UNIVERSIDAD TÉCNICA DEL NORTE

Logotipo

Descripción generada automáticamente

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Sistemas Computacionales

**Desarrollo de un módulo web para fortalecer la gestión de actividades académicas y administrativas como componente del entorno virtual de aprendizaje integrado (EVAI) para la empresa IERec.**

Trabajo de grado previo a la obtención del título de Ingeniero en sistemas Computacionales

Autor:

José Breiner Pai Gonzáles

Director:

Msc. Cosme MacArthur Ortega Bustamante

Ibarra – Ecuador

2021

# Cesión de derecho

# Certificación

# Dedicatoria

# Agradecimiento

# Tabla de Contenidos

[Cesión de derecho 2](#_Toc67608828)

[Certificación 2](#_Toc67608829)

[Dedicatoria 2](#_Toc67608830)

[Agradecimiento 2](#_Toc67608831)

[Tabla de Contenidos 2](#_Toc67608832)

[Índice de Figuras 4](#_Toc67608833)

[Índice de Cuadros 5](#_Toc67608834)

[Resumen 5](#_Toc67608835)

[Abstract 5](#_Toc67608836)

[INTRODUCCIÓN 5](#_Toc67608837)

[Antecedentes 5](#_Toc67608838)

[Situación Actual 5](#_Toc67608839)

[Planteamiento del Problema 5](#_Toc67608840)

[Objetivos 5](#_Toc67608841)

[Objetivo General 5](#_Toc67608842)

[Objetivos Específicos 5](#_Toc67608843)

[Alcance 5](#_Toc67608844)

[Arquitectura de Funcionamiento 5](#_Toc67608845)

[Justificación 5](#_Toc67608846)

[CAPÍTULO 1: MARCO TEÓRICO 5](#_Toc67608847)

[1.1 Herramientas Tecnológicas. 5](#_Toc67608848)

[1.1.1 NodeJS 5](#_Toc67608849)

[1.1.2 MongoDB 6](#_Toc67608850)

[1.1.3 Docker 6](#_Toc67608851)

[1.2 Arquitectura de Microservicios. 7](#_Toc67608852)

[1.2.1 Microservicios 7](#_Toc67608853)

[1.2.2 Requisitos 8](#_Toc67608854)

[1.3 Metodología de desarrollo. 8](#_Toc67608855)

[1.3.1 Metodología ágil. 9](#_Toc67608856)

[1.3.2 Metodología de desarrollo ágil XP 9](#_Toc67608857)

[1.3.2.1 Planeación 10](#_Toc67608858)

[1.3.2.2 Diseño 11](#_Toc67608859)

[1.3.2.3 Codificación 11](#_Toc67608860)

[1.3.2.4 Pruebas 11](#_Toc67608861)

[1.3.2.5 Producción 11](#_Toc67608862)

[1.4 Norma ISO 25010. 12](#_Toc67608863)

[1.4.1 ISO 25010 13](#_Toc67608864)

[1.4.2 Calidad del producto 14](#_Toc67608865)

[1.4.3 Compactibilidad 15](#_Toc67608866)

[1.4.4 Usabilidad 15](#_Toc67608867)

[1.4.5 Fiabilidad 16](#_Toc67608868)

[1.4.6 Seguridad 16](#_Toc67608869)

[1.4.7 Mantenibilidad 16](#_Toc67608870)

[1.4.8 Portabilidad 17](#_Toc67608871)

[CAPÍTULO II: DESARROLLO 17](#_Toc67608872)

[CAPÍTULO III: RESULTADOS 17](#_Toc67608873)

[Conclusiones 17](#_Toc67608874)

[Recomendaciones 17](#_Toc67608875)

[Glosario 17](#_Toc67608876)

[Referencias 17](#_Toc67608877)

[Anexos 19](#_Toc67608878)

# Índice de Figuras

[Ilustración 1 Comparativa entre contenedores y sistemas operativos linux tradicionales. (R. DOC, 2021) 7](#_Toc67608879)

[Ilustración 2 Comparación complejidad en la arquitectura de microservicios y un monolito (Auribox, 2017). 8](#_Toc67608880)

[Ilustración 3 Ciclo de vida de la programación extrema (Anwer et al., 2017). 10](#_Toc67608881)

[Ilustración 4 El costo de corregir defectos aumenta drásticamente a lo largo del proceso de desarrollo (Hovorushchenko, 2018). 13](#_Toc67608882)

[Ilustración 5 Modelo de calidad de uso de la Norma ISO/IEC 25010 (Cruz et al., 2020). 14](#_Toc67608883)

# Índice de Cuadros

# Resumen

# Abstract

# INTRODUCCIÓN

## Antecedentes

## Situación Actual

## Planteamiento del Problema

## Objetivos

### Objetivo General

### Objetivos Específicos

## Alcance

## Arquitectura de Funcionamiento

## Justificación

# CAPÍTULO 1: MARCO TEÓRICO

## Herramientas Tecnológicas.

### NodeJS

Ideado como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node.js está diseñado para crear aplicaciones network escalables (N. DOC, 2021).

Node.js es similar en diseño y está influenciado por sistemas como [Event Machine](https://github.com/eventmachine/eventmachine) de Ruby y [Twisted](https://twistedmatrix.com/trac/) de Python. Pero Node.js lleva el modelo de eventos un poco más allá. Incluye un [bucle de eventos](https://github.com/nodejs/node/blob/master/doc/topics/event-loop-timers-and-nexttick.md) como runtime de ejecución en lugar de una biblioteca. En otros sistemas siempre existe una llamada de bloqueo para iniciar el bucle de eventos. Por lo general, el comportamiento se define mediante devoluciones callbacks de llamada al iniciarse un script y al final se inicia un servidor a través de una llamada de bloqueo como EventMachine::run(). En Node.js, no existe como tal la llamada de inicio del evento de bucle o start-the-event-loop. Node.js simplemente entra en el bucle de eventos después de ejecutar el script de entrada y sale cuando no hay más devoluciones callbacks de llamada para realizar. Se comporta de una forma similar a JavaScript en el navegador - el bucle de eventos está oculto al usuario (N. DOC, 2021).

### 1.1.2 MongoDB

MongoDB es una base de datos de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado (M. DOC, 2021).

MongoDB almacena datos en documentos flexibles similares a JSON, por lo que los campos pueden variar entre documentos y la estructura de datos puede cambiarse con el tiempo (M. DOC, 2021).

### 1.1.3 Docker

Docker ayuda a los desarrolladores a dar vida a sus ideas al conquistar la complejidad del desarrollo de aplicaciones. Simplificamos y aceleramos los flujos de trabajo de desarrollo con un canal de desarrollo integrado y mediante la consolidación de los componentes de la aplicación. Utilizados activamente por millones de desarrolladores en todo el mundo, Docker Desktop y Docker Hub brindan una simplicidad, agilidad y opciones incomparables (D. DOC, 2021).

Las herramientas del contenedor, como Docker, ofrecen un modelo de implementación basado en imágenes. Esto permite compartir una aplicación, o un conjunto de servicios, con todas sus dependencias en varios entornos. Docker también automatiza la implementación de la aplicación (o conjuntos combinados de procesos que constituyen una aplicación) en este entorno de contenedores (R. DOC, 2021).

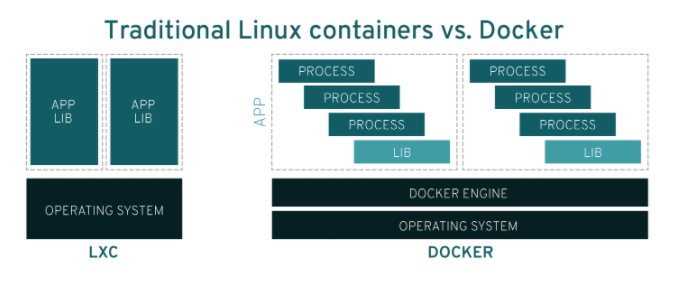


Ilustración 1 Comparativa entre contenedores y sistemas operativos linux tradicionales. (R. DOC, 2021)

## 1.2 Arquitectura de Microservicios.

### 1.2.1 Microservicios

En los últimos 3 o 4 años se ha acentuado a nivel mundial el desarrollo de aplicaciones Web como microservicios. Esta estrategia de desarrollo es una variante de SOA, con énfasis en la creación de pequeños sistemas desacoplados e intercomunicados, lo que conlleva múltiples beneficios en comparación a las aplicaciones Web monolíticas tradicionales. Entre las ventajas más importantes de usar microservicios está la capacidad de responder de manera rápida a las nuevas necesidades de los usuarios, ya que el proceso de desarrollo, comprobación y distribución de estas aplicaciones es mucho más expedito. Además, permite la independencia de las tecnologías a emplear en cada uno de los componentes, tales como lenguajes de programación, frameworks de desarrollo y servicios adicionales que pudiesen ser necesarios, sin repercutir en otros componentes (Navarro et al., 2017).

Los microservicios (Newman, 2015) son una realización del estilo arquitectónico orientado a servicios del software de construcción compuesto por pequeños servicios que se pueden implementar y escalar de forma independiente (Anon, 2020b). Cada microservicio tiene una funcionalidad empresarial independiente, se ejecuta como un proceso independiente y se comunica a través de mecanismos ligeros (Vayghan et al., 2021).

### 1.2.2 Requisitos

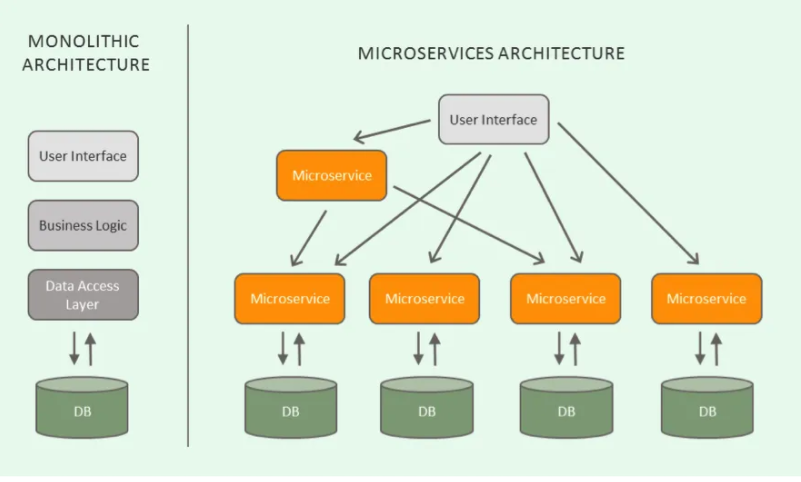
Uno de los requisitos fundamentales de la arquitectura de microservicios es que todos sus componentes puedan comunicarse entre sí, bajo un protocolo en común. En la actualidad la mayoría de los servicios realizan esto mediante peticiones a un conjunto determinado de endpoints HTTP, que corresponden a la Application Programming Interface (API) de los sistemas, que generalmente proporcionan información en formato JSON. Estos endpoints pueden ser de tipo público o privado; para estos últimos se requiere autenticación a través de mecanismos tales como OAuth [12] o JSON Web Tokens [14]. Este esquema de trabajo per-mite además la comunicación entre aplicaciones de escritorio y móviles, sin duplicar esfuerzos de desarrollo (Navarro et al., 2017).

Ilustración 2 Comparación complejidad en la arquitectura de microservicios y un monolito (Auribox, 2017).

## 1.3 Metodología de desarrollo.

metodología, es una palabra formada por tres vocablos griegos: metá que significa más allá, dos que significa camino y logos que significa estudio; considerando lo anterior como el estudio más allá del camino (Montero et al., 2018).

### 1.3.1 Metodología ágil.

Las metodologías de desarrollo de software AGILE proporcionan un paradigma de desarrollo iterativo y evolutivo con más énfasis en los requisitos cambiantes, la satisfacción del cliente y la colaboración en equipo (Anwer et al., 2017).

Los modelos ágiles son en realidad una colección de mejores prácticas y principios de ingeniería de software.

Estas metodologías surgieron en 2001 en respuesta a las limitaciones de las metodologías basadas en planes. La alta tasa de proyectos fallidos, cancelados y retrasados obligó a los profesionales del software a reconciliar los principios y prácticas de desarrollo (Anwer et al., 2017).

Estos principios pueden no ser nuevos para la industria del software, pero en el modelado ágil se utilizan con un enfoque diferente que los hace más flexibles y adaptables durante el desarrollo, además pueden adaptarse a las necesidades de desarrollo de software rápido.

Las metodologías ágiles presentan como principal particularidad la flexibilidad, los proyectos en desarrollo son subdivididos en proyectos más pequeños, incluye una comunicación constante con el usuario, son altamente colaborativos y es mucho más adaptable a los cambios. De hecho, el cambio de requerimientos por parte del cliente es una característica especial, así como también las entregas, revisión y retroalimentación constante (Montero et al., 2018).

### 1.3.2 Metodología de desarrollo ágil XP

La metodología extreme programming o XP, es la metodología ágil más conocida. Fue desarrollada por Kent Beck en la búsqueda por guiar equipos de trabajo pequeños o medianos, entre dos y diez programadores, en ambientes de requerimientos imprecisos o cambiantes La principal particularidad de esta metodología son las historias de usuario, las cuales corresponden a una técnica de especificación de requisitos, se trata de formatos en los cuales el cliente describe las características y funcionalidades que el sistema debe poseer (Montero et al., 2018).

Se llama “Programación Extrema”, por el hecho de que llevó al extremo aquellas prácticas que se consideraron útiles para desarrollar software de alta calidad. XP acentúa enormemente la satisfacción del cliente. La retroalimentación rápida y los lanzamientos frecuentes ayudan a gestionar los defectos cercanos a su origen. Una tasa de defectos más baja reduce el costo de desarrollo y da como resultado un producto final más aceptable a un costo menor (Anwer et al., 2017).

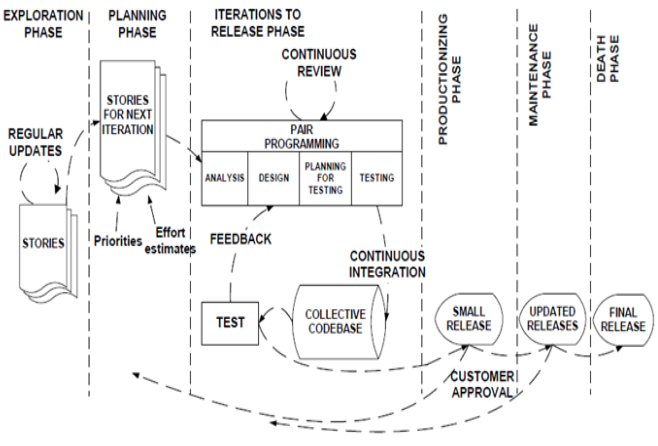


Ilustración 3 Ciclo de vida de la programación extrema (Anwer et al., 2017).

### 1.3.2.1 Planeación

Cada iteración comienza con la planificación de la iteración. En esta fase, los desarrolladores preparan un plan de sus actividades para implementar las características requeridas de la versión actual. Al igual que la planificación del lanzamiento, la planificación de la iteración también tiene fases de exploración, compromiso y dirección, pero el cliente no participa en este paso. Durante la planificación de la iteración, el programador selecciona las tareas para implementar y estima el costo, el tiempo y el esfuerzo necesarios para la tarea seleccionada. Se pueden asignar tareas a otros programadores para equilibrar la carga de trabajo (Anwer et al., 2017).

### 1.3.2.2 Diseño

El diseño XP sigue rigurosamente el principio MS (mantenlo sencillo). Un diseño sencillo siempre se prefiere sobre una representación más compleja. Además, el diseño guía la implementación de una historia conforme se escribe: nada más y nada menos. Se desalienta el diseño de funcionalidad adicional porque el desarrollador supone que se requerirá después (Cevallos, 2016).

### 1.3.2.3 Codificación

Un concepto clave durante la actividad de codificación (y uno de los aspectos del que más se habla en la XP) es la programación por parejas. XP recomienda que dos personas trabajen juntas en una estación de trabajo con el objeto de crear código para una historia. A medida que las parejas de programadores terminan su trabajo, el código que desarrollan se integra con el trabajo de los demás. En ciertos casos, esto lo lleva a cabo a diario un equipo de integración. En otros, las parejas de programadores tienen la responsabilidad de la integración. Esta estrategia de “integración continua” ayuda a evitar los problemas de compatibilidad de interfaces y brinda un ambiente de “prueba de humo” que ayuda a descubrir a tiempo los errores (Cevallos, 2016).

### 1.3.2.4 Pruebas

La creación de pruebas unitarias antes de que comience la codificación es un elemento clave del enfoque de XP, ya que esto asegura la calidad del software (Cevallos, 2016).

### 1.3.2.5 Producción

Al ser un proceso iterativo e incremental, XP ofrece software en versiones pequeñas. Una versión es una pequeña parte del software lanned que implementa algunas necesidades comerciales. Los lanzamientos frecuentes en XP permiten construir el sistema requerido en incrementos. Un ciclo de lanzamiento puede constar de una serie de iteraciones que pueden abarcar de 1 a 4 semanas. La fase de producción se trata de la implementación del software en versiones pequeñas. Para comprobar si el software está listo, se realizan las pruebas de aceptación, las pruebas del sistema y las pruebas de carga. Durante esta fase, los programadores reducen la velocidad a la que evoluciona el sistema. A medida que el riesgo se vuelve más importante, si un cambio debe pasar a la próxima versión o no (Anwer et al., 2017)

## 1.4 Norma ISO 25010.

En la actualidad, el software se encuentra en diversos campos de la actividad humana, por lo que resulta necesario para las empresas no solo definir ciertos criterios de calidad sino conocer la percepción que tiene el usuario respecto a él para lograr satisfacer realmente sus necesidades (Cruz et al., 2020).

Hoy en día, casi todas las áreas de la actividad humana están asociadas con sistemas informáticos y de información aplicada, que se basan en el software. El factor clave del uso eficaz del software y uno de los principales requisitos de los usuarios y las partes interesadas para el software moderno es la alta calidad. La calidad del software es el grado de conformidad del software con las necesidades de los clientes. Los estudios analíticos y revisiones sobre software muestran que ahora la crisis en el campo de la evaluación y aseguramiento de la calidad del software continúa. Exactamente los requisitos del software determinan las características necesarias de la calidad del software y su impacto en los métodos de evaluación cuantitativa de la calidad del software. Los proyectos de software con requisitos y especificaciones incompletos no pueden tener una implementación exitosa. Muchos errores de software surgen en la etapa de formulación de requisitos (10-23% de todos los errores de software). Cuanto antes sze descubra el defecto, más barato será repararlo. los defectos descubiertos después del lanzamiento del producto pueden costar casi 100 veces más repararlos que los defectos encontrados durante el proceso de requisitos (Hovorushchenko, 2018).

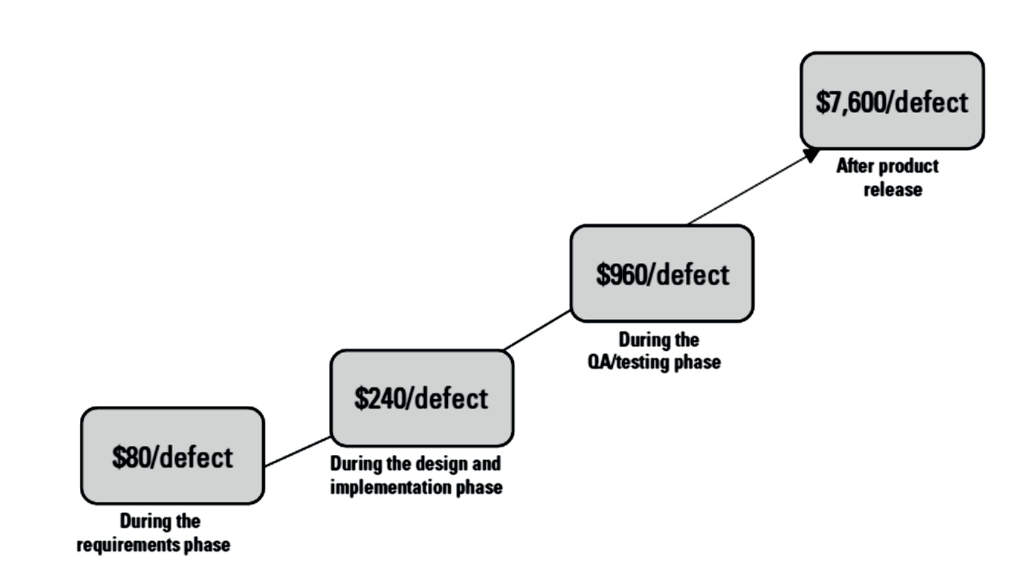


Ilustración 4 El costo de corregir defectos aumenta drásticamente a lo largo del proceso de desarrollo (Hovorushchenko, 2018).

Tratando de prevenir los costos elevados que provoca un software mal desarrollado, nace un estándar para ISO, específicamente la ISO 25010.

### 1.4.1 ISO 25010

La Norma ISO/IEC 25010 es un estándar internacional que define el modelo de calidad de un producto de software. El modelo de calidad representa la piedra angular en torno a la cual se establece el sistema para la evaluación de la calidad del producto. En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto de software determinado (Cruz et al., 2020).

ISO / IEC 25010 no incluye la evaluación de la funcionalidad y características específicas de una aplicación, ya que su valor solo puede medirse en relación con las necesidades de la organización adquirente. Por lo tanto, el modelo de Calidad en Uso busca cuantificar la "usabilidad" (efectividad, eficiencia y satisfacción) de la aplicación, cuando usuarios específicos intentan alcanzar sus objetivos específicos. ISO / IEC 25010 define la efectividad como la "Precisión e integridad con la que los usuarios logran objetivos específicos", y eficiencia como "recursos gastados en relación con la precisión e integridad con la que los usuarios logran objetivos" (Estdale & Georgiadou, 2018).

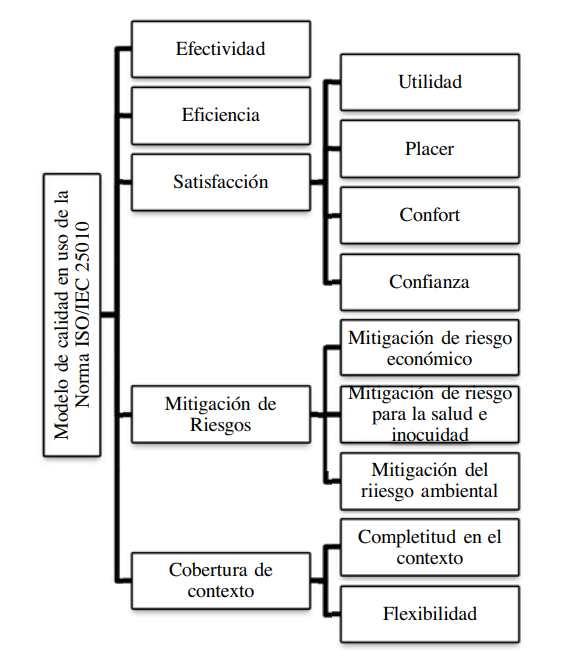


Ilustración 5 Modelo de calidad de uso de la Norma ISO/IEC 25010 (Cruz et al., 2020).

### 1.4.2 Calidad del producto

La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente estos requisitos (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) (ISO, 2015).

La eficiencia del desempeño incluye el comportamiento del tiempo, la utilización de recursos y la capacidad. En el caso de los productos de software, es posible que las medidas tomadas durante el desarrollo no utilicen la plataforma o el entorno exactos del adquirente, por lo que la extrapolación al servicio prestado puede resultar difícil (Estdale & Georgiadou, 2018).

### 1.4.3 Compactibilidad

Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software (ISO, 2015).

Esto se divide en interoperabilidad entre aplicaciones - el intercambio y uso de información, y la coexistencia - el impacto en otros productos que comparten la misma plataforma. Tenga en cuenta que estos se refieren a cómo se implementa la funcionalidad, por lo que quizás sea mejor considerarlo como "características", no como funciones. La coexistencia no es un concepto puramente pasivo. Incluso después de la llegada de los sistemas operativos de programación múltiple debidamente protegidos en las PC, sigue siendo importante que un producto se "comporte bien" en lugar de "se comporte mal", por lo que debe utilizar los servicios suministrados para compartir recursos de manera cooperativa con las otras aplicaciones (no controladas) de la plataforma (Estdale & Georgiadou, 2018).

### 1.4.4 Usabilidad

Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones (Fierro et al., 2019) .

Admite aplicar calidad antes de que se lance el producto, lo que permite medir la calidad en uso como una propiedad emergente. la usabilidad se puede medir mediante las subcaracterísticas relevantes del producto: idoneidad, reconocibilidad, facilidad de aprendizaje, operatividad, protección contra errores del usuario, estética de la interfaz de usuario y accesibilidad. El reconocimiento de la idoneidad se describe como "el grado en que los usuarios pueden reconocer si un producto o sistema es apropiado para sus necesidades", por lo que cubre la literatura de marketing del proveedor. La disponibilidad de capacitación formal y ayuda en línea ayuda en varias áreas, pero son características de la solución del proveedor y lo importante es la facilidad de uso resultante del producto (Estdale & Georgiadou, 2018).

### 1.4.5 Fiabilidad

Esto incluye disponibilidad y recuperabilidad. En el mundo real, rara vez se trata de funciones puras de la aplicación, sino de aspectos del servicio general, que obligan a los usuarios humanos y al personal de apoyo a cooperar para manejar lo que la aplicación no hace por sí misma (Estdale & Georgiadou, 2018).

Capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo determinados. (ISO, 2015).

### 1.4.6 Seguridad

la seguridad se utiliza para evaluar hasta qué punto un sistema protege la información y los datos (Sekarini et al., 2020).

Los clientes de hoy exigen mecanismos integrados para controlar el acceso, garantizar la integridad de los datos y proteger la confidencialidad. En un teléfono inteligente, gran parte de la seguridad que rodea a un servicio la proporciona el entorno operativo de uso, que coexiste con las otras aplicaciones y configuraciones elegidas por el propietario de la plataforma (Estdale & Georgiadou, 2018).

### 1.4.7 Mantenibilidad

Esta característica representa la capacidad del producto software para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas (ISO, 2015).

ISO / IEC 25010 define la mantenibilidad como "el grado de efectividad y eficiencia con el que un producto ... puede ser modificado por los mantenedores previstos". La descripción de la capacidad de mantenimiento incluye las modificaciones realizadas por “personal comercial u operativo, o usuarios finales”, por lo que incluiría la implementación de correcciones del proveedor (Estdale & Georgiadou, 2018).

### 1.4.8 Portabilidad

Capacidad del producto o componente de ser transferido de forma efectiva y eficiente de un entorno hardware, software, operacional o de utilización a otro. El software debe ser adaptable y tener buena capacidad para ser instalado y reemplazado (ISO, 2015) .

# CAPÍTULO II: DESARROLLO

# CAPÍTULO III: RESULTADOS

# Conclusiones

# Recomendaciones

# Glosario

# Referencias

Anwer, F., Aftab, S., Shah Muhammad Shah, S., Waheed, U., Shah, S. M., & Waheed, U. (2017). Comparative analysis of two popular agile process models: extreme programming and scrum. *International Journal of Computer Science and Telecommunications*, *8*(2), 1–7. www.ijcst.org

Auribox. (2017). *¿Qué son los cubos OLAP? - Auribox Training*. https://blog.auriboxtraining.com/desarrollo-web/microservicios/

Cevallos, K. (2016). *Metodología de Desarrollo Ágil: XP y Scrum – Ingeniería del software*. https://ingsotfwarekarlacevallos.wordpress.com/2015/05/08/metodologia-de-desarrollo-agil-xp-y-scrum/

Cruz, M. L. M. H., Segovia, M. E. C. G. M. E., Alvarez, M. D. C. M., Chan, M. J. R. C., Gonzalez, M. E. A. J. A. G., & Francisco Javier Barrera Lao, M. A. C. (2020). Analysis of the Quality in Use and Greenability with the ISO/IEC 25010 Standard. *Iberian Conference on Information Systems and Technologies, CISTI*, *2020*-*June*. https://doi.org/10.23919/CISTI49556.2020.9141017

DOC, D. (2021). *About Docker - Management & History | Docker*. https://www.docker.com/company

DOC, M. (2021). *¿Qué es MongoDB? | MongoDB*. https://www.mongodb.com/es/what-is-mongodb

DOC, N. (2021). *Acerca | Node.js*. https://nodejs.org/es/about/

DOC, R. (2021). *¿Qué es Docker?* https://www.redhat.com/es/topics/containers/what-is-docker

Estdale, J., & Georgiadou, E. (2018). Applying the ISO/IEC 25010 Quality Models to Software Product. *Communications in Computer and Information Science*, *896*, 492–503. https://doi.org/10.1007/978-3-319-97925-0\_42

Fierro, F. A. S., Manosalvas, C. A. P., Rodríguez, N. N. C., & Landeta, P. (2019). Performance efficiency analysis in augmented reality applications using the ISO / IEC / 25010 standard. *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, *2019*(E22), 256–267.

Hovorushchenko, T. (2018). Methodology of evaluating the sufficiency of information for software quality assessment according to ISO 25010. *Journal of Information and Organizational Sciences*, *42*(1), 63–85. https://doi.org/10.31341/jios.42.1.4

ISO. (2015). Iso 25010. In *Iso 25000* (p. 3). https://iso25000.com/index.php/normas-iso-25000/iso-25010

Montero, B. M., Cevallos, H. V., & Cuesta, J. D. (2018). Espirales revista multidisciplinaria de investigación. *Espirales Revista Multidisciplinaria de Investigación*, *2*(17). http://revistaespirales.com/index.php/es/article/view/269

Navarro, G., Nelson, B., & Tataryan, B. (2017). *Integración y evolución de sistemas de información del DCC*. Universidad de Chile. http://repositorio.uchile.cl/handle/2250/148981

Sekarini, D., Alfiani, F. S., & Rochimah, S. (2020). Security Characteristic Evaluation of New Students Admission Information System Based on ISO/IEC 25010 Quality Standard. *ICITEE 2020 - Proceedings of the 12th International Conference on Information Technology and Electrical Engineering*, 120–124. https://doi.org/10.1109/ICITEE49829.2020.9271756

Vayghan, L. A., Saied, M. A., Toeroe, M., & Khendek, F. (2021). A Kubernetes controller for managing the availability of elastic microservice based stateful applications. *Journal of Systems and Software*, 110924. https://doi.org/10.1016/j.jss.2021.110924

# Anexos