Facultad 2

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Sistema para automatizar la elaboración de documentos de consultorías a centros de datos que realiza el Departamento de Desarrollo de Componentes del Centro de Telemática v2.0

Autor: Alejandro Antonio Batista Zaldívar

Tutores: Ing. Osmar Capote Vázquez

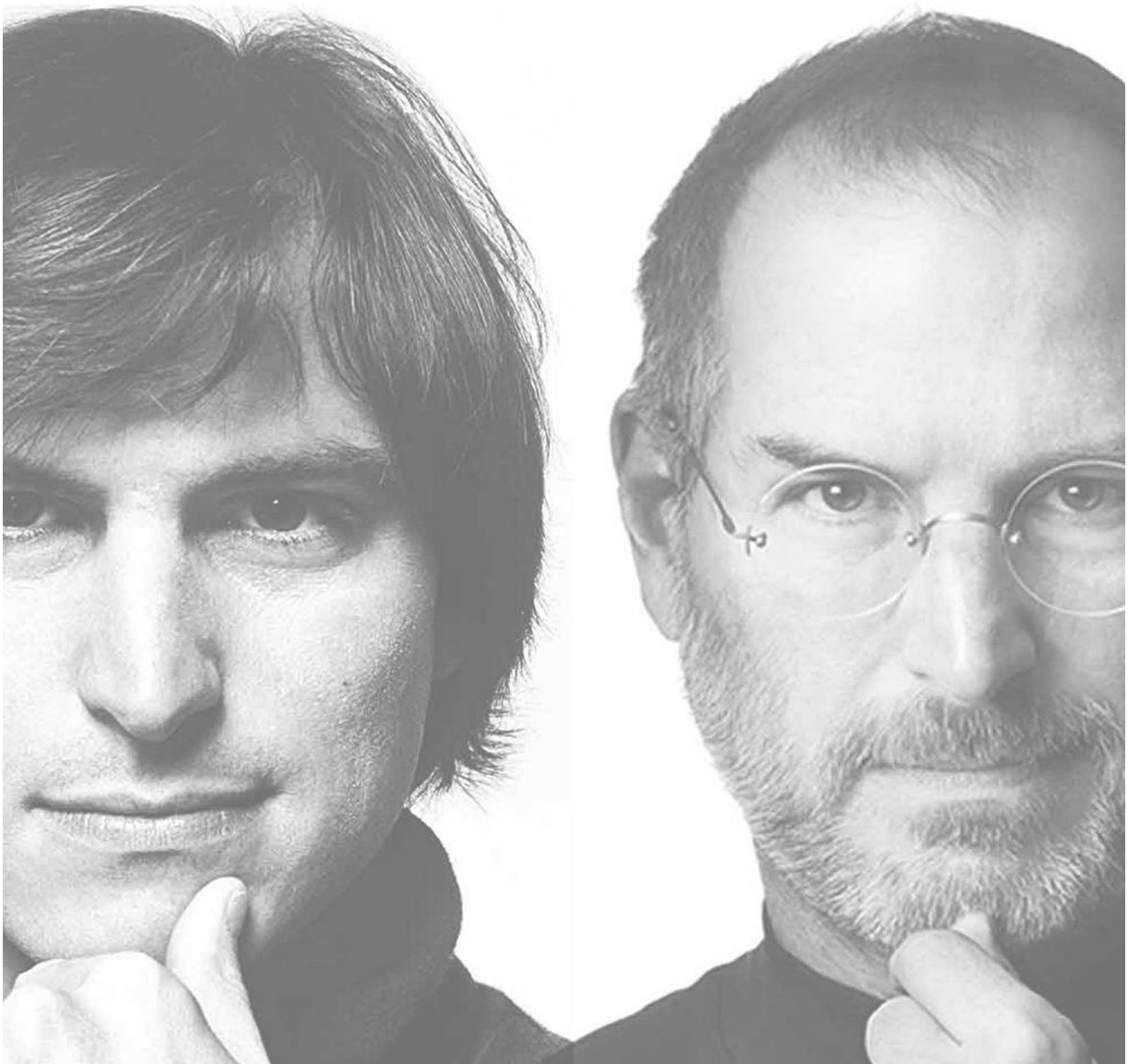
Ing. Roberto Antonio Infante Milanés

La Habana, junio de 2019

¨Las cosas no tienen que cambiar el mundo

para ser importantes¨

Steve Jobs



Declaración de autoría

Declaro ser autor de la presente tesis que tiene por título: “Sistema para automatizar la elaboración de documentos de consultorías a centros de datos que realiza el Departamento de Desarrollo de Componentes del Centro de Telemática v2.0” y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los 13 días del mes de junio del año 2019.

Alejandro Antonio Batista Zaldívar

_____

Firma del Autor

Ing. Roberto Antonio Infante Milanés

_____

Firma del Tutor

Ing. Osmar Capote Vázquez

_____

Firma del Tutor

Datos de contacto

Autor:

Alejandro Antonio Batista Zaldívar

Universidad de las Ciencias Informáticas

Email: aabatista@estudiantes.uci.cu

Tutor:

Ing. Roberto Antonio Infante Milanés

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de Granma, en el año 2015. Profesor del Departamento de Ingeniería y Gestión de Software de la Facultad, desde septiembre del 2015. Ha impartido las asignaturas de Sistemas de Bases de Datos 1 y 2 y de Programación 1.

Email: rainfantem@uci.cu

Tutor:

Ing. Osmar Capote Vázquez

Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas, en el año 2015. Desarrollador del Centro de Telemática. Actualmente es el jefe del Departamento de Desarrollo de Componentes de dicho centro.

Email: ocapote@uci.cu

Agradecimientos

Parece mentira que iba a llegar este día y no hay nada mejor que estar rodeado de todos ustedes, las personas que me acompañaron en esta aventura y me están acompañando en lo que se convertirá en uno de los días más importantes de mi vida.

En primer lugar, quiero agradecer a Google, por sacarme del apuro en incontables ocasiones y responderme a todas las preguntas que le hacía.

A mis profesores por ser parte fundamental en esta trayectoria, por aportarme conocimientos e inculcarme lo que es disciplina, dedicación y además enseñarme de que si nos ponemos para las cosas si podemos lograrlo.

A mis tutores por apoyarme y ayudarme en todo lo que pudieron ofrecer y haber sido guías en este proceso.

A mis amigos, los que están aquí presente y los que ya no están también. A aquellos de mi querida facultad 5, a mi querida Daylilis o dcroque como le digo de cariño, a Brunet, Acosta, Villavicencio, ya que junto a ellos aprendí a cogerle amor a la UCI y darme más ganas de llegar hasta el final.

También quiero agradecer a una de las grandes cosas que me dio la UCI, a mi amigo Asiel, que, aunque la distancia nos mantenga lejos esta amistad seguirá existiendo.

A mis amigos y compañeros de aula, de todos ellos recuerdo las discusiones, las risas, el estrés por las pruebas, y hoy la alegría de convertirnos todos en ingenieros.

Quiero agradecer a cuatro personas que siempre han estado conmigo desde que todo empezó, y espero no se pongan celosas por mencionarlas ahora casi al final, pero nunca se me ocurriría dejar de decir los nombres de Daniela o Esmeregilda como le diría en otras circunstancias, a mi querida Adiaris, ¿o prefieres que te diga tamara?, a Delpi y a Suni. Nunca voy a olvidar de los buenos y malos momentos que viví en esta universidad fueron al lado de ustedes.

El agradecimiento más importante lo quise guardar para el final, a mis hermanas, en especial en mi hermana Lisbet por ser testigo incansable del sacrificio que hacía para lograr esta meta, por darme siempre un beso y fuerte abrazo los domingos cuando tenía que regresar a la universidad, junto a ella agradecer a mi cuñado Eduardo, que sé que también está feliz porque ha sido testigo desde que comenzó mi carrera. A mis sobrinos Sheilita, Eduardito y Riannys, que han sido mi motivación y mi adrenalina, porque a ellos quiero darle todo lo que se merecen y el mejor de los ejemplos.

Quiero agradecer a mis padres, por creer en mí, por haber aguantado todos los dolores de cabeza que les provoqué, por desvelarse conmigo el día que yo tenía un examen, por no rendirse jamás ante mis incontables fracasos que luego se convirtieron en triunfos. Por haber sido mi mayor ejemplo de constancia, de bondad, de amor, de dedicación, por haber dedicado todo el tiempo del mundo a mí, sin pedirme nada a cambio. Son infinitas las razones para agradecerles, por darme los mejor que tenían, sus mejores años, los mejores consejos. Por eso hoy son merecedores de esta alegría. Por todos sus sacrificios y por confiar en mi MUCHAS GRACIAS.

Dedicatoria

A mis padres por creer en mí y acompañarme en esta maravillosa aventura, por animarme a no rendirme jamás.

"Esto es para los locos. Los inadaptados. Los rebeldes. Los alborotadores. Los que ven las cosas de otra manera... los que cambian las cosas. Los que empujan a la raza humana hacia adelante. Y mientras algunos pueden verlos como locos, nosotros los vemos como genios"

Resumen

La consultoría es una de las asistencias más solicitadas por las empresas, debido a que proporcionan un servicio de asesoramiento profesional que ayuda a los empresarios y a las entidades a alcanzar sus objetivos, detectando posibles problemas o necesidades que se deben cubrir u optimizar. En la Universidad de las Ciencias Informáticas se encuentra el Centro de Telemática, el cual cuenta con un departamento con especialistas destinados a realizar servicios de consultorías a instituciones, con la finalidad de diagnosticar el estado actual de los centros de datos.

En la actualidad el proceso de elaboración y levantamiento de información de los documentos de las consultorías se realiza con ayuda del sistema para automatizar la elaboración de documentos de consultorías a centros de datos, el mismo carece de funcionalidades que faciliten el trabajo de los especialistas ya que no les ofrece la posibilidad de diseñar una topología física de red, no es posible utilizar el sistema cuando se encuentran desconectados de una red de internet y no permite listar las debilidades, amenazas, fortalezas y oportunidades de cada uno de los centros de datos. En la presente investigación se exponen los resultados de los procesos de análisis, diseño, implementación y pruebas del sistema de elaboración de documentos de consultorías a centros de datos en su versión 2.0, la cual ofrece nuevas herramientas a los especialistas del Departamento de Desarrollo de Componentes para solucionar la problemática antes expuesta.

Palabras clave: centros de datos, consultorías, diagnósticos.

Abstract

Consultancy is one of the most requested assistance by companies, because they provide a professional advice service that helps employers and entities to achieve their objectives, detecting potential problems or needs that should be covered or optimized. The Telematics Center is located in the University of Computer Science, which has a department with specialists who are dedicated to consulting services to institutions, in order to diagnose the current state of the data centers.

Currently, the process of preparing and gathering information from consulting documents is done with the help of the system to automate the development of consulting documents to data centers, it lacks functionalities that facilitate the work of specialists since it does not offer them the possibility of designing a physical network topology, it is not possible to use the system when they are disconnected from an internet network and it does not allow to list the weaknesses, threats, strengths and opportunities of each of the data centers. In the present investigation the results of the processes of analysis, design, implementation and tests of the system of elaboration of documents of consultancies to data centers in its version 2.0 are exposed, which offers new tools to the specialists of the Department of Development of Components to solve the problem described above.

***Keywords**: data centers, consultancies, diagnostics.*

Índice

Índice de tablas

Índice de figuras

Introducción

En la actualidad Cuba se encuentra inmersa en un desarrollo científico y tecnológico, que contribuye a que la sociedad evolucione de forma acelerada, con la meta fundamental de alcanzar la plenitud en sus potencialidades. Por dicha razón se lleva a cabo un proceso de informatización que evidencia el uso de soluciones informáticas con el objetivo de automatizar los procesos de negocio. A partir de ello, entidades e instituciones optan por contar con centros de datos, los cuales son espacios que albergan en su interior múltiples servidores[1], unidades de almacenamiento, conexiones y cableado.

En aras de apoyar este proyecto el centro de Telemática (TLM), perteneciente a la facultad 2 de la Universidad de Ciencias Informáticas (UCI), ha desarrollado un sistema de elaboración de documentos de consultorías a centros de datos para el mejoramiento del trabajo en el Departamento de Desarrollo de Componentes. Los especialistas son los encargados de realizar las consultorías con el objetivo de diagnosticar el estado de la infraestructura y realizar un levantamiento de la información para evaluar la situación actual del local tecnológico de acuerdo a la experiencia de trabajo que tienen los especialistas con las empresas.

Desde un diagnóstico inicial realizado al sistema se detectan ciertas deficiencias que necesitan ser atendidas:

- El sistema no es capaz de listar de forma independiente las debilidades, amenazas, fortalezas y oportunidades que fueron identificadas en la institución visitada por el especialista, dificultando el trabajo a la hora de elaborar una propuesta de solución.
- Los especialistas del departamento solo pueden hacer uso del sistema cuando se encuentra dentro del departamento o conectado a la red UCI.
- El Departamento de Desarrollo de Componentes no cuenta con una herramienta propia que permita a los especialistas realizar el diseño de una topología física de

[1] Se refiere a servidores informáticos.

red, sin tener que hacer uso de herramientas secundarias, que en la mayoría de los casos son privativas[2].

Teniendo en cuenta los elementos planteados con anterioridad, se identifica como **problema a resolver:** ¿cómo facilitar el proceso de realizar consultorías a centros de datos que se lleva a cabo en el Departamento de Desarrollo de Componentes del Centro Telemática de la Facultad 2 en la Universidad de Ciencias Informáticas?

Se define como **objeto de estudio**: procesos de consultoría a centros de datos.

Estableciéndose como **objetivo general:** desarrollar la versión 2.0 del Sistema para automatizar la elaboración de documentos de consultorías a centros de datos, que realiza el Departamento de Desarrollo de Componentes del Centro Telemática de la Facultad 2.

Definiéndose como **campo de acción:** el proceso de consultoría a centros de datos que realiza el Departamento de Desarrollo de Componentes del Centro de Telemática de la Facultad 2.

Para dar cumplimiento al objetivo planteado, se definen las siguientes **tareas de investigación:**

1. Elaboración del marco teórico de la investigación a través de un estudio de los procesos que se llevan a cabo en las empresas que realizan consultorías.
2. Selección y estudio de las herramientas informáticas y tecnologías para poder desarrollar las aplicaciones que facilitan el proceso de consultoría a centros de datos que realiza el Departamento de Desarrollo de Componentes del Centro de Telemática.
3. Análisis, diseño e implementación de la solución propuesta.
4. Realización de pruebas de aceptación y pruebas unitarias de la solución propuesta.

[2] Se refiere a un programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo.

Para facilitar el cumplimiento del objetivo propuesto y de las tareas de investigación se emplean métodos teóricos y empíricos de la investigación científica.

Métodos teóricos:

- **Analítico-Sintético:** permite realizar el estudio teórico de la investigación, facilitando el análisis de documentos y la extracción de los elementos más importantes acerca de los procesos que se llevan a cabo en la realización de las consultorías.
- **Histórico-Lógico:** este método permite estudiar todo lo relacionado con los procesos que intervienen en la ejecución de consultorías, para así obtener un conocimiento histórico de su desarrollo y comportamiento a nivel internacional como nacional.

Métodos empíricos:

- **Observación:** se emplea con el objetivo de observar el funcionamiento de algunas instituciones que ofrecen servicios de consultorías a sus clientes, así como los conceptos asociados al proceso de realización de consultorías. Este método proporciona la guía para realizar un registro visual de las características comunes en los procesos que se llevan a cabo en este tipo de empresas e identificar los que puedan formar parte de la solución.
- **Modelación:** es la reproducción simplificada de la realidad que permite descubrir nuevas relaciones y cualidades del objeto. Este método permite la creación de modelos, (propuestas, alternativas, estrategias, etc.). En esta investigación a través del método empírico se realiza el modelado mediante diagramas. Esto permite reflejar la estructura, las relaciones y características de la solución propuesta. En este caso, con el uso de BPMN[3], se facilita también el diseño de clases necesario para la implementación de la aplicación.

[3] *Business Process Model and Notation*, en español Modelo y Notación de Procesos de Negocio.

En aras de estructurar el proceso de desarrollo del presente trabajo de una manera coherente y organizada, se dividió el mismo en un total de tres capítulos:

Capítulo 1. Fundamentos teóricos sobre las consultorías a centros de datos: se definen los conceptos asociados y se abordan los elementos teóricos más importantes que sirven de base para la realización de la investigación; también se proporciona una descripción de las herramientas y metodología de desarrollo de software a utilizar para dar solución al problema planteado.

Capítulo 2. Propuesta de solución: se describe la propuesta de solución creada, además del flujo actual de los procesos a automatizar quedando definidos los requisitos del negocio y los requisitos funcionales y no funcionales del sistema, así como todos los artefactos generados y se detalla la arquitectura del módulo y elementos del diseño.

Capítulo 3. Implementación y pruebas: se expone todo lo relacionado a los procesos de implementación de la propuesta de solución y se representan las pruebas realizadas al sistema, así como los resultados.

Capítulo I Fundamentos teóricos sobre las consultorías a centros de datos

En el presente capítulo se describen los conceptos asociados al dominio del problema. Se realiza un estudio de las soluciones existentes a nivel nacional e internacional para analizar las funcionalidades y características que se pueden tener en cuenta para el desarrollo de la solución propuesta. Además, se fundamenta la selección de la metodología de desarrollo de software, tecnologías y herramientas a utilizar para el desarrollo de la aplicación.

1.1. Conceptos asociados

Consultorías

Existen numerosas definiciones del término "consultoría" y de su aplicación a situaciones y problemas empresariales. Fritz Steele [4](1975) define: "Por consultoría se entiende cualquier forma de proporcionar ayuda sobre el contenido, proceso o estructura de una tarea o de un conjunto de tareas, en que el consultor no es efectivamente responsable de la ejecución de la tarea misma, sino que ayuda a los que lo son".

Centros de datos

Se denomina Centro de Proceso de Datos a aquella ubicación donde se concentran todos los recursos necesarios para el procesamiento de la información de una organización. Dichos recursos consisten esencialmente en unas dependencias[5], debidamente acondicionadas, de computadoras y redes de comunicaciones.

Un centro de datos es un edificio o sala de gran tamaño usada para mantener en él una gran cantidad de equipamientos electrónicos. Suelen ser creados y mantenidos por grandes organizaciones con el objeto de tener acceso a la información necesaria para sus operaciones en

[4] Recibió su licenciatura de la Universidad de Yale y un doctorado en Estudios de Organización de la *Sloan School of Management* del MIT. Ha trabajado principalmente como consultor externo en áreas de cambio organizativo.
[5] Se refiere a un espacio físico destinado para albergar equipos tecnológicos.

todo momento. Prácticamente todas las compañías, ya sean medianas o grandes, tienen algún tipo de centro de datos, mientras que las más grandes llegan a tener varios (Calleja y Cutuli, 2017).

Infraestructura tecnológica

La infraestructura tecnológica agrupa y organiza el conjunto de elementos tecnológicos que integran un proyecto, soportan las operaciones de una organización o sustentan una operación. Una infraestructura define el éxito de una empresa en la medida de que su robustez, calidad y sostenibilidad se traduce en incremento de la inversión en TI[6]. Una infraestructura sólida permite a un software operar de manera eficiente y eficaz durante el tiempo previsto con niveles altos de servicios[7] y prestaciones (Villafuerte y Rodrigo, 2017).

Topología de red

Es el arreglo físico o lógico en el cual los dispositivos o nodos de una red (ejemplo: computadoras, impresoras, servidores, *hubs*[8], *switches*, enrutadores, etc.) se interconectan entre sí sobre un medio de comunicación. Está compuesta por dos partes, la topología física, que es la disposición real de los cables (los medios) y la topología lógica, que define la forma en que los *hosts*[9] acceden a los medios.

Las topologías físicas que se utilizan comúnmente son de bus, de anillo, en estrella, en estrella extendida, jerárquica y en malla (Molina, 2017). Para una descripción detallada de los distintos tipos de topología física de red y los elementos que componen su diseño (Ver Anexo 1).

[6] Tecnologías de la informática.
[7] Se refiere a los servicios informáticos.
[8] Es un elemento de red que sirve para conectar varios equipos entre sí.
[9] Se usa en informática para referirse a las computadoras u otros dispositivos conectados a una red.

DAFO

El concepto DAFO[10], también conocido como FODA o DOFA, es una herramienta de estudio que aborda la situación de una compañía o proyecto empresarial. Para ello analiza sus características internas, como son las fortalezas y debilidades, además de las externas, como oportunidades y amenazas.

Este análisis consigue a conocer la verdadera situación en la que está una entidad o una empresa, pudiendo de esta manera organizar mejor la estrategia de futuro. El objetivo primordial del DAFO es que con la información que se consiga sobre su situación, la compañía pueda afrontar cambios organizativos o tomar decisiones que se adapten mejor a las exigencias del mercado (Aliaga, Gutiérrez-Braojos y Fernández-Cano, 2018).

1.2. Análisis de los procesos de servicio de consultorías a centros de datos que realiza el Departamento de Desarrollo de Componentes del Centro de Telemática

El Departamento de Componentes del Centro Telemática, cuenta con especialistas encargados de realizar las consultorías a centros de datos de distintas empresas del país. Los principales procedimientos que ocurren durante este proceso son los siguientes:

- Establecer contacto con el cliente por medio de citas, llamadas y correos electrónicos.
- Programar reunión con el cliente para llegar a un acuerdo y firmar un contrato.
- Análisis y estudio del centro de datos de la empresa a consultar.
- Recoger y seleccionar la información del centro de datos.
- Generar diagnóstico y diseño de la consultoría haciendo uso del sistema de elaboración de documentos de consultorías a centros de datos

[10] DAFO: Debilidad, Amenaza, Fortaleza, Oportunidad.

1.3. Estado actual de los procesos que llevan a cabo los centros que brindan servicios de consultorías

En la presente sección se realiza un estudio de empresas, instituciones y centros que ofrecen servicios de consultorías a sus clientes, analizando cada uno de los elementos que intervienen en los procesos que se ponen de manifiesto a la hora de brindar un servicio de este tipo.

MASTERTICS S.A.S

Figura 1 Logo de la empresa MASTERTICS S.A.S (Fuente: sitio web oficial de MASTERTICS S.A.S)

Es una firma especializada en tecnologías de la información que facilita el cumplimiento de los objetivos de sus clientes dedicándose a proporcionar personal, tecnología y conocimiento con altos estándares de calidad y cumplimiento. Tiene como misión la de proveer soluciones de alta calidad para las empresas mediante el uso de la tecnología, respetando el medio ambiente y contribuyendo al buen desarrollo de la sociedad.

Uno de los pilares de esta empresa es brindar servicios de consultorías a sus clientes, los principales procesos que intervienen en la realización de las consultorías son:

- Modernización de aplicaciones en una organización que busca maximizar la inversión realizada inicialmente.
- Migraciones de aplicaciones entre las diferentes versiones de ORACLE[11] y en algunos casos de aplicaciones no ORACLE.
- Desarrollo de software a la medida usando al máximo la tecnología informática, en herramientas ORACLE, para que los sistemas resultantes soporten la toma de decisiones

[11] Es una herramienta cliente/servidor para la gestión de base de datos.

en aquellas áreas en donde las soluciones del mercado no satisfacen las necesidades del cliente.

- Evaluación técnica de personal de sistemas. Aplicación de pruebas técnicas para validación de conocimientos a candidatos en el área de tecnología informática.

***Data Center* Consultores**

Figura 2 Logo de la empresa *Data Centers* Consultores (Fuente: sitio web oficial de *Data Centers* Consultores)

Es una empresa de tecnología con sede en Costa Rica y con oficinas en Perú y Colombia. Desde su fundación se ha convertido en una empresa líder en el diseño, auditoría, certificación y operación de proyectos de misión crítica, con más de 40 centros de datos certificados bajo estándares internacionales en toda la región Latinoamericana.

La Excelencia, la sostenibilidad y la innovación forman parte inherente de todos y cada uno de sus servicios. De esta forma garantizan a sus clientes proyectos diseñados y construidos bajo los más altos estándares, garantizando que la información crítica sea almacenada de manera segura, con alta disponibilidad, asegurando la continuidad del negocio.

Tienen como misión la creación de soluciones de alta tecnología, innovadoras y ambientalmente sostenibles, para garantizar que las personas puedan acceder e intercambiar información en cualquier momento y lugar. Entre las actividades que intervienen en el proceso de consultorías a centros de datos se destacan:

- Selección de sitio y evaluación de riesgo para proyectos de misión crítica.
- Construcción de mapas de amenazas y de vulnerabilidades, obteniendo a partir de ellos los mapas de índices de riesgo.

- Mejora de la infraestructura existente de centros de datos.
- Identificar las brechas y oportunidades de mejora de la infraestructura existente, llevando sus centros de datos al siguiente nivel.
- Ingeniería total y diseño de proyectos de misión crítica.
- Diseño de planos arquitectónicos, utilizando el software REVIT[12].

MIKROINTERACCIONES DE MEXICO

Figura 3 Logo oficial de la empresa Mikrointeracciones de México (Fuente: sitio web oficial de Mikrointeracciones de México)

Es una empresa líder dedicada al servicio electromecánico especializado, para ello cuentan con la herramienta más especializada del mercado, lo que les permite garantizar su trabajo y que sus clientes estén satisfechos. Mikrointeracciones surge de una visión clara de negocios, bajo un concepto innovador, creativo y confiable. Entre sus servicios destaca el de consultorías a centros de datos, el cual es un servicio de consulta que permite:

- Analizar el centro de datos o la infraestructura de TI.
- Planeación, implementación y realización de la transición a una infraestructura más eficaz.
- Creación de un entorno de TI simplificado y más ágil que pueda responder a las cambiantes condiciones del mercado.

[12] es un software de modelado de información de construcción, para *Microsoft Windows*, Permite al usuario diseñar con elementos de modelación y dibujo paramétrico.

- Pruebas de instalación y equipos eléctricos.
- Pruebas de sistemas contra incendios.

Otros de los servicios que ofrece esta empresa es la consultoría a instalaciones críticas y la consultoría a infraestructura crítica.

CIGET

Figura 4 Logo del Centro de Información y Gestión tecnológica perteneciente al Instituto de Información Científica y Tecnológica S (Fuente: sitio web oficial de IDICT)

En nuestro país se encuentra el Centro de información y Gestión Tecnológica (CIGET) de Villa Clara. Es una entidad de interfaz entre las organizaciones de producción de bienes y servicios y de investigación-desarrollo.

Está indisolublemente ligado a propiciar el desarrollo y la innovación tecnológica en el sector empresarial estatal del territorio. Brinda servicios de consultorías con enfoque de proceso y orientado al cliente, de manera flexible y proactiva para que garanticen rapidez de respuesta a las demandas del mercado, así como la satisfacción con los resultados. Contribuye a posibilitar la planificación estratégica y la toma de decisiones de las empresas y sectores priorizados con una amplia gama de servicios y productos. Los principales procesos que se llevan a cabo en esta institución son:

- Asesorías para el diagnóstico, documentación e implementación de Sistemas de Gestión de la Comunicación.
- Asesoría para el diagnóstico del sistema de inteligencia empresarial y estudios de mercado.

- Búsquedas especializadas de Información en base de datos internacionales.
- Asesorías para diagnóstico, documentación e implementación del sistema de innovación y propiedad Intelectual.
- Asesorías en seguridad informática, además de aplicaciones web.

Centro Internacional de La Habana, CIH, S.A

Figura 5 Logo oficial de la empresa CIH S.A (Fuente: sitio web oficial de CIH S.A)

Fundada en 1993, es una sociedad mercantil con más de 20 años en el mercado, que ofrece su experiencia en el campo de las consultorías, auditorias, servicios científico técnicos, transferencia de tecnologías y formación de capital humano, con profesionalidad, responsabilidad, y juicio profesional. Cuenta con más de 500 consultores y auditores acreditados, incluyendo ingenieros, profesores e investigadores de las más diversas áreas del conocimiento. Tienen la capacidad de ofrecer las más diversas soluciones a las demandas de sus clientes, con el respaldo de las universidades cubanas. Entre los principales servicios de consultoría que ofrece esta empresa destacan:

- Asesoría para la internacionalización de las empresas
- Gestión general
- Gestión contable y financiera
- Gestión de recursos humanos
- Gestión de la producción y la comercialización
- Gestión de la calidad
- Estudios de factibilidad de inversiones

- Logística
- Ahorro energético
- Informática
- Técnica de propósito específico

Departamento de Desarrollo de Componentes del Centro de Telemática perteneciente a la Universidad de las Ciencias Informáticas

Figura 6 Logo del Departamento de Desarrollo de Componentes (Fuente: elaboración propia)

La Universidad de las Ciencias Informáticas cuenta con centros de producción de software que están vinculados a varias esferas de la sociedad, uno de ellos es el Centro de Telemática, orientado al desarrollo de sistemas y servicios informáticos integrales para las telecomunicaciones y la seguridad informática. Dicho centro cuenta con el Departamento de Desarrollo de Componentes. Este último, dispone de especialistas destinados a realizar servicios de consultorías a diversas instituciones nacionales, con la finalidad de diagnosticar el estado actual de los centros de datos.

El servicio de consultoría brindado, se basa en los requerimientos establecidos por el departamento y como parte de este proceso, se debe:

- Generar un informe llamado Diagnóstico que contenga un levantamiento de información, guiado por los parámetros a evaluar que propone el departamento. En el informe se evidencia la situación actual de la infraestructura tecnológica que conforma al centro de datos evaluado.

- Confección del informe llamado Diseño, en el cual se propone mejorar aspectos en cuanto a la estructuración, tecnologías y servicios con que cuenta el centro de datos de acuerdo a los recursos que se dispone, del mismo modo, se recomienda adquirir nuevas tecnologías en caso de que la necesiten.

Estos dos procesos de llevan a cabo con ayuda del sistema para la elaboración de documentos de consultorías a centros de datos, el cual permite la generación de los informes de diagnóstico y diseño de forma automatizada.

Conclusiones del estudio

El estudio de los procesos que llevan a cabo las empresas y centros que brindan servicios de consultorías permitió analizar la competencia que concurre actualmente en el ámbito de las consultorías. Además, demostró que aún existen carencias de empresas que se enfoquen en las consultorías orientadas a los centros de datos y que utilicen herramientas informáticas que permitan una mejor gestión de los procesos que intervienen en la realización de las mismas.

En la mayoría de los casos ninguna de las empresas cumple con todas las necesidades del Departamento de Desarrollo de Componentes, ya que hay elementos que se tienen en cuenta en el departamento y no de igual forma en las empresas estudiadas. Los principales aspectos que se tuvieron en cuenta para realizar esta comparación fueron:

- **Centros de datos**: indica si el servicio de consultoría ofrecido por las empresas está enfocado a centros de datos.

- **DAFO:** indica si en el servicio de consultoría ofrecido existen procesos que permitan la identificación de las debilidades, amenazas, fortalezas y oportunidades de la empresa visitada.

- **Sistemas informáticos:** indica si la empresa que realiza la consultoría cuenta con un sistema informático que permita facilitar los procesos que se llevan a cabo y que pueda utilizarse estando desconectado al servicio de internet.

- **Diseñar topología:** indica si la empresa que ofrece servicios de consultoría a centros cuenta con una herramienta informática propia para realizar un diseño de la topología física de red.

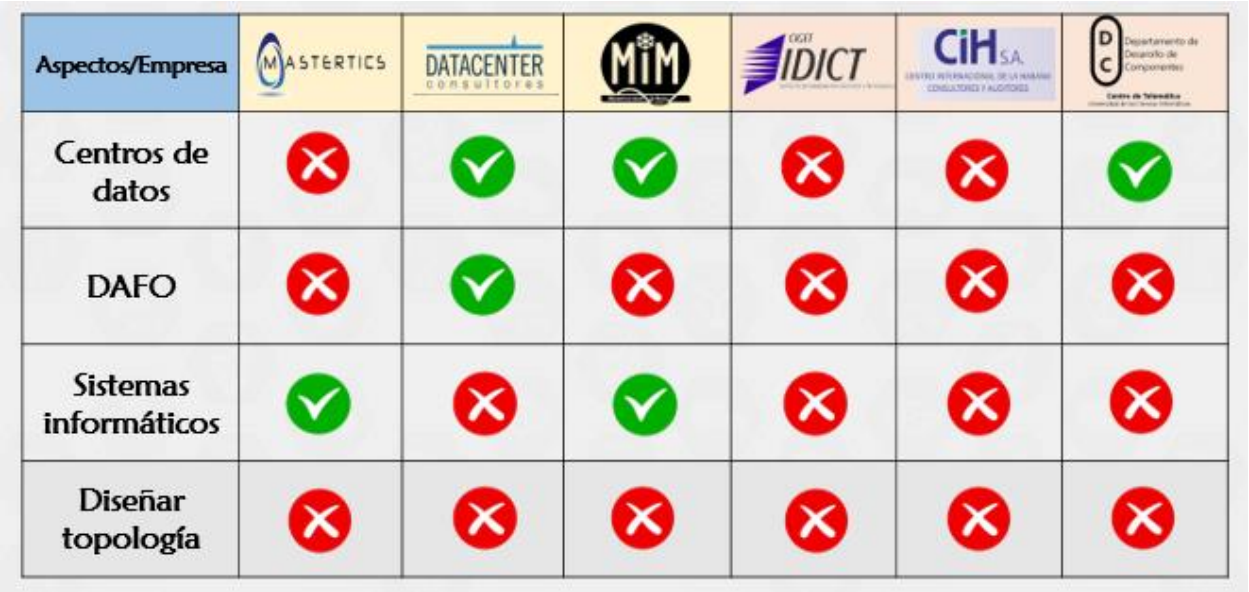


Figura 7 Comparación entre empresas y centros homólogos (Fuente: elaboración propia)

Se analizaron diferentes centros que ofrecen servicios de consultorías a entidades para apoyar la toma de decisiones, pero no existen ninguna que propongan una solución viable a la problemática planteada en esta investigación. Sin embargo, el estudio arrojó resultados útiles que permitió tomar experiencias para una mayor comprensión de aspectos relacionados con el diseño de los centros de datos. A partir de lo anteriormente expuesto, se propone realizar una nueva versión del sistema para automatizar la elaboración de documentos de consultorías a centros de datos (Xilema SASC), el cual, a pesar de encontrarse en explotación por los especialistas del Departamento de Desarrollo de Componentes, no cumple con las nuevas

exigencias y funcionalidades que faciliten el trabajo del personal del departamento encargado de realizar las consultorías a los centros de datos.

1.4. Metodología de desarrollo de software

Desarrollar productos de software con alta calidad depende de las actividades que conllevan a su construcción, donde la selección de la metodología más adecuada juega un papel importante. La correcta selección es fundamental para la planificación, gestión, control y evaluación de forma sistemática, lo que garantiza grandes probabilidades de éxito. En la actualidad existen dos tipos de metodologías para el desarrollo de software, primeramente, se crearon las metodologías tradicionales que se caracterizan por una planificación total del trabajo y una rigurosa definición de artefactos, roles, actividades, así como una extensa documentación limitando incluso la habilidad del equipo de desarrollo.

En el caso de las metodologías ágiles se centran más en el factor humano. Su base está en la simplificación sin renunciar a las prácticas esenciales que aseguran la calidad del producto, esto se refleja en el hecho de ser la persona, el principal factor de éxito en un proyecto, la habilidad está en centrarse más en los cambios que puedan surgir que seguir estrictamente un plan.

Teniendo en cuenta todos los elementos planteados sobre ambos tipos de metodologías se considera y define para la propuesta de solución enmarcarse en una metodología ágil. Tal decisión se debe a que se tiene un equipo de desarrollo pequeño, el cliente es parte fundamental del equipo y tiene la disposición de mantener una comunicación fluida y constante durante el proceso de desarrollo.

La metodología AUP[13] en su variación UCI se adapta a la configuración y ciclo de vida de la actividad productiva de la UCI. El lenguaje de trabajo es totalmente entendible logrando con su ejecución un producto de software con la calidad requerida. Consta de los principales aspectos positivos de la mayoría de las metodologías ágiles adaptándose correctamente a equipos de

[13] *Agile Unified Process*, en español Proceso Unificado Ágil.

desarrollo pequeños; es por esto que se decide utilizar la metodología AUP en su variación UCI, atravesando por sus tres fases de desarrollo (inicio, ejecución y cierre), además utilizando su escenario cuatro, ya que el negocio está bien definido, el cliente estará siempre acompañado por el desarrollador, existe buena comunicación con el cliente y es un proyecto de poca extensión.

1.5. Lenguajes, tecnologías y herramientas de desarrollo.

1.5.1. Marcos de trabajo

Django 1.7

Framework web implementado sobre el lenguaje de programación Python, perteneciente a la licencia BSD (licencia de software libre permisiva). Django brinda estructura al código fuente, fomenta las buenas prácticas de desarrollo web, lo que promueve un código legible y fácil de mantener. La implementación del patrón de diseño MTV (modelo-plantilla-vista), es una característica propia que contiene el *framework*, la cual contribuye a la organización de las distintas partes de la aplicación y a modificar éstas sin afectar cualquier otra pieza del software. Su alta escalabilidad le posibilita manejar el crecimiento continuo de trabajo de manera fluida sin perder calidad en los servicios (Ríos, Mora, Ordóñez y Sojos, 2016).

Xilema-Base-Web

Es un marco de trabajo desarrollado en el centro de Telemática (TLM) que está constituido por Django como *framework* base, librerías de JavaScript como son JQuery y *Backbone* (TLM 2016). Cumple con las pautas de la estrategia visual y estrategia marcaria de la Universidad de las Ciencias Informáticas.

1.5.2. Lenguaje de programación

Python 2.7.6

Python es un lenguaje de programación creado por Guido Van Rossum[14] a principios de los años 90. Es un lenguaje con una sintaxis muy limpia lo que favorece un código legible. Se trata de un lenguaje interpretado o de *script*, con tipado dinámico, multiplataforma y orientado a objetos (Raschka, 2015).

Es un lenguaje de programación de código abierto para el desarrollo de aplicaciones independientes y de s*cripting* en una gran cantidad de dominios de problemas. Es gratis, portable, y relativamente fácil de usar. Python es empleado tanto en proyectos grandes como pequeños, mejorando la productividad de los desarrolladores y la calidad del software (Lutz 2013).

Python ha sido seleccionado como lenguaje de programación por las características expuestas anteriormente, así como por la gran cantidad de librerías disponibles que pueden ser útiles en el presente trabajo.

JavaScript

JavaScript es un potente lenguaje de scripting basado en objetos. El código JavaScript puede ser embebido directamente en un documento HTML[15] e interpretado por un navegador web en el lado del cliente (su uso más extendido). Cuando se combina con el *Document Object Model* (DOM) definido por un navegador web, JavaScript permite crear o manipular dinámicamente el contenido HTML (Castillo, 2017).

1.5.3. Lenguaje de marcado

HTML 5

HTML, más que el lenguaje de marcado predominante usado para estructurar el contenido web, es realmente una tecnología web que interrelaciona varios estándares. En esta última revisión

[14] Guido van Rossum es un científico de la computación, conocido por ser el autor del lenguaje de programación Python. Nació y creció en los Países Bajos.
[15] *HyperText Markup Language*: Lenguaje de Marcas de Hipertexto

del lenguaje se incluyen varias características verdaderamente útiles, como son nuevos elementos semánticos y mejoras en los formularios (Kennedy y Musciano, 2017).

1.5.4. Lenguaje de modelado

UML (lenguaje de modelado unificado)

Lenguaje de Modelado Unificado (*Unified Modeling Language*) es la sucesión de una serie de métodos de análisis y diseño orientados a objetos. Fue creado para forjar un lenguaje de modelado visual común y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento (Ávila, Peña, Lugo y Martínez, 2018).

1.5.5. Bibliotecas utilizadas

Qt-*Opensources* 5.12.1

Es un entorno de trabajo multiplataforma orientado a objetos ampliamente usado para desarrollar programas que utilicen interfaz gráfica de usuario, así como también diferentes tipos de herramientas para la línea de comandos y consolas para servidores que no necesitan una interfaz gráfica de usuario. Está disponible para plataformas Linux/Unix, Windows y MacOSX (Pang, Igasaki y Maehara, 2016).

PyQT4-PyQT5

Es un conjunto de enlaces Python para el conjunto de herramientas Qt y está disponible para todas las plataformas soportadas por Qt, incluyendo Windows y Linux (Duarte, Teodoro y Freitas, 2017). PyQt ha sido seleccionado por las facilidades que ofrece para crear aplicaciones de tipo *desktop,* así como por la gran cantidad de librerías disponibles que pueden ser útiles en el presente trabajo.

1.5.6. Herramientas de desarrollo

Pycharm 2018.3.4

Es un Entorno de Desarrollo Integrado (IDE) desarrollado por la compañía Jetbrain. Es multiplataforma, hay binarios para: Windows, Linux y Mac OS X. Proporciona análisis de código, depuración gráfica y soporte para el desarrollo web con Django, entre otras bondades (islam 2015). Tiene un excelente soporte para Python, además de facilitar los procesos de búsquedas de errores y ejecución de tareas a través de consola (Islam, 2015).

1.5.7. Herramienta CASE

Visual Paradigm 8.0 Enterprise Edition

Es una herramienta CASE[16] que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores durante todos los pasos del ciclo de vida de desarrollo de un software. Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta (Karim, Liawatimena, Trisetyarso, Abbas y Suparta, 2017).

1.5.8. Herramientas de base de datos

PostgreSQL 9.4

Es un sistema de gestión de bases de datos objeto-relacional. Se encuentra distribuido bajo licencia BSD y tiene su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Esto ofrece como ventaja que un fallo en uno de los procesos no afecte al resto, garantizando así que el sistema continúe funcionando (Riggs, Krosing y Bartolini, 2015).

[16] Computer Aided Software Engineering (Ingeniería de Software Asistida por Computadora).

SQLite

Es una herramienta de software libre, que permite almacenar información en dispositivos empotrados de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware, como puede ser una PDA[17] o un teléfono celular. y lo más importante es que se puede usar en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, ya que existe compatibilidad al cien por ciento entre las diversas plataformas disponibles, haciendo que la portabilidad entre dispositivos y plataformas sea transparente (Oh, Kim, Lee y Moon, 2015).

Conclusiones del capítulo

1. Con el desarrollo del marco teórico se logró organizar y guiar el trabajo profundizando en el estudio de algunos conceptos fundamentales asociados a las consultorías a centros de datos.

2. Se realizó un estudio de las empresas, instituciones y centros que ofrecen servicios de consultorías a nivel nacional e internacional, además de cada uno de los procesos que intervienen en esta tarea.

3. Se seleccionaron las tecnologías, herramientas, lenguajes y técnicas para la solución del problema actual.

[17] *Personal Digital Assistant*: Asistente Digital Personal

Capítulo II Propuesta de solución

En este capítulo se exponen los elementos correspondientes a las disciplinas de Requisitos y Análisis y Diseño, de acuerdo a la metodología de desarrollo de software seleccionada. Además, se enuncian los requisitos del negocio, requisitos funcionales y no funcionales que el sistema debe cumplir, así como la arquitectura seleccionada. Se detallan los artefactos generados por cada una de las disciplinas correspondientes a la metodología de desarrollo AUP-UCI en su escenario cuatro.

2.1. Características de la propuesta de solución

En la presente investigación se propone el desarrollo de la versión 2.0 del sistema para automatizar la elaboración de documentos de consultorías a centros de datos que realiza el Departamento de Desarrollo de Componentes del Centro de Telemática. Este sistema cuenta con cuatro módulos en total: sistema, administración, diagnóstico y diseño. A continuación, se muestra una explicación detallada de cada uno de los correspondientes módulos.

Módulo Sistema: se encuentra integrado en el marco de trabajo Xilema-Base-Web y permite la gestión de usuarios, así como los permisos de cada uno de éstos y los grupos a los que pertenecen.

Módulo de Administración: es donde se asientan las bases para realizar las consultorías. En él, se gestionan los datos básicos acerca de las instituciones y el documento.

Módulo de Diagnóstico: en el módulo se gestiona la situación actual del centro de datos y todas las configuraciones necesarias que le permitan optimizar su estado. Está conformado por todos los parámetros a evaluar dentro del diagnóstico de la consultoría, en el que de forma predeterminada se introducirán todos los datos realizados en un levantamiento de información previo.

Módulo de Diseño: el módulo permite gestionar un diseño idóneo de acuerdo a las configuraciones establecidas en la norma por la cual se rigen los centros de datos. Permite valorar

los posibles cambios y mejoras de acuerdo a todos los parámetros evaluados en el módulo anterior.

Figura 8 Propuesta del sistema (Fuente: elaboración propia).

En la propuesta de solución actual se le adiciona al módulo de diagnóstico del Sistema para automatizar la elaboración de documentos de consultorías a centros de datos la posibilidad de adicionar y listar de forma independiente las debilidades, amenazas, fortalezas y oportunidades identificadas en la institución visitada.

Además de contar con este sistema, los especialistas del Departamento de Desarrollo de Componentes pueden disponer de dos herramientas:

Diseñar_Topología: Es una herramienta que permite a los especialistas diseñar una topología física de red con todos los elementos que la conforman (dispositivos y conexiones). Esta herramienta exporta una imagen que luego puede ser utilizada por el especialista para realizar el informe diagnóstico en el Sistema para automatizar la elaboración de documentos de consultorías a centros de datos.

Aplicación_Diagnóstico: Consiste en una aplicación de escritorio la cual permite a los especialistas realizar el diagnóstico sin necesidad de encontrarse conectado a internet. La misma

fue desarrollada con la librería PyQT del lenguaje de programación Python. En ella se muestra una interfaz con un formulario en el cual el especialista inserta los datos, estos datos son almacenados en una base de datos SQLite.

2.2. Requisitos de software

Según Pressman, los requisitos de software constituyen las necesidades de los clientes, las funcionalidades y las restricciones que debe cumplir el software. Los mismos se clasifican en funcionales y no funcionales (Pressman y Maxim, 2016).

2.2.1. Requisitos de negocio

Existen diferentes tipos de requisitos, casi tantos como implicados haya en un proyecto. Uno de estos tipos de requerimientos son los de negocio, los cuales definen los objetivos y problemas que la empresa quiere resolver con el producto. Deben estar basados en una necesidad real del usuario, sea esta conocida o no por él (Siegelaub, 2017).

En el Departamento de Desarrollo de Componentes se identificaron dos requerimientos del negocio, los cuales representan una solución y aportan mejoras al proceso de realización de las consultorías a centros de datos. Cada uno de los requisitos se describen a continuación:

- **Trabajar sin conexión:** La herramienta **Aplicación_Diagnóstico** deberá permitir a los especialistas del departamento realizar el diagnóstico en cualquier lugar que se encuentren, ya sea con conexión a internet o no.
- **Diseñar topología física de red:** La herramienta **Diseñar_Topología** deberá permitir a los especialistas hacer un diseño de la topología física de red de la institución a la cual se le realiza la consultoría y promover el uso de aplicaciones desarrolladas en el Centro de Telemática de la Universidad de las Ciencias Informáticas.

2.2.2. Requisitos funcionales

Los requisitos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe

comportar en situaciones particulares. En algunos casos, los RF de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Russell, Bennett, y Ghosh, 2019).

A continuación, se identifican los RF de la nueva versión del Sistema para automatizar la elaboración de documentos de consultorías a centros de datos que se quiere desarrollar, los cuales fueron validados y aceptados por el cliente (Ver Anexo 2)

RF1. Gestionar debilidades: el sistema deberá permitir gestionar (insertar, listar, modificar y eliminar) las debilidades que el usuario va introduciendo. Las debilidades se guardan en forma de listado.

RF2. Gestionar amenazas: el sistema deberá permitir gestionar (insertar, listar, modificar, eliminar) las amenazas que el usuario va introduciendo. Las amenazas se guardan en forma de listado.

RF3. Gestionar fortalezas: el sistema deberá permitir gestionar (insertar, listar, modificar, eliminar) las fortalezas que el usuario va introduciendo. Las fortalezas se guardan en forma de listado.

RF4. Gestionar oportunidades: el sistema deberá permitir gestionar (insertar, listar, modificar, eliminar) las oportunidades que el usuario va introduciendo. Las oportunidades se guardan en forma de listado.

2.2.3. Requisitos no funcionales

Las características no funcionales son aquellas características o cualidades que debe cumplir el sistema. Además, constituyen las propiedades o cualidades que el producto debe poseer para su correcto funcionamiento. A continuación, deben tenerse en cuenta los siguientes requisitos no funcionales:

Usabilidad:

- La aplicación deberá contar con una interfaz sencilla, descriptiva y fácil de usar por cualquier usuario con conocimientos básicos de computación.
- Debe estar adaptada a las pautas de la estrategia visual y marcaria de la Universidad de las Ciencias Informáticas.

Servidor

Software:

- Se recomienda hacer uso de algún navegador web.

Hardware:

- CPU Dual Core a 1.0 GHz o superior, 512 MB de RAM o superior, 20 GB de disco duro o superior.

Cliente

Software:

- Es necesario hacer uso de algún navegador web.

Hardware:

- CPU Pentium 300 MHz, 128 MB de RAM, 5 GB de disco duro.

Seguridad

Confidencialidad:

- La información manejada por el sistema está protegida de acceso no autorizado y divulgación, por lo cual se establecerá un nivel de acceso a la aplicación mediante usuario y contraseña, además, de los permisos asignados por usuarios.

Integridad:

- La información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes. Esto se garantiza a través de la integridad

referencial, la cual es una característica básica que proporciona PostgreSQL para garantizar la integridad de los datos almacenados. La integridad referencial es una función que está presente en las bases de datos relacionales que garantiza la coherencia de los datos entre relaciones aparejadas.

Disponibilidad:

- La aplicación deberá estar disponible en todo momento para aquellas personas que necesiten acceder y manejar la información.

2.3. Descripción del sistema

2.3.1. Descripción de los actores que interactúan con el sistema.

Un actor es una entidad externa al sistema representada por un ser humano, una maquina o un software que interactúa con el sistema. Representa un tipo particular de usuario del negocio más que un usuario físico, debido a que varios usuarios físicos pueden realizar el mismo papel en relación al negocio (Ávila, Peña, Lugo y Martínez, 2018).

Tabla 1 Descripción de los actores que interactúan con el sistema.

Actores	Descripción
Usuario	El sistema solo puede ser usado por los miembros del Departamento de Desarrollo de Componentes del Centro de Telemática, por tanto, los usuarios son los propios especialistas del departamento y tienen todos los privilegios dentro mismos para realizar la consultoría y elaborar el informe de diagnóstico y otros documentos relacionados.
Administrador	El administrador del sistema tienes todos los privilegios de los usuarios y además es el único que tiene la facultad de crear y gestionar los usuarios que pueden acceder al mismo.

2.3.2. Descripción de los requisitos funcionales

Los requisitos funcionales son la entrada esencial para realizar el análisis, diseño, implementación y pruebas del sistema. Es por ello que realizar la descripción de los mismos, es de gran importancia. La propuesta de solución presenta un total de 4 historias de usuario (HU). A continuación, se presenta la descripción de los requisitos a partir del modelo propuesto por la Historia de usuarios.

Las Historias de Usuario sirven para describir lo que el usuario desea ser capaz de hacer. Además, las historias de los usuarios se centran en el valor que viene de usar el sistema en lugar de una especificación detallada de lo que el sistema debe hacer. Están concebidos como un medio para fomentar la colaboración.

Las HU definidas se clasificaron atendiendo a diferentes criterios, facilitando así su organización y la correcta planificación del proyecto. Los parámetros definidos en cada una de las HU se detallan a continuación.

Número: para mantener el orden de las historias.

Nombre: para identificar la HU.

Prioridad:

1. **Alta:** constituyen funcionalidades principales del sistema, o forman parte esencial de la arquitectura del mismo.
2. **Media:** con funcionalidades importantes y de gran valor para el usuario pero que no impiden poner el proyecto en marcha si no se tienen.
3. **Baja:** con funcionalidades que sería deseable tener y podrían incluirse en caso de que hubiese recursos para ello.

Riesgo en desarrollo:

1. **Alto:** en caso de tener algún error de implementación, pueden afectar la disponibilidad del sistema.
2. **Medio:** en caso de presentar errores retrasan la entrega de la versión.
3. **Bajo:** en caso de presentar errores, estos pueden ser tratados con facilidad y no afectan el desarrollo del proyecto.

Iteración asignada: número de la iteración a la que ha sido asignada.

Puntos estimados: tiempo estimado de desarrollo para completar la HU. Un punto equivale a una jornada laboral de 8 horas de un día hábil. Los días hábiles son 5 días de la semana, de lunes a viernes.

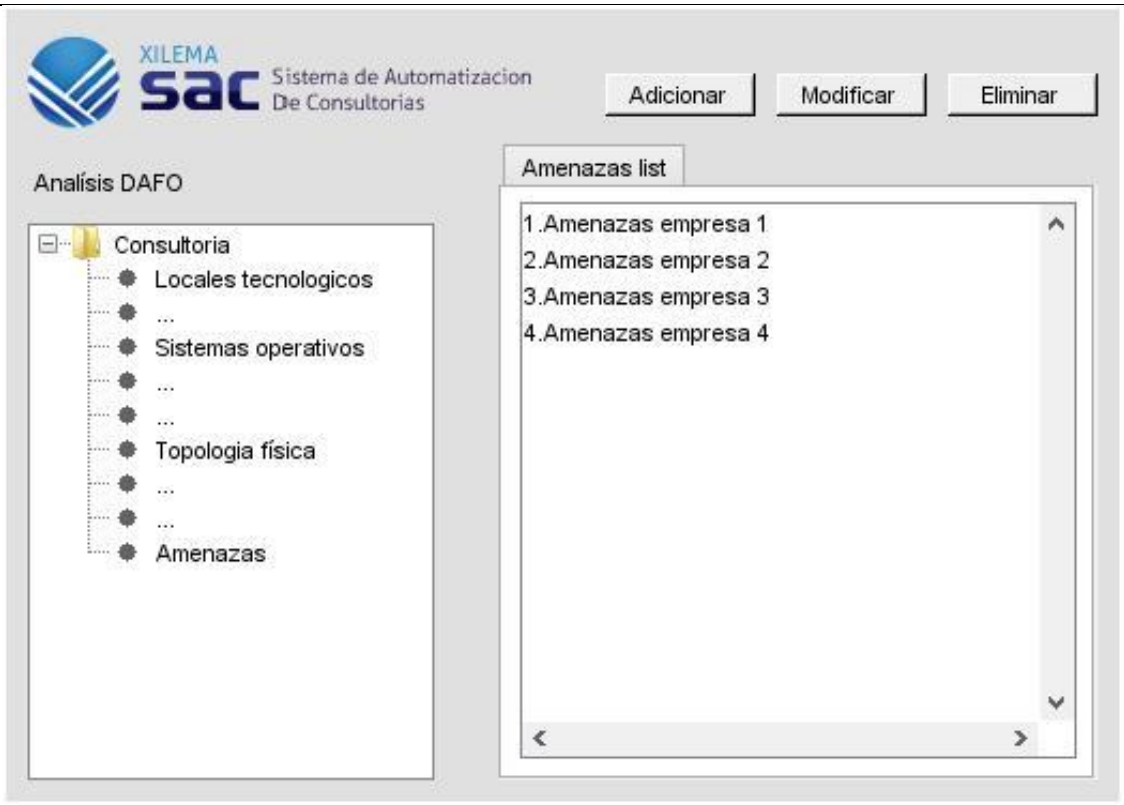
Descripción: es donde se define la funcionalidad que se quiere desarrollar.

Observaciones: detalles a tener en cuenta para desarrollar la HU correctamente.

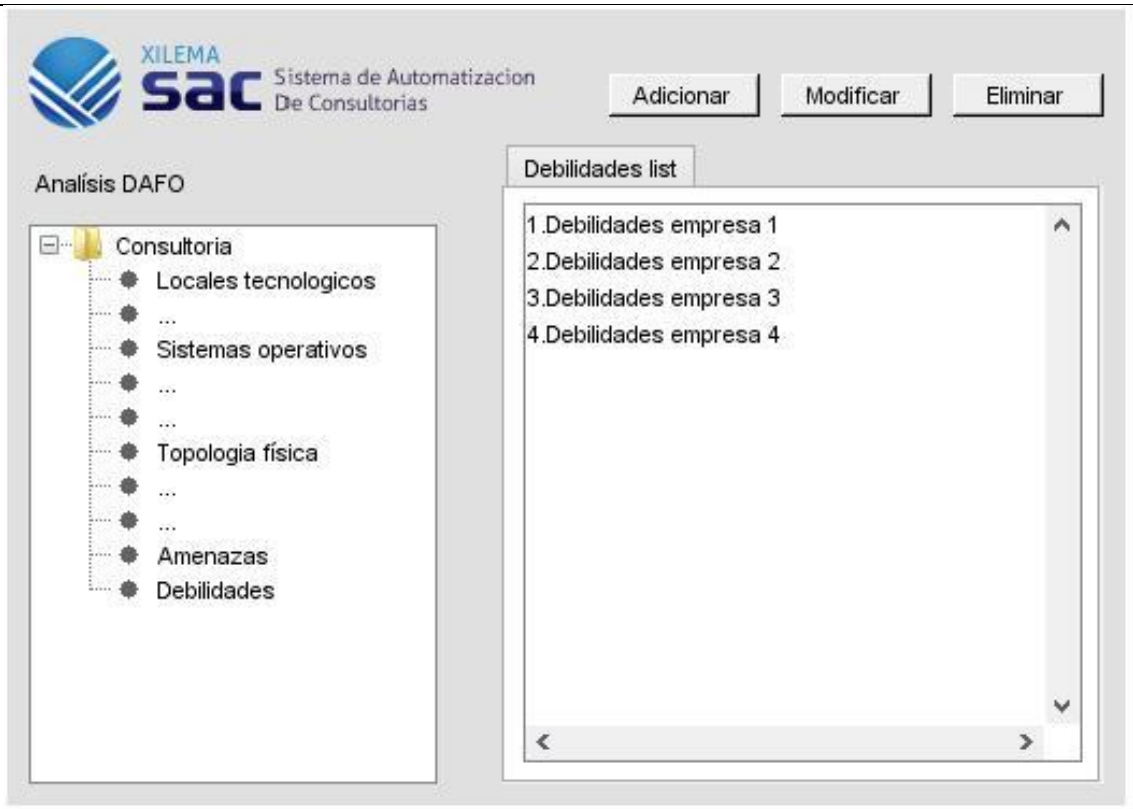
Prototipo de interfaz: es un boceto representativo de la vista de usuario.

Tabla 2 Descripción de los requisitos funcionales.

Numero: 1	**Nombre:** Gestionar amenazas
Prioridad en negocio: Media	**Riesgo de desarrollo:** Bajo
Puntos estimados: 0.2	**Iteración asignada:** 1
Programador responsable: Alejandro Antonio Batista Zaldívar	
Descripción: Permite al usuario gestionar (insertar, listar, modificar, eliminar) las amenazas dentro del módulo de diagnóstico del sistema.	
Observaciones: El usuario puede acceder al sistema siempre y cuando se haya autenticado anteriormente.	
Prototipo de interfaz:	



Numero: 2	**Nombre:** Gestionar debilidades
Prioridad en negocio: Media	**Riesgo de desarrollo:** Bajo
Puntos estimados: 0.2	**Iteración asignada:** 1
Programador responsable: Alejandro Antonio Batista Zaldívar	
Descripción: Permite al usuario gestionar (insertar, listar, modificar, eliminar) las debilidades dentro del módulo de diagnóstico del sistema.	
Observaciones: El usuario puede acceder al sistema siempre y cuando se haya autenticado anteriormente.	
Prototipo de interfaz:	



Numero: 3	**Nombre:** Gestionar oportunidades
Prioridad en negocio: Media	**Riesgo de desarrollo:** Bajo
Puntos estimados: 0.2	**Iteración asignada:** 2
Programador responsable: Alejandro Antonio Batista Zaldívar	
Descripción: Permite al usuario gestionar (insertar, listar, modificar, eliminar) las oportunidades dentro del módulo de diagnóstico del sistema.	
Observaciones: El usuario puede acceder al sistema siempre y cuando se haya autenticado anteriormente.	
Prototipo de interfaz:	

Numero: 4	**Nombre:** Gestionar fortalezas
Prioridad en negocio: Media	**Riesgo de desarrollo:** Bajo
Puntos estimados: 0.2	**Iteración asignada:** 2
Programador responsable: Alejandro Antonio Batista Zaldívar	
Descripción: Permite al usuario gestionar (insertar, listar, modificar, eliminar) las fortalezas dentro del módulo de diagnóstico del sistema.	
Observaciones: El usuario puede acceder al sistema siempre y cuando se haya autenticado anteriormente.	
Prototipo de interfaz:	

2.3.3. Estimación de esfuerzo por Historias de Usuario

Para el desarrollo del sistema propuesto en este trabajo se realizó una estimación del esfuerzo para cada una de las historias de usuario identificadas, llegándose a los resultados que se muestran en la siguiente tabla.

Tabla 3 Estimación de esfuerzo por historia de usuario

No.	Nombre de historia de usuario	Puntos de estimación (días hábiles)
1	Gestionar amenazas	0.2
2	Gestionar debilidades	0.2
3	Gestionar oportunidades	0.2
4	Gestionar fortalezas	0.2

2.3.4. Estimación de esfuerzo por Historias de Usuario

En el plan de iteraciones se establece cuántas iteraciones serán necesarias realizar sobre el sistema para su desarrollo. Sin embargo, se precisa establecer el contenido de trabajo para todas y cada una de ellas, es aquí donde se establece el plan de iteraciones, regulando la cantidad de historias de usuario a implementar dentro del rango establecido por la estimación efectuada. Tomando como referencia los aspectos antes tratados, la aplicación que se pretende construir se desarrollará en 2 iteraciones, explicada más detalladamente a continuación:

Iteración 1

Esta iteración tiene como objetivo la implementación de las historias de usuario Gestionar amenazas y Gestionar debilidades. Durante el transcurso de la misma se creará la base de la arquitectura del sistema con una funcionalidad mínima. Al final de esta se contará con una primera versión de prueba, la cual será mostrada al cliente con el objetivo de obtener una retroalimentación para el equipo de desarrollo.

Iteración 2

El objetivo de esta iteración es la implementación de las historias de usuario Gestionar fortalezas y Gestionar oportunidades. Al finalizar la misma se contará con una versión funcional del sistema, donde se podrán probar las funcionalidades. Esta versión igualmente se mostrará al cliente con el objetivo de realizar cambios necesarios en base a la opinión del mismo.

2.4. 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” (Arias y Durango, 2016). El Sistema para automatizar la elaboración de documentos de consultorías a centros de datos posee una arquitectura cliente-servidor. Puesto que resulta conveniente mantener la misma arquitectura para evitar futuras incompatibilidades, en este caso el sistema a desarrollar adquiere la arquitectura definida por la plataforma.

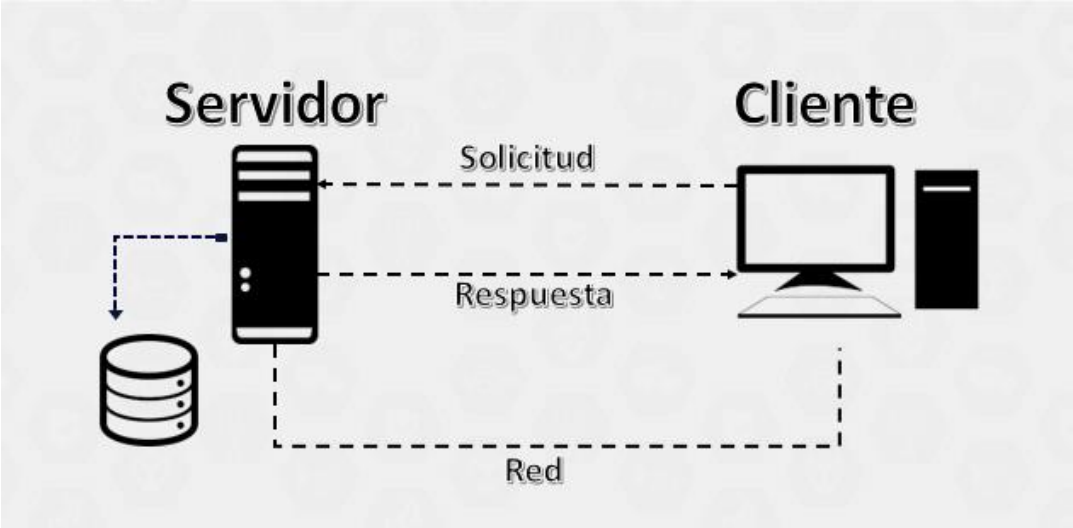


Figura 9 Arquitectura Cliente-Servidor (Fuente: elaboración propia)

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. La interacción cliente-servidor es el soporte de la mayor parte de la comunicación por redes.

2.4.1. Patrón arquitectónico

La arquitectura de Django es una variación del patrón MVC (Modelo Vista Controlador), el cual es una guía de alto nivel para el diseño de arquitecturas de aplicaciones que ofrezcan una fuerte interactividad con usuarios. Django es un *framework* MTV, del inglés "*Model – Template -View*" (Modelo Plantilla Vista), que intenta ser lo más funcional posible. En comparación con el patrón MVC, el Modelo en Django sigue como Modelo, a la Vista en Django se le denomina Plantilla y en el caso del Controlador se le denomina Vista.

Modelo: el modelo define los datos almacenados, se encuentra en forma de clase de Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros; también posee métodos. Todo esto permite indicar y controlar el comportamiento de los datos. En el sistema el modelo se evidencia en los ficheros *models.py.*

Vista: la vista se presenta en modo de funciones en Python y su propósito es determinar qué datos serán visualizados. El ORM[18] de Django permite escribir código Python en lugar de SQL para realizar las consultas que necesita la vista. La autentificación de servicios y la validación de datos a través de formularios son también tareas de las que se encarga la vista. En el sistema las vistas se encuentran implementadas en los ficheros *views.py.*

Plantilla: la plantilla es básicamente una página HTML con algunas etiquetas extras propias de Django, en sí no solo crea contenido en HTML sino también XML[19], CSS[20] y JavaScript. La plantilla recibe los datos de la vista y luego los organiza para la presentación al navegador web (Ríos, Mora, Ordóñez y Sojos, 2016). En el sistema las platillas las conforman los ficheros con extensión *html.*

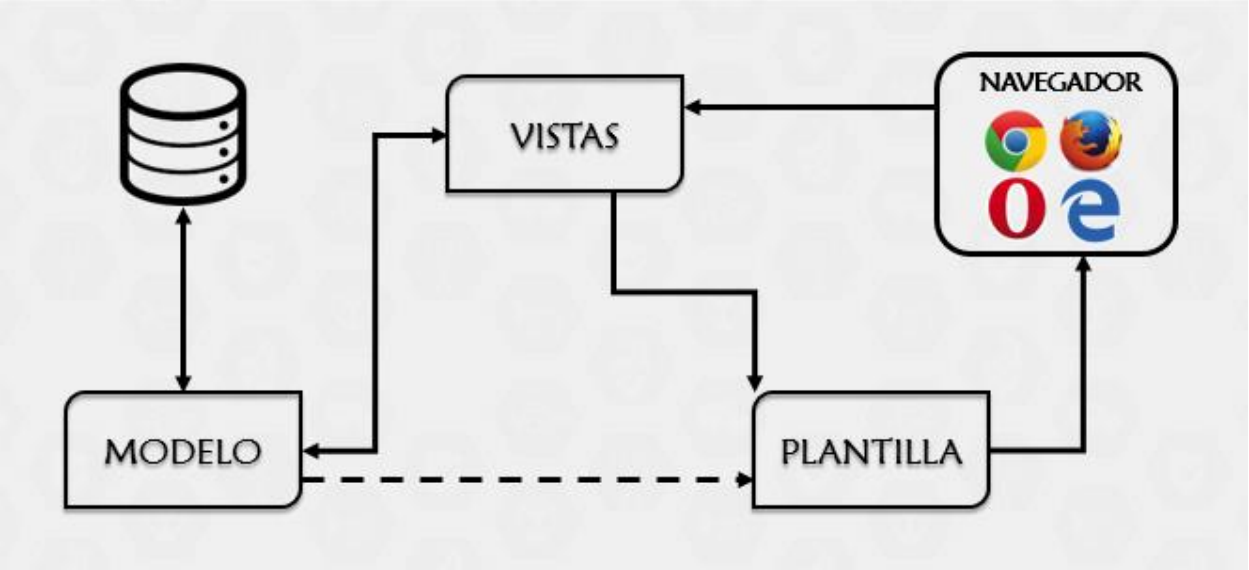


Figura 10 Funcionamiento MTV (Fuente: elaboración propia).

2.5. Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras

[18] *Object-Relational mapping:* mapeo objeto-relacional
[19] *Extensible Markup Language:* Lenguaje de Marcado Extensible
[20] *Cascading Style Sheets:* Hojas de estilo en cascada

restricciones relacionadas con el entorno de implementación, tienen impacto en el módulo, siendo la principal vía de acceso en la actividad de implementación (Unhelkar, Ambler, Armour, Miller, Basili y Valett, 2017).

Entre los artefactos que genera el modelo de diseño se encuentra el Diagrama de Clases del Diseño con estereotipos web donde se muestran las clases, interfaces y colaboraciones, así como las relaciones entre ellas.

A continuación, se expone la representación de los diagramas de clases de diseño:

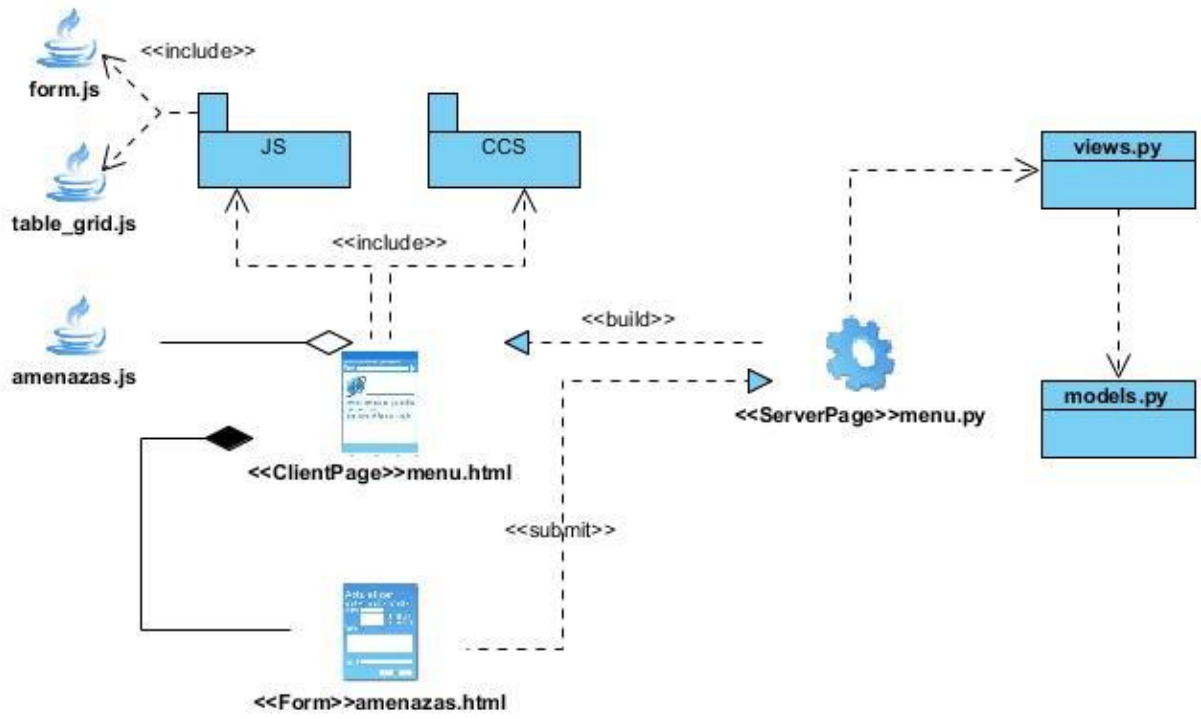


Figura 11 Diagrama de clase del diseño RF: Gestionar amenazas

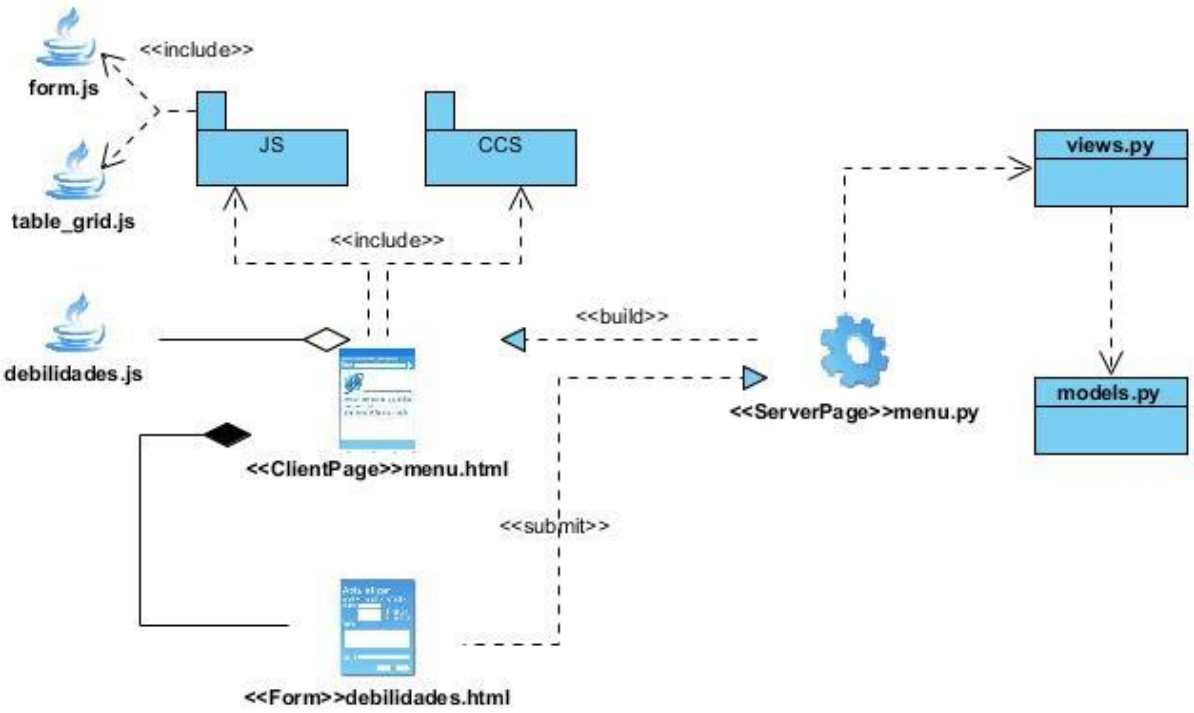


Figura 12 Diagrama de clase del diseño RF: Gestionar debilidades

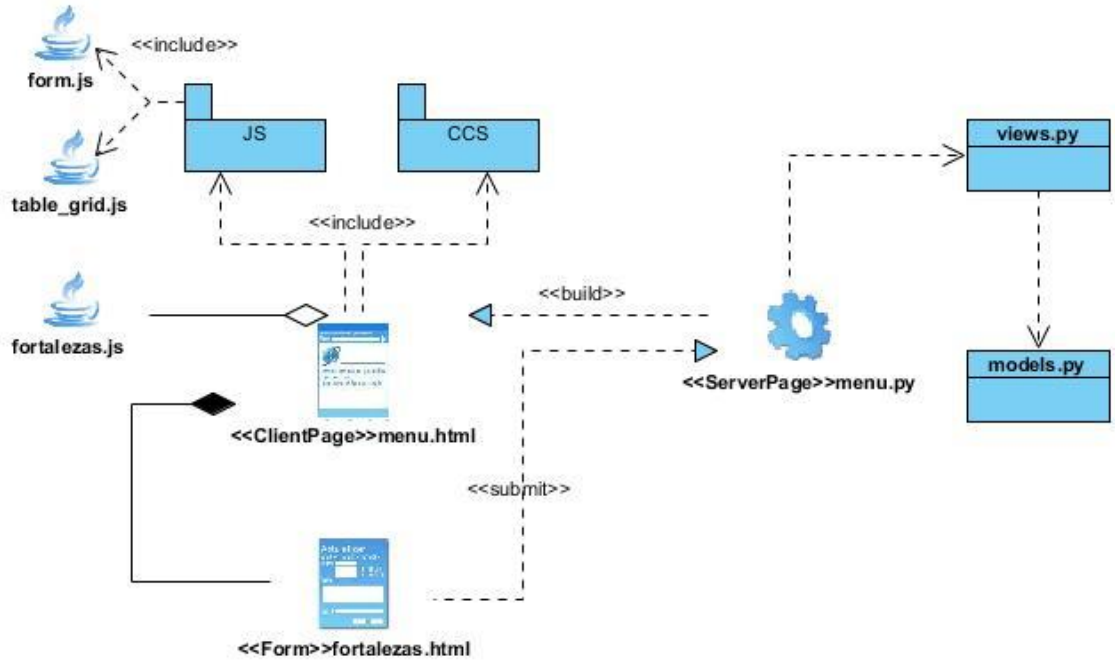


Figura 13 Diagrama de clase del diseño RF: Gestionar fortalezas

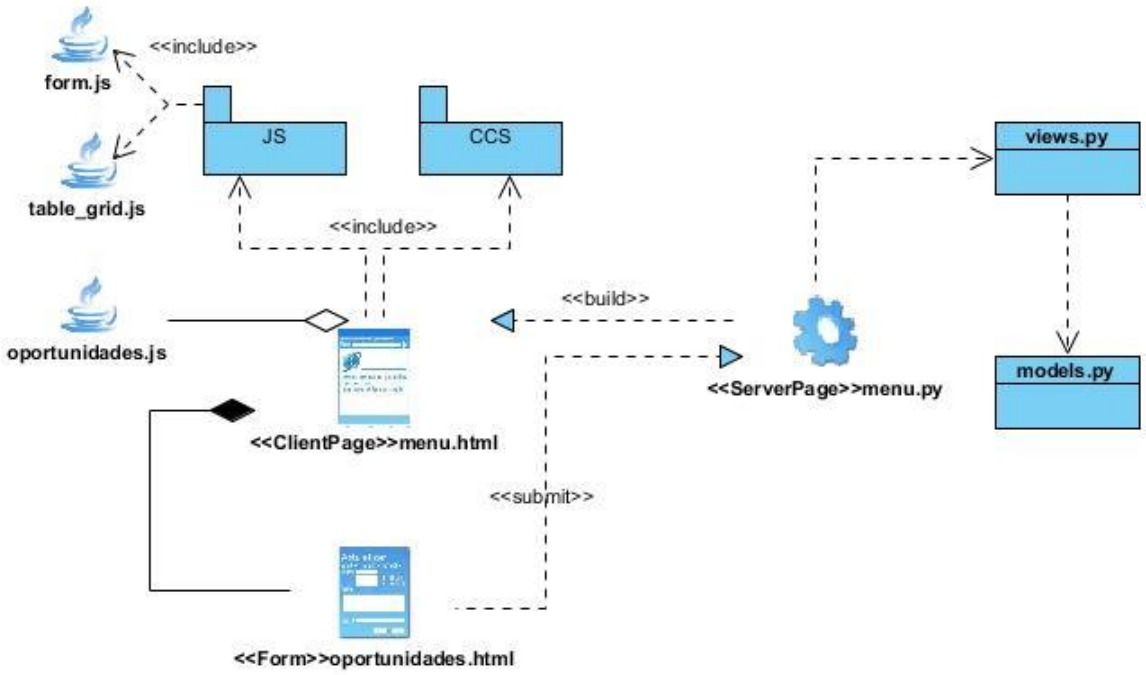


Figura 14 Diagrama de clase del diseño RF: Gestionar oportunidades

El *framework* Xilema Base Web cuenta con una serie de directorios que se estructuran de la siguiente manera:

src: en esta carpeta estará incluido todo el código de nuestro proyecto.

xilema: aquí estará los ficheros de configuración de nuestro proyecto como **settings.py**.

src/apps: carpeta donde estarán todas las aplicaciones.

src/locale: aquí se encuentran la internacionalización (i18n) general del proyecto, puedes tener una carpeta de i18n en cada aplicación.

src/media: utilizada para la gestión de archivos almacenados.

src/static: debemos tener las carpetas con cada tipo de contenido estático que decidamos incluir, en este caso tendremos 4 subdirectorios en static: **apps** (contenido estático de cada una de las aplicaciones realizadas), **css** (código css del proyecto), **images** (todas las imágenes del proyecto) y **js** (código css del proyecto).

src/vendor: aquí son ubicados componentes desarrollados por el Departamento de Desarrollo de Componentes como las aplicaciones en la carpeta **apps** y librerías **libs** de dicho directorio.

2.5.1 Patrones de diseño

Los patrones de diseño constituyen la base para la búsqueda de soluciones a problemas comunes en el proceso de desarrollo de software. Se caracterizan por su reusabilidad, flexibilidad y aplicabilidad a diferentes problemas de diseño en diversas circunstancias (Luna García, Mendoza González y Álvarez Rodríguez, 2015). El patrón es un esquema de solución que se aplica a un tipo de situación, esta aplicación del mismo no es mecánica, sino que requiere de adaptación y matrices.

Por otra parte (Gros, Escofe y Marimón, 2016) consideran que un patrón de diseño describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular y tienden a ser independientes de los lenguajes y paradigmas de programación. Para el diseño de la aplicación se hizo uso de los Patrones Generales de Software para Asignar Responsabilidades (GRASP), así como de los patrones de diseño.

Los Patrones (GRASP), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Los patrones GRASP utilizados fueron:

Alta cohesión: se aplica en la mayoría de las clases del diseño, ya que en cada una solo se implementan las funcionalidades que le corresponden.

Bajo acoplamiento: es la idea de tener las clases lo menos ligadas entre sí como sea posible. De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, lo que potencia la reutilización, y disminuye la dependencia entre las clases.

Ejemplo: los modelos se relacionan con un número pequeño de clases, como es el caso de la clase *institución*, ya que la mayoría de las clases del sistema están relacionadas únicamente con la clase institución.

Creador: el patrón creador ayuda a identificar quién debe ser el responsable de la creación o instanciación de nuevos objetos o clases, de forma tal que una instancia de un objeto solo pueda ser creada por el objeto que contiene la información necesaria para ello.

Ejemplo: el patrón se encuentra evidenciado en la utilización de la clase *institucion* la cuál es la única que contiene la información que se necesita para crear una instancia de ella.

Controlador: el patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, este sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

Ejemplo: el patrón se evidencia en las clases *serializers*, las cuales determinan los datos que se mostrarán en la plantilla.

Experto: se mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener.

Los patrones de diseño GoF, describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos. Los utilizados en el diseño de la solución son:

Active Record: el objetivo de este patrón es encapsular el acceso a una base de datos relacional y agrega lógica de dominio en esa información. Se evidencia en la implementación de los modelos de Django.

Identity Field: consiste en que las tablas de las bases de datos relacionales se encuentren identificadas. Se evidencia en las llaves primarias generadas a partir de la creación de los modelos de Django en la cual a cada tabla se le incorpora automáticamente la columna id (primary_key).

Command Pattern: encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma. Se evidencia en el siguiente fragmento de código, en el cual se aprecia cómo se encapsula la información generada en un objeto tipo HttpResponse.

Front Controller (Controlador Frontal): Django posee una implementación de Controlador Frontal que despacha las peticiones hacia métodos o clases, que en la práctica son páginas controladoras. Antes del despacho, la petición es procesada por varios filtros.

Decorator (Decorador): añade responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. En el siguiente ejemplo de código, se evidencia cómo se agrega la condición de método estático a una función a través del decorador *@staticmethod*.

2.6. Modelo de datos

El origen del modelo entidad-relación se encuentra en los estudios que se efectuaron por Peter Chen en la década de 1970. Este modelo se basa en la percepción del mundo real por medio de un conjunto de objetos básicos que serán las entidades y las relaciones existentes entre estos objetos (Díaz, 2016).

El modelo de datos se desarrolló a partir de la base de datos del conjunto de clases persistentes y sus asociaciones en el modelo de diseño, para su representación se utilizó la herramienta Visual Paradigm 8.0. En la Figura 15 se presentan el diagrama correspondiente al modelo entidad-relación de base de datos utilizada para el manejo de información del sistema propuesto.

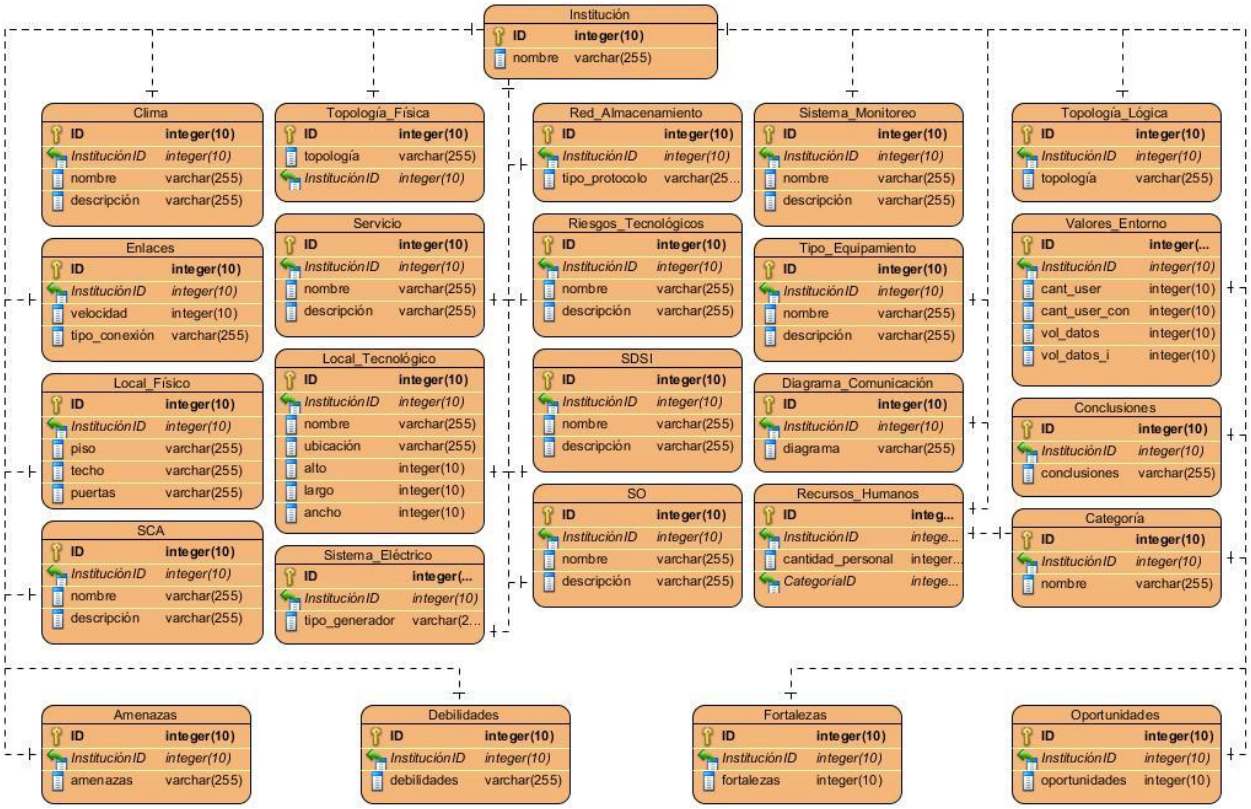


Figura 15 Modelo Entidad- Relación

Conclusiones del capítulo

1. Los requisitos funcionales y no funcionales identificados a partir del proceso de obtención de los requisitos y los artefactos generados, constituyeron una guía fundamental para la construcción de la propuesta de solución.

2. A partir de la definición de las HU se pudieron identificar las principales funcionalidades, así como la estimación del tiempo para la implementación de las mismas y el número de iteraciones necesarias para su posterior entrega el cliente.

3. Con la utilización del patrón arquitectónico MVT y la identificación de los patrones GRASP y GoF, se garantiza una mayor organización, reutilización de funciones y código legible.

Capítulo III Implementación y pruebas

En el presente capítulo se muestran los artefactos ingenieriles relacionados con la implementación y validación del sistema a desarrollar. Se describe el modelo de implementación, con los respectivos artefactos generados dentro de este. Además, se detallan las pruebas realizadas al sistema y los resultados de las mismas.

3.1. Modelo de implementación

El modelo de implementación permite representar como se implementan en términos de componentes, los elementos del modelo de diseño. Describe también como dependen los componentes unos de otros, y como se organizan de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en los lenguajes de programación utilizados (González, Calvache y Gómez, 2015).

3.2. Estándares de codificación

El código fuente es un conjunto de líneas que conforman un bloque de texto, escrito según las reglas sintácticas de algún lenguaje de programación destinado a ser legible por humanos. Es un programa en su forma original, tal y como fue escrito por el programador, no es ejecutable directamente por el computador, debe convertirse en lenguaje de máquina mediante compiladores, ensambladores o intérpretes (Campos y Martínez, 2015). Los estándares de programación mantienen e código legible para los miembros del equipo, facilitando los cambios. Para lograr este objetivo, se utilizó la Guía de estilo para el código Python (PEP 8) («PEP 8 -- *Style Guide for Python Code*»). Esta guía posee una gran cantidad de convenciones para escribir código legible, dentro de las cuales se destacan:

- Usar cuatro espacios por indentación.
- Nunca mezclar tabulaciones y espacios.
- Limitar todas las líneas a un máximo de caracteres (120 en este proyecto).
- Separar funciones de alto nivel y definiciones de clase con dos líneas en blanco, mientras que las definiciones de métodos dentro de una clase son separadas por una línea en

blanco.

- Codificación UTF-8 en todos los módulos.
- Las importaciones deben estar en líneas separadas.
- Evitar espacios en blanco innecesarios.
- Utilizar el estilo *CamelCase* para nombrar clases, y el *lower_case_with_underscores* para funciones y métodos.

3.3. Modelo de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo (Unhelkar, Ambler, Armour, Miller, Basili y Valett, 2017). Con el objetivo de representar los recursos de cómputo que necesita el módulo para su correcto funcionamiento se muestra a continuación el diagrama de despliegue (ver Figura 16) correspondiente a la solución propuesta.

Figura 16 Diagrama de despliegue.

En el diagrama se representa un nodo llamado **PC_Cliente** que realiza peticiones al nodo **Servidor Web** a través del puerto *HTTPS:443* y este a su vez se encuentra conectado al nodo **Servidor BD** por el puerto *TCP/IP: 5432*.

3.4. Proceso de pruebas

Una vez finalizada la implementación del sistema, se hace necesario comprobar que su funcionamiento sea correcto, verificando que las funcionalidades se ajusten a las especificaciones planteadas. Para ello se realizan las pruebas de software, con el fin de detectar defectos y asegurar que estos sean corregidos antes de la entrega del producto al cliente.

Las pruebas son actividades en las cuales un sistema o componente es ejecutable bajo condiciones o requerimientos específicos permitiendo que los resultados sean observados y registrados, estas se realizan con el objetivo de encontrar deficiencias existentes en el software (Unhelkar, Ambler, Armour, Miller, Basili y Valett, 2017). Durante el flujo de trabajo de pruebas se verifica el resultado final de la implementación, probando la estructura; tanto en la construcción interna como intermedia, así como las versiones finales.

3.4.1. Tipos de pruebas

Existe toda una tipología de pruebas aplicables a distintos contextos de desarrollo de aplicaciones. Según (Pressman y Maxim, 2016) existen siete tipos fundamentales de pruebas recomendadas sobre aplicaciones web las cuales son: pruebas de contenido, interfaz, navegación, componentes, configuración, seguridad y desempeño o rendimiento.

Para la presente solución se requiere la realización de un grupo de pruebas que redunden en la correcta validación del sistema, como son:

- **Pruebas unitarias**: son pruebas que realiza el equipo de desarrollo y que, según Pressman: verifiquen que los componentes unitarios están codificados bajo condiciones de robustez, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada (Pressman y Maxim, 2016).
- **Prueba de aceptación**: representa aquella fase del ciclo de vida de desarrollo de software en el que el equipo de desarrollo y el área usuaria de un sistema de información tiene que garantizar que el sistema desarrollado se corresponde con los requerimientos definidos (Ponce, Dominguez, Gutierrez y Escalona, 2014).

3.4.2. Aplicación de las pruebas

No todos los productos de software requieren la aplicación de todos los tipos de pruebas que existen, ya que cada una de ellas es aplicada para comprobar un determinado objetivo en determinado momento del ciclo de vida del software. A continuación, se muestra la aplicación de las pruebas que se le efectuaron a la solución.

Pruebas Unitarias

Se utilizó el módulo doctest, que forma parte de la librería estándar de Python. Este módulo busca en los comentarios del código, fragmentos que se vean como sesiones del intérprete interactivo de Python, y procede a ejecutar dichos fragmentos, verificando que resulten como se les ha indicado. Esto significa, que importando el módulo doctest, éste, buscará en los comentarios de nuestro código, todo fragmento que represente al interprete interactivo, para luego ejecutarlo.

Este tipo de pruebas fue aplicada a las clases y métodos implementados durante el desarrollo de la solución propuesta. Se realizaron 22 pruebas unitarias, las cuales se ejecutaron satisfactoriamente.

En la primera iteración se realizaron ocho pruebas unitarias. Los resultados de su ejecución con el entorno integrado de desarrollo pueden observarse en la figura 17.

```
1 items had no tests:
    __main__
8 items passed all tests:
   1 tests in __main__.clima_test
   1 tests in __main__.enlaces_test
   1 tests in __main__.local_fis_test
   1 tests in __main__.locales_tec_test
   1 tests in __main__.nombre_inst_test
   1 tests in __main__.sistema_control_test
   1 tests in __main__.sistema_electrico_test
   1 tests in __main__.sistemas_monitoreo_test
8 tests in 9 items.
8 passed and 0 failed.
Test passed.

Process finished with exit code 0
```

Figura 17 Resultados de las pruebas unitarias en la primera iteración.

En la segunda iteración se realizaron ocho pruebas unitarias. Los resultados de su ejecución con el entorno integrado de desarrollo pueden observarse en la figura 18.

```
1 items had no tests:
    __main__
8 items passed all tests:
   1 tests in __main__.certificaciones_test
   1 tests in __main__.diagrama_comunicacion_test
   1 tests in __main__.red_almacenamiento_test
   1 tests in __main__.rh_test
   1 tests in __main__.riesgos_tecnologicos_test
   1 tests in __main__.servicios_test
   1 tests in __main__.sistema_deteccion_test
   1 tests in __main__.tipo_equipamiento_test
8 tests in 9 items.
8 passed and 0 failed.
Test passed.

Process finished with exit code 0
```

Figura 18 Resultado de las pruebas unitarias en la segunda iteración

En la tercera y última iteración se realizaron seis pruebas unitarias. Los resultados de su ejecución con el entorno integrado de desarrollo pueden observarse en la figura 19.

```
1 items had no tests:
    __main__
6 items passed all tests:
   1 tests in __main__.exportar_diag_test
   1 tests in __main__.exportar_dis_test
   1 tests in __main__.sistemas_operativos_test
   1 tests in __main__.topologia_fisica_test
   1 tests in __main__.topologia_logica_test
   1 tests in __main__.valores_entorno_test
6 tests in 7 items.
6 passed and 0 failed.
Test passed.

Process finished with exit code 0
```

Figura 19 Resultado de las pruebas unitarias en la tercera iteración

Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Las pruebas de aceptación son consideradas como pruebas de caja negra. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución (Joskowicz 2008).

Para realizar las pruebas de aceptación correctamente se elaboraron casos de prueba para cada una de ellas. A continuación, se describen los casos de prueba de aceptación de las HU y de los requisitos de negocio identificados:

Tabla 4 Caso de Prueba de Aceptación-Insertar amenazas.

Caso de prueba de Aceptación
Nombre: insertar amenazas
Historia de usuario: Gestionar amenazas
Descripción: Se prueba que se inserte correctamente las amenazas
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 5 Caso de Prueba de Aceptación-Listar amenazas.

Caso de prueba de Aceptación
Nombre: listar amenazas
Historia de usuario: Gestionar amenazas
Descripción: Se prueba que se listen correctamente las amenazas
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, listar, modificar y eliminar elementos del listado.

Tabla 6 Caso de Prueba de Aceptación-modificar amenazas

Caso de prueba de Aceptación
Nombre: modificar amenazas
Historia de usuario: Gestionar amenazas
Descripción: Se prueba que se modifiquen correctamente las amenazas
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 7 Caso de Prueba de Aceptación-eliminar amenazas

Caso de prueba de Aceptación
Nombre: eliminar amenazas
Historia de usuario: Gestionar amenazas
Descripción: Se prueba que se elimine correctamente todas las amenazas de una institución seleccionada.
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 8 Caso de Prueba de Aceptación-insertar debilidades

Caso de prueba de Aceptación
Nombre: insertar debilidades
Historia de usuario: Gestionar debilidades
Descripción: Se prueba que se inserte correctamente las debilidades
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 9 Caso de Prueba de Aceptación-Listar debilidades

Caso de prueba de Aceptación
Nombre: listar debilidades
Historia de usuario: Gestionar debilidades
Descripción: Se prueba que se listen correctamente las debilidades
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, listar, modificar y eliminar elementos del listado.

Tabla 10 Caso de Prueba de Aceptación-modificar debilidades

Caso de prueba de Aceptación
Nombre: modificar debilidades
Historia de usuario: Gestionar debilidades
Descripción: Se prueba que se modifiquen correctamente las debilidades
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 11 Caso de Prueba de Aceptación-eliminar debilidades

Caso de prueba de Aceptación
Nombre: eliminar debilidades
Historia de usuario: Gestionar debilidades
Descripción: Se prueba que se elimine correctamente todas las debilidades de una institución seleccionada.
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 12 Caso de Prueba de Aceptación-insertar fortalezas

Caso de prueba de Aceptación
Nombre: insertar fortalezas
Historia de usuario: Gestionar fortalezas
Descripción: Se prueba que se inserte correctamente las fortalezas
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 13 Caso de Prueba de Aceptación-listar fortalezas

Caso de prueba de Aceptación
Nombre: listar fortalezas
Historia de usuario: Gestionar fortalezas
Descripción: Se prueba que se listen correctamente las fortalezas
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, listar, modificar y eliminar elementos del listado.

Tabla 14 Caso de Prueba de Aceptación-modificar fortalezas

Caso de prueba de Aceptación
Nombre: modificar fortalezas
Historia de usuario: Gestionar fortalezas
Descripción: Se prueba que se modifiquen correctamente las fortalezas
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 15 Caso de Prueba de Aceptación-eliminar fortalezas

Caso de prueba de Aceptación
Nombre: eliminar fortalezas
Historia de usuario: Gestionar fortalezas
Descripción: Se prueba que se elimine correctamente todas las fortalezas de una institución seleccionada.
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 16 Caso de Prueba de Aceptación-insertar oportunidades

Caso de prueba de Aceptación
Nombre: insertar oportunidades
Historia de usuario: Gestionar oportunidades
Descripción: Se prueba que se inserte correctamente las oportunidades
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 17 Caso de Prueba de Aceptación-listar oportunidades

Caso de prueba de Aceptación
Nombre: listar oportunidades
Historia de usuario: Gestionar oportunidades
Descripción: Se prueba que se listen correctamente las oportunidades
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, listar, modificar y eliminar elementos del listado.

Tabla 18 Caso de Prueba de Aceptación-modificar oportunidades

Caso de prueba de Aceptación
Nombre: modificar oportunidades
Historia de usuario: Gestionar oportunidades
Descripción: Se prueba que se modifiquen correctamente las oportunidades
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 19 Caso de Prueba de Aceptación-eliminar oportunidades

Caso de prueba de Aceptación
Nombre: eliminar oportunidades
Historia de usuario: Gestionar oportunidades
Descripción: Se prueba que se elimine correctamente todas las oportunidades de una institución seleccionada.
Precondiciones: El usuario debe estar autenticado y debe haber alguna institución insertada en el sistema.
Pasos de ejecución: • El usuario accede al módulo Diagnóstico. • Se muestra un listado de elementos. • El usuario será capaz de añadir, modificar y eliminar elementos del listado.

Tabla 20 Caso de Prueba de Aceptación-Diseñar topología física

Caso de prueba de Aceptación
Nombre: diseñar topología física
Historia de usuario: Diseñar topología física
Descripción: Se prueba que la herramienta **Diseñar_Topología** muestre la interfaz para poder diseñar una topología física de red.
Precondiciones:
Pasos de ejecución: • El usuario accede a la herramienta. • Selecciona cada uno de los elementos que desea incluir en el diseño de la topología. • Selecciona la opción guardar imagen. • Selecciona la ubicación donde quiere almacenar el fichero con extensión (.png).

Tabla 21 Caso de Prueba de Aceptación-Trabajar sin conexión

Caso de prueba de Aceptación
Nombre: trabajar sin conexión
Historia de usuario: Trabajar sin conexión
Descripción: Se prueba que la herramienta **Aplicación_Diagnóstico** funcione sin conexión e inserte los datos correctamente en la base de datos local SQLite.
Precondiciones: El usuario debe estar autenticado en la aplicación.
Pasos de ejecución: • El usuario accede a la herramienta **Aplicación_Diagnóstico** • Selecciona la opción Crear Diagnóstico. • Introduce los datos en el formulario. • Presiona el botón guardar. • Aparecen una tabla con los datos que han sido guardados en la base de datos.

Las pruebas de aceptación se realizaron a las HU 1 a la 4 en una primera iteración, las que arrojaron 4 no conformidad la cual fue corregida en su totalidad. En la segunda iteración se elaboraron los casos de pruebas enfocados a los requisitos del negocio (Diseñar topología física y trabajar sin conexión), en estos se encontraron 2 no conformidad que fueron resueltas.

Tabla 22 No conformidades detectadas por iteraciones.

No.	No Conformidades	No. de Iteración
1	No permite seleccionar la institución al momento de adicionar las amenazas.	1
2	No permite seleccionar la institución al momento de adicionar las debilidades.	1
3	Al momento de eliminar las oportunidades no se muestra la ventana emergente que pregunta al usuario está seguro de querer realizar la operación.	1

4	El sistema permite adicionar las fortalezas sin haber seleccionado una institución con anterioridad.	1
5	La herramienta no permite insertar un *router* en el diseño de la topología física de red	2
6	La herramienta no lista los datos guardados en la base de datos.	2

En el siguiente gráfico se resume los resultados de las pruebas de aceptación durante cada iteración del proceso de desarrollo de la solución propuesta.

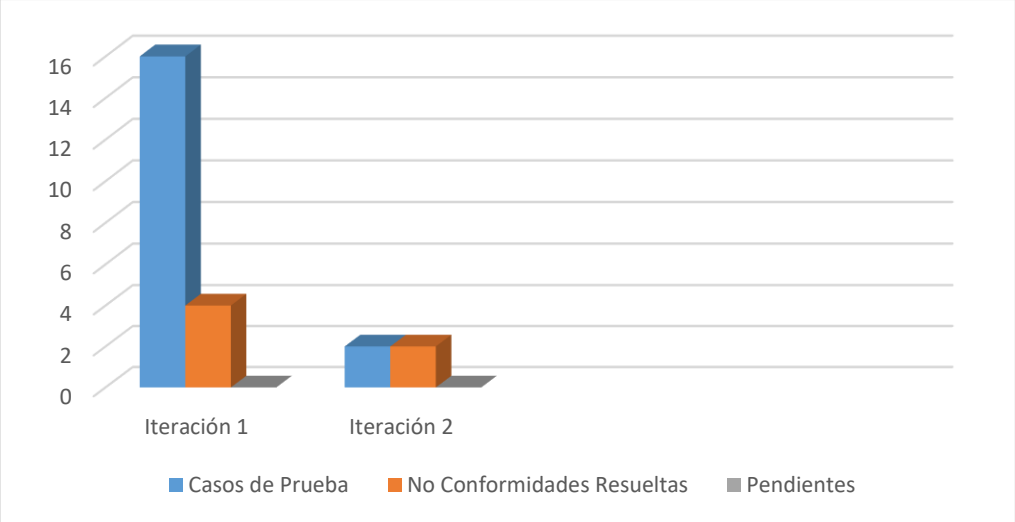


Figura 20 Comportamiento de las no conformidades por iteraciones

Conclusiones del capítulo

1. En este capítulo se explicó a partir de los resultados del diseño, la implementación del sistema para automatizar la elaboración de documentos de consultorías a centros de datos.
2. Mediante el diagrama de despliegue quedó definida la distribución física y lógica de la arquitectura del sistema y sus conexiones.

3. Se realizaron las pruebas unitarias y de aceptación con resultados satisfactorios, lo que contribuyó a elevar la calidad final del producto, validando cada una de las funcionalidades para evitar que el sistema llegue con errores al cliente.

Conclusiones

En el presente trabajo se desarrolló la versión 2.0 del Sistema para automatizar la elaboración de documentos de consultorías a centros de datos, el cual facilita el trabajo de los especialistas del Departamento de Desarrollo de Componentes que se encargan de realizar las consultorías a centros de datos, dando cumplimiento al objetivo propuesto al inicio de la investigación. Adicionalmente se concluye que:

1. Los fundamentos teóricos y metodológicos referentes a los procesos que se llevan a cabo en las instituciones que realizan consultorías, permitió sentar las bases de la investigación y proponer mejoras para los procesos que realiza el Departamento de Desarrollo de Componentes.
2. Se diseñó la propuesta de solución del componente, a partir de los artefactos correspondientes a la metodología de desarrollo AUP-UCI en su escenario 4, teniendo en cuenta el patrón arquitectónico Modelo-Vista-Plantilla, lo cual facilitó su implementación.
3. Con la implementación de las aplicaciones **Aplicación_Diagnóstico** y **Diseñar_Topología,** se obtuvieron dos herramientas que representan una mejora en las consultorías a centros de datos que realiza el Departamento de Desarrollo de Componentes.
4. La validación del componente mediante la aplicación de las pruebas de software: unitarias y de aceptación arrojaron resultados satisfactorios, quedando demostrado que el componente cumple con los requisitos funcionales.

Recomendaciones

Los resultados obtenidos en este trabajo sugieren una serie de recomendaciones para seguir perfeccionando la gestión del proceso de consultorías a centros de datos que realiza el Departamento de Desarrollo de Componentes del Centro de Telemática:

- Continuar con el desarrollo de la aplicación propuesta, agregando nuevas funcionalidades que permitan calcular el nivel de informatización del centro de datos visitado.
- Desarrollar una aplicación para el sistema operativo Android para que el servicio de consultorías pueda ser accedido mediante dispositivos móviles.
- Implementar una funcionalidad que permita sincronizar los datos almacenados en la base de datos SQLite generada por la **Aplicación_Diagnóstico** con la base de datos PostgreSQL del Sistema para automatizar la elaboración de documentos de consultorías a centros de datos.

Referencias Bibliográficas

1. **ALIAGA, F. M., GUTIÉRREZ-BRAOJOS, C., & FERNÁNDEZ-CANO, A., 2018**. Las revistas de investigación en educación: Análisis DAFO. *ASOCIACIÓN INTERUNIVERSITARIA DE INVESTIGACIÓN PEDAGÓGICA (AIDIPE) MIEMBRO DE LA EUROPEAN EDUCATIONAL RESEARCH (EERA)*, *36*(2), 563.
2. **ÁVILA, A. E. S., UN, F., PEÑA, M. J. A. C., LUGO, A. J. D., & MARTÍNEZ, A. R., 2018.** Utilidad del Lenguaje Unificado de Modelado (UML) en el desarrollo de software profesional dentro del sector empresarial y educativo.
3. **ARIAS, Á., & DURANGO, A., 2016.** *Ingeniería y Arquitectura del Software: 2ª Edición*. IT *Campus Academy*.
4. **CALLEJA, GUILLERMO Y CUTULI, ROBERTO, 2017.** Despliegue de un Sistema de Almacenamiento Distribuido (Ceph) en el Centro de Datos Universitario. El caso de la Universidad Nacional de Cuyo. 2017.
5. **CASTILLO, A. A., 2017.** *Curso de Programación Web: JavaScript, Ajax y jQuery*: IT Campus Academy.
6. **CAMPOS, S. G., & MARTÍNEZ, L. F. F., 2015.** Programación Extrema: Prácticas, Aceptación y Controversia. *Cultura Científica y Tecnológica*, (15).
7. **DUARTE, L., TEODORO, A. C. M., & FREITAS, A., 2017**. *GIS Open Source Application as a Support to a Hospital Morbidity Database*.
8. **DÍAZ, A. E., 2016.** Sitio Web "Aprender Modelo Entidad-Relación". *Revista Conrado*, *12*(56).
9. **GROS, B., ESCOFET, A., & MARIMÓN, M., 2016**. Los patrones de diseño como herramientas para guiar la práctica del profesorado/*The design patterns as tools to guide the practice of teachers. Revista Latinoamericana de Tecnología Educativa-RELATEC*, *15*(3), 11-25.
10. **GONZÁLEZ, J. D. Y., CALVACHE, C. J. P., & GÓMEZ, O. S. G., 2015**. Revisión sistemática acerca de la implementación de metodologías ágiles y otros modelos en micro, pequeñas y medianas empresas de software. *Revista Tecnológica-ESPOL*, *28*(5).
11. **GONZÁLEZ, J. P., DOMINGUEZ MAYO, F. J., GUTIÉRREZ RODRÍGUEZ, J. J., & ESCALONA CUARESMA, M. J., 2014.** Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental. *Ibersid*, *8*.
12. **ISLAM, Q. N., 2015.** *Mastering PyCharm*. Packt Publishing Ltd
13. **JOSKOWICZ, J., 2008.** Reglas y prácticas en *eXtreme Programming*. *Universidad de Vigo*, *22*.

14. **KARIM, S., LIAWATIMENA, S., TRISETYARSO, A., ABBAS, B. S., & SUPARTA, W., 2017**. *Automating functional and structural software size measurement based on XML structure of UML sequence diagram*. In *2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)* (pp. 24-28). IEEE.
15. **KENNEDY, B., & MUSCIANO, C., 2017**. *HTML & XHTML. The Definitive Guide*.
16. **LUNA GARCÍA, H., MENDOZA GONZÁLEZ, R., & ÁLVAREZ RODRÍGUEZ, F. J., 2015.** Patrones de diseño para mejorar la accesibilidad y uso de aplicaciones sociales para adultos mayores.
17. **MOLINA, C.E., 2017**. Fundamentos de Redes: Topologías de Red. . S.l.:
18. **OH, G., KIM, S., LEE, S. W., & MOON, B., 2015.** *SQLite optimization with phase change memory for mobile applications. Proceedings of the VLDB Endowment*, *8*(12), 1454-1465.
19. **PRESSMAN, R., & MAXIM, B., 2016.** *Engenharia de Software-8ª Edição*. McGraw Hill Brasil.
20. **PANG, D., IGASAKI, T., & MAEHARA, J. I., 2016**. *Long-term monitoring of heart rate variability toward practical use in intensive/high care unit. Paper presented at the 2016 9th Biomedical Engineering International Conference* (BMEiCON).
21. **RÍOS, J. M., MORA, N. L., ORDÓÑEZ, M. Z., & SOJOS, E. L., 2016**. Evaluación de los Frameworks en el Desarrollo de Aplicaciones Web con Python. *Revista Latinoamericana de Ingeniería de Software, 4*(4), 201-207.
22. **RASCHKA, S., 2015.** *Python machine learning*: Packt Publishing Ltd.
23. **RIGGS, S., CIOLLI, G., KROSING, H., & BARTOLINI, G. 2015**. *PostgreSQL 9 administration cookbook*. Packt Publishing Ltd.
24. **RUSSELL, S., BENNETT, T. D., & GHOSH, D. (2019).** *Software engineering principles to improve quality and performance of R software*. *PeerJ Computer Science*, *5*, e175
25. **SIEGELAUB, J. M., 2017**. How PRINCE2® Can Complement the PMBOK® Guide and Your PMP®.
26. **TLM, 2016**. *Xilema-Base-Web. Wiki TLM*. 2016. S.l.: s.n.
27. **UNHELKAR, B., AMBLER, S., ARMOUR, F., MILLER, G., BASILI, V., ... & VALETT, J. D., 2017.** Cooking Up Quality Software: Object-oriented Software Development Process. In *Software Engineering with UML* (Vol. 35, No. 1, pp. 1-18). New York: Prentice Hall PTR.

28. **VILLAFUERTE, JIHUALLANCA Y RODRIGO, EDWIN., 2017.** Sistema Help Desk para la gestión de la infraestructura tecnológica para la empresa Electro Puno SAA basado en ITIL V3. 2017.

Bibliografía

1. **ALIAGA, F. M., GUTIÉRREZ-BRAOJOS, C., & FERNÁNDEZ-CANO, A., 2018**. Las revistas de investigación en educación: Análisis DAFO. *ASOCIACIÓN INTERUNIVERSITARIA DE INVESTIGACIÓN PEDAGÓGICA (AIDIPE) MIEMBRO DE LA EUROPEAN EDUCATIONAL RESEARCH (EERA)*, *36*(2), 563.
2. **ANDERS, S., PYL, P. T., & HUBER, W., 2015**. *HTSeq—a Python framework to work with high-throughput sequencing data. Bioinformatics, 31*(2), 166-169.
3. **ÁVILA, A. E. S., UN, F., PEÑA, M. J. A. C., LUGO, A. J. D., & MARTÍNEZ, A. R., 2018.** Utilidad del Lenguaje Unificado de Modelado (UML) en el desarrollo de software profesional dentro del sector empresarial y educativo.
4. **ARIAS, Á., & DURANGO, A., 2016.** *Ingeniería y Arquitectura del Software: 2ª Edición*. IT *Campus Academy*.
5. **CALLEJA, GUILLERMO Y CUTULI, ROBERTO, 2017.** Despliegue de un Sistema de Almacenamiento Distribuido (Ceph) en el Centro de Datos Universitario. El caso de la Universidad Nacional de Cuyo. 2017.
6. **CASTILLO, A. A., 2017.** *Curso de Programación Web: JavaScript, Ajax y jQuery*: IT Campus Academy.
7. **CAMPOS, S. G., & MARTÍNEZ, L. F. F., 2015.** Programación Extrema: Prácticas, Aceptación y Controversia. *Cultura Científica y Tecnológica*, (15).
8. **CENTRO INTERNACIONAL DE LA HABANA, C. (2017**). Centro Internacional de La Habana, CIH. *from* http://www.cih.cu/es
9. **CIGET., 2014**. CIGET *from* http://www.cigetholguin.cu/
10. **CONSULTORES, D. C. (2019).** *Data Center Consultores. from* https://www.datacenterconsultores.com/
11. **DUARTE, L., TEODORO, A. C. M., & FREITAS, A., 2017**. *GIS Open Source Application as a Support to a Hospital Morbidity Database*.
12. **DÍAZ, A. E., 2016.** Sitio Web "Aprender Modelo Entidad-Relación". *Revista Conrado*, *12*(56).
13. **GROS, B., ESCOFET, A., & MARIMÓN, M., 2016**. Los patrones de diseño como herramientas para guiar la práctica del profesorado/*The design patterns as tools to guide the practice of teachers. Revista Latinoamericana de Tecnología Educativa-RELATEC*, *15*(3), 11-25.
14. **GONZÁLEZ, J. D. Y., CALVACHE, C. J. P., & GÓMEZ, O. S. G., 2015**. Revisión sistemática acerca de la implementación de metodologías ágiles y otros modelos

en micro, pequeñas y medianas empresas de software. *Revista Tecnológica-ESPOL*, *28*(5).

15. **GONZÁLEZ, J. P., DOMINGUEZ MAYO, F. J., GUTIÉRREZ RODRÍGUEZ, J. J., & ESCALONA CUARESMA, M. J., 2014.** Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental. *Ibersid*, *8*.
16. **GONZÁLEZ, H. D. L., 2016.** *Metodología de la investigación: propuesta, anteproyecto y proyecto*: Ecoe Ediciones.
17. **ISLAM, Q. N., 2015.** *Mastering PyCharm*. Packt Publishing Ltd
18. **JOSKOWICZ, J., 2008.** Reglas y prácticas en *eXtreme Programming*. *Universidad de Vigo*, *22*.
19. **KARIM, S., LIAWATIMENA, S., TRISETYARSO, A., ABBAS, B. S., & SUPARTA, W., 2017**. *Automating functional and structural software size measurement based on XML structure of UML sequence diagram*. In *2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)* (pp. 24-28). IEEE.
20. **KENNEDY, B., & MUSCIANO, C., 2017**. *HTML & XHTML. The Definitive Guide*.
21. **LUNA GARCÍA, H., MENDOZA GONZÁLEZ, R., & ÁLVAREZ RODRÍGUEZ, F. J., 2015.** Patrones de diseño para mejorar la accesibilidad y uso de aplicaciones sociales para adultos mayores.
22. **S.A.S, M., 2018**. MASTERTICS S.A.S. from https://www.mastertics.com/
23. **MEXICO, M. D., 2014.** MIKROINTERACCIONES DE MEXICO. *from* http://mikrointeracciones.com.mx/web/index.php?option=com_content&view=featured
24. **MOLINA, C.E., 2017**. Fundamentos de Redes: Topologías de Red. . S.l.:
25. **MACHADO, F. N. R., 2018.** *Análise e Gestão de Requisitos de Software–Onde nascem os sistemas*: Editora Saraiva.
26. **MONSALVE, C., ULLÓN, R., MAYA, R., & ROMERO, J., 2015.** Estudio del tamaño de los documentos de requerimientos de software como factor para la estimación del esfuerzo de inspección de requerimientos de software: Ingeniería de Software e Ingeniería del Conocimiento. Jornadas ….
27. **OH, G., KIM, S., LEE, S. W., & MOON, B., 2015.** *SQLite optimization with phase change memory for mobile applications. Proceedings of the VLDB Endowment*, *8*(12), 1454-1465.
28. **ORDOÑEZ, H., ESCOBAR, A., VELANDIA, D., & COBOS, C., 2015**. *Business Processes as a Strategy to Improve Requirements Elicitation in Extreme Programming (XP). Paper presented at the* Memorias del VII Congreso de Telematica CITA.

29. **PRESSMAN, R., & MAXIM, B., 2016.** *Engenharia de Software-8ª Edição*. McGraw Hill Brasil.
30. **PANG, D., IGASAKI, T., & MAEHARA, J. I., 2016**. *Long-term monitoring of heart rate variability toward practical use in intensive/high care unit. Paper presented at the 2016 9th Biomedical Engineering International Conference* (BMEiCON).
31. **PANTALEO, G., & RINAUDO, L., 2015**. *Ingeniería de software*: Alfaomega Grupo Editor.
32. **PINCIROLI, F., & ZELIGUETA, L., 2017**. *Modelado de negocios orientado a aspectos con AOP4ST. Paper presented at the XIX Workshop* de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).
33. **RÍOS, J. M., MORA, N. L., ORDÓÑEZ, M. Z., & SOJOS, E. L., 2016**. Evaluación de los Frameworks en el Desarrollo de Aplicaciones Web con Python. *Revista Latinoamericana de Ingeniería de Software, 4*(4), 201-207.
34. **RASCHKA, S., 2015.** *Python machine learning*: Packt Publishing Ltd.
35. **RUBIO, D. (2017).** *Beginning Django: Web Application Development and Deployment with Python*: Apress.
36. **RIGGS, S., CIOLLI, G., KROSING, H., & BARTOLINI, G. 2015**. *PostgreSQL 9 administration cookbook*. Packt Publishing Ltd.
37. **RUSSELL, S., BENNETT, T. D., & GHOSH, D. (2019).** *Software engineering principles to improve quality and performance of R software*. *PeerJ Computer Science*, *5*, e175
38. **SIEGELAUB, J. M., 2017**. How PRINCE2® Can Complement the PMBOK® Guide and Your PMP®.
39. **TLM, 2016**. *Xilema-Base-Web. Wiki TLM*. 2016. S.l.: s.n.
40. **UNHELKAR, B., AMBLER, S., ARMOUR, F., MILLER, G., BASILI, V., ... & VALETT, J. D., 2017.** Cooking Up Quality Software: Object-oriented Software Development Process. In *Software Engineering with UML* (Vol. 35, No. 1, pp. 1-18). New York: Prentice Hall PTR.
41. **VILLAFUERTE, JIHUALLANCA Y RODRIGO, EDWIN., 2017.** Sistema Help Desk para la gestión de la infraestructura tecnológica para la empresa Electro Puno SAA basado en ITIL V3. 2017.

Anexo <<1>> Topología de red

Definición La topología de red es la disposición física en la que se conecta una red de ordenadores. Si una red tiene diversas topologías se la llama mixta.

Definición de Nodo: El término nodo se refiere a un punto de intersección en el que confluyen dos o más elementos de una red de comunicaciones. De esta manera, si se refiere a una red de computadoras, cada una de las máquinas constituye un nodo. Y si a lo que se hace referencia es a una red de Internet, cada uno de los servidores también es considerado un nodo, y tienen un nombre propio de dominio y una dirección. También un nodo puede ser los *routers*, los *switchers*, etc. Para entenderlo mejor, no hay que pensar en un nodo como un elemento constituido solamente por una parte física, sino más bien considerarlo como una unidad funcional en donde tiene que haber tanto hardware como software. Por otra parte, al ser el punto de conexión de dos o más elementos, el nodo por lo general tiene la capacidad de recibir información, procesarla y enrutarla a otro u otros nodos. De esta manera, un nodo puede ser el punto de conexión para transmitir los datos, el punto desde el cual se redistribuye los datos hacia otros nodos y el punto final al que se transmiten los datos.

Topologías más comunes

Red en anillo: Topología de red en la que las estaciones se conectan formando un anillo. Cada estación está conectada a la siguiente y la última está conectada a la primera. Cada estación tiene un receptor y un transmisor que hace la función de repetidor, pasando la señal a la siguiente estación del anillo. En este tipo de red la comunicación se da por el paso de un token o testigo, que se puede conceptualizar como un cartero que pasa recogiendo y entregando paquetes de información, de esta manera se evita perdida de información debido a colisiones. Cabe mencionar que si algún nodo de la red se cae (termino informático para decir que está en mal funcionamiento o no funciona para nada) la comunicación en todo el anillo se pierde.

En un anillo doble, dos anillos permiten que los datos se envíen en ambas direcciones. Esta configuración crea redundancia (tolerancia a fallos). Evita las colisiones.

Red en árbol: Topología de red en la que los nodos están colocados en forma de árbol. Desde una visión topológica, la conexión en árbol es parecida a una serie de redes en estrella interconectadas. Es una variación de la red en bus, la falla de un nodo no implica interrupción en las comunicaciones. Se comparte el mismo canal de comunicaciones. Cuenta con un cable principal (*backbone*) al que hay conectadas redes individuales en bus.

Red en malla: La Red en malla es una topología de red en la que cada nodo está conectado a uno o más de los otros nodos. De esta manera es posible llevar los mensajes de un nodo a otro por diferentes caminos. Si la red de malla está completamente conectada no puede existir absolutamente ninguna interrupción en las comunicaciones. Cada servidor tiene sus propias conexiones con todos los demás servidores.

Red en bus: Topología de red en la que todas las estaciones están conectadas a un único canal de comunicaciones por medio de unidades interfaz y derivadores. Las estaciones utilizan este canal para comunicarse con el resto. La topología de bus tiene todos sus nodos conectados directamente a un enlace y no tiene ninguna otra conexión entre nodos. Físicamente cada *host* está conectado a un cable común, por lo que se pueden comunicar directamente, aunque la ruptura del cable hace que los hosts queden desconectados.

Red en estrella: Red en la cual las estaciones están conectadas directamente al servidor u ordenador y todas las comunicaciones se han de hacer necesariamente a través de él. Todas las estaciones están conectadas por separado a un centro de comunicaciones, concentrador o nodo central, pero no están conectadas entre sí. Esta red crea una mayor facilidad de supervisión y control de información ya que para pasar los mensajes deben pasar por el *hub* o concentrador, el cual gestiona la redistribución de la información a los demás nodos. La fiabilidad de este tipo de red es que el malfuncionamiento de un ordenador no afecta en nada a la red entera, puesto que cada ordenar se conecta independientemente del *hub*, el

costo del cableado puede llegar a ser muy alto. Su punto débil consta en el *hub* ya que es el que sostiene la red en uno.

Componentes básicos de una red

Los componentes básicos para poder montar una red local son:

Servidor: es una computadora utilizada para gestionar el sistema de archivos de la red, da servicio a las impresoras, controla las comunicaciones y realiza otras funciones. Puede ser dedicado o no dedicado. El sistema operativo de la red está cargado en el disco fijo del servidor, junto con las herramientas de administración del sistema y las utilidades del usuario. La tarea de un servidor dedicado es procesar las peticiones realizadas por la estación de trabajo. Estas peticiones pueden ser de acceso a disco, a colas de impresión o de comunicaciones con otros dispositivos. La recepción, gestión y realización de estas peticiones puede requerir un tiempo considerable, que se incrementa de forma paralela al número de estaciones de trabajo activas en la red. Como el servidor gestiona las peticiones de todas las estaciones de trabajo, su carga puede ser muy pesada. Se puede entonces llegar a una congestión, el tráfico puede ser tan elevado que podría impedir la recepción de algunas peticiones enviadas. Cuanto mayor es la red, resulta más importante tener un servidor con elevadas prestaciones. Se necesitan grandes cantidades de memoria RAM (Memoria de Acceso Aleatorio) para optimizar los accesos a disco y mantener las colas de impresión. El rendimiento de un procesador es una combinación de varios factores, incluyendo el tipo de procesador, la velocidad, el factor de estados de espera, el tamaño del canal, el tamaño del bus, la memoria caché, así como de otros factores.

Estaciones de Trabajo: se pueden conectar a través de la placa de conexión de red y el cableado correspondiente. Los terminales “tontos” utilizados con las grandes computadoras y mini computadoras son también utilizadas en las redes, y no poseen capacidad propia de procesamiento. Sin embargo, las estaciones de trabajo son, generalmente, sistemas inteligentes. Los terminales inteligentes son los que se encargan de sus propias tareas de procesamiento, así que cuanto mayor y más rápido sea el equipo,

mejor. Los terminales tontos en cambio, utilizan el espacio de almacenamiento, así como los recursos disponibles en el servidor.

Tarjetas de Conexión de Red (Interface Cards): permiten conectar el cableado entre servidores y estaciones de trabajo. En la actualidad existen numerosos tipos de placas que soportan distintos tipos de cables y topologías de red. Las placas contienen los protocolos y órdenes necesarios para soportar el tipo de red al que está destinada. Muchas tienen memoria adicional para almacenar temporalmente los paquetes de datos enviados y recibidos, mejorando el rendimiento de la red. La compatibilidad a nivel físico y lógico se convierte en una cuestión relevante cuando se considera el uso de cualquier placa de red. Hay que asegurarse que la placa pueda funcionar en la estación deseada, y de que existen programas controladores que permitan al sistema operativo enlazarlo con sus protocolos y características a nivel físico.

Cableado: una vez que tenemos las estaciones de trabajo, el servidor y las placas de red, requerimos interconectar todo el conjunto. El tipo de cable utilizado depende de muchos factores, que se mencionarán a continuación: Los tipos de cableado de red más populares son: par trenzado, cable coaxial y fibra óptica. Además, se pueden realizar conexiones a través de radio o microondas. Cada tipo de cable o método tiene sus ventajas. y desventajas. Algunos son propensos a interferencias, mientras otros no pueden usarse por razones de seguridad. La velocidad y longitud del tendido son otros factores a tener en cuenta el tipo de cable a utilizar.

Par Trenzado: consiste en dos hilos de cobre trenzado, aislados de forma independiente y trenzados entre sí. El par está cubierto por una capa aislante externa. Entre sus principales ventajas tenemos:

- Es una tecnología bien estudiada
- No requiere una habilidad especial para instalación
- La instalación es rápida y fácil

- La emisión de señales al exterior es mínima.
- Ofrece alguna inmunidad frente a interferencias, modulación cruzada y corrosión.

Cable Coaxial: se compone de un hilo conductor de cobre envuelto por una malla trenzada plana que hace las funciones de tierra. Entre el hilo conductor y la malla hay una capa gruesa de material aislante, y todo el conjunto está protegido por una cobertura externa. El cable está disponible en dos espesores: grueso y fino. El cable grueso soporta largas distancias, pero es más caro. El cable fino puede ser más práctico para conectar puntos cercanos.

El cable coaxial ofrece las siguientes ventajas:

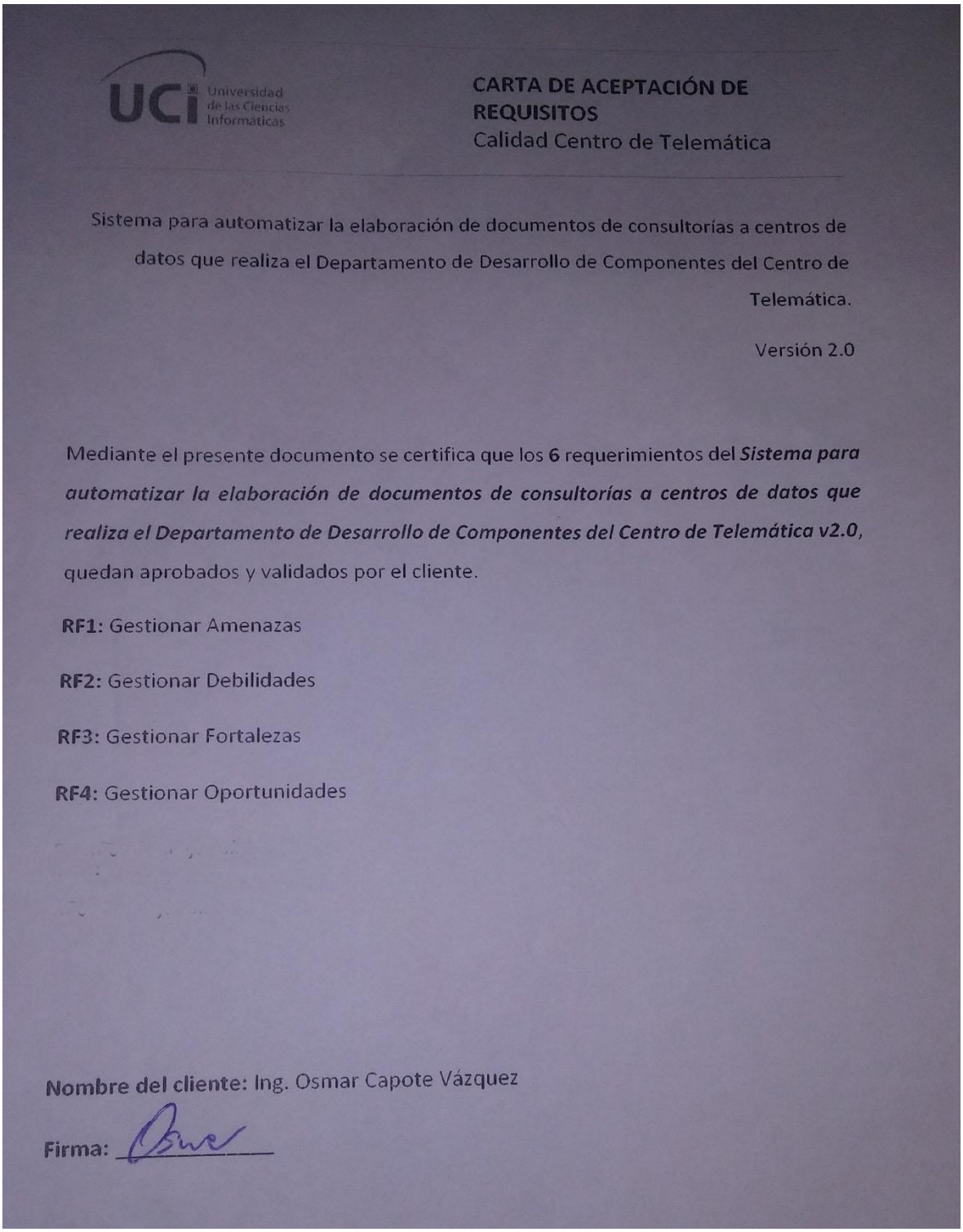
- Soporta comunicaciones en banda ancha y en banda base (Banda base es la señal de una sola transmisión en un canal, banda ancha significa que lleva más de una señal y cada una de ellas se transmite en diferentes canales, hasta su número máximo de canal).
- Es útil para varias señales, incluyendo voz, video y datos.
- Es una tecnología bien estudiada.

Conexión fibra óptica: esta conexión es cara, pero permite transmitir la información a gran velocidad e impide la intervención de las líneas. Como la señal es transmitida a través de luz, existen muy pocas posibilidades de interferencias eléctricas o emisión de señal. El cable consta de dos núcleos ópticos, uno interno y otro externo, que refractan la luz de forma distinta. La fibra está encapsulada en un cable protector.

Ofrece las siguientes ventajas:

- Alta velocidad de transmisión
- No emite señales eléctricas o magnéticas, lo cual redunda en la seguridad
- Inmunidad frente a interferencias y modulación cruzada.
- Mayor economía que el cable coaxial en algunas instalaciones.

Anexo <<2>> Acta de aceptación de requisitos



UCI Universidad de las Ciencias Informáticas

CARTA DE ACEPTACIÓN DE REQUISITOS

Calidad Centro de Telemática

Sistema para automatizar la elaboración de documentos de consultorías a centros de datos que realiza el Departamento de Desarrollo de Componentes del Centro de Telemática.

Versión 2.0

Mediante el presente documento se certifica que los **6** requerimientos del ***Sistema para automatizar la elaboración de documentos de consultorías a centros de datos que realiza el Departamento de Desarrollo de Componentes del Centro de Telemática v2.0***, quedan aprobados y validados por el cliente.

RF1: Gestionar Amenazas

RF2: Gestionar Debilidades

RF3: Gestionar Fortalezas

RF4: Gestionar Oportunidades

Nombre del cliente: Ing. Osmar Capote Vázquez

Firma:

Figura 21 Acta de aceptación de requisitos funcionales