

UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERA EN SISTEMAS,
ELECTRONICA E INDUSTRIAL

CARRERA DE SOFTWARE

MANEJO Y CONFIGURACIÓN DE SOFTWARE

**Herramientas para la automatización de la
construcción del sistema**

Grupo # 7

INTEGRANTES:

- NUÑEZ CHRISTIAN
- SÁNCHEZ EZEQUIEL
- TOAPANTA DIEGO
- VILLACRÉS SILVIA

NIVEL: IV SW “A”

DOCENTE: Ing. Leonardo Torres

FECHA: 03/07/2021

I. Tema: Herramientas para la automatización de la construcción del sistema

II. Introducción

La CI/CD es un método para distribuir aplicaciones a los clientes con frecuencia mediante el uso de la automatización en las etapas del desarrollo de aplicaciones. Los principales conceptos que se atribuyen a la CI/CD son la integración continua, la distribución continua y la implementación continua. La CI/CD es una solución para los problemas que puede generar la integración del código nuevo a los equipos de desarrollo y de operaciones.

La sigla CI significa la integración continua, que es un proceso de automatización para los desarrolladores. Si la CI tiene éxito, los cambios del código nuevo en una aplicación se diseñan, se prueban y se combinan periódicamente en un repositorio compartido. Esto soluciona el problema de que se desarrollen demasiadas divisiones de una aplicación al mismo tiempo, porque podrían entrar en conflicto entre sí.

La CD se refiere a la distribución o la implementación continuas, los cuales son conceptos relacionados que suelen usarse indistintamente. Ambos conceptos se refieren a la automatización de las etapas posteriores del canal, pero a veces se usan por separado para explicar la cantidad de automatización que se está incorporando[1].

A continuación, se detallarán algunas herramientas que pueden ayudar a la automatización en cuanto a la construcción del software.

III. Desarrollo

GITLAB CI (CONTINUOS INTEGRATION)

1. Definición

GitLab CI / CD es una herramienta integrada en GitLab para el desarrollo de software a través de metodologías continuas: Integración continua (CI), Entrega continua (CD) y Despliegue continuo (CD)

GitLab ofrece un programa propio de integración continua que funciona con la conocida herramienta de control de versiones. Los pipelines pueden configurarse y adaptarse así a los requisitos de cada proyecto. Además, es compatible con GitLab CI Docker. Gitlab es un servicio web de control de versiones y desarrollo de software colaborativo basado en Git. Además de gestor de repositorios, el servicio ofrece también alojamiento de wikis y un sistema de seguimiento de errores, todo ello publicado bajo una Licencia de código abierto. Permite gestionar, administrar, crear y conectar los repositorios con diferentes aplicaciones y hacer todo tipo de integraciones con ellas, ofreciendo un ambiente y una plataforma en cual se puede realizar las varias etapas de su SDLC/ADLC y DevOps[2].

Está integrado en Gitlab SCM. Puede crear tuberías utilizando archivos gitlab-ci.yml y manipularlos a través de una interfaz gráfica. Por lo tanto todos los miembros de los equipos (incluidos los no técnicos) tienen acceso rápido y fácil al estado del ciclo de vida de las aplicaciones[3].

2. Historia

Gitlab nace en el 13 de octubre de 2011, actualmente pertenece a la empresa GITLAB inc. Fue escrito por los programadores ucranianos Dmitriy Zaporozhets y Valery Sizov en el lenguaje de programación Ruby1 con algunas partes reescritas posteriormente en Go, inicialmente como una solución de gestión de código fuente para colaborar con su equipo en el desarrollo de software. Luego evolucionó a una solución integrada que cubre el ciclo de vida del desarrollo de software, y luego a todo el ciclo de vida de DevOps. La arquitectura tecnológica actual incluye Go, Ruby on Rails y Vue.js[4].

3. Ventajas

- La más importante sería que se puede migrar desde Jenkins y Cicle Ci
- Es OpenSource
- Rápido, Seguro, económico
- No depende de otras aplicaciones para su funcionamiento
- Integración con Docker sin necesidad de configurar nada, incluyendo un Docker registry privado por proyecto.
- Es rápida su configuración del servidor de Integración Continua.
- No dependes de contenedores o servicios externos y todo lo que gestionas está realmente instalado en la máquina.
- Mayor curva de configuración inicial de tu servidor. La ventaja de tener archivos yml más simples se ve descompensada porque tienes que instalar todo lo necesario. Por tanto, solo compensa si el servidor va a alojar proyectos con tecnologías similares[5].

4. Desventajas

- No está escrito en español para su configuración, pero tiene mucha documentación para su aprendizaje

5. Características

- Está basado en un modelo de fichero de configuración escrito en YAML y almacenado en la raíz de cada repositorio.
- Tiene relación entre los sistemas tanto Integración Continua como la gestión de versiones.
- Prueba de rendimiento del navegador: Determina rápidamente el impacto en el rendimiento del navegador de los cambios de código pendientes.
- Prueba de rendimiento de carga: Determina rápidamente el impacto en el rendimiento del servidor de los cambios de código pendientes[6].

- Calidad del código: Analiza la calidad de tu código fuente.
- Informes de prueba unitaria: Identifica las fallas de la secuencia de comandos directamente en las solicitudes de combinación.
- Usar imágenes de Docker: Utiliza GitLab y GitLab Runner con Docker para crear y probar aplicaciones.
- Implementar tableros: Verifica el estado actual y el estado de cada entorno de CI / CD que se ejecuta en Kubernetes.
- Escaneo de dependencias: Analiza sus dependencias en busca de vulnerabilidades conocidas.
- Informes de prueba de seguridad: Verifica las vulnerabilidades de la aplicación[4].

6. Flujo de trabajo de GitLab CI / CD

Con la implementación de un código y trabajando localmente en los cambios propuestos. Se puede enviar sus confirmaciones a una rama de funciones en un repositorio remoto alojado en GitLab. El empuje activa la canalización de CI / CD para su proyecto. Luego, GitLab CI / CD:

1. Ejecuta scripts automatizados (secuencialmente o en paralelo) para:
2. Cree y pruebe su aplicación.
3. Obtenga una vista previa de los cambios en una aplicación de revisión, lo mismo que vería en su localhost.
4. Después de que la implementación funcione como se esperaba:
5. Obtenga su código revisado y aprobado.
6. Fusionar la rama de características en la rama predeterminada.
7. GitLab CI / CD implementa sus cambios automáticamente en un entorno de producción.
8. Si algo sale mal, puede revertir los cambios[7].

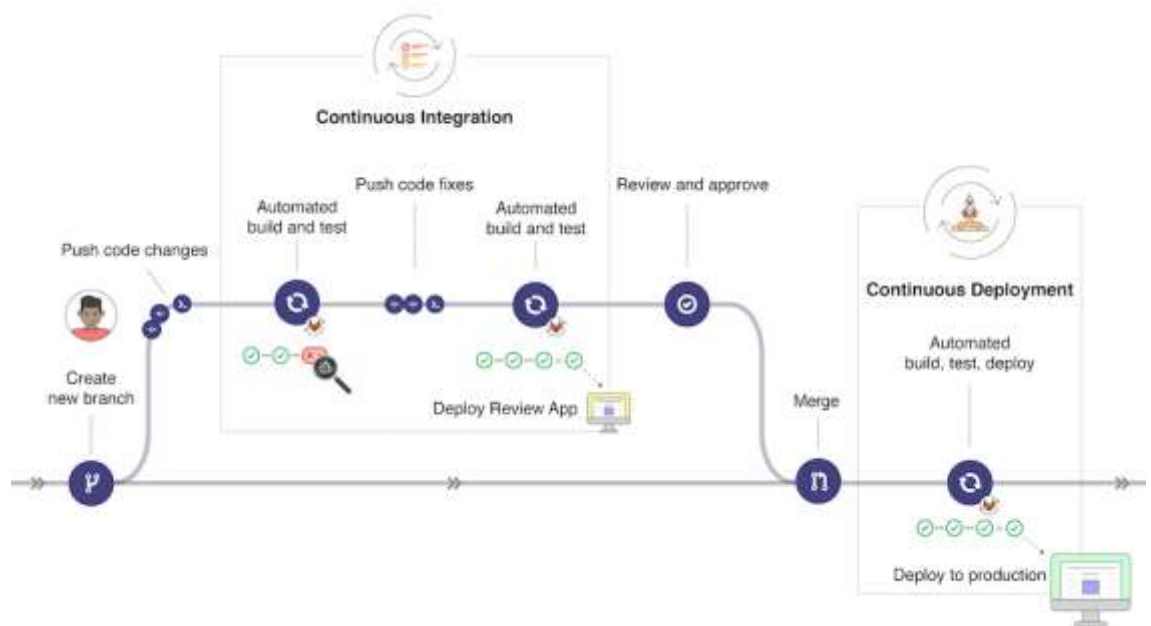


Ilustración 1: Flujo de trabajo de Gitlab CI

7. Implementación

- Cloud, SaaS, Web
- Windows (local)
- Linux (local)
- iPhone (móvil)
- iPad (móvil)

8. Costos

Inicialmente el producto se publicó como un software completamente libre bajo la licencia MIT. Sin embargo, tras la división del proyecto en julio de 2013 en dos versiones distintas, GitLab CE (Community Edition) y GitLab EE (Enterprise Edition), finalmente en febrero de 2014 GitLab EE se comenzó a desarrollar bajo una licencia privativa, con características que no están presentes en la versión libre.

GitLab CI es muy económico de \$0 a \$ 40 / mes puede ejecutar muchos trabajos simultáneos sin problemas[5].

JENKINS

1. Definición



Jenkins es un servidor automatizado de integración continua de código abierto totalmente gratuito capaz de organizar una cadena de acciones que ayudan a lograr el proceso de integración continua de manera automatizada.

Jenkins completamente escrito en Java y es una aplicación conocida y reconocida por DevOps de todo el mundo, más de 300.000 instalaciones y + 15.500 estrellas en Github lo respaldan. La razón por la que Jenkins se hizo tan popular es porque se encarga de supervisar las tareas repetitivas que surgen dentro del desarrollo de un proyecto. Al usar Jenkins, las compañías de software pueden acelerar su proceso de desarrollo del código; ya que Jenkins puede automatizar, agilizar y aumentar el ritmo de toda la compilación y las pruebas de los proyectos. Además, Jenkins puede ser implementado a lo largo de todo el ciclo de vida completo del desarrollo. Desde la fase construcción inicial, la fase de pruebas, en la documentación del software, en su implementación y en todas las demás etapas existentes dentro del ciclo de vida que desees aplicar.

2. Historia de Jenkins

Kohsuke Kawaguchi, desarrollador de Java que trabajaba en SUN Microsystems, estaba cansado de construir el código y corregir errores repetidamente. En 2004, creó un servidor de automatización llamado Hudson que automatizaba la tarea de compilación y el trabajo de la realización de pruebas. En 2011, Oracle, propietario de Sun Microsystems, tuvo una disputa con la comunidad de código abierto de Hudson y decidió bifurca el proyecto y renombrarlo como Jenkins. Tanto Hudson como Jenkins continuaron operando de manera independiente. Pero en poco tiempo, Jenkins fue siendo utilizado en muchos más proyectos y junto a ellos fueron apareciendo muchos más nuevos contribuyentes al código fuente. Hudson, en la actualidad, se encuentra descontinuado; puesto que, con el paso del tiempo, Jenkins se hizo más y más popular[8].

3. Ventajas de usar Jenkins

- Jenkins admite arquitectura basada en la nube para que pueda implementar Jenkins en plataformas basadas en la nube.
- El código se crea y prueba tan pronto como el desarrollador lo confirma. Jenkins compilará y probará el código muchas veces durante el día. Si la compilación es exitosa, Jenkins desplegará la fuente en el servidor de prueba y notificará al equipo de implementación. Si la compilación falla, Jenkins notificará los errores al equipo de desarrolladores.
- El código se crea inmediatamente después de que cualquiera de los desarrolladores actualiza.
- Dado que el código se construye después de cada actualización de un solo desarrollador, es fácil detectar qué código causó los fallos a la hora de la compilación.
- El proceso automatizado de construcción y pruebas ahorra mucho tiempo y reduce muchos posibles defectos.
- El código se implementa después de cada compilación y prueba exitosas.

- El ciclo de desarrollo es más rápido. Las nuevas funciones están disponibles antes, para los usuarios por lo que se aumentan las ganancias[9].

4. Herramientas de integración continua

- API
- Autenticación
- Control de versiones
- Controles o permisos de acceso
- Entrega continua
- Gestión de aplicaciones
- Gestión de pruebas de software
- Gestión de tests
- Gestión del pipeline
- Implementación continua
- Panel de actividades
- Proyecciones
- Supervisión

5. Asistencia

- E-mail/Help Desk
- Preguntas frecuentes/foro
- Base de conocimientos
- Asistencia telefónica

6. Implementación

- Cloud, SaaS, Web
- Mac (desktop)
- Windows (desktop)
- Linux (desktop)

CIRCLE CI

1. Definición

CircleCI es una de las plataformas de integración y entrega continuas (CI / CD) más grandes del mundo y un centro central para pasar el código de la idea a la función. CircleCI es una de las herramientas de DevOps más utilizadas, capaz de manejar más de 1 millón de desarrollos por día, lo que les brinda acceso único a los datos sobre las capacidades de su equipo de diseño y la ejecución de código.

Empresas como Spotify, Coinbase, Stitch Fix y BuzzFeed están utilizando esta solución para aumentar la productividad de sus equipos de ingeniería, llevar mejores productos al mercado y llevarlos al mercado más rápido.

1. Historia Circle CI

Fundada en 2011, con sede en San Francisco y empleados remotos en todo el mundo, CircleCI se encuentra entre Scale Venture Partners, Threshold Ventures (anteriormente DFJ), Baseline Ventures, Top Tier Capital, Industry Ventures y Heavybit. Está respaldado por una empresa conjunta de sociedades de capital, Harrison Metal Capital, Owl Rock Capital Partners, Next Equity Partners.

2. Características Circle CI

- Nos permite seleccionar Build Environment.
- Admite múltiples lenguajes como Linux, incluidos C, Javascript, NET, PHP, Python y Ruby.
- Con la compatibilidad con Docker, puede configurar su propio entorno personalizado.
- Cuando se activa una nueva compilación, automáticamente elimina la compilación en cola o en ejecución.
- Divide la prueba en varios contenedores para equilibrar y reducir el tiempo total de construcción.
- Evita que las personas que no sean administradores cambien la configuración importante del proyecto.
- Enviar aplicaciones sin errores para mejorar las calificaciones de su tienda de Android e iOS.
- Almacenamiento en caché optimizado y paralelismo para un rendimiento rápido.
- Integración con herramientas VCS.

3. Herramientas de automatización

- Api
- Admite la ejecución
- Análisis estático
- Conformidad Unicode
- Garantía de calidad
- Gestión de la conformidad
- Gestión de requisitos
- Herramientas de colaboración
- Mover y copiar
- Pruebas con palabras clave
- Tests basados en los requisitos
- Tests basados en modelos
- Tests de seguridad
- Tests parametrizados

- Verificación de scripts de test
- Vista jerárquica

4. Asistencia

- E-mail/Help Desk
- Preguntas frecuentes/foro
- Base de conocimientos
- Asistencia telefónica
- Asistencia 24/7
- Chat

5. Implementación

- Cloud, SaaS, Web
- Mac (desktop)
- Windows (desktop)
- Linux (desktop)
- Windows (local)
- Linux (local)
- Android (móvil)
- iPhone (móvil)
- iPad (móvil)

6. Precio Circle CI

- Versión gratuita con funciones limitadas
- Versión de suscripción por 30,00 US\$/mes
- Prueba gratis por un mes de la versión de suscripción.

BAMBOO



1. Definición

Es una herramienta de integración, implementación y entrega continua, que además permite la gestión de versiones, despliegue, la ejecución de pruebas automáticas, el cual al ser parte de los productos Atlassian, se integra con otras plataformas, opera en plataforma web y su licencia es de propietario basado en los siguientes objetivos:

- Integración continua de código.
- Entrega continua de artefactos.
- Gestión de releases.
- Despliegue continuo de releases.
- Trazabilidad con JIRA y Stash.
- Adaptable a mis propios procesos.

2. Historia de Bamboo.

Desarrollado por el equipo de desarrollo ATLASSIAN, que al principio disponía de versiones para servidor y la nube, sin embargo, esta última se dejó de distribuir a partir del 2016.

3. Características y funciones.

- Pipeline Flexible. - Permite adaptar el proceso a las necesidades particularidades, creando el concepto de stage y job. Esta flexibilidad permite la paralelización y serialización de trabajos según necesidades con la posibilidad de añadir pasos manuales y dependencia entre planes, triggers según el resultado de otras construcciones.
- Auto Branching. - Permite gestionar automáticamente los planes de construcción cuando se crea nuevas ramas en el repositorio, disponiendo de un plan de construcción por rama, manteniendo el 100% del código bajo control.
- Integración continua. – Soluciona problemas por la generación de versiones en diferentes ramas, para ello mantiene una integración en base a dos modelos el primero es Branch Updater en el cual mantiene la rama de funcionalidad actualizada en función de la rama base, y el segundo es Gatekeeper en el que mantiene actualizada la rama base, esto genera una estrategia optimista.
- Gestión del ciclo de vida. – Capacidad de entender los conceptos de construcción, artefacto, release, entorno y despliegue, relacionarlos y gestionarlos, a fin de gestionarlos en múltiples entornos. Obtiene además una trazabilidad continua entre los tres niveles: construcción entrega y despliegue.
- Integración con la Suite. – Como parte de los productos desarrollados por Atlassian, es posible integrar esta herramienta con Jira, Stash, confluence. Adicionalmente permite la integración con amazon web services y con Docker, nodejs entre otros.
- Extensibilidad. - Se extiende utilizando scripts, con add-ons del Marketplace de atlassian, o mediante desarrollo de add-ons propios.
- Otros. – Además de lo mencionado dispone de: gestión de tiempos y bloqueos, escalabilidad y caducidad, potencia de customización, esquemas de seguridad flexibles, Notificaciones a email, Jabber, HipChat, Interfaz sencilla y amigable[10].

4. Especificaciones.

- Plataforma en contenedor web.
- Licencia propietaria.
- Constructores de Windows (MSBuild, Nant, Visual Studio).
- Constructores Java (Ant, Maven).
- Otros constructores (Scripts personalizados, Herramientas en línea de comandos)
- Integraciones (IntelliJ IDEA, Eclipse, FishEye, Jira, Bitbucket, Github)

5. Ventajas

- Flujos de trabajo de ramificaciones de git integrados.
- Proyectos de implementación incorporados.
- Integración con JIRA
- Integración con GITBUCKET Server.
- API REST
- Automatización de pruebas
- Permisos sencillos a nivel empresarial

6. Desventajas.

- No permite incorporar bifurcaciones
- Su costo es elevado.

7. Costos

Existe dos modelos de distribución: para equipos pequeños y para equipos en crecimiento, los cuales disponen de una versión de prueba que dura 30 días.

Para la versión de equipos pequeños el costo es de 10 dólares, esta versión no cuenta con agentes remotos.

Para la versión de equipos en crecimiento el costo es de 1500 dólares como base y este se incrementa por los agentes remotos requeridos[11].

8. Interfaz.

The screenshot shows the Bamboo web interface. At the top, there's a navigation bar with 'My Bamboo', 'Build', 'Deploy', and 'Reports'. Below that, the project name 'C DB PostgreSQL 9.2' is displayed. The 'Tests' tab is selected, showing a 'Tests summary' section. A dropdown menu indicates 'Showing All builds'. Below this, a table lists the 'Top 10 most broken tests'. The table has four columns: 'Test', 'Times broken', 'Job', and 'Most recent builds'. All four tests listed have a 'Times broken' value of 18 and are associated with the 'Func Branches + MultiRepo' job. The tests are related to overriding build strategies and manual branch creation.

Test	Times broken	Job	Most recent builds
BranchesTest testOverridingBuildStrategyWorks on testOverridingBuildStrategy (com.acceptance.tests.branches.Bran	18	Func Branches + MultiRepo	#86, #75, #73 (15 more...)
BranchesTest testUserCanEnforceManualTriggerOnNewBranches on testUse Builds)(com.acceptance.tests.branches.Bran	18	Func Branches + MultiRepo	#86, #75, #73 (15 more...)
BranchesTest testManualCreationOfBranch on testManualCreationOfBranch - (com.acceptance.tests.branches.Bran	18	Func Branches + MultiRepo	#86, #75, #73 (15 more...)
BranchesTest testNoRepositoryDefinedWontThrowSomeExceptions on testNo Builds)(com.acceptance.tests.branches.Bran	18	Func Branches + MultiRepo	#86, #75, #73 (15 more...)

Ilustración 2: Interfaz de la herramienta Bamboo

My BambooBuildDeployReports

Bamboo administration


Atlassian Marketplace for Bamboo

Search the Marketplace

Staff-picked

All categories


All paid & free



TFS Repository
Stellarity Software • Vendor supported
[INTEGRATIONS](#) [REPOSITORY CONNECTORS](#)
Need Bamboo builds from TFS repositories? With this plugin you can do it!

★★★★ (4)
97 installations
Paid via Atlassian


Free trial
Buy now



Run CLI Actions in Bamboo
Bob Swift Atlassian Add-ons (an Appfire company) • Atlassian Verified
[ADMIN TOOLS](#) [TASKS](#)
Run Atlassian® Command Line Interface actions from a Bamboo build. Task for each of the CLI clients. Now it is easy to automate doing actions on your Confluence®, JIRA®, Bamboo, Bitbucket, HipChat, Crucible, or FishEye® instances.

★★★★ (1)
63 installations
Paid via Atlassian

Free trial
Buy now



Tasks for AWS (Bamboo AWS Plugin)
Utoolity • Atlassian Verified
[INTEGRATIONS](#) [TASKS](#)
Tasks for your DevOps workflows -- use Bamboo build and deployment projects to deploy and operate AWS and Docker resources via CloudFormation.

★★★★ (7)
323 installations
Paid via Atlassian

Free trial
Buy now

Ilustración 3:Extensiones de Bamboo

IV. Conclusiones

Gitlab CI es una herramienta con amplias ventajas como su precio que va de lo gratuito a lo privado, además que ofrece servicios como tener ramas privadas e importar proyectos que estén en otras herramientas como Jenkins o Circle CI.

Jenkins prueba continuamente las compilaciones del proyecto y es capaz de mostrar los errores que aparezcan a lo largo de las primeras etapas del desarrollo. Al usar Jenkins, se puede acelerar el proceso de desarrollo del código; ya que Jenkins puede automatizar, agilizar y aumentar el ritmo de toda la compilación y las pruebas de los proyectos.

Circle CI es una excelente alternativa para Jenkins que es una de las herramientas de construcción más utilizada. Esta herramienta tiene bastante utilidad y diferentes aplicaciones que nos ayudarán a tener un mejor y correcto desarrollo de software. Así mismo, nos ayuda a administrar de una manera eficaz la construcción de nuestro sistema.

La herramienta Bamboo, dispone de una integración con múltiples herramientas que permiten una sincronización con varios elementos de gestión, esto debido fundamentalmente a que es desarrollado por Atlassian quienes han desarrollado múltiples plataformas de gestión a distintos niveles en el desarrollo de software.

V. Recomendaciones

Se debe tener muy en cuenta las características de los equipos locales a la hora de usar una herramienta de esta naturaleza ya que podría ocasionar daños.

Las integraciones continuas se pueden romper regularmente debido a algunos pequeños cambios de configuración. La integración continua se detendrá y, por lo tanto, requiere cierta atención del desarrollador.

Es importante conocer cada una de las funcionalidades de estas aplicaciones ya que esto nos permitirá tener un correcto desarrollo de sistemas y evitar posibles errores y problemas en etapas avanzadas del desarrollo.

Se deberá evaluar las características del proyecto y la necesidad de integración del código ya que existen diversas herramientas tanto libres como de pago que, entregando pocas o muchas funcionalidades, esto permitirá optimizar la inversión con resultados óptimos en la construcción del software.

VI. Referencias

- [1] «¿Qué son la integración/distribución continuas (CI/CD)?» <https://www.redhat.com/es/topics/devops/what-is-ci-cd> (accedido jul. 03, 2021).
- [2] «GitLab CI/CD | GitLab». <https://docs.gitlab.com/ee/ci/> (accedido jul. 03, 2021).
- [3] «GitLab», *Capterra*. <https://www.capterra.ec/software/159806/gitlab> (accedido jul. 03, 2021).

- [4] Izertis, «Integración continua rápida y sencilla con GitLab CI». <https://www.izertis.com/es/-/blog/integracion-continua-rapida-y-sencilla-con-gitlab-ci> (accedido jul. 03, 2021).
- [5] U. V. Rodríguez, «Tipos de runner en GitLab CI: ¿Shell o docker?», *Adictos al trabajo*, abr. 29, 2020. <https://www.adictosaltrabajo.com/2020/04/29/tipos-de-runner-en-gitlab-ci-shell-o-docker/> (accedido jul. 03, 2021).
- [6] «CI/CD pipelines | GitLab». <https://docs.gitlab.com/ee/ci/pipelines/> (accedido jul. 03, 2021).
- [7] «Why we chose GitLab CI for our CI/CD solution», *GitLab*, oct. 17, 2016. <https://about.gitlab.com/blog/2016/10/17/gitlab-ci-oohlala/> (accedido jul. 03, 2021).
- [8] «Jenkins», *Capterra*. <https://www.capterra.ec/software/171026/jenkins> (accedido jul. 03, 2021).
- [9] «Jenkins: Ejecución de Jobs - Aprende a Montar un Entorno de Integración Continua (V)», *Roberto Crespo*, ago. 29, 2015. <http://www.robertocrespo.net/kaizen/aprende-a-montar-un-entorno-de-integracion-continua-v-ejecucion-de-jobs-en-jenkins/> (accedido jul. 03, 2021).
- [10] Atlassian, «Bamboo Continuous Integration and Deployment Build Server», *Atlassian*. <https://www.atlassian.com/software/bamboo> (accedido jul. 03, 2021).
- [11] «396597.pdf». Accedido: jul. 03, 2021. [En línea]. Disponible en: <https://repositorio.unican.es/xmlui/bitstream/handle/10902/12058/396597.pdf?sequence=1&isAllowed=y>