ~/.ssh/id_uwsgi.pub uwsgi@10.10.10.1
```

Se configura SSH para siempre ocupar esta llave en sus conexiones y conocer ese conexion por un alias:

```
vim ~/.ssh/config

# y se agrega las siguientes lineas:
Host uwsgi
      HostName 10.10.10.1
      User uwsgi
      Port 22
      IdentityFile /home/nyx/.ssh/id_uwsgi
```

Se puede probar la configuracion con:

```
ssh uwsgi
```

6.3.1. GitEDU

Para levantar el servicio de GitEDU se guarda el archivo `/etc/uwsgi/sites/gitedu.ini` con los siguientes contenidos:

```
[uwsgi]
project = GitEDU
project_location = %(project)
username = uwsgi
base = /home/%(username)
environment = %(project)/env

chdir = %(base)/%(project_location)
home = %(base)/%(environment)
module = %(project).wsgi:application
logto = /usr/share/uwsgi.%(project).log

master = true
processes = 2

uid = %(username)
gid = uwsgi
http-socket = 127.0.0.1:8002
```

Se crea el archivo log con los permisos adecuados:

```
touch /usr/share/uwsgi.GitEDU.log
chown uwsgi:uwsgi /usr/share/uwsgi.GitEDU.log
```

Se reinicia el servicio de uWSGI para que se levanta con el nuevo archivo de configuracion que representa el respectivo servicio:

```
systemctl restart uwsgi
```

Cualquier error que se da, se lo puede investigar mediante el archivo de log `/usr/share/uwsgi.gitedu.log`.

Se crea una ubicacion para archivos estaticos y se le da los permisos adecuados:

```
mkdir -p /static/service/uwsgi/gitedu/static
find /static -type d -exec chmod +x -c {} \;
chown -c uwsgi:root /static/service/uwsgi
chown -cR uwsgi:root /static/service/uwsgi/gitedu
find /static/service/uwsgi/gitedu -type f -exec chmod -c 644
   {} \;
```

Se cambia de usuario para clonar el repositorio:

```
su - uwsgi
git clone https://gitlab.com/nishedcob/GitEDU.git GitEDU
cd GitEDU
git checkout gitedu-deploy
```

Se cambia el settings.py del servicio:

```
STATIC_ROOT = "/static/service/uwsgi/gitedu/static"
```

Se puede crear y instalar el entorno virtual con:

```
source activate.sh
```

Se copia los archivos al directorio que utiliza NGinX para servir los archivos estaticos, este paso se tiene que repetir siempre y cuando los archivos estaticos cambien:

```
python manage.py collectstatic
```

Para la configuracion de NGinX se escribe las siguientes lineas en el archivo `/etc/nginx/sites-available/gitedu`:

```
server {
        listen 8000;
        listen [::]:8000;
        server_name _;
        location /static/ {
                root /static/service/uwsgi/gitedu;
        }
        location / {
                proxy_pass http://127.0.0.1:8002;
        }
}
```

Se agrega la nueva configuracion de NGinX, valida y renicia el servicio de NGinX:

```
ln -s -r /etc/nginx/sites-available/gitedu \
    /etc/nginx/sites-enabled/31-gitedu
nginx -t
systemctl restart nginx
```

Se puede validar que el servicio esta funcionando en visitar: `http://10.10.10.1:8000/auth/login`

Con Git y SSH se automatiza el despliegue con cada push del servicio de la siguiente manera:

1. Crear un directorio de scripts de apoyo en el raiz del proyecto:

   ```
   mkdir bin
   ```

2. Crear un script (`bin/deploy.sh`) para volver a despliegar la aplicacion que utiliza variables para ofrecer un alto nivel de reutilizacion y capacidad para modificacion rapido:

   ```
   #! /bin/bash

   SYSTEMCTL=/bin/systemctl
   SUDO=/usr/bin/sudo
   ```

```
SERVICE=uwsgi
BASE_PATH=/home/uwsgi/GitEDU/GitEDU
PYTHON=env/bin/python
MANAGE=manage.py
STATIC_COMMAND="collectstatic --no-input"
MIGRATE_COMMAND="migrate"
STAT=status
RESET=restart

$BASE_PATH/$PYTHON $BASE_PATH/$MANAGE $STATIC_COMMAND
$BASE_PATH/$PYTHON $BASE_PATH/$MANAGE $MIGRATE_COMMAND
$SYSTEMCTL $STAT $SERVICE
$SUDO $SYSTEMCTL $RESET $SERVICE
$SYSTEMCTL $STAT $SERVICE
```

Este script copia los archivos estaticos a NGinX, migra la base de datos y reinica el servidor de aplicacion con un reporte del estatus del mismo antes y despues de su reinicio. El mismo script debe ser ejecutable.

3. Crear un hook post-receive de Git (version minimo 2.4.12) `.git/hooks/post-receive`:

```
#!/bin/bash
#
## Originally:
## #!/bin/sh
# An example hook script to prepare a packed repository
    for use over
# dumb transports.
#
# To enable this hook, rename this file to "post-receive
    ".

BASE_PATH="/home/uwsgi/GitEDU/GitEDU"
GIT_DIR_PATH="$BASE_PATH/.git"
DEPLOY_SCRIPT_PATH="$BASE_PATH/bin/deploy.sh"
CHECKOUT="checkout"
SERVICE_NAME="GitEDU"
DEPLOYMENT_BRANCH="gitedu-deploy"
GIT="/usr/bin/git"
WORK_TREE="--work-tree=$BASE_PATH"
GIT_DIR="--git-dir=$GIT_DIR_PATH"
UPDATE_INFO="$WORK_TREE $GIT_DIR update-server-info"

$GIT $UPDATE_INFO
echo "Directorio actual: " `pwd`
echo "Actualizando el servicio $SERVICE_NAME..."
while read oldrev newrev ref
do
    if [[ $ref =~ .*/"$DEPLOYMENT_BRANCH"$ ]];
    then
```

```
            echo "Referencia $DEPLOYMENT_BRANCH recibido.
                Desplegando rama $DEPLOYMENT_BRANCH..."
            echo "Ref: $ref"
            $GIT $WORK_TREE $GIT_DIR $CHECKOUT &&
                $DEPLOY_SCRIPT_PATH
        else
            echo "Ref $ref recibido exitosamente. No voy a
                hacer nada: solo se deplega la rama
                $DEPLOYMENT_BRANCH en este servidor."
        fi
    done
    echo "Servicio $SERVICE_NAME actualizado..."
```

Este script se ejecuta cada vez que se recibe nuevos commits el repositorio y comprueba si los mismos representan la rama `$DEPLOYMENT_BRANCH` para proceder a hacer el despliegue con el mismo, caso contrario no realiza ningun operacion al respeto. Este script debe ser ejecutable (es una forma rapida de desactivar esta funcionalidad del despliegue automatico).

4. Finalmente hay la necesidad de configurar Git para que esta dispuesto a recibir commits en la rama actual ya que en los repositorios que no son *bare* por defecto se rechazo dichos cambios:

   ```
   git config receive.denyCurrentBranch updateInstead
   ```

Finalmente en el repositorio de desarrollo, se puede agrega un nuevo remote que representa el entorno de produccion para su subida por SSH:

```
git remote add localssh_gitedu_deploy uwsgi:/home/uwsgi/GitEDU
```

Y ahora se puede actualizar produccion con:

```
git push localssh_gitedu_deploy --all
```

Para disponer de acceso facil al servicio de forma local, se agrega la siguiente configuracion a NGinX en el archivo `/etc/nginx/sites-available/gitedu-proxy`:

```
server {
        listen 80;
        listen [::]:80;
        server_name gitedu.localhost;
        location / {
                proxy_pass http://127.0.0.1:8000;
        }
}
```

y el mismo se enlaza previo a la activacion de la configuracion:

```
ln -s -r /etc/nginx/sites-available/gitedu-proxy /etc/nginx/
    sites-enabled/41-gitedu-proxy
nginx -t
systemctl restart nginx
```

6.3.2. EduNube

Para levantar el servicio de EduNube se guarda el archivo `/etc/uwsgi/sites/edunube.ini` con los siguientes contenidos:

```
[uwsgi]
project = EduNube
project_location = %(project)/%(project)
username = uwsgi
base = /home/%(username)
environment = %(project_location)/env

chdir = %(base)/%(project_location)
home = %(base)/%(environment)
module = %(project).wsgi:application
logto = /usr/share/uwsgi.%(project).log

master = true
processes = 2

uid = %(username)
gid = uwsgi
http-socket = 127.0.0.1:8011
```

Se crea el archivo log con los permisos adecuados:

```
touch /usr/share/uwsgi.EduNube.log
chown uwsgi:uwsgi /usr/share/uwsgi.EduNube.log
```

Se reinicia el servicio de uWSGI para que se levanta con el nuevo archivo de configuracion que representa el respectivo servicio:

```
systemctl restart uwsgi
```

Cualquier error que se da, se lo puede investigar mediante el archivo de log `/usr/share/uwsgi.EduNube.log`.

Se crea una ubicacion para archivos estaticos y se le da los permisos adecuados:

```
mkdir -p /static/service/uwsgi/edunube/static
find /static -type d -exec chmod +x -c {} \;
chown -c uwsgi:root /static/service/uwsgi
chown -cR uwsgi:root /static/service/uwsgi/edunube
find /static/service/uwsgi/edunube -type f -exec chmod -c 644
    {} \;
```

Se cambia de usuario para clonar el repositorio:

```
su - uwsgi
git clone https://gitlab.com/nishedcob/GitEDU.git EduNube
cd EduNube
git checkout edunube-deploy
```

Se cambia el settings.py del servicio:

```
STATIC_ROOT = "/static/service/uwsgi/edunube/static"
```

Se puede crear y instalar el entorno virtual con:

```
source activate.sh
```

Se copia los archivos al directorio que utiliza NGinX para servir los archivos estaticos, este paso se tiene que repetir siempre y cuando los archivos estaticos tambien:

```
python manage.py collectstatic
```

Para la configuracion de NGinX se escribe las siguientes lineas en el archivo `/etc/nginx/sites-available/edunube`:

```
server {
        listen 8010;
        listen [::]:8010;
        server_name _;
        location /static/ {
                root /static/service/uwsgi/edunube;
        }
        location / {
                proxy_pass http://127.0.0.1:8011;
        }
}
```

Se agrega la nueva configuracion de NGinX, valida y renicia el servicio de NGinX:

```
ln -s -r /etc/nginx/sites-available/edunube \
    /etc/nginx/sites-enabled/32-edunube
nginx -t
systemctl restart nginx
```

Se puede validar que el servicio esta funcionando en visitar: `http://10.10.10.1:8010/auth/login`

Con Git y SSH se automatiza el despliegue con cada push del servicio de la siguiente manera:

1. Crear un directorio de scripts de apoyo en el raiz del proyecto:

   ```
   mkdir bin
   ```

2. Crear un script (`bin/deploy.sh`) para volver a despliegar la aplicacion que utiliza variables para ofrecer un alto nivel de reutilizacion y capacidad para modificacion rapido:

   ```
   #! /bin/bash

   SYSTEMCTL=/bin/systemctl
   SUDO=/usr/bin/sudo
   SERVICE=uwsgi
   BASE_PATH=/home/uwsgi/EduNube/EduNube
   PYTHON=env/bin/python
   MANAGE=manage.py
   STATIC_COMMAND="collectstatic␣--no-input"
   MIGRATE_COMMAND="migrate"
   STAT=status
   RESET=restart

   $BASE_PATH/$PYTHON $BASE_PATH/$MANAGE $STATIC_COMMAND
   $BASE_PATH/$PYTHON $BASE_PATH/$MANAGE $MIGRATE_COMMAND
   $SYSTEMCTL $STAT $SERVICE
   ```

```
$SUDO $SYSTEMCTL $RESET $SERVICE
$SYSTEMCTL $STAT $SERVICE
```

Este script copia los archivos estaticos a NGinX, migra la base de datos y reinica el servidor de aplicacion con un reporte del estatus del mismo antes y despues de su reinicio. El mismo script debe ser ejecutable.

3. Crear un hook post-receive de Git (version minimo 2.4.12) `.git/hooks/post-receive`:

```
#!/bin/bash
#
## Originally:
## #!/bin/sh
# An example hook script to prepare a packed repository
    for use over
# dumb transports.
#
# To enable this hook, rename this file to "post-receive
    ".

BASE_PATH="/home/uwsgi/EduNube"
GIT_DIR_PATH="$BASE_PATH/.git"
DEPLOY_SCRIPT_PATH="$BASE_PATH/EduNube/bin/deploy.sh"
CHECKOUT="checkout"
SERVICE_NAME="EduNube"
DEPLOYMENT_BRANCH="edunube-deploy"
GIT="/usr/bin/git"
WORK_TREE="--work-tree=$BASE_PATH"
GIT_DIR="--git-dir=$GIT_DIR_PATH"
UPDATE_INFO="$WORK_TREE $GIT_DIR update-server-info"

$GIT $UPDATE_INFO
echo "Directorio actual: " `pwd`
echo "Actualizando el servicio $SERVICE_NAME..."
while read oldrev newrev ref
do
    if [[ $ref =~ .*/"$DEPLOYMENT_BRANCH"$ ]];
    then
        echo "Referencia $DEPLOYMENT_BRANCH recibido.
            Desplegando rama $DEPLOYMENT_BRANCH..."
        echo "Ref: $ref"
        $GIT $WORK_TREE $GIT_DIR $CHECKOUT &&
            $DEPLOY_SCRIPT_PATH
    else
        echo "Ref $ref recibido exitosamente. No voy a
            hacer nada: solo se deplega la rama
            $DEPLOYMENT_BRANCH en este servidor."
    fi
done
echo "Servicio $SERVICE_NAME actualizado..."
```

Este script se ejecuta cada vez que se recibe nuevos commits el repositorio y comprueba si los mismos representan la rama `$DEPLOYMENT_BRANCH` para proceder a hacer el despliegue con el mismo, caso contrario no realiza ningun operacion al respeto. Este script debe ser ejecutable (es una forma rapida de desactivar esta funcionalidad del despliegue automatico).

4. Finalmente hay la necesidad de configurar Git para que esta dispuesto a recibir commits en la rama actual ya que en los repositorios que no son *bare* por defecto se rechazo dichos cambios:

```
git config receive.denyCurrentBranch updateInstead
```

Finalmente en el repositorio de desarrollo, se puede agrega un nuevo remote que representa el entorno de produccion para su subida por SSH:

```
git remote add localssh_edunube_deploy uwsgi:/home/uwsgi/EduNube
```

Y ahora se puede actualizar produccion con:

```
git push localssh_edunube_deploy --all
```

Para disponer de acceso facil al servicio de forma local, se agrega la siguiente configuracion a NGinX en el archivo `/etc/nginx/sites-available/edunube-proxy`:

```
server {
        listen 80;
        listen [::]:80;
        server_name edunube.localhost;
        location / {
                proxy_pass http://127.0.0.1:8010;
        }
}
```

y el mismo se enlaza previo a la activacion de la configuracion:

```
ln -s -r /etc/nginx/sites-available/edunube-proxy /etc/nginx/
    sites-enabled/42-edunube-proxy
nginx -t
systemctl restart nginx
```

6.3.3. GitServerHTTPEndpoint

Para que el usuario git puede levantar un servidor uwsgi para su servicio GitServerHTTPEndpoint, se le agrege al grupo uwsgi:

```
adduser git uwsgi
```

Para levantar el servicio de GitServerHTTPEndpoint se guarda el archivo `/etc/uwsgi/sites/gitserverhttpendpoint.ini` con los siguientes contenidos:

```
[uwsgi]
project = GitServerHTTPEndpoint
project_location = %(project)
username = git
base = /home/%(username)
environment = %(project)/env
```

```
chdir = %(base)/%(project_location)
home = %(base)/%(environment)
module = %(project).wsgi:application
logto = /usr/share/uwsgi.%(project).log

master = true
processes = 2

uid = %(username)
gid = uwsgi
http-socket = 127.0.0.1:8021
```

Se crea el archivo log con los permisos adecuados:

```
touch /usr/share/uwsgi.gitserverhttpendpoint.log
chown uwsgi:git /usr/share/uwsgi.gitserverhttpendpoint.log
```

Se reinicia el servicio de uWSGI para que se levanta con el nuevo archivo de configuracion que representa el respectivo servicio:

```
systemctl restart uwsgi
```

Cualquier error que se da, se lo puede investigar mediante el archivo de log `/usr/share/uwsgi.gitserverhttpendpoint.log`.

Se crea una ubicacion para archivos estaticos y se le da los permisos adecuados:

```
mkdir -p /static/service/uwsgi/githttpserverendpoint/static
find /static -type d -exec chmod +x -c {} \;
chown -c uwsgi:root /static/service/uwsgi
chown -cR git:root /static/service/uwsgi/
    githttpserverendpoint
find /static/service/uwsgi/githttpserverendpoint -type f -
    exec chmod -c 644 {} \;
```

Se cambia el settings.py del servicio:

```
STATIC_ROOT = "/static/service/uwsgi/githttpserverendpoint/static"
```

Se copia los archivos al directorio que utiliza NGinX para servir los archivos estaticos, este paso se tiene que repetir siempre y cuando los archivos estaticos cambien:

```
python manage.py collectstatic
```

Para la configuracion de NGinX se escribe las siguientes lineas en el archivo `/etc/nginx/sites-available/gitserverhttpendpoint`:

```
server {
        listen 8020;
        listen [::]:8020;
        server_name _;
        location /static/ {
                root /static/service/uwsgi/githttpserverendpoint;
        }
        location / {
                proxy_pass http://127.0.0.1:8021;
        }
}
```

Se agrega la nueva configuracion de NGinX, valida y renicia el servicio de NGinX:

```
ln -s -r /etc/nginx/sites-available/gitserverhttpendpoint \
    /etc/nginx/sites-enabled/33-gitserverhttpendpoint
nginx -t
systemctl restart nginx
```

Se puede validar que el servicio esta funcionando en visitar: `http://10.10.10.1:8020/auth/login`

Con Git y SSH se automatiza el despliegue con cada push del servicio de la siguiente manera:

1. Crear un directorio de scripts de apoyo en el raiz del proyecto:

   ```
   mkdir bin
   ```

2. Crear un script (`bin/deploy.sh`) para volver a despliegar la aplicacion que utiliza variables para ofrecer un alto nivel de reutilizacion y capacidad para modificacion rapido:

   ```
   #! /bin/bash

   SYSTEMCTL=/bin/systemctl
   SUDO=/usr/bin/sudo
   SERVICE=uwsgi
   BASE_PATH=/home/git/GitServerHTTPEndpoint
   PYTHON=env/bin/python
   MANAGE=manage.py
   STATIC_COMMAND="collectstatic␣--no-input"
   MIGRATE_COMMAND="migrate"
   STAT=status
   RESET=restart

   $BASE_PATH/$PYTHON $BASE_PATH/$MANAGE $STATIC_COMMAND
   $BASE_PATH/$PYTHON $BASE_PATH/$MANAGE $MIGRATE_COMMAND
   $SYSTEMCTL $STAT $SERVICE
   $SUDO $SYSTEMCTL $RESET $SERVICE
   $SYSTEMCTL $STAT $SERVICE
   ```

   Este script copia los archivos estaticos a NGinX, migra la base de datos y reinica el servidor de aplicacion con un reporte del estatus del mismo antes y despues de su reinicio. El mismo script debe ser ejecutable.

3. Crear un hook post-receive de Git (version minimo 2.4.12) `.git/hooks/post-receive`:

   ```
   #!/bin/bash
   #
   ## Originally:
   ## #!/bin/sh
   # An example hook script to prepare a packed repository
       for use over
   # dumb transports.
   #
   ```

```
# To enable this hook, rename this file to "post-receive
    ".

BASE_PATH="/home/git/GitServerHTTPEndpoint"
GIT_DIR_PATH="$BASE_PATH/.git"
DEPLOY_SCRIPT_PATH="$BASE_PATH/bin/deploy.sh"
CHECKOUT="checkout"
SERVICE_NAME="GitServerHTTPEndpoint"
DEPLOYMENT_BRANCH="deployment"
GIT="/usr/bin/git"
WORK_TREE="--work-tree=$BASE_PATH"
GIT_DIR="--git-dir=$GIT_DIR_PATH"
UPDATE_INFO="$WORK_TREE $GIT_DIR update-server-info"

$GIT $UPDATE_INFO
echo "Directorio actual: " `pwd`
echo "Actualizando el servicio $SERVICE_NAME..."
while read oldrev newrev ref
do
    if [[ $ref =~ .*/"$DEPLOYMENT_BRANCH"$ ]];
    then
        echo "Referencia $DEPLOYMENT_BRANCH recibido.
            Desplegando rama $DEPLOYMENT_BRANCH..."
        echo "Ref: $ref"
        $GIT $WORK_TREE $GIT_DIR $CHECKOUT &&
            $DEPLOY_SCRIPT_PATH
    else
        echo "Ref $ref recibido exitosamente. No voy a
            hacer nada: solo se deplega la rama
            $DEPLOYMENT_BRANCH en este servidor."
    fi
done
echo "Servicio $SERVICE_NAME actualizado..."
```

Este script se ejecuta cada vez que se recibe nuevos commits el repositorio y comprueba si los mismos representan la rama `$DEPLOYMENT_BRANCH` para proceder a hacer el despliegue con el mismo, caso contrario no realiza ningun operacion al respeto. Este script debe ser ejecutable (es una forma rapida de desactivar esta funcionalidad del despliegue automatico).

4. Finalmente hay la necesidad de configurar Git para que esta dispuesto a recibir commits en la rama actual ya que en los repositorios que no son *bare* por defecto se rechazo dichos cambios:

```
git config receive.denyCurrentBranch updateInstead
```

Para disponer de acceso facil al servicio de forma local, se agrega la siguiente configuracion a NGinX en el archivo `/etc/nginx/sites-available/gitserverhttpendpoint-p`

```
server {
        listen 80;
        listen [::]:80;
        server_name gitserverhttpendpoint.localhost;
```

```
        location / {
                proxy_pass http://127.0.0.1:8020;
        }
}
```

y el mismo se enlaza previo a la activacion de la configuracion:

```
ln -s -r /etc/nginx/sites-available/gitserverhttpendpoint-
    proxy /etc/nginx/sites-enabled/43-gitserverhttpendpoint-
    proxy
nginx -t
systemctl restart nginx
```

Capítulo 7

Conclusiones

7.1. Resultados

El principal resultado obtenido es el realizar un prototipo de la plataforma propuesta donde se integra tres servicios cuyas funcionalidades esenciales son:

GitEDU, ofrece una autenticación clásica y autenticación de LTI, manejo de namespace y repositorios. Adicionalmente cuenta con un editor de código en línea para gestionar archivos dentro de un repositorio, esto, llevado bajo un sistema de control de versiones interno.

EduNube, mediante su API, ofrece toda la funcionalidad necesaria para que cada uno de los usuarios de GitEDU pueda ejecutar el código que desarrolló dentro de la plataforma. El servicio está planteado para extenderse con nuevas funcionalidades a futuro, por ejemplo el dar soporte a la ejecución de más lenguajes de programación (mediante contenedores de Docker), y la calificacion del código escrito por los usuarios.

GitServerHTTPEndpoint, mediante su API, permite que los cambios realizados en el editor de GitEDU sean persistentes en repositorios de Git externos para su consumo por parte de aplicaciones y usuarios externos.

7.2. Conclusiones

A lo largo del desarrollo de este trabajo de titulación, se ha llegado a las siguientes conclusiones:

- En base a la investigación de trabajos relacionados se puede proyectar que el sistema prototipo desarollado podría convertirse en una herramienta valiosa para la docencia de la Universidad Técnica Particular de Loja, con la finalidad de enseñar y evaluar a los estudiantes de programación. Y por lo tanto el prototipo realizado es el primer paso para cumplir con esta finalidad, pero para ello se requiere de un mayor trabajo a futuro.
- Para el desarrollo de sistemas educativos, LTI puede ser una buena solución a tomar en cuenta, para facilitar el flujo de autenticación entre diferentes sistemas de aprendizaje y enseñanza.
- Kubernetes y Docker son herramientas muy potentes y fáciles de implementar en soluciones de mayor complejidad para generar entornos confiables, construir sistemas de producción escalables y portables, así como para ejecutar trabajos por lotes con mayor grado de seguridad y con un alto rendimiento.

Estas tecnologías permitieron cumplir con los requerimientos del servicio de EduNube que se encarga de ejecutar el código de usuarios finales.

- Para la implementación de control de versiones externo en el presente proyecto se ha concluido que cualquier servidor de Git es altamente pesado, especialmente GitLab que en su estado pasivo consume muchos recursos, y también GitWeb que tiene inconvenientes cada vez que recibe nuevos objetos de Git sobre HTTP.
- Para los datos que no bien estructurados o con campos de longitud altamente variable, como en el caso del código escrito por los estudiantes, las bases de datos no-relacionales como MongoDB, son una buena alternativa frente a las bases de datos relacionales además de simplificar el proceso.
- Es importante que el interfaz de un sistema que se expone frente a los usuarios finales, en este caso, el editor de código en linea, "GitEDU", dispone de un fuerte combinación de tecnologías para obtener una interfaz llamativa y funcional en base a jQuery, AJAX, Bootstrap, XTerm.js y Ace Code Editor.
- Django/Python orientado a objetos permite desarollar con mayor rapidez además de hacer referencia al principio DRY (No te repitas) y permite la extensibilidad del código a futuro.
- La combinacion de Python 3 con las librerías selecionadas para realizar los respectivos backends, como: Django, PyModm, Bcrypt, PyJWT, Requests y ipython, forman una combinación potente para el desarollo rápido de servicios de producción, siempre y cuando se maneje de forma adecuada el diseño de los mismos y el manejo de distintos tipos de datos.

7.3. Recomendaciones

En base a lo que se ha aprendido en el camino de este trabajo de titulación, se puede realizar las siguientes recomendaciones:

- Cuando se desarrolla sistemas con un enfoque educativo, se recomienda reutilizar una implementación de LTI para evitar problemas de integración con el software nuevo que se desarrolla.
- Para las interfaces web, se recomienda reutilizar el trabajo de terceros (siempre y cuando se cuente con la licencia adecuada), con el fin de poder dar mayor enfoque a la funcionalidad del backend.
- Para trabajar con cualquier base de datos, sea relacional o no relacional, lo más recomendable es trabajar con un ORM que permita abstraer la base de datos y facilitar el desarrollo y persistencia de datos en la misma. PyMODM es un ORM potente para combinar el poder de Python con MongoDB y un API bastante similar al ORM de Django.
- Siempre se recomienda utilizar nuevas tecnologías como Kubernetes, los cuales pueden ser útiles para resolver problemas actuales ya que ofrecen nuevas perspectivas y soluciones que antes no han sido consideradas.

- Siempre que se pueda, se recomienda no trabajar con GitLab, a menos que se disponga de los recursos necesarios para ello y tambien de las características avanzadas del mismo. En entornos donde sea posible, lo más recomendable es trabajar con Git sobre SSH, que además de ser más seguro, tiene mayor soporte por parte de Git y mayor inteligencia para la sincronización cuando se trabaja con este protocolo, que puede dar como resultado mayor eficiencia en uso de red.

7.4. Trabajos Futuros

Se considera los siguientes aspectos que se podrian y/o se deberian trabajar a futuro, los cuales no esta en ningun orden especifico:

- Arreglar Errores en la funcionalidad existente, por ejemplo:
  - EduNube no actualiza de forma adecuada repositorios de ejecuccion, lo cual resulta muchas veces en la ejecuccion de una version antigua del mismo.
  - EduNube no genera IDs de ejecuccion de forma adecuada.
- Autenticacion en GitEDU y/o otros servicios mediante LDAP.
- Sincronizacion en tiempo real de codigo editado en GitEDU y sus respectivo backends de codigo, posiblemente mediante el uso de websockets.
- Socializacion de vistas para editar codigo (GitEDU) con la finalidad de promover interaccion y collaboracion entre usuarios sobre los mismos.
- Un sistema de permisos para el editor de codigo (GitEDU).
- Distintas formas de calificacion, incluyendo:
  - Calificacion Manual por parte de professores.
  - Calificacion Automatizado por parte del sistema (posiblemente en forma de un servicio nuevo?) en base a pruebas unitarias definidas por profesores.
    - En base a expresiones regulares aplicadas a la salida.
    - En base a análisis lexico/semantico de la salida.
    - En base a pruebas unitarias.
    - En base a una combinación de los anteriores (pesos respectivos de cada metodo/aspecto de calificación definidos por el profesor; como una rubica).
  - Calificacion Hibrida que combina los anteriores.
- Sincronizacion de Notas (generados por la(s) modalidad(es) de calificación anteriores) por LTI con los sistemas adecuadas para el manejo de los mismos, por ejemplo los LMS.
- Mas backends de Git para el servicio GitServerHTTPEndpoint, como por ejemplo GitLab o Djacket.
- Mas backends de persistencia de codigo para el servicio GitEDU como Redis o GitLab.

- Mas backends de virtualizacion para EduNube como OpenStack, Docker, etc.
- Aumentar la seguridad de Kubernetes, tal vez con el uso de Namespace (clusters virtuales) dentro de la misma, quotas de recursos (para no dejar a ningun usuario ocupar todo el procesamiento), y limites de tiempo de ejecucción.
- Versionamiento de APIs en todos los servicios.
- Auditoria de Seguridad del Sistema Desarrollado, especialmente en el caso de los APIs que no manejan estados y solo se protegen con API tokens JWT y a lo mucho TLS.
- Mejor gestion de la configuracion, actualmente hay componentes de algunos servicios que dejen de funcionar o que funcionan de forma inadecuada cuando no disponen de sus servicios dependientes. Debe ser configurable cuales servicios existen o no, dinamico la manera en que se encuentran y cada servicio tolerante a fallos en los demas servicios.
- Convertir los servicios en imagenes de Docker para montar los mismos en un cluster de Kubernetes y realizar un estudio de alta disponabilidad/escalabilidad.
- Implementar el sistema de plantillas de GitEDU.
- Utilizar realmente el Navbar de GitEDU, que actualmente solo tiene "Cerrar session" de forma estatica.
- Migrar servicios a Django 2.x (la proxima version de soporte a largo plazo de Django esta planificado como el 2.3) ya que en realizar esta migracion de version, actualmente se rompe la forma en que se llevan los URIs con un namespace para cada app.
- Automatizar y aumentar las pruebas unitarias de la aplicacion de acuerdo con el plan de pruebas original.
- Uso bidireccional del Git (por el momento es unidireccional, GitEDU solo guarda en los repositorios de GitWeb, nunca recupera codigo de usuarios guardado alli, que obviamente a futuro podria dificultar la interaccion entre el usuario y el sistema, en obligarle a siempre editar proyectos dentro de la plataforma).
- Mayor soporte para caracteristicas de Git como ramas, tags, etc.
- Llevar metadatos de lenguaje de programacion / ejecutor seleccionado para repositorios en GitEDU, para su uso al momento de llamar al API de EduNube, se puede realizar la llamada adecuada (actualmente esta quemada el uso del ejecutor de Python 3).
- Recolleccion de datos para apoyar la toma de decisiones estrategicas.
- Soporte para documentacion en linea, como Wikis basados en Markdown para promover otra manera en que Estudiantes y Profesores pueden expresar problemas y soluciones.

Apéndice A

Documento de Visión

A.1. Propósito

Ayudar a mejorar los métodos de enseñanza que ofrece la Universidad Técnica Particular de Loja en cuanto a la programación y uso de base de datos para las carreras de Sistemas y Electrónica.

Alcance Se propone un sistema de edición de código en línea que a su vez integra LMS externos (para autenticación y notas), un servidor de control de versiones externo (para la persistencia de código), y un servicio web de ejecución de código en línea de una forma segura, eficaz y eficiente (para dar un ambiente de ejecución y pruebas tanto para los usuarios del sistema como para calificar de una forma automática).

A.2. Definiciones, Acrónimos y Abreviaciones

LMS Learning Management System (Sistema de Gestión de Aprendizaje)

LTI Learning Tools Interoperability (estandar de Interoperabilidad entre Herramientas de Aprendizaje)

GitEDU sistema de Git EDUcation

A.2.1. Posición y Oportunidad de Negocios

Con el avance continuo de la tecnología y su introducción en mas aspectos de la vida diaria, hay una necesidad creciente de ingenieros en sistemas y electrónica que puedan programar y entender el software que hace todo funcionar en adición a las bases de datos que están por detrás de estos mismos sistemas.

Es por aquella razón que ahora está de moda ofrecer plataformas en línea para la enseñanza dinámica de la programación. GitEDU espere ofrecer las mismas funcionalidades a un costo institucional menor a través de la integración con sistemas existentes e innovación para proveer una mejor experiencia de usuario, tanto estudiantes como profesores y llevar a cabo un mejor proceso de aprendizaje.

CUADRO A.1: Definición del Problema.

El problema de	enseñar y evaluar programación
afecta a	los estudiantes y docentes de las carreras de Sistemas y Electrónica
el impacto de lo cual es	el uso ineficiente de recursos universitarios en la enseñanza de la programación
Una solución exitosa seria	una aplicación web que ofrezca un editor de código en línea, la capacidad de ejecutar este código para proveer mejor interacción con los estudiantes, la capacidad de ejecutar pruebas unitarias para automatizar el proceso de calificaciones, la persistencia de código en un repositorio de control de versiones remoto para la fácil revisión de profesores y estudiantes, y la integración transparente con sistemas de gestión de aprendizaje externos para la autenticación de usuarios y la respectivo registro de notas.

CUADRO A.2: Posición de Producto.

Para	docentes y estudiantes de la Universidad Técnica Particular de Loja
Quienes	tienen dificultades en la enseñanza y aprendizaje con la programación
GitEDU	es una plataforma web
Que	provee un espacio para la interacción entre profesores y alumnos para la enseñanza y aprendizaje de la programación y uso de las bases de datos
A diferencia de	otras plataformas altamente costosas que no se integran completamente con sistemas existentes de la universidad ni permiten alta interacción entre docentes y alumnos
Nuestro producto	da mayor capacidad para interacción entre estudiantes y sus docentes y se integra bien con las sistemas existentes para dar una mejor experiencia a todos los involucrados a un costo menor.

A.3. Usuarios e Interesados

A.3.1. Demográfica del Mercado

En el año 2011, la Universidad Técnica Particular de Loja contaba con aproximadamente 4000 estudiantes presenciales y 24000 estudiantes a distancia con una tendencia creciente [UTPL-Datos-Estadisticos]. En la experiencia personal del autor, las carreras de Sistemas y Electrónica, por lo menos en la modalidad presencial, juntos representan aproximadamente un 10 % de todos los estudiantes en la universidad lo cual daría un mercado de estudiantes afectados por un nuevo sistema de aproximadamente un mínimo 2800 estudiantes. Según el directorio de docentes de la universidad, son 60 profesores en el departamento de Ciencias de la Computación y Electrónica [UTPL-Directorio-Docentes]. Con eso se puede estimar un mínimo de 2860 usuarios lo los cuales el sistema propuesto podría llegar a afectar.

Es precisamente la parte de la población, de usuario potenciales mencionado anteriormente, que está en el proceso de enseñar, evaluar y aprender habilidades

de programación y consultar bases de datos que forman la base de usuarios de la aplicación.

Nombre	**Descripción**	**Responsabilidades**
Analista	Trabaja con el Asesor Principal y Auxiliar para entender bien las necesidades institucionales para poder llevar un buen ingeniería de requerimientos y diseño del sistema propuesto.	Definir bien el problema para analizarlo, generar requerimientos en base a las necesidades para diseñar y documentar componentes del sistema final para el beneficio del Arquitecto de Software, Programador, Gestor de Proyecto y futuro mano de obra en el proyecto.
Arquitecto de Software	Trabaja con el Asesor Principal y Auxiliar para definir una arquitectura que garantiza que se cumpla con los atributos de calidad que requiere el sistema y será compatible con la infraestructura y sistemas institucionales ya existentes.	Diseñar los modelos de interacción entre todos los componentes internos y externos del sistema para con ello lograr un flujo eficaz y eficiente, que también cumple con los parámetros de los requerimientos no funcionales, en el sistema final.
Gestor de Proyecto	Trabaja con el Asesor Principal y Auxiliar para evaluar, estimar y establecer el alcance, los recursos y el cronograma del proyecto.	Distribuir de manera eficiente y eficaz los recursos para ayudar el analista, arquitecto de software, programador y administrador de sistemas y bases de datos cumplir dentro de los recursos, alcance y cronograma preestablecido.
Programador	Trabaja con el Asesor Principal, Asesor Auxiliar, y Analista para implementar soluciones técnicas que cumplen con el diseño dado por el analista.	Desarrollar el sistema en todos sus componentes.
Administrador de Sistemas y Bases de Datos	Trabaja con el Asesor Principal y Auxiliar para desplegar la aplicación en la institución.	El despliegue correcto del sistema con todos sus componentes en la institución respectivo.
Asesor Principal	Trabaja con el Asesor Auxiliar para poder aconsejar de la mejor manera el Analista, Arquitecto de Software, Gestor de Proyecto, Programador y Administrador de Sistemas y Bases de Datos.	La definición de necesidades y aprobación del producto final.

Nombre	Descripción	Responsabilidades
Asesor Auxiliar	Trabaja con el Asesor Principal para poder aconsejar de la mejor manera al Analista, Arquitecto de Software, Gestor de Proyecto, Programador y Administrador de Sistemas y Bases de Datos.	La definición de necesidades y aprobación del producto final.
Asesor de Documentación	Trabaja con el Analista y Gestor de Proyecto para asegurar la calidad de la documentación que se genera a lo largo del proyecto y que se cumple con el cronograma establecido entre el gestor del proyecto y los asesores principales y auxiliares.	La aprobación de los avances en la documentación y la documentación completa al final del proyecto.

CUADRO A.3: Resumen de Interesados.

CUADRO A.4: Resumen de Usuarios.

Nombre	Descripción	Responsabilidades	Interesado
Estudiantes	Usuario final primario del sistema	Usa la aplicación para cumplir con las tareas, pruebas y exámenes que propone el docente	Los mismos
Professores	Usuario final primario del sistema	Usa la aplicación para ingresar tareas, pruebas y exámenes a los estudiantes y calificar los mismos	Los mismos
Administradores	Quienes administran el sistema en su ambiente de despliegue	Mantener el sistema	El mismo

A.3.2. Ambiente de Usuario

El sistema será disponible para el uso de los usuarios que estén en el campus universitario y en sus hogares.

A.3.3. Perfiles de Interesados

Para entender a fondo cada clase de interesado, a continuacion se presenta un análisis de los mismos.

CUADRO A.5: Perfil de Interesado: Analista.

Descripción	El analista del equipo de desarrollo
Tipo	Miembro del Equipo de Desarrollo
Responsabilidades	Definir bien el problema para analizarlo, generar requerimientos en base a las necesidades para diseñar y documentar componentes del sistema final para el beneficio del Arquitecto de Software, Programador, Gestor de Proyecto y futuro mano de obra en el proyecto.
Criterio de éxito	Llevaar a cabo exitosamente el proyecto bajo todos sus requerimientos funcionales y no funcionales
Involucramiento	En cada fase del proyecto
Entregables	Documentación del sistema y su funcionamiento
Comentarios / Preocupaciones	Que se cumpla con todas las necesidades institucionales

CUADRO A.6: Perfil de Interesado: Arquitecto de Software.

Descripción	El arquitecto de software en el equipo de desarrollo
Tipo	Miembro del Equipo de Desarrollo
Responsabilidades	Diseñar los modelos de interacción entre todos los componentes internos y externos del sistema para con ello lograr un flujo eficaz y eficiente, que también cumple con los parámetros de los requerimientos no funcionales, en el sistema final.
Criteroa de éxito	Que se lleve a cabo exitosamente el proyecto bajo todos sus requerimientos no funcionales
Involucramiento	En cada fase del diseño y despliegue
Entregables	Modelos Arquitectónicos del Sistema y su interacción con otros sistemas
Comentarios / Preocupaciones	Que cumpla con todas las atributos de calidad que la institución manda

CUADRO A.7: Perfil de Interesado: Gestor de Proyecto.

Descripción	El gestor de proyecto en el equipo de desarrollo
Tipo	Miembro del Equipo de Desarrollo
Responsabilidades	Distribuir de manera eficiente y eficaz los recursos para ayudar el analista, arquitecto de software, programador y administrador de sistemas y bases de datos cumplir dentro de los recursos, alcance y cronograma preestablecido.
Criterio de éxito	Que se lleve a cabo exitosamente el proyecto bajo todos sus requerimientos y dentro de los recursos y cronograma preestablecido.
Involucramiento	En cada fase del proyecto
Entregables	Documentación de la gestión del proyecto y el adecuado seguimiento y control interno a lo largo del mismo
Comentarios / Preocupaciones	Que se cumpla con todo el proyecto dentro de los recursos y cronograma preestablecido

CUADRO A.8: Perfil de Interesado: Programador.

Descripción	El programador del equipo de desarrollo
Tipo	Miembro del Equipo de Desarrollo
Responsabilidades	Desarrollar el sistema en todos sus componentes.
Criterio de éxito	Que se lleve a cabo exitosamente el proyecto bajo todos sus requerimientos funcionales y no funcionales
Involucramiento	En cada fase del desarrollo del proyecto
Entregables	Código del Sistema
Comentarios / Preocupaciones	Que se logra programar segun la especificación que el analista le da

CUADRO A.9: Perfil de Interesado: Administrador de Sistemas y Bases de Datos.

Descripción	El administrador de sistemas y bases de datos del equipo de desarrollo
Tipo	Miembro del Equipo de Desarrollo
Responsabilidades	El despliegue correcto del sistema con todos sus componentes en la institución respectiva.
Criterio de éxito	Que se lleve a cabo exitosamente el proyecto bajo todos sus requerimientos funcionales y no funcionales
Involucramiento	En cada fase del desarrollo y despliegue del proyecto
Entregables	Documentación de los Servicios Desplegados
Comentarios / Preocupaciones	Que se logre desplegar la aplicación según la especificación del arquitecto de software

CUADRO A.10: Perfil de Interesado: Asesor Principal.

Descripción	El asesor principal para equipo de desarrollo
Tipo	Asesor del Equipo de Desarrollo
Responsabilidades	La definición de necesidades y aprobación del producto final.
Criteria de Exito	La entrega del producto final
Involucramiento	En cada fase del proyecto
Entregables	Aprobación del Producto Final
Comentarios / Preocupaciones	Que se logra realizar la aplicación.

CUADRO A.11: Perfil de Interesado: Asesor Auxiliar.

Descripción	El asesor auxiliar para equipo de desarrollo
Tipo	Asesor del Equipo de Desarrollo
Responsabilidades	La definición de necesidades y aprobación del producto final.
Criterio de éxito	La entrega del producto final
Involucramiento	En cada fase del proyecto
Entregables	Aprobación del producto final
Comentarios / Preocupaciones	Que se logre realizar la aplicación.

CUADRO A.12: Perfil de Interesado: Asesor de Documentación.

Descripción	El asesor de documentación para equipo de desarrollo
Tipo	Asesor del Equipo de Desarrollo
Responsabilidades	La aprobación de los avances en la documentación y la documentación completa al final del proyecto.
Criterio de éxito	La entrega de la documentación final
Involucramiento	En cada fase del proyecto
Entregables	Aprobación de la Documentación Final y los respectivos avances
Comentarios / Preocupaciones	Que se logre realizar la documentación.

A.3.4. Perfiles de Usuarios

Para entender a fondo cada clase de usuario se da a continuación un analisis de los mismos.

CUADRO A.13: Perfil de Usuario: Estudiante.

Descripción	Estudiantes de la Modalidad Presencial y a Distancia de las carreras de Sistemas y Electrónica
Tipo	Usuario Final Primario
Responsabilidades	Probar el sistema
Criterio de éxito	Que el sistema sea de utilidad para su experiencia educativa
Involucramiento	En la fase de pruebas
Entregables	Ninguno
Comentarios / Preocupaciones	Que la aplicación sea fácil de usar

CUADRO A.14: Perfil de Usuario: Profesor.

Descripción	Profesores de Programación y de Bases de Datos de la Modalidad Presencial y a Distancia de las carreras de Sistemas y Electrónica
Tipo	Usuario final primario
Responsabilidades	Probar el sistema
Criterio de éxito	Que el sistema sea de utilidad para su docencia
Involucramiento	En la fase de pruebas
Entregables	Ninguno
Comentarios / Preocupaciones	Que la aplicación les facilite el proceso de enseñanza

CUADRO A.15: Perfil de Usuario: Administrador.

Descripción	Administradores Institucionales de Sistemas y de Bases de Datos
Tipo	Usuario final secundario
Responsabilidades	Mantener el Sistema
Criteria de Exito	Que el sistema sea fácil de mantener
Involucramiento	En las fases de pruebas y despliegue
Entregables	Ninguno
Comentarios / Preocupaciones	Que la aplicación sea lo suficientemente documentado

A.3.5. Necesidades de Interesados y Usuarios Principales

En base a los interesados y usuarios principales definidos, se define las necesidades del sistema a continuacion.

Necesidad	**Prioridad**	**Preocupaciones**	**Solución Seleccionado**	**Soluciones Propuestas**
Seguridad en el Ambiente de Ejecución de Código	Alta	Las partes del sistema que se encargan de ejecución de código, necesitan alta protección contra usuario maliciosos y daños accidentales.	Ver las soluciones propuestas	Virtualización: ▪ Hipervisor de Tipo 1 ▪ Hipervisor de Tipo 2 ▪ Contenerización
Extensibilidad	Alta	El sistema debe soportar funcionalidades agregadas en el futuro.	Aplicación orientada a la Modularidad, ver la solución propuesta para mayor detalle.	Modularidad entre componentes del sistema para facilitar el proceso de agregar componentes nuevos o reemplazar componentes existentes sin tocar los demás
Facilidad en Autenticación	Mediana	El sistema debe ser fácil para autenticar todos sus usuarios.	Autenticación contra LMS existente mediante LTI	Autenticación contra LMS existente mediante LTI

Necesidad	**Prioridad**	**Preocupaciones**	**Solución Seleccionado**	**Soluciones Propuestas**
Facilidad en Notas	Mediana	El sistema debe ser capaz de registrar notas en sistemas externas sin interacción del profesor.	Registro de Notas mediante LTI	Registro de Notas mediante LTI
Facilidad en Persistencia de Código	Mediana	El sistema, sin necesidad de interacción del usuario debe persistir el código escrito en un servidor de algún sistema de control de versiones externa	Transferencia de código mediante sistemas de control de versiones ya existentes	Transferencia de código mediante sistemas de control de versiones ya existentes

CUADRO A.16: Necesidades de Interesados y Usuarios Principales

A.3.6. Alternativas y Competidores

1. Repl.it
2. Cloud9 IDE

A.4. Vista General de Producto

A.4.1. Perspectiva del Producto

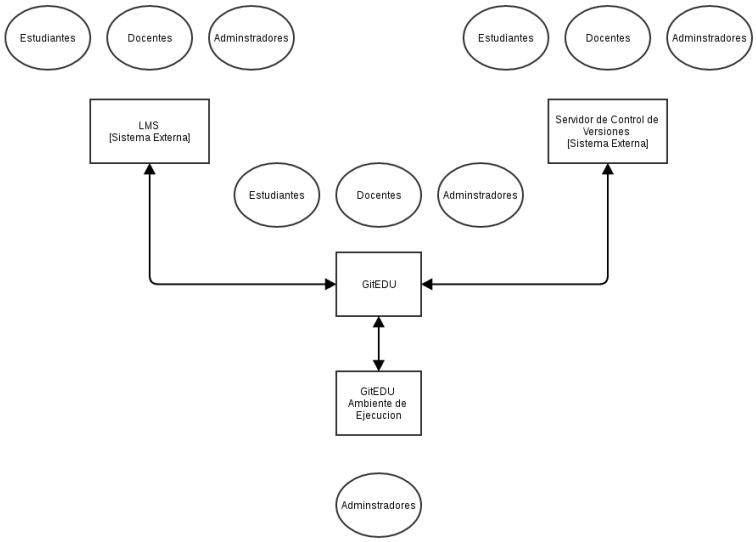

FIGURA A.1: Perspectiva del Producto.

A.4.2. Resumen de Capacidades

CUADRO A.17: Resumen de Capacidades.

Beneficio al Cliente	**Característica de Apoyo**
Facilidad de autenticación	Autenticación por LTI contra LMS institucional
Gestión de la configuración para el código escrito	Uso de servidores externos de control de versiones
Ejecución y pruebas uunitarias en línea	sistema de apoyo con ambiente de ejecución de código
Generación y sincronización de notas	calificación a través de pruebas unitarias, sincronización de notas a través de LTI con LMS institucional

A.4.3. Presuposiciones y Dependencias

1. La institución seguirá usando Moodle y Open EDX como sus plataformas y proveedores de LMS.

2. La institución seguirá usando como proveedor de un servidor de control de versiones de la institución, un servidor institucional con GitLab Community Edition.

3. Alternativas como Repl.it y Cloud9 IDE no dan la funcionalidad necesaria o son demasiados costosos para su uso general por la institución.

A.5. Caracteristicas de Producto

1. Caracteristicas de Autenticación
   - *a*) Autenticarse por LTI
   - *b*) Autenticarse con Usuario y Contraseña
   - *c*) Cerrar Session
2. Caracteristicas de Editar Código
   - *a*) Nuevo Proyecto
   - *b*) Nuevo Proyecto basado en otro Proyecto
   - *c*) Ver Proyectos
   - *d*) Ver Detalles de un Proyecto
   - *e*) Editar Detalles de un Proyecto
   - *f*) Gestionar Usuarios de un Proyecto
   - *g*) Gestionar Permisos de un Proyecto
   - *h*) Abrir Proyecto
   - *i*) Editar Archivos en un Proyecto
   - *j*) Crear Nuevo Archivo en un Proyecto
   - *k*) Borrar Archivo de un Proyecto
   - *l*) Interactuar con Usuarios dentro de un Proyecto
3. Caracteristicas de Persistencia de Codigo
   - *a*) Guardar cambios en un Proyecto
   - *b*) Subir cambios en un Proyecto
   - *c*) Bajar cambios en un Proyecto
4. Caracteristicas de Ejecutar y Calificar Código
   - *a*) Ejecutar Codigo en ambiente interactivo en el navegador
   - *b*) Agregar Pruebas Unitarias a un Proyecto
   - *c*) Ejecutar Pruebas Unitarias en un Proyecto
   - *d*) Calificar un Proyecto
   - *e*) Calificiar un Proyecto de forma Automática en base a Pruebas Unitarias
   - *f*) Enviar Calificaciones a un Sistema Externo

A.6. Precedencia y Prioridades

CUADRO A.18: Precedencia y Prioridades.

Prioridad	Característica (Según su número)
Alta	1.a, 1.c, 2.a, 2.b, 2.c, 2.h, 2.i, 3.a, 3.b, 4.a
Media	1.b, 2.f, 2.g, 2.j, 3.c, 4.b, 4.c, 4.d, 4.f
Baja	2.d, 2.e, 2.k, 2.l, 4.e

A.7. Restricciones

A.7.1. Seguridad

Protección contra ataques en la red
Aislamiento de ambientes de ejecución de código de usuarios

A.7.2. Extensibilidad

Soporte al nivel de sistema y documentación para extensión con más lenguajes de programación y motores de base de datos al futuro

A.7.3. Usabilidad

Ser usable
Requerir un mínimo de interacción del usuario para que puede enfocarse en realizar sus responsabilidades en el sistema

A.7.4. Escalabilidad

Tener capacidad para escalar frente mayor carga en el futuro

A.7.5. Rendimiento

Minimizar las requerimientos mínimos de hardware requerido

A.8. Otros Requisitos de Producto

A.8.1. Normas

Ninguna.

A.8.2. Requisitos de Sistema

Ninguno.

A.8.3. Requisitos de Rendimiento

Ninguno.

A.8.4. Requisitos Ambientales

Ninguno.

A.9. Requisitos de Documentación

A.9.1. Manual del Programador

Documentación para explicar el funcionamiento de la aplicación para que futuros desarrolladores puedan extender sin mayor dificultad la aplicación.

A.9.2. Manual de Mantenimiento

Documentación para definir los procesos de mantenimiento de la aplicación para guiar la gobernanza y administración del mismo.

A.9.3. Manual de Usuario

Documentación para enseñar el uso de la aplicación a docentes y alumnos.

Apéndice B

Especificación de Requisitos de Software

B.1. Introducción

B.1.1. Propósito

El siguiente documento espera dar una descripción detallada de los requisitos del Sistema Git Education (GitEDU) con la finalidad de definir la intención de la misma, declarar de forma completa todos los componentes a ser desarrollados del sistema, y también explicarlas limitaciones del sistema además de sus interacciones e interfaces con otros sistemas. Se destina el siguiente documento para revisión por el equipo de asesores y como una referencia de alcance para el equipo de desarrollo.

B.1.2. Alcance

GitEDU es un sistema de integración para promover la programación estudiantil y el sistema educativo que soporta el mismo. El sistema final debe disponer de buena documentación y alta calidad de código para sostener su futuro desarrollo y mantenimiento por parte de una comunidad de profesionales y profesionales en formación.

Docentes de las carreras que involucran la enseñanza de programación, actualmente tanto la carrera de Sistemas Informáticos y Computación como la carrera de Electrónica y Telecomunicaciones, puedan crear dentro de la plataforma, deberes, talleres, pruebas y exámenes, y a través de los LMS institucionales, compartir los mismas con sus alumnos. Los alumnos pueden entrar a través de un enlace que les comparte su docente dentro del LMS institucional y con un canal de comunicación LTI, la plataforma GitEDU les identifica y autentica sin interacción del usuario para que el estudiante puede empezar directamente con el trabajo que tiene que realizar sin digitar sus credenciales. En el curso de su actividad, se va guardando su progreso periódicamente contra un servidor de control de versiones institucional (GitLab CE) y en cualquier momento el estudiante puede probar e interactuar con su código a través de un terminal virtual de Linux que se encuentra dentro de la misma interfaz. Una vez que se termine su trabajo y desee enviarlo, se lo envía y el código escrito se auto califica contra un conjunto de pruebas unitarias ocultas que el docente agregó a la actividad al crearlo. La nota que se genera se lo envía al LMS institucional a través del mismo canal LTI. En caso de que el docente decide no utilizar pruebas unitarias o calificar cada trabajo a mano o realizar una calificación híbrida entre las pruebas unitarias y la parte a mano, no se reflejarán las notas en el LMS hasta que el docente haya terminado de calificar el código en la plataforma de GitEDU.

Además el sistema requiere una conexión de internet y la disponibilidad de sistemas de apoyo como el servidor de control de versiones, el LMS y el servicio de ejecución de código.

B.1.3. Definiciones, Acrónimos y Abreviaciones

CUADRO B.1: Definiciones, Acrónimos y Abreviaciones para GitEDU

Termino	**Definición**
Docente / Profesor	Alguien que pertenece a la facultad de la institución y cuya responsabilidad es enseñar a sus alumnos / estudiantes programación.
Alumno / Estudiante	Un miembro del cuerpo estudiantil de la institución y está en el proceso de aprender la programación.
Administrador	Quien administra y mantiene el sistema.
Materia / Clase	Un conjunto de Docente(s) y Alumno(s) que están el proceso educativo de programación juntos en una aula física o virtual.
GitLab CE	GitLab Community Edition, un servidor de control de versiones que utiliza el VCS Git
LMS	Learning Management System, un sistema que apoya en la interacción educativa diaria entre Docentes y sus Alumnos
LTI	Learning Tools Interoperability, una estándar y protocolo que permite la comunicación e interacción entre varias herramientas educativas.
VCS	Sistema de Control de Versiones, permite almacenar y visualizar una historia de versiones de archivos o código para dar trazabilidad para lo mismo.

B.1.4. Referencias

Como fuente de la plantilla para el ERS, se ocupó: `http://www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs_example_2010_group2.pdf`

B.2. Descripción General

B.2.1. Perspectiva de Producto

La plataforma GitEDU se descompone en dos subsistemas: el subsistema de editar código y el subsistema de ejecutar código. El subsistema de editar código consiste

en la integración de autenticación y notas por el lado del LMS, el uso de persistencia del código por el servidor de control de versiones por otro lado, el consumo de un servicio de ejecución de código por un tercer lado y un editor de código en línea. El subsistema de ejecutar código consiste en esperar llamadas del subsistema anterior, validar la autenticidad, establecer permisos, limitaciones y parámetros de ejecución, gestionar recursos para ejecución, la ejecución interactiva y devolución de resultados al otro subsistema.

Debido a la necesidad de mantener la seguridad y prevenir la ejecución indebida de código, el subsistema debe ser aislado de todos los usuarios y solo accedido desde el otro subsistema a través de un API que permite su interoperabilidad. La solución para la ejecución de código debe ser liviana debido al gran número de usuarios concurrentes que puede tener a la vez.

Funcionalidades de Producto: para entrar al subsistema de editar código en línea, se debe autenticar contra el LMS institucional. El editor de código en línea debe soportar una variedad grande de lenguajes y estar abierto a la agregación de más lenguajes en un futuro. La persistencia de código en un servidor de control de versiones debe ser transparente y no requerir ningún interacción del usuario. El interfaz para interactuar con el ambiente virtual debe ser simple y fácil de manejar sin mayor conocimiento de las tecnologías que le respaldan. Se debe poder acompañar el editor de código con documentación que guía el estudiante en explicar o resolver el problema actual. La manera en que los docentes pueden generar nuevos ejercicios, su documentación y pruebas unitarias para la calificación debe ser intuitivo.

B.2.2. Características de usuario

Los tres tipos de usuarios que interactúan con el sistema son Docentes, Alumnos y Administradores. Cada tipo de usuario tiene sus propias necesidades y requerimientos.

Docentes, necesitan un sistema que sea fácil de manejar y que simplifique y automatice el proceso de enseñar y evaluar a los estudiantes. Eso significa tener una interfaz intuitiva para los docentes que les permita escribir, documentar y generar pruebas unitarias para probar funcionalidad de código que escriben los estudiantes para los ejercicios.

Alumnos, necesitan un sistema que les brinde una buena experiencia de aprendizaje y les proporcione todas las herramientas que requieran para el desarrollar en línea. Igual que sus profesores, deben disponer de una interfaz intuitiva que les ayude a cumplir con las responsabilidades de estudiar, aprender y demostrar conocimientos en una cantidad de tiempo óptimo.

Administradores necesitan un sistema que sea seguro y cuyos componentes sean fáciles de mantener y extender a lo largo del tiempo. Para ello se necesita tener una buena documentación del funcionamiento del sistema y todos sus componentes para tener la misma como referencia.

B.2.3. Limitaciones

Se requiere de hardware con alta capacidad de virtualización y conectividad al internet y la intranet institucional para poder brindar la mejor experiencia al usuario.

B.2.4. Suposiciones y Dependencias

Se supone que siempre habrá la conectividad necesaria para poder usar todos los componentes del sistema.

B.2.5. División de Requerimientos

En caso de que falta recursos de tiempo o algún otro recurso, se limitará el alcance del proyecto para enfocarse en los requerimientos de una prioridad más alta y dejar los demás requerimientos para trabajos futuros.

B.3. Requisitos Específicos

B.3.1. Interfaces Externos

Interfaces de Hardware

El sistema que interactúa directamente con los usuarios no tiene ningún requisito especial de hardware solo que si se debe contar con memoria, disco y procesador suficiente para un servidor web, servidor de aplicación y servidor de base de datos. El subsistema de virtualización debe contar con los recursos necesarios y el hardware adecuado para soportar la carga del muchos estudiantes a la vez utilizando la tecnología seleccionada.

Interfaces de Software

El sistema que interactúa directamente con los usuarios debe ser capaz de autenticar los mismos contra LMS existentes y sincronizar notas que tiene con el mismo sistema LMS. Además el código que se desarrolla dentro de esta plataforma debe ir contra el servidor del sistema de control de versiones institucional. Y finalmente debe haber comunicación mediante APIs que permite el intercambio de código, entradas y salidas de ejecución en adición a notas resultantes de pruebas unitarias entre el sistema de desarrollo en línea y el sistema de ejecución de código en línea.

Interfaces de Comunicación

La comunicación entre los dos subsistemas es vital para lograr el nivel de funcionalidad que los usuarios finales requieren. Además, como aplicación en línea, se supone que todos los usuarios finales deben tener conectividad previa y durante el uso del sistema.

B.3.2. Requerimientos Funcionales

1. **Estudiantes**

   a) **Autenticación mediante LTI**

   Título: Autenticación mediante LTI

   Descripción: Un estudiante debe poder autenticarse en la aplicación por medio del estándar LTI a través de su sesión abierta en sistemas LMS institucionales.

Razón: Para reducir el número de credenciales que un estudiante necesita recordar y actualizar en adición a promover el uso del sistema en brindar mayor facilidad de acceso.

Dependencias: Ninguno.

b) **Ambiente Completo de Programación**

Título: Ambiente Completa de Programación

Descripción: Un estudiante debe contar con un ambiente de programación completa dentro de la aplicación con el cual puede desarrollar y probar de forma continua.

Razón: Para dar a los estudiantes todas la herramientas necesarias para cumplir con sus responsabilidades, estudio y aprendizaje dentro de un solo entorno.

Dependencias: Ninguno.

c) **Persistencia de Codigo**

Título: Persistencia de Código

Descripción: El código que se escribe dentro del plataforma debe constar respaldado externamente en un servidor de control de versiones independiente.

Razón: Para llevar incrementalmente respaldos del código que escriben estudiantes.

Dependencias: Ambiente Completa de Programación.

d) **Sistema de Documentación**

Título: Sistema de Documentación

Descripción: Tanto estudiantes como sus profesores deben disponer de las herramientas necesarias para documentar y expresar tanto necesidades como funcionalidades de código desarrollado en el plataforma de una manera clara y efectiva.

Razón: Para ayudar profesores y sus alumnos llevar más efectivamente la parte de enseñanza y comunicación que requiere un sistema educativa de esta naturaleza.

Dependencias: Ambiente Completa de Programación.

e) **Sistema de Compartir**

Título: Sistema de Compartir

Descripción: Un usuario del sistema debe poder compartir sus proyectos de programación y la documentación que acompañan los mismos con otros usuarios.

Razón: Para promover cooperación entre usuarios y interacción entre estudiantes y profesores que son importantes para el proceso de enseñanza y aprendizaje.

Dependencias: Ambiente Completa de Programación.

1) **Sistema de Permisos**

Título: Sistema de Permisos

Descripción: Usuarios deben poder controlar el acceso y la naturaleza del mismo de otros usuarios con los mismos se ha compartido el código.

Razón: Para brindar seguridad para el sistema de compartir y para proveer la flexibilidad necesario para tener una gran variedad de usos posibles al mismo sistema, con el suficiente nivel de seguridad (a través de los permisos) para cada uno de ellos.

Dependencias: Sistema de Compartir.

2) **Editar Varios Usuarios al Mismo Tiempo**

Título: Editar Varios Usuarios al Mismo Tiempo

Descripción: Usuarios quienes tengan acceso a un código compartido previamente y tienen permisos de editarlo deben poder editarlo al mismo tiempo de tal forma que los cambios que realizan aparecen a todos en tiempo real.

Razón: Para promover cooperación entre usuarios y interacción entre estudiantes y profesores que son importantes para el proceso de enseñanza y aprendizaje.

Dependencias: Sistema de Compartir.

f) **Sistema de Interacción Social**

Título: Sistema de Interacción Social

Descripción: Un usuario debe poder comunicar en tiempo real con otros usuarios con el cual se ha compartido un código y quienes lo tienen abierto al mismo tiempo.

Razón: Para promover cooperación entre usuarios y interacción entre estudiantes y profesores que son importantes para el proceso de enseñanza y aprendizaje.

Dependencias: Sistema de Compartir.

2. **Professores**

a) **Organizar Estudiantes y Aulas**

Título: Organizar Estudiantes y Aulas

Descripción: Un profesor debe poder organizar sus estudiantes y aulas para poder revisar los mismos y llevarlos del mejor forma.

Razón: Para reducir la cantidad de tiempo que necesita dedicar a la parte administrativo de sus materias para maximizar su tiempo para enseñanza de estudiantes.

Dependencias: Ninguno.

b) **Crear Pruebas, Exámenes, Proyectos, Talleres y Deberes para Estudiantes**

Título: Crear Pruebas, Exámenes, Proyectos, Talleres y Deberes para Estudiantes

Descripción: Un profesor debe poder crear dentro de la aplicación pruebas, exámenes, proyectos, talleres y deberes de los cuales se les puede asignar a sus aulas y estudiantes.

Razón: Para reducir la cantidad de tiempo que necesita dedicar a la parte administrativo de sus materias para maximizar su tiempo para enseñanza de estudiantes.

Dependencias: Organizar Estudiantes y Aulas.

c) **Autocalificación de Código**

Título: Autocalificación de Código

Descripción: Un profesor debe poder escribir pruebas unitarias dentro de la plataforma, los cuales califica de forma automática los pruebas/exámenes/proyectos/talleres y deberes que les asigna a los estudiantes.

Razón: Para reducir la cantidad de tiempo que necesita dedicar a calificar sus estudiantes dentro de sus materias para maximizar su tiempo para enseñanza de los mismos.

Dependencias: Crear Pruebas, Exámenes, Proyectos, Talleres y Deberes para Estudiantes.

d) **Sincronización de Notas**

Título: Sincronización de Notas

Descripción: El sistema debe ser capaz de sincronizar las notas que tiene con sistemas de terceros mediante el protocolo LTI.

Razón: Para reducir la cantidad de tiempo que necesita dedicar a calificar sus estudiantes y a la parte administrativa dentro de sus materias para maximizar su tiempo para enseñanza de los mismos.

Dependencias: Autocalificación de Código.

3. **Administradores**

a) **Integración con Sistemas Existentes**

1) **LMS Institucionales**

Título: Integración con LMS Institucionales Existentes

Descripción: La integración de la plataforma con LMS Institucionales Existentes debe dar las funcionalidades requeridas de autenticación y sincronización de notas de tal forma que su despliegue y mantenimiento sea sencilla y fácil.

Razón: Para reducir los recursos humanos necesarios para asegurar un buen despliegue y mantenimiento de la aplicación.

Dependencias: Autenticación mediante LTI, Sincronización de Notas.

2) **Servidores de Control de Versiones Institucionales**

Título: Integración con Servidores de Control de Versiones Institucionales Existentes

Descripción: La integración de la plataforma con Servidores de Control de Versiones Institucionales Existentes debe dar las funcionalidades requeridas de autenticación y sincronización de notas de tal forma que su despliegue y mantenimiento sea sencilla y fácil.

Razón: Para reducir los recursos humanos necesarios para asegurar un buen despliegue y mantenimiento de la aplicación.

Dependencias: Persistencia de Código.

b) **Recolección de Datos**

Título: Recolección de Datos

Descripción: La plataforma debe recolectar datos acerca de su uso con un fin de ayudar en la toma de decisiones institucionales.

Razón: Para ser un apoyo en la decisiones estratégicos.

Dependencias: Sincronización de Notas.

B.3.3. Requerimientos No Funcionales

1. **Numero de Estudiantes Suportados en Paralelo**

   Título: Número de Estudiantes Suportados en Paralelo

   Descripción: La plataforma debe soportar un número alto de estudiantes usando el mismo en paralelo (a la escala de 90 estudiantes).

   Razón: Para permitir sostener un cierto número de paralelos de una materia a la vez y de esta forma tomar un examen a todos en paralelo.

2. **Tiempo de Respuesta**

   Título: Tiempo de Respuesta

   Descripción: La plataforma debe ser capaz de responder dentro de un tiempo mínimo de un segundo a cualquier consulta de sus usuarios.

   Razón: Para ser útil a sus usuarios finales.

B.3.4. Limitaciones de Diseño

1. **Uso de Disco**

   Se considera que cada repositorio de código que se cree debe tener un tamaño máximo de 1 GB y el entorno virtual asociado con cualquier proyecto no debe exceder 10 GB.

2. **Uso de Memoria**

   El sistema de escribir código en línea no debe necesitar más de una GB de memoria, pero se considera que el subsistema de ejecución de código en línea debe disponer de mínimo 512 MB por cada usuario que espera suportar en paralelo.

3. **Uso de Processador**

   El sistema de escribir código en línea no debe necesitar más de cuatro núcleos físicos de procesador, pero se considera que el subsistema de ejecución de código en línea debe disponer de mínimo un núcleo equivalente a 1 GHz por cada usuario que espera suportar en paralelo.

B.3.5. Atributos del Sistema de Software

1. **Disponibilidad**

   El sistema debe ser disponible para peticiones en todo momento de clases cuando puede ser necesitado.

2. **Seguridad**

   El sistema debe ofrecer niveles óptimos de seguridad para prevenir modificación de notas, daño a equipos físicos y modificación de entornos virtuales de otros usuarios.

3. **Rendimiento**

   No se debe notar la diferencia en tiempos de respuesta y rendimiento entre una carga muy baja y muy alta.

4. **Concurrencia**

   Debe ser capaz de sostener 90 usuarios en paralelo.

5. **Mantenibilidad**

   Debe ser modular para permitir su mantenimiento al largo plazo y el intercambio de nuevos componentes.

6. **Estabilidad**

   A lo largo del tiempo debe presentar un mínimo de fallos que afectan a funcionalidad para el usuario final y de esta manera ofrecer una alta disponibilidad.

7. **Escalabilidad**

   Debe tener capacidad de escalar a números de usuarios mayores con la agregación de hardware adicional.

B.4. Plan de Prioridades y Despliegue

B.4.1. Selección de Método de Prioridades

Se da prioridad a cada requerimiento en base a cuales de los mismos son críticos, es decir funcionalidades básicas del mismo sistema y cuales solo son funcionalidades de valor agregado. En base a estas mismas, se considera también las dependencias entre requerimientos para empezar el desarrollo en las funcionalidades de que dependen otras funcionalidades. Eso se realiza junto con los stakeholders del proyecto para determinar cuales funcionalidades son críticos y cuales se los puede hacer si el tiempo y los recursos se los permiten.

B.4.2. Plan de Entregas de Software

Se plantea desarrollar el sistema en cuatro fases de los cuales cada uno se termina con una entrega de software. En la primera fase se autentica por LTI, en la segunda se realiza un editor de código en línea, en la tercera se realiza la persistencia periódica en el servidor de control de versiones y en la cuarta se realiza el subsistema de virtualización y ejecución de código el mismo que permite a usuarios probar su codigo en linea y califica codigo escrito por estudiantes. Dentro de cada fase se plantea desarrollar las otras características de valor agregado mencionadas previamente siempre y cuando se da el tiempo y los recursos para su cumplimiento. Al final de las cuatro fases y una fase final de pruebas, se procede a trabajar en el despliegue del mismo sistema con los autoridades responsable para el mismo.

Apéndice C

Preparaciones de Desarrollo

C.1. Servicios Nativos

C.1.1. Servidor de Git

Instalar paquetes para un interfaz web sencilla de Git:

```
apt install git gitweb fcgiwrap
```

Se crea un usuario del sistema operativo solo para Git:

```
adduser git
```

Se autentica como el usuario creado:

```
su - git
```

Para tener repositorios ejemplares, se clona algunos repositorios:

```
git clone --bare\
        https://gitlab.com/nishedcob/GitEDU.git\
        repositories/GitEDU.git
git clone --bare\
        https://gitlab.com/ArqAppGrpBravoEarleyVargas/\
        GitEduERP.git\
        repositories/GitEduERP.git
```

Entrar al directorio de repositorios y arreglar permisos para permitir git push desde repositorios remotos:

```
cd repositories/
for repo in `ls`; do
        cd $repo;
    pwd;
    chmod -R g+ws .;
    chgrp -R git .;
    git --bare update-server-info;
    cp hooks/post-update.sample hooks/post-update;
    chmod a+x hooks/post-update;
    cd ..;
done
```

Después se configura NGinX para trabajar con gitweb:

```
# NGinX Config:
server {
        listen 80;
        listen [::]:80;
```

```
        server_name git.localhost 192.168.99.1 10.10.10.1;
        root /usr/share/gitweb;
        access_log /var/log/nginx/gitweb.access.log;
        # static repo files for cloning over https
        location ~ ^.*\.git/objects/([0-9a-f]+/[0-9a-f]+\
                |pack/pack-[0-9a-f]+.(pack|idx))$ {
                root /home/git/repositories/;
        }

        # requests that need to go to git-http-backend
        location ~ ^.*\.git/(HEAD|info/refs|objects/info/.*\
                |git-(upload|receive)-pack)$ {
                root /home/git/repositories;

                fastcgi_pass unix:/var/run/fcgiwrap.socket;
                fastcgi_param SCRIPT_FILENAME   /usr/lib/\
                        git-core/git-http-backend;
                fastcgi_param PATH_INFO          $uri;
                fastcgi_param GIT_PROJECT_ROOT  /home/git/\
                        repositories;
                fastcgi_param GIT_HTTP_EXPORT_ALL "";
                fastcgi_param REMOTE_USER $remote_user;
                include fastcgi_params;
        }

        # send anything else to gitweb if it's not a real file
        try_files $uri @gitweb;
        location @gitweb {
                fastcgi_pass unix:/var/run/fcgiwrap.socket;
                fastcgi_param SCRIPT_FILENAME   /usr/share/\
                        gitweb/gitweb.cgi;
                fastcgi_param PATH_INFO          $uri;
                fastcgi_param GITWEB_CONFIG      /etc/gitweb\
                        .conf;
                include fastcgi_params;
        }
}
```

Además se configura gitweb con la siguiente configuración:

```
# Edit /etc/gitweb.conf
# path to git projects (<project>.git)
#$projectroot = "/var/lib/git";
$projectroot = "/home/git/repositories";
```

Prueba de que funciona:

```
git clone http://git.localhost/nishedcob/GitEDU.git \
        GitEDU-test
```

Dado que el mismo GitWeb no tenga problemas con bajada de datos (`git fetch` / `git pull` / `git clone`) pero si tiene problemas con subida de datos como se lo require en EduNube para armar repositorios de ejecuccion validadas, se ve una necesidad de habilitar subida al mismo por SSH:

Como root en el servidor de Git, se cambia la clave del usuario de Git:

```
passwd git
```

Como el usuario de EduNube en el servidor para el mismo:

```
cd .ssh/
# Deja la llave generada sin clave:
ssh-keygen -f id_git
# Copia la llave publica al servidor
ssh-copy-id -i ~/.ssh/id_git.pub git@10.10.10.1
# Prueba que funciona
ssh -i ~/.ssh/id_git -vvv git@10.10.10.1
# Guardar la configuracion:
cat >> ~/.ssh/config < EOF

Host git
      HostName 10.10.10.1
      User git
      Port 22
      IdentityFile $HOME/.ssh/id_git

EOF
# Probar con:
ssh -vvv git
```

C.2. Servicios Virtualizados

C.2.1. Maquina Virtual de Xen para Moodle

Para el ambiente de Moodle (LMS contra el cual se ha llevado el desarrollo), se crea una maquina virtual de Debian Stretch (9) con 1 GiB de RAM, 1 CPU virtual, 6 GiB de disco, 512 MiB de intercambio y una dirección IP fija de 10.10.10.10. Los resultados del mismo comando se puede ver en la figura C.1.

```
xen-create-image --hostname=debian-moodle\
        --ip=10.10.10.10 --netmask=255.255.255.0\
        --gateway=10.10.10.1 --memory=1024mb\
        --vcpus=1 --lvm=Xephyr-VG --pygrub\
        --dist=stretch --force --size=6144mb\
        --swap=512mb
```

Se renombró el archivo de configuración de la maquina virtual generado en el paso anterior para temas de consistencia.

```
mv /etc/xen/debian-moodle.cfg\
        /etc/xen/domU-debian-moodle.cfg
```

Un bug de Xen-Tools causa que no se instala correctamente un núcleo de Linux en la maquina virtual y por lo tanto es necesario entrar al mismo con un Chroot y instalar los paquetes faltantes (y hacer las adecuadas configuraciones para permitir su arranque independiente de ayuda externa)[1].

[1] 6 meses despues de la redaccion de estos pasos, y Xen-Tools ya no tiene este bug, motivo por el cual se puede emitir este paso si es que la maquina virtual esta funcionando de forma correcta.

```
General Information
-------------------
Hostname       :  debian-moodle
Distribution   :  stretch
Mirror         :  http://httpredir.debian.org/debian
Partitions     :  swap            512mb (swap)
                  /               6144mb (ext4)
Image type     :  full
Memory size    :  1024mb
Bootloader     :  pygrub

Networking Information
----------------------
IP Address 1   : 10.10.10.10 [MAC: 00:16:3E:4F:72:84]
Netmask        : 255.255.255.0
Gateway        : 10.10.10.1


Creating swap on /dev/Xephyr-VG/debian-moodle-swap
Done

Creating ext4 filesystem on /dev/Xephyr-VG/debian-moodle-disk
Done
Installation method: debootstrap
Done

Running hooks
Done

No role scripts were specified.  Skipping

Creating Xen configuration file
Done

No role scripts were specified.  Skipping
Setting up root password
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
All done


Logfile produced at:
        /var/log/xen-tools/debian-moodle.log

Installation Summary
--------------------
Hostname        :  debian-moodle
Distribution    :  stretch
MAC Address     :  00:16:3E:4F:72:84
IP Address(es)  :  10.10.10.10
SSH Fingerprint :  SHA256:sDJVDxSgNioXe5JFLHalJ+f9WLsLc1+UQrFrwlL48I4 (DSA)
SSH Fingerprint :  SHA256:Lxs1CbEcWBMM2LWgcDIovcWWhrR8D6xG79V7V2wJzQY (RSA)
Root Password   :  N/A
```

FIGURA C.1: Crear maquina virtual para Moodle.

```
        mount /dev/Xephyr-VG/debian-moodle-disk /mnt
        mount -o bind /proc /mnt/proc
        mount -o bind /sys /mnt/sys
        mount -o bind /dev /mnt/dev
        cp /etc/resolv.conf /mnt/etc/resolv.conf
        chroot /mnt /bin/bash
        apt install linux-image-amd64
        vim.tiny /boot/grub/menu.lst
        # Revisar que los archivos referenciados existen
        #                de verdad por ejemplo:
        # Replace initrd.img- con initrd.img
    # guarda y sale
        exit
        umount /mnt/proc
        umount /mnt/sys
        umount /mnt/dev
        umount /mnt
```

Para levantar la maquina virtual:

```
xl create /etc/xen/domU-debian-moodle.cfg -c
```

Se debe seleccionar la tercera opción (Default Kernel).

Se puede dar una revisión a la configuración de red para asegurarse de que esta correcto:

```
vim.tiny /etc/network/interfaces
```

Debe contener:

```
auto eth0
iface eth0 inet static
 address 10.10.10.10
 gateway 10.10.10.1
 netmask 255.255.255.0
```

En el presente caso, Xen-Tools logró configurar esta parte de forma correcta.

A continuación se procede con la instalación de Moodle:

```
# actualiza el sistema
apt update
apt upgrade

# instala dependencias
apt install apache2 php7.0 mysql-server php7.0-mysql
apt install libapache2-mod-php7.0 php7.0-gd php7.0-curl
apt install php-xml php-zip php-mbstring php-soap
apt install php7.0-xmlrpc php7.0-intl
vim.tiny /etc/php/7.0/apache2/php.ini

# agrega:
extension=mysql.so
extension=gd.so

# edita:
```

```
memory_limit = 40M
# dejado con el valor por defecto de 128M

post_max_size = 80M
upload_max_filesize = 80M

# guarda y sale

# reinicia apache para coger los cambios
systemctl restart apache2
```

A continuación se configura la base de datos:

```
# clave de root es root
mysqladmin -u root password "root"

# iniciar session como root
mysql -u root -p

# crear base de datos y hacer que ocupa UTF-8
mysql> CREATE DATABASE moodle;
mysql> ALTER DATABASE moodle charset=utf8;
mysql> exit;

## No se implemento ##
# Moodle queja de UTF8

# Se podria arreglar
# (antes de instalar Moodle)
# con:

mysql -u root -p

mysql> ALTER DATABASE moodle charset=utf8mb4;
mysql> exit;
######################

systemctl restart mysql
```

Se realiza la instalación de la ultima versión de Moodle (3.3 con sus respectivos patches de fallas desde que el mismo salio):

```
# descargar
wget https://download.moodle.org/download.php/direct/stable33/\
        moodle-latest-33.tgz

# descomprimir
tar -zxvf moodle-latest-33.tgz

# meter en la ubicacion para apache
mv moodle /var/www
```

```
# ir a ubicacion para apache
cd /var/www

# crear ubicacion para datos
mkdir moodledata

# arreglar permisos
chown -R www-data:www-data moodle
chown -R www-data:www-data moodledata
chmod -R 755 moodle
chmod -R 755 moodledata

# modificar configuracion de apache
vim.tiny /etc/apache2/sites-available/000-default.conf

# editar
DocumentRoot "/var/www/moodle"

# guardar y salir

# reiniciar apache para aplicar los cambios
systemctl restart apache2
```

Arreglar la base de datos para que acepte conexiones desde Moodle:

```
mysql -u root -p

GRANT ALL PRIVILEGES on *.* to
        'root'@'localhost' IDENTIFIED BY 'root';
GRANT ALL PRIVILEGES on *.* to
        'root'@'localhost' IDENTIFIED BY 'root';
FLUSH PRIVILEGES;
exit;
```

Para seguir con la instalación se abre un navegador con la dirección `http://10.10.10.10/` para seguir las instrucciones que se le lleve por toda la configuración inicial del Moodle.

Al final se agregue un trabajo de cron para ayudar con los tareas periódicas que Moodle requiere para su mantenimiento continuo:

```
crontab -u www-data -e
# add line:
*/10 * * * * /usr/bin/php
                /var/www/moodle/admin/cli/cron.php
                        >/dev/null
```

Estos pasos fueron adaptados de la guía oficial del proyecto de Moodle para instalación en Debian (Moodle Community, 2014).

Ahora que todo esta funcionando se recomienda editar /boot/grub/menu.lst para comentar las entradas del pyGrub que son defectuosas (los primeros dos) para que se puede levantar la maquina virtual sin intervención humana.

C.2.2. Máquina Virtual de Xen

Para crear el servidor de Git con GitLab Community Edition, se va a empezar con las mismas piezas del servidor del LMS/Moodle, es decir una maquina virtual de Debian 9. El mismo se crea como una maquina virtual de Debian Stretch (9) con 1 GiB de RAM, 1 CPU virtual, 6 GiB de disco, 512 MiB de intercambio y una dirección IP fija de 10.10.10.11.

```
xen-create-image --hostname=debian-gitlab\
        --ip=10.10.10.11 --netmask=255.255.255.0\
        --gateway=10.10.10.1 --memory=1024mb\
        --vcpus=1 --lvm=Xephyr-VG --pygrub\
        --dist=stretch --force --size=6144mb\
        --swap=512mb
```

Se renombró el archivo de configuración de la maquina virtual generado en el paso anterior para temas de consistencia.

```
mv /etc/xen/debian-gitlab.cfg\
        /etc/xen/domU-debian-gitlab.cfg
```

Un bug de Xen-Tools causa que no se instala correctamente un núcleo de Linux en la maquina virtual y por lo tanto es necesario entrar al mismo con un Chroot y instalar los paquetes faltantes (y hacer las adecuadas configuraciones para permitir su arranque independiente de ayuda externa)[2].

```
mount /dev/Xephyr-VG/debian-gitlab-disk /mnt
mount -o bind /proc /mnt/proc
mount -o bind /sys /mnt/sys
mount -o bind /dev /mnt/dev
cp /etc/resolv.conf /mnt/etc/resolv.conf
chroot /mnt /bin/bash
apt install linux-image-amd64
vim.tiny /boot/grub/menu.lst
# Revisar que los archivos referenciados existen
#               de verdad por ejemplo:
# Replace initrd.img- con initrd.img
# guarda y sale
exit
umount /mnt/proc
umount /mnt/sys
umount /mnt/dev
umount /mnt
```

Para levantar la maquina virtual:

```
xl create /etc/xen/domU-debian-gitlab.cfg -c
```

Se debe seleccionar la tercera opción (Default Kernel).

Se puede dar una revisión a la configuración de red para asegurarse de que esta correcto:

```
vim.tiny /etc/network/interfaces
```

Debe contener:

[2] 6 meses despues de la redaccion de estos pasos, y Xen-Tools ya no tiene este bug, motivo por el cual se puede emitir este paso si es que la maquina virtual esta funcionando de forma correcta.

```
auto eth0
iface eth0 inet static
 address 10.10.10.11
 gateway 10.10.10.1
 netmask 255.255.255.0
```

En el presente caso, Xen-Tools logró configurar esta parte de forma correcta.
A continuación se procede con la instalación de Gitlab:

```
# actualiza el sistema
apt update
apt upgrade

# A partir del Debian 9, Gitlab CE esta ofrecido
#    en los repositorios oficiales:
apt install gitlab

# Arreglar problema con API
vim.tiny /usr/share/gitlab-shell/config.yml
# cambiar gitlab_url a "http://10.10.10.11/"

/usr/share/gitlab-shell/bin/check
```

Visitar `http://10.10.10.11/` y configurar la contraseña del usuario root. Con el usuario root configurado se procede a crear un usuario especial para la aplicación con el nombre de usuario 'GitEdu'. Este usuario lo damos permisos de administración y también se genera un token en esta dirección: `http://10.10.10.11/profile/personal_access_tokens`. El token se debe guardar para su uso después (GitLab nunca le vuelve a mostrar).

Ahora que todo esta funcionando se recomienda editar /boot/grub/menu.lst para comentar las entradas del pyGrub que son defectuosas (los primeros dos) para que se puede levantar la maquina virtual sin intervención humana.

C.3. Kubernetes

C.3.1. Instalación de Kubernetes

Para instalar la ultima version del cliente de Kubernetes, tanto en desarrollo como en produccion, solo se necesita bajar la ultima version de kubectl desde los repositorios de Google y poner el mismo dentro del $PATH:

```
# Bajar la ultima version de Kubectl para Linux:
curl -LO https://storage.googleapis.com/kubernetes-release/\
        release/$(curl -s https://storage.googleapis.com/\
        kubernetes-release/release/stable.txt)\
        /bin/linux/amd64/kubectl
# Hacerlo ejecutable
chmod +x kubectl
# Revisar el $PATH actual
echo $PATH
# Ubicar dentro del $PATH
mv kubectl ~/bin/
# verificar instalacion con:
```

```
kubectl --help
# o con:
kubectl version
# Este segundo commando debe dar un error de servidor hasta\
#       que se instala y configura el servidor al cual se\
#       debe connectar
```

C.3.2. MiniKube

Dentro del ambiente de desarrollo, fue ocupado un cluster de Kubernetes, de solo un nodo, virtualizado en VirtualBox. La instalacion del mismo se detalle a continuacion:

```
# Se descarga desde los repositorios de Google el commando de
    MiniKube
curl -Lo minikube https://storage.googleapis.com/minikube/
    releases/v0.23.0/minikube-linux-amd64
# Se lo hace ejecutable
chmod +x minikube
# se lo ubica en el $PATH
mv minikube ~/bin/
# para validar la instalacion:
minikube --help
# o
minikube version
```

Para levantar el cluster de MiniKube y realizar la configuracion automatica del entorno para el uso del mismo, se utiliza el commando:

```
minikube start
```

Para bajar el cluster de MiniKube:

```
minikube stop
```

C.4. Docker

C.4.1. Tipo de Contenedor: Shell-Executor

Debian

El Debian Shell-Executor se crea con los siguientes lineas en el Dockerfile:

```
FROM debian:stretch

ENV shell=/bin/sh
ENV user=user

# Greatly increases image size, optional:
#RUN apt-get update && apt-get install -y lsb-release

RUN mkdir -p /code && echo "USER=$user" && echo "SHELL=$shell
    " && useradd -ms $shell $user && chown -v $user:$user /
    code
```

```
VOLUME ["/code"]

ENTRYPOINT ["/bin/sh"]
CMD ["/code/exec.sh"]
```

La primera linea, "FROM ..." define el imagen base, en este caso la versión actual de Debian Stretch (Debian 9.x). La tercera y cuarta linea, "ENV ...", definen variables de entorno que se puede cambiar previa a la construcción del imagen en adición a sus valores por defecto. La sexta y séptima linea, documenta y define un paso opcional donde se instala un paquete adicional, lsb-release, que provee información estandarizado de la distribución de Linux en ejecución. Este paso opcional agrega mucho peso al contenedor y por lo tanto no se lo ha dejado comentado. La novena y décima linea define comandos preliminares para preparar el contenedor como:

- Crear una carpeta /code"donde se puede copiar el código a ser ejecutado desde afuera del contenedor
- Visualización los variables de entorno para temas informativos y de debug
- La creación del usuario que vamos a utilizar dentro del contenedor
- Dar el usuario los permisos adecuados para que tiene acceso a la carpeta creada

La onceava linea define la carpeta creada como un volumen de Docker, el cual se monta externamente al crear y ejecutar el contenedor. La decimotercera linea define el argumento cero del contenedor al momento de iniciar la misma, en este caso el shell que se encuentra del contenedor. La decimocuarta linea define los argumentos con que se debe ejecutar el comando definido previamente, en este caso el script de entrada, cargado desde un sistema de ficheros externo al contenedor.

Alpine Linux

El Alpine Shell-Executor se crea con los siguientes lineas en el Dockerfile:

```
FROM alpine:3.6

ENV shell=/bin/sh
ENV user=user

RUN mkdir -p /code && echo "USER=$user" && echo "SHELL=$shell"\
&& echo "SHELL is not used in this Dockerfile" &&\
    adduser -D $user && chown -v $user:$user /code
VOLUME ["/code"]

ENTRYPOINT ["/bin/sh"]
CMD ["/code/exec.sh"]
```

La primera linea, "FROM ..." define el imagen base, en este caso la versión actual de Alpine Linux (Alpine 3.6). La tercera y cuarta linea, "ENV ...", definen variables de entorno que se puede cambiar previa a la construcción del imagen en adición a sus valores por defecto. La sexta, séptima y octava lineas define comandos preliminares para preparar el contenedor como:

- Crear una carpeta /code"donde se puede copiar el código a ser ejecutado desde afuera del contenedor

- Visualización los variables de entorno para temas informativos y de debug
- La creación del usuario que vamos a utilizar dentro del contenedor
- Dar el usuario los permisos adecuados para que tiene acceso a la carpeta creada

La novena linea define la carpeta creada como un volumen de Docker, el cual se monta externamente al crear y ejecutar el contenedor. La onceava linea define el argumento cero del contenedor al momento de iniciar la misma, en este caso el shell que se encuentra del contenedor. La duodécima linea define los argumentos con que se debe ejecutar el comando definido previamente, en este caso el script de entrada, cargado desde un sistema de ficheros externo al contenedor.

Plantilla de Ejecuccion

De acuerdo con el punto de entrada de los Dockerfiles definidos previamente, se realizo una plantilla de ejecucion para codigo Shell (`https://gitlab.com/nishedcob/shell-co` con el siguiente archivo como definicion del `exec.sh`:

```
#! /bin/sh

DEBUG=false
NULL="/dev/null"
PRINT_NULL="> $NULL 2> $NULL"

if $DEBUG; then
        echo ""

        whoami
        echo "says hello world!"
        echo "from Docker!"

        echo ""

        lsb_release -a
        uname -a

        echo ""

        cat /etc/os-release

        echo ""
fi

/code/main.sh
```

Si se define la bandera/variable `DEBUG` como true, el mismo script imprime mucha informacion acerca del sistema operativo en uso, caso contrario se procede directamente a la ejecuccion del archivo `main.sh` del mismo repositorio.

C.4.2. Tipo de Contenedor: Python3-Executor

Debian

El Debian Python3-Executor se define de la siguiente manera a través de su Dockerfile:

```
FROM registry.gitlab.com/nishedcob/gitedu/shell-executor:
    debian-stretch

RUN apt-get update && apt-get install -y python3 python3-dev
    python3-pip virtualenv

VOLUME ["/code"]
WORKDIR "/code"
```

La primera y segunda linea define herencia del contenedor Debian Shell-Executor que definimos previamente. La tercera y cuarta linea instala Python3, Pip3 y Virtualenv dentro del contenedor. La sexta linea vuelve a declarar el volumen de Docker porque al aparecer la versión actual de Docker a propósito no suporta herencia de esta linea en los Dockerfile. La séptima linea define la ubicación inicial utilizada cuando se inicia el contenedor.

Alpine Linux

El Alpine Python3-Executor se define a través del Dockerfile que esta a continuación:

```
FROM registry.gitlab.com/nishedcob/gitedu/shell-executor:
    alpine-3.6

RUN apk update && apk add python3 python3-dev py-virtualenv

VOLUME ["/code"]
WORKDIR "/code"
```

La primera y segunda linea define herencia del contenedor Alpine Shell-Executor que definimos previamente. La tercera linea instala Python3, Pip3 y Virtualenv dentro del contenedor. La quinta linea vuelve a declarar el volumen de Docker porque al aparecer la versión actual de Docker a propósito no suporta herencia de esta linea en los Dockerfile. La sexta linea define la ubicación inicial utilizada cuando se inicia el contenedor.

Plantilla de Ejecuccion

De acuerdo con el punto de entrada de los Dockerfiles definidos previamente, se realizo una plantilla de ejecucion para codigo Python 3 (`https://gitlab.com/nishedcob/python3-c` con el siguiente archivo como definicion del `exec.sh`:

```
#! /bin/sh

DEBUG=false
NULL="/dev/null"
PRINT_NULL="> $NULL 2> $NULL"
```

```
if $DEBUG; then
        echo ""

        whoami
        echo "says␣hello␣world!"
        echo "from␣Docker!"

        echo ""

        lsb_release -a
        uname -a

        echo ""

        cat /etc/os-release

        echo ""
fi

os_id=$(grep "^\(ID\|id\)=" /etc/os-release | awk -F= '{print
    $2}')

if $DEBUG; then
        echo "Detected␣OS␣ID:␣$os_id"
fi

env_dir=env-$os_id

if $DEBUG; then
        echo "System␣Python␣3:"
        python3 --version
fi

if [ -d $env_dir ] ; then
    rm_env="rm␣-rdv␣$env_dir"
    if [ -d $env_dir/bin ]; then
        if [ -f $env_dir/bin/python3 ]; then
                    if $DEBUG; then
                echo "VirtualEnv␣Python␣3:"
                $env_dir/bin/python3 --version
                    fi
        else
                    if $DEBUG; then
                echo "No␣VirtualEnv␣Python"
                eval $rm_env
            else
                eval "$rm_env␣$PRINT_NULL"
                    fi
        fi
    else
                if $DEBUG; then
```

```
            echo "No␣VirtualEnv␣Bin"
            eval $rm_env
        else
            eval "$rm_env␣$PRINT_NULL"
                fi
    fi
else
    if $DEBUG; then
        echo "No␣VirtualEnv"
    fi
fi

if $DEBUG; then
    echo ""

    echo "=========================="
    echo "|␣VirtualEnv␣Management:␣|"
    echo "=========================="
fi

if [ ! -d $env_dir ]; then
    if $DEBUG; then
        echo "Creating␣VirtualEnv..."
        echo "--------------------"
    fi

        create_virtualenv="virtualenv␣--python=python3␣
           $env_dir"
    if $DEBUG; then
        eval $create_virtualenv
    else
        eval "$create_virtualenv␣$PRINT_NULL"
    fi

    if $DEBUG; then
        echo "----------------------------------------"
    fi
fi

if $DEBUG; then
    echo "Installing␣Dependencies␣with␣Pip..."
    echo "----------------------------------------"
fi

pip_install="$env_dir/bin/pip3␣install␣-r␣requirements.txt"
if $DEBUG; then
    eval $pip_install
else
    eval "$pip_install␣$PRINT_NULL"
fi
```

```
if $DEBUG; then
    echo ""

    echo "====================="
    echo "| Executing Python: |"
    echo "====================="
fi

$env_dir/bin/python3 main.py
```

Si se define la bandera/variable `DEBUG` como true, el mismo script imprime mucha informacion acerca del sistema operativo en uso y las operaciones necesarias previo a la ejecucion del codigo del usuario, caso contrario se realiza las operaciones necesarias sin imprimir nada previo a la ejecuccion del archivo `main.py` del mismo repositorio.

C.4.3. Tipo de Contenedor: PostgreSQL-Executor

Debian

El Debian PostgreSQL-Executor se define con el siguiente Dockerfile:

```
FROM registry.gitlab.com/nishedcob/gitedu/shell-executor:
    debian-stretch

RUN apt-get update && apt-get install -y postgresql
    postgresql-client

RUN echo "Starting PostgreSQL Cluster..." ; /usr/bin/
    pg_ctlcluster 9.6 main start && echo "Started cluster!" ||
     echo "Failed to start cluster!"; su - postgres -c "
    createuser user && createdb -O user userdb"; echo "
    Stopping PostgreSQL Cluster..." ; /usr/bin/pg_ctlcluster
    9.6 main stop && echo "Stopped cluster!" || echo "Failed
    to stop cluster!";

VOLUME ["/code"]
WORKDIR "/code"
```

La primera y segunda linea define herencia del contenedor Debian Shell-Executor que definimos previamente. La tercera y cuarta linea instala PostgreSQL dentro del contenedor. La sexta hasta décima linea levanta el motor de base de datos PostgreSQL con la finalidad de crear un usuario y base de datos por defecto sobre el cual se puede trabajar. Finalizando este proceso se desactiva el contenedor para reducir el tamaño del mismo y no introducir comportamiento desconocido. La duodécima linea vuelve a declarar el volumen de Docker porque al aparecer la versión actual de Docker a propósito no suporta herencia de esta linea en los Dockerfile. La decimotercera linea define la ubicación inicial utilizada cuando se inicia el contenedor.

Alpine Linux

El Alpine PostgreSQL-Executor se define con el siguiente Dockerfile:

```
FROM registry.gitlab.com/nishedcob/gitedu/shell-executor:
    alpine-3.6

RUN apk update && apk add postgresql && su - postgres -c "
    export␣PGDATA=/var/lib/postgresql/data␣&&␣initdb"

RUN echo "Starting␣PostgreSQL␣Cluster..." ; mkdir -p /run/
    postgresql && chown -R postgres:postgres /run/postgresql
    && chmod 755 /run/postgresql && mkdir -p /var/run/
    postgresql && chown -R postgres:postgres /var/run/
    postgresql && chmod 2777 /var/run/postgresql && su -
    postgres -c "export␣PGDATA=/var/lib/postgresql/data␣&&␣
    postgres␣&" && echo "Started␣cluster!" || echo "Failed␣to␣
    start␣cluster!"; sleep 5s && netstat -tupln && su -
    postgres -c "createuser␣user␣&&␣createdb␣-O␣user␣userdb";
    echo "Stopping␣PostgreSQL␣Cluster..." ; killall postgres
    && echo "Stopped␣cluster!" || echo "Failed␣to␣stop␣cluster
    !";

VOLUME ["/code"]
WORKDIR "/code"
```

La primera y segunda linea define herencia del contenedor Alpine Shell-Executor que definimos previamente. La tercera y cuarta linea instala PostgreSQL dentro del contenedor en adición a inicializar la base de datos. La sexta hasta decimoséptima linea levanta el motor de base de datos PostgreSQL con la finalidad de crear un usuario y base de datos por defecto sobre el cual se puede trabajar. Finalizando este proceso se desactiva el contenedor para reducir el tamaño del mismo y no introducir comportamiento desconocido. La decimonovena linea vuelve a declarar el volumen de Docker porque al aparecer la versión actual de Docker a propósito no suporta herencia de esta linea en los Dockerfile. La vigésima linea define la ubicación inicial utilizada cuando se inicia el contenedor.

Plantilla de Ejecuccion

De acuerdo con el punto de entrada de los Dockerfiles definidos previamente, se realizo una plantilla de ejecucion para codigo SQL de PostgreSQL (`https://gitlab.com/nishedcob/p` con el siguiente archivo como definicion del `exec.sh`:

```
#! /bin/sh

DEBUG=false
NULL="/dev/null"
PRINT_NULL=">␣$NULL␣2>␣$NULL"

if $DEBUG; then
        echo ""

        whoami
        echo "says␣hello␣world!"
        echo "from␣Docker!"
```

```
        echo ""

        lsb_release -a
        uname -a

        echo ""

        cat /etc/os-release

        echo ""
fi

os_id=$(grep "^\(ID\|id\)=" /etc/os-release | awk -F= '{print
    $2}')

if $DEBUG; then
        echo "Detected OS ID: $os_id"

        echo ""
fi

if $DEBUG; then
        echo "Starting PostgreSQL Cluster..."
fi

if [ "$os_id" = 'debian' ] ; then
        debian_pg_start="/usr/bin/pg_ctlcluster 9.6 main
           start"
        if $DEBUG; then
                eval $debian_pg_start && echo "Started
                   cluster!" || echo "Failed to start cluster
                   !";
        else
                eval "$debian_pg_start $PRINT_NULL"
        fi
else
    if [ "$os_id" = 'alpine' ] ; then
        md="mkdir -vp"
        pg_r_dir="/run/postgresql"
        pg_vr_dir="/var/run/postgresql"
        pg_data="/var/lib/postgresql/data"
        pg_user="postgres"
        pg_cmd="postgres &"
        ch_own="chown -R $pg_user:$pg_user"
        ch_mod_rd="chmod -v 775 $pg_r_dir"
        ch_mod_vrd="chmod -v 2777 $pg_vr_dir"
        su_pg="su - $pg_user -c"
        exprt="export PGDATA=$pg_data"
        start_pg="$su_pg \"$exprt && $pg_cmd\""
        validation="echo \"Started cluster!\" || echo \"
           Failed to start cluster!\""
```

```
        if $DEBUG; then
                cmd="$md $pg_r_dir && $ch_own $pg_r_dir &&
                    $ch_mod_rd && $md $pg_vr_dir && $ch_own
                    $pg_vr_dir && $ch_mod_vrd && $start_pg &&
                    $validation"
        else
                cmd="$md $pg_r_dir $PRINT_NULL && $ch_own
                    $pg_r_dir $PRINT_NULL && $ch_mod_rd
                    $PRINT_NULL && $md $pg_vr_dir $PRINT_NULL
                    && $ch_own $pg_vr_dir $PRINT_NULL &&
                    $ch_mod_vrd $PRINT_NULL && $start_pg
                    $PRINT_NULL"
        fi
        eval $cmd
        sleep 10s
    else
        if $DEBUG; then
                echo "Unknown/Unsupported OS";
        fi
        exit 1;
    fi
fi

if $DEBUG; then
        echo ""
fi

init_db="su - user -c \"psql -U user userdb -f /code/init.sql
    \""
if $DEBUG; then
        eval $init_db
else
        eval "$init_db $PRINT_NULL"
fi
su - user -c "psql -U user userdb -f /code/main.sql"
```

Si se define la bandera/variable `DEBUG` como true, el mismo script imprime mucha informacion acerca del sistema operativo en uso y las operaciones necesarias previo a la ejecucion del codigo del usuario, incluyendo el levantamiento de la base de datos y inicializacion del mismo, caso contrario se realiza las operaciones necesarias sin imprimir nada previo a la ejecuccion del archivo `main.sql` del mismo repositorio.

Apéndice D

Desarollo de GitEDU

D.1. Configuracion de la Base de Datos Relacional

Primero es necesario crear y configurar un usuario y base de datos para ser ocupado:

```
postgresql-setup initdb
systemctl enable postgresql
systemctl start postgresql
systemctl status postgresql
vim /etc/postgresql/9.6/main/pg_hba.conf
# agregar una linea antes de las lineas similares y
# que diga (sin el numeral adelante):
#local   all       postgres                peer
systemctl restart postgresql
systemctl status postgresql
su - postgres
psql

        postgres=# CREATE USER giteduser WITH PASSWORD 'g1T3d_$3r';
        postgres=# CREATE DATABASE gitedudb WITH OWNER giteduser;
        postgres=# \q
psql gitedudb -U postgres
        gitedudb=# CREATE SCHEMA giteduapp AUTHORIZATION giteduser;
        gitedudb=# ALTER USER giteduser SET search_path TO giteduapp;
        gitedudb=# \q
psql gitedudb -U giteduser
        gitedudb=> SELECT current_schema();
                Debe decir: giteduapp
        gitedudb=> \q
```

Segundo, se abre el settings.py (dentro de la carpeta GitEDU/GitEDU) y se remplaza (para desarrollo, en producción debe llevar valores distintos) el atributo DATABASES con lo siguiente:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'gitedudb',
        'USER': 'giteduser',
        'PASSWORD': 'g1T3d_$3r',
        'HOST': '127.0.0.1',
        'PORT': '5432',
```

```
    }
}
```

D.2. Configuracion de la Base de Datos No Relacional

Para crear la base de datos de MongoDB:

mongo

```
use gitEduDB
db.createUser(
    {
        user: "gitEduUser",
        pwd: "G1TedU$3r",
        roles: [ "readWrite", "dbAdmin" ]
    }
)
```

El mismo se define en el settings de la siguiente manera:

```
NOSQL_DATABASES = {
    'nosql': {
        'NAME': 'gitEduDB',
        'USER': "gitEduUser",
        'PASSWORD': 'G1TedU$3r',
        'HOST': '127.0.0.1',
        'PORT': '27017',
    }
}
```

D.3. Compatibilidad con EduNube en el Mismo Repositorio

Para reducir el numero de repositorios involucrados en este trabajo de titulación, se ha optado por llevar el desarrollo de EduNube dentro del mismo repositorio, con una separación de dependencias con otro entorno virtual y aislamiento de código en desarrollo con varias ramas de Git. Por lo tanto, se realizo una nueva rama de Git con el comando:

```
git checkout -b gitedu
```

Y una refacturación del entorno virtual a ser "env-ge" en lugar de "env", en lugar de "requirements.txt", utilizar "requirements.ge.txt" y un nuevo script de activar el entorno, el cual se activa ahora con "source activate-ge.sh":

```
#! /usr/bin/head -n 2
# run with 'source activate-ge.sh'
PROJECT=ge
ENV_DIR=env-$PROJECT
if [ ! -d $ENV_DIR ]; then
        virtualenv --python=python3 $ENV_DIR
fi
source $ENV_DIR/bin/activate
pip3 install -r requirements.$PROJECT.txt
```

Para desarrollar ambos sistemas en paralelo, y poder trabajar en ramas independientes con ambos sistemas levantados al mismo tiempo para probar y desarrollar características de integración, se clono de forma local el repositorio:

```
git clone GitEDU GitEDU-copy
```

Para el desarrollo se ocupa el puerto 8000 de localhost para GitEDU y el puerto 8001 de localhost para EduNube:

```
# Levantar GitEDU en el ambiente de desarrollo:
cd GitEDU-copy
source activate-ge.sh
git checkout gitedu
cd GitEDU
python manage.py runserver 8000

# Levantar EduNube en el ambiente de desarrollo:
cd GitEDU
source activate-en.sh
git checkout edunube
cd EduNube
python manage.py runserver 8001
```

[1]

D.4. Autenticacion por LTI

Primero es necesario que este activo y levantado el ambiente del LMS como se documenta en la sección 4.1.2 de instalación de Moodle. Para el desarrollo de este trabajo de titulación se esta considerando una instalación en un servidor aparte (virtualizado en el mismo equipo) en la dirección IP 10.10.10.10.

Instalación de dependencias desde GitHub (no se encuentran en los repositorios oficiales de Pip/Pypi; además para superar problemas de dependencias en las librerías, se ha optado para ocupar forks personales del autor con las mejores necesarias para su funcionamiento):

```
pip install git+https://github.com/nishedcob/django-app-lti
    @master#egg=django-app-lti
pip install git+https://github.com/nishedcob/
    django-auth-lti@master#egg=django-auth-lti
```

Como son dependencias no se encuentran en los repositorios oficiales, su manejo dentro del requirements.txt también tiene que ser especial ya que un 'pip freeze' no los guardaran correctamente dentro del mismo. En lugar de eso hay que agregar dos lineas al requirements.txt para el manejo de estas dependencias:

```
-e git+https://github.com/nishedcob/django-app-lti.git
    @38b32989e22b189345e421b183684f9b5453e99a
    #egg=django-app-lti
-e git+https://github.com/nishedcob/django-auth-lti.git
    @71c9da8d0aa07ebc3139bf3f113b5c521d61b1f1
    #egg=django-auth-lti
```

[1] Para produccion, se cambio el puerto 8001 de EduNube para 8010

Se pone a editar el settings.py (dentro de GitEDU/GitEDU) con las siguientes configuraciones:

- a INSTALLED_APPS agregamos las siguientes lineas:

```
'django_auth_lti',
'django_app_lti',
```

- a MIDDLEWARE agregamos la siguiente linea:

```
'django_auth_lti.middleware.LTIAuthMiddleware',
```

- a AUTHENTICATION_BACKENDS agregamos la siguiente linea:

```
'django_auth_lti.backends.LTIAuthBackend',
```

  Si es que no existe AUTHENTICATION_BACKENDS lo creamos con los siguientes valores:

```
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
    'django_auth_lti.backends.LTIAuthBackend',
)
```

- también agregamos los siguientes atributos:

  - LTI_SETUP

```
LTI_SETUP = {
    "TOOL_TITLE": "GitEDU",
    "TOOL_DESCRIPTION": "Sistema para Programar en Linea",
    "LAUNCH_URL": "lti:launch",
    "LAUNCH_REDIRECT_URL": "ideApp:decode",
    "INITIALIZE_MODELS": False,
    "EXTENSION_PARAMETERS": {
        "10.10.10.10": {
            "privacy_level": "public",
            "course_navigation": {
                "enabled": "true",
                "default": "disabled",
                "text": "GitEDU LMS Playground",
            }
        }
    }
}
```

  - LTI_OAUTH_CREDENTIALS con texto aleatorio (fue ocupado OpenSSL, específicamente el comando `openssl rand -hex 10` para generar los valores de abajo[2]).

```
LTI_OAUTH_CREDENTIALS = {
    "GitEduLMS_Playground": "b2e0158c3cb4ddb0202d",
        # (Para pruebas)
    "GitEduLMS_Playground_Assignments":
```

[2]Un ambiente de producción debe ocupar valores distintos

```
                "57b3a14734566c49bcaf",
                # (Para deberes/examenes/pruebas/talleres/etc)
            "GitEduLMS_Playground_Classes":
                "f7a0b6accc2631779e84",
                # (Para materias)
        }
```

También se edita el urls.py (dentro de la misma dirección que el settings.py) con las siguientes lineas:

```
from django.conf.urls import include
import django_app_lti.urls

# dentro de:
urlpatterns = [
    # agregar:
    url(r'^lti/', include(django_app_lti.urls, namespace="lti")),
]
```

Para arreglar un problema de una versión muy desactualizada de ims-lti-py, se agrego la siguiente linea al requirements.txt (para ocupar un fork del propio librería por el propio autor para resolver los problemas dados):

```
-e git+https://github.com/nishedcob/ims_lti_py.git
    @a6576d7892ea4f69b76572788b118aaa4cdcf749
    #egg=ims_lti_py-develop
```

Para arreglar un problema de limpieza de datos en oauth2, se agrego la siguiente linea al requirements.txt (para ocupar un fork del propio librería por el propio autor para resolver los problemas dados):

```
-e git+https://github.com/nishedcob/python-oauth2.git
    @176fc35aa35d626afcb6a23459482a4c96782c88
    #egg=oauth2
```

Después se migra la base de datos:

```
python manage.py makemigrations
python manage.py migrate
```

D.5. Migracion para llenar tabla AuthenticationType

Con una migración se puede llenar de forma automática el catalogo de AuthenticacionType:

```
def fill_auth_types(apps, schema_editor):
    auth_type = models.AuthenticationType
    classic = auth_type(name="Clasica")
    classic.save()
    lti = auth_type(name="LTI")
    lti.save()


class Migration(migrations.Migration):
```

```
    dependencies = [
        ('authApp',
           '0002_authenticationtype_userauthentication'),
    ]

    operations = [
        migrations.RunPython(fill_auth_types),
    ]
```

D.6. Accesibilidad a Datos de Autetnicacion de LTI

Primero al settings se agrega dos atributos que indiquen cual llave se considera el sistema para compartir materias y otro para compartir deberes/exámenes/pruebas/talleres/etc...:

```
LTI_ASSIGNMENTS_KEY = 'GitEduLMS_Playground_Assignments'
LTI_CLASSES_KEY = 'GitEduLMS_Playground_Classes'
```

Para controlar las partes de la configuración que se presenta a los usuarios finales, agregamos un nuevo campo de configuración al settings:

```
LTI_CONFIG_EXPOSE = {
    "LTI_KEYS": True,
    "LTI_ASSIGNMENT_KEY": True,
    "LTI_CLASS_KEY": True,
    "LTI_OTHER_KEYS": False,
    "LTI_SETUP": False,
}
```

Para que los profesores (como usuario final de GitEdu) también tengan acceso a estos credenciales para poderlos ocupar, se cree una vista en la ubicación '/auth/lti/credentials' que devuelve un JSON con los datos de LTI:

```
class LTICredentialsView(View):

    def get(self, request):
        if not request.user.is_authenticated:
            raise PermissionError("No tiene acceso a esta
               vista hasta que se autentica...")

        lti_expose = settings.LTI_CONFIG_EXPOSE

        lti_cred_json = {}

        if lti_expose['LTI_KEYS']:
            if lti_expose['LTI_ASSIGNMENT_KEY']:
                lti_cred_json['LTI_ASSIGNMENT_KEY'] = {
                    settings.LTI_ASSIGNMENTS_KEY:
                        settings.LTI_OAUTH_CREDENTIALS
                            [settings.LTI_ASSIGNMENTS_KEY]
                }
            if lti_expose['LTI_CLASS_KEY']:
                lti_cred_json['LTI_CLASS_KEY'] = {
```

```
                    settings.LTI_CLASSES_KEY:
                        settings.LTI_OAUTH_CREDENTIALS
                            [settings.LTI_CLASSES_KEY]
                }
            if lti_expose['LTI_OTHER_KEYS']:
                lti_cred_json['LTI_OTHER_KEYS'] =
                    settings.LTI_OAUTH_CREDENTIALS

        if lti_expose['LTI_SETUP']:
            lti_cred_json['LTI_SETUP'] =
                settings.LTI_SETUP

        return JsonResponse(lti_cred_json)
```

D.7. Connectividad al API de GitLab

La conexión a la API de GitLab se realiza de la siguiente manera:

```
gitlab_default_srv = GITLAB_DEFAULT_SERVER


def connect_to_gitlab_token(protocol=None,
        host=None, port=None, token=None):
    if host is None:
        return None
    # gitlab_conn = gitlab.Gitlab(protocol
            + host + port, token)
    gitlab_conn = gitlab.Gitlab(protocol
            + host, token)
    #gitlab_conn.auth()
    return gitlab_conn


def connect_to_settings_gitlab_token(
        indx=GITLAB_DEFAULT_SERVER):
    return connect_to_gitlab_token(
            protocol=GITLAB_SERVERS
                [indx]['API_PROTOCOL'],
            host=GITLAB_SERVERS[indx]
                ['HOST'],
            port=GITLAB_SERVERS[indx]
                ['API_PROTOCOL'],
            token=GITLAB_SERVERS[indx]
                ['TOKEN'])


def connect_to_gitlab_user_password(protocol=None,
        host=None, port=None, user=None,
        password=None):
    if host is None:
        return None
```

```
    gitlab_conn = gitlab.Gitlab(protocol + host
            + port, email=user, password=password)
    gitlab_conn.auth()
    return gitlab_conn


def connect_to_settings_gitlab_user_password(
        indx=GITLAB_DEFAULT_SERVER):
    return connect_to_gitlab_user_password(
            protocol=GITLAB_SERVERS[indx]
                    ['API_PROTOCOL'],
            host=GITLAB_SERVERS[indx]['HOST'],
            port=GITLAB_SERVERS[indx]
                    ['API_PROTOCOL'],
            user=GITLAB_SERVERS[indx]['USER'],
            password=GITLAB_SERVERS[indx]
                    ['PASSWORD'])


def connect_to_settings_gitlab(
        indx=GITLAB_DEFAULT_SERVER):
    if GITLAB_SERVERS[indx]['WITH_TOKEN']:
        return connect_to_settings_gitlab_token(
                indx)
    elif GITLAB_SERVERS[indx]['WITH_CRED']:
        return connect_to_settings_gitlab_user_password(
                indx)
    else:
        return None

gitlab_srv = None
try:
    gitlab_srv = connect_to_settings_gitlab(
            gitlab_default_srv)
except Exception as e:
    print("No se pudo connectar a GitLab")
    print(e)
```

Apéndice E

Desarollo de EduNube

E.1. Configuracion de Base de Datos Relacional

Primero es necesario crear y configurar un usuario y base de datos para ser ocupado:

```
su - postgres
psql
        postgres=# CREATE USER edunubeser WITH PASSWORD '3d?
            N_6E';
        postgres=# CREATE DATABASE edunubedb WITH OWNER
            edunubeser;
        postgres=# \q
psql edunubedb -U postgres
        edunubedb=# CREATE SCHEMA edunubeapp AUTHORIZATION
            edunubeser;
        edunubedb=# ALTER USER edunubeser SET search_path TO
            edunubeapp;
        edunubedb=# \q
psql edunubedb -U edunubeser
        edunubedb=> SELECT current_schema();
                Debe decir: edunubeapp
        edunubedb=> \q
```

Segundo, se abre el settings.py (dentro de la carpeta EduNube/EduNube) y se remplaza (para desarrollo, en producción debe llevar valores distintos) el atributo DATABASES con lo siguiente:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'edunubedb',
        'USER': 'edunubeser',
        'PASSWORD': '3d?N_6E',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

E.2. Configuracion de Base de Datos No Relacional

Para crear la base de datos de MongoDB:

mongo

```
use eduNubeDB
db.createUser(
    {
        user: "eduNubeUser",
        pwd: "3d?N_6E",
        roles: [ "readWrite", "dbAdmin" ]
    }
)
```

El mismo se define en el settings de la siguiente manera:

```
NOSQL_DATABASES = {
    'nosql': {
        'NAME': 'eduNubeDB',
        'USER': "eduNubeUser",
        'PASSWORD': '3d?N_6E',
        'HOST': '127.0.0.1',
        'PORT': '27017',
    }
}
```

E.3. Implementacion de API Tokens con JWT

A continuación se presenta la lógica para descifrar y cifrar los tokenes con el fin de generar/actualizarlo y también poderlo validar contra una tabla interna de la base de datos a lo que un cliente se lo envía:

```
def decode_api_token(api_token=None):
    if api_token is None:
        raise ValueError("API_Token can't be None")
    return jwt.decode(api_token.token, api_token.secret_key,\
        algorithms=[api_token.token_algo])

def update_api_token(api_token=None, regen_secret_key=False):
    if api_token is None:
        raise ValueError("API_Token can't be None")
    if regen_secret_key or api_token.secret_key is None or\
            len(api_token.secret_key) == 0:
        api_token.secret_key = bcrypt.gensalt()
            # Generate Random Unique Secret_Key
    api_token.edit_date_in_token =\
        api_token.edit_date.__str__()
    payload = {
        'app_name': api_token.app_name,
        'created_date': api_token.created_date.__str__(),
        'edit_date': api_token.edit_date_in_token,
        'expires': api_token.expires
    }
    if api_token.expires:
        if api_token.expire_date is not None:
```

```
            payload['expire_date'] = api_token.expire_date
    api_token.token = jwt.encode(payload,\
        api_token.secret_key, algorithm='HS256')\
        # Generate Token
    api_token.save()
```

Como parte de la administración de este aspecto del sistema de autenticación, se encuentra implementado las funcionalidades de crear, actualizar, leer y borrar estos tokens de autenticación, siempre y cuando el usuario logueado tenga el permiso "auth_admin.manage_tokens". Si no se crea este permiso, como es el caso del ambiente de desarrollo, la política de Django es solo permitir cuentas de superusuario para acceder a estas vistas.

Apéndice F

Pruebas de GitEDU

F.1. Prueba de CodePersistenceBackend

Con respeto al sistema GitEDU, el componente mas importante de probar fue el backend de persistencia de codigo, automatizado de la siguiente manera:

```
from pymongo import ASCENDING, DESCENDING
from ideApp.CodePersistenceBackends.MongoDB.mongodb_models
    import ChangeModel, ChangeFileModel, NamespaceModel,
    RepositoryModel, RepositoryFileModel
from GitEDU.settings import
    CODE_PERSISTENCE_BACKEND_MANAGER_CLASS,
    load_code_persistence_backend_manager

manager = load_code_persistence_backend_manager(
    CODE_PERSISTENCE_BACKEND_MANAGER_CLASS)

print("code␣persistence␣backend␣manager:␣<%s>" %manager)

print("Changes␣in␣Ascending␣Order:")
for change in list(ChangeModel.objects.raw({}).order_by([("
    timestamp", ASCENDING)])):
     print(change)

print("Changes␣in␣Descending␣Order:")
for change in list(ChangeModel.objects.raw({}).order_by([("
    timestamp", DESCENDING)])):
     print(change)

namespace = "nishedcob"
repository = "test"
file_path = "folder/hello.py"

manager.sync("NRF")
namespace_objs = manager.get_namespace(namespace=namespace)
print(namespace_objs)
namespace_obj = manager.select_preferred_backend_object(
    result_set=namespace_objs)
print(namespace_obj)
repository_objs = manager.get_repository(namespace=
    namespace_obj, repository=repository)
```

```
print(repository_objs)
repository_obj = manager.select_preferred_backend_object(
   result_set=repository_objs)
print(repository_obj)
repository_file_objs = manager.get_file(namespace=
   namespace_obj, repository=repository_obj, file_path=
   file_path)
print(repository_file_objs)
repository_file_obj = manager.select_preferred_backend_object
   (result_set=repository_file_objs)
print(repository_file_obj)
print(repository_file_obj.persistence_object)
print(repository_file_obj.persistence_object.pk)
file_edits = ChangeFileModel.objects.raw({
    'file': repository_file_obj.persistence_object.pk,
    'file_path': file_path
})
print(list(file_edits))
edits = []
for file_edit in file_edits:
    print(file_edit.change.pk)
    change = ChangeModel.objects.raw({'_id': file_edit.change
       .pk}).first()
    print(change)
    edit = {
        'id': change.change_id,
        'author': change.author,
        'timestamp': change.timestamp.as_datetime().__str__()
           ,
        'namespace': namespace,
        'repository': repository,
        'file_path': file_edit.file_path
    }
    print(edit)
    edits.append(edit)
    print(edits)
print(edits)
edits = edits.sort(key=lambda k: k['timestamp'])
print(edits)
```

Apéndice G

Pruebas de EduNube

G.1. Expirimento de Extraccion de ID de Ultimo Commit de un Repositorio de Git

```
# coding: utf-8
from apiApp.VirtualizationBackends.Kubernetes import
    KubernetesVirtualizationBackend
kvb = KubernetesVirtualizationBackend()
kvb.get_id_last_git_commit(repository_path='/home/nyx/GitEDU'
    )
kvb.get_id_last_git_commit(repository_path='/home/nyx/GitEDU-
    copy')
```

G.2. Prueba Inicial de Ejecuccion con Kubernetes

```
# coding: utf-8

import time

from apiApp.VirtualizationBackends.Kubernetes import Py3KubernetesVirtualizatio

# values for test:
kvb = Py3KubernetesVirtualizationBackend()
job_name = 'py3-pytest-3'
# with minikube
#git_repo = 'http://192.168.99.1/python3-code-executor-template.git'
# with Kubernetes on Xen
git_repo = 'http://10.10.10.1/python3-code-executor-template.git'
kubernetes_working_tmp_dir = '/tmp'

# prepare test manifest
manifest = kvb.build_job_template(job_name=job_name, git_repo=git_repo)
manifest_path = '%s/%s.json' %(kubernetes_working_tmp_dir, job_name)
kvb.write_json_manifest(path=manifest_path, json_data=manifest, overwrite=True

# create job with manifest
print(kvb.job_status(job_id=job_name))
print(kvb.kubectl_create_from_manifest_file(manifest_path=manifest_path))

# job info
```

```
print(kvb.job_describe(job_id=job_name))

# get job status & wait until job completion
print(kvb.job_get(job_id=job_name))
while not kvb.job_finished(job_id=job_name):
    print(kvb.job_status(job_id=job_name))
    time.sleep(1)
print(kvb.job_status(job_id=job_name))
print(kvb.job_get(job_id=job_name))

# job info
print(kvb.job_describe(job_id=job_name))

# job output
print(kvb.job_logs(job_id=job_name))

# delete job
print(kvb.kubectl_delete_from_manifest_file(manifest_path=manifest_path))
```

G.3. Prueba Inicial de Backend de RepoSpecs

```
# coding: utf-8
from apiApp.Validation import RepoSpec
py3_code_exec_template_repospec = RepoSpec.create(repo="
    python3-code-executor-template", parent=None)
token = py3_code_exec_template_repospec.token
print(token)
secret_key = py3_code_exec_template_repospec.secret_key
print(secret_key)
py3_code_exec_template_repospec = RepoSpec.RepoSpec.objects.
    get(repo="python3-code-executor-template")
# token == py3_code_exec_template_repospec.token
# secret_key == py3_code_exec_template_repospec.secret_key
# RepoSpec.validate_repospec(py3_code_exec_template_repospec.
    token) == True
# RepoSpec.decode_repospec(repospec=
    py3_code_exec_template_repospec.token,
#                          stored_repospec=
    py3_code_exec_template_repospec)
#                          == {'repo': "python3-code-executor
    -template"}
# RepoSpec.decode_repospec(repospec=
    py3_code_exec_template_repospec.token) == {'repo': "
    python3-code-executor-template"}
# RepoSpec.decode(stored_repospec=
    py3_code_exec_template_repospec, repospec=None,
    decode_stored=True)
#                  == {'repo': "python3-code-executor-
    template"}
```

```
py3_code_exec_template_repospec = RepoSpec.update_repospec(
    repospec=py3_code_exec_template_repospec, parent="http
    ://192.168.1.100/shell.git", repo=
    py3_code_exec_template_repospec.repo, regen_secret_key=
    False)
# token != py3_code_exec_template_repospec.token
# secret_key == py3_code_exec_template_repospec.secret_key
token = py3_code_exec_template_repospec.token
print(token)
secret_key = py3_code_exec_template_repospec.secret_key
print(secret_key)
py3_code_exec_template_repospec = RepoSpec.update_repospec(
    repospec=py3_code_exec_template_repospec, parent="http
    ://192.168.1.100/shell.git", repo=
    py3_code_exec_template_repospec.repo, regen_secret_key=
    True)
# token != py3_code_exec_template_repospec.token
# secret_key != py3_code_exec_template_repospec.secret_key
token = py3_code_exec_template_repospec.token
print(token)
secret_key = py3_code_exec_template_repospec.secret_key
print(secret_key)
# RepoSpec.validate_repospec(repospec=
    py3_code_exec_template_repospec.token) == True
py3_code_exec_template_repospec = RepoSpec.RepoSpec.objects.
    get(repo="python3-code-executor-template")
# token == py3_code_exec_template_repospec.token
# secret_key == py3_code_exec_template_repospec.secret_key
token = py3_code_exec_template_repospec.token
print(token)
secret_key = py3_code_exec_template_repospec.secret_key
print(secret_key)
py3_code_exec_template_repospec = RepoSpec.update_repospec(
    repospec=py3_code_exec_template_repospec, parent=None,
    repo=py3_code_exec_template_repospec.repo,
    regen_secret_key=True)
# token != py3_code_exec_template_repospec.token
# secret_key != py3_code_exec_template_repospec.secret_key
token = py3_code_exec_template_repospec.token
print(token)
secret_key = py3_code_exec_template_repospec.secret_key
print(secret_key)
py3_code_exec_template_repospec = RepoSpec.update_repo(
    old_repo=py3_code_exec_template_repospec.token)
# token == py3_code_exec_template_repospec.token
# secret key == py3_code_exec_template_repospec.secret_key
py3_code_exec_template_repospec = RepoSpec.update_repo(
    old_repo=py3_code_exec_template_repospec.token,
    regen_secret_key=True)
# token != py3_code_exec_template_repospec.token
# secret_key != py3_code_exec_template_repospec.secret_key
```

```
from apiApp.Validation import RepoSpec
postgresql_code_exec_template_repospec = RepoSpec.create(repo
    ="postgresql-code-executor-template", parent=None)
token = postgresql_code_exec_template_repospec.token
print(token)
secret_key = postgresql_code_exec_template_repospec.
    secret_key
print(secret_key)
postgresql_code_exec_template_repospec = RepoSpec.RepoSpec.
    objects.get(repo="postgresql-code-executor-template")
# str(token, 'utf-8') ==
    postgresql_code_exec_template_repospec.token
# str(secret_key, 'utf-8') ==
    postgresql_code_exec_template_repospec.secret_key
# RepoSpec.validate_repospec(
    postgresql_code_exec_template_repospec.token) == True
# RepoSpec.decode_repospec(repospec=
    postgresql_code_exec_template_repospec.token,
#                             stored_repospec=
    postgresql_code_exec_template_repospec)
#                             == {'repo': "postgresql-code-
    executor-template"}
# RepoSpec.decode_repospec(repospec=
    postgresql_code_exec_template_repospec.token) == {'repo':
    "postgresql-executor-template"}
# RepoSpec.decode(stored_repospec=
    postgresql_code_exec_template_repospec, repospec=None,
    decode_stored=True)
#                     == {'repo': "postgresql-code-executor-
    template"}
```

G.4. Cliente Ejemplar de API de RepoSpecs

```
import requests

# Would normally be imported from settings, but for
    demonstration purposes:
#from EduNube.settings import EDUNUBE_CONFIG
EDUNUBE_CONFIG = {
    "protocol": "http",
    "host": "10.10.10.1",
    "port": 8010,
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9."
             "eyJhcHBfbmFtZSI6IkdpdEVEVSIsImV4cGlyZ"
             "XMiOmZhbHNlLCJjcmVhdGVkX2RhdGUiOiIyMD"
             "E3LTExLTEyIDE3OjMzOjE3LjY0MTY4MSIsImV"
             "kaXRfZGF0ZSI6IjIwMTctMTEtMTIgMjA6MzY6"
             "NDAuMjc5NDAyIn0.825oh2rZUlIPZFaP_UbYP"
             "DpdsXTE0XCaNsia-3NnGuc"
```

```
}


class EduNubeRepoSpecHTTPconsumer:

    create_operation = "create"
    get_operation = "get"
    get_or_create_operation = "get_or_create"
    edit_operation = "edit"

    url_template = "%s://%s:%d/api/repospec/%s"

    def build_url(self, protocol, host, port, operation):
        return self.url_template %(protocol, host, port,
           operation)

    def build_inicial_payload(self):
        return dict()

    def post_url(self, url, payload):
        return requests.post(url, data=payload)

    def validate_url(self, url):
        return url


class ConfigEduNubeRepoSpecHTTPconsumer(
   EduNubeRepoSpecHTTPconsumer):

    protocol = None
    host = None
    port = None
    object_type = None
    token = None

    def __init__(self, protocol=None, host=None, port=None,
       object_type=None, token=None):
        self.protocol = self.protocol if protocol is None
           else protocol
        self.host = self.host if host is None else host
        self.port = self.port if port is None else port
        self.object_type = self.object_type if object_type is
            None else object_type
        self.token = self.token if token is None else token

    def build_inicial_payload(self):
        return {
            'token': self.token
        }

    def build_config_url(self, operation):
```

```
        return self.build_url(protocol=self.protocol, host=
            self.host, port=self.port, operation=operation)


class DefaultConfigEduNubeRepoSpecHTTPconsumer(
    ConfigEduNubeRepoSpecHTTPconsumer):
    protocol = EDUNUBE_CONFIG.get('protocol')
    host = EDUNUBE_CONFIG.get('host')
    port = EDUNUBE_CONFIG.get('port')
    token = EDUNUBE_CONFIG.get('token')


class EduNubeRepoSpecConsumer(
    DefaultConfigEduNubeRepoSpecHTTPconsumer):

    def _validate_findable(self, repo=None, repospec_token=
        None):
        if repo is None and repospec_token is None:
            raise ValueError("repo␣and␣repospec_token␣can't␣
                both␣be␣None")

    def _make_call(self, operation, payload):
        url = self.build_config_url(operation=operation)
        return self.post_url(url=url, payload=payload)

    def _if_not_none_add_to_payload(self, payload,
        payload_index, value):
        if value is not None:
            payload[payload_index] = value
        return payload

    def create(self, repo, parent_repo=None):
        payload = self.build_inicial_payload()
        payload['repo'] = repo
        payload = self._if_not_none_add_to_payload(payload=
            payload, payload_index='parent', value=parent_repo
            )
        return self._make_call(operation=self.
            create_operation, payload=payload)

    def get(self, repo=None, repospec_token=None):
        self._validate_findable(repo=repo, repospec_token=
            repospec_token)
        payload = self.build_inicial_payload()
        payload = self._if_not_none_add_to_payload(payload=
            payload, payload_index='repo', value=repo)
        payload = self._if_not_none_add_to_payload(payload=
            payload, payload_index='repospec_token', value=
            repospec_token)
        return self._make_call(operation=self.get_operation,
            payload=payload)
```

```
    def get_or_create(self, repo, parent_repo=None,
        repospec_token=None):
        payload = self.build_inicial_payload()
        payload['repo'] = repo
        payload = self._if_not_none_add_to_payload(payload=
            payload, payload_index='parent', value=parent_repo
            )
        payload = self._if_not_none_add_to_payload(payload=
            payload, payload_index='repospec_token', value=
            repospec_token)
        return self._make_call(operation=self.
            get_or_create_operation, payload=payload)

    def edit(self, repo=None, repospec_token=None,
        parent_repo=None, new_repo=None, regen_secret_key=
        False):
        self._validate_findable(repo=repo, repospec_token=
            repospec_token)
        payload = self.build_inicial_payload()
        payload = self._if_not_none_add_to_payload(payload=
            payload, payload_index='repo', value=repo)
        payload = self._if_not_none_add_to_payload(payload=
            payload, payload_index='repospec_token', value=
            repospec_token)
        payload = self._if_not_none_add_to_payload(payload=
            payload, payload_index='parent', value=parent_repo
            )
        payload = self._if_not_none_add_to_payload(payload=
            payload, payload_index='new_repo', value=new_repo)
        if type(regen_secret_key) == bool:
            payload['regen_secret_key'] = regen_secret_key
        return self._make_call(operation=self.edit_operation,
            payload=payload)
```

G.5. Prueba inicial de API de Ejecuccion y Jobs

```
import json

from execute_status_example_client import
    EduNubePy3ExecuteStatusConsumer

en_py3_es_consumer = EduNubePy3ExecuteStatusConsumer()

namespace = 'nishedcob'
repository = 'test'

inicial_execution_create = en_py3_es_consumer.create(
    namespace=namespace, repository=repository)
```

```
print("Inicial Exec Create Code: %d" %
    inicial_execution_create.status_code)
print("Inicial Exec Create Reply Data: %s" %
    inicial_execution_create.text)
inicial_execution_create_reply_data = json.loads(
    inicial_execution_create.text)
print("Inicial Exec Create Reply Data (JSON Loads): %s" %
    inicial_execution_create_reply_data)

print()
print("
    ----------------------------------------------------------------------------
    ")
print()

inicial_execution_id = inicial_execution_create_reply_data.
    get('id')

inicial_execution_status = en_py3_es_consumer.status(id=
    inicial_execution_id)

print("Inicial Exec Status Code: %d" %
    inicial_execution_status.status_code)
print("Inicial Exec Status Reply Data: %s" %
    inicial_execution_status.text)
inicial_execution_status_reply_data = json.loads(
    inicial_execution_status.text)
print("Inicial Exec Status Reply Data (JSON Loads): %s" %
    inicial_execution_status_reply_data)

print()
print("
    ----------------------------------------------------------------------------
    ")
print()

inicial_execution_result = en_py3_es_consumer.result(id=
    inicial_execution_id)
print("Initial Exec Result Code: %d" %
    inicial_execution_result.status_code)
print("Inicial Exec Result Reply Data: %s" %
    inicial_execution_result.text)
inicial_execution_result_reply_data = json.loads(
    inicial_execution_result.text)
print("Inicial Exec Result Reply Data (JSON Loads): %s" %
    inicial_execution_result_reply_data)
```

G.6. Cliente de API de Ejecuccion y Jobs

```
import requests
```

```
import json

# Would normally be imported from settings, but for
    demonstration purposes:
#from EduNube.settings import EDUNUBE_CONFIG
EDUNUBE_CONFIG = {
    "protocol": "http",
    "host": "10.10.10.1",
    "port": 8010,
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9."
             "eyJhcHBfbmFtZSI6IkdpdEVEVSIsImV4cGlyZ"
             "XMiOmZhbHNlLCJjcmVhdGVkX2RhdGUiOiIyMD"
             "E3LTExLTEyIDE3OjMzOjE3LjY0MTY4MSIsImV"
             "kaXRfZGF0ZSI6IjIwMTctMTEtMTIgMjA6MzY6"
             "NDAuMjc5NDAyIn0.825oh2rZUlIPZFaP_UbYP"
             "DpdsXTE0XCaNsia-3NnGuc"
}


class EduNubeExecuteStatusHTTPConsumer:

    create_operation = 'create'
    status_operation = 'status'
    result_operation = 'result'

    url_template = "%s://%s:%d/api/execute/%s"

    def build_base_url(self, protocol, host, port, operation)
        :
        return self.url_template %(protocol, host, port,
            operation)

    def build_inicial_payload(self):
        return dict()

    def post_url(self, url, payload):
        return requests.post(url, data=payload)

    def validate_url(self, url):
        return url


class ConfigEduNubeExecuteStatusHTTPConsumer(
    EduNubeExecuteStatusHTTPConsumer):

    protocol = None
    host = None
    port = None
    object_type = None
    token = None
```

```
    def __init__(self, protocol=None, host=None, port=None,
       object_type=None, token=None):
        self.protocol = self.protocol if protocol is None
           else protocol
        self.host = self.host if host is None else host
        self.port = self.port if port is None else port
        self.object_type = self.object_type if object_type is
            None else object_type
        self.token = self.token if token is None else token

    def build_inicial_payload(self):
        return {
            'token': self.token
        }

    def build_base_config_url(self, operation):
        return self.build_base_url(protocol=self.protocol,
           host=self.host, port=self.port, operation=
           operation)


class DefaultConfigEduNubeExecuteStatusHTTPConsumer(
   ConfigEduNubeExecuteStatusHTTPConsumer):
    protocol = EDUNUBE_CONFIG.get('protocol')
    host = EDUNUBE_CONFIG.get('host')
    port = EDUNUBE_CONFIG.get('port')
    token = EDUNUBE_CONFIG.get('token')


class EduNubeExecuteStatusConsumer(
   DefaultConfigEduNubeExecuteStatusHTTPConsumer):

    def _prepare_repo_suffix(self, language, namespace,
       repository):
        return "%s/%s/%s/" %(language, namespace, repository
           )

    def _prepare_id_suffix(self, language, id):
        return "%s/%s/" %(language, id)

    def _make_call(self, operation, suffix, payload=None):
        if payload is None:
            payload = self.build_inicial_payload()
        url = "%s/%s" %(self.build_base_config_url(operation
           =operation), suffix)
        return self.post_url(url=url, payload=payload)

    def create(self, language, namespace, repository):
        return self._make_call(
            operation=self.create_operation,
```

```
            suffix=self._prepare_repo_suffix(language=
                language, namespace=namespace, repository=
                repository)
        )

    def status(self, language, id):
        return self._make_call(
            operation=self.status_operation,
            suffix=self._prepare_id_suffix(language=language,
                id=id)
        )

    def result(self, language, id):
        return self._make_call(
            operation=self.result_operation,
            suffix=self._prepare_id_suffix(language=language,
                id=id)
        )


class EduNubeLanguageExecuteStatusConsumer(
    EduNubeExecuteStatusConsumer):

    language = None

    def create(self, namespace, repository, language=None):
        if language is None:
            language = self.language
        return super().create(language=language, namespace=
            namespace, repository=repository)

    def status(self, id, language=None):
        if language is None:
            language = self.language
        return super().status(language=language, id=id)

    def result(self, id, language=None):
        if language is None:
            language = self.language
        return super().result(language=language, id=id)


class EduNubeShellExecuteStatusConsumer(
    EduNubeLanguageExecuteStatusConsumer):
    language = 'shell'


class EduNubePy3ExecuteStatusConsumer(
    EduNubeLanguageExecuteStatusConsumer):
    language = 'python3'
```

```
class EduNubePGSQLExecuteStatusConsumer(
    EduNubeLanguageExecuteStatusConsumer):
     language = 'postgresql'
```

Apéndice H

Pruebas de GitServerHTTPEndpoint

H.1. Cliente Ejemplar de API con pruebas integradas

```
# import config from somewhere, for example with Django I
   would use a settings file
#from GitEDU.settings import GIT_SERVER_HTTP_ENDPOINT_CONFIG

# In this case, I will just place an example config here,
   extracted from GitEDU:
GIT_SERVER_HTTP_ENDPOINT_CONFIG = {
    "protocol": "http",
    "host": "127.0.0.1",
    "port": 8020,
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9."
             "eyJleHBpcmVzIjpmYWxzZSwiYXBwX25hbWUiO"
             "iJHaXRFRFUiLCJlZGl0X2RhdGUiOiIyMDE3LT"
             "ExLTI1IDIwOjI2OjIzLjQ3MDgxNyswMDowMCI"
             "sImNyZWF0ZWRfZGF0ZSI6IjIwMTctMTEtMjUg"
             "MjA6MjY6MjMuNDcwNzUwKzAwOjAwIn0.jVHEm"
             "UAgJcQy7sU-qyULAnAiIrBAPNbeDjOiwjk5EE"
             "k"
}

import requests


class GitServerHTTPEndpointConsumer:

    create_operation = 'create'
    edit_operation = 'edit'
    edit_mv_operation = "%s/mv" % edit_operation
    edit_contents_operation = "%s/contents" % edit_operation

    url_template = "%s://%s:%d/api/%s/%s/%s"

    def build_url(self, protocol, host, port, object_type,
       operation, object_path):
```

```
        return self.url_template %(protocol, host, port,
            object_type, operation, object_path)

    def build_inicial_payload(self):
        return dict()

    def post_url(self, url, payload):
        return requests.post(url, data=payload)

    def validate_url(self, url):
        if not url.endswith("/"):
            url += "/"
        return url


class ConfigGitSrvHTTPEpConsumer(
    GitServerHTTPEndpointConsumer):

    protocol = None
    host = None
    port = None
    object_type = None
    token = None

    def __init__(self, protocol=None, host=None, port=None,
        object_type=None, token=None):
        self.protocol = self.protocol if protocol is None
            else protocol
        self.host = self.host if host is None else host
        self.port = self.port if port is None else port
        self.object_type = self.object_type if object_type is
            None else object_type
        self.token = self.token if token is None else token

    def build_inicial_payload(self):
        return {
            'token': self.token
        }

    def build_config_url(self, operation, object_path):
        return self.build_url(protocol=self.protocol, host=
            self.host, port=self.port, object_type=self.
            object_type, operation=operation, object_path=
            object_path)


class DefaultConfigGitSrvHTTPEpConsumer(
    ConfigGitSrvHTTPEpConsumer):
    protocol = GIT_SERVER_HTTP_ENDPOINT_CONFIG.get('protocol'
        )
    host = GIT_SERVER_HTTP_ENDPOINT_CONFIG.get('host')
```

```
    port = GIT_SERVER_HTTP_ENDPOINT_CONFIG.get('port')
    token = GIT_SERVER_HTTP_ENDPOINT_CONFIG.get('token')


class NamespaceGitSrvHTTPEpConsumer(
   DefaultConfigGitSrvHTTPEpConsumer):
    object_type = 'ns'

    def build_call_url(self, operation, namespace):
        url = self.build_config_url(operation=operation,
           object_path=namespace)
        if not url.endswith("/"):
            url += "/"
        return url

    def create_call(self, namespace):
        payload = self.build_inicial_payload()
        url = self.build_call_url(operation=self.
           create_operation, namespace=namespace)
        return self.post_url(url=url, payload=payload)

    def edit_call(self, namespace, new_namespace):
        payload = self.build_inicial_payload()
        payload['new_namespace'] = new_namespace
        url = self.build_call_url(operation=self.
           edit_operation, namespace=namespace)
        return self.post_url(url=url, payload=payload)


class RepositoryGitSrvHTTPEpConsumer(
   DefaultConfigGitSrvHTTPEpConsumer):
    object_type = 'repo'

    def build_object_path(self, namespace, repository):
        return "%s/%s" %(namespace, repository)

    def build_call_url(self, operation, namespace, repository
       ):
        object_path = self.build_object_path(namespace=
           namespace, repository=repository)
        url = self.build_config_url(operation=operation,
           object_path=object_path)
        if not url.endswith("/"):
            url += "/"
        return url

    def create_call(self, namespace, repository):
        payload = self.build_inicial_payload()
        url = self.build_call_url(operation=self.
           create_operation, namespace=namespace, repository=
           repository)
```

```
        return self.post_url(url=url, payload=payload)

    def edit_call(self, namespace, repository, new_repository
        , new_namespace=None):
        payload = self.build_inicial_payload()
        if new_namespace is not None:
            payload['new_namespace'] = new_namespace
        payload['new_repository'] = new_repository
        url = self.build_call_url(operation=self.
            edit_operation, namespace=namespace, repository=
            repository)
        return self.post_url(url=url, payload=payload)


class FileGitSrvHTTPEpConsumer(
    DefaultConfigGitSrvHTTPEpConsumer):
    object_type = 'file'

    def build_object_path(self, namespace, repository,
        file_path):
        return "%s/%s/%s" %(namespace, repository, file_path
            )

    def build_call_url(self, operation, namespace, repository
        , file_path):
        object_path = self.build_object_path(namespace=
            namespace, repository=repository, file_path=
            file_path)
        return self.build_config_url(operation=operation,
            object_path=object_path)

    def create_call(self, namespace, repository, file_path,
        commit=True):
        payload = self.build_inicial_payload()
        payload['commit'] = commit
        url = self.build_call_url(operation=self.
            create_operation, namespace=namespace, repository=
            repository, file_path=file_path)
        return self.post_url(url=url, payload=payload)

    def edit_mv_call(self, namespace, repository, file_path,
        new_file_path, new_repository=None, new_namespace=None
        ):
        payload = self.build_inicial_payload()
        if new_namespace is not None:
            payload['new_namespace'] = new_namespace
        if new_repository is not None:
            payload['new_repository'] = new_repository
        payload['new_file_path'] = new_file_path
```

```
        url = self.build_call_url(operation=self.
            edit_mv_operation, namespace=namespace, repository
            =repository, file_path=file_path)
        return self.post_url(url=url, payload=payload)

    def edit_contents_call(self, namespace, repository,
        file_path, contents):
        payload = self.build_inicial_payload()
        payload['contents'] = contents
        url = self.build_call_url(operation=self.
            edit_contents_operation, namespace=namespace,
            repository=repository, file_path=file_path)
        return self.post_url(url=url, payload=payload)

    def create_and_edit_contents_call(self, namespace,
        repository, file_path, contents):
        return [
            self.create_call(namespace=namespace, repository=
                repository, file_path=file_path, commit=False)
                ,
            self.edit_contents_call(namespace=namespace,
                repository=repository, file_path=file_path,
                contents=contents)
        ]


# to test: set test to True and run "python manage.py -c "
    import ideApp.git_server_http_endpoint"
test = False
if test:
    namespace = 'nishedcob3'
    repository = 'test2'
    file_path = 'folder/test2.py'

    # Example File Creation Call:
    file_consumer = FileGitSrvHTTPEpConsumer()
    file_create = file_consumer.create_call(namespace=
        namespace, repository=repository, file_path=file_path,
        commit=False)

    print('url: %s' % file_create.request.url)
    print('data: %s' % file_create.request.body)

    contents = "print('hello world')\n"

    # Example File Edit Call:
    file_edit = file_consumer.edit_contents_call(namespace=
        namespace, repository=repository, file_path=file_path,
        contents=contents)

    print('url: %s' % file_edit.request.url)
```

```
print('data: %s' % file_edit.request.body)

file_path = 'folder/test3.py'

file_create_and_edit = file_consumer.
    create_and_edit_contents_call(namespace=namespace,
    repository=repository, file_path=file_path, contents=
    contents)
print("FILE CREATE")
file_create = file_create_and_edit[0]
print('url: %s' % file_create.request.url)
print('data: %s' % file_create.request.body)

print("FILE EDIT")
file_edit = file_create_and_edit[1]
print('url: %s' % file_edit.request.url)
print('data: %s' % file_edit.request.body)
```

Apéndice I

Ubicacion de Repositorios de Codigo y Sus Licencias

I.1. Ubicacion de Codigo Fuente

Se puede encontrar el codigo fuente, divido en repositorios de este trabajo de titulacion en la siguiente direcciones con las licencias respectivas:

GitEDU y EduNube Servicios Principales para Editar y Ejecutar codigo en Linea.
Licencia: **GNU General Public License v3.0**.
Repositorios y Espejos:

GitLab `https://gitlab.com/nishedcob/GitEDU`

GitHub `https://gitlab.com/nishedcob/GitEDU`

Submodules y Directorios:

3rd_party/ Directorio con Librerias de Terceros ubicado de tal forma que GitHub no lo cuenta en los estadisticas del repositorio.

EduNube/ Directorio que forma la raiz del proyecto de Django que es el servicio EduNube.

GitEDU/ Directorio que forma la raiz del proyecto de Django que es el servicio GitEDU.

GitEDU-forked-libs Submodulo que agrupa librerias forkeadas.
Repositorios y Espejos:

GitLab `https://gitlab.com/nishedcob/Thesis-GE-Libraries`

GitHub `https://github.com/nishedcob/Thesis-GE-Libraries`

Submodulos:

django-app-lti Liberia Base para LTI en Django, forkeado y actualizado para funcionar con Python 3.
Licencia: **Harvard Copyright License**.
Repositorios y Espejos:

GitLab `https://gitlab.com/nishedcob/django-app-lti`

GitHub `https://github.com/nishedcob/django-app-lti`

django-auth-lti Backend de Autenticacion LTI en Django, forkeado y actualizado para funcionar con Python 3.
Licencia: **Apache 2.0**.
Repositorios y Espejos:

GitLab `https://gitlab.com/nishedcob/django-auth-lti`
GitHub `https://github.com/nishedcob/django-auth-lti`

ims_lti_py Libreria de LTI para Python, forkeado y actualizado para funcionar con Python 3.
Licencia: **Top Hat Monocle Corp. Copyright License**.
Repositorios y Espejos:
GitLab `https://gitlab.com/nishedcob/ims_lti_py`
GitHub `https://github.com/nishedcob/ims_lti_py`

python-oauth2 Libreria de Python para cumplir con los estandares de OAuth, forkeado y actualizado para funcionar con Python 3.
Licencia: **MIT**.
Repositorios y Espejos:
GitLab `https://gitlab.com/nishedcob/python-oauth2`
GitHub `https://github.com/nishedcob/python-oauth2`

GitServerHTTPEndpoint Submodule para el Servicio GitServerHTTPEndpoint.
Licencia: **GNU General Public License v3.0**.
Repositorios y Espejos:
GitLab `https://gitlab.com/nishedcob/GitServerHTTPEndpoint`
GitHub `https://github.com/nishedcob/GitServerHTTPEndpoint`

UTPL-Engineering-Thesis el codigo/proyecto de LaTeX para generar este documento.
Licencia: **Creative Commons Attribution Non Commercial Share Alike 4.0**.
Repositorios y Espejos:
GitLab `https://gitlab.com/nishedcob/UTPL-Engineering-Thesis`
GitHub `https://github.com/nishedcob/UTPL-Engineering-Thesis`

UTPL-GP4.2-Thesis-Defense los diapositivos (Reveal.js) utilizados para defender la tesis (para la GP 4.2).
Licencia: **MIT** de Reveal.js.
Repositorios y Espejos:
GitLab `https://gitlab.com/nishedcob/UTPL-GP4.2`
GitHub `https://github.com/nishedcob/UTPL-GP4.2`

db/ Directorio desactualizado con algunos archivos relacionados a la base de datos tanto relacional como no relacional.

docker/ Directorio con todos los archivos relacionados a la creacion de los seis imagenes de Docker utilizados en el proyecto.

kubernetes/ Directorio con los expirimentos realizados con Kubernetes. Tambien contiene los submodulos detallados a continuacion:

postgresql-code-executor-template Plantilla de Ejecuccion para uso con imagenes de Docker `postgresql-executor` dentro de su volumen `/code`.
Licencia: **GNU General Public License v3.0**.
Repositorios y Espejos:

GitLab `https://gitlab.com/nishedcob/postgresql-code-executor-template`

GitHub `https://github.com/nishedcob/postgresql-code-executor-template`

python3-code-executor-template Plantilla de Ejecuccion para uso con imagenes de Docker `python3-executor` dentro de su volumen `/code`.
Licencia: **GNU General Public License v3.0**.
Repositorios y Espejos:

GitLab `https://gitlab.com/nishedcob/python3-code-executor-template`

GitHub `https://github.com/nishedcob/python3-code-executor-template`

shell-code-executor-template Plantilla de Ejecuccion para uso con imagenes de Docker `shell-executor` dentro de su volumen `/code`.
Licencia: **GNU General Public License v3.0**.
Repositorios y Espejos:

GitLab `https://gitlab.com/nishedcob/shell-code-executor-template`

GitHub `https://github.com/nishedcob/shell-code-executor-template`

Se puede clonar todo el repositorio y submodules con el commando:

```
git clone --recursive https://gitlab.com/nishedcob/GitEDU.git
```

O en repositorios ya existentes, siempre se puede actualizar submodulos con el comando:

```
git submodule update --init --recursive --remote
```

Bibliografía

Almurayh, Abdullah y Sudhanshu Semwal (2014). «Xen web-based terminal for learning virtualization and cloud computing management». En: *Proc. World Congress on Engineering and Computer Science*. Vol. 1, págs. 329-333.

Amin, Nada (2017). *io.livecode.ch*. URL: http://io.livecode.ch/ (visitado 30-04-2017).

Baum, Ian y col. (2017). *GitLab Docker images*. URL: https://docs.gitlab.com/omnibus/docker/ (visitado 06-02-2018).

Beal, Vangie. *CSS - Cascading Style Sheets*. URL: http://www.webopedia.com/TERM/C/CSS.html (visitado 19-03-2017).

— *HTML - HyperText Markup Language*. URL: http://www.webopedia.com/TERM/H/HTML.html (visitado 19-03-2017).

— *HTTP - HyperText Transfer Protocol*. URL: http://www.webopedia.com/TERM/H/HTTP.html (visitado 19-03-2017).

— *JavaScript*. URL: http://www.webopedia.com/TERM/J/JavaScript.html (visitado 19-03-2017).

Berners-Lee, Tim, Roy Fielding y Larry Masinter (2005). *Uniform Resource Identifier (URI): Generic Syntax*. URL: https://tools.ietf.org/html/rfc3986 (visitado 19-03-2017).

Bravo, Marcelo, Nicholas Earley y Pricilla Vargas (2017). *GitEduERP*. Mirror: https://gitlab.com/ArqAppGrpBravoEarleyVargas/GitEduERP. URL: https://git.taw.utpl.edu.ec/ArqAppGrpBravoEarleyVargas/GitEduERP (visitado 30-03-2017).

Chacon, Scott y Ben Straub (2014a). *1.1 Getting Started - About Version Control*. URL: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control (visitado 24-02-2017).

— (2014b). *1.3 Getting Started - Git Basics*. URL: https://git-scm.com/book/en/v2/Getting-Started-Git-Basics (visitado 24-02-2017).

— (2014c). *7.4 Git Tools - Signing Your Work*. URL: https://git-scm.com/book/tr/v2/Git-Tools-Signing-Your-Work (visitado 24-02-2017).

Cisco. *Tunneling*. URL: http://www.cisco.com/c/en/us/products/ios-nx-os-software/tunneling/index.html (visitado 04-03-2017).

Cloud9 IDE, Inc. *Your development environment, in the cloud*. URL: https://c9.io/ (visitado 19-03-2017).

ComputerHope (2017). *HTTP*. URL: https://www.computerhope.com/jargon/h/http.htm (visitado 19-03-2017).

Cruise, Brit (2012a). *RSA encryption: Step 1*. URL: https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/v/intro-to-rsa-encryption (visitado 19-03-2017).

— (2012b). *What is cryptography?* URL: https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/intro-to-cryptography (visitado 19-03-2017).

Debian Community (2017). *Xen*. URL: https://wiki.debian.org/Xen (visitado 12-08-2017).

DigiCert Inc. *What is an SSL certificate?* URL: https://www.digicert.com/ssl/ (visitado 19-03-2017).
Docker Inc. (2017a). *Docker*. URL: https://www.docker.com/ (visitado 30-04-2017).
— (2017b). *What is a Container*. URL: https://www.docker.com/what-container (visitado 30-04-2017).
— (2017c). *What is Docker*. URL: https://www.docker.com/what-docker (visitado 30-04-2017).
Dougiamas, Martin y Free Software Foundation (2016). *GNU GENERAL PUBLIC LICENSE*. URL: https://git.moodle.org/gw?p=moodle.git;a=blob;f=COPYING.txt;h=94a9ed024d3859793618152ea559a168bbcbb5e2;hb=HEAD (visitado 17-02-2017).
edX. *About Open edX*. URL: https://open.edx.org/about-open-edx (visitado 25-04-2017).
Ellingwood, Justin (2015). *The Docker Ecosystem: An Overview of Containerization*. URL: https://www.digitalocean.com/community/tutorials/the-docker-ecosystem-an-overview-of-containerization (visitado 30-04-2017).
Flomenberg, Jake. *The Next Wave in Software: Open Adoption Software (OAS)*. URL: https://www.accel.com/interests/TheNextWaveInSoftwareOAS (visitado 21-02-2017).
Free Software Foundation (2007). *GNU General Public License*. URL: https://www.gnu.org/licenses/gpl.html (visitado 24-02-2017).
— (2016). *Licenses*. URL: https://www.gnu.org/licenses/licenses.en.html (visitado 24-02-2017).
Freyd, Régis y col. (2017). *GitLab Products*. URL: https://about.gitlab.com/gitlab-com/ (visitado 04-03-2017).
Geerling, Jeff (2014). *A brief history of SSH and remote access*. URL: https://www.jeffgeerling.com/blog/brief-history-ssh-and-remote-access (visitado 04-03-2017).
GitLab Inc. (2017). *Host GitLab on GitHost.io*. URL: https://githost.io/ (visitado 04-03-2017).
GlobalSign. *What is an SSL Certificate?* URL: https://www.globalsign.com/en/ssl-information-center/what-is-an-ssl-certificate/ (visitado 15-03-2017).
— (2016). *SSL vs. TLS - What's the Difference?* URL: https://www.globalsign.com/en/blog/ssl-vs-tls-difference/ (visitado 19-03-2017).
Google. *Explore the Storage Features of Drive*. URL: https://www.google.com/drive/using-drive/ (visitado 23-03-2017).
Hughes, Phil y col. (2017). *The platform for modern developers*. URL: https://about.gitlab.com/ (visitado 23-03-2017).
IMSGlobal. *Contributing Members, Affiliates, and Alliance Participants*. URL: https://www.imsglobal.org/membersandaffiliates.html (visitado 16-02-2017).
— *Learning Tools Interoperability® Background*. URL: https://www.imsglobal.org/activity/learning-tools-interoperability (visitado 16-02-2017).
Kangas, Erik (2016). *SSL versus TLS – What's the difference?* URL: https://luxsci.com/blog/ssl-versus-tls-whats-the-difference.html (visitado 19-03-2017).
Kaspersky Labs USA. *What is a Tunneling Protocol?* URL: https://usa.kaspersky.com/internet-security-center/definitions/tunneling-protocol (visitado 15-03-2017).

Lees-Miller, John (2015). *New: Collaborate Online and Offline with Overleaf and Git (beta)*. URL: https://www.overleaf.com/blog/195-new-collaborate-online-and-offline-with-overleaf-and-git-beta (visitado 23-03-2017).

López, Jorge (2017). Conversaciones en el curso de 2017.

McKendrick, Russ y Scott Gallagher (2017). *Mastering Docker - Second Edition: Master this widely used containerization tool*. 2nd Revised edition. Packt Publishing - ebooks Account. ISBN: 9781787280243. URL: http://amazon.com/o/ASIN/1787280241/.

Mindflash. *Mindflash Learning Management*. URL: https://www.mindflash.com/about/ (visitado 17-02-2017).

— *What is an LMS?* URL: https://www.mindflash.com/learning-management-systems/what-is-an-lms/ (visitado 17-02-2017).

Moodle Community (2014). *Installing Moodle on Debian based distributions*. URL: https://docs.moodle.org/33/en/Installing_Moodle_on_Debian_based_distributions (visitado 12-08-2017).

— (2016). *Features*. URL: https://docs.moodle.org/32/en/Features (visitado 17-02-2017).

Nierop, Ernst van y col. (2017). *Host your projects on GitLab.com*. URL: https://about.gitlab.com/gitlab-com/ (visitado 04-03-2017).

OpenBSD. *OpenSSH Features*. URL: http://www.openbsd.org/openssh/features.html (visitado 15-03-2017).

— (2016). *SSH(1)*. URL: http://man.openbsd.org/OpenBSD-current/man1/ssh.1 (visitado 15-03-2017).

OpenCampus. *For Universities*. URL: https://www.opencampus.net/home/univerities (visitado 21-02-2017).

— *OpenCampus*. URL: https://www.opencampus.net/home/ (visitado 21-02-2017).

— *Technology*. URL: https://www.opencampus.net/home/technology (visitado 21-02-2017).

opensource.com. *What is OpenStack?* URL: https://opensource.com/resources/what-is-openstack (visitado 15-01-2018).

OpenSSH. *OpenSSH*. URL: https://www.openssh.com/ (visitado 15-03-2017).

OpenStack Community. *Nova Docker*. URL: https://wiki.openstack.org/wiki/Docker (visitado 15-01-2018).

— (2018). *Nova Feature Support Matrix*. URL: https://docs.openstack.org/nova/latest/user/support-matrix.html (visitado 15-01-2018).

Oracle. *VirtualBox*. URL: https://www.virtualbox.org/ (visitado 30-04-2017).

Overleaf y Writelatex Limited. *Collaborative Writing and Publishing*. URL: https://www.overleaf.com/ (visitado 23-03-2017).

Pipinellis, Achilleas y col. (2017). *Next-generation developer collaboration software*. URL: https://about.gitlab.com/features/ (visitado 23-03-2017).

Project, QEMU. *QEMU*. URL: http://www.qemu.org/ (visitado 25-04-2017).

Raymond, Eric (1999). «The cathedral and the bazaar». En: *Philosophy & Technology* 12.3, pág. 23.

Repl.it y Neoreason, Inc. (2017). *repl.it is a cloud coding environment for...* URL: https://repl.it/ (visitado 19-03-2010).

Rose, Scott (2015). *Distributed or Centralized Development?* URL: http://blogs.collab.net/uncategorized/distributed-or-centralized-development (visitado 24-02-2017).

Salinas, Diana Maritza Ortega y Juan Carlos Sánchez Castillo (2007). *Sistema para la estructuración lógica de encuestas online para el centro de asesoría empresarial y social*

(cades);y call center de la UTPL. URL: http://dspace.utpl.edu.ec/handle/123456789/14471 (visitado 30-03-2017).

Schluting, Charlie (2006). *Networking 101: Understanding Tunneling*. URL: http://www.enterprisenetworkingplanet.com/netsp/article.php/3624566/Networking-101-Understanding-Tunneling.htm (visitado 15-03-2017).

Sijbrandij, Sid y col. (2017). *About Us*. URL: https://about.gitlab.com/about/ (visitado 04-03-2017).

Smith, Brett (2014). *A Quick Guide to GPLv3*. URL: https://www.gnu.org/licenses/quick-guide-gplv3.html (visitado 24-02-2017).

SSL.com Corp (2016). *FAQ: What is SSL?* URL: http://info.ssl.com/article.aspx?id=10241 (visitado 19-03-2017).

Stallman, Richard Mathew (2014). *About the GNU Operating System*. URL: https://www.gnu.org/gnu/about-gnu.html (visitado 21-02-2017).

— (2016a). *FLOSS and FOSS*. URL: https://www.gnu.org/philosophy/floss-and-foss.en.html (visitado 21-02-2017).

— (2016b). *What is free software?* URL: https://www.gnu.org/philosophy/free-sw.html (visitado 21-02-2017).

— (2016c). *Why Open Source misses the point of Free Software*. URL: https://www.gnu.org/philosophy/open-source-misses-the-point.html (visitado 21-02-2017).

— (2017a). *Free Software Is Even More Important Now*. URL: https://www.gnu.org/philosophy/free-software-even-more-important.html (visitado 21-02-2017).

— (2017b). *Linux and the GNU System*. URL: https://www.gnu.org/gnu/linux-and-gnu.html (visitado 23-02-2017).

Stevens, Marc y col. (2017). *The first collision for full SHA-1*. URL: https://shattered.io/static/shattered.pdf (visitado 24-02-2017).

Teimouri, Davoud (2016). *Operating-system-level virtualization*. URL: https://www.teimouri.net/operating-system-level-virtualization/ (visitado 30-04-2017).

The Kubernetes Authors (2018a). *Cluster Networking*. URL: https://kubernetes.io/docs/concepts/cluster-administration/networking/ (visitado 06-02-2018).

— (2018b). *Installing kubeadm*. URL: https://kubernetes.io/docs/setup/independent/install-kubeadm/ (visitado 06-02-2018).

— (2018c). *Kubernetes: Production-Grade Container Orchestration*. URL: https://kubernetes.io/ (visitado 15-01-2018).

— (2018d). *Using kubeadm to Create a Cluster*. URL: https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/ (visitado 06-02-2018).

Tholeti, Bhanu P (2011). *Learn about hypervisors, system virtualization, and how it works in a cloud environment*. URL: https://www.ibm.com/developerworks/cloud/library/cl-hypervisorcompare/ (visitado 25-04-2017).

Torres, Mauricio Javier Castillo (2008). *Metodología para implementación de cursos virtuales con herramientas Web 2.0*. URL: http://dspace.utpl.edu.ec/handle/123456789/17020 (visitado 30-03-2017).

University of East Anglia. *What is a MOOC? Massive Open Online Courses Explained*. URL: https://www.uea.ac.uk/study/short-courses/online-learning/what-is-a-mooc (visitado 25-04-2017).

University System of Georgia. *About Learning Repository*. URL: https://go.view.usg.edu/shared/Documentation/9.4.1/Instructor/9.4.1%

20Instructor%20Course%20Designer%20Help/learningrepository/about_learning_repository.htm (visitado 17-02-2017).

UTPL (2017a). *DATOS ESTADÍSTICOS*. URL: http://www.utpl.edu.ec/utpl/informacion-general/datos-estadisticos (visitado 05-04-2017).

— (2017b). *DEPARTAMENTO DE CIENCIAS DE LA COMPUTACION Y ELECTRONICA*. URL: http://www.utpl.edu.ec/directorio/index.php?ban=2&id_citte=1&id_dep=118&nom_dep=DEPARTAMENTO%20DE%20CIENCIAS%20DE%20LA%20COMPUTACION%20Y%20ELECTRONICA (visitado 05-04-2017).

— (2017c). *OpenCampus Acerca De*. URL: http://opencampus.utpl.edu.ec/about (visitado 21-02-2017).

— (2017d). *UTPL Tecnologías Avanzadas de la Web GitLab Community Edition*. URL: https://git.taw.utpl.edu.ec/ (visitado 04-03-2017).

Vanderbilt University. *Course Management Systems*. URL: https://cft.vanderbilt.edu/guides-sub-pages/course-management-systems/ (visitado 25-04-2017).

Venugopal, Suchitra (2016). *Cloud orchestration technologies: Explore your options*. URL: https://www.ibm.com/developerworks/cloud/library/cl-cloud-orchestration-technologies-trs/index.html (visitado 15-01-2018).

VMWare, Inc. *What is Virtualization?* URL: http://www.vmware.com/solutions/virtualization.html (visitado 25-04-2017).

— (2014). *What is Virtualization?* URL: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/whitepaper/techpaper/vmw-white-paper-secrty-vsphr-hyprvsr-uslet-101.pdf (visitado 25-04-2017).

Webopedia, ITBusinessEdge y QuinStreet Inc. *REST - Representational State Transfer*. URL: http://www.webopedia.com/TERM/R/REST.html (visitado 19-03-2017).

Webopedia Staff. *URI - Uniform Resource Identifier*. URL: http://www.webopedia.com/TERM/U/URI.html (visitado 19-03-2017).

Wilson, George, Michael Day y Beth Taylor (2011). *KVM: Hypervisor Security You Can Depend On*. URL: ftp://public.dhe.ibm.com/linux/pdfs/LXW03004-USEN-00.pdf (visitado 25-04-2017).

Xen Community (2016). *Xen Project Software Overview*. URL: https://wiki.xen.org/wiki/Xen_Project_Software_Overview (visitado 30-04-2017).

Índice alfabético