Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

INTRODUCCIÓN A DEVOPS: ANÁLISIS DE LA METODOLOGÍA Y EJEMPLO DE APLICACIÓN PRÁCTICA

Alumno: Jesús García Rodríguez

Tutor: Prof. D. Víctor Manuel Rivas Santos
Dpto: Informática

Septiembre, 2022

ÍNDICE

TABLA DE ILUSTRACIONES

1. INTRODUCCIÓN

En este primer apartado, dedicado a presentar el trabajo que se va a realizar, vamos a ver cuáles son los motivos por los que se ha elegido este tema, además de enumerar los propósitos que se desean alcanzar y de qué manera se lograrán.

1.1. Motivación

El desarrollo de software supone un factor de alta importancia en la actualidad teniendo en cuenta la considerable transformación digital que nuestra sociedad está experimentando en prácticamente cualquier ámbito: nuestra vida cada vez está más informatizada con el objetivo de que el ser humano experimente mejoras en comunicaciones, accesibilidad a la información, comodidad en su día a día u optimización del tiempo, entre otras cosas. Es por eso que este trabajo ha sido enfocado en este concepto.

Ahora bien, concretando aún más, se ha escogido DevOps como tema principal, ya que supone una metodología que cada vez más empresas adoptan para llevar a cabo su actividad laboral en cuanto a desarrollo de software se refiere, y esto se debe a varios motivos: se considera un método ágil, lo que supone una mayor rapidez y eficiencia durante el proceso de producción, ofrece un cierto nivel de adaptabilidad con respecto a las empresas, genera resultados con mayor frecuencia, y fomenta climas más colaborativos y fusionados en los equipos de trabajo, lo que beneficia a la calidad del trabajo realizado.

Por último, mencionar que DevOps, tanto como metodología e, incluso, como simple concepto, resulta bastante desconocido en general, especialmente en la población ajena al ámbito del desarrollo software, algo que veremos más adelante en otros apartados. Aún así, personalmente yo tampoco conocía DevOps hasta empezar a realizar el análisis de la metodología, por lo que durante estos meses he ido adquiriendo nuevos conceptos sobre esta, e incluso sobre el desarrollo software en general.

1.2. Objetivos

Con la realización de este TFG, se pretende llevar a cabo un análisis en profundidad sobre la metodología DevOps. Para ello, se estudiarán ciertos aspectos y componentes relacionados con la misma, como su origen, los principios sobre los que se sustenta, aspectos positivos y negativos, herramientas más comunes para aplicar la metodología, papeles a adoptar y características dentro de los equipos propios de DevOps, métodos a seguir durante el proceso de desarrollo y métricas aplicables que actúan como indicadores para poder conocer el progreso del proceso de producción y el rendimiento de los equipos de trabajo.

Además, se desea incluir una parte más enfocada a la investigación, enfocada en la población, con el objetivo de conocer en qué medida la metodología es conocida en la sociedad y cuáles serían sus niveles de aceptación, por lo que se ha creado y lanzado una breve encuesta que trata de abarcar estos aspectos.

Por último, se llevará a cabo el desarrollo de aplicación web simple, como ejemplo para cubrir la parte práctica de este trabajo, donde trataremos de simular como sería el proceso de desarrollo software haciendo uso de DevOps.

2. DEVOPS: PRIMERA TOMA DE CONTACTO CON LA METODOLOGÍA

¿Qué es lo primero que se nos viene a la cabeza cuando alguien nos menciona el término DevOps? La realidad es que prácticamente cualquier persona que no esté relacionada con el mundo de la informática y el desarrollo software desconoce tal concepto y su origen. Por ello, pretendo dedicar la primera parte del proyecto a aportar información a cualquier lector para que le sea posible conocer qué es DevOps, de donde viene dicho término, qué función aspira a desempeñar y los objetivos que busca conseguir.

DevOps se define como una metodología de trabajo con origen en el ámbito de la informática, cuyo principal objetivo es optimizar el desarrollo de aplicaciones de

manera que se reduzca el tiempo de trabajo, se lleve a cabo una mejor coordinación entre todas las partes que intervienen en dicho desarrollo y, así, se ofrezcan mejores resultados al cliente. [1]

"El término DevOps, que es una combinación de los términos ingleses development (desarrollo) y operations (operaciones), designa la unión de personas, procesos y tecnología para ofrecer valor a los clientes de forma constante" [2]. El propio concepto DevOps proviene de la unión de otros dos: desarrollo y operaciones. Por ello, y antes de continuar, vamos a analizar estos dos nuevos términos para averiguar sobre qué tratan.

Por un lado, tenemos el concepto de desarrollo software, que suele ser bastante conocido o al menos escuchado por cualquier persona, esté o no relacionada con el mundo de la informática. Dicho concepto trata sobre el proceso mediante el cual se lleva a cabo la producción, evaluación y mantenimiento de productos software. Por otra parte, está el concepto de operaciones TI, basado en la unión de 3 conceptos pertenecientes a una empresa, que serían las personas, los procesos y la tecnología, con la intención de sacar el máximo partido posible de estos tres elementos para que la organización obtenga beneficio de ello en diferentes aspectos.

2.1. Historia y origen de DevOps

El desarrollo de software supone una actividad que surgió hace muchos más años que DevOps. Es por ello que podemos preguntarnos acerca de cómo se llevaba a cabo dicha actividad antes de que esta metodología se creara. Lo cierto es que hasta que este momento llegó, existían marcos de trabajo de carácter más convencionales, caracterizados por un mayor nivel de aislamiento entre los diferentes sectores pertenecientes al proceso de creación de software. Esto reflejaba una serie de aspectos negativos que afectaban de diferente manera a todo lo relacionado con esta actividad, pasando por una deficiencia en la comunicación interna, una extensión de la duración de dicho proceso, dificultades a la hora de incluir modificaciones e innovaciones, y una falta de regularidad de aportaciones de trabajo. Todo ello se reflejaba en la productividad del equipo de trabajo, la calidad de los resultados que se

obtenían, un incremento del precio del producto y el esfuerzo dedicado, una mayor cantidad de fallos y situaciones de conflicto, y una entrega más retardada al cliente. [4]

Analizando DevOps en cuanto a su aparición se refiere, debemos saber que esta metodología es relativamente nueva, pues podemos situar su aparición en 2007 en manos de Patrick Debois (ver ilustración 1), un consultor de TI de origen belga, cuya intención es reducir algo que hemos mencionado previamente: la existente brecha interna en algunos casos entre los componentes que forman un proyecto software. Aun así, la fundación de la metodología no ocurrió exactamente en 2007, pues únicamente fue el momento en el que su creador, al detectar un existente problema entre el sector de desarrollo y el de operaciones, decide emprender una investigación en busca de posibles soluciones para mejorar esta situación.

Ilustración 1. Patrick Debois [39]

Debois pasa durante dos años, en los que mantiene presente la problemática existente en el mundo del desarrollo, acudiendo a exposiciones celebradas a nivel mundial, en las que recopila información sobre temas tratados de importancia en su investigación. Es a mediados de 2009, cuando obtiene la clave para llevar a cabo soluciones aplicables a su preocupación: colaboración de grupos de desarrolladores con el ámbito de operaciones.

Unos meses más tarde, en octubre, organiza una exposición para presentar los conceptos sobre el tema interpretados a raíz de las ideas que ha ido tomando durante esos años, la cual toma sede en la ciudad de Gante, en Bélgica, y para ello busca una

denominación con la que designar a dicha exposición. Toma en cuenta los dos ámbitos en los que pretende establecer cooperación (*Development* y *Operations*), y de ahí surge la conferencia *DevOpsDays*, evento al que deciden asistir un gran elenco de profesionales del medio. Una vez finalizado, se decide crear una etiqueta en Twitter para que la gente exponga cualquier aspecto u opinión relacionado con dicho evento: DevOps, por lo que, finalmente, la metodología adquiere esta designación. [4]

En una entrevista publicada por DEVOPS LATAM el 5 de mayo de 2020, Debois responde a preguntas sobre la tecnología que estamos estudiando: afirma, según opinión propia, que cuando se está llevando a cabo un proceso de desarrollo software, un primer paso que se debería tomar es el de tratar de conocer y comprender qué problemas están presentes en los demás desarrolladores, además de tener en cuenta los que nos puedan surgir a nosotros mismos, con el objetivo de analizar una situación en la que no nos encontramos solos y hacer fluir de mejor manera al equipo; además, menciona el concepto "*Pipeline*", que en el ámbito de DevOps se entiende como una forma de trabajo que busca conseguir que los desarrolladores puedan producir código de una manera más coherente, y lo incluye como factor de éxito en conjunto con los objetivos de la empresa. [5]

2.2. Cultura de DevOps

La metodología DevOps adopta una doctrina especialmente enfocada en una sólida colaboración y coordinación presente entre los componentes del equipo de trabajo, y cuya prioridad siempre sea el cliente y sus necesidades, con la intención de poder ofrecerle siempre los mejores y más ajustados resultados y, de esta forma, el producto cumpla con los requisitos requeridos por parte del mismo. Dicha cultura busca, de alguna manera, crear avances en entornos de trabajos tradicionales, los cuales generalmente han funcionado siempre de manera más dispersa.

Esta disciplina, además, pretende atribuir un grado de responsabilidad del producto muy equitativo en la totalidad del equipo de trabajo, ampliando las funciones de cada uno de sus miembros. De esta manera, personal de desarrollo y operaciones

establecerán un vínculo más cohesionado, lo que favorecerá una sintetización del desarrollo y mantenimiento del producto.

Otra característica de la cultura que adopta DevOps es la autonomía en sus equipos de trabajo. Para ello, debe existir un gran nivel de confianza en ellos y tienen que estar suficientemente equipados para llevar a cabo una toma de decisiones rápida, eso sí, siempre buscando estar lo más adaptada al resultado que el cliente busca. Cuando un equipo autónomo es capaz de llevar a cabo modificaciones en el producto por cuenta propia (siempre con el pensamiento de que son cambios lo más adecuados posibles), veremos que el tiempo de trabajo se ve considerablemente reducido. Además, la automatización es otro de los aspectos que está considerado de vital importancia en esta cultura.

Indagando aún más en esta metodología, podemos ver que ofrece un compromiso con la constante mejora del equipo de desarrollo a través de un rápido feedback. Esto se puede dar gracias al alto nivel de coordinación y comunicación que DevOps persigue en los equipos de trabajo, algo que, en entornos más aislados, sería muy complicado de conseguir. [6]

3. HERRAMIENTAS DE DEVOPS

Ya hemos explicado qué es DevOps y cuál es su objetivo, es decir, qué pretende mejorar y/o solucionar con su aparición. Aun así, tan sólo nos encontramos ante una metodología de trabajo a seguir, es decir, únicamente es una teoría sobre la que basar una práctica, por lo que depende de una serie de herramientas junto con las que pueda cumplir sus propósitos. Y hemos de decir que nos encontramos ante una metodología que hace uso de varias herramientas en lugar de una única, con el fin de sacar partido de diversas funcionalidades. Por ello, se ha llevado a cabo una investigación sobre las usadas por DevOps en la práctica, en la que se han encontrado numerosas opciones. De todas ellas, se ha hecho una selección entre las más destacadas, que, a su vez, son divididas en categorías en función al uso y funcionalidad que ofrece cada una de ellas. Esta clasificación está basada en una guía publicada por IBM sobre

DevOps. Vamos a ver cuáles son estos grupos y analizaremos las herramientas más conocidas de cada uno de ellos. [7]

3.1. Herramientas de gestión de proyectos

Estas herramientas se encargan de que los equipos puedan exponer una serie de requisitos para un proyecto, interpretarlos para establecer un conjunto de tareas y supervisarlas hasta su cumplimiento. La gestión de proyectos, vista como una actividad dentro del desarrollo software, engloba funciones a llevar a cabo, como la organización del trabajo, la cooperación del equipo o la valoración del progreso. Como ejemplos de este tipo de herramientas, podemos encontrar *GitHub Issues* o *Jira*.

3.1.1. Jira

Jira es una aplicación con una multitud de funcionalidades destinadas para la gestión de proyectos. Aun así, a la hora de su creación, el objetivo de esta herramienta era llevar a cabo el tratamiento de fallos y problemas que se diesen durante el desarrollo de dichos proyectos.

Ilustración 2. Logo Jira [40]

El origen de Jira se sitúa en 2002, cuando Atlassian decide crear una herramienta destinada para manejar conflictos, como hemos mencionado anteriormente. Actualmente, la herramienta es capaz de cubrir numerosas funciones,

como el manejo de requerimientos, tareas, proyectos, fallos, etc. Es por ello por lo que Jira puede ser usada por personas con diferentes tipos de roles dentro de un equipo. [8]

El funcionamiento de Jira es muy simple: está basado en el uso de una serie de elementos, a los que se les conoce como *issues*, pertenecientes al proyecto sobre el que estemos trabajando, que se encargan de identificar ítems que deben resolverse dentro del mismo. Existen varios tipos de issues según la funcionalidad que desempeñen, como pueden ser tareas, errores, ideas, historias, entre otros. Todos ellos son totalmente configurables según lo decida el usuario, pues cada tipo de issue dispone de una serie de aspectos modificables. Además, para cada uno de ellos, existe un flujo distinto, el cual representa el estado en el que se encuentra dicha issue en cada momento. Para que el equipo pueda controlar las issues del proyecto en el que se esté trabajando, se puede acceder a un tablero, el cual informa sobre la totalidad de los ítems en relación con el trabajo que se está llevando a cabo en el momento, el restante y el que ya ha sido finalizado. Además, Jira dispone de un lenguaje de consulta muy parecido al que se utiliza en las bases de datos, para acceder de una forma más profunda a los datos que recoge la aplicación: este lenguaje es conocido como *JQL* (*Jira Query Language*). [9]

Jira es una herramienta útil para múltiples tipos de proyectos, aunque está especialmente enfocada como recurso propio de metodologías ágiles.

3.2. Canalizaciones de CI/CD

Las herramientas de este grupo tienen como objetivo automatizar todo el proceso de desarrollo software. En este caso, encontramos como ejemplos Jenkins, *CircleCI*, *Spinnaker* o *ArgoCD*.

3.2.1. Jenkins

Jenkins es la herramienta más conocida y utilizada en el ámbito de la integración y distribución continuas. Básicamente, consiste en un servidor de código

abierto cuya utilidad se centra en gestionar proyectos de forma autónoma con el objetivo de aportar rapidez al desarrollo y beneficiar al equipo en cuanto a la integración de modificaciones.

Ilustración 3. Logo Jenkins [41]

El origen de Jenkins tiene lugar 2011, en manos del programador informático de origen japonés Kohsuke Kawaguchi, aunque la herramienta comenzó a elaborarse en 2004 dentro del proyecto Hudson. [10]

Entender cómo trabaja Jenkins es bastante sencillo: en primer lugar, una vez que un miembro del equipo realice una aportación al repositorio de trabajo, la herramienta estará al tanto de ello, pues constantemente lleva a cabo chequeos en busca de nuevas modificaciones. En ese momento, se encarga de compilar y se genera un build, que puede dar dos resultados. Si fracasa, se comunica al equipo sobre el error; en cambio, si resulta válido, pasa a un proceso de pruebas del que se obtendrán unos resultados que serán comunicados al equipo. Una vez se haya completado todo esto, Jenkins vuelve a repetir la secuencia. Todo este proceso de funcionamiento constituye lo que es un pipeline de Jenkins, concepto que ya vimos en otro de los apartados anteriores. Este consta de una serie de elementos: el primero sería el fichero donde se recogería el propio pipeline, conocido como Jenkinsfile. Dentro de él, se encuentran las fases o Stages, que comprenden cualquier conjunto de acciones que suponen un proceso concreto, y estas a su vez contienen a los pasos o Steps, que constituyen acciones simples. [11]

Con todo esto, podemos deducir que Jenkins es una herramienta que ofrece un método más autónomo de tratar y probar el código sobre el que trabaja un equipo. Además, ofrece una serie de ventajas respecto a otras herramientas: es una herramienta disponible para cualquier usuario, sin coste alguno, multiplataforma y que ofrece una amplia funcionalidad gracias a los complementos de los que dispone. Aun así, la interfaz que Jenkins ofrece resulta bastante obsoleta y complicada de comprender, al igual que algunos de sus complementos.

3.3. Repositorios de código fuente colaborativo

Los repositorios almacenes de código accesibles donde el equipo de trabajo puede ir desarrollando sobre una misma base, y a la que pueden añadir aportaciones y modificaciones, siempre bajo una regulación mediante un sistema de control de versiones. Dentro de este grupo, encontramos opciones muy conocidas y utilizadas por millones de usuarios, tales como *GitHub* o *GitLab*.

3.3.1. GitHub

GitHub es la herramienta colaborativa mayormente elegida a nivel global por los programadores para el desarrollo y la gestión de proyectos software al trabajar en equipo, pues ofrece un servicio basado en almacenamiento remoto de archivos accesible desde cualquier dispositivo con conexión a la red. Además, dispone de un sistema de control de versiones, ficheros creados automáticamente junto con los proyectos con información sobre el producto que se ofrece, ya sean datos sobre el proceso de desarrollo del mismo, sobre los desarrolladores, guías de ayuda para la instalación y uso de la herramienta, y muchas más funcionalidades.

Ilustración 4. Logo GitHub [42]

GitHub, como organización, surgió en 2007 en manos de Chris Wanstrath, P.J. Hyett, Tom Preston-Werner y Scott Chacon, cuatro empresarios tecnológicos. Aun así, no fue hasta febrero de 2008 cuando se produjo el lanzamiento de la herramienta conocida actualmente como GitHub. Más adelante, en 2018, la empresa Microsoft llevó a cabo la compra de GitHub por más de 6.000 millones de euros, según informó la parte compradora. [12]

GitHub es una herramienta de acceso libre para cualquier usuario de la web sin coste alguno, aunque dispone de planes de pago que ofrecen más funcionalidades en función de la cuota seleccionada. Para hacer uso de este servicio, el usuario debe disponer o crear una cuenta con la que acceder y en la que almacenar información sobre el progreso que llevará a cabo. Cuando se crea un repositorio, se ofrece la opción de decidir la visibilidad del mismo, pudiendo ser público (accesible para cualquier usuario, aunque no esté registrado en el servicio) o privado (sólo accesible para usuarios a los que previamente se les ha permitido el acceso al mismo). Aunque GitHub ofrece una interfaz de usuario muy intiuitiva y fácil de comprender y utilizar, también puede ser controlada mediante una consola de línea de comandos instalable que ofrece Git para sus herramientas, conocida como *Git Bash*.

Como hemos comentado anteriormente, este servicio ofrece multitud de funcionalidades de bastante utilidad para el desarrollo software en equipo. En primer lugar, nos encontramos con un sistema de control de versiones, encargado principalmente de mantener un constante chequeo sobre el código almacenado en el repositorio en busca de nuevas modificaciones, creando un registro donde

almacenarlas y ordenarlas en diferentes versiones del producto que está siendo desarrollado, ofreciendo además la posibilidad de retomar cualquiera de estas versiones en caso de que se produzca algún contratiempo durante el desarrollo. El sistema de control de versiones (VCS) que utiliza esta herramienta es conocido como Git, mencionado anteriormente.

Otra de las funcionalidades que esta herramienta nos ofrece consiste en un espacio perteneciente a cada proyecto, en el que será posible almacenar información sobre el mismo a modo de ofrecer la posibilidad a otros usuarios de participar ofreciendo colaboración y mejoras, o también para recibir reseñas de terceros. Este espacio tendrá la misma visibilidad que el proyecto. Además, cada proyecto contendrá, al menos, un fichero conocido como *README.md*, encargado de recopilar información sobre las capacidades del producto y su utilidad, pautas de ayuda para hacer uso del mismo e información sobre los colaboradores en el desarrollo del proyecto, tal y como hemos comentado a modo introductorio previamente. [13]

Dentro de cada repositorio, tendremos disponibles varias opciones: por ejemplo, una sección en la que se muestre un plan de pautas ordenadas para seguir durante el desarrollo, conocida como *GitHub Issues*; una sección dedicada a seguridad, en la que se ofrecen opciones como la creación de políticas de seguridad y configuración de avisos y alertas; una sección en la que podremos acceder a información sobre la actividad llevada a cabo en el repositorio; y una página de modificación de la configuración del repositorio, en la que se nos ofrecen múltiples opciones que establecer a nuestra preferencia.

3.4. Marcos de automatización de pruebas

Las herramientas que pertenecen a esta categoría tienen como objetivo automatizar el proceso de pruebas de software aplicado al proyecto que se está desarrollando. Encontramos marcos de prueba, como pueden ser *Selenium*, *Appium*, *Katalon*, *Robot Framework* o *Serenity*.

3.4.1. Selenium

Selenium es una herramienta de automatización de testeo de software cuya función pasa por llevar a cabo inspecciones sobre el trabajo resultante de los desarrolladores, de manera que los equipos, en la medida de lo posible, reduzcan tiempo y coste en llevar a cabo manualmente estos procesos de evaluación.

Ilustración 5. Logo Selenium [43]

Su origen se remonta a 2004, año en el que Jason Huggins crea esta herramienta, inspirado por la necesidad de un servicio de automatización de pruebas de software, debido a que en ese momento estaba desarrollando un proyecto web el cual requería numerosos procesos de testeo. [14]

El uso de Selenium ofrece una serie de ventajas bastante interesantes. Constituye un proyecto Open Source y su uso es gratuito. Además, ofrece la posibilidad de grabar, modificar y depurar los test que se lleven a cabo. Con el objetivo de ampliar el uso de esta herramienta, Selenium se ha hecho compatible con los principales navegadores elegidos por los usuarios, además de presentarse como un servicio multiplataforma al poder desplegarse en los principales sistemas operativos. A la hora de crear pruebas, también ofrece la posibilidad de utilizar múltiples lenguajes de programación, generalmente conocidos. Por último, con respecto al proceso de testeo, Selenium permite llevar a cabo varias pruebas de forma simultánea.

Selenium ofrece una serie de componentes con los que trabajar, en lugar de centrarlos todos en una sola herramienta. En primer lugar, tenemos *Selenium IDE*: como bien sabemos, un IDE (Integrated Development Enviroment) en el ámbito de la informática es un entorno en el que llevar a cabo desarrollo software, por lo que

aplicado a la herramienta que estamos viendo, podemos definirlo como el entorno utilizado por los usuarios de Selenium para crear y desarrollar distintos test de evaluación. Ofrece varias funcionalidades: usando el IDE se pueden crear tests independientes, o crear un conjunto completo de pruebas, también permite, como hemos mencionado anteriormente, la ejecución simultánea de pruebas en varios navegadores, algo que es necesario para considerar válido el proceso de testeo y que es conocido como proceso de pruebas de navegación cruzada, ofrece la opción de depurar las pruebas, y dispone de numerosas órdenes que el usuario puede utilizar. Por otro lado, tenemos *Selenium RC*, basado principalmente en un servidor que actúa como mediador entre las órdenes y los navegadores sobre los que se aplican las pruebas, y unas bibliotecas en la parte del cliente. Para hacer uso de esta herramienta, debemos tener la aplicación de control remoto (*Selenium Remote Control Server*). En la actualidad, este componente se considera bastante complejo, y por ello se ha sustituido prácticamente por otro de los componentes que vamos a ver: *Selenium WebDriver*, que consiste en una aplicación de automatización de pruebas de interfaz de usuario, aunque a diferencia de la versión de control remoto expuesta anteriormente, esta vez la comunicación con el navegador está libre de intermediarios. Por ello, esta versión de la herramienta se considera como una mejora de la anterior, además de que cuenta con una estructura mucho más manejable y, al establecer comunicación directa, trabaja con mayor rapidez; otra mejora que presenta con respecto a la versión anterior es la disponibilidad para dispositivos móviles. Finalmente, tenemos un último componente, *Selenium Grid*. Su objetivo principalmente es la ejecución simultánea de varios testeos en diferentes dispositivos, y esto lo lleva a cabo haciendo uso de un servidor (HUB) que se encarga de enlazar máquinas remotas (nodos) donde se llevan a cabo dichos procesos de testeo. [15]

3.5. Herramientas de gestión de configuración

Este tipo de herramientas tienen como objetivo mejorar la infraestructura usada por el equipo de trabajo, implementando en ella código a modo de evitar procesos manuales y adquirir la mayor automatización a la hora de usar dicha infraestructura. Entre ellas, encontramos *Ansible* (*Red Hat*), *Chef*, *Puppet* o *Terraform*.

3.5.1. Ansible

Ansible es una herramienta que permite a quienes la usan gestionar procesos de configuración, instalación de software y otras acciones de manera automática, lo que ofrece comodidad y rapidez a los usuarios, algo muy preciado en la filosofía de DevOps.

Ilustración 6. Logo Ansible [44]

Esta herramienta, cuyo software es Open Source y de uso gratuito, fue creada por Michael DeHaan en 2012, año en el que se llevó a cabo el lanzamiento de su primera versión. Tres años después, en 2015, la empresa Red Hat lleva a cabo la adquisición de Ansible con la intención de ofrecer a sus clientes un servicio orientado a la filosofía DevOps. [16]

Ansible, a modo de configuración, presenta dos opciones: la primera hace uso de un fichero conocido como *playbook* (ver figura 3.6), que es un fichero en el que se incluirán todas las variables necesarias para la realización de una actividad concreta; la segunda, mediante un conjunto de directorios estructurado, que contendrán ficheros similares al que acabamos de mencionar.

```
---

 - hosts: apache
   sudo: yes

   tasks:
      - name: install apache2

        apt: name=apache2 update_cache=yes state=latest …
```

Ilustración 7. Ejemplo de *playbook* [45]

Ansible lleva a cabo su funcionamiento haciendo uso del protocolo SSH, el cual permite administrar a distancia equipos remotos haciendo uso de la red. Antes de hacer uso de la herramienta, debemos asegurarnos que todos los sistemas que se vayan a gestionar cuenten con Python, y debido al soporte que Ansible ofrece, el equipo desde el que se lleve a cabo la gestión de los demás no puede tener como sistema operativo Windows. Una vez se esté haciendo uso de la herramienta, el dispositivo desde el que se llevará a cabo todo el proceso (controlador) establecerá conexión con los demás equipos (nodos), en los que ejecutará una serie de módulos, que serán suprimidos una vez hayan sido instaladas las aplicaciones que se precisan. [17]

Ansible ofrece, además, una funcionalidad conocida como roles, la cual ofrece a quienes la usan la posibilidad de organizar según una distribución determinada elementos usados para la gestión remota de los equipos. Esta distribución se basa en directorios dictaminados por Ansible (ver figura 3.7), y cada cual incluye un fichero (*main.yml*) que contiene el conjunto de tareas que va a llevar a cabo cada uno de los roles (ver figura 3.8). Los ficheros con esta extensión (*.yml*) contienen información representada en *YAML*, un lenguaje de serialización de datos considerado como más comprensible que otros para los usuarios. Además, como los roles pueden ser usados más de una vez y por más de un usuario, se puede hacer una búsqueda sobre roles disponibles en una de las herramientas complementarias de Ansible que recopila multitud de roles creados por usuarios de este servicio, conocida como *Galaxy*. [18]

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them here
(optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # main playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier
tasks/                    # task files included from playbooks
    webservers-extra.yml  # <-- avoids confusing playbook with task files

roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the
webtier role
    monitoring/           # ""
    fooapp/               # ""
```

Ilustración 8. Ejemplo de directorio Ansible [46]

Como podemos observar en la figura 3.7, nos encontramos ante un ejemplo de directorio con el que trabaja Ansible, tal y como hemos mencionado anteriormente. Si prestamos especial atención, veremos que dentro del mismo existe un apartado dedicado a los roles también vistos previamente, donde se recoge su información más

relevante (variables para el rol, tareas que realizará cada uno, plantillas para cambios en los mismos, metadatos, etc.).

```
---
# file: roles/common/tasks/main.yml

- name: be sure ntp is installed
  yum:
    name: ntp
    state: present
  tags: ntp

- name: be sure ntp is configured
  template:
    src: ntp.conf.j2
    dest: /etc/ntp.conf
  notify:
    - restart ntpd
  tags: ntp

- name: be sure ntpd is running and enabled
  service:
    name: ntpd
    state: started
    enabled: yes
  tags: ntp
```

Ilustración 9. Ejemplo de fichero *main.yml* [47]

Aquí tenemos un ejemplo de archivo *main.yml*, el cuál ha sido recientemente mencionado. En este fichero se recogen las diferentes tareas que son asignadas a distintos roles, las cuáles son configuradas mediante una serie de parámetros que podemos establecer y modificar según sea necesario.

3.6. Herramientas de monitoreo

Su función es la detección y solución de conflictos en el sistema, además de encargarse de la recolección y tratamiento de la información a tiempo real para observar la forma en la que las modificaciones que se producen en el código influyen en la aplicación. Para ello, se ofrecen herramientas tales como *Nagios*, *Prometheus* o *Splunk*.

3.6.1. Nagios

Nagios es una herramienta de control de máquinas (para la parte de hardware) y procesos (para la parte de software), la cual informa a sus usuarios sobre el estado de los mismos y notifica cuándo se está produciendo sobre ellos un uso inadecuado. [19]

Ilustración 10. Logo Nagios [48]

El objetivo de Nagios y su funcionamiento son aspectos bastante sencillos y fáciles de comprender. Nagios, para llevar a cabo su labor, se basa en una estructura en la que exista un servidor y un cliente: en la parte del servidor, se incluirá el programa que llevará a cabo el proceso de monitoreo, mientras que la parte del cliente estará destinada a ser controlada. Para el servicio de monitorización, debemos tener en cuenta varios tipos de elementos: los hosts, que constituyen cualquier sistema sobre el cual decidamos llevar a cabo un monitoreo; los servicios, que se corresponden con las propiedades de los hosts que se desean observar; los comandos, que tienen como objetivo manejar los procesos de monitoreo; y los contactos, que indican a quien deben ir dirigidas las alertas cuando se produzca cualquier caso de conflicto, las cuales pueden ser notificadas por diferentes vías, como un email, un mensaje, etc. Otra alternativa que ofrece Nagios consiste en monitoreos llevados a cabo por el servidor con la ayuda de otra aplicación que es la que busca acceder a los datos, conocida como Nagios Service Check Acceptor.

Una vez se lleva a cabo el chequeo, el servidor comunica los resultados haciendo uso de uno de estos valores: 0 para indicar que todo ha ido bien y no se ha encontrado ningún conflicto; 1 para indicar que se ha generado una alerta; 2 para indicar que ha ocurrido un conflicto importante; y 3 cuando no ha sido posible precisar un resultado concreto.

El uso de Nagios ofrece multitud de ventajas, la razón por la que es una herramienta con gran cantidad de usuarios: es bastante práctica en relación a qué características ofrece, además de que dispone de una multitud de componentes para su uso y es bastante adaptable a sus usuarios. Aun así, no todo son ventajas si tenemos en cuenta que Nagios no cuenta con una guía de uso en castellano, por lo que utilizar esta herramienta puede resultar bastante complicado. [20]

3.7. Herramientas de comentarios continuos

Se encargan de recopilar información y estadísticas sobre los usuarios y las acciones que llevan a cabo, con el objetivo de obtener datos que informen sobre aspectos relacionados con el cliente, como pueden ser su opinión sobre el producto, problemas experimentados, etc.

3.8. DevOps y la nube

Como hemos visto, DevOps es una metodología que busca establecer una mayor cohesión entre las diferentes partes pertenecientes al proceso de desarrollo software, en la que se promueven aspectos como una comunicación más óptima con respecto al equipo de trabajo, automatización de procesos o entregas más frecuentes y continuas, entre otros. Para llevar esto a cabo y alcanzar los objetivos que esta metodología persigue, suele recurrirse a lo que conocemos como *Cloud Computing*, que en definitiva se refiere a servicios informáticos en la nube.

Ilustración 11. Concepto de *Cloud Computing* [49]

El concepto de *Cloud Computing* hace referencia a una serie de herramientas y tecnologías accesibles a través de la red, las cuales ofrecen servicios generalmente de pago contratables por cualquier usuario o empresa para poder llevar a cabo labores de trabajo de una manera más eficiente que haciendo uso de técnicas tradicionales. En un enfoque más dirigido a DevOps, los servicios en la nube proporcionan una amplia gama de productos y prestaciones de diferentes tipos y destinadas a facilitar la realización de distintas actividades relacionadas con el alojamiento de información de forma remota, comunicación y celebración de reuniones entre miembros de los grupos de trabajo y con el cliente, organización y gestión de proyectos y labores de desarrollo, autogestión de procesos y tareas, etc. Dichas utilidades aportan una serie de beneficios reflejados en la práctica: permite una conexión más extendida entre desarrolladores y profesionales al reducir la necesidad de llevar a cabo encuentros presenciales, lo que ofrece mayor libertad a los equipos, el software sobre el cual se está trabajando adquiere mayor nivel de accesibilidad al estar disponible en repositorios y almacenes en la nube, se incrementa la seguridad en el desarrollo contra pérdidas de información y amenazas, los costes de personal e infraestructura se ven reducidos considerablemente y se reduce el tiempo de inacción al establecer procesos autónomos y constantes. [21]

3.8.1. Azure

Azure es una tecnología basada en un conjunto de utilidades a los profesionales del desarrollo de software para llevar a cabo sus proyectos y labores de desarrollo. Aunque dispone de un modo de desarrollo local basado en el uso de un servidor propio, ofrece también servicios para trabajar en la nube, lo cual se ajusta a todo lo que acabamos de ver.

Ilustración 12. Logo Microsoft Azure [50]

Entre estas utilidades, encontramos almacenes de código remotos (*Azure Repos*), utilidades para CI/CD (*Azure Pipelines*) correspondientes a conceptos vistos en otros apartados, aplicaciones destinadas a la organización y gestión de proyectos (*Azure Boards*), aplicaciones para aplicar procesos de testeo (*Azure Test Plans*), y espacios colaborativos para compartir información (*Azure DevOps Wiki*). [22]

4. MÉTODOS DE DEVOPS

Cuando las empresas adoptan la filosofía DevOps como forma de trabajo, hacen uso de una serie de métodos que buscan aportar velocidad, ahorro y optimización en las labores de los equipos de desarrollo. A diferencia de las herramientas, que previamente hemos visto y analizado, los métodos están más enfocados a un planteamiento teórico de cómo organizar y distribuir el trabajo a realizar para que, posteriormente, los equipos de desarrollo lo trasladen a la práctica. A continuación, veremos los tres casos más conocidos y utilizados dentro de la metodología DevOps.

4.1. Scrum

Nos encontramos ante la metodología ágil más conocida, Scrum. Su principal objetivo es facilitar la labor de trabajo del equipo en cuanto al desarrollo llevado a cabo, pudiendo ofrecer así mejores resultados finales. El uso de este método está enfocado a proyectos de mayor nivel de complejidad.

Ilustración 13. Logo Scrum [51]

La técnica Scrum contiene una serie de elementos implícitos: **roles** dentro de un **equipo**, **reglas**, **eventos** y **artefactos**.

En primer lugar, encontramos el Equipo Scrum (*Scrum Team*), cuyas dos propiedades destacadas son autoorganización (siempre escogen la mejor manera de cumplir con su labor de forma autónoma) y multifuncionalidad (el propio equipo es autosuficiente por sí mismo). Está formado por una serie de miembros que cumplen unos roles establecidos: el Dueño del Producto (*Product Owner*), quien se encarga de comunicar al equipo un panorama de lo que se pretende conseguir, algo que plasma en el

Product Backlog, que veremos más adelante en otro apartado; y, por otra parte, tenemos el *Scrum Master*, figura líder cuya función se basa en hacer que el equipo cumpla con su labor para obtener los resultados esperados.

Scrum cuenta también con una serie de eventos encargados de mantener controlado el progreso del desarrollo del equipo sobre el producto de forma regular. Como primer evento, nos encontramos con el *Sprint*, considerado como un período de tiempo no más largo de un mes en el que se desarrolla un resultado parcial del producto que puede ser usado, aunque no con toda la funcionalidad al completo. A estas entregas se les conoce como incrementos. La finalización de un *Sprint* supone el comienzo del siguiente, a excepción del *Sprint Final*, donde se pretende disponer del resultado final del producto.

Dentro de un *Sprint*, se encuentran y se basan todos los demás eventos. Por un lado, tenemos los *Daily Scrums* (reuniones diarias), que son reuniones fechadas a una misma hora y lugar diariamente, en las que en 15 minutos (tiempo que duran estas reuniones), el equipo de trabajo se reúne para comentar sobre el trabajo realizado en las últimas 24 horas y, a partir de esto, planificar el trabajo que se realizará hasta la siguiente reunión; otro de los eventos es el *Sprint Planning* (planificación del sprint), en el que se reúne el equipo para discutir y decidir sobre qué es lo que se pretende llevar a cabo durante el *Sprint* y cuál será el resultado esperado al finalizar. Una vez finalizado un *Sprint*, se lleva a cabo otro evento conocido como *Sprint Review* (revisión del sprint), en el que se lleva a cabo una revisión del trabajo realizado durante el tiempo que se ha dedicado al último incremento. A partir de aquí, se decide si se deben aplicar cambios sobre el *Product Backlog* a modo de añadir mejorías sobre el producto. Por último, una vez haya finalizado esta reunión de revisión del producto, se lleva a cabo el último de los eventos que nos queda por ver: el *Sprint Retrospective* (retrospectiva del sprint). Esta reunión, la cual tiene lugar antes del siguiente *Sprint Planning*, tiene como objetivo obtener valoraciones sobre el desarrollo del último Sprint, es decir, realizar autocrítica sobre cómo se está llevando a cabo la forma de trabajo del equipo a modo de poder considerar modificaciones en la misma para obtener siempre mejores resultados.

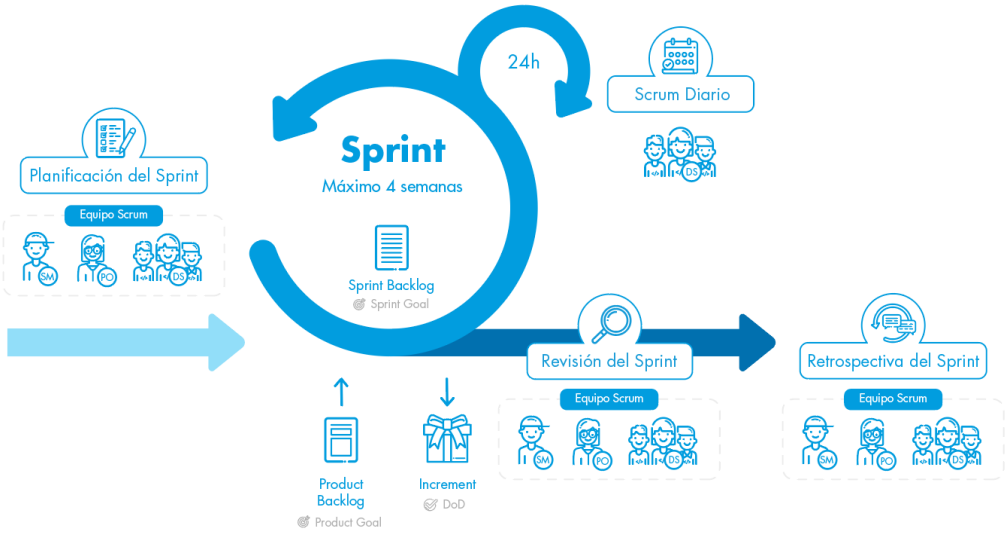

Ilustración 14. Esquema conceptual de Scrum [52]

Una vez visto esto, trataremos sobre los artefactos propios de *Scrum*. Estos elementos se encargan de ofrecer una visión sobre lo que es el producto a desarrollar, los requisitos que se ofertan para la obtención del mismo y la organización que se está llevando a cabo durante el progreso de trabajo. De esta manera, encontramos 3 artefactos: el *Product Backlog* (mencionado anteriormente), que es un registro de los requerimientos modificable durante el desarrollo del producto, y del que es propietario el *Product Owner*; el *Sprint Backlog*, correspondiente al Sprint en el cual se está trabajando en el momento, recoge los requisitos que se están desarrollando durante dicho *Sprint*, y acepta cambios mientras dicho período aún no haya finalizado. Finalmente, tenemos el *Incremento*, supone una acumulación de todos los requisitos desarrollados hasta la entrega del mismo, en la que se ofrece una versión del producto parcialmente utilizable.

Un último aspecto de Scrum son las reglas: algunas de ellas han sido mencionadas previamente para explicar todos los aspectos que ya se han visto de la metodología, como por ejemplo la duración de algunos eventos, el orden y el momento concretos en el que se deben dar dichos eventos, las restricciones del *Daily Scrum*, etc. Otras reglas que Scrum establece es la improrrogabilidad de las fechas establecidas para la finalización de los *Sprints* y la imposibilidad de modificar la planificación que se ha hecho para el *Sprint* en curso, pues esto afectaría al seguimiento que se lleva a cabo

sobre el ritmo de trabajo; además, no se considerará que un requisito ha sido finalizado hasta el momento en que pase por una validación determinada. [23]

4.2. Kanban

Kanban, visto como una metodología de trabajo, tiene como objetivo ayudar a los equipos de desarrollo a organizar sus labores dentro de un proyecto, aportando una visión general de estas que irá siendo modificada a medida que se lleva a cabo el desarrollo de dicho proyecto. De esta manera, se busca que el equipo conozca cuál es la distribución a seguir y que pueda llevarse a cabo un seguimiento informativo en todo momento sobre el estado de las asignaciones en las que se divide el proyecto.

El origen de Kanban se remonta hace unos siglos: el término *Kanban* aparece aproximadamente a principios del siglo XVII en Japón, cuando el comercio buscaba atraer el interés de la clientela. Dicho término, que proviene del japonés, hace referencia a una especie de cartel o letrero utilizado para trasmitir información de forma visual. Es por ello que, si trasladamos este término al ámbito de la organización de proyectos, obtenemos como resultado una serie de tarjetas en las que plasmar la distribución de su desarrollo, dividida en tareas que serán recogidas en dichas tarjetas.

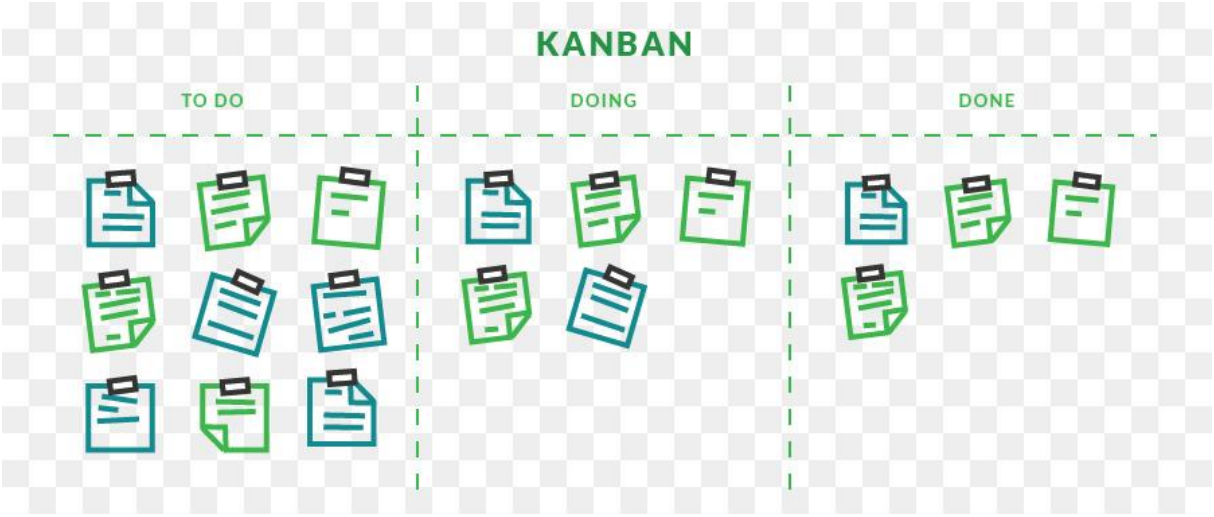

Ilustración 15. Concepto de tablero Kanban [53]

Tiempo después, aproximadamente a mediados del siglo XX, la empresa Toyota que conocemos hoy en día atravesaba momentos complicados: su producción, en relación con la competencia americana, no lograba alcanzar los niveles esperados, algo que la empresa decidió tratar de cambiar a base de realizar modificaciones en cuanto a la forma de llevar a cabo las labores de producción. Por aquel entonces, el ingeniero industrial Taiichi Ohno acababa de acceder a la empresa y fue escalando puestos rápidamente. Ante el presente problema que se planteaba en la organización, encontró varios aspectos negativos, entre los cuales estaba el problema de una producción con excedentes. La solución a esto podía llevarse a cabo ejecutando una producción basada en la necesidad instantánea, es decir, producir únicamente lo que se pedía y en el momento que se pedía, algo que radicaba en una duda sobre cómo indicar el requerimiento de nuevas producciones. Para intentar averiguar cómo llevar a cabo esta situación, emprendió un viaje hasta las empresas americanas, donde observó la manera en la que trabajaban y producían de forma eficiente, basada en el uso de tarjetas para indicar la necesidad de nuevas fabricaciones a raíz de lo que los clientes solicitaban. En el momento de regreso, adoptó dicha forma de producción, la cual supuso un beneficio en cuanto a la gestión del inventario y aportó eficiencia al proceso de fabricación. Esto, finalmente, hizo que la empresa experimentara un auge realmente significativo dentro del sector.

Desde aquellos años, hasta prácticamente la actualidad, dicha forma de trabajo tuvo tales niveles de aceptación por todos los beneficios que aportaba a quienes requerían a su uso, que muchas organizaciones decidieron comenzar a adoptar esta metodología, incluido el sector de desarrollo software, el cual es en el que centramos nuestra atención. [24]

Cuando se sigue un ritmo de trabajo definido por la metodología Kanban, es necesario tener en cuenta varios aspectos. Se deben observar todos los elementos que recoge el tablero de trabajo, con el fin de ser consciente en todo momento sobre las labores a realizar, y utilizar tarjetas de distintos tipos en función de la variedad de la información y lo que muestran, a modo de facilitar la comprensión de quien recurre a ellas. Además, se debe ser coherente a la hora de adquirir una determinada carga de trabajo, pues esta debe ser razonable y posible de llevar a cabo, evitando las

sobrecargas en la medida de lo posible. También resulta interesante evaluar lo que se está llevando a cabo en cuanto al tiempo que conlleva.

Existe una herramienta con la que poder desarrollar un proyecto en base a esta metodología que acabamos de ver: tableros de Kanban. Estos se basan en el uso de tarjetas que recogen la división del trabajo a realizar en tareas, que son asignadas a los diferentes miembros del grupo de desarrollo.

Ilustración 16. Logo Kanban tool [54]

Un tablero de Kanban dispone de varios componentes: elementos ópticos, mencionados anteriormente, con los que conseguir una mayor comprensión de lo que refleja dicho tablero, columnas, las cuales constituyen una secuencia de pasos o estados por los que pasa una actividad desde su inicio hasta su finalización, también existen unos indicadores del número máximo de tarjetas que puede contener una columna al mismo tiempo, a modo de regular la carga de trabajo que se pretende asumir, puntos de compromiso, que suponen el instante en el que acuerda una innovación y se lleva a cabo, y puntos de entrega que coinciden con la disposición al usuario de lo que se ha producido.

Existen dos modalidades de tableros Kanban: disponemos de tableros físicos, los cuales siguen el mismo funcionamiento previamente indicado, haciendo uso de anotaciones mediante posits o elementos similares, los cuales serán añadidos a un tablero que dispondrá de columnas por las que irán pasando dichos posits a medida que se vaya progresando en las labores de trabajo. Aunque resulte una forma de organización realmente práctica, no resulta válida siempre. Cuando nos encontramos ante una modalidad de trabajo a distancia, este tipo de tableros pierde prácticamente toda su efectividad y se debe recurrir a una herramienta basada en software que permita el acceso a todos los miembros del equipo, independientemente del lugar y el

momento en el que se encuentren. Es por ello que existe otra modalidad para esta herramienta: tableros virtuales. Para un escenario en el que las labores de desarrollo se llevan a cabo de forma no presencial, este tipo de tableros resulta de mucha utilidad, pues permite a los equipos plasmar toda la información necesaria en un espacio al que todo el equipo puede acceder desde cualquier dispositivo y desde cualquier lugar, además de poder hacerlo en cualquier momento al tener disponible dicha herramienta durante la totalidad del tiempo, por lo que ofrece, así, más disponibilidad y libertad a los miembros del equipo. Este tipo de tableros también basa su estructura en todo lo que hemos visto anteriormente, haciendo uso de columnas y de ítems que recojan las tareas a realizar, las cuales irán atravesando dichas columnas. Además, este tipo de herramientas siempre suele resultar de fácil comprensión y manejo, ya que ofrecen interfaces muy intuitivas y que no suelen ser complejas, aunque con multitud de funcionalidades para configurar, de forma que se adapten al máximo a las necesidades de cada proyecto. [25]

4.3. Scrumban

Con respecto a los dos métodos que acabamos de ver, es cierto que existen multitud de equipos que utilizan Scrum, aunque muchos otros prefieren hacer uso de Kanban. Aun así, ciertos aspectos de las metodologías pueden resultar bastante interesantes para ambos equipos, por lo que, en ciertas ocasiones, un equipo que utilice normalmente Scrum pretenda comenzar a hacer uso de funcionalidades de Kanban, y, al contrario, equipos fieles a Kanban empiecen a interesarse por aspectos que ofrezca Scrum. Por tanto, principalmente es por ello por lo que surgió una nueva metodología resultado de una conjugación de ambas, conocida como Scrumban, la cual pretende reunir las mejores características de cada una de ellas. Por tanto, su creación tiene como objetivo principal servir de ayuda para los equipos que pretenden cambiar de una metodología a otra, minimizando en la medida de lo posible la dificultad que supondría pasar de una a otra de manera radical. Visto esto, aunque es una metodología de la que cualquiera puede hacer uso, es cierto que está más orientada a equipos acostumbrados a trabajar con Scrum o Kanban. De hecho, si profundizamos aún más, veremos que esta metodología está especialmente más enfocada a Kanban que a Scrum por el hecho de no imponer tantas limitaciones como

lo hace esta última, y es incluso algo más utilizada por equipos Scrum que necesiten trasladarse parcialmente a Kanban. Sí que es cierto que los resultados que se obtienen mediante la técnica Scrum suelen ser bastante satisfactorios: el problema aparece cuando un equipo pretende adherirse a dicha metodología, ya que no resulta algo fácil y requiere demasiado tiempo y personal de apoyo para adquirir formación, cosa que no interesa a algunas empresas en ciertas ocasiones.

Ilustración 17. Concepto de Scrumban [55]

Antes de continuar analizando esta metodología, vamos a ver qué aspectos son diferentes entre Scrum y Kanban. Para empezar, debemos saber que el desarrollo en Scrum se divide en ciclos de tiempo predefinidos, los cuales son inaplazables, y que, como vimos anteriormente, conocemos como *Sprints*, durante los cuales no se admiten modificaciones del trabajo acordado en la pertinentes reuniones; por su parte, en Kanban, el trabajo atraviesa un régimen de estados que indican como va progresando, y en este caso sí se permiten modificaciones de ítems que aún no se encuentren en el estado inicial o no lo hayan sobrepasado. Aun así, estos aspectos de ambos métodos son los que limitan la carga que se está llevando a cabo en el momento. Otra diferencia es la asignación de distintos papeles a los miembros del equipo, cosa que no ocurre en Kanban: además, como vimos anteriormente, los equipos Scrum son característicos por contar con miembros con cualificaciones

variadas, mientras que Kanban admite grupos con habilidades más específicas. Por último, en cuanto a la organización de la carga de trabajo, en Scrum se calcula de manera aproximadamente al tiempo que llevará la realización de las tareas según el ritmo de los equipos, algo que no ocurre en Kanban. Como vemos, son numerosas las diferencias entre ambas metodologías, y aunque son más destacadas, existen otras.

Visto esto, pasamos a ver cuáles son las características de la nueva metodología, resultado de la combinación de las otras dos. De la parte de Scrum, se mantiene el hecho de llevar a cabo revisiones y añadir mejoras cuando finalizan los sprints, en las reuniones convenientes, las labores toman prevalencia según nivel de dificultad y de requerimiento, y se consensuarán aspectos relacionados con la culminación de tareas a modo de asegurar que todos los miembros del equipo comprendan de la misma manera y se eviten malentendidos. Por otro lado, de la parte de Kanban, las labores se encuentran disponibles en una lista conjunta, y cada miembro del equipo accede a ella para comenzar a realizar cualquiera de esas labores; además, la carga de trabajo vigente será limitada a modo de no experimentar aglomeraciones, y el sistema de gestión del trabajo se ajustará al uso de un tablero con ítems a realizar, los cuales se encontrarán en varios estados atendiendo al progreso que se lleve a cabo. Por último, respecto a características propias de Scrumban, encontramos una igualdad dentro del grupo de trabajo sin la figura definida de un jefe de equipo, algo que da la posibilidad a todos los miembros de deliberar y llevar a cabo modificaciones; además, Scrumban no impone el establecer una fecha de finalización de un proyecto, algo positivo para cuando nos encontramos ante dificultades para estimaciones de duración por cualquier motivo.

Finalmente, analizamos las virtudes e inconvenientes del uso de Scrumban. Por una parte, es beneficioso a la hora de adaptar proyectos que han comenzado a llevarse a cabo sin ajustarse a ninguna metodología de coordinación de proyectos, además resulta interesante, como hemos indicado anteriormente, para trabajar cuando se necesita flexibilidad en cuanto a modificaciones, algo que encontramos ante trabajos de grandes dimensiones o de larga duración; también ofrece como punto positivo la opción de introducir modificaciones y llevar a cabo deliberaciones en el momento que sean necesarios, sin tener que esperar a reuniones preestablecidas, y

por último el grado de libertad que brinda a los miembros de un grupo al no existir tantas restricciones como en otras metodologías. Sin embargo, como es lógico, presenta también aspectos en contra. Si bien es cierto que Scrumban ofrece mucha más capacidad de autogestión a los equipos, esto puede provocar también situaciones de desorden y conflictos; además, debido a que es un método bastante reciente en el tiempo, carece de referencias sobre las que guiarse y estándares a los que ajustarse. Por tanto, consideramos que Scrunban es una metodología que ofrece multitud de aspectos positivos y beneficiosos, al mismo tiempo que presenta inconvenientes a tener en cuenta, por lo que la decisión sobre qué método usar debe comenzar por realizar un análisis en busca de aspectos en los que el equipo de trabajo funciona correctamente, y aspectos en los que no se estiman resultados no tan satisfactorios. [26]

5. PRINCIPIOS DE DEVOPS, VENTAJAS E INCONVENIENTES

Cuando un equipo decide hacer uso de la metodología DevOps, debe previamente tener claros una serie de fundamentos sobre los que se esta se sustenta, que definen los objetivos que se pretenden alcanzar y la manera en la que llevar a cabo las labores de desarrollo para alcanzarlos. Por ello, existen varios principios que resumen la filosofía DevOps.

En primer lugar, es de vital importancia hacer girar todo el trabajo y lo que se está produciendo en torno al cliente, pues este será quien obtenga los resultados finales y haga uso de ellos. Es por ello que se debe acceder a recursos innovadores para brindar al cliente la mayor calidad del producto; además, el equipo debe estar abierto a cambios cuando sean necesarios, es decir, cuando se den situaciones en las que algo se puede mejorar, realizarlo de manera diferente o simplemente por motivos de funcionamiento no válido.

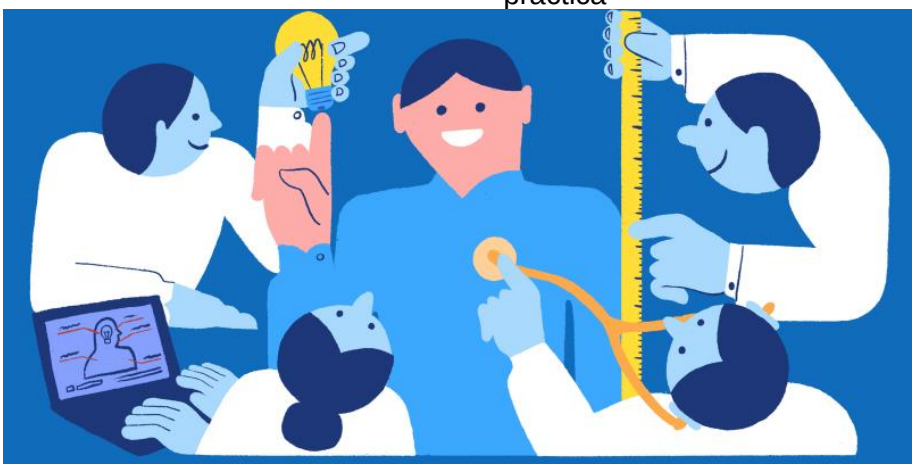

Ilustración 18. Concepto de acción centrada en el cliente [56]

Otro de los principios de DevOps trata sobre los miembros del equipo de desarrollo y su fiel compromiso con el proyecto desde su inicio hasta su finalización, asumiendo en todo momento una responsabilidad con las labores que se están llevando a cabo y con el producto que se está creando. Esto tiene una lógica bastante simple y fácil de comprender: un equipo DevOps busca la cooperación de todos sus miembros, es decir, no basta con que uno de ellos realice sus asignaciones y se desentienda de las de sus compañeros, pues la filosofía de DevOps persigue fomentar una buena comunicación y forma conjunta de trabajar entre los componentes del grupo de trabajo. De este modo, cuando uno de los miembros experimente conflictos durante el desarrollo de alguna de las partes, pueda encontrar un punto de apoyo por parte de los demás compañeros, algo que evitaría retrasos en las labores de trabajo y generaría mejores resultados.

Ilustración 19. Concepto de responsabilidad extremo a extremo [57]

Un aspecto bastante importante que la metodología presenta es la obligación de trabajar constantemente por progresar y perfeccionar las labores de desarrollo. Esto trata sobre una actitud perseverante por progresar para obtener siempre los mejores resultados, y para ello se debe realizar un análisis sobre el trabajo que se está llevando a cabo y de qué forma se consigue. Por ello, es fundamental que, durante el desarrollo, los equipos de trabajo se sientan en el derecho de poder experimentar fallos, para así adquirir experiencia, algo comprensible teniendo en cuenta que una de las fases del desarrollo de proyectos consiste en evaluar lo que se ha creado para comprobar que su funcionamiento sea correcto. Determinar qué aspectos deben ser modificados no suele resultar una tarea complicada, pero el hecho de hallar la mejor forma de solventar dichos aspectos puede resultar bastante complejo.

Ilustración 20. Concepto de mejora continua [58]

La automatización es otro de los pilares fundamentales de DevOps, como bien hemos visto en puntos anteriores. Si recordamos la definición de DevOps, veremos que uno de los objetivos que esta herramienta pretende alcanzar es una optimización del proceso de desarrollo en cuanto a tiempo, esfuerzo y costes. Aquí es donde las herramientas que hemos estudiado previamente juegan su papel, encargándose de llevar a cabo labores de trabajo de forma autónoma, logrando así reducir tiempos al máximo y haciendo más ligera la carga de trabajo del equipo a modo poder dedicar sus esfuerzos a otras tareas mientras que las herramientas cumplen su función. Incluso en ciertas ocasiones, automatizar procesos permite que se lleven a cabos tareas con alto nivel de complejidad, o incluso imposibles de realizar por medio de la mano de obra humana.

Ilustración 21. Concepto de automatización de procesos [59]

En DevOps, el equipo de desarrollo juega un papel fundamental, y la gestión interna del mismo es de vital importancia para un buen desempeño en las labores de trabajo. Uno de los aspectos primordiales de la filosofía de DevOps es la cooperación entre los miembros del grupo, así como el sentido de la incumbencia, es decir, hacerse cargo de todo lo que suceda durante el proceso de desarrollo. Además, como hemos

mencionado en otras ocasiones, los miembros del equipo poseen una cualificación variada, con destrezas aplicables a diferentes ámbitos, aportando y adquiriendo feedback entre ellos.

Ilustración 22. Concepto de equipos multidisciplinares y fusionados [60]

Cuando hablamos sobre el desarrollo de un proyecto para crear el producto que el cliente requiere, es importante ir más allá de lo que simplemente se nos está pidiendo. Es decir, antes de comenzar con las labores de trabajo, es fundamental analizar, no sólo lo que el cliente nos pide, sino el objetivo que debe satisfacer el producto que vamos a ofrecerle o lo que realmente este necesita. [27]

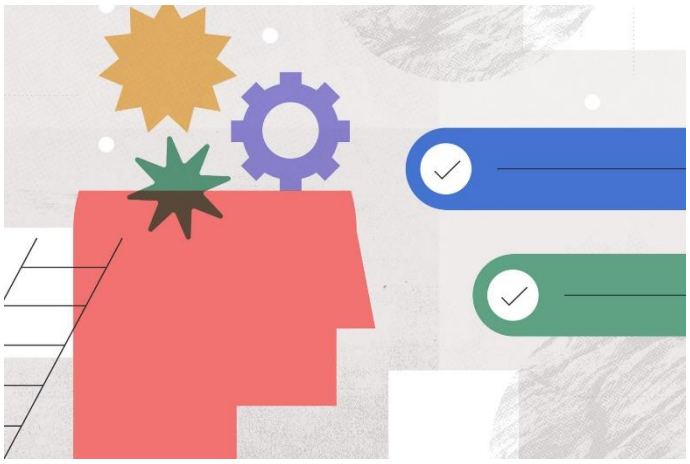

Ilustración 23. Concepto de creación en torno al objetivo final [61]

Por último, un aspecto muy importante de DevOps, aunque también resulte primordial en otras metodologías, está relacionado con los procesos de evaluación y testeo del trabajo que se está realizando, para poder analizar casuísticas de fallos para poder solventarlos y adquirir experiencia, con el objetivo de ofrecer resultados óptimos.

Ilustración 24. Concepto de procesos de testeo [62]

Como vemos, hacer uso de la metodología DevOps implica aceptar unos compromisos firmes con el proyecto sobre el que se está trabajando y con el cliente al que estamos ofreciendo un producto y/o un servicio. Esto aporta una serie de aspectos positivos que se reflejan tanto en las labores de desarrollo como en los resultados que se obtienen. Por tanto, vamos a analizar qué ventajas experimentan los equipos al adoptar esta metodología.

5.1. Ventajas

En primer lugar, DevOps ofrece **un ambiente de trabajo realmente fusionado**, pues como hemos comentado anteriormente, los miembros del equipo aportan variedad de habilidades como punto de apoyo entre ellos, además de mostrar

implicación por la totalidad del proyecto y no únicamente por las correspondientes asignaciones individuales, es decir, todos los miembros tendrán en cuenta las labores que se están llevando a cabo durante el desarrollo aunque no estén dentro de sus atribuciones, por lo que se creará un clima de trabajo mucho más participativo que ofrecerá mayor nivel de tranquilidad a los trabajadores, y esto conllevará a una obtención más rápida de resultados y con mayor valor.

Otra de las ventajas que ofrece DevOps es **una reducción del tiempo de desarrollo y gastos** a raíz de la integración de herramientas que permiten llevar a cabo procesos de forma autónoma, como hemos visto previamente. Además, aplicando esta técnica de autogestión de procesos, se experimenta también una reducción de fallos en el desarrollo del proyecto, al mismo tiempo que los períodos para corregir fallos se llevan a cabo con menor duración.

Además, **DevOps permite hacer frente de una mejor manera a los contratiempos y la aparición de trabajo que no entraba dentro de lo planeado**, algo que consigue con el hecho de establecer prioridad a las distintas tareas a realizar para poder gestionar mejor las labores de desarrollo dentro de una organización de las mismas. Por tanto, adaptar estas eventualidades dentro de la planificación a seguir que ya se había concebido resulta un proceso bastante asequible al hacer uso de DevOps, además de contribuir a la reducción de pérdidas ante posibles situaciones de incidencia.

Por otro lado, nos fijamos en la constante integración y entrega que caracterizan a esta metodología. Estos aspectos favorecen **una mayor destreza y adecuación del producto**, al experimentar constantes modificaciones, mejoras y nuevas versiones que se encargan de eliminar residuos de las anteriores. Además, esto permite ofrecer con más asiduidad resultados al cliente, que, a su vez, este podrá ir probando y valorando. De esta manera, este podrá sugerir a partir de la experiencia con las entregas del producto nuevas innovaciones o aspectos que modificar para perfeccionar la funcionalidad de lo que se le está ofreciendo. [28]

Por tanto, podemos considerar que DevOps resulta una metodología que ofrece ventajas en prácticamente todos los ámbitos dentro del desarrollo software, ya sea en

referencia al equipo, al ritmo de trabajo, a la calidad de los resultados y a la satisfacción del cliente, entre otros aspectos.

5.2. Inconvenientes

Ahora bien, existen también una serie de factores que no resultan tan favorables y se deben tener en cuenta antes de recurrir a esta metodología, pues, aunque presente numerosos aspectos positivos, también nos encontraremos con ciertas complicaciones.

En primer lugar, debemos tener en cuenta que es una metodología que necesita períodos de transición donde se modifican numerosos aspectos, por lo que **requiere de bastante paciencia y perseverancia**. Además, atendiendo al principal concepto que define DevOps, se produce una combinación de labores de desarrollo con labores de operaciones, por lo que suele resultar complicado acostumbrarse a lidiar con ambas al mismo tiempo.

Anteriormente hemos analizado las herramientas de las que se pueden hacer uso dentro de la metodología, que son bastantes. Por tanto, el hecho de **elegir cuáles van a ser usadas para nuestro proyecto** puede resultar también un proceso complejo.

Por último, sabemos que uno de los pilares fundamentales de la filosofía de DevOps es el trabajo en equipo de forma robusta. Aun así, y siendo realistas, estos pueden experimentar **problemas de coordinación y cooperación**, ya sea por opiniones diferentes, malos entendidos o muchos más aspectos discordantes, pues cada persona tiene su propio punto de vista y tratar de compaginarlo con el de los demás compañeros puede resultar complicado en numerosas ocasiones. [29]

6. EQUIPO DEVOPS: ROLES

Cuando analizamos los diferentes aspectos de la metodología DevOps, o generalizando, en cualquier metodología de desarrollo, comprobamos que uno de los

aspectos más importantes, incluso se podría considerar el más imprescindible, resulta ser el equipo de trabajo. En este caso, nos encontramos con que existen una serie de papeles que adoptan sus miembros, los cuales definen claramente sus funciones dentro del equipo. Vamos a ver cuáles son los principales roles que nos podemos encontrar dentro de un equipo DevOps y en qué consisten. [30]

6.1. DevOps Evangelist

Considerado como el máximo responsable del grupo y la figura líder, tiene como principal objetivo la correcta coordinación del equipo en sus labores de trabajo. Su labor pasa por transmitir toda la filosofía de trabajo de la metodología a los demás compañeros y aportarles una motivación de cara a desempeñar sus funciones, además de asegurarse que todos están cualificados para cumplir con sus asignaciones y que el desarrollo del proyecto se está llevando a cabo correctamente y según la planificación.

Ilustración 25. Concepto de DevOps Evangelist [63]

Además, es una figura vital en el momento en el que una organización decide hacer una transición para adoptar la metodología DevOps como marco de trabajo, pues este se encargará de gestionar dicha transición.

Por tanto, su finalidad es analizar y gestionar los aspectos que contribuyen a obtener resultados satisfactorios.

6.2. Release Manager

El administrador de entregas juega un papel fundamental dentro de esta metodología, en referencia a uno de los aspectos más destacados de la filosofía de DevOps, basado en disponibilidad de versiones del producto en menos tiempo.

Su objetivo pasa por supervisar y manejar el transcurso de las labores de trabajo hasta el momento en el que se producen lanzamientos de nuevas versiones, pasando por conocer las distintas herramientas que se utilizarán en el proyecto y mostrar destreza a la hora de aplicarlas en la práctica.

Ilustración 26. Concepto de Release Manager [64]

Un administrador DevOps se diferencia de uno tradicional en el hecho de que se implica de forma más amplia y en partes más relacionadas con el ámbito técnico.

6.3. Automation Architect

El experto en automatización, conocido en ciertas ocasiones también como ingeniero DevOps, es la figura dentro del proyecto que lleva a cabo acciones y procesos de optimización. Esto se consigue analizando y decidiendo qué elementos y procesos se pueden automatizar y escogiendo las herramientas más apropiadas para ello y que mejor se adapten al trabajo que se está realizando.

Ilustración 27. Concepto de Automation Architect [65]

Como acabamos de indicar, este rol también se asocia con el de un ingeniero DevOps, el cual se define como aquel que posee una cualificación muy variada que cubra todos los ámbitos pertenecientes al proceso de desarrollo de un proyecto software. Además, debe ser capaz de trabajar entre los distintos equipos de la organización y conocer plenamente la metodología DevOps, en qué consiste y cómo ponerla en práctica de manera correcta, así como las herramientas disponibles y cómo hacer uso de ellas. Ahora bien, en este caso, la figura del ingeniero DevOps está más enfocada a la automatización de tareas de desarrollo, para así hacer de este un proceso más ágil y que se lleve a cabo en el menor tiempo posible.

6.4. Software Developer and Tester

Es aquel que participa en las tareas relacionadas con la construcción del proyecto mediante la interpretación de las asignaciones que le son impuestas, además de aplicar una serie de labores de testeo posteriormente y evaluar el funcionamiento de lo que se ha creado.

Ilustración 28. Concepto de Software Developer and Tester [66]

Este rol, dentro de DevOps, es innovador respecto a la figura de un desarrollador software tradicional, pues como podemos ver, no sólo se limita a producir, sino a evaluar y chequear lo que produce, por lo que se ve ampliado su abanico de atribuciones y responsabilidades. Además, se encarga de introducir innovaciones y del tratamiento de fallos.

6.5. Quality Assurance Engineer

El experto en garantía de calidad es aquel cuyo objetivo se basa en asegurar que el producto el cual se va a entregar al cliente tenga el máximo valor posible en

compromiso con las necesidades que este presenta. Para ello, define unos estándares a los que el producto debe ajustarse. Estos estándares de calidad comprenden una serie de requisitos y restricciones que consigan que el producto lleve a cabo las funciones por las cuales ha sido creado y poder así ofrecer al cliente los mejores resultados.

Ilustración 29. Concepto de Quality Assurance Engineer [67]

Por así decirlo, esta figura se encarga de trasmitir a los desarrolladores el comportamiento que debe tener y los resultados que debe aportar el producto sobre el que se ha trabajado y que va a ser evaluado y testeado por ellos, además de decidir cuáles son las herramientas más indicadas para llevar a cabo los procesos de testeo.

Dentro de este rol, podemos ver a un experto en garantía de calidad de dos maneras diferentes atendiendo al papel que desempeña. Por un lado, tenemos a un ingeniero QA en un ámbito de asesoría, en el que dicho experto colabora con los ingenieros de desarrollo para que estos mejoren sus técnicas de evaluación y testeo, además de aportarles conocimiento a la hora de aprender a evaluar correctamente la calidad de lo que se está produciendo. El segundo caso está relacionado con un

ámbito basado en tácticas en las que el ingeniero de calidad delibera sobre los modelos a seguir para que el producto adquiera el valor deseado.

6.6. Security and Compliance Engineer

Por último, nos trasladamos a un ámbito bastante importante dentro del mundo de la informática y las tecnologías de la información. La seguridad es un aspecto fundamental que nunca debe caer en el olvido o considerar como secundario. Es por ello que esta figura de ingeniero se encarga de mantener en todo momento un ambiente de trabajo protegido contra posibles amenazas y daños.

Ilustración 30. Concepto de Software and Compliance Engineer [68]

Frecuentemente, la seguridad constituye un factor que se suele tener en cuenta y establecer cuando el proceso de producción ha finalizado o se encuentra en las últimas etapas, algo no muy acertado. Por eso, DevOps pretende que los aspectos de seguridad sean trabajados desde que comienzan las labores de trabajo.

7. DEVOPS EN COMPARACIÓN CON OTRAS METODOLOGÍAS

Con la realización de este proyecto, analizamos y comprendemos qué es la metodología DevOps, cuáles son sus principales características y cómo se puede llevar a la práctica. Aun así, tenemos que saber que existen otras metodologías a las que podemos recurrir y que vamos a ver y comparar con DevOps, para conocer qué aspectos son similares y en cuáles difieren.

7.1. SRE (Site Reliability Engineering)

Esta metodología, conocida como ingeniería de fiabilidad del sitio, tiene como objetivo la creación de productos y servicios software que presenten más confianza e incrementos cada vez más rápidos. Su origen se remonta a 2003, cuando Benjamin Treynor se puso al mando de un grupo de trabajo que pretendía hacer de Google un elemento de fiabilidad, repartiendo las labores del grupo en dos particiones iguales, de las cuáles una estaría enfocada al ámbito de desarrollo y otra al de operaciones.

Haciendo una comparativa con DevOps, vemos que son dos metodologías que comparten muchos aspectos. Ambas promueven la cooperación dentro del equipo de trabajo y pretenden acabar con el aislamiento que existe entre la parte de desarrollo y la de operaciones. Ahora bien, sí que es cierto que SRE muestra un especial interés por la fiabilidad reflejada en los productos que desarrolla; además, aunque se pretenda establecer una equidad entre ambas partes mencionadas anteriormente, la de desarrollo adopta un papel más enfocado al liderazgo sobre la de operaciones. Por su parte, y como bien sabemos, DevOps se decanta por la velocidad en el proceso de desarrollo, además de implicarse en la automatización y aplicarla dentro de este en la medida de lo posible. [31]

7.2. Agile

Agile es una metodología de dirección de procesos de desarrollo software, la cual se sustenta en una serie de principios. En primer lugar, se prioriza la figura humana por encima de los elementos de carácter técnico, además de mostrar especial interés por un correcto funcionamiento del producto sobre el que se está trabajando y

por las necesidades del cliente, y pretende que el equipo de desarrollo esté preparado y sepa reaccionar y actuar en situaciones en las que se produzcan modificaciones no contempladas dentro de la planificación del trabajo establecida.

La aparición de la metodología Agile se sitúa en 2001, aunque es cierto que sus orígenes pueden provenir desde incluso los años 60. El motivo de su creación se relaciona con la necesidad de un marco de trabajo que permitiese la introducción de modificaciones en el trabajo evitando, en la medida de lo posible, que afectasen al curso normal del trabajo o influyesen económicamente en las estimaciones inicialmente realizadas. [32]

Si comparamos Agile con DevOps, debemos tener en cuenta que son dos metodologías muy parecidas entre sí. Es más, a veces incluso se permite y es conveniente hacer un uso combinado de las dos en lugar de decantarse por una o por otra. Aun así, muestran pequeñas diferencias entre ellas. En primer lugar, nos encontramos con el objetivo principal de Agile, el cual se centra primordialmente en el concepto de las modificaciones de forma reiterada y en reforzar el vínculo con el cliente, al mismo tiempo que DevOps busca reforzar el vínculo entre desarrollo y operaciones y muestra una preferencia por los continuos incrementos del producto. En cuanto al equipo de trabajo, Agile se caracteriza por grupos de menor tamaño al contrario que DevOps. Por último, Agile presta una especial atención a aspectos técnicos del producto, mientras que DevOps se centra en la puesta a disponibilidad del cliente. [33]

8. ENCUESTA SOBRE DEVOPS: NIVELES DE ACEPTACIÓN DE LA METODOLOGÍA Y SUS PRINCIPALES ASPECTOS POR PARTE DE LA SOCIEDAD

Tal y como se comentó en el apartado de introducción, DevOps supone un concepto generalmente desconocido por parte de la sociedad. Sí que es cierto que existe gran parte de la población que no tiene prácticamente relación o mayor interés por el mundo de la informática y el desarrollo software, por lo que tienden a

desconocer otros aspectos, como metodologías, herramientas o términos y conceptos propios de dicho ámbito, entre otras cosas.

Por esta razón, se ha llevado a cabo una encuesta dirigida a la población en general, sin distinguir a razón de edades ni conocimientos informáticos, para conocer realmente en qué grado es conocido el término DevOps, la opinión que recibe acerca de sus principales características y hasta qué punto sería la metodología elegida como marco de trabajo. Para ello, se han lanzado 16 preguntas reflejadas en un cuestionario (ver apartado 8.1) gestionado por la herramienta Google Forms, de las cuáles las tres primeras nos aportan información sobre la parte demográfica de la población que ha participado en el cuestionario, y las trece restantes tratan sobre DevOps y los principios sobre los que se sustenta.

Esta encuesta ha sido diseñada teniendo en cuenta que, al estar dirigida a participantes sin distinción ninguna, debe ser comprendida por la totalidad de los encuestados. Es decir, hay preguntas que, si se formulan totalmente enfocadas a la metodología y al desarrollo de software, probablemente una parte de las personas que contribuyan a la realización del estudio no será capaz de responderlas, ya que puede ser que no comprendan su significado, o simplemente resulte complicado ponerse en situación para deliberar correctamente. Por tanto, en algunos casos, se han utilizado una serie de situaciones imaginarias, similares a escenarios de desarrollo software, para lograr que la gente pueda comprender el sentido de las cuestiones y responder con total capacidad.

Para alcanzar niveles aceptables de participación, el cuestionario ha sido distribuido por las redes sociales WhatsApp e Instagram, y se han obtenido un total de 95 respuestas, las cuáles cada una pertenece a un participante distinto, pues se ha restringido el formulario para que cada persona pueda completarlo una única vez. Además, los datos han sido recopilados de forma anónima, únicamente distinguiendo resultado por edad, sexo y ocupación para comprobar los niveles de distribución de la información recolectada.

8.1. Cuestiones planteadas en la encuesta

Este es el listado de las preguntas incluidas en la encuesta:

- ¿Tienes conocimientos de informática y/o te interesa el ámbito del desarrollo de software?
- ¿Conoces la metodología DevOps o has oído hablar de ella?
- De cara a un empleo, ¿crees que el trabajo en equipo es un aspecto positivo?
- Teniendo en cuenta que un equipo de trabajo puede estar formado por varios miembros, los cuales pueden tener puntos de vista bastante diferentes y optar por formas diferentes de realizar las labores de trabajo, ¿qué grado de facilidad crees que existe a la hora de establecer coordinación para que todos los miembros trabajen en un ambiente cómodo y fusionado?
- Suponemos una situación imaginaria y simple, ajena al ámbito de la informática: estamos trabajando en una empresa dedicada a la producción alimentaria, en la cual existen varios tipos de trabajadores según las labores que desempeñan: uno realiza la parte de cultivo y cosecha, otro se encarga del procesamiento, otro lleva a cabo los controles de calidad y otro se encarga del envasado. Elige la opción que consideres más adecuada: a) Es mejor que todos los miembros tengan conocimientos algo más genéricos sobre todas las labores que se llevan a cabo en la empresa. b) Es mejor que cada miembro esté especializado en su labor, aunque no sea capaz de desempeñar ninguna de las labores ajenas a su persona.
- Con respecto a las responsabilidades del trabajador, elige la opción más adecuada: a) Considero que es mejor que cada trabajador se haga responsable de sus asignaciones, desentendiéndose de las atribuciones de los demás. b) Considero que es mejor que los trabajadores compartan responsabilidades.
- Suponemos, dentro de la misma empresa que hemos puesto como ejemplo, que en la parte de procesamiento surge un conflicto y se

paraliza el proceso de producción completo hasta encontrar una solución. ¿En qué ámbito de los anteriores crees que sería mejor reaccionar ante esta situación?

- ¿Consideras ahora que la responsabilidad compartida es un factor beneficioso dentro de un equipo de trabajo?
- ¿En qué grado crees que una empresa debe tener en cuenta a un cliente que requiera de sus productos o servicios?
- ¿Crees que los resultados pueden mejorar cuando la producción gira en torno al cliente?
- ¿Crees que los equipos de trabajo se pueden permitir cometer fallos durante el proceso de producción?
- ¿Crees que la automatización de procesos es un aspecto beneficioso para una empresa?
- Por último, vamos a analizar un ejemplo real. WhatsApp, entre muchas otras aplicaciones, es una herramienta la cual ha ido evolucionando a lo largo de los años mediante la implementación de nuevas funcionalidades: confirmaciones de lectura, envío de archivos, opciones de llamada y videollamada, etc. Estas novedades no han sido incluidas en la aplicación al mismo tiempo, sino que se han ido desarrollando y poniendo a disposición de los usuarios progresivamente. Visto esto, elige la opción que consideres más adecuada: a) Prefiero una aplicación que incluya todas las funciones que ofrece en una sola entrega, aunque tenga que esperar más tiempo para que esté disponible. b) Prefiero una aplicación que no tenga muchas funcionalidades desde el principio, pero que funcione correctamente, la cual vaya adquiriendo novedades de forma progresiva y se incluyan cada vez más funciones en pequeños períodos de tiempo.

8.2. Población encuestada

Como hemos indicado anteriormente, esta encuesta pretendía ser dirigida a cualquier grupo de población, sin tener en cuenta generaciones ni conocimientos. Esta es la distribución que siguen las respuestas obtenidas según la edad de los participantes:

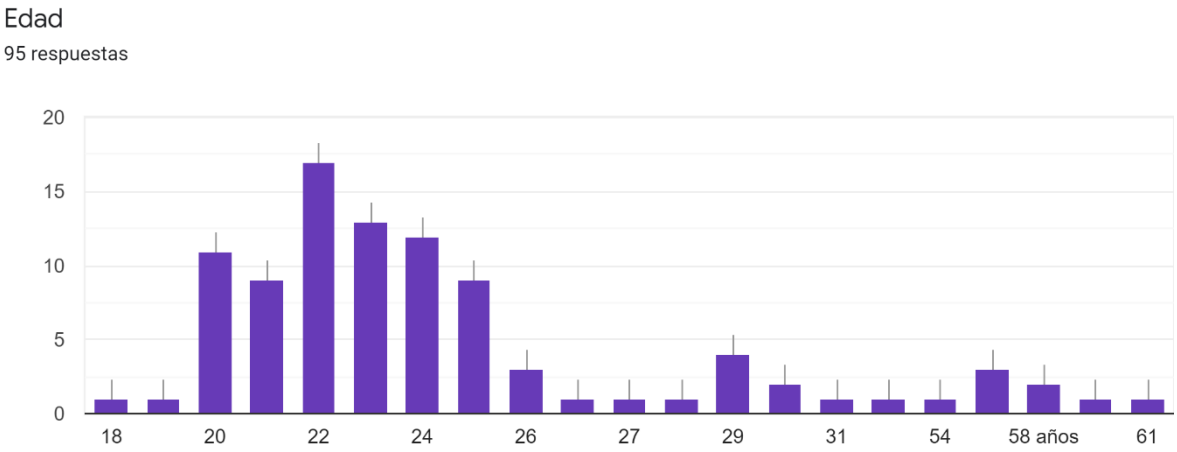

Ilustración 31. Gráfico de población encuestada según edad [69]

Como podemos apreciar, existe una gran variación entre las personas encuestadas, aunque podemos apreciar que los datos obtenidos son mayoritariamente por parte de generaciones jóvenes, pues prácticamente el 80% de las respuestas obtenida pertenecen a participantes de menos de 25 años. La razón de esta distribución dentro de la población se debe, en gran parte, al uso de las nuevas tecnologías y las redes sociales, algo más usual en los jóvenes; aun así, estos últimos años, se ha experimentado un incremento de uso de las mismas por parte de las generaciones más adultas. Esto puede reflejar en los resultados una mayor relación de los participantes con el mundo de la informática, pues las nuevas generaciones cada vez están más familiarizadas con el uso de tecnologías de la información.

Por otro lado, se ha obtenido los porcentajes correspondientes al género de los participantes:

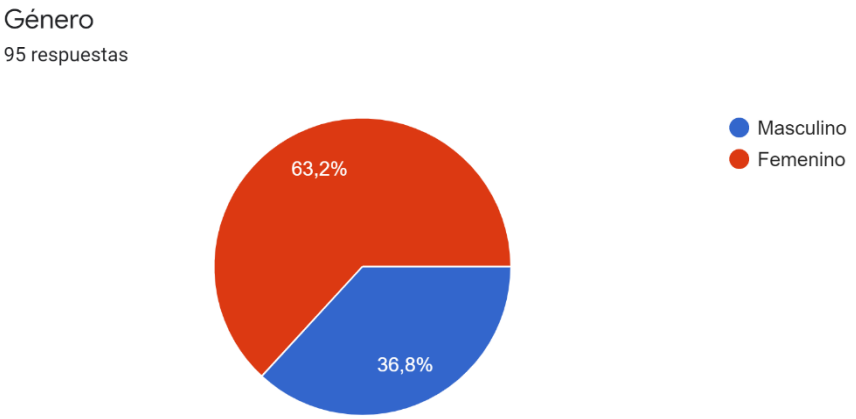

Ilustración 32. Gráfico de población encuestada según género [70]

Y, por último, conocemos también la ocupación de los encuestados de forma general:

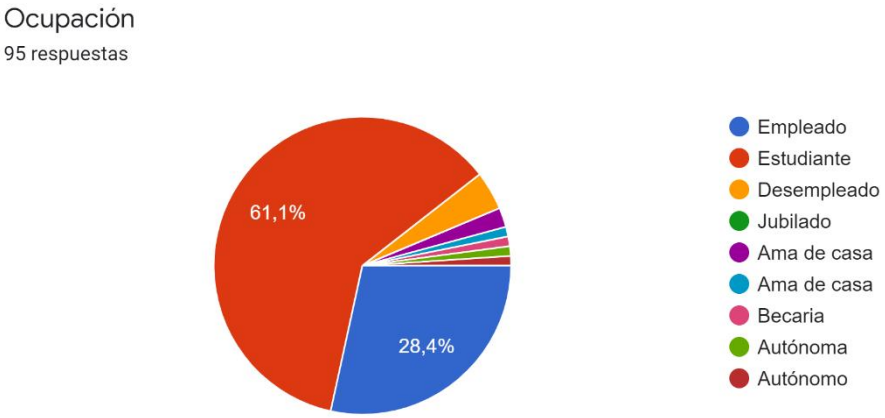

Ilustración 33. Gráfico de población encuestada según ocupación [71]

Como podemos apreciar, la mayoría de los participantes son estudiantes, algo comprensible si miramos los datos estadísticos relacionados con la edad. Aun así, existe una parte de los encuestados que se encuentra en el mundo laboral, la cual puede resultar interesante al conocer desde primera línea aspectos de DevOps, como pueden ser el trabajo en equipo o las responsabilidades del trabajador.

8.3. Interpretación de resultados y obtención de conclusiones

A partir de las preguntas que se han planteado en la encuesta, se han obtenido una serie de resultados que nos informan acerca de lo conocida que es la metodología DevOps en la sociedad y el grado de aceptabilidad que se muestra antes sus aspectos más importantes.

En primer lugar, se ha preguntado a los participantes sobre su interés o conocimientos relacionados con el mundo de la informática y el desarrollo software.

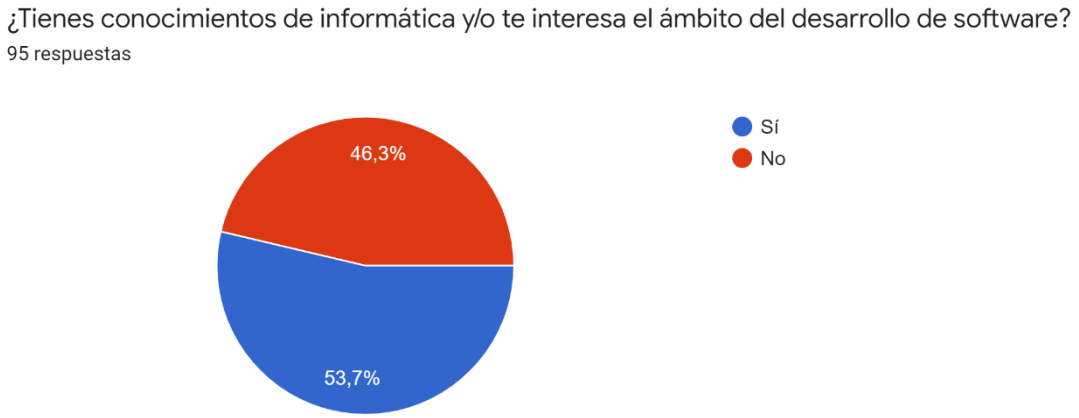

Ilustración 34. Gráfico de población encuestada según relación con la informática [72]

Como podemos ver, prácticamente la mitad de los encuestados han afirmado estar vinculados a este entorno. Por tanto, y de cara a extraer conclusiones posteriormente, podremos evaluar lo popular que es la metodología dentro del ámbito del que proviene.

Siguiendo con el análisis, se ha preguntado de forma directa el conocimiento, sea profundo o no, sobre DevOps, para conocer su popularidad.

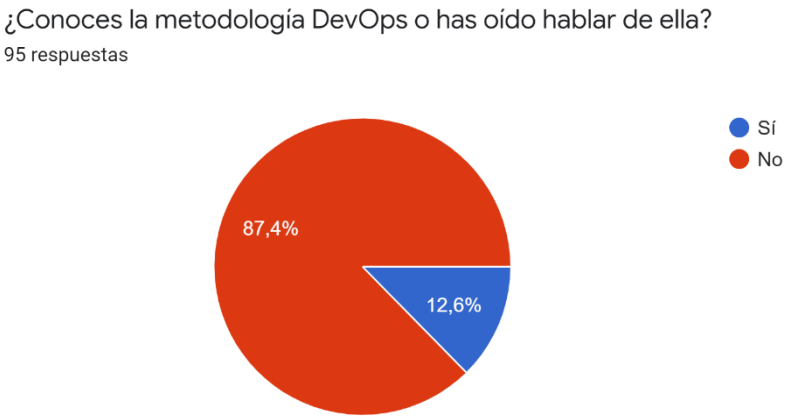

Ilustración 35. Gráfico de población encuestada según si conoce DevOps [73]

Aquí podemos comprobar que, prácticamente, es una metodología que no es conocida entre los encuestados. Debemos tener en cuenta que estos resultados aún no reflejan una distinción entre gente relacionada con el mundo de las TI y gente ajena a ello, algo que veremos más adelante en un apartado de conclusiones. Los motivos de desconocimiento pueden estar relacionados con la reciente aparición de esta metodología, pues apenas hace 15 años de este suceso, o bien podría ser por un desinterés general sobre este mundo, lo cual investigaremos atendiendo a la anterior pregunta. Incluso podría deberse a una simple falta de información sobre metodologías disponibles para el desarrollo software.

Analizando estas dos preguntas en conjunto, para averiguar qué porcentaje de los participantes que conoce la metodología mantiene relación con el mundo del software, obtenemos los siguientes resultados: de las 95 personas que han realizado la encuesta, 51 (53.7%) afirman estar vinculados o, al menos, mostrar interés por el ámbito de la informática. Ahora bien, de esos 51, tan sólo 11 (un 11.6% del total de participantes) conocen DevOps, y de los 44 restantes ajenos al ámbito del software, tan sólo 1 conoce la metodología. Por tanto, ahora sí podemos afirmar que en general, el método DevOps es bastante desconocido en la sociedad.

Uno de los aspectos sobre los que se ha pedido opinión popular es el trabajo en equipo, uno de los pilares fundamentales de DevOps.

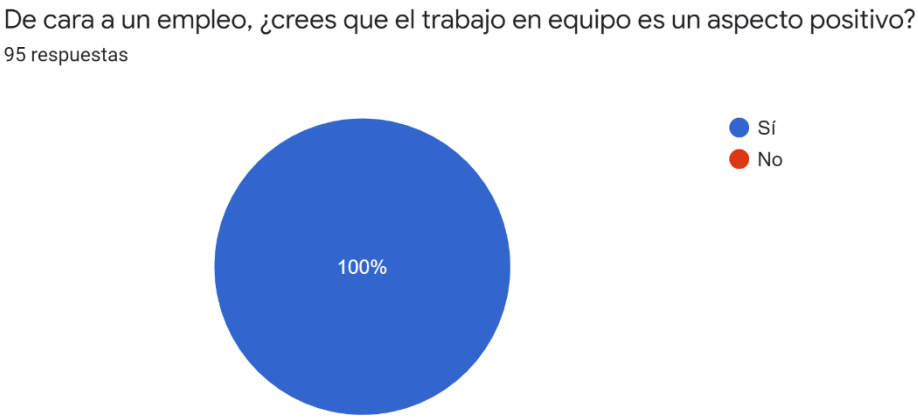

Ilustración 36. Gráfico de opinión del trabajo en equipo [74]

Nos encontramos ante una situación de consenso, en la que se demuestra que el trabajo en equipo es considerado un aspecto positivo generalmente. Además, debemos tener en cuenta que no solamente existen grupos de trabajo en un ámbito laboral, pues en un escenario estudiantil, los alumnos pueden requerir de formar parte de un equipo para la realización de proyectos pertenecientes a las diferentes asignaturas de su formación y, como bien sabemos, la mayor parte de los encuestados son estudiantes. Por tanto, podemos considerar que en general, se han obtenido experiencias positivas al poner en práctica el trabajo en equipo.

Continuando con el análisis sobre el trabajo en equipo, se ha planteado una visión sobre el mismo, en la que los encuestados se han visto inmersos en una situación real que se da en los grupos de trabajo: como bien sabemos, estos están formados por una serie de miembros, los cuales pueden tener opiniones, formas de trabajar y ritmos de desarrollar sus labores muy dispares, lo cual puede suponer un obstáculo a la hora de establecer una coordinación y un clima cómodo de trabajo.

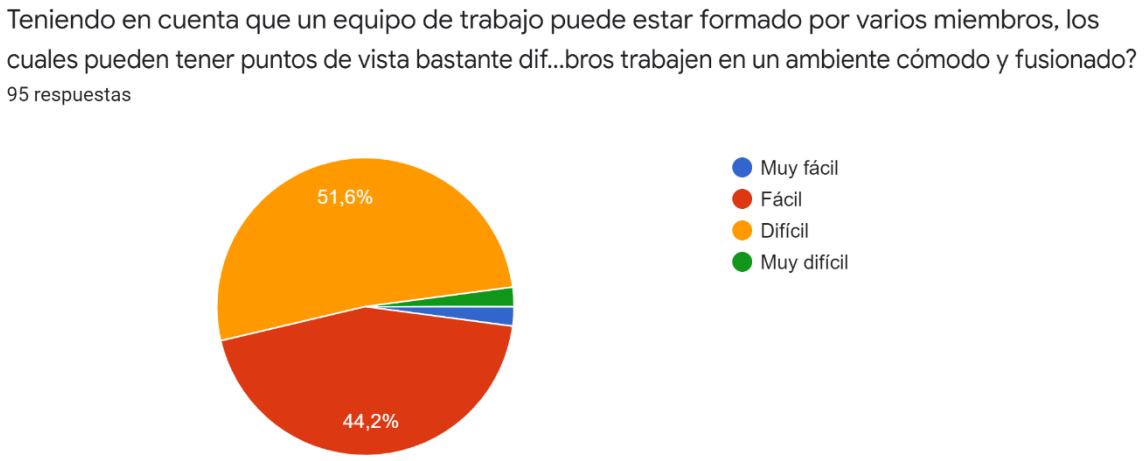

Ilustración 37. Gráfico de opinión sobre equipos fusionados [75]

Como vemos, las opiniones sobre este aspecto están bastante equilibradas, aunque comprobamos que la gente se decanta ligeramente por atribuir un carácter de dificultad al hecho de poner de acuerdo a todos los miembros de un equipo. Aun así, prácticamente nadie lo considera ni extremadamente fácil, ni tampoco imposible de conseguir.

Otro de las preguntas que se han formulado está directamente relacionada con las responsabilidades de los miembros del equipo de desarrollo. Para tratar de poner en situación a todos los encuestados de una forma más comprensible, se ha planteado un escenario basado en las labores de trabajo dentro de una empresa de producción alimentaria, en la que existen varias secciones con puestos de trabajo y asignaciones diferentes.

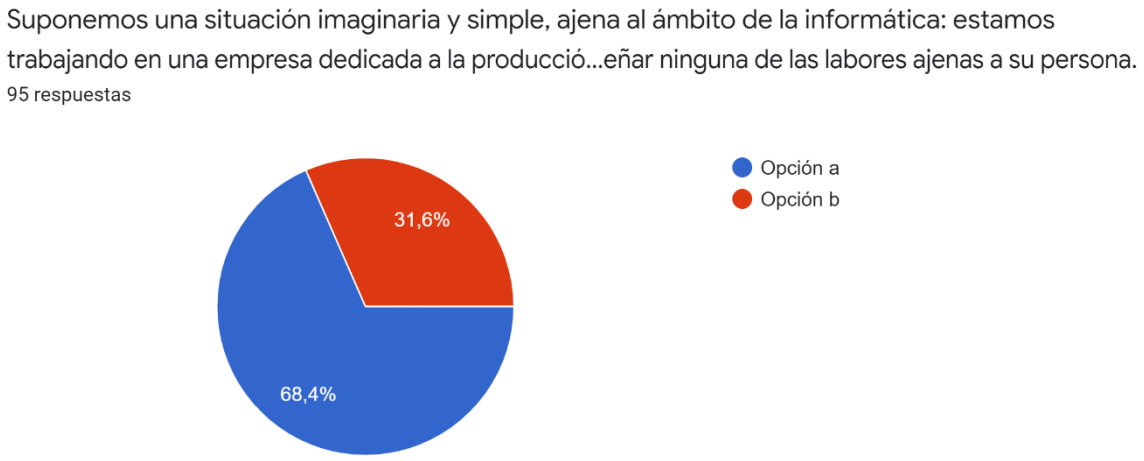

Ilustración 38. Gráfico de opinión sobre equipos multidisciplinares [76]

Queremos saber qué opina la gente sobre la formación y el nivel de especialización que poseen los miembros del equipo: comprobamos que se aprecia una clara preferencia sobre equipos multidisciplinares capaces de desarrollar varias funciones, algo que persigue la metodología DevOps.

Por otro lado, existe otro aspecto dentro de la metodología relacionado directamente con las asignaciones de los miembros del equipo. Como hemos visto en el apartado 5, DevOps impone un marco de trabajo caracterizado por una responsabilidad de sus trabajadores, no solo por las tareas de las cuáles se encargan cada uno individualmente, sino por las demás también.

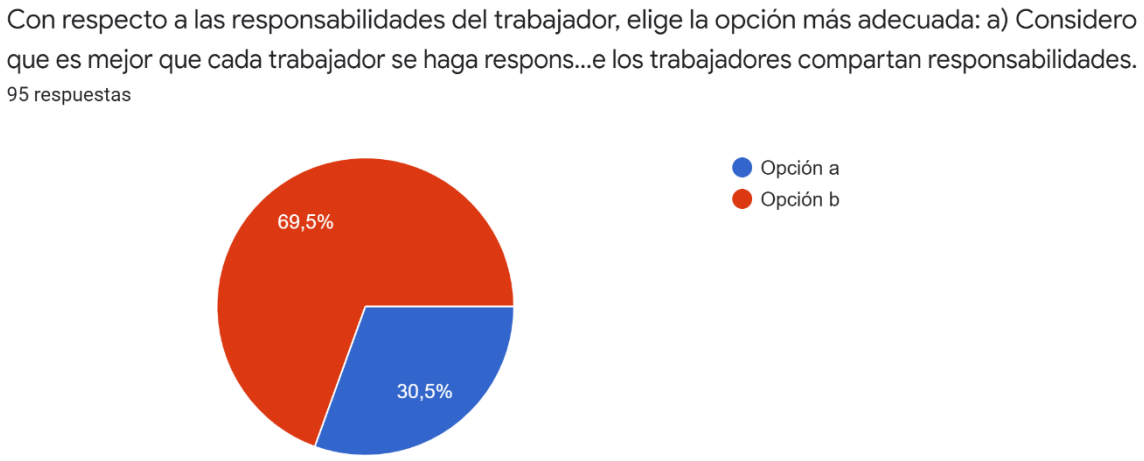

Ilustración 39. Gráfico de opinión a priori sobre responsabilidad compartida [77]

En respuesta a esta cuestión, obtenemos resultados que reflejan una preferencia por un ambiente de responsabilidades compartidas, en las que todos los miembros del equipo muestran su compromiso con la totalidad de las labores de trabajo, prácticamente un 70% de los encuestados apoyan esta opción como la más acertada.

Además, haciendo uso del ejemplo anterior, se pone a los participantes ante una situación de conflicto en la empresa que afecta directamente a los procesos de producción y, de esta forma, se lanza una pregunta para saber si la opción de responsabilidades en común sería la más acertada para poder tratar lo más rápido posible dicho problema.

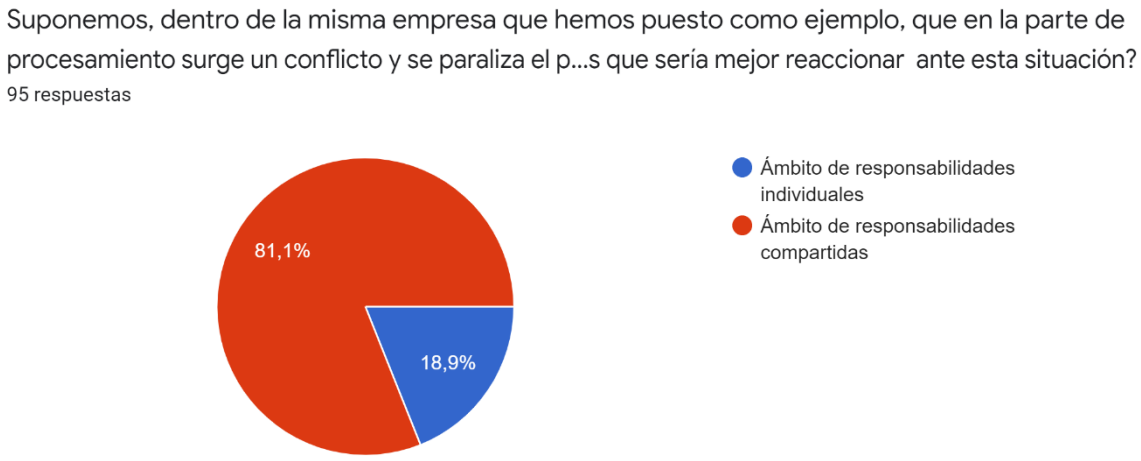

Ilustración 40. Gráfico de opinión sobre situación real según responsabilidad compartida [78]

Comprobamos que, evidentemente, de acuerdo con los resultados de la pregunta anterior, la gente opina que un escenario de responsabilidades compartidas resultaría más adecuado a la hora de actuar para resolver el conflicto y retomar el ritmo de trabajo normal. Como podemos ver, existen más votos a favor de dicho escenario una vez formulada esta cuestión: esto puede deberse al simple hecho de hacer que los participantes imaginen, de una manera más realista, qué podría ocurrir ante la situación problemática que se les ha planteado, lo que puede ayudarles a decantarse por una opción u otra de una manera más clara. Analizamos los datos para ver qué ha cambiado: con respecto a la pregunta anterior, en la que prácticamente el 70% de los encuestados votaron a favor de un escenario de responsabilidades compartidas, vemos que, de los 29 que votaron en preferencia a responsabilidades individualizadas, 19 han votado en esta pregunta a favor de una situación de responsabilidades conjuntas para resolver el conflicto; en cambio, de los 66 que preferían responsabilidades compartidas, 8 eligen un escenario de responsabilidades individuales para solucionar el problema. Por tanto, podemos decir que en general, los encuestados se han decantado algo más por la opción de compartir responsabilidades una vez ha sido planteado el ejemplo.

Por último, en relación a las responsabilidades compartidas, se ha vuelto a preguntar a los encuestados sobre si, a raíz de este ejemplo de situación, su opinión se ha visto condicionada en relación a este aspecto.

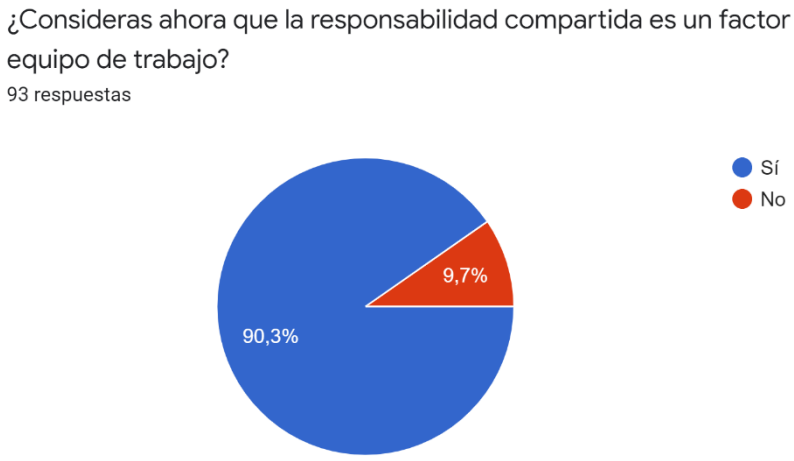

Ilustración 41. Gráfico de opinión a posteriori sobre responsabilidad compartida [79]

Aquí podemos ver claramente cómo se ha producido un cambio de opiniones respecto al principio que estamos analizando sobre DevOps, donde claramente se aprecia que tan sólo un 10% de ellos sigue considerando que prefiere individualidad en las labores de trabajo. Si comparamos los resultados de esta pregunta con la que inicialmente se propuso sobre este concepto, teniendo en cuenta que hemos planteado una situación similar a lo que ocurriría en la realidad, podemos ver que, de los 29 que en un principio mostraron preferencia por la individualidad de responsabilidades, 21 se decantan por la otra opción con respecto a este concepto tras proponer una situación imaginaria como ejemplo, lo que supone un 75% aproximadamente de individuos que han cambiado de idea hacia la responsabilidad compartida.

Otro de los aspectos sobre los cuales se ha querido conocer la opinión popular está relacionado con el cliente, una figura que la filosofía DevOps considera de vital importancia de cara al proceso de desarrollo.

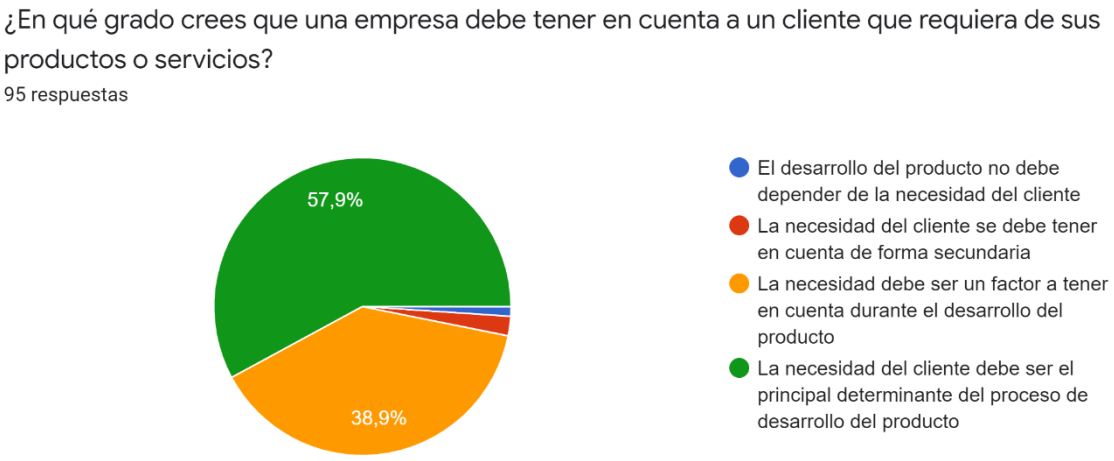

Ilustración 42. Gráfico de opinión de importancia sobre acción centrada en el cliente [80]

Los resultados reflejan, prácticamente, una unanimidad de opiniones que afirman que el cliente es un factor de importancia, en mayor o menor medida, cuando una empresa lleva a cabo el desarrollo de productos y servicios. De hecho, destaca una preferencia por un elevado nivel de consideración que se debe mostrar por quienes van a recibir y utilizar lo que la organización produzca.

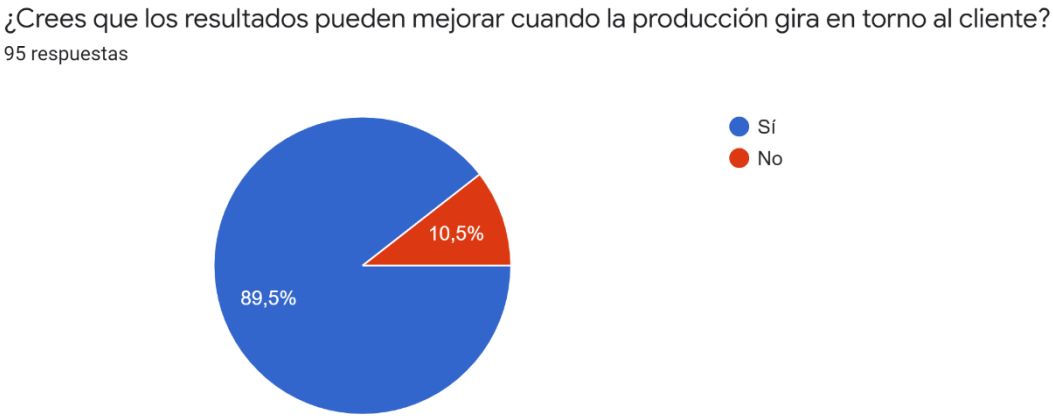

Ilustración 43. Gráfico de opinión de eficacia sobre acción centrada en el cliente [81]

Es por ello, por lo que en la siguiente pregunta se aprecia cómo los encuestados consideran que el establecer la figura del cliente como central dentro del proceso de desarrollo resulta favorable para los resultados que se obtienen a partir de ello.

Si recordamos lo visto en el apartado 5 sobre los que se sustenta DevOps, es muy importante hacer que los equipos de trabajo no se sientan presionados, por lo que se debe contemplar y asumir que puedan darse situaciones en las que se cometan fallos.

Ilustración 44. Gráfico de opinión sobre posibilidad de fallar durante la producción [82]

Esta vez, las opiniones relacionadas con este aspecto se encuentran mucho más divididas, donde prácticamente la mitad de los encuestados comprenden que, durante el desarrollo, el grupo de trabajo pueda fallar en ciertas ocasiones.

Continuando con los aspectos sobre los cuáles se han analizado opiniones, nos encontramos con la automatización de procesos, algo que caracteriza a DevOps en gran parte.

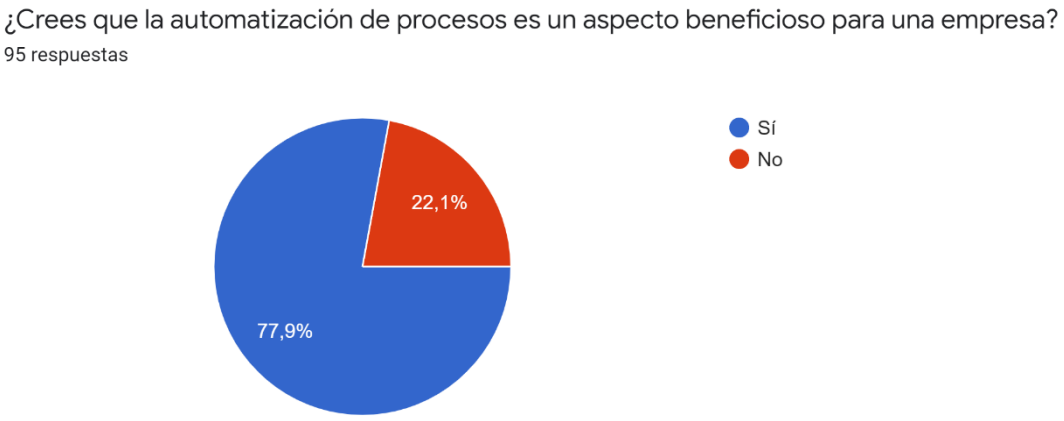

Ilustración 45. Gráfico de opinión sobre automatización de procesos [83]

Se ha preguntado, de forma directa, cuál es la opinión general sobre dicho concepto en relación a si constituye un factor positivo para una empresa, a lo que algo más de un 75% de personas han contestado afirmando esto.

Para finalizar la encuesta, se ha analizado un último aspecto de la filosofía DevOps, el cual supone prácticamente el principal objetivo que la metodología busca alcanzar. Hablamos de alcanzar un ritmo constante de incrementos en los que se vayan ofreciendo resultados continuamente al cliente. Para tratar que conseguir que los participantes de la encuesta comprendieran en mayor grado posible dicho concepto, se ha planteado una nueva situación en la que se esto suceda: este caso se basa en una aplicación, la cual se creó hace unos años, ofreciendo muchísimas menos funcionalidades de las que dispone a día de hoy, las cuáles se han ido incluyendo de forma continua con el paso del tiempo. Dicha situación es real, la aplicación también existe y se trata de la herramienta de mensajería y comunicación pionera entre los usuarios a nivel mundial: WhatsApp.

Ilustración 46. Gráfico de opinión según situación real [84]

En este caso, se ha preguntado a los encuestados sobre su elección entre una aplicación con las características que hemos descrito anteriormente, o, por el contrario, una aplicación la cual se comercialice desde el primer momento con todas sus funcionalidades implementadas y disponibles, pero para la cual haya que esperar más tiempo hasta que esté operativa. Y una vez más, los participantes se decantan por la primera opción, propia de la metodología.

Por tanto, y vistos los resultados obtenidos a través de la encuesta, podemos obtener dos conclusiones claras: la primera nos dice que DevOps es una metodología poco conocida en general por la sociedad; la segunda, según la distribución de preguntas según los participantes de la encuesta en cuanto a edad, ocupación y relación con el mundo de la informática, nos dice que, independientemente de si la metodología es conocida o no, o incluso si las personas tienen conocimientos sobre informática o interés, se muestra una clara aceptación generalizada sobre los principios en los que se sustenta la metodología y sus principales aspectos. Por todo ello, podríamos decir que sería una metodología bastante acertada a ojos de la sociedad.

9. MÉTRICAS DE DEVOPS

Uno de los factores más importantes dentro de la metodología que contribuyen a alcanzar el éxito son las métricas de DevOps. Para entender qué vamos a ver en este apartado y qué lugar ocupa dentro del método que estamos estudiando, comenzamos por definir bien dicho concepto y en qué consiste.

Una métrica, dentro del ámbito de DevOps, consiste en un conjunto de conceptos y pautas con los que llevar a cabo un análisis, el cual nos ofrecerá una serie de datos e información que serán interpretados para conocer el estado del proceso y cómo se está llevando a cabo, con el objetivo de establecer una supervisión del mismo. Las métricas se llevan a cabo con el objetivo de introducir perfeccionamientos, siempre que sean necesario y viable.

La metodología DevOps presenta cuatro métricas principales que nos ofrecen información relevante sobre el funcionamiento del desarrollo y el equipo. [34]

9.1. Período de cambios

Se trata de un intervalo de duración, conocido también como *Lead Time for Changes*, el cual comprende el tiempo que transcurre desde el momento en el que se admiten nuevas modificaciones hasta que consiguen llegar a la etapa de despliegue, es decir, hasta que están listas y disponibles. Durante este período, el código atraviesa algunas fases, como la de pruebas, implementación, etc.

Haciendo uso de esta métrica, podemos extraer bastante información sobre el progreso del trabajo. Cuando este período es demasiado prolongado, puede suponer un indicador de problemas de rendimiento en las labores, o incluso una mala gestión y organización de las mismas, aunque también podría deberse a modificaciones de amplio tamaño o procesos que no estén automatizados. Por el contrario, si este período es de corta duración, sabremos que estamos ante un grupo de trabajo con alta capacidad de actuación ante modificaciones y contratiempos. Para equipos de excelente productividad, este período dura de uno a siete días.

Esta métrica se obtiene registrando puntos temporales, los cuales corresponden a la introducción de una modificación y a su puesta en disponibilidad. Una vez hayamos obtenido estas dos referencias, se comprueba la diferencia de tiempos entre ambas y así sabremos qué duración tiene el período de cambios. Con estas medidas, podemos hacer una estimación promedio, pues no con todas las modificaciones obtendremos los mismos resultados.

9.2. Período de restablecimiento

Esta métrica, conocida también con las siglas MTTR (*Mean Time To Restore*), está basada en el período promedio de tiempo que pasa desde que se produce un problema hasta que se soluciona y se reestablece el funcionamiento normal del proceso de desarrollo. Entre otras cosas, representa información sobre la rapidez de respuesta que mostraría el equipo ante una situación de conflicto, aunque este período no dependa únicamente de este factor.

Cuando hacemos uso de esta métrica, estamos analizando varios aspectos dentro del proceso de desarrollo. Un período de restablecimiento corto puede ser indicador de una considerable robustez del sistema, o también, como hemos indicado anteriormente, un equipo preparado ante contratiempos. Por el contrario, un período de elevada duración puede denotar problemas en los procesos de reparación, o una mala gestión de la misma. Incluso podría deberse a una falta de experiencia en la resolución por parte del equipo, o simplemente incidencias en la comunicación de situaciones de fallo para su reparación. Se estima que, para equipos de niveles excelentes de productividad, el MTTR es inferior a 24 horas.

La mejor opción para enfrentar problemas de servicio es hacer uso de los conocidos *runbooks*, que son guías de apoyo utilizadas para gestionar procesos iterativos o conflictos. Estos pueden ser utilizados de forma tradicional, aunque también pueden tener un funcionamiento autónomo, incluso pueden presentarse en una forma híbrida. Aun así, existen otras muchas opciones para hacer que el MTTR sea un período con la menor duración posible, como el uso de técnicas de seguimiento y observación en busca de conflictos de forma temprana o, incluso, prestando atención al concepto de los avisos y cómo se llevan a cabo.

9.3. Proporción de fallos por modificaciones

Conocida también como CFR (*Change Failure Rate*), se trata de un indicador que nos informa sobre la cantidad de modificaciones que han resultado fallidas con respecto al total de las modificaciones introducidas.

Como resulta evidente, en cualquier situación en la que se utilice esta métrica se busca obtener valores reducidos para la misma, pues esto supondrá una correcta integración de modificaciones dentro del sistema. Para ello, los equipos tienen que tratar de aplicar continuas pruebas sobre el trabajo que se está realizando, para comprobar y asegurar que las nuevas integraciones funcionan correctamente.

Cuando nos encontramos con un elevado porcentaje de errores, podemos relacionarlo con la realización de procesos a mano o con bajo rendimiento. En cambio, si el porcentaje es reducido se puede justificar con buenas labores de captación de fallos y procesos adecuados de pruebas y testeo del software.

Según el informe *Accelerate State of Devops 2021*, los porcentajes más óptimos de fallos según la categoría de los equipos de desarrollo deben ser: hasta un 15% para los equipos de excelente productividad, entre un 15% y un 30% para los considerados como nivel alto, y mayor de 30% para los de normal y menor rendimiento.

9.4. Frecuencia de implementación

Esta métrica se encarga de informar sobre la cantidad y la regularidad con la que se realizan aportaciones dentro del proceso de desarrollo. Este aspecto es muy importante e interesante de controlar a la hora de medir el rendimiento en los equipos de trabajo, pues nos fijaremos en qué medida se incluyen nuevas implementaciones y el tamaño de las mismas.

Una situación ideal, según DevOps, estaría basada en realizar aportaciones de manera muy continua y que estas tengan un tamaño más reducido, lo que favorecería

al momento de aplicar procesos de testeo y, además, reducir la necesidad de aplicar modificaciones sobre dichas aportaciones al existir menos factores que pueda dar lugar a fallos o conflictos.

Si nos encontramos con un valor reducido para la frecuencia que estamos viendo puede deberse a un bajo rendimiento del equipo, o a pocas aportaciones al proceso de desarrollo, o incluso que el equipo ha decidido realizar implementaciones de mayor tamaño agrupando modificaciones.

10. PRÁCTICAS DE DEVOPS

Ya hemos visto toda la parte teórica e informativa sobre DevOps, en relación a qué es, los principales pilares sobre los que se sustenta, sus objetivos a alcanzar, qué propone para conseguirlos y utilidades disponibles sobre las que apoyarnos. Por tanto, vamos a analizar con más profundidad cuáles son las prácticas aplicables a nuestros proyectos al adoptar esta metodología.

10.1. Integración, entrega y despliegue continuos

Estas prácticas son muy importantes para la consecución de los objetivos de la filosofía DevOps, pues contribuyen a una obtención de resultados de manera más productiva y veloz.

La CI (*Continious Integration*) está basada en la inclusión de aportaciones de desarrollo a un proyecto de forma frecuente, a modo de ir fusionándolas de manera constante y reiterada junto con el trabajo ya producido, chequeado y validado. Además, gracias a la integración continua, se crea un procedimiento automático que se encarga de los procesos de compilación y testeo de los proyectos sobre los que se está trabajando, lo que ofrece más fiabilidad al producto, comodidad al equipo de trabajo y rapidez en el proceso de desarrollo.

Esta es una muy buena e importante práctica a llevar a cabo, pues el hecho de realizar aportaciones de pequeño tamaño, pero de forma más frecuente, contribuye a

reducir las posibilidades de que aparezcan fallos y problemas, y, en caso de que esto ocurra, se facilita el proceso de resolución de los mismos. Por esto, dicha práctica requiere que se trabaje añadiendo las aportaciones del equipo a una rama principal en la que estará alojado el progreso del proyecto, y para ello, se debe recurrir a un VCS (*Version Control System*) que controle dichas aportaciones, comprobando los posibles errores que puedan surgir al realizar la unión; además, junto con el VCS, se llevan a cabo otros procesos de evaluación.

La integración continua se lleva a cabo de la siguiente forma: una vez que un desarrollador requiere añadir nuevas modificaciones, el sistema lleva a cabo la unión de estas con el progreso original alojado en los almacenes de código, momento en el que se iniciará una comprobación de dichas aportaciones que luego pasarán por su correspondiente proceso de testeo para ser validado.

Por otro lado, tenemos la entrega continua, la cual se usa para la puesta en disponibilidad del producto, que puede verse fraccionado en diferentes versiones. Esto pasa por una serie de procesos, en los cuales se aplican procesos de evaluación de forma autónoma y alojamiento de las nuevas modificaciones en el almacén de código correspondiente.

Finalmente, tenemos el despliegue continuo, el cual supone el paso final de este proceso. Este consiste en la puesta en disponibilidad al usuario de cualquier modificación que haya atravesado todas las fases anteriores, en el cual ya no se necesita personal que lleve a cabo esto, lo que supone un respiro al grupo de trabajo al poder dedicar sus esfuerzos en el desarrollo del producto y, de esta forma, poder desentenderse en cierta manera de esta última fase. [35]

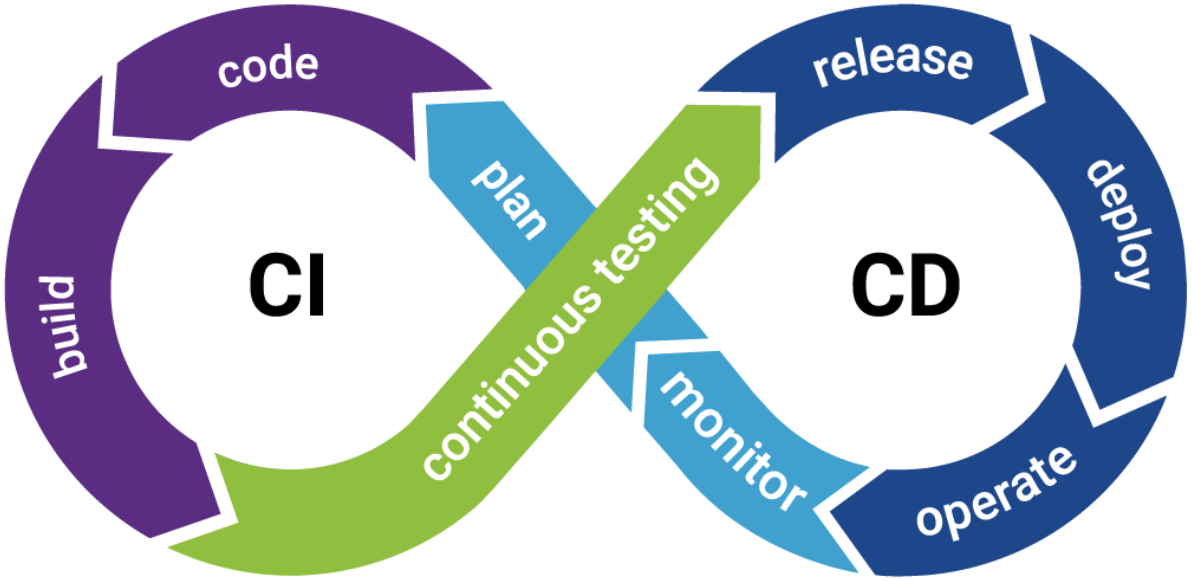

Ilustración 47. Ciclo DevOps [85]

Para entender mejor esta práctica, podemos utilizar un símil, basado en una tubería, que representa el camino que atraviesa el software pasando por una serie de fases hasta alcanzar un estado de disponibilidad.

Comienza por la etapa de construcción, en la que se procesa y se interpreta código que proviene de diferentes aportaciones que han sido incluidas en el proyecto y unificadas. Tras esto, llegamos a la etapa de testeo, que se da una vez haya finalizad la construcción. Esta fase irá dedicada a la aplicación de una serie de procesos de evaluación del trabajo ya procesado anteriormente, en la que se llevarán a cabo los correspondientes chequeos. Esto nos llevará a la etapa de puesta en marcha, en la que se usará un servidor para comprobar el funcionamiento del producto, y desembocará en una etapa donde se aplicarán otra serie de chequeos. Finalmente, si todo ha ido bien, el software estará listo para pasar por producción y entrega en cualquier momento.

Aun así, a pesar de que todo esto constituye un proceso lineal, pueden darse situaciones de fallos en cada una de las etapas que hemos visto, a las que se debe reaccionar alertando al equipo sobre dichos conflictos para su resolución. Una vez se hayan subsanado, se vuelve a atravesar este camino, por lo que se entiende que,

hasta que el ciclo de vida del software haya sido completado, se pueden realizar varias repeticiones. [36]

10.2. Pruebas y monitoreo continuos

Ya hemos visto el concepto de las pruebas y procesos de testeo a lo largo de los puntos anteriores, pero aquí vamos a enfocarlas como una de las prácticas más beneficiosas de la metodología DevOps. Como bien sabemos, el proceso de evaluación del trabajo realizado supone un aspecto primordial para cualquier empresa, al igual que para la filosofía de DevOps. Pero si nos paramos a pensar esto, llegaremos a la conclusión de que la cantidad de pruebas que se deben realizar atendiendo a lo que dictamina la metodología es inviable cuando estas las lleva a cabo el equipo de trabajo por su propia mano.

La práctica de pruebas continuas trata de paliar esta imposibilidad al establecer los procesos de testeo de forma autónoma, haciendo uso de las herramientas pertinentes que contribuyan a ello. Dicha práctica busca, además, establecer una especie de carácter constante a estos procesos, pues se llevan a cabo sobre cualquier modificación que se incluye en software. De esta forma, evitamos una acumulación de fallos, además de mantener un control ininterrumpido sobre el progreso que estamos obteniendo y el funcionamiento del producto. Debemos tener en cuenta que los testeos aplicados al producto van cambiando, en función de la etapa en la que se encuentre, el progreso que haya experimentado, o muchos otros factores. Por así decirlo, cuanto más avance a lo largo de su ciclo de vida, más complejos y elaborados serán los testeos que se lleven a cabo.

Por otro lado, tenemos el monitoreo continuo como otra práctica de DevOps. El monitoreo nos ofrece información acerca de la productividad del software que se ha creado para ofrecer al cliente, con el objetivo de supervisar los niveles de eficacia y eficiencia que caracterizan a nuestro producto para, posteriormente, y si fuese necesario, llevar a cabo modificaciones para mejorar dichas características. Hemos de saber que el monitoreo no lleva a cabo la búsqueda de fallos en el producto, pues de esto se encargan los procesos de pruebas: simplemente trata de obtener la máxima

optimización del servicio que estamos ofreciendo y analizar adecuadamente qué estamos construyendo y qué se podría perfeccionar. [37]

10.3. Automatización

Como bien hemos entendido analizando la metodología DevOps y su filosofía, la automatización supone un aspecto que proporciona mayor rapidez al proceso de desarrollo y mejor calidad de los resultados. Cuando este concepto lo trasladamos a la práctica, debemos analizar el funcionamiento del equipo de trabajo, qué procesos se van a llevar a cabo dentro del desarrollo y cuáles se pueden automatizar, y las herramientas más adecuadas para llevar esto a cabo.

En cuanto a las prácticas que hemos visto previamente, contienen procesos los cuales son compatibles con una automatización de los mismos. Dentro de la integración continua, los procesos de compilación y fusión de aportaciones de código, en los cuales se produce la detección de fallos y conflictos, se realizan de forma autónoma; en el ámbito de evaluación, los procesos de comprobación de errores también se llevan a cabo automáticamente, pues en numerosas ocasiones, suponen una gran dificultad o, incluso imposibilidad de realizar de forma manual, ya sea por la gran cantidad de chequeos que haya que aplicar o por elevadas cuantías de tiempo que haya que dedicar a estos procesos. Y, como acabamos de ver recientemente, el despliegue también supone otro proceso que se realiza mecánicamente sin necesidad de personal para ello. [38]

11. EJEMPLO DE APLICACIÓN PRÁCTICA

Una vez hemos visto y analizado la metodología en profundidad, finalizaremos con un ejemplo en el que se pongan en práctica los conceptos estudiados sobre DevOps. Para ello, se ha creado una aplicación web sencilla con el propósito de simular cómo sería el desarrollo software adoptando esta metodología en cuestión, teniendo en cuenta que no es posible reproducir exactamente cómo sería un caso real por los siguientes motivos: en primer lugar, el desarrollo ha sido llevado a cabo

únicamente por una persona que ha llevado a cabo tanto la parte de desarrollo como la parte de operaciones; por otro lado, debemos ser conscientes que dicha persona no es profesional DevOps, es decir, no cuenta con una cualificación equiparable a la de especialistas de este ámbito.

11.1. Descripción de la aplicación

La aplicación web en cuestión trata sobre un sencillo sistema web, accesible a través de Internet mediante cualquier navegador haciendo uso de este enlace *https://bit.ly/3qeqeS1*, el cual permite a sus usuarios crear y gestionar una serie de tareas personales. Para ello, en su versión original, la aplicación ofrece al usuario un espacio privado en el que almacenar y consultar los ítems que desee crear, por lo que uno de los requisitos para acceder al sistema es proporcionar una serie de credenciales de acceso para identificar a cada usuario. Por tanto, los usuarios previamente deben disponer de una cuenta registrada o, en su defecto, crear una nueva y acceder con los datos de acceso elegidos en el proceso de registro.

El desarrollo de la aplicación se ha dividido en tres incrementos o entregas de versiones funcionales de la misma, siguiendo esta distribución:

- **V1.0.** La aplicación se ha equipado con las siguientes características y funcionalidades: el usuario dispone de un formulario de registro para la creación de una nueva cuenta en el sistema, además de una vía de acceso a la aplicación mediante *login*, dispone de un espacio personal desde el que puede crear nuevas tareas a través de un formulario; además, el usuario puede cerrar sesión cuando lo necesite.
- **V2.0.** La aplicación se ha actualizado, ofreciendo una nueva funcionalidad para eliminar tareas, accesible desde una nueva página en la que se ofrece la información individual de la tarea en concreto a eliminar. Además, se han aplicado una serie de estilos CSS para ofrecer al usuario una interfaz más cuidada e intuitiva.
- **V3.0.** Finalmente, la aplicación se ha adquirido una última funcionalidad, la cual permite al usuario eliminar su cuenta del sistema y, por consecuencia, las tareas que haya registrado previamente. Esta última opción es accesible desde la página personal de cada usuario.

11.2. Herramientas utilizadas para el desarrollo software

Durante el proceso de desarrollo de la aplicación, se han utilizado una serie de herramientas que han contribuido en cuanto a facilidad y rapidez, algunas vistas en los apartados 3 y 4.

Para llevar a cabo la gestión del proyecto, se ha utilizado un **tablero Kanban** (ver apartado 4.2), a través del cual se han establecido una serie de historias de usuario con requisitos para la aplicación. Dichos elementos han sido establecidos con un determinado nivel de prioridad, con el objetivo de ir enfocando las labores de desarrollo en las tareas de mayor importancia.

Por otro lado, se ha elegido **GitHub** como repositorio de código, que actúa de intermediario entre el área local de desarrollo y el servidor donde se aloja la aplicación. Además, debido a la gran cantidad de funcionalidades que esta herramienta ofrece, se ha utilizado para otros aspectos dentro del proceso de desarrollo, como por ejemplo la automatización de pruebas.

A continuación, dentro del siguiente apartado veremos de qué manera se han utilizado estas herramientas.

11.3. Proceso de desarrollo de la aplicación

El proceso de desarrollo comienza con la primera fase, en la que se lleva a cabo un análisis sobre la idea de aplicación que queremos crear y los requerimientos que queremos que cumpla. Esto es lo que se conoce como la gestión de proyectos, la cual se encarga de organizar la carga de trabajo y mostrar cuál es el progreso de trabajo. Para llevar a cabo esto, hemos utilizado un **tablero Kanban**, mediante el cual se han creado un conjunto de historias de usuario que representan los requisitos que pretendemos que nuestra aplicación cumpla.

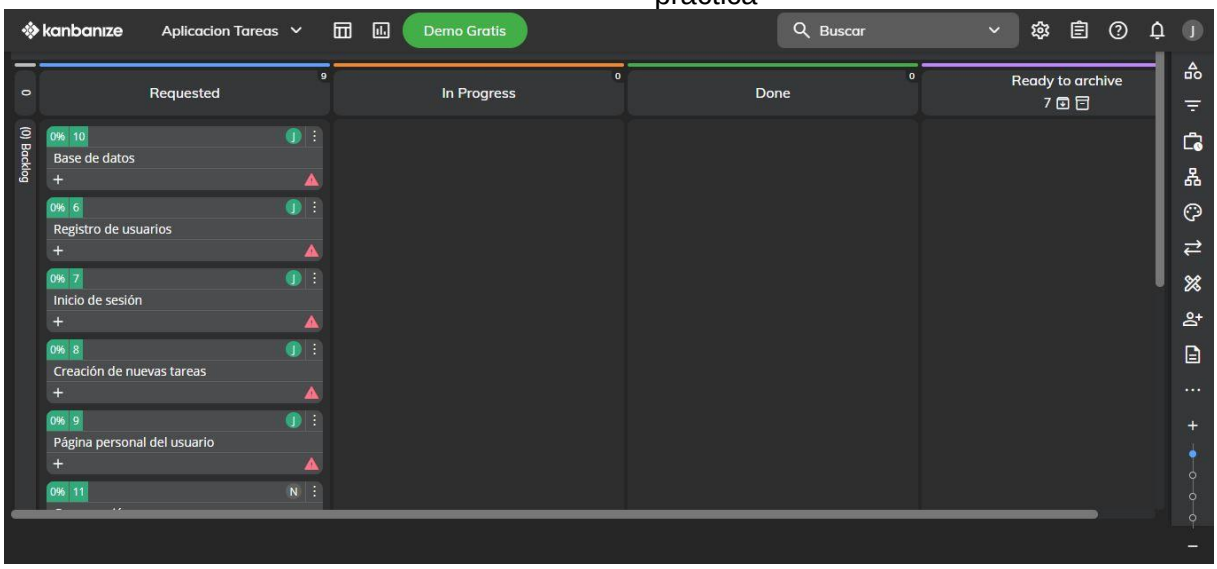

Ilustración 48. Tablero Kanban correspondiente al proyecto de nuestra aplicación [86]

Como podemos apreciar, se han creado una serie de tarjetas para representar dichas historias de usuarios, colocadas automáticamente en la columna inicial y listas para comenzar a trabajar con ellas. A medida que atraviesen el período en el que se desarrollen y finalicen, se irán desplazando por las columnas del tablero, por lo que el desarrollador será consciente en todo momento sobre el estado del progreso en el que se encuentra el proyecto.

Para acceder al tablero que se ha utilizado para la realización de este proyecto (ver figura 11.1), utilizaremos este enlace *https://tfgjesus2122.kanbanize.com/ctrl_board/2*.

Una vez hemos declarado la funcionalidad que requerimos para la aplicación y repartido la carga de trabajo en diferentes tareas, pasamos a la fase de diseño e implementación, en la que llevaremos a cabo la construcción de la aplicación mediante la creación de código que aporte forma y funcionalidad a la misma. Para ello, es necesario disponer de un entorno de desarrollo en el que crear código, lo que se conoce como IDE (*Integrated Development Enviroment*). En nuestro caso, hemos elegido *Visual Studio Code* en su versión 1.70.1, debido a que es un editor de código fácil de utilizar y con una gran cantidad de documentación disponible para resolver cualquier duda sobre su uso, además de ofrecer la posibilidad de trabajar con Git desde el mismo IDE.

La aplicación web ha sido escrita en PHP y MySQL, por lo que requerimos tanto de un servidor que interprete dicho código PHP, como de un servicio de base de datos al que se conecte la aplicación para el almacenamiento de datos que esta maneje. Por ello, en un principio se ha optado por utilizar el paquete *Xampp* en su versión 8.1.6, el cual ofrece servicios de servidor Apache y base de datos MariaDB de MySQL Server.

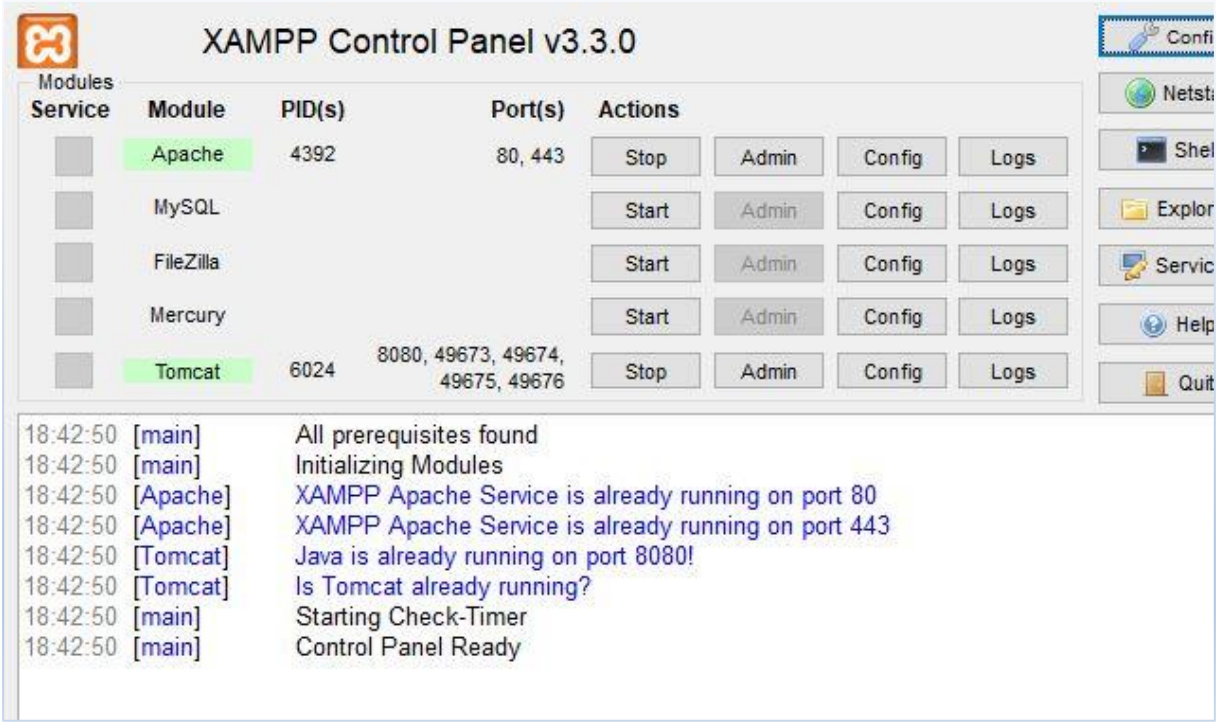

Ilustración 49. Panel de control Xampp [87]

Como podemos ver, Xampp ofrece un panel de control para administrar y configurar los servicios que ofrece. Para comprobar que el servidor Apache funciona correctamente, simplemente debíamos utilizar un navegador cualquiera para acceder a la dirección *localhost*, y como respuesta debería redirigirnos a la siguiente

Ilustración 50. Página principal del servidor Apache de Xampp [88]

Este servidor Apache nos servirá para ir comprobando el funcionamiento de la aplicación y la interfaz que ofrece a medida que el desarrollo va progresando. Aunque en un principio se haya utilizado MySQL de Xampp como servicio de base de datos para la administración de la aplicación, era necesario que la base de datos de la misma estuviese actualizada y disponible en todo momento para el correcto funcionamiento del sistema, por lo que se ha recurrido a un servicio de base de datos remota. Para ello, se ha creado una cuenta en el servicio de servidor MySQL *https://db4free.net/*. Una vez hecho esto, dicho servicio nos ofrece una herramienta propia de MySQL para la administración de base de datos, conocida como *phpMyAdmin*, mediante la cual llevaremos a cabo la creación de una base de datos con sus correspondientes tablas y atributos. En la figura 11.4 podemos apreciar cómo sería esta herramienta recientemente mencionada.

Ilustración 51. Tabla tareas de la base de datos aplicaciontareas - PHPMyAdmin [89]

Una vez vistos los requisitos, sabemos que la aplicación requiere de un registro de usuarios para poder hacer uso de la misma, por lo que es evidente que dichos usuarios deben ser almacenados en la base de datos, por lo que se ha creado una tabla específica para ello con los atributos a registrar de cada uno de ellos.

Ilustración 52. Tabla usuarios de la base de datos aplicaciontareas - PHPMyAdmin [90]

De la misma manera, es necesario otro espacio para almacenar el conjunto de tareas que cada usuario desee registrar en el sistema, por lo que se ha creado una nueva tabla con estos atributos.

Una vez la base de datos haya sido creada correctamente, pasamos a administrar GitHub. Lo primero que hemos hecho ha sido crear un nuevo repositorio remoto, accesible mediante este enlace *https://github.com/jgr00059/tfgjesus-2122* donde iremos almacenando nuestro progreso en cuanto al código de la aplicación se refiere.

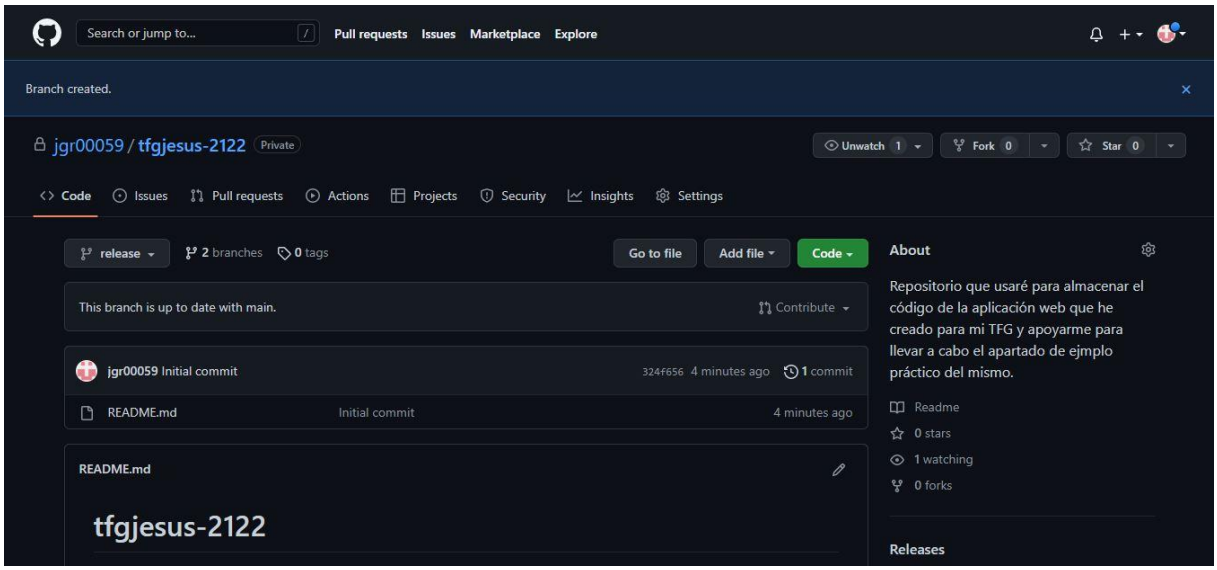

Ilustración 53. Repositorio GitHub donde se aloja nuestro proyecto [91]

Como vemos, inicialmente el repositorio cuenta únicamente con un fichero README.md, el cual tiene como objetivo almacenar información relacionada con el proyecto que se aloja en dicho repositorio. A partir de aquí, reproducimos el repositorio de manera local en nuestro equipo para empezar a trabajar y poder subir contenido a remoto.

Una vez hecho esto, debemos tener en cuenta que la aplicación debe alojarse en un servidor para que sea accesible y desplegable para cualquier usuario que desee hacer uso de la misma. Es por ello que, en un principio, recurrimos al servicio de hosting gratuito *https://www.000webhost.com/*. En nuestro caso particular, resultó bastante fácil de manejar y administrar, pero experimentamos un problema relacionado con el uso de Git en dicho servidor, por lo que recurrimos como segunda opción a un servicio VPS (*Virtual Private Server*) de pago perteneciente a la empresa IONOS, accesible mediante este enlace *https://www.ionos.es/*.

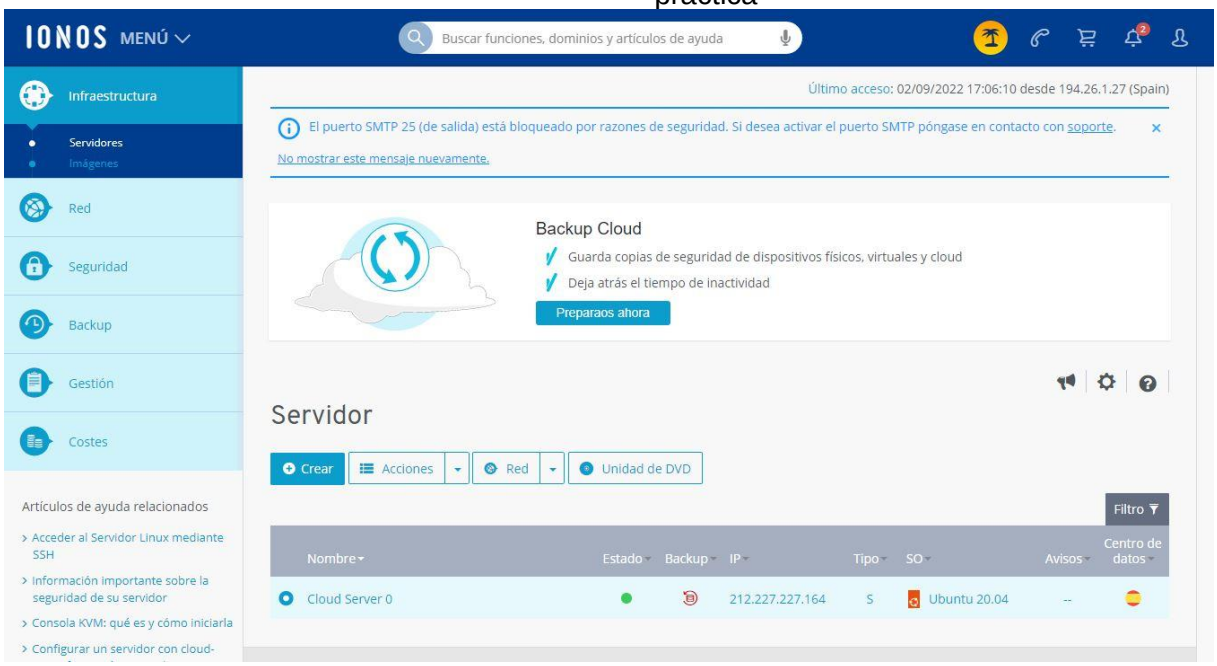

Ilustración 54. Servidor Ionos contratado para alojar nuestra aplicación [92]

En la figura 11.7 se aprecia la página de administración y configuración del servidor en cuestión, el cual no incluye en el plan contratado la opción de crear un dominio, por lo que ofrece una dirección IP como método de acceso al mismo. Por eso, se ha recurrido a *https://bitly.com/pages/home/v1*, un servicio acortador de URLs que genera un pequeño enlace que redirecciona al sitio web deseado.

A continuación, a través de un terminal, se ha establecido conexión mediante SSH con el servidor en cuestión para su administración. En primer lugar, se ha clonado el repositorio Git dentro del host remoto, para más adelante ir actualizando con el contenido que se vaya subiendo al repositorio. Pero, como bien sabemos, una de las características de DevOps es la automatización de procesos, principalmente el paso de la parte de desarrollo a la de operaciones, y en concreto el proceso de despliegue de la aplicación. Por tanto, se ha creado un cron en el servidor, el cual consiste en un proceso en segundo plano encargado de ejecutar acciones de forma periódica siguiendo unos intervalos de tiempo concretos.

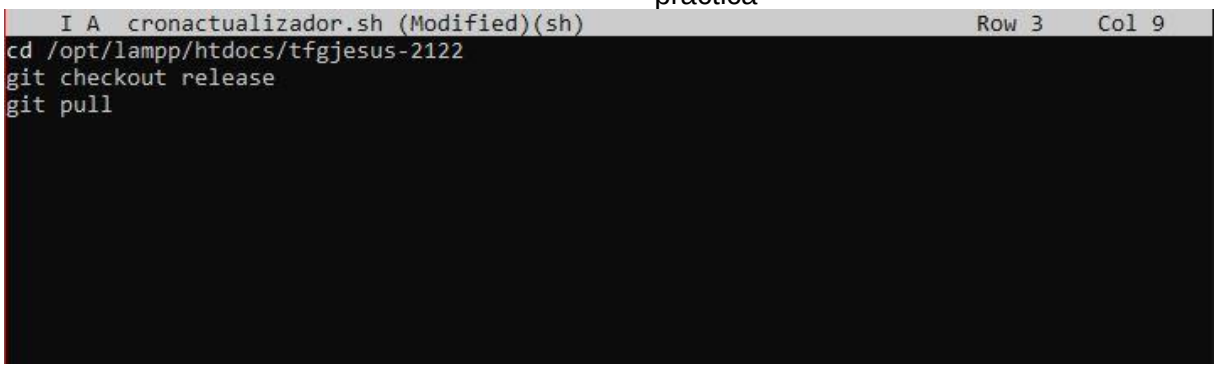

Ilustración 55. Script del cron creado para la automatización del despliegue [93]

En nuestro, lo que tratamos de conseguir es que el servidor se encargue de plasmar el contenido que se vaya subiendo al repositorio, teniendo en cuenta que previamente dicho código ha debido de pasar las pertinentes pruebas y ser validado como correctamente funcional. Por tanto, como podemos apreciar en la figura 11.8, lo que tratamos de llevar a cabo con este cron es acceder a la rama *release* (la cual será la que almacenará los incrementos de la aplicación listos para su uso) y sincronizar el contenido de la misma en el servidor.

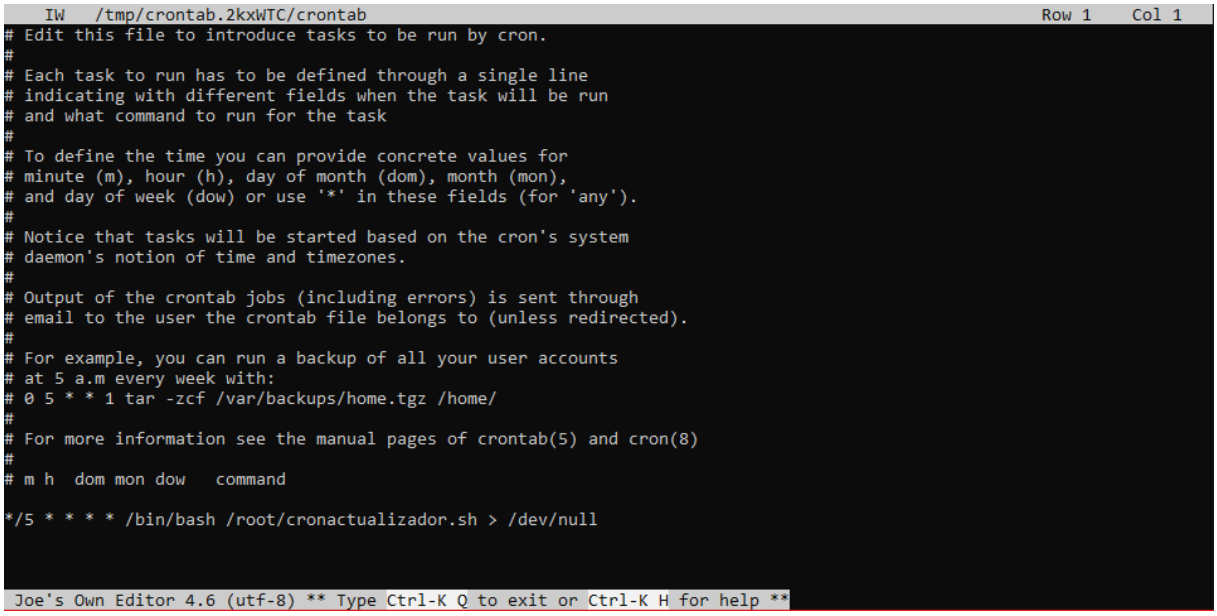

Ilustración 56. Directorio Crontab que contiene el comando para ejecutar el cron [94]

Una vez tenemos listo esto, accedemos al fichero *crontab*, el cual se encarga de ejecutar una serie de comandos con una frecuencia de tiempo concreta. En la figura

11.9 podemos apreciar, en la última línea, el comando correspondiente a nuestro cron, el cual dictamina que cada 5 minutos se ejecute el script que hemos visto previamente. Por tanto, con esto quedaría automatizado el proceso de despliegue de la aplicación, y así, comenzamos realmente con el proceso de diseño e implementación de la aplicación web, dividido como hemos comentado anteriormente en tres incrementos.

11.3.1. Primer incremento (V1.0)

En primer lugar, comenzamos estableciendo una página de inicio a la que acceder cuando hagamos uso de nuestra aplicación, por lo que hemos diseñado una sencilla interfaz en la que se ofrecen opciones para el registro de usuarios o, en caso de disponer de cuenta en el sistema, acceder al mismo mediante *login*.

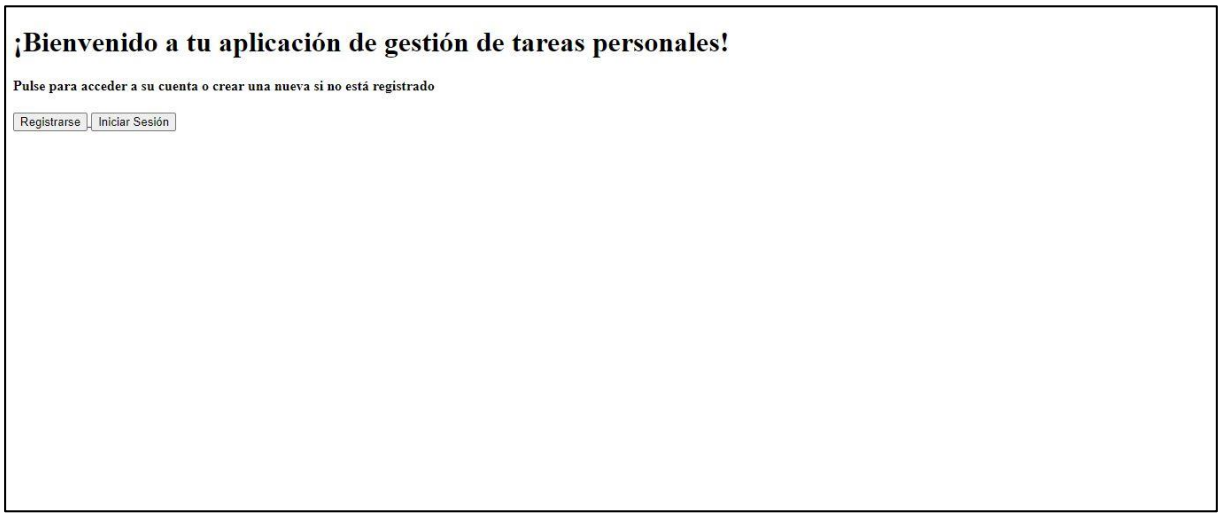

Ilustración 57. Página principal de nuestra aplicación V1.0 [95]

Como podemos apreciar en la figura 11.10, se trata de una simple y básica aplicación a nivel de apariencia y funcionalidad.

A continuación, debemos definir una conexión a la base de datos antes de definir los métodos de *registro* y *login*, los cuales requerirán de dicha conexión para almacenar los datos de nuevos usuarios y comprobar credenciales de acceso de registrados en el sistema. Antes de continuar con el proceso, hemos de comentar que, con respecto al proceso de pruebas, hemos incluido un test aplicable al proyecto para

simular como sería la automatización del proceso de pruebas como parte del uso de la metodología DevOps. Dicho test consiste en comprobar que la conexión a la base de datos es correcta, y para ello realiza una serie de comprobaciones muy sencillas.

Para el proceso de automatización de pruebas, se ha utilizado una funcionalidad de GitHub, conocida como *GitHub Actions*. En concreto, se ha implementado un workflow (proceso automatizado), el cuál aplica automáticamente un test almacenado en un fichero PHP cada vez que se sube contenido al repositorio (ver figura 11.11).

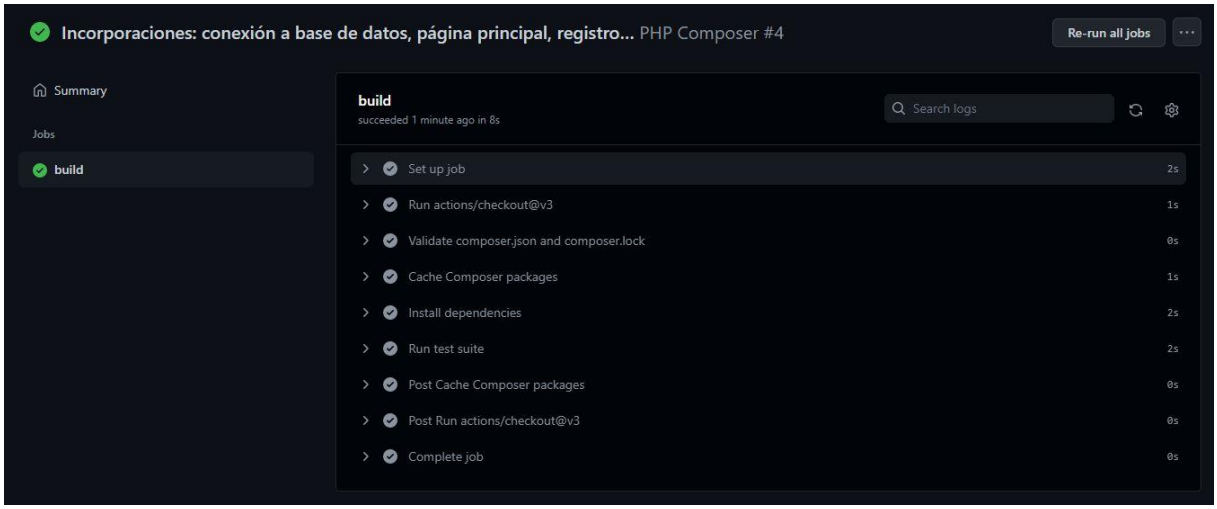

Ilustración 58. Worflow para automatización de pruebas [96]

Al crear el workflow, debemos tener en cuenta que necesitamos especificar cuáles son las pruebas a ejecutar. Para ello, hemos creado el fichero *composer.json* (ver figura 11.12), el cual incluye principalmente las dependencias necesarias en el proyecto. Además, hemos tenido que incluir una serie de parámetros obligatorios, ajustándonos al esquema que debe seguir dicho fichero. Esta información está disponible para su consulta en este enlace *https://getcomposer.org/doc/04-schema.md*.

```
{} composer.json > ...
{
    "name": "jgr00059/tfgjesus2122",
    "description": "Prueba de test sobre la aplicación",
    "license": "proprietary",
    "authors": [
        {
            "name": "Jesús García Rodríguez"
        }
    ],
    "scripts": {
        "test-jesus": ["@php test.php"]
    }
}
```

Ilustración 59. Fichero composer.json [97]

Una vez la conexión esté bien implementada, construimos la página de registro de nuevos usuarios, la cual consistirá en un simple formulario en el que se introducirán datos a almacenar en el sistema acerca de los mismos (nombre, apellidos, email y contraseña).

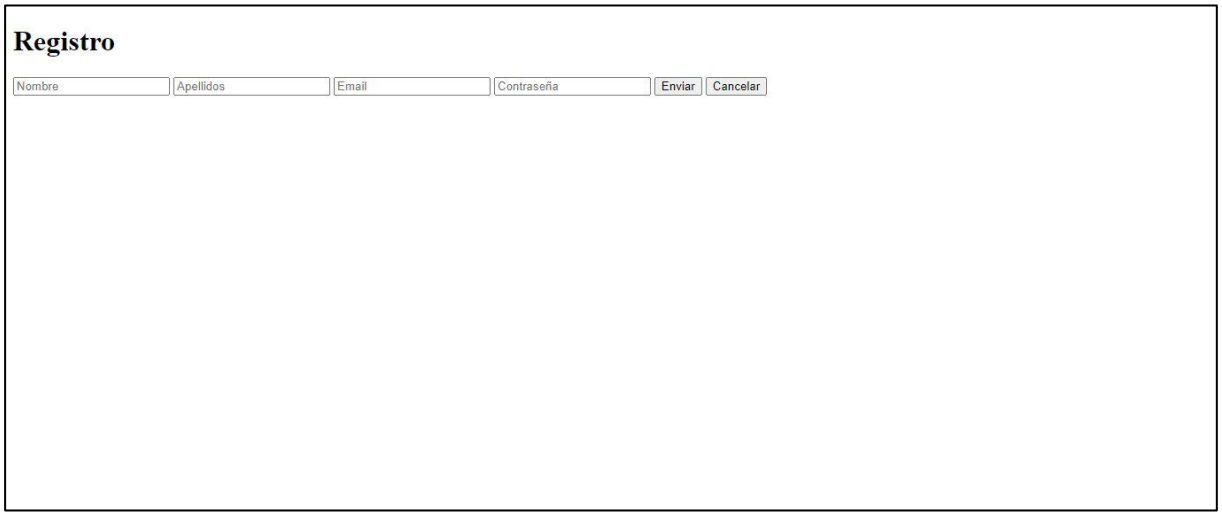

Ilustración 60. Página de registro de nuestra aplicación V1.0 [98]

De la misma forma, definimos también la página correspondiente al inicio de sesión en el sistema, que incluye otro formulario en el que enviar datos al servidor (email y contraseña), que a su vez comprobará si existen dichas credenciales de

acceso registradas en la base de datos para permitir o rechazar dicha conexión al sistema.

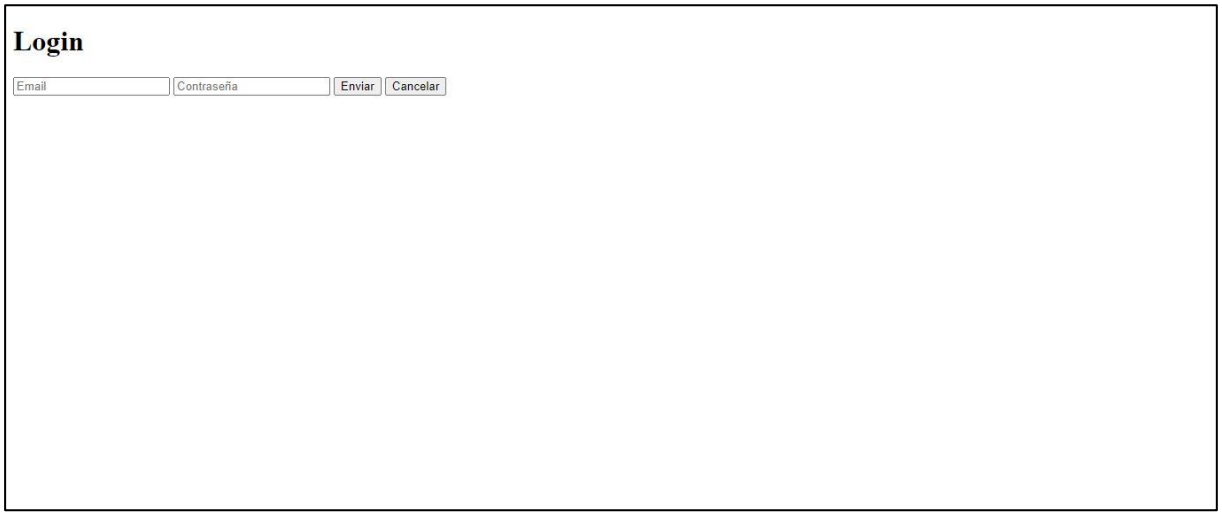

Ilustración 61. Página de login de nuestra aplicación V1.0 [99]

Ahora, implementamos la página personal del usuario, en la cual se muestran las tareas creadas por el mismo y ofrece la opción de crear nuevas tareas o cerrar sesión.

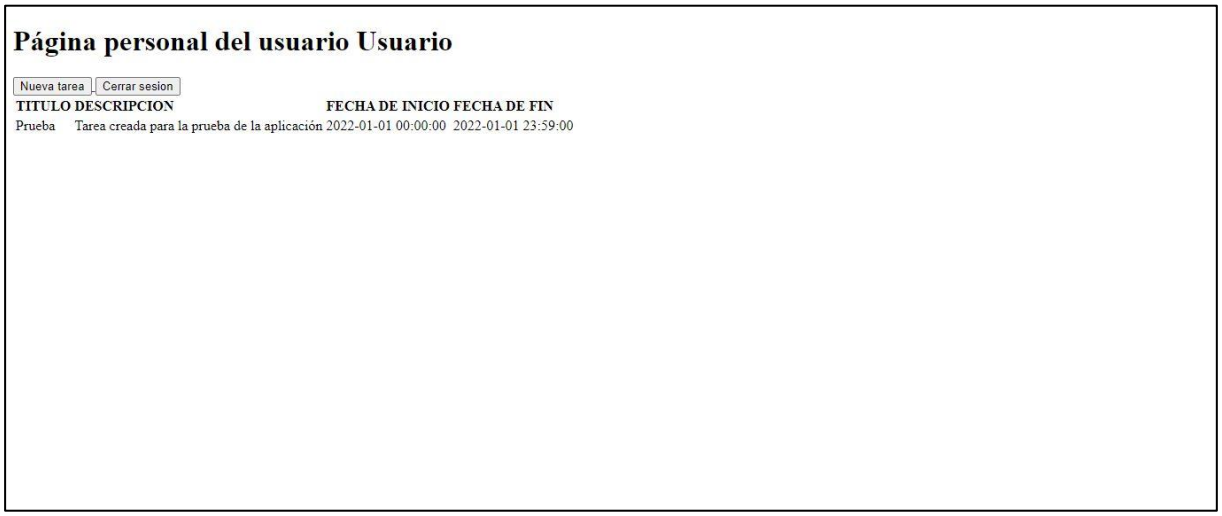

Ilustración 62. Página principal del usuario de nuestra aplicación V1.0 [100]

Para finalizar, creamos la página correspondiente a la inserción de nuevas tareas, basada en un formulario con una serie de campos que almacenarán información sobre las mismas (título, descripción, fecha de inicio y fecha de fin de la

tarea). Además, creamos el fichero encargado de gestionar el cierre de sesión del sistema, el cual no tiene vista asociada.

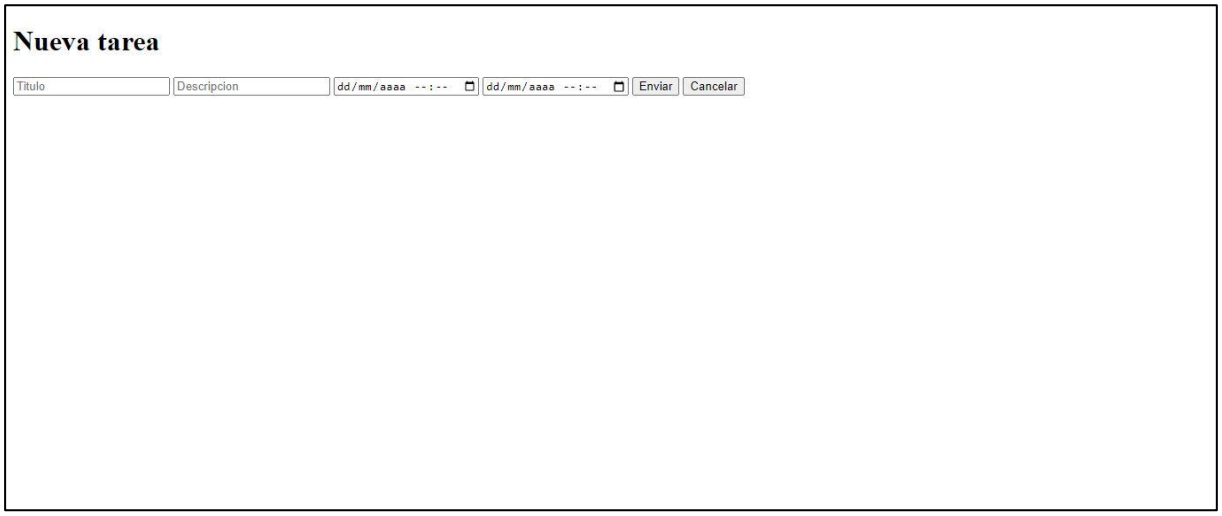

Ilustración 63. Página de registro de nueva tarea de nuestra aplicación V1.0 [101]

Una vez hayamos llevado a cabo todo esto, subimos este contenido al repositorio Git para aplicar las pertinentes pruebas y, posteriormente, incluirlo en la rama *release* para su despliegue automático en el servidor. Así, habremos concluido con las historias de usuario más prioritarias del tablero que creamos para la gestión del proyecto, pudiendo así ofrecer una versión de la aplicación mínimamente funcional (v1.0).

11.3.2. Segundo incremento (V2.0)

En el momento en el que la primera entrega se haya puesto disponible en el servidor para su uso, podemos continuar con el proceso de desarrollo. En esta segunda versión de la aplicación, una de las nuevas actualizaciones consiste en aplicar una serie de estilos con el objetivo de ofrecer al usuario una interfaz mucho más cuidada e intuitiva, algo que resulta muy importante a la hora de ofrecer un producto software a un cliente.

Ilustración 64. Página de registro de nuestra aplicación V2.0 [102]

Como podemos apreciar en esta imagen, la apariencia de la página de inicio ha cambiado con respecto a la primera versión (ver figura 11.10), ofreciendo ahora algo más de orden y estilo, algo que se ha logrado simplemente haciendo uso de una configuración básica CSS.

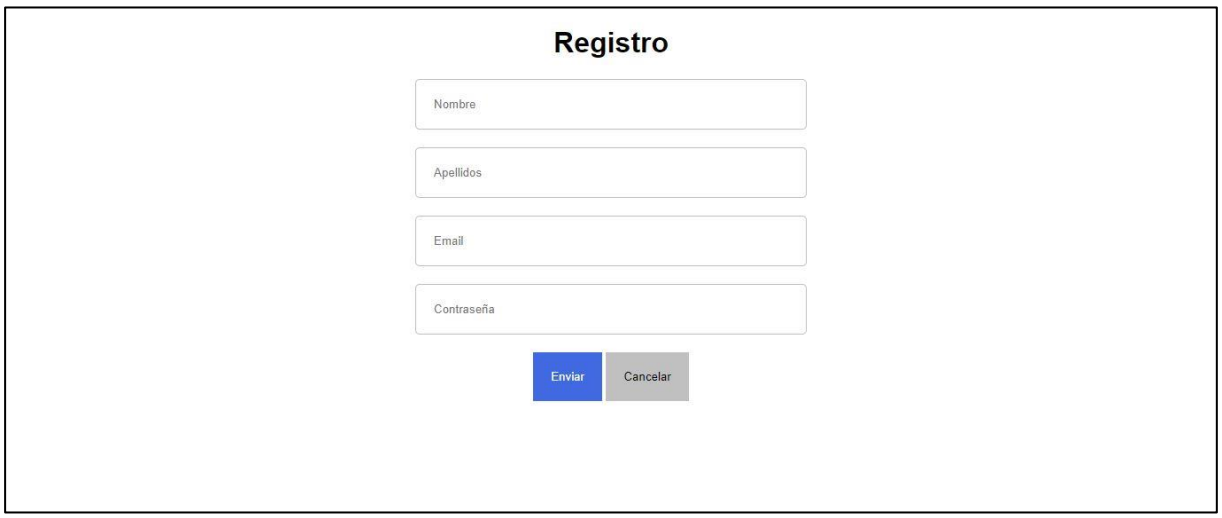

Ilustración 65. Página de registro de nuestra aplicación V2.0 [103]

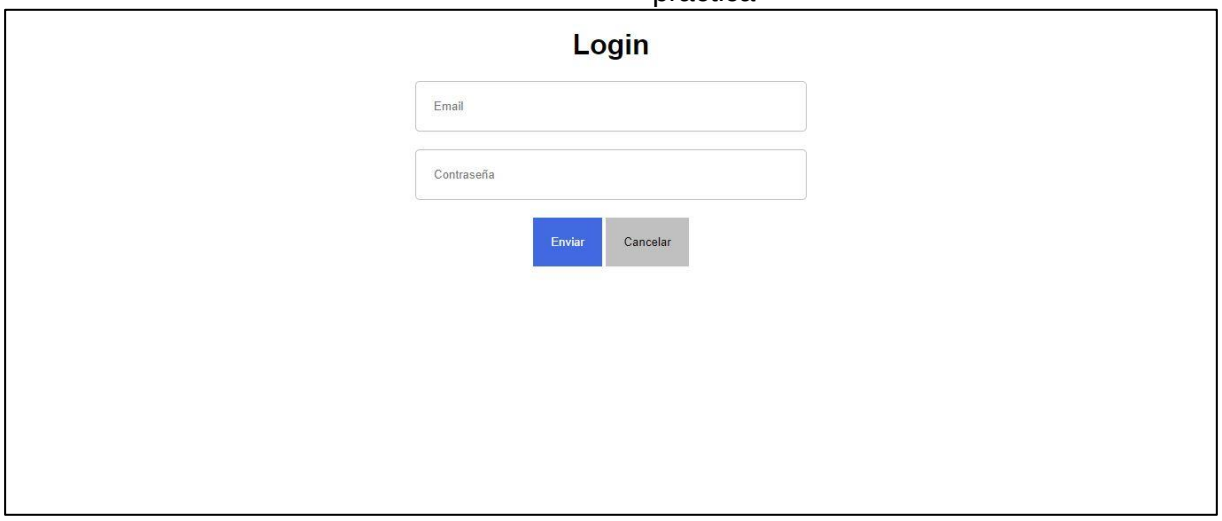

Ilustración 66. Página de login de nuestra aplicación V2.0 [104]

Ilustración 67. Página de registro de nueva tarea de nuestra aplicación V2.0 [105]

Lo mismo ocurre con las páginas de registro de usuarios (figura 11.18), *login* (figura 11.19) y registro de nuevas tareas (figura 11.20), las cuáles ofrecen ahora formularios mucho más cuidados y ordenados.

Ilustración 68. Página principal del usuario de nuestra aplicación V2.0 [106]

De la misma forma, la página principal del usuario también ha sido modificada en cuanto a la interfaz, ofreciendo ahora información sobre las tareas del usuario de una manera más visual.

Además, en esta nueva versión de la aplicación, se ha incluido una nueva funcionalidad que permite al usuario borrar tareas de su espacio personal. Para ello, se ha tenido que crear una nueva vista en la que acceder a esta opción (ver figura 11.22).

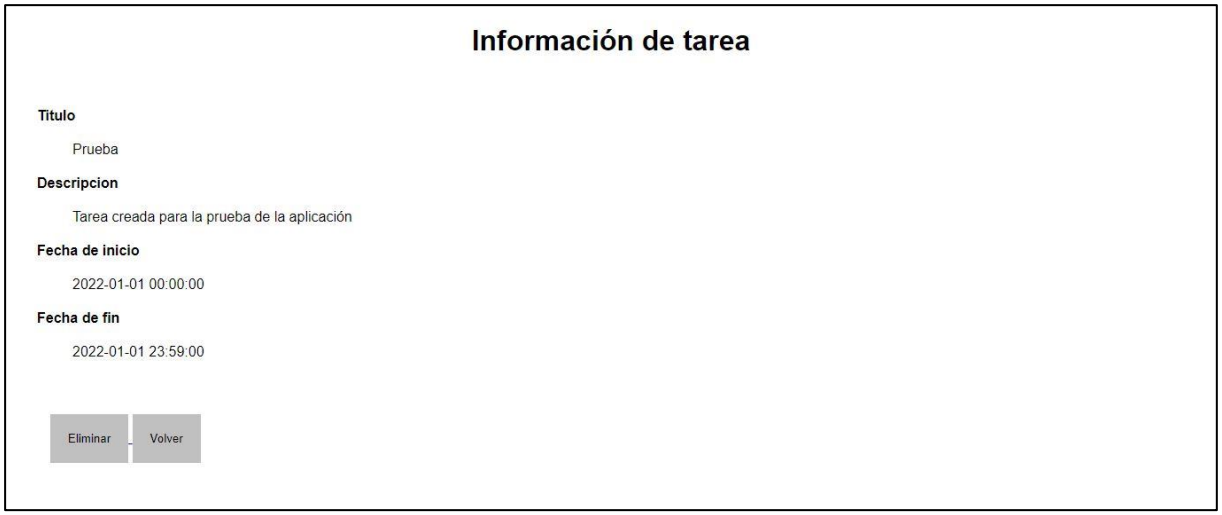

Ilustración 69. Página de información de tarea V2.0 [107]

Esta vista, además, ofrece al usuario la información completa sobre una tarea en concreto, además de la opción previamente mencionada de eliminar dicho elemento del sistema.

11.3.3. Tercer incremento (V3.0)

Para finalizar, se ha creado una última versión de la aplicación, la cual tan sólo ha incluido una nueva funcionalidad que permite al usuario eliminar su cuenta del sistema y, en consecuencia, las tareas que haya almacenado en la misma.

Como hemos podido ver, se ha desarrollado una aplicación web muy simple, pero hemos de recordar que, con la realización de este ejemplo práctico no pretendíamos demostrar tanto habilidades de programación y diseño web, sino la capacidad de seguir la metodología DevOps para el desarrollo de proyectos software, automatizando en todo momento y siempre que sea posible los distintos procesos necesarios para llevar a cabo dicho desarrollo y ofrecer continuamente resultados al cliente, motivo por el cual se ha decidido dividir el proyecto en tres incrementos con pocas actualizaciones en lugar de tan sólo dos.

12. CONCLUSIONES

Una vez hayamos concluido con el trabajo, obtendremos, como resultado del mismo, una serie de conclusiones sobre nuevos conocimientos acerca del tema elegido y/o más ámbitos relacionados con el mismo, y sobre el mismo trabajo realizado.

En nuestro caso, podemos concluir diciendo que, efectivamente, DevOps supone una metodología de gran importancia en el ámbito del desarrollo software, pues las empresas que decidan adoptarla podrán experimentar mejoras en sus procesos productivos, tales como una optimización del tiempo de trabajo, un abaratamiento de los costes de producción, mejoras de calidad de los resultados y un mayor grado de satisfacción del cliente, todo esto siempre y cuando las mismas

dediquen todos sus esfuerzos para adaptarse a la metodología y, en la medida de lo posible, cumplir con la filosofía DevOps y con los principios que la definen.

Por último, con respecto a la aplicación práctica de DevOps, hemos de decir que se ha logrado, en gran parte, cumplir con la metodología en cuanto al proceso de desarrollo se refiere, pues se han podido hacer uso de herramientas para el mismo, además de aplicar automatización de una serie de tareas que contribuyen a concluir con dicho proceso.

BIBLIOGRAFÍA

[1] *¿Qué es DevOps y para qué sirve?*. Recuperado el 07 de febrero de 2022, de https://www.netapp.com/es/devops-solutions/what-is-devops/

[2] *¿Qué es DevOps? Explicación de DevOps*. Recuperado 07 de febrero de 2022, de https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops/

[3] Redacción KeepCoding. (2022). *Historia y origen de DevOps*. Recuperado el 21 de mayo de 2022, de https://keepcoding.io/blog/historia-y-origen-de-devops/

[4] Paul, F. (2014). *The incredible true story of how DevOps got its name*. Recuperado el 21 de mayo de 2022, de https://newrelic.com/blog/nerd-life/devops-name

[5] Devops Latam. (2020). *Entrevista Patrick Debois*. Recuperado el 16 de marzo de 2022, de https://devopslatam.com/entrevista-patrick-debois/

[6] *Cultura de DevOps*. Recuperado el 24 de marzo de 2022, de https://www.atlassian.com/es/devops/what-is-devops/devops-culture

[7] Crawford, A. (2019). *¿Qué es DevOps?*. Recuperado el 06 de abril de 2022, de https://www.ibm.com/cloud/learn/devops-a-complete-guide

[8] Muradas, Y. (2020). *Qué es Jira*. Recuperado el 28 de abril de 2022, de https://openwebinars.net/blog/que-es-jira/

[9] Nissen, R. (2021). *Dando tus primeros pasos: Conceptos básicos de Jira*. Recuperado el 04 de mayo de 2022, de https://blog.deiser.com/es/primeros-pasos-con-conceptos-basicos-de-jira

[10] Sentrio. (2021). *Introducción a Jenkins: ¿qué es, para qué sirve y cómo funciona?*. Recuperado el 21 de abril de 2022, de https://sentrio.io/blog/que-es-jenkins/

[11] *Pipeline – Jenkins*. Recuperado el 21 de abril de 2022, de https://www.jenkins.io/doc/book/pipeline/

[12] *History of GitHub*. Recuperado el 23 de abril de 2022, de https://pslmodels.github.io/Git-Tutorial/content/background/GitHubHistory.html

[13] Siles, F. (2015). *Un día el desarrollador despertó y descubrió que GitHub se había convertido en el centro de la programación*. Recuperado el 23 de abril de 2022, de https://www.xataka.com/aplicaciones/un-dia-el-desarrollador-desperto-y-descubrio-que-github-se-habia-convertido-en-el-centro-de-la-programacion

[14] Moreno, O. (2020). *Introducción a Selenium*. Recuperado el 23 de abril de 2022, de http://oscarmoreno.com/selenium/

[15] *Selenium: una herramienta de automatización*. Recuperado el 24 de abril de 2022, de https://es.acervolima.com/ingenieria-de-software-selenium-una-herramienta-de-automatizacion/

[16] *Ansible*. Recuperado el 24 de abril de 2022, de https://www.mclibre.org/consultar/webapps/lecciones/ansible-1.html

[17] Pérez, M. (2017). *Qué es Ansible*. Recuperado el 24 de abril de 2022, de https://openwebinars.net/blog/que-es-ansible/

[18] Redacción KeepCoding. (2022). *¿Qué son los roles de Ansible y para qué sirven?*. Recuperado el 24 de abril de 2022, de https://keepcoding.io/blog/que-son-los-roles-de-ansible-y-para-que-sirven/

[19] North Team. (2022). *¿Qué es Nagios?*. Recuperado el 25 de abril de 2022, de https://www.north-networks.com/que-es-nagios/

[20] (2017). *Nagios: todos los procesos de red a tu alcance*. Recuperado el 27 de abril de 2022, de https://www.ionos.es/digitalguide/servidores/herramientas/nagios-todos-los-procesos-de-red-a-tu-alcance/

[21] *¿Qué es cloud computing?*. Recuperado el 22 de mayo de 2022, de https://cloud.google.com/learn/what-is-cloud-computing?hl=es

[22] (2022). *What is Azure DevOps?*. Recuperado el 22 de mayo de 2022, de https://docs.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops

[23] Schwaber, K., Sutherland, J. (2016). *La guía definitive de Scrum: las reglas del juego*. Recuperado el 20 de abril de 2022, de https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf

[24] *La historia de Kanban*. Recuperado el 07 de mayo de 2022, de https://kanbantool.com/es/guia-kanban/historia-de-kanban

[25] Rehkopf, M. *¿Qué es un tablero de Kanban?*. Recuperado el 8 de mayo de 2022, de https://www.atlassian.com/es/agile/kanban/boards#:~:text=Un%20tablero%20de%20kanban%20es,orden%20de%20su%20trabajo%20diario.

[26] Laoyan, S. (2022). *Scrumban: lo mejor de dos metodologías ágiles*. Recuperado el 09 de mayo de 2022, de https://asana.com/es/resources/scrumban

[27] Álvarez, A. (2019). *DevOps Fundamentals y los principios DevOps definidos por DASA*. Recuperado el 10 de mayo de 2022, de https://netmind.net/es/devops-fundamentals-y-los-principios-devops/

[28] Buchanan, I. *Ventajas de DevOps*. Recuperado el 11 de mayo de 2022, de https://www.atlassian.com/es/devops/what-is-devops/benefits-of-devops

[29] Mell, E. (2020). *Supere estos cinco desafíos DevOps*. Recuperado el 11 de mayo de 2022, de https://www.computerweekly.com/es/cronica/Supere-estos-cinco-desafios-DevOps

[30] *6 essential DevOps roles you need on your team*. Recuperado el 12 de mayo de 2022, de https://www.pagerduty.com/resources/learn/essential-devops-roles/

[31] (2019). *Qué es SER y en qué se parece (y diferencia) de DevOps*. Recuperado el 13 de mayo de 2022, de https://discoverthenew.ituser.es/devops/2019/02/que-es-sre-y-en-que-se-parece-y-diferencia-de-devops

[32] *Fundamentos de la metodología Agile*. Recuperado el 13 de mayo de 2022, de https://www.wrike.com/es/project-management-guide/fundamentos-de-la-metodologia-agile/

[33] *DevOps vs Agile*. Recuperado el 13 de mayo de 2022, de https://www.javatpoint.com/devops-vs-agile#:~:text=DevOps%20is%20a%20practice%20of,%2C%20small%2C%20and%20rapid%20releases.&text=DevOps%20purpose%20is%20to%20manage,is%20to%20manage%20complex%20projects.

[34] Sentrio. (2021). *Cómo medir el éxito DevOps a través de las Four Key Metrics*. Recuperado el 19 de mayo de 2022, de https://sentrio.io/blog/four-key-metrics/#:~:text=M%C3%A9tricas%20DevOps%3A%20Four%20Key%20Metrics&text=Estas%20m%C3%A9tricas%20se%20pueden%20usar,y%20consiguen%20una%20mayor%20eficiencia.

[35] Cubillo, M. (2020). *DevOps: mejores prácticas*. Recuperado el 20 de mayo de 2022, de https://www.encora.com/es/blog/devops-mejores-practicas

[36] *Cómo construir una tubería CI/CD desde cero*. Recuperado el 20 de mayo de 2022, de https://programmerclick.com/article/4731859977/

[37] (2020). *La importancia del seguimiento en DevOps*. Recuperado el 20 de mayo de 2022, de https://discoverthenew.ituser.es/devops/2020/10/la-importancia-del-seguimiento-en-devops

[38] (2019). *Claves de los procesos de automatización en DevOps*. Recuperado el 20 de mayo de 2022, de https://oasys-sw.com/claves-de-los-procesos-de-automatizacion-en-devops/

[39] Fuente: https://thebroadcastknowledge.com/tag/patrick-debois/

[40] Fuente: https://salesdorado.com/es/automatizaci%C3%B3n/herramientas-de-gesti%C3%B3n-de-proyectos/avis-jira/

[41] Fuente: https://icon-icons.com/es/icono/jenkins-logo/167854

[42] Fuente: http://adictosalainformatica.com/quien-somos/informacion-profesional/github-logo/

[43] Fuente: https://promdevelop.com/es/technologies/selenium/

[44] Fuente: https://kangaroot.net/solutions/ansible

[45] Fuente: https://servernotfound.es/ansible-uso-y-ejemplo-de-playbooks/

[46] Fuente: https://docs.ansible.com/ansible/latest/user_guide/sample_setup.html#sample-directory-layout

[47] Fuente: https://docs.ansible.com/ansible/latest/user_guide/sample_setup.html#sample-task-and-handler-files-in-a-function-based-role

[48] Fuente: https://icon-icons.com/icon/nagios-logo/168955

[49] Fuente: https://devclass.com/2018/08/30/devops-report/

[50] Fuente: https://www.kindpng.com/imgv/hiobTRJ_microsoft-azure-logo-svg-hd-png-download/

[51] Fuente: https://geeks.ms/jorge/2007/05/09/explicando-scrum-a-mi-abuela/

[52] Fuente: https://netmind.net/es/scrum-el-pasado-y-el-futuro/

[53] Fuente: https://www.freepng.es/png-4ha7ww/

[54] Fuente: https://comparecamp.com/kanban-tool-review-pricing-pros-cons-features/

[55] Fuente: https://content.intland.com/blog/agile/scrum-kanban-scrumban

[56] Fuente: https://sinapsismx.com/innovacion/customer-centric-la-razon-del-triunfo-de-los-grandes/

[57] Fuente: https://setting.com.br/blog/gestao-empresarial/ferramentas-gestao-de-projetos/

[58] Fuente: https://www.eviciti.com.mx/blog-post/metodologias-para-la-mejora-continua-empresarial/

[59] Fuente: https://itcomunicacion.com.mx/automatizacion-de-procesos-por-donde-empezar/

[60] Fuente: https://www.modulo.com.br/leidigital-lgpd/equipe-multidisciplinar-500x500px/

[61] Fuente: https://asana.com/es/resources/begin-with-the-end-in-mind

[62] Fuente: https://builtin.com/devops

[63] Fuente: https://castor.com.co/charla-devops-para-lideres/

[64] Fuente: https://blog.inedo.com/tag/release-management

[65] Fuente: https://cdn.thenewstack.io/media/2022/08/974ee97e-productivity-1995786_1280-1024x598.jpg

[66] Fuente: https://insights.dice.com/2019/11/25/worldwide-developers-study/

[67] Fuente: https://www.webcreek.com/en/blog/technology/importance-software-quality-assurance/

[68] Fuente: https://blogs.vmware.com/cloud/2020/12/01/cybersecurity-challenges-2021-predicted-prepare/

[69] Fuente: Imagen de autoría propia

[70] Fuente: Imagen de autoría propia

[71] Fuente: Imagen de autoría propia

[72] Fuente: Imagen de autoría propia

[73] Fuente: Imagen de autoría propia

[74] Fuente: Imagen de autoría propia

[75] Fuente: Imagen de autoría propia

[76] Fuente: Imagen de autoría propia

[77] Fuente: Imagen de autoría propia

[78] Fuente: Imagen de autoría propia

[79] Fuente: Imagen de autoría propia

[80] Fuente: Imagen de autoría propia

[81] Fuente: Imagen de autoría propia

[82] Fuente: Imagen de autoría propia

[83] Fuente: Imagen de autoría propia

[84] Fuente: Imagen de autoría propia

[85] Fuente: https://www.flagship.io/ci-cd/

[86] Fuente: Imagen de autoría propia

[87] Fuente: Imagen de autoría propia

[88] Fuente: Imagen de autoría propia

[89] Fuente: Imagen de autoría propia

[90] Fuente: Imagen de autoría propia