Facultad 2

Sistema informático para el soporte del proceso de evaluación de la gestión de infraestructuras TI

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autores: Henry Álvarez Díaz
Fabian Govin Morales
Tutores: Asis. Ing. Víctor Alejandro Roque Domínguez
P.T.Ing**.** Mónica Peña Casanova, Dra. C.

La Habana, diciembre de 2023
Año 65 de la Revolución

DECLARACIÓN DE AUTORÍA

Los autores del trabajo de diploma con título ***"Sistema informático para el soporte del proceso de evaluación de la gestión de infraestructuras TI"*** conceden a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declaran como únicos autores de su contenido. Para que así conste firman la presente a los ____ días del mes de diciembre del año 2023.

Henry Álvarez Díaz

_____
Firma del Autor

Fabian Govin Morales

_____
Firma del Autor

Ing. Víctor Alejandro Roque Domínguez

_____
Firma del Tutor

Dra. C. Mónica Peña Casanova

_____
Firma del Tutor

DATOS DE CONTACTO

Autor: Henry Álvarez Díaz

Correo: henryad@estudiantes.uci.cu

Institución: Universidad de las Ciencias Informáticas

Autor: Fabián Govin Morales

Correo: fabiangm@estudiantes.uci.cu

Institución: Universidad de las Ciencias Informáticas

Tutor: Asis. Ing. Víctor Alejandro Roque Domínguez

Correo: varoque@uci.cu

Institución: Universidad de las Ciencias Informáticas

Resumen: Graduado de Ingeniería en Ciencias Informáticas en el 2016. Profesor Asistente. Con 7 años como docente, ha impartido en pregrado las asignaturas de: Teleinformática, Redes y Seguridad Informática, Redes de computadoras, y Aplicaciones y Servicios Telemáticos.

Tutora: P.T.Ing**.** Mónica Peña Casanova, Dra. C.

Correo: monica@uci.cu

Institución: Universidad de las Ciencias Informáticas

Resumen: Graduada de Ingeniería en telecomunicaciones y equipos y componentes electrónicos 1997. Profesor Titular. Máster en Telemática. Doctora en Ciencias Técnicas. Decana de la Facultad 2 en la Universidad de las Ciencias Informáticas. 23 años de docente en Gestión de Redes y Servicios de Telecomunicaciones, Arquitectura de Redes TCP/IP, Teleinformática, Redes y Seguridad Informática, Configuración de Equipamiento Activo de Redes.

AGRADECIMIENTOS

Por parte de Henry:

A mi madre, que sé que ha soñado este momento tanto como yo y que gracias a ella se ha hecho realidad.

A mi padre que siempre ha sido un gran ejemplo para mí, gracias por tu comprensión y por saber esperar pacientemente.

A mis tutores por saber guiarnos con mucho ímpetu, por brindarnos su conocimiento y ser parte del este proceso de capacitación. A todos mis profesores que de alguna manera aportaron su granito de arena, y que sepan que siempre los llevaré presentes en mi vida profesional

A mis amistades, pues sin ellos esto tampoco fuese posible, gracias por esos bellos momentos que fueron muchos y gracias por esta hermosa etapa en mi vida.

Por parte de Fabián:

Mis padres y mi hermano, las palabras se quedan cortas por guiarme en este camino de la vida a desarrollarme como persona con valores, dedicación y visión a un buen camino como base una buena educación a lograr este momento que tanto ha sido anhelado sirviendo como forma de enorgullecerlos por su sacrificio y confianza, tuvimos muchas dificultades y sin embargo velaban porque tuviera lo necesario para seguir adelante y no rendirme.

A mis tutores por guiarnos en el desarrollo de este proyecto con sus dotes y conocimientos sobre las materias que desarrollamos hasta cumplir con el objetivo propuesto. A mis maestros, por impartirnos buenas clases para desarrollar nuestros conocimientos y visión en los diferentes campos de esta carrera, por el sacrificio de esas consultas a altas horas de las noches para salir bien en las pruebas.

Mis amistades, que son como hermanos para mí e influyeron en el desarrollo del amor por la UCI con: esos momentos vividos, esa hermandad es de los mejores regalos que me llevo.

A los compañeros de aula, y a los que continuaron junto a mí a través de los años que luego compartimos apartamento, les agradezco por todo la ayuda y bondad ofrecida en ciertos momentos de dudas y vicisitudes. En especial agradezco a mi compañero de tesis que tuvo gran papel en el desarrollo de este proyecto.

Por último, a mi pareja, que en tan poco tiempo se ha comportado de forma estelar y desinteresada apoyándome y dándome alientos, amor, afecto y fuerza para sentirme tan bien. Hemos luchado codo a codo en todo momento, sin excusas y justificaciones, agradezco su presencia en mi vida y sus ganas a los deseos y planes.

RESUMEN

En la actualidad, las infraestructuras de Tecnología de la Información (TI) desempeñan un papel fundamental en el funcionamiento y éxito de las empresas y organizaciones. Son responsables de garantizar el correcto funcionamiento de los sistemas. Para evaluar las TI en estas entidades, es necesario contar con capacidades o al menos una hoja de ruta para su desarrollo, así como algún modelo de referencia que permita determinar el estado actual de la organización en cuanto a la gestión TI y hacia dónde deben dirigirse los esfuerzos para obtener el máximo beneficio.

El mayor problema surge al realizar el seguimiento manual de este proceso, lo cual resulta largo y tedioso. Además, se pueden perder documentos importantes que contienen los resultados de las pruebas realizadas. Por lo tanto, surge la necesidad de desarrollar un sistema informático que gestione la evaluación de la infraestructura TI en pequeñas y medianas empresas utilizando modelos de madurez.

Durante el proceso de desarrollo del software, se aplicó la metodología XP en todas sus etapas. Como resultado, se logró implementar un sistema informático que administra de manera eficiente el proceso evaluativo en organizaciones, utilizando modelos de madurez y permitiendo emitir evaluaciones a los usuarios. Las pruebas demostraron que el sistema funciona correctamente y es capaz de resolver el problema planteado.

Palabras clave: evaluación de la gestión TI, gestión de infraestructuras TI, métricas para evaluar, sistemas informáticos.

Abstract

Currently, Information Technology (IT) infrastructures play a fundamental role in the operation and success of companies and organizations. They are responsible for ensuring the correct functioning of the systems. To evaluate IT in these entities, it is necessary to have capabilities or at least a roadmap for its development, as well as some reference model that allows determining the current state of the organization in terms of IT management and where they should go efforts to obtain maximum benefit.

The biggest problem arises when manually tracking this process, which is long and tedious. In addition, important documents containing the results of the tests carried out may be lost. Therefore, the need arises to develop a computer system that manages the evaluation of IT infrastructure in small and medium-sized companies using maturity models.

During the software development process, the XP methodology was applied in all its stages. As a result, it was possible to implement a computer system that efficiently manages the evaluation process in organizations, using maturity models and allowing users to issue evaluations. The tests showed that the system works correctly and is capable of solving the problem posed.

Keywords: *computer systems, IT management evaluation, IT infrastructure management, metrics to evaluate.*

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En los últimos años se ha apreciado un incremento del uso de las tecnologías de la información en las organizaciones, haciéndolas cada vez más dependientes. Por lo que el entorno actual exige servicios que se gestionen de forma eficiente, con calidad y que se adapten proactivamente a los cambios tecnológicos en constante evolución, para satisfacer las necesidades del cliente ofreciendo así una ventaja competitiva a la organización.

(Piattini et al., 2007) destaca que la experiencia ha demostrado que la gestión de servicios TI y la calidad en el nivel de servicio no es algo que se pueda obtener únicamente con fuertes inversiones en tecnología o personal altamente cualificado, sino que es el resultado de una gestión y planificación a nivel empresarial. Su éxito proviene del esfuerzo en construir una infraestructura adecuada e implantar una serie de procesos y prácticas de gestión; así como también, de potenciar la labor de los gestores y la utilización de las métricas para el seguimiento y control del progreso.

En la actualidad, las infraestructuras de Tecnología de la Información (TI) juegan un papel fundamental en el funcionamiento y éxito de las empresas y organizaciones. Estas infraestructuras son responsables de garantizar el correcto funcionamiento de los sistemas, redes y aplicaciones que permiten a las organizaciones llevar a cabo sus operaciones diarias.

Dado el gran nivel de crecimiento que presentan las empresas cada año, se impone la necesidad de contar con procesos de evaluación de madurez que aseguren el cumplimiento de los estándares de calidad y eficiencia necesarios. Estos procesos permiten evaluar el nivel de desarrollo y capacidad de las organizaciones en cuanto a la gestión de sus infraestructuras TI basados en una serie de criterios y mejores prácticas establecidas por expertos en el campo, que permiten medir el grado de eficacia y eficiencia con el que una organización gestiona sus recursos tecnológicos.

Al evaluar la madurez de las organizaciones en cuanto a la gestión de sus infraestructuras TI, se pueden identificar posibles riesgos o deficiencias que podrían afectar negativamente su desempeño. Esto permite tomar acciones preventivas o correctivas para minimizar estos riesgos y garantizar un funcionamiento óptimo de las infraestructuras.

El modelo de madurez es el grado en el que una compañía asimila o integra buenas prácticas en lo que respecta a la dirección de diversos programas o proyectos. Comprende

diversos factores, como herramientas de medición, criterios de evaluación, entre otros. Además, es muy utilizado para realizar y refinar los procesos de desarrollo de software. Mediante este modelo, se pueden analizar todas las competencias en cuanto a dirección de proyectos, para luego compararlas con estándares nacionales o internacionales. De esta comparación se puede establecer qué es lo que se necesita cambiar o mejorar. (Razón Social: Universidad ESAN , 2023)

El éxito de la introducción de las TI en las organizaciones está determinado por la gestión que se realiza de las mismas, a través de la habilitación de un conjunto de sus capacidades utilizando un grupo de facilitadores que forman parte de las prácticas de gestión. Se hace imperativo, entonces, el alineamiento de la infraestructura TI a los objetivos de la organización, por lo que se transita desde elementos de la alta gerencia hasta los aspectos técnicos para la implementación y explotación de las tecnologías, y la experiencia del usuario final para el que se habilita esta.

Para lograr este alineamiento en entornos de determinada complejidad caracterizados por dispositivos heterogéneos y dispersos, con cortos períodos de obsolescencia; múltiples servicios sometidos a variados requerimientos regulatorios; y usuarios que hacen uso de ellos desde contextos diversos, trabajadores que definen las actividades técnicas para la explotación de estas, y regidos por los intereses del área directiva de la organización. Se necesitan capacidades o al menos una hoja de rutas para el desarrollo de estas, o algún modelo de referencia que permita establecer en qué estado está la organización respecto a la gestión TI y hacia donde se deben dirigir los esfuerzos para sacar el mayor partido a las infraestructuras en pos del cumplimiento de los objetivos.

La gestión de la infraestructura TI y su alineación con los procesos de la organización puede planificarse empleando un modelo de madurez. Estos modelos forman la base para comunicar y extender capacidades; y ayudan en la construcción de planes. Los modelos de madurez para gestión TI constituyen un método para evaluar y medir el estado de la gestión TI en las organizaciones, además describen las capacidades de una gestión efectiva y en particular brinda una visión de dónde se debe mejorar.

Durante el proceso de evaluación y emisión de los resultados se recoge información que se analiza, y se tienen como base para tomar decisiones y establecer los planes de mejoras. También se deben recoger los resultados de la medición de los indicadores y los datos

relacionados con el estado de la infraestructura TI. Por lo que la disponibilidad de los datos es muy importante en la aplicación del modelo de madurez.

De la situación planteada anteriormente, surge como **problema de investigación:**

¿Cómo agilizar el proceso de evaluación de la madurez de las infraestructuras TI en pequeñas y medianas organizaciones?

Partiendo de esa problemática se define como **objeto de estudio:** modelos de madurez para la evaluación de la gestión de infraestructuras TI. Para resolver la problemática identificada se ha propuesto como **objetivo general:** Desarrollar un sistema informático para gestionar la evaluación de la infraestructura TI en pequeñas y medianas empresas empleando modelos de madurez, quedando claro el **campo de acción:** soporte para el proceso de evaluación de infraestructuras TI en pequeñas y medianas organizaciones.

Para darle cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos**:

1. Fundamentar los referentes teóricos-metodológicos relacionados con el proceso de evaluación de infraestructuras TI empleando modelos de madurez.
2. Diseñar un sistema informático para la gestión de información relacionada con la evaluación de infraestructuras TI usando modelos de madurez.
3. Implementar el sistema informático teniendo en cuenta los artefactos obtenidos del diseño.
4. Validar el correcto funcionamiento del sistema informático propuesto a partir de los métodos definidos en la investigación.

Para alcanzar los resultados esperados en esta investigación se aplicaron diferentes métodos de investigación:

Métodos de Investigación Teóricos

Estos métodos permiten estudiar las características del objeto de investigación que no son observables directamente, crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad. Para su ejecución se apoyan en el proceso de análisis y síntesis. (León & González, 2020a)

Analítico – Sintético: Se utilizará para el análisis de materiales bibliográficos, herramientas, tecnologías y lenguajes necesarios para desarrollar el sistema informático. Permitirá seleccionar la metodología de desarrollo más apropiada para darle solución a la investigación.

Inductivo – Deductivo: Permitirá llegar al planteamiento del objetivo, además de la extracción de las ideas fundamentales para la elaboración y fundamentación teórica del trabajo de diploma, fue utilizado para el razonamiento de la información consultada, llegando a un grupo de conocimientos particulares y generales sobre los procesos de gestión de información.

Métodos de Investigación Empíricos

Estos métodos describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional.(León & González, 2020a) (León & González, 2020)

Observación: Se utilizará para observar el comportamiento y las características de los sistemas similares.

Entrevista: Se utilizará para validar la propuesta de solución al entrevistar al personal especializado en temáticas como la gestión de procesos de madurez.

El presente documento se encuentra estructurado de la siguiente manera:

Capítulo I: Fundamentos y referentes teórico-metodológicos sobre el sistema informático para el proceso de evaluación de madurez de las organizaciones. En este capítulo se realiza un estudio preliminar de sistemas y funcionalidades haciendo una revisión bibliográfica de la temática actual de estudio, con el objetivo de caracterizar y profundizar las herramientas, lenguajes, tecnologías y metodología para dar cumplimiento al objetivo de la investigación.

Capitulo II: Exploración y diseño de la propuesta de solución. En este capítulo se obtienen las funcionalidades que conformarán la propuesta de solución al objetivo de la investigación. Se definen los requisitos funcionales y no funcionales con los que debe cumplir la propuesta de solución, así como la arquitectura del sistema.

Capitulo III: Implementación y pruebas del sistema. En este capítulo se plantea la construcción de las principales funcionalidades que tendrá el sistema. Se presentan los estándares de programación utilizados y las principales tareas de ingeniería a realizar, así como los resultados de las pruebas realizadas a la aplicación, para comprobar la efectividad.

Capítulo I: Fundamentos teórico-metodológicos sobre el proceso de evaluación de la gestión de infraestructuras TI en las organizaciones

En el presente capítulo se fundamentan conceptos relacionados con el proceso de evaluación de madurez en organizaciones. Se caracterizan las herramientas y tecnologías de desarrollo que se utilizarán para la implementación del sistema, así como elementos teóricos que servirán de base a la investigación del problema planteado.

1.1. Definiciones fundamentales vinculadas a la investigación

En este apartado se presentan las definiciones más relevantes relacionadas con el objetivo de la investigación.

Sistema informático de gestión.

Un sistema informático de gestión es un conjunto integrado de componentes de hardware, software y procesos diseñados para gestionar y coordinar las actividades y recursos de una organización. Estos sistemas están diseñados para automatizar y optimizar tareas y procesos clave, lo que ayuda a mejorar la eficiencia, la toma de decisiones y los resultados empresariales.(Laudon et al., 2012).

Estos sistemas pueden abarcar diferentes áreas funcionales de una organización, como la gestión financiera, la gestión de recursos humanos, la gestión de inventario, la gestión de ventas, la gestión de proyectos, entre otras. Permiten recopilar, almacenar, procesar y analizar datos relevantes para cada área, lo que facilita la toma de decisiones informadas y basadas en datos.

Los sistemas informáticos de gestión pueden incluir diferentes tipos de software, como sistemas de planificación de recursos empresariales (ERP), sistemas de gestión de bases de datos, sistemas de gestión de relaciones con los clientes (CRM), sistemas de gestión de inventario, sistemas de gestión de contenido, entre otros. Estos sistemas se adaptan a las necesidades específicas de cada organización y pueden ser personalizados y configurados según los requisitos y procesos empresariales particulares.(Laudon et al., 2012)

Componentes de un sistema informático de gestión.

Los sistemas informáticos de gestión ayudan a mejorar la eficiencia operativa, reducir errores, gestionar recursos de manera efectiva y brindar información en tiempo real para una toma de decisiones más informada. Los sistemas informáticos de gestión se componen

de varios elementos interrelacionados que trabajan en conjunto para recopilar, almacenar, procesar y analizar datos relevantes para la gestión y la toma de decisiones en una organización, dentro de algunos elementos se encuentran:

- **Hardware:** Incluye los dispositivos físicos necesarios para ejecutar el sistema informático de gestión, como servidores, computadoras, dispositivos de almacenamiento, redes y otros componentes técnicos requeridos para el funcionamiento del sistema.
- **Software:** Consiste en las aplicaciones y programas informáticos que permiten la gestión de las diferentes áreas funcionales de una organización. Estos programas pueden ser desarrollados internamente o adquiridos a proveedores externos. Algunos ejemplos comunes incluyen sistemas de contabilidad, sistemas de recursos humanos, sistemas de gestión de inventario, sistemas de facturación, entre otros.
- **Base de datos:** Es un componente central del sistema informático de gestión que almacena y organiza los datos relevantes para la organización. La base de datos permite el almacenamiento seguro y estructurado de información, que luego es utilizada por el software para realizar consultas, generar informes y facilitar la toma de decisiones.
- **Integración de sistemas:** Los sistemas informáticos de gestión se diseñan para integrarse con otras aplicaciones y sistemas utilizados en una organización. Esto permite la transferencia de datos y la sincronización de información entre diferentes sistemas, evitando la duplicación de datos y asegurando la coherencia de la información en toda la organización.
- **Escalabilidad y flexibilidad:** Los sistemas informáticos de gestión están diseñados para adaptarse al crecimiento y cambio de una organización. Deben ser escalables para manejar volúmenes de datos crecientes y soportar un mayor número de usuarios. También deben ser flexibles para adaptarse a los cambios en los procesos empresariales y las necesidades de la organización, permitiendo personalizaciones y configuraciones según sea necesario.

Modelo de Madurez.

Los modelos de madurez son herramientas orientadas a la mejora de los procesos en las organizaciones delineando caminos de evolución, son modelos de referencia que evalúan la situación actual de una organización, con el fin de identificar y priorizar medidas de mejora

(Bertolli et al., 2017). Estos modelos indican que existen patrones predecibles en la evolución de las organizaciones, los cuales se encuentran distribuidos en etapas evolutivas, formando así una hoja de ruta que facilita la toma de decisiones, cada una de las etapas tiene un grado de superioridad y complejidad mayor con respecto a la inmediatamente anterior, es así como la etapa inicial representa que una organización tiene pocas capacidades en el dominio considerado, la etapa más alta de madurez representa la madurez total, las intermedias representan un camino de progresión continua de las capacidades de la organización (Dijk, 2017). Dentro de cada etapa existe un conjunto de variables que proporcionan los criterios, características y condiciones que deben cumplirse para avanzar en cada una de ellas y posteriormente alcanzar el nivel de madurez establecido internamente en la organización. En este orden de ideas es de vital importancia el análisis del contexto en el que se encuentra la organización para poder derivar y priorizar recomendaciones en medidas de mejora con el fin de alcanzar niveles de madurez más altos a medida que se dé el cumplimiento de los criterios de cada una de las etapas.(Becker et al., 2010)

Es importante hacer énfasis en que los modelos de madurez proporcionan un plan de acción que es utilizado por las organizaciones al querer implementar buenas prácticas para una o más áreas dentro de sus procesos, partiendo de un estado donde las practicas estén mal definidas e incoherentes a un nivel de innovación continua, este procedimiento ha demostrado ser efectivo a través de una amplia aplicación en organizaciones industriales y gubernamentales, el desempeño organizacional se puede medir comparando las practicas institucionales con las practicas esenciales obtenidas de los modelos con el fin de establecer, implementar, evaluar y mejorar los procesos dentro de una organización. (Duarte & Martins, 2013)

Infraestructuras TI.

Las empresas y las instituciones basan, hasta cierto punto, el desarrollo y la entrega de sus productos y/o servicios en las Tecnologías de la Información (TI), por lo que elegir la infraestructura de TI adecuada, para mantener un nivel de servicio apropiado, forma parte muy importante de su estrategia de desarrollo. Gestionar los dispositivos que forman parte de la infraestructura de TI es una tarea que puede resultar muy complicada, sobre todo cuando se trata de un gran número de dispositivos y, además, éstos se encuentran dispersos geográficamente. (González Trejo, 2021)

1.2. Estado actual de soluciones informáticas para el soporte de la gestión TI y su evaluación

Se decide juzgar los sistemas que se relacionen con el objetivo del presente trabajo. En el estudio se tendrán en cuenta sistemas internacionales.

1.2.1. Sistemas internacionales

BMC Remedy IT Service Management Suite: BMC Remedy IT Service Management Suite es una solución de gestión de servicios de TI (ITSM, por sus siglas en inglés) desarrollada por BMC Software. Esta suite de software está diseñada para ayudar a las organizaciones a gestionar eficientemente sus servicios de TI, automatizar procesos, mejorar la resolución de incidencias y problemas, y optimizar la entrega de servicios a los usuarios finales.

Algunas características y funcionalidades clave de BMC Remedy IT Service Management Suite incluyen:

- **Gestión de incidentes:** Permite registrar, rastrear y gestionar de manera efectiva los incidentes reportados por los usuarios, asegurando una resolución rápida y eficiente.
- **Gestión de problemas:** Ayuda a identificar las causas raíz de los problemas recurrentes y facilita la implementación de soluciones permanentes para prevenir futuras interrupciones en los servicios de TI.
- **Gestión de cambios:** Facilita la planificación, aprobación y ejecución de cambios en la infraestructura de TI, garantizando una gestión controlada y minimizando los riesgos asociados con los cambios.
- **Gestión de activos y configuración:** Proporciona un inventario completo y preciso de los activos de TI de la organización, permitiendo un seguimiento adecuado de su ciclo de vida, así como la gestión de configuraciones y relaciones entre los diferentes elementos de la infraestructura.

ServiceNow: es una plataforma de gestión de servicios empresariales (ESM, por sus siglas en inglés) basada en la nube. Ofrece una amplia gama de aplicaciones y soluciones para la gestión de servicios de TI, recursos humanos, servicio al cliente, seguridad, desarrollo de aplicaciones y más. ServiceNow se utiliza en organizaciones de diversos tamaños y sectores para automatizar y agilizar sus procesos empresariales.

Algunas características y funcionalidades clave de ServiceNow incluyen:

- **Gestión de servicios de TI (ITSM):** Incluye funciones como la gestión de incidentes, problemas, cambios, activos y configuraciones, así como la gestión de solicitudes de servicio y la gestión del conocimiento.
- **Gestión de servicios empresariales (ESM):** ServiceNow se extiende más allá de los servicios de TI y ofrece soluciones para la gestión de recursos humanos, gestión de instalaciones, gestión de finanzas, gestión de marketing, entre otros. Estas aplicaciones permiten a las organizaciones gestionar de manera eficiente sus servicios y recursos en diferentes áreas funcionales.
- **Integración y gestión de servicios en la nube:** ServiceNow facilita la integración de servicios en la nube y la gestión de múltiples proveedores de servicios, lo que permite a las organizaciones administrar y controlar de manera centralizada los servicios y contratos.
- **Análisis y generación de informes:** ServiceNow proporciona capacidades de análisis y generación de informes que permiten a las organizaciones obtener información sobre el rendimiento de los servicios, identificar tendencias, tomar decisiones informadas y cumplir con los requisitos de informes y auditorías.

HP Service Manager: es una solución de gestión de servicios de TI (ITSM) desarrollada por Hewlett-Packard (HP), ahora conocida como Micro Focus. HPSM es una plataforma integral que ayuda a las organizaciones a gestionar y mejorar sus servicios de TI, optimizar los flujos de trabajo y ofrecer una mejor experiencia al cliente.

Algunas características y funcionalidades clave de HP Service Manager incluyen:

- **Gestión de incidentes:** Permite registrar, gestionar y resolver los incidentes reportados por los usuarios de manera eficiente, asegurando una rápida restauración del servicio.
- **Gestión de problemas:** Ayuda a identificar las causas raíz de los problemas recurrentes y facilita la implementación de soluciones permanentes para prevenir futuras interrupciones en los servicios de TI.
- **Gestión de activos y configuración:** Proporciona un inventario completo y preciso de los activos de TI de la organización, permitiendo un seguimiento adecuado de su ciclo de vida y la gestión de configuraciones.

- **Gestión del conocimiento:** Facilita la creación, almacenamiento y búsqueda de base de conocimientos, permitiendo a los usuarios acceder a soluciones y mejores prácticas para resolver problemas comunes.

Necesidad de desarrollo de un nuevo sistema: Acceder a estos sistemas supone un desafío para Cuba, ya que, aunque la mayoría son sistemas privados o de pago, también son de uso en línea.

1.3. Selección de la metodología de desarrollo de software

En este epígrafe se realiza el análisis del contexto para el desarrollo del sistema informático para, a partir, de las características principales seleccionar la metodología de desarrollo de software adecuada.

1.3.1. Selección del enfoque para el desarrollo del sistema informático

Para obtener una guía sobre cuáles aspectos evaluar para la selección del enfoque de desarrollo, se empleó el método propuesto por Boehm y Turner (2003). Ellos plantean que se debe conciliar un balance entre los enfoques ágiles y basados en planes, y cada uno tiene sus fortalezas en campos específicos de aplicación. No obstante, no descartan la posibilidad de decantarse por un enfoque u otro.

Según (Boehm & Turner, 2003, p. 33) existen cinco factores críticos para el análisis de del enfoque a seleccionar, estos son: tamaño del equipo, criticidad del producto, dinamismo de los cambios, cultura del equipo y personal con que se cuenta. A continuación, se provee una descripción breve de cada uno de los factores anteriormente mencionados (Rodríguez et al., 2013):

Tamaño: Este criterio se utiliza para representar la cantidad de personas involucradas. Pueden tenerse en cuenta el nivel de complejidad que pueda presentarse en la comunicación entre los miembros del proyecto y los costos que pueden provocar cambios esperados.

Criticidad: Se utiliza para evaluar la naturaleza del daño ocasionado por defectos que no hayan sido detectados al producto. Su evaluación puede ser cualitativa.

Dinamismo: Representa la rapidez con la que pueden estar cambiando los requerimientos del proyecto.

Personal: Representa la proporción del personal con experiencia alta, media y baja. Los métodos orientados al plan no se ven afectados negativamente por este factor pues no interesa el nivel de experiencia con la que cuenten los miembros del equipo.

Cultura: Las organizaciones y las personas que relaciona el proyecto pueden depender de la confianza o de la relación contractual. Esto refleja el nivel de ceremonia necesario y aceptado: documentación, control, formalismo en las comunicaciones.

Evaluación de los factores según las características del proyecto

Tamaño: el equipo para el desarrollo es de 3 integrantes.

Criticidad: según el área que abarca el proyecto y el impacto se puede clasificar como de utilidad.

Personal: según los niveles de habilidad y conocimiento de los recursos humanos del proyecto, se procede a determinar el por ciento del total que representa cada nivel.

Tabla 1 Representación de los recursos humanos por nivel

Nivel	Cantidad	Porciento
1B	0	0%
1A	2	66,7%
2	1	33,3%

Dinamismo: se espera que los requerimientos definidos no sufran demasiados cambios pues el cliente tiene bien definido el funcionamiento del sistema.

Cultura: con la experticia declarada según los niveles de los integrantes del equipo del proyecto la adaptación a los cambios se puede definir como adecuada.

Una vez analizados todos los factores la Figura 1 muestra la representación de los criterios analizados en la Estrella de Boehm y Turner en el proyecto.

La forma obtenida sugiere con claridad la aplicación de un enfoque ágil pues los vértices que representan los valores de Personal, Criticidad y Tamaño, se ubican en territorio ágil. Por otro lado, el criterio de Dinamismo apunta a la utilización de un enfoque tradicional y la Cultura se muestra en un área de incertidumbre donde la agilidad y el formalismo se encuentran balanceados. Por lo que los autores de la investigación definen como el enfoque a seguir el ágil.

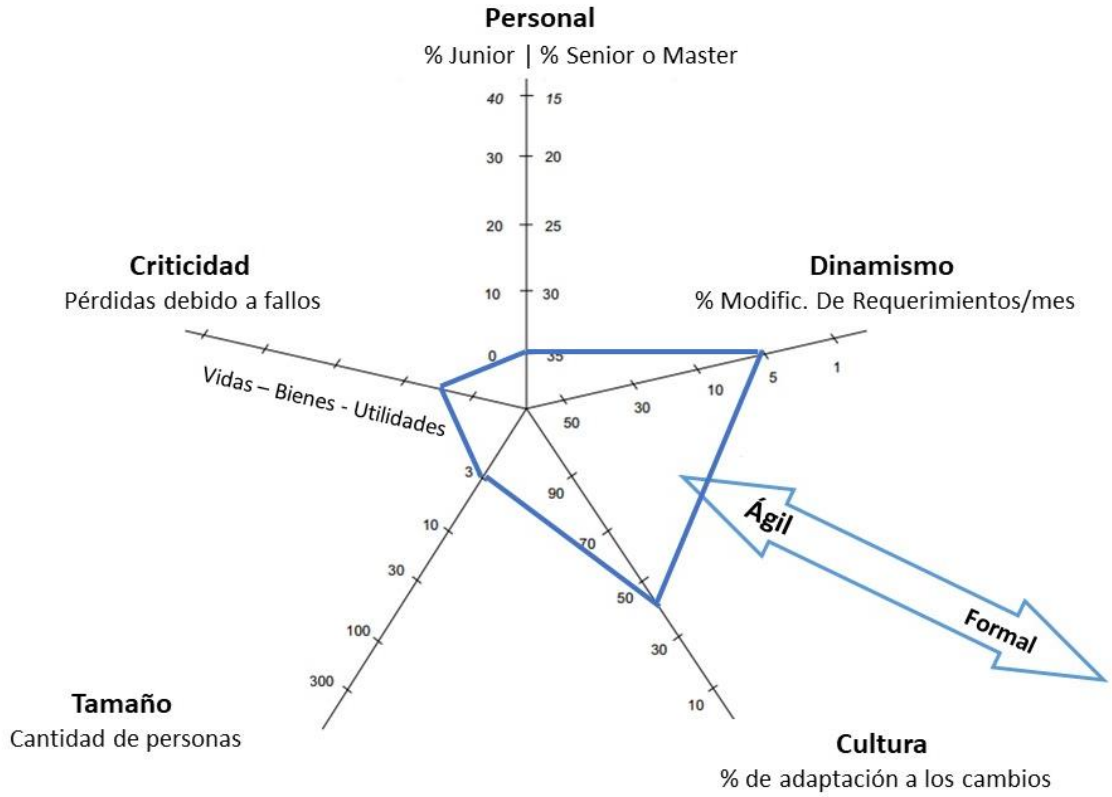

Figura 1 Estrella Boehm y Turner del proyecto

1.3.2. Metodología de desarrollo de software

Pero para poder hablar de calidad, se tuvo que generar un gran proceso histórico que sigue en constante evolución, como son las Metodologías de Desarrollo de Software. Estas proponen como objetivo principal presentar un conjunto de técnicas tradicionales, modernas y ágiles de modelado de sistemas que permitirían desarrollar un software con calidad, Incluyendo heurísticas de construcción y criterios de comparación de modelos de sistemas (Egas & Játiva, 2008).

Teniendo en cuenta el subepígrafe 1.3.1, a continuación, se analizan metodologías de desarrollo de software ágiles con el fin de seleccionar la adecuada para el presente trabajo.

1.3.3. Metodologías ágiles

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 (Cadavid et al., 2013).

Entre las metodologías ágiles más utilizadas tenemos a XP y a SCRUM:

Tabla 2 Comparación entre las metodologías ágiles SCRUM y XP

Características	SCRUM	XP
Más enfocado en los procesos		
Más enfocado en las personas	X	X
Resultados rápidos	X	X
Cliente activo	X	X
Manejo del tiempo	X	X
Refactorización del código		X
Iterativo	X	X
Respuesta a los cambios	X	X

A partir de las características que se reflejan en la tabla anterior, los autores seleccionan XP como metodología de desarrollo para la implementación del sistema informático.

1.3.4. Metodología Programación Extrema (XP)

La metodología extreme programming o XP 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. Los defensores de XP creen, que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos (Siami-Namini et al., 2018)

Características de la metodología XP:

- **Comunicación constante:** Fomenta la comunicación continua y efectiva entre todos los miembros del equipo de desarrollo.

- **Desarrollo iterativo e incremental:** Se basa en ciclos cortos de desarrollo llamados iteraciones. El equipo de desarrollo trabaja en pequeñas unidades de trabajo y realiza entregas frecuentes al cliente para obtener su retroalimentación temprana.
- **Diseño simple y refactorización:** Se enfatiza la simplicidad en el diseño del software. En XP, se busca mantener un código limpio y de alta calidad a través de la refactorización continua.
- **Integración continua:** XP promueve la integración continua del código, lo que implica fusionar y probar regularmente los cambios realizados por los desarrolladores en un repositorio central.
- **Enfoque orientado al cliente:** XP pone un fuerte énfasis en la participación y colaboración del cliente.

1.3.5. Fases de la metodología XP

Fase 1 – Exploración.

Es una actividad para recabar requerimientos que permite que los miembros técnicos del equipo XP entiendan el contexto del negocio para el software y adquieran la sensibilidad de la salida y características principales y funcionalidad que se requieren. El proyecto comienza recopilando las HU, las que constituyen a los tradicionales casos de uso. Una vez obtenidas estas HU, los programadores evalúan rápidamente el tiempo de desarrollo de cada una (PRESSMAN, 2010).

- **Historias de Usuarios (HU):** Las Historias de Usuario representan una breve descripción del comportamiento del sistema, constituyen una herramienta para capturar y comunicar los requisitos del software desde la perspectiva del usuario final, enfocándose en el valor que se espera obtener con cada funcionalidad.
- **Iteraciones:** Como XP se desarrolla en etapa, facilita su realización; a estas etapas se les llama iteraciones y las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en estos ciclos de iteraciones, de acuerdo al orden preestablecido

Fase 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. XP estimula el uso de las tarjetas CRC (clase-responsabilidad-colaborador) como un mecanismo eficaz

para pensar en el software en un contexto orientado a objetos. Las tarjetas CRC identifican y organizan las clases orientadas a objetos que son relevantes para el incremento actual de software (PRESSMAN, 2010)

- **Tarjetas CRC:** Las tarjetas CRC son una técnica utilizada para definir las responsabilidades y colaboraciones entre las clases en el diseño de un sistema de software, facilitando la comunicación y comprensión del diseño entre los miembros del equipo de desarrollo.

Fase 3 - Implementación

La codificación es un proceso que se realiza en forma paralela al diseño y la cual está sujeta a varias observaciones por parte de XP. En esta fase, se implementa el software de acuerdo con las funcionalidades priorizadas. El equipo de desarrollo trabaja en ciclos cortos, generalmente de una o dos semanas, para entregar incrementos funcionales del software. Se utilizan técnicas como la programación en parejas y las pruebas continuas para garantizar la calidad del código. (PRESSMAN, 2010)

Fase 4 – Pruebas

Según XP se debe de ser muy estricto con las pruebas, solo se deberá liberar una nueva versión si ésta ha pasado con el cien por ciento de la totalidad de las pruebas. En caso contrario se emplea el resultado de esta para identificar el error y solucionarlo con mecanismos ya definidos. Las pruebas unitarias que se crean deben implementarse con el uso de una estructura que permita automatizarlas (de modo que puedan ejecutarse en repetidas veces y con facilidad). Esto estimula una estrategia de pruebas de regresión siempre que se modifique el código. Las pruebas de aceptación XP, también llamadas pruebas del cliente, son especificadas por el cliente y se centran en las características y funcionalidad generales del sistema que son visibles y revisables por parte del cliente. Las pruebas de aceptación se derivan de las historias de los usuarios que se han implementado como parte de la liberación del software (PRESSMAN, 2010)

Selección de la metodología: Se escoge por parte del equipo de trabajo la metodología XP debido a su capacidad para facilitar una comunicación fluida entre el cliente y el desarrollador. Además, es una opción accesible para equipos de desarrollo pequeños que necesitan entregar el software incremental en un corto período de tiempo. Esta metodología también es adecuada para desarrolladores con un nivel medio de experiencia.

1.4. Herramientas, lenguajes y tecnologías a emplear en la solución

1.4.1. Lenguajes de desarrollo

Python 3.9.3: es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario "traducirlo" a lenguaje máquina (Mueller, 2023).

Python es un lenguaje sencillo de leer y escribir debido a su alta similitud con el lenguaje humano. Además, se trata de un lenguaje multiplataforma de código abierto y, por lo tanto, gratuito, lo que permite desarrollar software sin límites (Mueller, 2023). Con el paso del tiempo, Python ha ido ganando adeptos gracias a su sencillez y a sus amplias posibilidades, sobre todo en los últimos años, ya que facilita trabajar con inteligencia artificial, big data, machine learning y data science, entre muchos otros campos en auge.

JavaScript 1.6: es un lenguaje de programación interpretado, o sea, no requiere compilación. Es utilizado especialmente en páginas web embebido en el código HTML o similares. La mayoría de los navegadores pueden interpretar los códigos JavaScript incluidos en las páginas web (Kereki, 2023, p. 8).

Con JavaScript se pueden extender las posibilidades de las páginas web como, por ejemplo, evitar que se pueda copiar el texto de una página, botones para agregar automáticamente una página a favoritos, crear barras de scroll, abrir popups, cambiar el puntero del mouse, rotar banners, validar formularios, etc. En la solución se pone de manifiesto este lenguaje en la implementación de las páginas web que serán vistas por el cliente, además de los formularios y las tablas (Kereki, 2023).

HTML 5: esta especificación define la quinta mayor revisión del núcleo del lenguaje de la Word Wide Web: el lenguaje de marcado de hipertexto (HTML). En esta versión, se introducen nuevas características para ayudar a los autores de aplicaciones Web, se introducen nuevos elementos basados en la investigación de las prácticas prevalecientes de autoría, y se ha prestado especial atención a la definición de los criterios de conformidad clara para los agentes de usuario en un esfuerzo por mejorar la interoperabilidad (Rebah et al., 2022).

CSS3: Es un lenguaje de hojas de estilos en cascadas creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos de su presentación y es imprescindible para crear páginas web complejas. Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes (Rebah et al., 2022).

1.4.2. Tecnologías para la investigación

Django versión 4.0.5: Django es un framework de desarrollo web de código abierto escrito en Python. Fue creado para facilitar la creación de aplicaciones web complejas y escalables, siguiendo el patrón de diseño Modelo-Vista-Controlador (MVC). Django proporciona una gran cantidad de funcionalidades y herramientas que permiten a los desarrolladores ahorrar tiempo y esfuerzo al crear aplicaciones web (Melé & Belderbos, 2022). Algunas de las características destacadas de Django incluyen un ORM (Object-Relational Mapping) para interactuar con la base de datos, un sistema robusto de autenticación y autorización, soporte para internacionalización y localización, y una interfaz administrativa preconstruida (Melé & Belderbos, 2022). Django también promueve buenas prácticas de desarrollo web, como la seguridad, el rendimiento y la escalabilidad.

Bootstrap 5.2.0: Es un *framework* de código abierto que utiliza diseños basados en CSS y HTML (bin Uzayr, 2022). Se trata de un marco popular que se ocupa solo de las aplicaciones *frontend* (formularios, componentes de la interfaz y complementos de JavaScript). Bootstrap es uno de los *frameworks frontend* más populares y fue creado por Mark Otto y Jacob Thornton para fomentar la coherencia entre las numerosas bibliotecas disponibles en el mundo del desarrollo web (bin Uzayr, 2022).

SQLite: Es un SGBD (Sistema Gestor de Base de Datos) relacional, famoso por su pequeño tamaño. A diferencia de otros sistemas cliente-servidor el motor de SQLite no es un proceso independiente, lo que hace que la latencia sea menor y el acceso más eficiente. Debido a su facilidad de uso, su pequeño tamaño y su versatilidad SQLite es utilizado en una gran variedad de aplicaciones. Se ejecuta en muchas plataformas, es de dominio público y por tanto sin costo (Siahaan & Sianipar, 2019).

1.4.3. Herramientas de desarrollo

Visual Studio Code v1.84: Es compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un lenguaje dado. Es

un editor de código fuente desarrollado por Microsoft para Windows, Linux; macOS y Web. Incluye soporte para depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código (Kahlert & Giza, 2016). También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es gratuito y de código abierto, aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft. Visual Studio Code se basa en Electron, un frameworks que se utiliza para implementar Chromium y Node.js como aplicaciones para escritorio, que se ejecuta en el motor de diseño Blink (Kahlert & Giza, 2016). Aunque utiliza el frameworks Electron, el software no usa Atom y en su lugar emplea el mismo componente editor (Monaco) utilizado en Visual Studio Team Services.

Conclusiones del capítulo

Con la investigación realizada se llegó a la conclusión de que no se cuenta con un sistema de gestión en el país que permita la evaluación de procesos de madurez de infraestructura TI. La descripción de los principales conceptos posibilitó adquirir una mayor comprensión de los temas relacionados con el objeto de estudio, por otra parte, el análisis de diferentes herramientas y tecnologías permitió alcanzar los conocimientos necesarios para seleccionar las adecuadas para el desarrollo de la solución.

Capítulo II: Exploración y diseño de la propuesta de solución

En este capítulo se muestran las características y funcionalidades del sistema a implementar. Se transita por cada una de las fases que propone la metodología XP, exponiendo así las historias de usuarios (HU) que se encuentran en el sistema y la estimación de su esfuerzo. Como artefactos fundamentales se destacan: las HU, Requisitos funcionales, Plan de entrega, Plan de iteraciones y las Tarjetas de clases-responsabilidad-colaboración (CRC). Al finalizar este capítulo quedará materializada la propuesta del sistema que da cumplimiento al objetivo trazado en la investigación.

2.1. Propuesta de solución

Se propone desarrollar un sistema informático que facilite el proceso de evaluación de madurez en organizaciones. El sistema se encargará de administrar todo el proceso evaluativo, permitiendo evaluar el estado actual de la infraestructura de tecnología de la información y emitir criterios de evaluación al personal responsable. Además, los informes finales generados por este sistema estarán disponibles en formato PDF, lo cual facilitará su utilización fuera del mismo.

Para llevar a cabo el proceso de evaluación, se establecerá un rol de **Administrador** encargado de gestionar los roles, permisos y usuarios en el sistema. Este rol no tendrá acceso a otras partes del sistema. El rol **Experto** podrá gestionar los modelos de madurez y las métricas que se utilizarán en dichos modelos. El rol **Consultor** podrá gestionar las organizaciones que serán evaluadas, seleccionando las métricas correspondientes según el tipo de organización. Por último, el rol **Evaluador** será responsable de emitir un criterio de evaluación a las organizaciones por medio de las métricas seleccionadas anteriormente por el Consultor. Además, este rol será el único con permiso para generar y exportar los resultados en formato PDF.

2.2. Fase 1: Exploración del sistema informático

En esta etapa, se iniciará la comunicación con el cliente con el fin de identificar los requisitos del sistema. Además, se buscará adaptar la metodología utilizada a las características específicas de la solución, realizando las iteraciones y ajustes necesarios.

2.2.1. Requisitos funcionales

Los requisitos funcionales se utilizaron para describir los servicios que se espera que el sistema cumpla para satisfacer las necesidades del usuario. Aportan una visión más detallada de lo que se va a implementar en el sistema. Aunque la metodología de desarrollo seleccionada no incluye los Requisitos Funcionales y No Funcionales como parte de sus artefactos, los autores determinaron que se hacía necesario la identificación de estos requerimientos para una mejor comprensión del negocio.

La *Tabla 3* muestra los distintos requisitos funcionales definidos para el sistema, su descripción y prioridad.

Tabla 3: Requisitos Funcionales

Número	Requisitos	Descripción	Prioridad
RF1	Insertar Modelo de Madurez (nombre, descripción)	Permite crear un nuevo modelo de madurez en el sistema.	Alta
RF2	Mostrar Modelo de Madurez (nombre, descripción, creado por, modificado por)	Permite mostrar un modelo de madurez.	Alta
RF3	Editar Modelo de Madurez (nombre, descripción)	Permite editar un modelo de madurez.	Alta
RF4	Eliminar Modelo de Madurez (id)	Permite eliminar un modelo de madurez.	Alta
RF5	Insertar Métrica (nombre, descripción)	Permite insertar una nueva métrica.	Alta
RF6	Editar Métrica (nombre, descripción)	Permite editar una métrica.	Alta
RF7	Eliminar Métrica (id)	Permite eliminar una métrica.	Alta
RF8	Mostrar Métrica (nombre, descripción,	Permite mostrar una métrica.	Alta

	fecha de registro, fecha de modificación)		
RF9	Insertar Organización (nombre de la organización, dirección, teléfono, correo electrónico, sitio web)	Permite crear una nueva organización en el sistema.	Alta
RF10	Mostrar Organización (nombre de la organización, dirección, teléfono, correo electrónico, sitio web)	Permite mostrar una organización.	Alta
RF11	Eliminar Organización (id)	Permite eliminar una organización.	Alta
RF12	Editar Organización (nombre de la organización, dirección, teléfono, correo electrónico, sitio web)	Permite editar una organización.	Alta
RF13	Autenticar Usuario (usuario, contraseña)	Permite autenticar usuarios al sistema.	Alta
RF14	Insertar Usuario (usuario, contraseña, nombre)	Permite crear un usuario en sistema.	Alta
RF15	Eliminar Usuario (id)	Permite eliminar un usuario.	Alta
RF16	Editar Usuario (usuario, contraseña, nombre)	Permite editar un usuario.	Media
RF17	Mostrar Usuario (usuario, contraseña, nombre)	Permite mostrar un usuario.	Alta

RF18	Generar Reporte	Permite generar un reporte.	Media
RF19	Exportar Reporte	Permite exportar un reporte.	Media

2.2.2. Requisitos no funcionales

Según (Sommerville, 2006), los requisitos no funcionales son aquellos que describen las características y restricciones del sistema, en lugar de sus funciones específicas. Estos requisitos se centran en aspectos como el rendimiento, la seguridad, la usabilidad, la fiabilidad y la escalabilidad del sistema. Los requisitos no funcionales son importantes para garantizar que el sistema cumpla con los estándares de calidad y satisfaga las necesidades de los usuarios.

La *Tabla 4* muestra la lista de los requisitos no funcionales que debe cumplir el sistema informático en cuanto a Usabilidad, Seguridad, Rendimiento y Mantenibilidad.

Tabla 4: Requisitos No Funcionales

Número	Usabilidad
RNF1	El sistema debe poseer una interfaz intuitiva para el usuario y con colores agradables para la vista.
RNF2	El sistema debe asegurar que los tiempos de respuesta sean de manera rápida para una experiencia fluida.
RNF3	El sistema notifica al usuario la existencia de datos incorrectos o incompletos cuando accede a funcionalidades como registrar o modificar.
RNF4	Todos los mensajes para interactuar con los usuarios deben ser lo suficientemente intuitivos y en idioma español.
RNF5	El sistema debe ser compatible con diferentes navegadores web como Google Chrome (v85.0.5 o superior), Mozilla Firefox (v93.0 o superior) y Microsoft Edge (v82.0.21 o superior)
	Seguridad
RNF6	El sistema no permitirá el acceso con datos incorrectos.

RNF7	Realizará la autenticación y autorización de manera correcta para proteger los datos de los usuarios.
RNF8	El sistema se desarrollará con tecnologías seguras.
RNF9	Los usuarios tendrán acceso a los diferentes sitios del sistema en dependencia de su rol.
RNF10	Solo el rol Administrador podrá modificar los privilegios y permisos de los demás usuarios.
	Rendimiento
RNF11	El sistema tendrá tiempos de carga rápidos para minimizar la espera del usuario.
RNF12	Optimizará el uso de los recursos del sistema.
	Mantenibilidad
RNF13	El sistema se implementará mediante código limpio, modular y bien documentado para facilitar futuras actualizaciones.
RNF14	Se le realizaran pruebas al software para garantizar la calidad.
	Entorno
RNF15	La Pc del cliente deberá tener una CPU con microprocesador con al menos 1.0 GHz de velocidad y 256 MB de memoria RAM o superior. Espacio en disco 80 GB

2.2.3. Historias de Usuarios

Las historias de usuario (HU) es la técnica utilizada por XP para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales.

La *Tabla 5* explica cada uno de los componentes utilizados para la elaboración de las historias de usuario.

Tabla 5: Estructura de modelado Historia de Usuario

Historia de Usuario	
Número: Permite identificar a una HU.	**Usuario:** Rol o roles que utilizarán la funcionalidad del sistema descrita en la HU.

Nombre de la Historia: Describe de manera general a una HU	
Prioridad del Negocio: Grado de importancia que representa para el cliente la HU.	**Riesgo en Desarrollo:** Valor en cuanto a nivel de complejidad que una HU representa a los desarrolladores.
Puntos Estimados: Número de semanas que se necesitará para el desarrollo de una HU.	**Iteración Asignada:** Número de iteración en la que se implementa una HU.
Programador Responsable: Persona encargada de programar cada HU.	
Descripción: Información detallada de una HU.	
Observaciones: Campo opcional utilizado para aclarar, si es necesario, la descripción de una HU.	

Las tablas 5, 6 y 7 muestran las historias de usuarios 1, 2 y 7 respectivamente. El resto de las HU se incluyen en el **Anexo I.**

Tabla 6: HU Administrar modelos de Madurez.

Historia de Usuario	
Número: 1	**Usuario:** Experto.
Nombre de la Historia: Administrar modelos de Madurez.	
Prioridad del Negocio: Alta (Alta, Media, Baja)	**Riesgo en Desarrollo:** Medio (Alto, Medio, Bajo)
Puntos Estimados: 2	**Iteración Asignada:** 1
Programador Responsable: Henry	
Descripción: El rol **Experto** podrá gestionar los modelos de madurez (Insertar, Modificar, Eliminar, Editar)	
Observaciones: Solo el rol **Experto** tendrá permisos de modificar los modelos de madurez.	

Tabla 7: HU Administrar Métricas

Historia de Usuario	
Número: 2	**Usuario:** Experto.
Nombre de la Historia: Administrar métricas	

Prioridad del Negocio: Alta (Alta, Media, Baja)	**Riesgo en Desarrollo:** Alto (Alto, Medio, Bajo)
Puntos Estimados: 1	**Iteración Asignada:** 1
Programador Responsable: Henry	
Descripción: El rol **Experto** podrá gestionar las métricas (Insertar, Modificar, Eliminar, Editar)	
Observaciones:	

Tabla 8: HU Exportar Datos

Historia de Usuario	
Número: 7	**Usuario:** Evaluador.
Nombre de la Historia: Exportar datos	
Prioridad del Negocio: Alta (Alta, Media, Baja)	**Riesgo en Desarrollo:** Medio (Alto, Medio, Bajo)
Puntos Estimados: 1	**Iteración Asignada:** 3
Programador Responsable: Henry	
Descripción: El sistema deberá permitir al usuario **Evaluador** exportar los resultados de las evaluaciones a un archivo PDF para su posterior manipulación.	
Observaciones: Solo el usuario **Evaluador** tendrá acceso a la funcionalidad de exportación de los datos.	

2.2.4. Estimación de esfuerzo por HU

Tabla 9: Estimación de Esfuerzos

Número HU	Historias de Usuarios	T. E
1	Administrar modelos de Madurez	**2**
2	Administrar métricas	**1**
3	Administrar organizaciones a evaluar	**2**
4	Administrar usuarios del sistema.	**0.5**
5	Acceder al sistema	**0.5**
6	Generar reporte	**1**

7	Exportar datos a PDF	**1**
Historias de Usuarios Total: 7		**T.E Total: 8**

2.2.5. Iteraciones

Se definen varias iteraciones sobre el sistema antes de ser entregado y un número determinado de historias de usuario a implementar. El plan de entregas está compuesto por iteraciones de no más de tres semanas. Al final de la última iteración el sistema estará listo para entrar en producción.

Nota: Tiempo Estimado (T.E) está dado en semanas. Una semana en XP equivale a 5 días laborales (de lunes a viernes), en jornadas de 8 horas de trabajo por día (8am – 12 m y 1 pm – 5pm).

Después de haber definido las HU y estimado el esfuerzo para la realización de cada una de ellas, se tomó la decisión de dividir el sistema en 3 iteraciones, tal como se muestra en la *Tabla 10*.

Tabla 10: Plan de Duración de las Iteraciones

Iteraciones	Historias de Usuarios	Tiempo Estimado
1	Administrar modelos de Madurez	**3**
	Administrar métricas	
2	Administrar organizaciones a evaluar	**3**
	Administrar usuarios del sistema.	
	Acceder al sistema	
3	Generar reporte	**2**
	Exportar datos a PDF	

2.2.6. Plan de entregas.

La *Tabla 11* muestra el plan de entregas desarrollado para dar solución al problema planteado.

Tabla 11: Plan de Entregas

Número de la Iteración	Duración total de Iteraciones	Fecha Inicio	Fecha Final
1	3 semanas	25/09/2023	13/10/2023
2	3 semanas	16/10/2023	03/11/2023
3	2 semanas	06/11/2023	17/11/2023

2.3. Fase 2: Diseño del sistema informático

En esta sección se realiza la descripción de la arquitectura de software y los patrones de diseño aplicados a la propuesta de solución.

2.3.1. Patrones de arquitectura de software.

La arquitectura de software de un sistema o programa de computación es la estructura o estructuras del sistema, que comprenden elementos de software, las propiedades externamente visibles de estos elementos, y las relaciones entre ellos (Larman, 2003)

Modelo-Vista-Plantilla

El empleo del framework de desarrollo Django, implica una serie de decisiones de diseño. Django sigue una arquitectura Modelo-Vista-Controlador, solo que hace una adaptación de esta a Modelo-Vista-Plantilla (a partir de ahora MTV por sus siglas en inglés, Model Template View). Por tanto, el sistema propuesto hereda una arquitectura MTV.

Modelo

Contiene toda la información sobre los datos. Cada una de las entidades de la base de datos se encuentra en el modelo en forma de clases de Python, y sus atributos se almacenan en variables con ciertos parámetros.

Vista

Es la capa de la lógica de negocios, contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Esta capa sirve de “puente” entre el modelo y la plantilla, se presenta en forma de funciones de Python y su función principal es determinar qué datos serán visualizados en las plantillas.

Plantilla

Define la interfaz de las páginas web, o sea, cómo se van a mostrar los datos al usuario. También posee algunas etiquetas propias del framework Django.

2.3.2. Patrones de diseño

Un patrón de diseño provee un esquema para refinar componentes de un sistema de software y la forma en que se relacionan entre sí. De manera más simple, un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos (Larman, 2003).

Patrones de diseño GRASP

Los patrones GRASP (por sus siglas en inglés, General Responsibility Assignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (Demestre & González, 2019)

Describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones (Larman, 2003)

Los patrones pertenecientes a este conjunto son: Experto, Controlador, Creador, Alta Cohesión y Bajo Acoplamiento.

En la investigación se utilizaron los siguientes patrones:

Patrón Experto: Es el responsable de asignar la responsabilidad de realizar una tarea a la clase que tiene la información necesaria para completarla. El patrón Expert se puede encontrar en las funciones crear_metrica y editar_metrica. En ambas funciones, se utilizan instancias de MetricaForm para crear y editar métricas respectivamente. Estas instancias son las responsables de manejar la lógica y la validación de los datos relacionados con las métricas.

```python
def editar_metrica(request, id):
    metrica = Metricas.objects.get(id=id)

    if request.method == 'POST':
        form = OrganizacionForm(request.POST, instance=metrica)
        if form.is_valid():
            form.save()
            return redirect('listar_metricas', pk=metrica.modelo.id)
    else:
        form = OrganizacionForm(instance=metrica)

    return render(request, 'config/listMetrica.html', {'form': form})

```

Figura 2 Utilización en el código del patrón experto

Patrón Creador: Se utiliza para asignar la responsabilidad de crear instancias de objetos a una clase específica. Esta clase "creadora" tiene la información y el contexto adecuados para crear y configurar correctamente los objetos. Se encuentra en la función crear_modelo, donde se crea una instancia de ModeloForm para crear un nuevo modelo y se asignan responsabilidades de creación de objetos a esta clase.

```python
def crear_modelo(request):
    if request.method == 'POST':
        form = ModeloForm(request.POST)
        if form.is_valid():
            modelo = form.save(commit=False)
            modelo.creado_por = request.user
            modelo.save()

            return redirect('modelo_list')
    else:
        form = ModeloForm()
    return render(request, 'config/crearModelo.html', {'form': form})
```

Figura 3 Utilización en el código del patrón Creador

Bajo Acoplamiento: El principio de Bajo Acoplamiento se refiere a la reducción de las dependencias entre clases y componentes. Esto se logra asignando responsabilidades de manera que las clases dependan lo menos posible entre sí. El bajo acoplamiento facilita la modificación y extensión del código. El principio de Bajo Acoplamiento se cumple en general en el código, ya que las funciones utilizan parámetros y dependencias mínimas. Además, se utilizan funciones y clases proporcionadas por Django que favorecen el bajo acoplamiento, como render, redirect, get_object_or_404, JsonResponse, etc.

```
48  def delete_metrica(request, pk):
49      modelo = get_object_or_404(Metricas, pk=pk)
50      modelo.delete()
51      return JsonResponse({'mensaje': 'El registro ha sido eliminado exitosamente.'})
```

Figura 4 Utilización en el código del patrón bajo acoplamiento

Patrones de diseño GOF

Patrones publicados por Gamma, Helm, Johnson y Vlossodes en 1995: patrones de la banda de los cuatro (del inglés, Gang of Four). Esta serie de patrones permiten ampliar el lenguaje, aprender nuevos estilos de diseño y además introducir más notación UML. Existen 23 patrones GoF de los cuales 15 se utilizan con frecuencia. Los patrones de diseño del grupo GoF se clasifican en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Demestre & González, 2019)

En la investigación se utilizaron los siguientes patrones:

Decorador: Este patrón se evidencia en la clase Login require (decorador incorporado en Django por defecto, para acceder a una clase).

2.3.3. Tarjetas de Clase-Responsabilidad-Colaboración (CRC).

Las *Tablas 12 y 13* muestran las tarjetas CRC para las clases Modelo y Métricas respectivamente. El resto de las tarjetas CRC se incluyen en **Anexo II.**

Tabla 12 Tarjeta CRC clase Modelo

Clase: Modelo	
Responsabilidad:	**Colaboración:**
Almacenar información de un modelo (nombre, descripción, fecha de creación, etc.). Realizar operaciones de creación, edición y eliminación de modelos.	Clase Metricas
Atributos: nombre, descripcion, fecha_registro, fecha_modificacion, creada_por, modificada_por	

Tabla 13 Tarjeta CRC clase Métricas

Clase: Metricas	
Responsabilidad:	**Colaboración:**

Almacenar información de una métrica (nombre, valor, fecha de creación, etc.). Realizar operaciones de creación, edición y eliminación de métricas.	Con las vistas y el formulario MetricaForm relacionado con la creación, edición y eliminación de métricas.
Atributos: nombre, descrip, estado, fecha_registro, fecha_modificacion	

Conclusiones del capítulo

La participación del cliente junto a los desarrolladores como lo plantea la metodología XP, jugó un papel clave para establecer las prioridades en el correcto análisis de la solución sistema informática para el soporte del proceso de evaluación de madurez en infraestructuras TI.

Se elaboraron los artefactos correspondientes a la etapa de exploración de la metodología XP, determinando como resultado que el proceso de desarrollo durará un total de 8 semanas, divididas en tres iteraciones; tres semanas para las iteraciones 1 y 2, y 2 para la iteración 3, de forma tal que el equipo de desarrollo y el cliente obtendrían un producto funcional al concluir cada iteración.

Se confeccionaron además las tarjetas CRC, las cuales permitieron establecer el marco de relaciones para las clases del diseño, lo cual ayudará a una mejor comprensión de la manera en que se encuentra estructurada el sistema.

Capítulo III: Implementación y pruebas del sistema informático

En el presente capítulo se detallan los aspectos fundamentales de los procesos de implementación y pruebas de la solución propuesta en el Capítulo II. Se describen los estándares de codificación utilizados durante la implementación del sistema informático para el soporte del proceso de evaluación en infraestructuras TI, así como las diferentes tareas de ingeniería (a partir de ahora TI) que propone la metodología XP a desarrollarse en las tres iteraciones que conforman el proyecto. También se podrá encontrar la validación de la solución propuesta mediante la aplicación de pruebas a los niveles de unidad, sistema y aceptación.

3.1. Implementación de la solución

En esta fase, se presentan los estándares de codificación utilizados por los programadores en el sistema. Además, las HU presentadas en el capítulo anterior se descomponen en Tareas de Investigación, y son asignadas a uno de los desarrolladores del sistema, por lo que se describen en un lenguaje técnico y no en un lenguaje entendible para el cliente.

3.1.1. Estándares de codificación

Los estándares de codificación son pautas y reglas establecidas para la escritura de código de programación. Estos estándares son utilizados para asegurar que el código sea consistente, legible, fácil de mantener y seguir las mejores prácticas de programación. El objetivo principal de los estándares de codificación es mejorar la calidad del código y facilitar la colaboración entre desarrolladores.

Estructura del código:

- Cada línea de código tiene un límite máximo de 80 caracteres.
- Se utilizan líneas en blanco para separar funciones y clases. Dentro de las funciones para separar secciones lógicas.
- Se utilizan nombres descriptivos y en minúsculas para las variables, funciones y paquetes, separando las palabras con guiones bajos.

```
4   def listar_modelo(request):
5       return render(request, 'listmodelo.html')
6
7   def listar_entidad(request):
8       return render(request, 'listentidad.html')
9
```

Figura 5 Ejemplo de la aplicación de las premisas en la estructura del código

Otros aspectos:

- Ordenar las importaciones alfabéticamente dentro de cada sección.
- Evitar el uso de grandes espacios en blanco innecesarios.
- Utilizar paréntesis alrededor de expresiones condicionales para mejorar la legibilidad.

```
1   from app.modelo.views import listar_modelo
2   from app.modelo.views import listar_entidad
3   from app.modelo.views import login_index
4   from app.modelo.views import evaluaciones
5   from django.urls import path
6
```

Figura 6 Ejemplo de la aplicación de otros aspectos relacionados los estándares de codificación

3.1.2. Tareas de ingeniería

A continuación, se evidencian las tareas ingeniería en las que fueron desglosadas las HU mencionadas anteriormente, el resto de las Tareas de ingeniería se encuentran en el **Anexo III.** Para la confección de cada una de las tareas se utilizó la siguiente tabla:

Tabla 14 Estructura tareas de ingeniería

Tarea de Ingeniería	
Número de tarea: Permite identificar a una Tarea.	**Número de HU:** Número de la HU a la cual pertenece.
Nombre de Tarea: Describe de manera general a una Tarea	
Tipo de Tarea: Tipo de tarea, dígase diseño, desarrollo, prueba	**Puntos Estimados:** Representación de la cantidad de tiempo estimado en días que se utilizará para su realización.
Programador Responsable: Persona encargada de programar cada Tarea.	
Descripción: Información detallada de una Tarea de Ingeniería.	

Tabla 15 Tarea de Ingeniería 1: Mostrar modelo de Madurez

Tarea de Ingeniería	
Número de tarea: 1	**Número de HU:** 1
Nombre de Tarea: Mostrar modelo de Madurez	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Programador Responsable: Henry	
Descripción: Al autenticarse con el rol Experto se muestran los modelos de Madurez existentes en el sistema. (nombre, descripción, creado por, modificado por)	

Tabla 16 Tarea de Ingeniería 2: Insertar nuevo modelo de madurez

Tarea de Ingeniería	
Número de tarea: 2	**Número de HU:** 1
Nombre de Tarea: Insertar nuevo modelo de Madurez	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Programador Responsable: Henry	
Descripción: El usuario Experto podrá insertar un nuevo modelo de madurez en el sistema (nombre, descripción)	

Tabla 17 Tarea de Ingeniería 3: Eliminar modelo de madurez

Tarea de Ingeniería	
Número de tarea: 3	**Número de HU:** 1
Nombre de Tarea: Eliminar modelo de Madurez	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3
Programador Responsable: Henry	
Descripción: El usuario Experto podrá eliminar un modelo de madurez en el sistema. (id)	

Tabla 18 Tarea de Ingeniería 17: Autenticar usuarios

Tarea de Ingeniería	
Número de tarea: 17	**Número de HU:** 5
Nombre de Tarea: Autenticar usuarios	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 5
Programador Responsable: Henry	
Descripción: Se crean las funcionalidades necesarias que permitan autenticar a los usuarios en el sistema. (usuario, contraseña)	

Tabla 19 Tarea de Ingeniería 18: Generar reportes

Tarea de Ingeniería	
Número de tarea: 18	**Número de HU:** 6
Nombre de Tarea: Generar los reportes	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 5
Programador Responsable: Henry	
Descripción: Se crean las funcionalidades necesarias que permitan generar reportes de las evaluaciones emitidas por el usuario Evaluador.	

3.2. Pruebas de software

Según (Sommerville, 2006), las pruebas de software se definen como el proceso de ejecutar un programa con el fin de descubrir errores. Este proceso también incluye la verificación de que el software cumple con sus requisitos y está funcionando correctamente. Las pruebas de software pueden incluir pruebas de unidad, pruebas de integración, pruebas de sistema y pruebas de aceptación, entre otras. El objetivo de las pruebas de software es garantizar la calidad del software y detectar posibles fallos antes de que el software se ponga en producción. Las pruebas de sistema demuestran que los componentes son compatibles, que interactúan correctamente y que transfieren los datos correctos en el momento adecuado a través de sus interfaces.

3.2.1. Pruebas unitarias

Según (Sommerville, 2006), las pruebas unitarias se definen como pruebas que se centran en la verificación de que unidades individuales de código funcionan como se espera. Estas pruebas se realizan a nivel de componente o módulo y su objetivo es asegurar que cada unidad de software funciona correctamente de forma aislada antes de ser integrada con otras unidades. Las pruebas unitarias suelen ser realizadas por los propios desarrolladores y pueden incluir la ejecución de casos de prueba específicos, la revisión del código y el análisis estático del mismo. El objetivo principal de las pruebas unitarias es garantizar la calidad y fiabilidad de cada unidad de código

Pruebas de caja blanca

Según (Pressman, 2010) las pruebas de caja blanca se centran en la estructura interna del software, es decir, en el código fuente y en la lógica interna de los componentes. Estas pruebas se realizan con un conocimiento detallado de la implementación del software y su objetivo es validar la lógica interna del programa, asegurando que todas las instrucciones del código se ejecuten correctamente y que todas las ramas de decisión sean evaluadas.

Para la comprobación de los códigos y asegurar la ejecución de todas las estructuras presentes se selecciona el método de rutas básicas. A continuación, se presenta un ejemplo de la definición de las rutas básicas en un fragmento de código.

Partiendo del fragmento de código de la función edit_modelo se realiza la agrupación de las sentencias (ver *Figura 7*) para definir la correlación de acuerdo al orden de ejecución construyéndose el grafo (G) de flujo (ver *Figura 8*) para determinar los caminos básicos.

Para el cálculo de la complejidad ciclomática V (G), que indica la cantidad de caminos básicos independientes, se emplea la fórmula:

$$V (G) = E – N +2,$$

donde E es el número de aristas del gráfico de flujo y N el número de nodos del grafo de flujo.

En el caso propuesto el desarrollo de la fórmula anterior es como sigue a continuación:

$$V (G) = 5-5+2,$$

$$V (G) = 2,$$

por lo que se necesitan dos caminos básicos que aseguren la comprobación de todas las sentencias. Luego se definen las rutas independientes:

Ruta 1: 1, 4 y 5

Ruta 2: 1, 2 y 3

```
# Editar modelos
@user_passes_test(pertenece_a_grupo)
def edit_modelo(request, pk):
    modelo = get_object_or_404(Modelo, pk=pk)

    if request.method == 'POST':
        form = ModeloForm(request.POST, instance=modelo)
        if form.is_valid():
            modelo = form.save(commit=False)
            modelo.modificado_por = request.user
            modelo.save()

            return redirect('modelo_list')
    else:
        form = ModeloForm(instance=modelo)
    return render(request, 'config/editarModelo.html', {'form': form})
```

Figura 7 Agrupación del código para definir el orden de ejecución de la función edit_model.

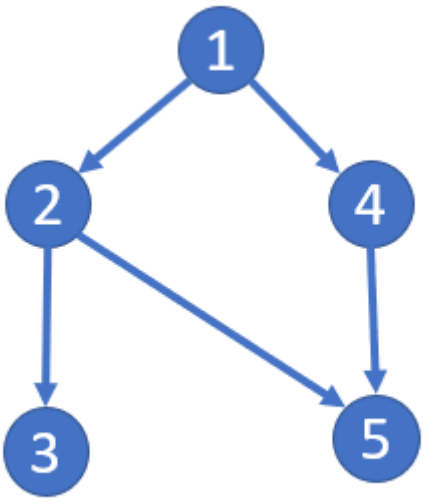

Figura 8 Grafo de flujo para la función edit_model

Teniendo en cuenta el análisis de rutas básicas en el código, se proporcionan los datos adecuados para comprobar la ejecución de las rutas definidas a través de la realización de pruebas unitarias.

Pruebas unitarias

Las pruebas unitarias son fundamentales para garantizar la calidad y el correcto funcionamiento de un sistema. A continuación, se detallan los pasos a seguir.

1. Configurar el entorno de pruebas:

- En el archivo de configuración settings.py, se puede especificar una base de datos separada para las pruebas o utilizar una base de datos en memoria para mayor eficiencia.

2. Crear casos de prueba:

- Se debe crear un archivo separado para las pruebas y definir los casos de prueba como métodos dentro de esa clase.

3. Configurar datos de prueba:

- Se pueden crear instancias de modelos, insertar registros en la base de datos, entre otras acciones necesarias.

4. Escribir pruebas:

- Dentro de cada método de prueba, se debe escribir el código para verificar el comportamiento esperado del sistema.

5. Ejecutar las pruebas:

- Utilizando el comando de gestión de Django desde la línea de comandos, se pueden ejecutar las pruebas unitarias.

```python
class OrganizacionModelTestCase(TestCase):
    def setUp(self):
        self.organizacion = Organizacion.objects.create(
            nombre='Prueba Organizacion',
            direccion='Prueba Direccion',
            telefono='58641237',
            correo_electronico='test@ejemplo.com',
            sitio_web='http://ejemplo.com'
        )

        self.modelo = Modelo.objects.create(nombre='Test Modelo')

        self.organizacion.modelos.add(self.modelo)

    def test_organizacion_creation(self):
        self.assertEqual(self.organizacion.nombre, 'Prueba Organizacion')
        self.assertEqual(self.organizacion.direccion, 'Prueba Direccion')
        self.assertEqual(self.organizacion.telefono, '56873459')
        self.assertEqual(self.organizacion.correo_electronico, 'prueba@ejemplo.com')
        self.assertEqual(self.organizacion.sitio_web, 'http://ejemplo.com')
        self.assertEqual(str(self.organizacion), 'Prueba Organizacion')
        self.assertEqual(self.organizacion.modelos.count(), 1)
        self.assertEqual(self.organizacion.modelos.first(), self.modelo)
```

Figura 9 Pruebas unitarias

3.2.2. Pruebas de aceptación

Tabla 20 HU1_TAREA1

Caso de Prueba de aceptación	
Código: HU1_Tarea1	**Historia de Usuario:** Administrar modelos de Madurez.
Nombre: Insertar Modelo de Madurez	
Descripción: El usuario hace clic en el botón: *Nuevo registro*	
Condición de ejecución: Rellenar los campos obligatorios del formulario.	
Pasos de ejecución: - En el menú principal de Modelos, el usuario hace clic en el botón *Nuevo*. - El usuario llena los campos para añadir un nuevo modelo de madurez. - El usuario presiona el botón *Aceptar*.	
Resultado de la prueba: Se insertará un nuevo modelo de madurez y se mostrará un mensaje que indica que se ha insertado satisfactoriamente.	
Evaluación de la prueba: Satisfactoria.	

Tabla 21 HU1_TAREA2

Caso de Prueba de aceptación	
Código: HU1_Tarea2	**Historia de Usuario:** Administrar modelos de Madurez.
Nombre: Mostrar Modelo de Madurez:	
Descripción: Muestra un listado con los modelos de madurez registrados en el sistema.	
Condición de ejecución: Existencia de modelos de madurez registrados en la base de datos.	
Pasos de ejecución: - El usuario accede al menú principal de modelos de madurez. - Se muestra un listado con los modelos de madurez registrados en el sistema.	

Resultado de la prueba: El sistema muestra un listado de los modelos de madurez existentes en la base de datos y las diferentes acciones a realizar (Nuevo, Editar y Eliminar).
Evaluación de la prueba: Satisfactoria.

Una vez realizadas las pruebas al sistema se detectaron un total de 18 no conformidades determinadas. Dichas conformidades fueron clasificadas en significativas, no significativas y recomendaciones, tal y como se muestra en la Figura 21.

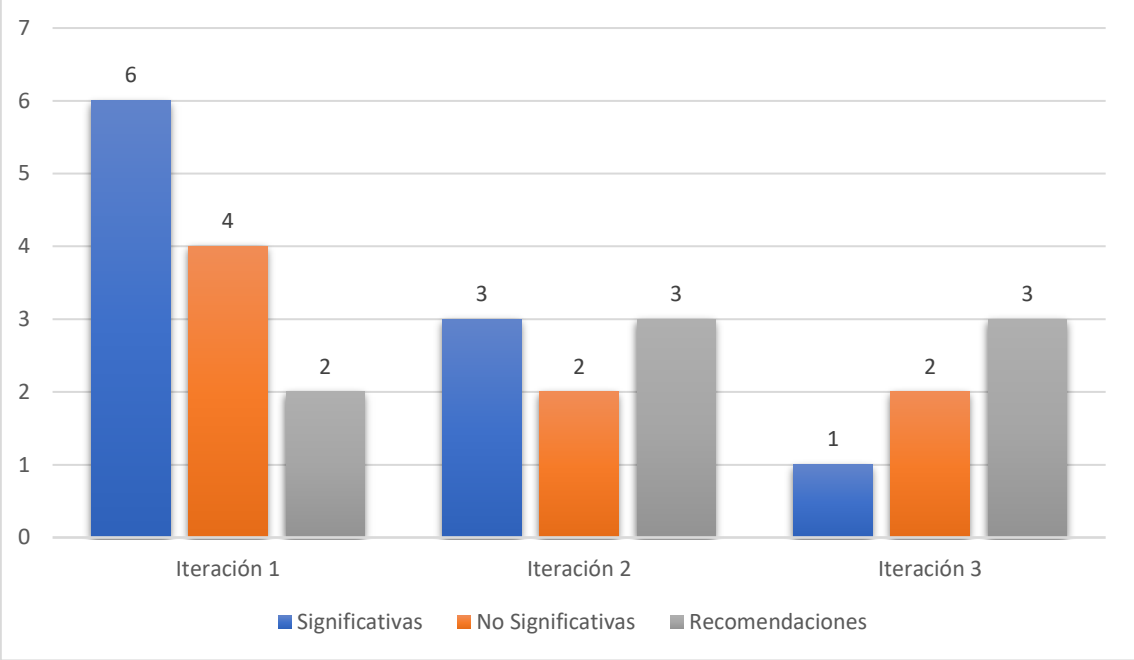

Figura 10 Gráfico de no conformidades detectadas en las pruebas de aceptación por iteración.

Conclusiones del capítulo

En el presente capítulo se expuso el proceso de implementación de la solución propuesta, el cual dio lugar a un producto que cumple con las pautas definidas en el modelo descrito en el Capítulo 2. Se describieron las pruebas diseñadas para validar la implementación del sistema informático para el soporte del proceso de evaluación en infraestructuras TI, que sirvieron como elemento clave para validar que la solución cumple con las expectativas del cliente.

CONCLUSIONES FINALES

A partir de la investigación realizada y el desarrollo del sistema informático propuesto, se pueden concluir los siguientes aspectos:

- La gestión eficiente de las infraestructuras de TI es crucial para el funcionamiento exitoso de las organizaciones en el entorno actual.
- La aplicación de metodologías de desarrollo de software y el uso de herramientas y tecnologías adecuadas son fundamentales para garantizar la calidad y el éxito en la implementación de un sistema informático.
- El conjunto pruebas unitarias y de aceptaciones realizadas permitieron verificar y validar el cumplimiento de las funcionalidades demostrando que el sistema informático propuesto reúne las condiciones tecnológicas para su uso y satisface las necesidades del cliente.

REFERENCIAS BIBLIOGRÁFICAS

Becker, J., Niehaves, B., Poeppelbuss, J., & Simons, A. (2010). *Maturity models in IS research*.

Bertolli, M. P., Roark, G. Y., Urrutia, S. B., & Chiodi, F. J. (2017). Revisión de modelos de madurez en la medición del desempeño. *Inge Cuc*, *13*(1), 70-83.

bin Uzayr, S. (2022). *Mastering Bootstrap: A Beginner's Guide*. CRC Press. https://api.taylorfrancis.com/content/books/mono/download?identifierName=doi&identifierValue=10.1201/9781003310501&type=googlepdf

Boehm, B., & Turner, R. (2003). Observations on Balancing Discipline and Agility. En *Proceedings of the Agile Development Conference (ADC'03)* (p. 39). https://doi.org/10.1109/ADC.2003.1231450

Cadavid, A. N., Martínez, J. D. F., & Vélez, J. M. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, *11*(2), 30-39.

Demestre, D. H., & González, Y. H. (2019). Diseño de un Sistema de Gestión de Información de Recursos Humanos. *Serie Científica de la Universidad de las Ciencias Informáticas*, *12*(3), 31-40.

Dijk, F. W. (2017). *Adopting the Cloud: A multi-method approach towards developing a cloud maturity model*. University of Twente.

Duarte, D., & Martins, P. V. (2013). A maturity model for higher education institutions. *Journal of Spatial and Organizational Dynamics*, *1*(1), 25-44.

Egas, L., & Játiva, J. (2008). *Evolución de las Metodologías de Desarrollo de la Ingeniería de Software en el Proceso la Ingeniería de Sistemas Software, Creación: 2008; Recuperado: 9 mayo 2015*.

González Trejo, M. (2021). *Gestión de la infraestructura de TI*.

Kahlert, T., & Giza, K. (2016). Visual Studio Code Tips & Tricks Vol. 1. *Microsoft Deutschland GmbH*. https://download.microsoft.com/download/8/A/4/8A48E46A-C355-4E5C-8417-E6ACD8A207D4/VisualStudioCode-TipsAndTricks-Vol.1.pdf

Kereki, F. (2023). *Mastering JavaScript Functional Programming: Write clean, robust, and maintainable web and server code using functional JavaScript and TypeScript*. Packt Publishing Ltd.

Larman, C. (2003). *UML y Patrones. Segunda Edición. Sl*.

Laudon, K. C., Laudon, J. P., & Alegre, S. C. (2012). *Sistemas de información gerencial* (Vol. 12). Pearson Educación Naucalpan de Juárez.

León, R. A. H., & González, S. C. (2020a). *El paradigma cuantitativo de la investigación científica*. Editorial Universitaria (Cuba).

León, R. A. H., & González, S. C. (2020b). *El paradigma cuantitativo de la investigación científica*. Editorial Universitaria (Cuba).

Melé, A., & Belderbos, B. (2022). *Django 4 By Example: Build powerful and reliable Python web applications from scratch*. Packt Publishing Ltd.

Mueller, J. P. (2023). *Beginning Programming with Python For Dummies*. John Wiley & Sons.

Piattini, M., García, F., & Caballero, I. (2007). Calidad de sistemas informáticos. *Alfaomega–Ra-Ma, México*.

PRESSMAN, R. S. (2010). Ingeniería de software enfoque práctico. Pressman. PDF. *Ingeniería del software, un enfoque práctico*.

Pressman, R. S. (2010). *Ingeniería de Software: Un enfoque práctico* (Séptima). McGraw Hill New York.

Rebah, H. B., Boukthir, H., & Chedebois, A. (2022). *Website Design and Development with HTML5 and CSS3*. John Wiley & Sons. https://books.google.com/books?hl=es&lr=&id=Vn5WEAAAQBAJ&oi=fnd&pg=PR11&dq=html+5+and+css3&ots=eJn_O_ZK6L&sig=jgcBVt8mO2CRsBZoydkMKzXMHg8

Rodríguez, Y. R., Kile, S. B. F., & Romero, A. E. R. (2013). Aplicación de la estrella de Boehm y Turner al proyecto laboratorios virtuales. *UCV-HACER. Revista de Investigación y Cultura*, *2*(1), 20-29.

Siahaan, V., & Sianipar, R. H. (2019). *Learn SQLite with Python: Building Database-Driven Desktop Projects*. Sparta Publishing.

Siami-Namini, S., Tavakoli, N., & Namin, A. S. (2018). A comparison of ARIMA and LSTM in forecasting time series. *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, 1394-1401.

Sommerville, C. J. (2006). *The decline of the secular university*. Oxford University Press.

ANEXOS

Anexo I: Historias de Usuario para el sistema informático

Tabla 22: HU Administrar organizaciones

Historia de Usuario	
Número: 3	**Usuario:** Consultor.
Nombre de la Historia: Administrar organizaciones	
Prioridad del Negocio: Alta (Alta, Media, Baja)	**Riesgo en Desarrollo:** Medio (Alto, Medio, Bajo)
Puntos Estimados: 2	**Iteración Asignada:** 2
Programador Responsable: Henry	
Descripción: El rol **Consultor** podrá gestionar organizaciones a evaluar (Insertar, Modificar, Eliminar, Editar)	
Observaciones:	

Tabla 23: HU Gestionar Usuarios

Historia de Usuario	
Número: 4	**Usuario:** Administrador.
Nombre de la Historia: Administrar usuarios	
Prioridad del Negocio: Alta (Alta, Media, Baja)	**Riesgo en Desarrollo:** Bajo (Alto, Medio, Bajo)
Puntos Estimados: 0.5	**Iteración Asignada:** 2
Programador Responsable: Henry	
Descripción: El Sistema tendrá definido por defecto un usuario administrador, el cual tendrá acceso a todas las funcionalidades del sistema	
Observaciones: El Administrador del sistema será el único usuario que tendrá acceso a todas las funcionalidades del sistema.	

Tabla 24: HU Acceder al sistema

Historia de Usuario	
Número: 5	**Usuario:** Administrador, Experto, Consultor, Evaluador

Nombre de la Historia: Acceder al sistema	
Prioridad del Negocio: Alta (Alta, Media, Baja)	**Riesgo en Desarrollo:** Medio (Alto, Medio, Bajo)
Puntos Estimados: 0.5	**Iteración Asignada:** 2
Programador Responsable: Henry	
Descripción: Los usuarios que interactúan con el sistema tendrán un nombre de usuario y una clave única con la que podrán ingresar.	
Observaciones: Solo los usuarios que estén registrados en el sistema tendrán acceso	

Tabla 25: HU Generar Reporte

Historia de Usuario	
Número: 6	**Usuario:** Evaluador.
Nombre de la Historia: Generar reporte	
Prioridad del Negocio: Alta (Alta, Media, Baja)	**Riesgo en Desarrollo:** Medio (Alto, Medio, Bajo)
Puntos Estimados: 1	**Iteración Asignada:** 3
Programador Responsable: Henry	
Descripción: Los Reportes que se generan serán los resultados de las evaluaciones realizadas a las entidades. Se visualizará las evaluaciones emitidas por cada métrica evaluada.	
Observaciones: Solo el usuario **Evaluador** tendrá acceso a las evaluaciones emitidas.	

Anexo II Tarjetas CRC para el sistema informático.

Tabla 26 Tarjeta CRC clase Organización

Clase: Organización	
Responsabilidad:	**Colaboración:**
Almacenar información de una organización (nombre, dirección, teléfono, etc.). Realizar operaciones de creación, edición y eliminación de organizaciones.	Clase Modelos Colabora con la clase User para el campo creado_por, que representa al usuario que creó la organización.
Atributos: nombre, direccion, telefono, correo_electronico, sitio_web	

Anexo III Tareas de ingenierías para el sistema informático.

Tabla 27 Modificar modelo de Madurez

Tarea de Ingeniería	
Número de tarea: 4	**Número de HU:** 1
Nombre de Tarea: Modificar modelo de Madurez	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Programador Responsable: Henry	
Descripción: Al autenticarse como Experto se accede a la opción editar modelos de Madurez. (nombre, descripción)	

Tabla 28 Insertar Métrica

Tarea de Ingeniería	
Número de tarea: 5	**Número de HU:** 2
Nombre de Tarea: Insertar Métrica	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Henry	
Descripción: El usuario Experto podrá insertar en el sistema una nueva métrica asociada a un modelo de madurez (nombre, descripción)	

Tabla 29 Editar Métrica

Tarea de Ingeniería	
Número de tarea: 6	**Número de HU:** 2
Nombre de Tarea: Editar Métrica	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Henry	
Descripción: Al autenticarse como Experto se selecciona el modelo de madurez del cual se desea modificar una métrica y a continuación se selecciona la métrica a editar. (nombre, descripción)	

Tabla 30 Eliminar Métrica

Tarea de Ingeniería	
Número de tarea: 7	**Número de HU:** 2
Nombre de Tarea: Eliminar Métrica	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Henry	
Descripción: El usuario Experto podrá eliminar una métrica existente en el sistema. (id)	

Tabla 31 Mostrar Métrica

Tarea de Ingeniería	
Número de tarea: 8	**Número de HU:** 2
Nombre de Tarea: Mostrar Métrica	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Programador Responsable: Henry	
Descripción: Al autenticarse con el rol Experto se muestran las métricas existentes por cada modelo de madurez en el sistema. (nombre, descripción, fecha de registro, fecha de modificación)	

Tabla 32 Insertar Organización

Tarea de Ingeniería	
Número de tarea: 9	**Número de HU:** 3
Nombre de Tarea: Insertar Organización	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Henry	
Descripción: El usuario Consultor podrá insertar una nueva organización a evaluar en el sistema (nombre de la organización, dirección, teléfono, correo electrónico, sitio web)	

Tabla 33 Mostrar Organización

Tarea de Ingeniería	
Número de tarea: 10	**Número de HU:** 3
Nombre de Tarea: Mostrar Organización	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2
Programador Responsable: Henry	
Descripción: Al autenticarse con el rol Consultor se muestran las organizaciones a evaluar existentes en el sistema. (nombre de la organización, dirección, teléfono, correo electrónico, sitio web)	

Tabla 34 Eliminar Organización

Tarea de Ingeniería	
Número de tarea: 11	**Número de HU:** 3
Nombre de Tarea: Eliminar Organización	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Henry	
Descripción: El usuario Consultor podrá eliminar una organización existente en el sistema. (id)	

Tabla 35 Editar Organización

Tarea de Ingeniería	
Número de tarea: 12	**Número de HU:** 3
Nombre de Tarea: Editar Organización	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Programador Responsable: Henry	
Descripción: Al autenticarse como Consultor se accede a la opción editar organización. (nombre de la organización, dirección, teléfono, correo electrónico, sitio web)	