Universidad de las Ciencias Informáticas

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

Título:

Directorio v2.0 de la Universidad de las Ciencias Informáticas

Autores:

- Luis A. Justiz Munder
- José C. Peñalver Peñalver

Tutores:

- Ing. Jany Coto García
- Ing. Jorge J. Pérez Hechavarría

Co-tutora:

- Ing. Arlennys S. Velázquez
- Ing. Yoandrys Pacheco Geréz

La Habana, Cuba. 2016
"Año 58 de la Revolución"

"Los proveedores de software están intentando hacer sus productos más amigables para el usuario. Su mejor aproximación hasta el momento ha sido tomar sus antiguos folletos y estampar las palabras 'amigable para el usuario' en la portada"

Bill Gates

DECLARACIÓN DE AUTORÍA

Declaramos ser autores del presente trabajo de diploma y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____

José C. Peñalver Peñalver
Autor

_____

Luis A. Justiz Munder
Autor

_____

Ing. Jany Coto García
Tutora

_____

Ing. Jorge J. Pérez Hechavarría
Tutor

Resumen

La presente investigación describe la realización de la versión 2.0 del Directorio para la Universidad de las Ciencias Informáticas, desarrollado en la Dirección de Informatización. Este sistema permite a la comunidad realizar búsquedas de personas, teléfonos, cumpleaños y aplicaciones de la Universidad. Además, contribuye a corregir deficiencias presentadas por la solución que lo antecede, dentro de la que se destaca la actualización de las tecnologías de desarrollo. Se incorporan nuevas funcionalidades como: la creación de listas de personas y teléfonos, un mecanismo de actualización de datos, un *API REST* que brinda la información para que otras aplicaciones puedan consumir los datos de las personas y el diseño web *responsive* o adaptativo para los dispositivos, desde ordenadores de escritorio a *tablets* y móviles. El proceso de desarrollo de software estuvo guiado por la metodología Desarrollo Ágil con Calidad que combina las buenas prácticas del nivel 2 del Modelo de Madurez de la Capacidad Integrado con las de dirección y desarrollo ágil de proyectos de software. Para la implementación del sistema se utilizó *Symfony* como marco de trabajo, *PostgreSQL* como sistema gestor de bases de datos y *NetBeans* como entorno integrado de desarrollo.

Palabras clave: adaptativo, *API REST*, búsquedas, directorio, mecanismo de actualización, *responsive*.

Índice de Contenido

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

El mundo de hoy presenta una explosión de nuevos métodos y herramientas, es un mundo comunicado, en evolución, pero en medio de todo el compendio de novedades que lo rodea y que a veces lo sumerge en el asombro de adelantos importantes, la trama es siempre la misma, las necesidades son similares y el verdadero éxito está en aplicar los conocimientos que la humanidad ha logrado compilar a través de los años en el uso de nuevas herramientas que los tiempos actuales ponen a disposición (1).

En estos vertiginosos y significativos progresos, las Tecnologías de la Información y la Comunicación (TIC) juegan un papel decisivo, lo que posibilita alcanzar una visión integral y globalizante, pues están presentes en todas las actividades y procesos que se realizan en las organizaciones, como herramienta necesaria en la gestión y búsqueda de información y como garantía para alcanzar la eficiencia, eficacia y efectividad. A medida que la información es más creciente e ilimitada, resulta necesario para las instituciones desarrollar mecanismos que permitan localizarla, manipularla y tratarla. Para ello el uso de las TIC ha permitido el desarrollo de sistemas automatizados con búsquedas especializadas que permiten encontrar datos específicos de un tema determinado a gran velocidad. Búsquedas que se pueden iniciar por medio de palabras clave que identifican de una forma u otra el tema de interés de las personas. Dentro de las vías que se pueden utilizar para realizar este proceso se encuentran los directorios.

Los directorios son "guías" que permiten localizar personas, organismos y entidades públicas o privadas; se puede definir como una lista de personas o de organizaciones ordenadas alfabética o sistemáticamente, proporcionando direcciones, cargos, funciones y datos similares, de tal manera que permite al usuario localizar, identificar y obtener informaciones acerca de las personas u organismos existentes en una industria, mercado, grupo, sector de actividad o disciplina determinada (2). El Directorio es un sistema que consulta y organiza ficheros a partir de una búsqueda determinada, que pueden estar organizados jerárquicamente, es decir, puede mostrar la información de distintas formas y maneras posibles en dependencia de lo descrito por el usuario.

En Cuba, son numerosas las instituciones que implementan su uso como una estrategia eficiente en el intercambio de información y localización del personal que las conforman, pues su función primordial es brindar información a los usuarios para que estos puedan consultar y nutrirse de los datos específicos a partir de sus necesidades.

Varios son los ejemplos donde se evidencian este tipo de sistemas de búsqueda, uno de ellos es la Guía Telefónica, la cual llega de forma anual y en formato impreso a los usuarios que tienen contrato de telefonía

fija con ETECSA (Empresa de Telecomunicaciones de Cuba S. A.). También cuenta con un sistema electrónico, llamado Páginas Amarillas que muestra información comercial de entidades del país. Otro ejemplo se evidencia en el Directorio Turístico que brinda información de los lugares más visitados dentro del país. Existen otras organizaciones como las universidades y centros de estudios que utilizan sistemas informáticos similares para consultar datos específicos de toda su comunidad.

La Universidad de las Ciencias Informáticas (UCI), es una de las instituciones que no ha quedado exenta de estos avances tecnológicos y junto a ellos persigue la estrategia de informatizar los procesos que se llevan a cabo en la misma. Para ello ha implementado diferentes servicios como: el Sistema de Gestión Universitaria, el portal Intranet, el periódico Mella, el Directorio de personas, entre otros, que sirven de fuente de información para que la comunidad universitaria pueda nutrirse de ellos.

Según una encuesta realizada a una muestra de 155 personas de la Universidad (ver Anexo1) y una entrevista al Director de Redes y Servicios Telemáticos (ver Anexo 2), se obtuvo como resultado que uno de los servicios más solicitados en la institución es el Directorio UCI. Sin embargo, en ocasiones no satisface las necesidades del cliente, pues carece de diferentes funciones que son de interés para el usuario. Por ejemplo: no cuenta con notificaciones de cumpleaños a través del correo electrónico, no presenta la dirección de algunos lugares dentro de la Universidad, así como su geolocalización. No describe las aplicaciones web que contiene dentro y no presenta un diseño adaptable para los dispositivos móviles, lo que provoca que el usuario no se sienta complacido.

Además, carece de integración con los servicios del portal de la Intranet y el Sistema de Información Geográfica de la Universidad (SIGUCI). Solo visualiza el perfil[1] principal de una persona, aunque presente varios perfiles simultáneamente. Cuenta con dos esquemas[2] de base de datos que funcionan como pasarela[3] y mantenerlas actualizadas se convierte en un proceso engorroso para el equipo de trabajo. Actualmente se encuentra desarrollado sobre el CMS *Drupal*[4]*,* y por las constantes variaciones en sus versiones no se encuentra soportada en la última versión estable que presenta dicha tecnología de desarrollo.

[1] Rol de una persona dentro de la Universidad.

[2] Contenedor de tablas, vistas, procedimientos, funciones, entre otros que se acomodan como cajas anidadas.

[3] Instancia por donde la información pasa entre dos entidades distintas.

[4] Sistema de gestión de contenidos que se utiliza para crear sitios web dinámicos.

Por ser un producto de alta demanda por la comunidad universitaria, resultaría beneficioso desarrollar un sistema que cubra todas las necesidades referentes a la divulgación de la información en la Universidad, que mejore sus servicios y que permita la correcta gestión de la información que maneja a partir de las herramientas que brinda la tecnología en virtud del desarrollo.

A partir de la problemática antes planteada el **problema de investigación** es el siguiente: ¿Cómo mejorar el directorio UCI en el proceso de búsqueda de información?

Para dar solución al problema anteriormente expuesto se enmarca el **objeto de estudio** en el proceso de búsqueda de información, enfocando el **campo de acción** en el proceso de búsqueda de información básica de los usuarios, aplicaciones, teléfonos y cumpleaños de la UCI.

Como **objetivo general** se plantea desarrollar el sistema Directorio v 2.0 de la Universidad de las Ciencias Informáticas para mejorar los servicios actuales utilizando el marco de trabajo Symfony 2. Se definen los siguientes **objetivos específicos**:

- Caracterizar el marco teórico conceptual de la investigación.
- Analizar la implementación del sistema que se encuentra en explotación.
- Diseñar la propuesta de solución a partir del proceso de desarrollo de software utilizado.
- Desarrollar la versión 2.0 del Directorio UCI.
- Validar mediante pruebas funcionales los resultados obtenidos con la solución.

Se propone como **idea a defender**: con el desarrollo de una nueva versión del Sistema Directorio UCI, se mejora el proceso de búsqueda de información básica de cada usuario y de la institución.

Para dar cumplimiento a los objetivos específicos se proponen como **tareas de investigación:**

- Realización de encuestas a los usuarios sobre el uso de la versión actual del sistema en aras de obtener un resultado que sirva de punto de partida en la investigación.
- Caracterización del estado del arte de sistemas y tendencias existentes relacionadas con los directorios y sistemas de búsqueda, para lograr un mejor entendimiento del objeto de estudio y el campo de acción.
- Estudio del proceso de negocio del sistema actual que permita comprender la problemática planteada.
- Revisión del diseño de los componentes visuales, arquitectura, servidores o fuentes de datos y algoritmos de búsqueda utilizados que aporten nuevas ideas al desarrollo de la solución.
- Identificación de los requerimientos necesarios para elaborar la propuesta de solución.

- Modelado de la base de datos para representar y describir la estructura de datos y la relación entre ellos.
- Migración de las funcionalidades de la versión 1.0.
- Implementación de las nuevas funcionalidades descritas que garanticen el cumplimiento del objetivo planteado en la investigación.
- Implementación de un mecanismo de actualización de datos.
- Construcción del *API REST* para brindar datos de personas.
- Realización de las pruebas de *software* para garantizar el correcto funcionamiento de la solución.

Durante el desarrollo de la investigación se emplean los siguientes **métodos científicos:**

Métodos empíricos

- **Encuesta:** se realiza con el objetivo de recolectar información relacionada con el Directorio UCI y así obtener la opinión de los usuarios referente a la necesidad de contar con un sistema que responda a sus necesidades como cliente.

  La población seleccionada para realizar la encuesta está compuesta por 155 personas, de ellas 103 estudiantes, 25 profesores y el resto especialistas de la producción, directivos, entre otros. Para su aplicación se utiliza el método Aleatorio Simple, técnica de muestreo en la que todos los elementos que forman el universo y que, por lo tanto, están descritos en el marco muestral, tienen idéntica probabilidad de ser seleccionados para la muestra. En la encuesta se realizaron 6 preguntas con varios incisos donde los interesados respondieron de acuerdo a sus necesidades y prioridades (Anexo 1 y 3).

Métodos teóricos

- **Analítico-Sintético:** permite realizar un análisis de los elementos relacionados con los sistemas de búsqueda y directorios para posteriormente sintetizar dicho estudio y definir los diferentes conceptos que se manejan en la investigación. Así como un estudio de las herramientas y tecnologías a utilizar durante el desarrollo de las mismas.
- **Modelación:** se aplica para representar por medio de modelos, diagramas y mapas el proceso y funcionamiento de sistemas de búsqueda y directorios, para tener una mejor comprensión del objeto de estudio.

Luego de terminada la investigación y comenzar a utilizar los resultados de la misma, se obtendrá como beneficio la mejora del proceso de búsqueda de información de personas, cumpleaños, teléfonos y aplicaciones de la UCI que facilite la comunicación y retroalimentación de toda la comunidad universitaria.

El presente trabajo investigativo se estructura en tres capítulos:

Capítulo 1. Fundamentación teórica: incluye un estudio del estado del arte donde se muestran los principales conceptos tratados en la investigación sobre los sistemas de búsquedas y directorios. Se realiza un análisis de diferentes directorios y se hace una breve referencia a las herramientas, tendencias, técnicas, metodologías y *software* usados en el mundo para dar solución a problemas similares que apoyen a la construcción de la solución.

Capítulo 2. Presentación de la solución propuesta: se realiza un análisis de la propuesta de solución que permita construir el modelo conceptual y definir mediante las técnicas de obtención de requisitos, los requisitos funcionales y no funcionales. Además, de describir la arquitectura, patrones de diseño y de base de datos que se utilizan en la investigación y así poder elaborar el diagrama de despliegue y el modelo de base de datos.

Capítulo 3. Construcción y validación de la propuesta de solución: se describe la implementación de la solución a través de los estándares de codificación, se validan los requisitos funcionales atendiendo a diferentes técnicas estudiadas. Se elaboran y realizan las pruebas al software a través de las estrategias de pruebas seleccionadas y se representan los resultados de las mismas.
El documento incluye las conclusiones generales, bibliografías referenciadas y consultadas, así como los anexos que sirven de apoyo a la investigación.

Capítulo1: Fundamentación teórica

Los directorios son herramientas de apoyo que permiten realizar búsquedas de información en dependencia de los criterios especificados por el usuario. En el presente capítulo se realiza un estudio del estado del arte donde se muestran los principales conceptos y aspectos tratados en la investigación. Se valoran los sistemas existentes relacionados al dominio del problema y se realiza una breve descripción de la metodología de desarrollo y tecnologías a utilizar definidas por la Dirección de Informatización (DIN).

1.1 Conceptos y aspectos asociados al dominio del problema

Para una mejor comprensión de la investigación se exponen un conjunto de aspectos importantes que están estrechamente relacionados al dominio del problema.

1.1.1 Sistema de información

Los sistemas de información son un conjunto de elementos utilizados para la administración de datos, que se encuentran coordinados entre sí para su uso posterior (3). En el sitio web EcuRed lo detallan como: un conjunto de elementos interrelacionados con el propósito de prestar atención a las demandas de información de una organización, para elevar el nivel de conocimientos que permitan un mejor apoyo a la toma de decisiones y desarrollo de acciones (4).

Otros autores como Peralta (2008), de una manera más acertada define sistema de información como: conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. Teniendo en cuenta el equipo computacional necesario para que el sistema de información pueda operar y el recurso humano que interactúa con el Sistema de Información (5).

Un sistema de información realiza cuatro actividades básicas (Figura 1):

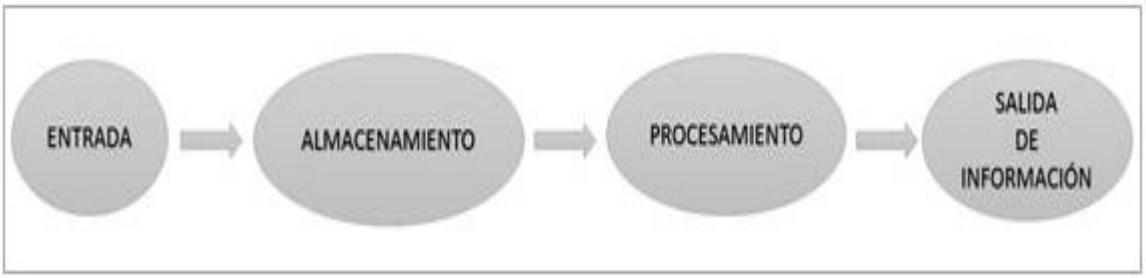


Figura 1: Actividades de un sistema de información

1.1.2 Sistemas de búsqueda de información

Los sistemas de búsqueda de información son sistemas automáticos de recuperación de información que almacenan datos sobre páginas web en una base de datos. Son un conjunto de elementos interrelacionados destinados a la captura, depuración, almacenamiento, recuperación, actualización y tratamiento de dato (6); por lo que constituyen un eslabón importante en la devolución de resultados asociados a la información que desee el usuario.

Aunque buscar información en la red no sea fácil, existen herramientas de búsqueda que ayudan a encontrar lo que se desea, haciendo coincidir las palabras clave que el interesado indica en la búsqueda con la información más relevante que existe en la red.

1.1.3 Tipos de sistemas de búsquedas

Dentro de las herramientas que se utilizan para la búsqueda de información, se encuentran los **Motores de búsqueda o metabuscadores** que son sistemas de búsqueda por palabras clave, consistentes en bases de datos, incorporan automáticamente páginas web mediante "*robots*"[5] de búsqueda por la red.

Dichos "*robots*" o "arañas", recorren las páginas recopilando información sobre los contenidos de los sitios de Internet en bases de datos. Estas bases de datos contienen, generalmente, el título de la página, una descripción del sitio, palabras clave e información de sus enlaces (7).

Otros son los **Índices temáticos o directorios** que son sistemas de búsqueda por temas o categorías jerarquizados, aunque también incluyen sistemas de búsqueda por palabras clave. Se trata de bases de datos de direcciones web elaboradas "manualmente", es decir, hay personas que se encargan de asignar cada página web a una categoría o tema determinado (7).

Los directorios representan contenedores de información descriptiva basada en atributos, apoyado en sofisticadas capacidades de filtrado. Por lo general no admiten transacciones complicadas como las que se encuentran en los sistemas de bases de datos diseñados para manejar grandes y complejos volúmenes de datos. Las actualizaciones de los directorios son normalmente cambios simples, o todo o nada, siempre y cuando estén permitidos (8). Es decir, es una "Base de datos" especializada específicamente diseñada para la búsqueda de información.

[5] Programas que buscan continuamente por todos los servidores de Internet construyendo un índice de lo hallado. También conocidos como «arañas» por su continuo desplazamiento sobre la red.

A continuación, se muestra una tabla con un resumen de las características de los Motores de búsqueda y Directorios (ver Tabla 1).

Tabla 1: Características de los Motores de búsqueda y Directorios

Características / Buscadores	Motores de búsqueda	Directorios
Actualización de la información	Automática por la red	De forma manual por el administrador
Información almacenada en la página	Toda	Solo los campos más relevantes como son: título, palabras clave, entre otros
Almacenamiento de la información	Una base de datos propia	Directorios, clasificados en categorías
Realización de las búsquedas	Mediante la ecuación de búsqueda	Jerárquicamente según las categorías establecidas
Presentación de los resultados	Orden de relevancia según criterios establecidos en la ecuación de búsqueda	Listado de todos los documentos correspondientes en la categoría, sin ningún criterio de presentación
Tipo de información a localizar	Específicas	Temas generales

1.1.4 Base de datos

Una base de datos (cuya abreviatura es *BD*) es una entidad en la cual se pueden almacenar datos de manera estructurada, con la menor redundancia posible. Diferentes programas y usuarios deben poder utilizar estos datos. Por lo tanto, el concepto de base de datos generalmente está relacionado con el de red ya que se debe poder compartir esta información (9). Para otros autores es un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos (10); capaces de coleccionar, recopilar, estructurar los datos; además de permitir operar información de manera organizada, segura, confiable y en grandes cantidades.

1.2 Los Directorios y las Bases de datos

Generalmente se describe un directorio como una base de datos especializada cuyas características la apartan de una base de datos relacional de propósito general. Una de estas características especiales es que son accedidas (*búsqueda o lectura*) mucho más que actualizadas (*escritura*). Muchos usuarios pueden estar consultando el número de teléfono de una persona, o buscando una estación de trabajo con un programa concreto, pero generalmente tanto el número de teléfono de la persona, como los programas instalados en una estación no cambian con excesiva frecuencia. Por ello:

- Los directorios están optimizados para accesos en lectura, frente a las bases de datos convencionales, que se encuentran optimizadas para lectura y escritura.
- Los directorios están optimizados para almacenar información relativamente estática, por lo que no son recomendables para almacenar datos que cambian con frecuencia como por ejemplo la carga de una estación de trabajo.
- Los directorios no soportan transacciones. Las transacciones son operaciones de base de datos que permiten controlar la ejecución de una operación compleja, de modo que dicha operación se completa totalmente o no se ejecuta en absoluto.
- El tipo de información que se almacena generalmente en el directorio no requiere una consistencia estricta y se considera aceptable que el número de teléfono de una persona no esté actualizado de forma temporal.
- La mayoría de las bases de datos convencionales utilizan el Lenguaje de Consulta Estructurado (SQL por sus siglas en inglés)[6], que permite el desarrollo de funciones de consulta y actualización muy complejas, a costa del tamaño y complejidad de la aplicación.
- Los directorios de Protocolo Ligero de Acceso a Directorios (LDAP[7] por sus siglas en inglés) utilizan un servicio de directorio de protocolo simplificado y optimizado que puede ser utilizado para la construcción de aplicaciones simples y pequeñas (11).

1.3 Sistemas de directorios internacionales

En el mundo se han desarrollado diferentes aplicaciones que gestionan de manera automatizada la información que se genera. En la investigación se realiza un análisis de estas soluciones con el fin de obtener una aproximación de las funcionalidades que debe tener la solución. A continuación, se detallan sus principales características y se hace una valoración acerca de los resultados obtenidos:

1.3.1 Directorio de Internet de Oracle

El Directorio de Internet de Oracle (OID por sus siglas en inglés) se presenta como un directorio abierto, enfocado a simplificar la administración de usuarios y aplicaciones, así como a ofrecer a los usuarios finales un punto único de entrada a los recursos de la red, la información y las aplicaciones. Proporciona una

[6] Lenguaje declarativo de acceso a bases de datos relacionales que especifica diversos tipos de operaciones en ellas.

[7] Protocolo estándar para el acceso a directorio.

estructura de directorios escalable y de plataformas cruzadas que aporta fiabilidad y seguridad en las operaciones informáticas vía Internet. En cuanto a las capacidades de seguridad, potencia la administración de los directorios al integrar los certificados de claves públicas de los clientes, las agendas de correo electrónico y los privilegios de acceso, permitiendo, del mismo modo, mantener políticas de enrutado[8], objetos de administración de sistemas y servicios de calidad (12).

1.3.2 Directorio de la Universidad de Navarra

El Directorio de la Universidad de Navarra cuenta con un buscador básico, una búsqueda avanzada y un buscador de personas, este último permite buscar a: profesionales y alumnos de la institución. En su portada presenta un abecedario de la A-Z que, en dependencia del criterio de búsqueda especificado, mostrará un listado con la información relacionada. Además, muestra los centros asociados a la institución con su número de teléfono, *fax*[9] y correo electrónico. También presenta una versión en inglés y un vínculo[10] al sitio oficial de la universidad (13).

1.3.3 Directorio de la Universidad de Harvard

El Directorio de la Universidad de Harvard es un módulo integrado al sistema de dicha universidad, lo que implica que sea totalmente dependiente de este, recoge información referente a estudiantes, profesores y trabajadores, que son los que conforman el personal del centro. La búsqueda puede ser realizada por nombre, teléfono, dirección, departamento, disciplina académica, responsabilidad de trabajo, universitarios oficiales y decanos (14).

1.3.4 Directorio de la Universidad de Stanford

Es un módulo contenido dentro del sitio de la Universidad que depende de este sistema para su funcionamiento, no tiene una vía de acceso independiente, brinda el servicio de buscar personas u organizaciones en la web, la búsqueda de una organización puede ser realizada a través del nombre de la organización o el código de la misma; para las personas los datos pueden ser el nombre, correo electrónico, teléfono del trabajo o identificación (15).

[8] Selección del camino en una red de computadoras por donde se envían datos.

[9] Sistema de comunicación que permite mandar y recibir información gráfica a través de la línea telefónica.

[10] Apuntadores que sirven para saltar de una información a otra, o de un servidor web a otro, al navegar por la red.

1.3.5 Guía Telefónica de la Universidad de Costa Rica

Forma parte del sitio oficial de la universidad, dependiendo totalmente de este. Una vez seleccionado, muestra la guía telefónica en su página principal. Donde de forma estática, muestra los números de teléfonos de las distintas áreas con las que cuenta la universidad, ordenados alfabéticamente (16).

1.4 Sistemas de directorios nacionales

Cuba no se ha quedado al margen de los adelantos tecnológicos y emplea estos servicios para el disfrute de la sociedad, dependiendo de las necesidades de cada institución. A continuación, se describen varios sistemas de directorios existentes en el país.

1.4.1 Directorio Telefónico de ETECSA:

El Directorio Telefónico de ETECSA se ha convertido en un soporte informativo de referencia para la sociedad y una útil herramienta en la gestión comercial y la vida cotidiana de todos los cubanos. Su estructura interna se presenta en diferentes secciones:

Páginas Informativas: se ubican al inicio de cada Directorio Telefónico y cuentan con un índice de contenidos, en el se puede encontrar información sobre los principales servicios que brinda ETECSA.

Páginas Blancas: reportan en orden alfabético la razón social, nombres y apellidos, dirección y números telefónicos de los abonados residenciales y entidades de cada localidad.

Páginas Azules**:** reagrupa por orden alfabético la razón social, dirección y números telefónicos correspondientes a los Organismos de la Administración Central del Estado y otras Entidades Nacionales, así como su representación en las provincias contenidas en los volúmenes de cada zona.

Páginas Verdes**:** destinadas a las informaciones de entidades del Estado, en ella se publican los trámites que debe realizar la población.

Páginas Amarillas**:** aparecen clasificadas por categorías todas aquellas entidades que ubican su información comercial bajo un rubro o categoría preferencial. Se encuentra organizada por categorías de actividades relacionadas con la producción y los servicios, subcategorías, provincias (en orden geográfico) y alfabéticamente (17).

1.4.2 Directorio Telefónico del Instituto Politécnico José A. Echevarría:

El Directorio Telefónico del Instituto politécnico José A. Echevarría es una solución desarrollada por la propia institución. A través de él se puede consultar la información de los estudiantes y trabajadores de la institución. Además, muestra una guía de teléfonos en forma de tabla con los datos: extensión, local,

contacto y la ubicación por piso de la cada facultad; posibilitando al usuario imprimir, enviar por correo o escribir un comentario sobre la información mostrada (18).

1.4.3 Directorios de la Universidad de las Ciencias Informáticas:

La Universidad de las Ciencias Informáticas, como parte de la comunicación institucional cuenta con un portal de cara a internet que permite mantener informado del acontecer en la Universidad y de las principales actividades que allí se desarrollan dentro del área de formación, producción, investigación y extensionista. Por otra parte, está el Portal de la Intranet Universitaria, donde se muestran los diferentes acontecimientos e informaciones de interés para la comunidad.

El directorio del Portal de la Universidad solo muestra la información de los principales directivos y algunas de las personas involucradas en el proceso de investigación. Este agrupa las búsquedas por tres criterios: personas, universidades y teléfonos; a continuación, se describen brevemente sus funcionalidades.

El Directorio de Personas: muestra los datos generales de los principales directivos de la Universidad y las personas que participan activamente en el proceso de investigación. Permite buscar por categoría docente (*Instructor recién graduado, Instructor, Asistente, Auxiliar y Titular*) y por categoría científica (*Máster y Doctor*).

El Directorio de universidades cubanas: muestra un listado de universidades y centros universitarios que existen en el país. Este permite buscar por provincias.

El Directorio Telefónico: muestra los teléfonos de las áreas de la Universidad y de sus principales directivos. Permite buscar por el nombre del área (19).

Por su parte, el Portal de la Intranet Universitaria cuenta con un enlace al Directorio UCI, sistema que posibilita la búsqueda de todas las personas que estudian o laboran en la institución. El mismo permite realizar búsquedas por diferentes criterios dentro de lo que se incluyen:

Directorio de personas: permite realizar búsquedas a través de criterios, tales como: *Nombre, Apellidos, Solapín, No. Expediente y Usuario.* Además de apoyarse en diversos filtros de búsquedas como*: Categoría, Cargo, Área, Provincia, Municipio, entre otros.*

El Directorio Telefónico: permite buscar por los criterios: *Teléfono, Área, Ubicación y Nombre de pizarra.* Además de contar con las secciones de *Páginas amarillas* y *Urgencias.*

El Directorio de Aplicaciones UCI: muestra un listado con todas las aplicaciones por área de procesos de la universidad, con el Nombre, URL[11] y la Visibilidad del sitio ya sea institucional, nacional o internacional.
Los Cumpleaños: muestra un abecedario de la A-Z y el calendario del año actual, para apoyar la búsqueda del usuario (20).

1.5 Resultado del análisis de las soluciones existentes

Después de analizar varios sistemas de búsquedas de información se ha concluido que, a pesar de aportar amplios conocimientos acerca de las características y tipos de búsquedas propias, varias de estas soluciones evidencian particularidades que no son requeridas, pues no son independientes, lo que impide su utilización como componente de la propuesta de solución. Algunas no presentan buscadores, es decir, la información que muestran no es dinámica, lo que imposibilita al usuario realizar búsquedas más específicas de lo que desea.

Aunque es importante destacar que, dentro de estas soluciones, el Directorio UCI aporta aspectos importantes para el desarrollo de la investigación, tales como: la búsqueda por palabras clave, la forma de mostrar los resultados de las búsquedas en dependencia de la categoría de la persona (*estudiantes, eventuales, familiar, servicios, tercerizados*[12]*, trabajador y visitante*) y por ser un sistema totalmente independiente.

Este sistema también presenta limitaciones que hacen que las personas no se sientan totalmente complacidas con los servicios que brinda, pues no permite crear al usuario sus propias listas de personas, teléfonos y cumpleaños. No cuenta con notificaciones de cumpleaños a través de correo electrónico, no presenta la dirección de algunos lugares dentro de la Universidad, ni la geolocalización de los mismos. Solo visualiza el perfil principal de la persona lo que imposibilita conocer si la misma presenta otros cargos dentro del centro. No cuenta con un diseño adaptable para los dispositivos móviles, problema que para los usuarios no sería beneficioso dado los avances tecnológicos en la actualidad. Estas limitaciones fueron conocidas a través de una encuesta realizada a una muestra de personas en la Universidad (Anexo 4).

Es importante señalar que para el equipo de trabajo que actualmente es el encargado del sistema, existen varios inconvenientes que provocan que, a la hora de dar soporte a la aplicación, se retrase el proceso de actualización de los datos, pues las bases de datos con las que hoy cuenta, no presentan un mecanismo

[11] Localizador Uniforme de Recursos es una dirección que se asigna a cada uno de los recursos disponibles en la red para que estos puedan ser localizados o identificados.

[12] Personal perteneciente a una empresa externa que brinda servicios a la Universidad.

que permita que se realice de forma rápida y sencilla para el equipo de trabajo. Esta solución también carece de integración con otros servicios que brinda la institución como la Intranet y SIGUCI. Estas desventajas traen como consecuencia que la información no esté actualizada o que en los diferentes directorios de almacenamiento de información se encuentre por diversos motivos duplicidad de datos, información incompleta o no coincidan los datos entre sí.

Actualmente está desarrollado en *Drupal,* y por las constantes variaciones en sus versiones no es posible actualizar el núcleo a la última versión estable que presenta dicha tecnología de desarrollo.

Basándose en estas limitaciones y con el objetivo de mejorar el proceso de flujo de información, se decide agregar nuevas funcionalidades al Directorio UCI, que respondan a las necesidades y preferencias de los usuarios en la Universidad.

1.6 Tendencias, tecnologías y herramientas a utilizar

Las tecnologías estudiadas y analizadas a continuación, para el desarrollo de la solución, se definieron por el grupo de arquitectura del Departamento de Tecnología de la Dirección de Informatización (DIN).

1.6.1 Metodología de desarrollo de software

La selección de una metodología de desarrollo de *software* adecuada, es un factor determinante en el éxito de un proyecto. Aunque no existe una metodología absoluta, algunas se ajustan mejor que otras, a las características y necesidades específicas de los proyectos de desarrollo. Dentro de las metodologías existen dos grandes grupos, las conocidas metodologías tradicionales y las metodologías ágiles. Las primeras enfatizan en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto y es recomendada para los proyectos con grandes equipos de desarrollo. Mientras que las ágiles dan mayor importancia a la capacidad de respuesta a los cambios, se enfatiza en la satisfacción del cliente y promueve el trabajo en equipo (21).

1.6.1.1 Desarrollo Ágil con Calidad

La metodología seleccionada de acuerdo a las características de los proyectos de la DIN será Desarrollo Ágil con Calidad (DAC), proceso de software que combina las buenas prácticas del nivel 2 del Modelo de Madurez de la Capacidad Integrado (CMMI[13] por sus siglas en inglés) con las buenas prácticas de la dirección y desarrollo ágil de proyectos de software. Es un proceso colaborativo, recursivo-iterativo,

[13] Modelo de calidad del software que clasifica las empresas en niveles de madurez.

incremental y guiado por procesos y requisitos. Su modelo del proceso es una adaptación del modelo en Cascada a los modelos Programación Extrema y Desarrollo concurrente. La metodología está enfocada a proyectos pequeños o proyectos grandes divididos en sub-proyectos que desarrollan software de gestión basado en componentes.

Este proceso tiene ocho actividades del marco de trabajo del proceso común llamadas Fases o Procesos del Ciclo de Vida: Inicio, Análisis y Diseño Arquitectónico, Requisitos, Construcción, Cierre de iteración, Liberación, Transición y Cierre, ocurriendo las iteraciones concurrentes entre las fases de Requisitos, Construcción y Cierre de iteración. Además, entre las fases de Requisitos y Construcción puede ocurrir un ciclo pues a medida que los requisitos son especificados estos pueden ir entrando a la fase de Construcción. El proceso tiene también dos Áreas de Procesos de Protección: Gestión de proyectos y Soporte, así como dos Fases o Procesos Horizontales cuyas tareas están presentes en varias de las fases del proceso común en forma de subprocesos: Arquitectura y Planificación (22). La Figura 2 muestra el modelo de proceso de la metodología DAC.

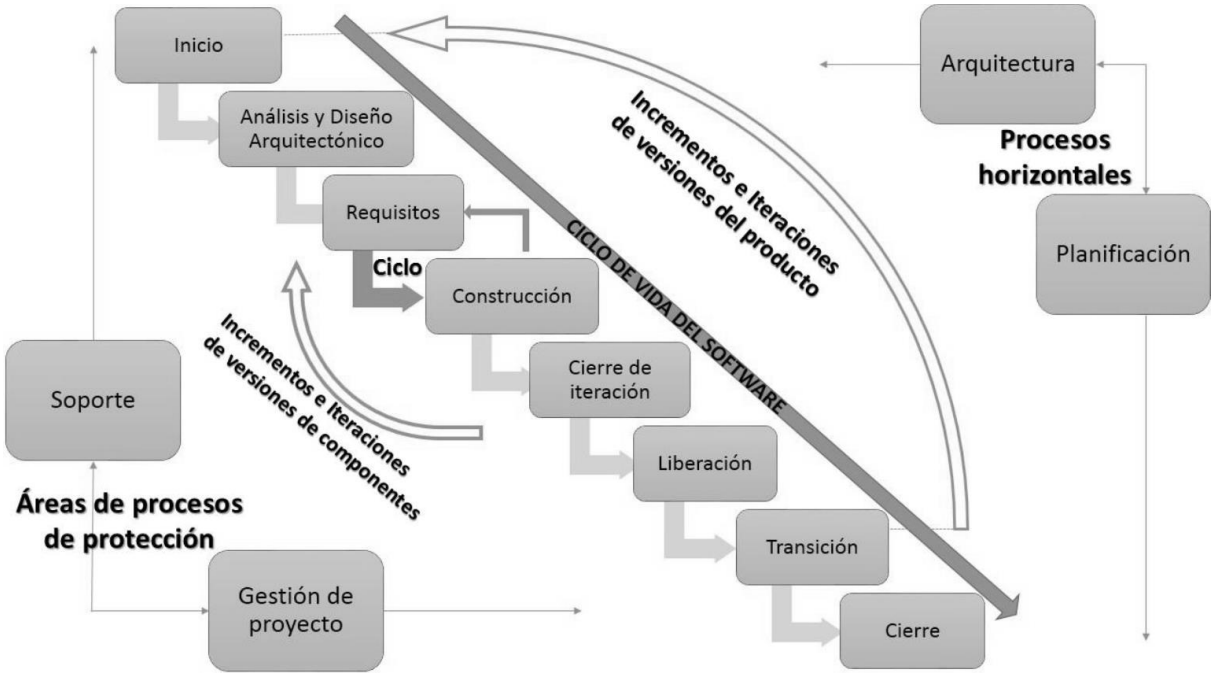


Figura 2: Modelo del proceso de Desarrollo Ágil con Calidad

El uso de la metodología DAC permitirá mantener un proyecto organizado, planificado y bien estructurado donde mediante las entregas a nivel de versiones del producto se podrá desarrollar de manera ágil y

concurrente lo que garantiza un producto con calidad y satisfacción del cliente en etapas tempranas del proyecto.

1.6.2 Marco de trabajo o *framework*

El marco de trabajo constituye un elemento importante a la hora de desarrollar una solución, pues es capaz de proveer una arquitectura sólida y más extensible para la implementación de los diferentes componentes en un sistema informático.

1.6.2.1 Marco de trabajo Symfony 2.8

El marco de trabajo para el desarrollo de la investigación es Symfony. Ofrece aplicaciones que cuentan con un código claro y organizado consistentemente, además que promueve la reutilización y permite a los nuevos desarrolladores ser productivos con mayor rapidez.

Es un marco de trabajo que ayuda a simplificar el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona una estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas. Symfony está diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, lo que permite al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

A principios del 2009 fue anunciada la versión 2 del mismo, la cual hace un cambio radical tanto en la arquitectura interna como en la filosofía de trabajo respecto a sus versiones anteriores. Esta ideado para sacar mayor provecho a las nuevas características de la versión 5.3 del lenguaje de Preprocesador de Hipertexto (PHP[14] por sus siglas en inglés), lo que lo convierte en unos de los *framework* con mayor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite remplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto. Además, incorpora las mejores ideas de otros marcos de trabajo incluso aunque estos no están implementados en PHP (23).

Algunas ventajas de Symfony2 son:

- Permite desarrollar software más rápido y mejor que con PHP simple.

[14] Lenguaje de código abierto adecuado para el desarrollo web.

- Es altamente personalizable y proporciona varias formas para realizar la misma tarea.
- Está diseñado para volver a lo básico: las herramientas de desarrollo que te permiten desarrollar más rápido y construir aplicaciones más robustas.

1.6.3 Lenguajes de programación

Los lenguajes de computadoras son una sintaxis codificada usada por los programadores para comunicarse con ella. Es el único lenguaje que entienden las computadoras, tanto los programas de software como el hardware. El lenguaje le permite al usuario dictar los comandos que la computadora debe entender para procesar los datos (24).

1.6.3.1 Lenguaje Unificado de Modelado UML 2.0

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema. Brinda un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocios, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de *software* reutilizables (25).

1.6.3.2 Lenguaje de Marcas de Hipertexto HTML 5

Lenguaje de Marcas de Hipertexto *(*HTML, por sus siglas en inglés), es el lenguaje de marcado predominante para la construcción de páginas web. En sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto e imágenes. Es un estándar a cargo de la *World Wide Web Consortium (*W3C[15], por sus siglas en inglés), organización dedicada a la estandarización de varias tecnologías ligadas a la web en lo referente a su escritura e interpretación. Es un lenguaje de hipertexto, que permite escribir texto de forma estructurada y agradable. Su nivel de complejidad es bajo por lo que no necesita de grandes conocimientos cuando se cuenta con un editor de páginas web. Sus archivos son pequeños, permite un despliegue rápido, es fácil de aprender y lo admiten la mayoría de los navegadores (26).

1.6.3.3 Hojas de Estilo en Cascada CSS 3

Hojas de Estilo en Cascada (CSS, por sus siglas en inglés), hace referencia a un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML. Su objetivo es separar la estructura

[15] Consorcio internacional que produce recomendaciones para la World Wide Web (WWW).

de un documento de su presentación. La información de estilo puede ser adjuntada como un documento separado o en el mismo documento HTML. Las Hojas de Estilo en Cascada en su versión 3 están divididas en varios documentos separados, llamados "módulos". Donde cada módulo añade nuevas funcionalidades a las definidas en CSS 2, de manera que se preservan las anteriores para mantener la compatibilidad (27).

1.6.3.4 Procesador de Hipertexto PHP 5.9.12

Preprocesador de Hipertexto (PHP, por sus siglas en inglés), es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en páginas web estáticas en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser utilizada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en varios de los sistemas operativos y plataformas sin ningún costo (28).

1.6.3.5 Java Script 1.8

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, es utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y lenguaje C. Se utiliza principalmente en su forma del lado del cliente implementado como parte de un navegador web, permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Es orientado a objetos, al disponer de herencia, la cual se realiza siguiendo el paradigma de programación basada en prototipos y las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. JavaScript es una excelente solución para poner en práctica la validación de datos de un formulario en el lado del cliente (29).

1.6.3.6 Lenguaje de Consulta Estructurado SQL

Lenguaje de Consulta Estructurado (SQL, por sus siglas en inglés), es un lenguaje declarativo de alto nivel vinculado con la gestión de bases de datos de carácter relacional, que permite la especificación de distintas clases de operaciones entre estas. SQL, al manejar conjuntos de registros y no registros individuales, ofrece una elevada productividad en la codificación y en la orientación a objetos. Se habla por tanto de un lenguaje normalizado que permite trabajar con varios lenguajes de programación (ejemplo PHP) en combinación con cualquier tipo de base de datos (30).

1.6.4 Herramientas de desarrollo

Las herramientas informáticas son programas, aplicaciones o simplemente instrucciones usadas para efectuar otras tareas de modo más sencillo. Cada herramienta se crea y diseña para una o varias funciones determinadas, por lo que existen diversos tipos de herramientas informáticas según el campo al que se dediquen (31).

1.6.4.1 Herramienta de modelado Visual Paradigm para UML 8.0

Herramienta multiplataforma de modelado visual UML y una herramienta de Ingeniería de Software Asistida por Computadoras (CASE[16], por sus siglas en inglés) muy potente y fácil de utilizar. Soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción y despliegue. Se puede realizar el modelado, la captura de requisitos, diseño de base de datos, modelado de procesos de negocio. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta también proporciona una mejor interfaz gráfica de usuario y una mayor base de datos de esquema de apoyo (32).

1.6.4.2 Herramienta para la construcción de prototipos Evolus Pencil 2.0.5

Herramienta de código abierto disponible para todas las plataformas y construida con el propósito de ofrecer la creación de diagramas y prototipos de interfaz de usuario ya sea de páginas web, software de escritorio o cualquier otra interfaz que necesite ser diseñada de forma rápida y sencilla. La versión independiente viene ideal para quienes no quieran ejecutar Pencil en su navegador. La versión de escritorio es más rápida que su versión para navegador, lo que es una ventaja (33).

1.6.4.3 Entorno de Desarrollo Integrado NetBeans 8.0

Plataforma de desarrollo modular para una amplia gama de tecnologías de desarrollo de aplicaciones. Incluye un editor avanzado en varios idiomas, depurador y perfiles, así como herramientas para el control de versiones y la colaboración de desarrolladores. Admite diferentes lenguajes de programación mediante los que se pueden crear aplicaciones gráficas. Posibilita que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Las aplicaciones construidas a partir de módulos se pueden extender agregándole nuevos módulos, debido a que permiten ser desarrollados

[16] Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software.

independientemente. Las aplicaciones implementadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software (34).

1.6.4.4 Administrador de base de datos PgAdmin III 1.22

Herramienta de código abierto para la administración de bases de datos PostgreSQL. Diseñada para responder a las necesidades de los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La aplicación incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar comandos programados y soporte para la replicación. La conexión al servidor puede hacerse mediante la familia de protocolos de internet (TCP/IP por sus siglas en inglés) y encriptarse mediante el protocolo de Capa de Conexión Segura (SSL por sus siglas en inglés) para mayor seguridad (35).

1.6.4.5 Sistema Gestor de Base de Datos PostgreSQL 9.4.1

PostgreSQL es un sistema gestor de bases de datos objeto-relacional, distribuido bajo la licencia BSD[17] y de código fuente libre; por lo que puede ser utilizado, modificado y distribuido por todo el mundo de forma gratuita. Incluye características de la programación orientada a objetos, como: herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. Tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados. PostgreSQL utiliza un modelo Cliente-Servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando (36).

1.6.4.6 Suite Pentaho 6.0

Es una herramienta de Inteligencia de Negocio (BI[18] por sus siglas en inglés) desarrollada bajo la filosofía del software libre para la gestión y toma de decisiones empresariales. La plataforma ha sido desarrollada bajo el lenguaje de programación *Java* y tiene un ambiente de implementación también basado en *Java*. Es la primera versión del servidor de clase empresarial con nuevas características para automatizar, gestionar y mejorar todas las etapas de la tubería de datos. Como parte de Pentaho 6 se combinan y aparean atributos físicos y virtuales. Ofrece etapas bajo encriptado para la extracción de información por medio de la API

[17] Licencia de software otorgada principalmente para los sistemas desarrollado bajo *Berkeley Software Distribution* (BSD).

[18] Habilidad para transformar los datos en información, y la información en conocimiento, de forma que se pueda optimizar el proceso de toma de decisiones en los negocios.

REST[19] y su transmisión también es encriptada. Además, la información de linaje puede ser almacenada en un sistema de archivos o almacén de datos encriptado para protegerla de futuras modificaciones (37).

1.6.4.7 Servidor Web NGINX

Es un servidor web que usa el Protocolo de Transferencia de Hipertexto (HTTP[20] por sus siglas en inglés), es de código abierto e incluye servicios de correo electrónico. Está listo para ser utilizado como un *proxy*[21] inverso para cachear el contenido de la web. Se utiliza para equilibrar la carga entre los servidores *back-end*[22], como también para ser utilizado como caché en un servidor *back-end* lento. Su arquitectura, es diferente al modelo tradicional, de crear una instancia por cada *request o* solicitud. Procesa decenas de miles de conexiones simultáneas en un proceso compacto y con varios núcleos de Unidad Central de Procesamiento (CPU por sus siglas en inglés). Además, se compone de módulos que se incluyen en tiempo de compilación. Eso significa que el usuario descarga el código fuente y selecciona qué módulos quiere utilizar. Hay módulos para la conexión a clones de aplicaciones, balanceo de carga, servidor *proxy*, y otros. No hay módulo para el lenguaje de Programación Interpretado (PHP por sus siglas en inglés), ya que *Nginx* puede interpretar código PHP en sí mismo. Es multiplataforma, ligero y de alto rendimiento (38).

1.6.4.8 JQuery 1.11.1

JQuery es un nuevo tipo de biblioteca o marco de trabajo de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones e incluir la tecnología JavaScript asíncrono y XML (AJAX por sus siglas en inglés) en el sistema. JQuery, al igual que otras librerías, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Es decir, con las funciones propias de la librería se logran grandes resultados en menos tiempo y espacio. La gran ventaja es que permite cambiar el contenido de la página web sin necesidad de recargarla, utilizando el Modelo de Objetos del Documento (DOM por sus siglas en inglés) y AJAX de manera extremadamente sencilla gracias a su sintaxis (39).

[19] Interfaz de Programación de Aplicaciones y Transferencia de estado representacional es un servicio que provee de funciones que dan la capacidad de hacer uso de un servicio web, dentro de una aplicación propia, de manera segura.

[20] Protocolo de transferencia de hipertextos.

[21] Actúa de intermediario entre un explorador web e Internet.

[22] Traducido al español significa detrás-final, dorsal final, motor. Es el software que corre en el servidor, estado final de un proceso.

1.6.4.9 Bootstrap 3.2

Es un framework que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. Ha sido desarrollado por Twitter que recientemente liberó su versión 2.0. La mayor ventaja es que se pueden crear interfaces que se adapten a los distintos navegadores (responsive design) apoyándonos en un framework potente con numerosos componentes webs que ahorrarán mucho esfuerzo y tiempo (40).

1.6.4.10 AngularJS 1.5

Es un framework JavaScript de desarrollo de aplicaciones web en el lado cliente, viene de la mano de los chicos de Google y se podría decir que utiliza el patrón MVC (Model-View-Controller), aunque ellos mismos lo definen más bien como un *MVW(Model-View-Whater)*[23] (41).

1.7 Conclusiones parciales

Luego del análisis realizado en el capítulo se concluye:

El estudio realizado de los temas relacionados al proceso de búsqueda de información sirvió de punto de partida para fundamentar la base teórica de la investigación.

El análisis de las principales características y aplicaciones de los sistemas de búsqueda, así como la experiencia adquirida, posibilitó el desarrollo de una nueva versión del sistema Directorio UCI, contribuirá a mejorar el proceso de búsqueda de información en la Universidad de las Ciencias Informáticas.

Se utilizó la metodología DAC para estructurar, planear y guiar el proceso de desarrollo, así como el marco de trabajo Symfony 2 que sigue un conjunto de buenas prácticas determinadas para el desarrollo de software.

Se emplearon tecnologías, lenguajes y herramientas establecidas por la Dirección de Informatización, que permitieron profundizar los conocimientos necesarios para el desarrollo de la propuesta de solución.

[23] Se definen como Modelo-Vista-*Lo-que-sea para* huir de la discusión entre Modelo Vista Vista Modelo vs Modelo Vista Controlador y Modelo Vista Presentador.

Capítulo 2: Propuesta de solución

Es prioridad para la UCI la automatización de sus principales procesos para un mejor funcionamiento y organización en la búsqueda de información que se genera. Para que esta informatización sea satisfactoria y cumpla con las expectativas, es vital el análisis de dichos procesos. Lo cual posibilita un mejor entendimiento entre el cliente y los desarrolladores del sistema a desarrollar. En este capítulo se identifican conceptos y objetos relacionados con la propuesta de solución y para ello se propone la realización de un modelo de dominio. Se detallan los requerimientos funcionales y no funcionales, así como las técnicas empleadas para su obtención y se realiza un estudio de la arquitectura, patrones de diseño y de bases de datos a utilizar en la solución.

2.1 Modelo de dominio

Existen varias alternativas para llevar a cabo el modelamiento de un sistema, algunas de las metodologías existentes para el desarrollo de software proponen la realización de un modelo de negocio para el caso en el que los procesos dentro del entorno estén claramente definidos, y la realización de un modelo de dominio para los escenarios en los que no puedan identificarse tales procesos del negocio.

Para la modelación del sistema propuesto no se lograron definir procesos específicos en el entorno, solamente identificar conceptos y objetos relacionados con el mismo, por ello se propone la realización de un modelo de dominio.

El Modelo de Dominio (*o Modelo Conceptual*) es una representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés. Captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema (42).

El objetivo de la realización del modelo de dominio es comprender y describir las clases más importantes dentro del contexto del sistema. La Figura 3 muestra el modelo de dominio de la presente investigación.

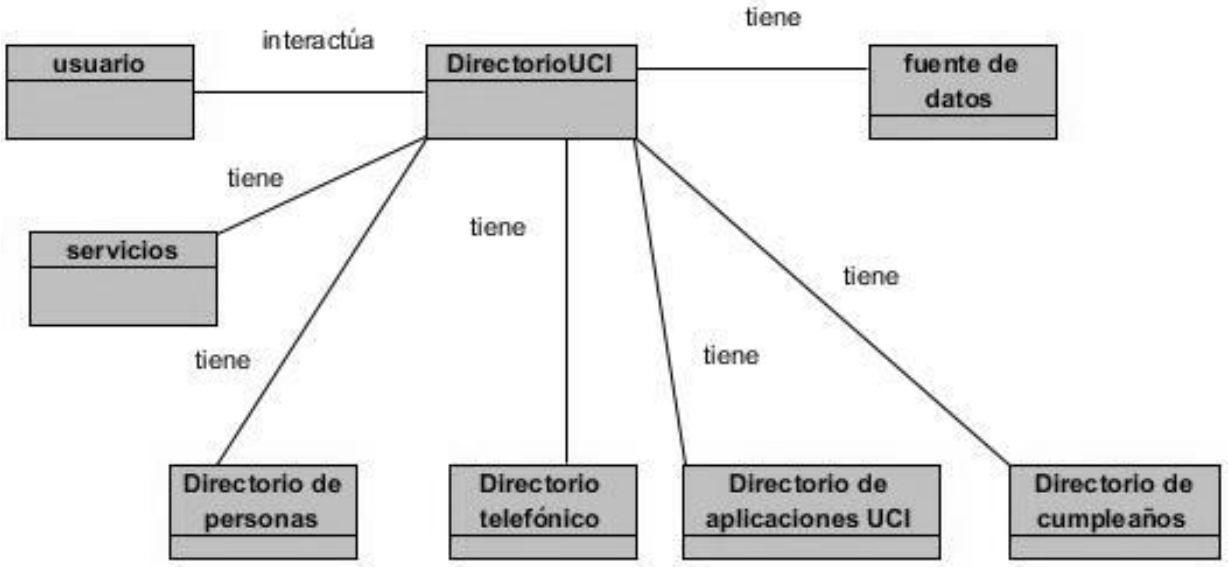


Figura 3: Modelo de dominio

Para un mayor entendimiento del modelo de dominio y con el objetivo de lograr un lenguaje que sea común para la comprensión del contexto del sistema se muestran a continuación los conceptos más importantes utilizados en el modelo:

Usuario: representa a todo el personal de la Universidad que accede al Directorio UCI para consultar información.

Directorio UCI: servicio que brinda el directorio de personas, teléfonos, cumpleaños y aplicaciones que posibilita a los usuarios la búsqueda de información.

Directorio de Personas: posibilita a los usuarios la búsqueda de personas de la Universidad.

Directorio de Teléfonos: posibilita a los usuarios la búsqueda de números telefónicos de la Universidad.

Directorio de Aplicaciones: posibilita a los usuarios la búsqueda de aplicaciones de la Universidad.

Cumpleaños: posibilita a los usuarios la búsqueda de fechas de cumpleaños de las personas de la Universidad.

Servicios: permite la interacción a los sistemas y servicios externos que contiene el Directorio UCI.

Fuente de datos: proporciona la información que es mostrada a través de los directorios de personas, teléfonos, aplicaciones y cumpleaños.

2.2 Descripción del sistema propuesto

La descripción de cualquier producto o solución informática es una idea, modelo o visión del software que se quiera implementar y esto permite que el cliente y el equipo de trabajo puedan imaginar cómo quedará el producto y la utilidad del mismo.

El Directorio en su versión 2.0, tiene como objetivo realizar búsquedas rápidas y completas atendiendo a los diferentes criterios de búsquedas especificados, que permitan facilitar la comunicación e interacción de los usuarios. Además, del desarrollo de funcionalidades para efectuar búsquedas simples y avanzadas sobre la información de personas, teléfonos, aplicaciones y cumpleaños en la Universidad.

Desde cada directorio se podrán gestionar cada uno de sus datos, lo que les permitirá crear sus listas personales, una vez que el usuario esté registrado en el sistema. Por su parte, el Directorio de personas contará con una opción que permita mostrar la geolocalización de locales dentro del centro. El de Cumpleaños, permitirá la notificación vía correo electrónico, en caso de ser seleccionada la opción por el usuario en el menú de Configuración que contendrá la nueva solución y el de Aplicaciones permitirá la gestión de las mismas, para darle una mejor utilización e interacción con el usuario. Además, se le agregará al sistema el diseño web *responsive* o adaptativo que es una técnica que busca la correcta visualización de una misma página en distintos dispositivos, desde ordenadores de escritorio a *tablets* y móviles.

Mediante la utilización de la *suite* o paquete *Pentaho Data Integration* (PDI), se hará uso de la herramienta *spoon*, encargada del diseño gráfico de transformaciones que permiten la Extracción, Transformación, Transporte y Carga de datos (ETTLs), para lograr un mecanismo de actualización entre la base de datos Pasarela_Directorio y la fuente de datos de la Universidad. Esta base de datos permitirá que las personas tengan asociados diversos perfiles, por ejemplo, que un trabajador tenga el cargo “Profesor” y a su vez el cargo de “Especialista B”. Además, que la información esté más organizada y en un único espacio de almacenamiento, para que su acceso sea directo y sin hacer un conjunto de búsquedas en diferentes sistemas.

Esta solución no está implementada sobre la misma arquitectura del sistema anterior, sino que utiliza Symfomy 2 para brindar un desarrollo ágil, robusto, flexible y de fácil integración con otros sistemas implementados en la institución.

Para una mejor comprensión de la propuesta de solución a continuación se ofrecen los artefactos ingenieriles que ayudan a entender de una forma técnica el desarrollo de la solución.

2.3 Especificación de requisitos de software

Una vez que se ha analizado el dominio del problema, es de vital importancia dejar plasmado lo que debe hacer el sistema una vez creado. Para definir esto, se especifican los requisitos, que no son más que las condiciones o capacidades que tienen que ser alcanzadas por un sistema para satisfacer las necesidades del cliente. Su objetivo es identificar y documentar lo que en realidad se necesita (43). Se clasifican en dos grupos: requisitos funcionales y requisitos no funcionales.

2.3.1 Técnicas de obtención de requisitos

La obtención de requerimientos es una de las cuatro actividades que define la Ingeniería de Requisitos. Es el proceso donde los interesados en un sistema de software descubren, revelan y entienden sus requerimientos. Existen diferentes técnicas para identificar los requisitos, dentro de las que se encuentran: sesiones de tormentas de ideas, entrevistas, observación de campo, revisión de la documentación técnica, análisis de sistemas existentes, ingeniería inversa, simulaciones y prototipos (44).

De las técnicas antes mencionadas, para la captura de requerimientos de la solución propuesta se utilizaron las siguientes:

Entrevistas: se realizaron reuniones con las partes interesadas, donde se efectuaron una serie de preguntas para obtener una mayor comprensión de la problemática existente. Se ifentificaron varios de los requerimientos de la solución. Su diseño está accesible en el Anexo 5.

Prototipos: un prototipo es una versión inicial de un sistema de software que se utiliza para demostrar los conceptos, probar las opciones de diseño y conocer más acerca del problema. Los prototipos se le mostraron al cliente con el objetivo de su validación, quienes proporcionaron requisitos adicionales o modificaron varios de los anteriores.

Análisis de sistemas existentes: se analizaron distintos sistemas ya desarrollados que están relacionados con el proceso de búsqueda de información, lo que permitió la obtención de ideas que ampliaran el desarrollo de la investigación. Muchas de las funcionalidades presentes en los sistemas estudiados sirvieron de base para las que debe cumplir la propuesta de solución.

2.3.2 Requisitos funcionales del sistema

Antes de comenzar el diseño y la implementación de cualquier sistema, es necesario saber qué debe hacer dicho sistema y cómo debe hacerlo, dejar bien claro lo que los clientes desean del sistema a desarrollar.

Para obtener exitosamente esta información y desarrollar con calidad el producto. En la Tabla 2 se especifican los requisitos funcionales del sistema Directorio UCI versión 2.0.

Tabla 2: Listado de requisitos funcionales del sistema

No.	Nombre del requisito	Complejidad	Prioridad
RFD1	Realizar búsquedas en los directorios sin autenticación	Baja	Alta
RFD2	Permitir comunicación entre usuarios autenticados y administradores del sistema	Baja	Baja
RFD3	Notificar cumpleaños vía correo electrónico	Alta	Alta
RFD4	Establecer mecanismo para la actualización de los datos	Alta	Alta
RFD5	Crear una *API REST* para los datos de las personas	Media	Alta
RFD6	Autenticar usuario	Baja	Alta
Directorio de personas			
RFD7	Realizar búsqueda de persona(s)	Baja	Alta
RFD8	Realizar búsqueda avanzada de persona(s) • Nombres • Apellidos • Usuario • Solapín • Apartamento • Categoría • Cargo (*si no es estudiante*) • Área • Provincia • Municipio • Género • Año *(si es estudiante)* • Grupo *(si es estudiante)* • Teléfono	Media	Alta
RFD9	Mostrar datos de la(s) persona(s) buscada(s)	Media	Alta
RFD10	Ampliar la imagen cuando se le hace clic	Baja	Media
RFD11	Permitir enviar correo electrónico a las personas que poseen cuenta de correo electrónico	Baja	Media
RFD12	Mostrar ubicación residencial si es interno (mediante el Sistema de Información Geográfica de la Universidad (SIGUCI))	Alta	Media
Directorio de personas con usuario autenticado			
RFD13	Crear lista	Media	Media
RFD14	Adicionar persona(s) a una lista	Media	Media
RFD15	Eliminar persona(s) de una lista	Media	Media
RFD16	Modificar lista de personas	Media	Media
RFD17	Mostrar personas de una lista	Media	Media
RFD18	Seleccionar todos los elementos de una lista	Baja	Media
RFD19	Enviar correo electrónico a integrantes de una lista o varias listas	Baja	Media

RFD20	Exportar datos de personas previamente seleccionados de una lista	Baja	Media
RFD21	Importar listas de personas	Media	Media
Directorio Telefónico			
RFD22	Realizar búsqueda de teléfono(s)	Baja	Alta
RFD23	Realizar búsqueda avanzada de teléfono(s) • Teléfono • Área • Nombre en pizarra	Media	Alta
RFD24	Mostrar datos de la búsqueda de teléfonos	Baja	Alta
RFD25	Mostrar páginas amarillas • Docencia • Centros de Reportes • Salud • Comercio y gastronomía • Comunicaciones • Servicios Generales • Tecnología • Información • Organizaciones • Transporte • Copextel • Seguridad y Protección • Deporte • Residencia	Baja	Alta
RFD26	Mostrar teléfonos de urgencias • Puestos de Mandos • Transporte • Cuerpo de guardia del hospital • Aguas de la Habana • Mantenimiento • SEPCOM • Información	Baja	Alta
RFD27	Mostrar la ubicación del local al que pertenece el resultado de la búsqueda (mediante el Sistema de Información Geográfica de la Universidad (SIGUCI))	Media	Media
Directorio Telefónico con usuario autenticado			
RFD28	Adicionar agenda	Alta	Media
RFD29	Adicionar teléfono(s) a un agenda	Media	Media
RFD30	Eliminar teléfono(s) de una agenda	Media	Media
RFD31	Modificar agenda de teléfono	Media	Media
RFD32	Mostrar teléfonos de una agenda	Media	Media
RFD33	Seleccionar todos los elementos de una agenda telefónica	Baja	Media

RFD34	Exportar agenda	Baja	Media
RFD35	Importar agenda	Baja	Media
Cumpleaños			
RFD36	Realizar búsqueda de cumpleaños	Baja	Alta
RFD37	Realizar búsquedas avanzadas de cumpleaños • Nombres • Apellidos • Grupo • Área • Apartamento	Media	Alta
RFD38	Realizar búsqueda estática por inicial del nombre	Baja	Alta
RFD39	Mostrar datos de la búsqueda de cumpleaños	Baja	Alta
RFD40	Mostrar cumpleañeros del día	Baja	Alta
RFD41	Mostrar calendario para realizar búsqueda por fechas	Baja	Alta
Cumpleaños con usuario autenticado			
RFD42	Mostrar opción de recordar cumpleaños	Baja	Media
RFD43	Mostrar opción dejar de recordar los cumpleaños	Baja	Media
RFD44	Adicionar cumpleaños recordados	Baja	Media
RFD45	Eliminar cumpleaños recordados	Baja	Media
RFD46	Modificar cumpleaños recordados	Baja	Media
RFD47	Mostrar cumpleaños recordados	Media	Media
RFD48	Mostrar configuración de cumpleaños • Cantidad de días con antelación de la notificación • Vía de notificación (correo electrónico, propio directorio, ambas vías)	Media	Baja
Directorio de Aplicaciones con usuario autenticado (solo rol administrador)			
RFD49	Adicionar aplicaciones	Media	Media
RFD50	Eliminar aplicaciones	Baja	Media
RFD51	Modificar aplicaciones	Baja	Media
Directorio de Aplicaciones			
RFD52	Mostrar aplicaciones	Baja	Alta
RFD53	Realizar búsquedas simples con el uso de los criterios: nombre, *tipo de visibilidad o su dirección.*	Baja	Alta
RFD54	Agrupar por esferas (facultades, docencia, comunidades, entre otros.)	Media	Alta

2.3.3 Requisitos no funcionales del sistema

Los requisitos no funcionales imponen restricciones en el diseño o la implementación. Son propiedades o cualidades que el producto debe tener. En la Tabla 3 se muestran los requisitos no funcionales definidos para la propuesta de solución.

Tabla 3: Listado de requisitos no funcionales del sistema

Usabilidad	
RNFD1	La terminología del menú debe ser constante en todo el sitio. No pueden existir páginas internas donde existan elementos diferentes del menú o simplemente no aparezcan.
RNFD2	El sistema será utilizado por todo el personal que tenga acceso a la red UCI y necesite hacer uso del mismo.
Fiabilidad	
RNFD3	En caso de fallos el sistema debe notificar a los administradores para su posterior solución y de la misma forma notificarle al usuario sin que se pueda comprometer la seguridad e integridad del sistema.
RNFD4	Los usuarios deben de autenticarse para poder acceder a las listas de personas, agenda telefónica y cumpleaños recordados.
Eficiencia	
RNFD5	El sistema debe responder en un tiempo menor a los 30 segundos.
RNFD6	El sistema deberá soportar una conexión simultánea de al menos 1000 usuarios.
Soporte	
RNFD7	El producto debe recibir mantenimiento ante cualquier fallo que ocurra.
RNFD8	Se debe lograr una solidez de los datos de tal manera que se minimice la concurrencia entre las búsquedas por parte de los usuarios y actualización de la base de datos por los administradores.
RNFD9	El sistema debe ser escalable, para en un futuro incorporarle nuevas funcionalidades en caso de ser necesarias.
Hardware	
RNFD10	Debe existir una red de área local para permitir el acceso al sistema.
RNFD11	Para uso del cliente: PC Pentium 3 o superior, CPU 1GHz o superior, 128 MB de memoria RAM mínimo 512 RAM recomendada o superior.
RNFD12	Para el desarrollo: CPU Intel Pentium 4 o superior, 2GHz o superior, 1 GB RAM o superior, 160 GB HDD o superior.
RNFD13	Para uso del servidor: CPU 3GHz, memoria RAM de 6GB 160 GB HDD.
Apariencia o interfaz externa	
RNFD14	Interfaz amigable y sencilla.
RNFD15	El sistema debe ser responsive o adaptativo, es decir, adaptable a todas las resoluciones.
RNFD16	La interfaz contará con menús para acceder a cada una de las vistas del sistema.
RNFD17	Su diseño gráfico debe ser acorde a las pautas de diseño de la Universidad.
Restricciones del diseño	
RNFD18	El software deberá ser desarrollado en su totalidad con tecnologías y componentes de código abierto.
Seguridad	
RNFD19	Cada usuario de acuerdo a su jerarquía en el sistema tendrá acceso a las funcionalidades permitidas.
RNFD20	Establecer permiso de solo lectura a la carpeta donde se encuentra alojado el sitio.
RNFD21	Garantizar la edición de la información solo al personal que tiene permiso a realizar esta actividad.

Confidencialidad	
RNFD22	Debe mantenerse la consistencia de los datos en correspondencia con la realidad, mediante la actualización con las fuentes correspondientes.
Disponibilidad	
RNFD23	Los usuarios deben tener acceso a la información desde cualquier dispositivo sin que los mecanismos utilizados para la seguridad de los datos retrasen la obtención de los mismos.
RNFD24	El sistema deberá ser accesible desde cualquier punto de la red que se disponga.
Portabilidad	
RNFD25	El sistema debe ser multiplataforma.

2.3.4 Descripción de requisitos funcionales

Con el objetivo de presentar los requisitos funcionales de una manera más consistente y detallada, se realizó la descripción de cada requisito funcional. Se definieron secciones para describir las acciones que se deben llevar a cabo por el usuario para cumplir con la funcionalidad en el sistema. A continuación, en las Tablas 4, 5 y 6 se muestran varias descripciones de requisitos de la propuesta de solución.

Tabla 4: Realizar búsqueda avanzada de persona(s)

No.	Nombre del requisito	Descripción del requisito	Complejidad	Prioridad
RFD 8	Realizar búsqueda avanzada de persona(s)	Se muestra la interfaz donde el usuario podrá seleccionar la búsqueda que desea realizar ya sea por: Personas, Teléfonos, Cumpleaños o Aplicaciones. El usuario selecciona la opción “Personas”, donde se muestra una interfaz con un buscador que permite buscar por: *Nombres, Apellidos, Solapín, No. Expediente, Usuario.* Además, permite filtrar la búsqueda por: *Nombres, Apellidos, Usuario, Solapín, Categoría, Cargo, Área, Provincia, Municipio, Género, Grupo, Apartamento y Teléfono.* El usuario selecciona o introduce sus palabras clave para la búsqueda y el sistema mostrará las personas que coincidan con los criterios de búsquedas especificados.	Media	Alta
Prototipo				

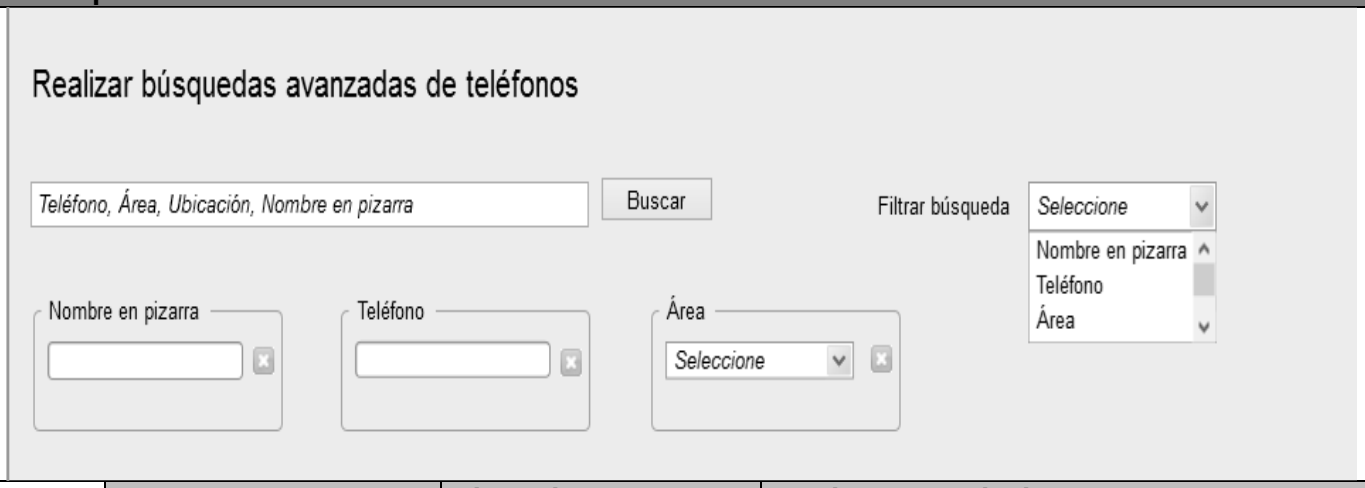
	Campos	**Tipos de Datos**	**Reglas o Restricciones**
	Buscador	Campo de texto	Permite buscar por los criterios: *Nombres, Apellidos, Solapín, No. Expediente, Usuario.*
	Nombres	Campo de texto	Campo opcional
	Apellidos	Campo de texto	Campo opcional
	Usuario	Campo de texto	Campo opcional
	Solapín	Campo de texto	Campo opcional
	Categoría	Campo de selección	Campo opcional
	Cargo	Campo de selección (si no es estudiante)	Campo opcional
	Áreas	Campo de selección	Campo opcional
	Provincia	Campo de selección	Campo opcional
	Municipio	Campo de selección	Campo opcional
	Sexo	Campo de selección	Campo opcional
	Grupo	Campo de texto (solo para estudiantes)	Campo opcional
	Año	Campo de selección (solo para estudiantes)	Campo opcional
	Apto	Campo de texto (si es interno)	Campo opcional
	Teléfono	Campo de texto (si es interno)	Campo opcional
	Observaciones	1. El usuario no deberá estar autenticado en el sistema. 2. El usuario deberá llenar al menos uno de los campos mostrados para obtener los resultados de la búsqueda.	

		3. Si el usuario introduce uno de los criterios especificados incorrectamente o que no exista en el sistema, se muestra un mensaje *"No hay resultados de la búsqueda"* 4. Si el usuario introduce uno de los criterios especificados correctamente se mostrarán todos los datos relacionados al criterio especificado.

Tabla 5: Realizar búsqueda avanzada de teléfono(s)

No.	Nombre del requisito	Descripción del requisito	Complejidad	Prioridad
RFD23	Realizar búsqueda avanzada de teléfono(s)	Se muestra la interfaz donde el usuario podrá seleccionar la búsqueda que desea realizar ya sea por: Personas, Teléfonos, Cumpleaños o Aplicaciones. El usuario selecciona la opción "Teléfonos", al dar clic en ella se muestra una interfaz con un buscador que permite buscar por: *Teléfono, Área, Ubicación, Nombre en pizarra.* Además, permite filtrar la búsqueda por: *Nombre en pizarra, Teléfono y Área.* El usuario escoge o introduce sus palabras clave para la búsqueda y el sistema mostrará los teléfonos que coincidan con los criterios de búsquedas especificados.	Media	Alta

Prototipo

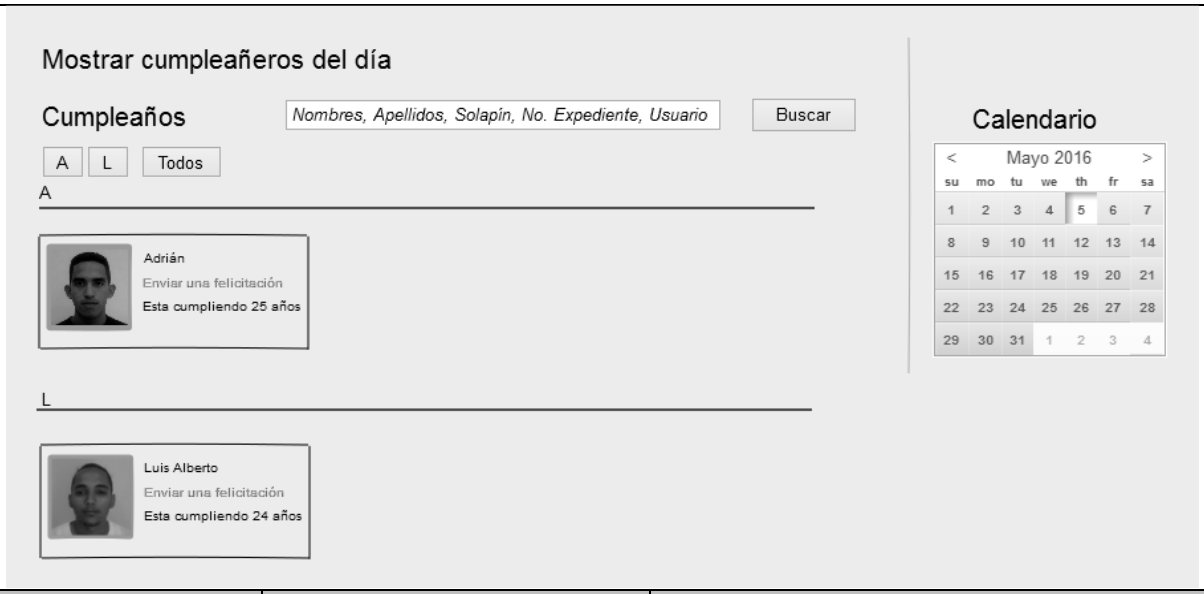


	Campos	Tipos de Datos	Reglas o Restricciones
	Buscador	Campo de texto	Permite buscar por los criterios: *Teléfono, Área, Ubicación, Nombre en pizarra.*
	Nombre en pizarra	Campo de texto	Campo opcional
	Teléfono	Campo de texto	Campo opcional

	Área	Campo de selección	Campo opcional
	Observaciones	1. El usuario no deberá estar autenticado en el sistema. 2. El usuario deberá llenar al menos uno de los campos mostrados para obtener los resultados de la búsqueda. 3. Si el usuario introduce uno de los criterios especificados incorrectamente o que no exista en el sistema, se muestra un mensaje *"No hay resultados de la búsqueda".* 4. Si el usuario introduce uno de los criterios especificados correctamente se mostrarán todos los datos relacionados al criterio especificado.	

Tabla 6: Mostrar cumpleañeros del día

No.	Nombre del requisito	Descripción del requisito	Complejidad	Prioridad
RFD39	Mostrar cumpleañeros del día	Se muestra la interfaz donde el usuario podrá seleccionar la búsqueda que desea realizar ya sea por: Personas, Teléfonos, Cumpleaños o Aplicaciones. El usuario selecciona la opción "Cumpleaños", al dar clic en ella se muestra una interfaz con todas las personas que cumplen años ese día. Además, muestra un buscar por: *Nombres, Apellidos, Usuario, Solapín, Número de expediente.* Muestra un calendario que resalta el día actual de la búsqueda. El usuario tiene la posibilidad de escoger por inicial o introducir las palabras clave para realizar la búsqueda del cumpleaños de la persona especificada.	Media	Media
Prototipo				



	Campos	**Tipos de Datos**	**Reglas o Restricciones**
	Buscar	Campo de texto	Permite buscar por los criterios: *Nombres, Apellidos, Solapín, No. Expedientes, Usuario.*
	Observaciones	1. El usuario no deberá estar autenticado en el sistema. 2. Si introduce un caracter en el buscador, se muestra un mensaje de advertencia *"El criterio debe tener longitud de 3 caracteres como mínimo".* 3. Si el usuario introduce uno de los criterios especificados incorrectamente o que no exista en el sistema, se muestra un mensaje *"No hay resultados de la búsqueda"* 4. Si el usuario introduce uno de los criterios especificados correctamente se mostrarán todos los datos relacionados al criterio especificado.	

2.3.5 Mapa de navegación

Los mapas de navegación se utilizan para ofrecer una visión global de todo el sistema y mostrar todos los flujos y enlaces que existen entre las diferentes pantallas. Una vez descrita la solución propuesta se muestran como quedaron distribuidas las funcionalidades de la propuesta de solución (Figura 4):

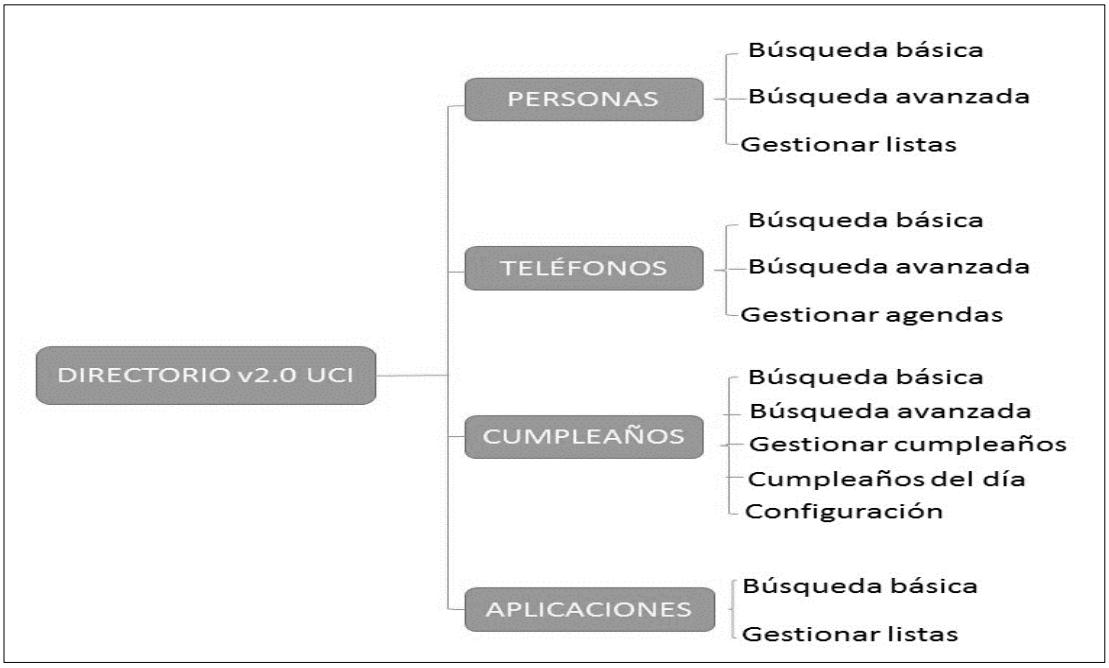


Figura 4: Mapa de navegación

2.4 Descripción de la arquitectura

La arquitectura de un software es la estructura u organización de un sistema que incluye los componentes de este, las propiedades visibles externas de esos componentes y las relaciones que existen entre ellos. Su diseño considera dos niveles: el diseño de datos y el diseño arquitectónico. El primero permite representar los componentes de la arquitectura y las definiciones de clases, el segundo se concentra en representar la estructura de software, sus propiedades e interacciones. La propuesta de solución específica para el desarrollo de dicho sistema el estilo arquitectónico Cliente-Servidor haciendo uso del patrón Modelo-Vista-Controlador (MVC).

2.4.1 Estilo arquitectónico Cliente-Servidor

En el desarrollo de la investigación el estilo arquitectónico propuesto es Cliente-Servidor, que es un modelo de aplicación distribuido en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes (45). Un cliente realiza peticiones a otro programa conocido como servidor que le da respuesta. Normalmente, el servidor es una máquina bastante potente que puede actuar como servidor de aplicaciones, de depósito de datos, o sencillamente para brindar

determinados servicios. Por otro lado, los clientes suelen ser estaciones de trabajo que realizan varias solicitudes o peticiones al servidor. Ambas partes deben estar conectadas entre sí mediante una red.
La Figura 5 es una representación gráfica del estilo arquitectónico.

Figura 5: Representación de la arquitectura Cliente-Servidor

La *International Business Machines* (IBM[24], por sus siglas en inglés) define al modelo Cliente-Servidor como: *"La tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo o a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o "clientes", resultan en un trabajo realizado por otros computadores llamados servidores".*
Cliente-Servidor es una arquitectura muy usada, debido a las ventajas y potencialidades que proporciona. La separación entre cliente y servidor es de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un solo programa. Los tipos específicos de servidores incluyen los servidores web, de archivo, correo y bases de datos (46).

2.4.2 Patrón de arquitectura

El patrón arquitectónico Modelo-Vista-Controlador (MVC) separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio (47).

[24] IBM: Empresa multinacional que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

Modelo: es la representación de la información con la que el sistema opera, por lo que gestiona los accesos, las consultas y actualizaciones de los datos del sistema. Interactúa con los controladores de la aplicación, respondiendo sus peticiones de acceso al sistema de persistencia de información.

Vista: es la presentación de la información en el sistema. Se comunica con el controlador del que recibe los datos para su visualización y captura los eventos que desencadenan acciones de acceso a datos.

Controlador: reacciona a los eventos desencadenados por el cliente, ejecutando la acción adecuada e invoca peticiones a los modelos que necesite cuando se hace alguna solicitud sobre la información. El controlador hace de intermediario entre la vista (forma en que se presenta la información) y el modelo (proveedor de la información). En la Figura 6 se muestra la representación del Patrón MVC.

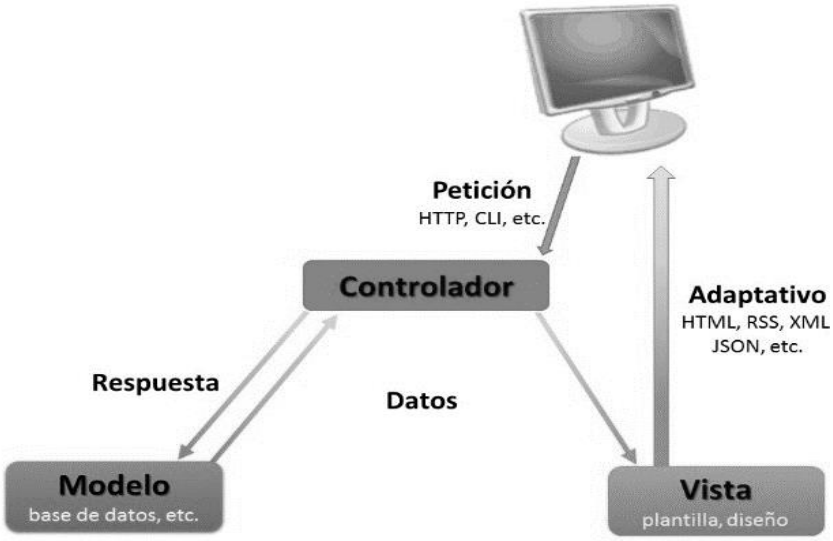


Figura 6: Representación del Patrón Modelo-Vista-Controlador

Symfony basa su funcionamiento en el patrón arquitectónico MVC, donde la capa del controlador está dividida en dos partes: el controlador frontal, que es el único punto de entrada a la aplicación para un entorno dado y las acciones, que contienen la lógica de las páginas, y la vista, que está compuesta por diversas partes, estando cada una de ellas especialmente preparadas para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones (48).

El usuario de Symfony debe conocer con cierto detalle el patrón MVC, ya que dota a la aplicación de una estructura más visible, y ayuda al programador de aplicaciones web a colocar cada cosa en su sitio y a construir aplicaciones altamente modulares, extensibles, portables, mantenibles y, sobre todo, vivas durante mucho tiempo.

2.4.3 Diagrama de despliegue

El diagrama de despliegue es un diagrama estructurado que muestra la arquitectura desde el punto de vista del despliegue. Se utiliza para mostrar la estructura física del sistema, incluyendo las relaciones entre el *hardware* y el *software* que se despliega, estas relaciones son representadas por los protocolos de comunicación que se utilizan para acceder a cada uno (49). A continuación se representa el diagrama de despliegue, estructurado por todos los nodos necesarios para acceder al Directorio UCI v2.0.

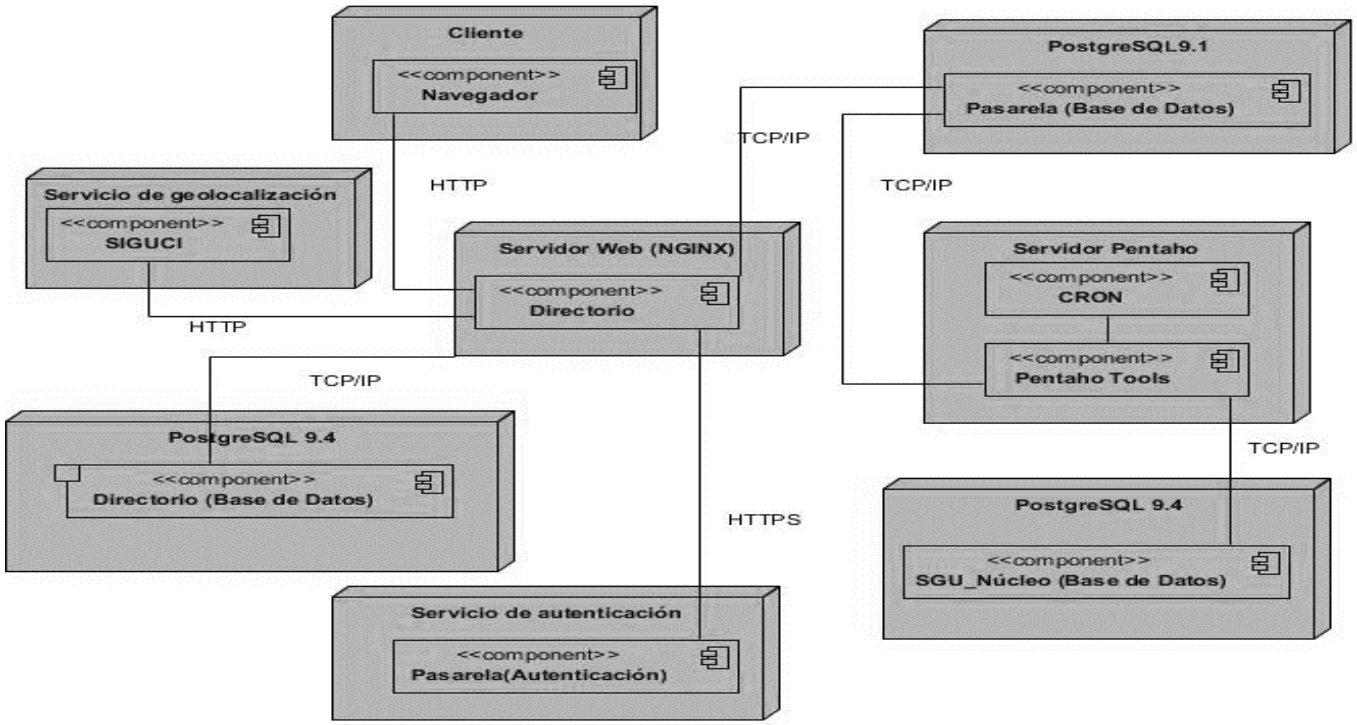


Figura 7: Representación del modelo de despliegue

Cliente (Navegador): su función es acceder e interactuar con el sistema según sus necesidades. Al estar la aplicación desarrollada sobre la web, la máquina cliente necesita de muy pocas prestaciones puesto que solo necesita un navegador web para poder acceder al sistema y realizar las operaciones necesarias.

Servidor Web NGINX (Directorio): aquí reside la capa de presentación del sistema, la que es accedida por las máquinas clientes a través de un navegador web. Contiene las funcionalidades del sistema.

PostgreSQL 9.4 Directorio (Base de Datos): encargado de almacenar la información generada del sistema.

PostgreSQL 9.1 Pasarela (Base de Datos): por donde la información pasa entre dos entidades distintas.

Servidor Pentaho (CRON): mecanismo de actualización entre la base de datos Pasarela_Directorio y la fuente de dato de la Universidad.

Servicio de autenticación (Pasarela): representa el servicio web que se consulta para realizar el proceso de autenticación de los usuarios.

Servicio de geolocalización (SIGUCI): representa el Sistema de Información Geográfica de la UCI, brinda la ubicación a través de un mapa.

PostgreSQL 9.4 SGU_Núcleo (Base de Datos): representa el sistema que se encarga de la gestión de la información de las personas de la Universidad.

HTTP: protocolo de transferencia de hipertexto, es un estándar de red que sigue el esquema petición-respuesta entre un cliente y un servidor.

HTTPS: protocolo seguro de transferencia de hipertexto, basado en el protocolo HTTP.

TCP/IP: familia de protocolos de Internet es un conjunto de estándares de red en la que se basa la red global y que permite la transmisión de datos entre redes de computadoras sin importar el tamaño.

2.4.4 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Cada patrón describe un problema que ocurre una y otra vez en el entorno y describe también el núcleo de la solución al problema. Esta solución provee una forma confiable, segura y probada para resolver problemas recurrentes en el diseño de software. Los patrones de diseño tienen como finalidad precisar en detalle los subsistemas y componentes de la aplicación (50).

Patrones para asignar responsabilidades

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP, por sus siglas en inglés) representan los principios básicos para la asignación de responsabilidades a objetos expresados en forma de patrones. Aunque son considerados más bien como una serie de buenas prácticas de aplicación recomendable en el diseño de software (51). Dentro de los patrones utilizados para estructurar el diseño del sistema se destacan:

Experto: se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad. Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos

o miles de responsabilidades. Si estas se asignan en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y puede presentar la oportunidad de reutilizar los componentes en futuras aplicaciones.

Symfony utiliza este patrón puesto que *Propel* es la librería que usa para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y se encarga de generar las clases *Peer* las cuales poseen las funcionalidades requeridas para obtener los datos almacenados en las entidades.

Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos. Esta es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades pertenecientes a ella. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento.

Symfony en la clase *Actions* define acciones dentro de las cuales se crean objetos de las clases *PEER*, que son las clases representativas de las entidades; evidenciando de este modo que la clase *Actions* es "creador" de dichas entidades.

Alta Cohesión: es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión garantiza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Un conjunto de clases con baja cohesión trae como consecuencia una serie de desventajas como: difíciles de comprender, reutilizar, conservar y las afectan constantemente los cambios. Symfony permite asignar responsabilidades con una alta cohesión, por ejemplo, la clase *Actions* tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el software sea flexible frente a grandes cambios.

Controlador: define que se le debe asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. El uso del patrón se evidencia en las clases controladoras que se encargan de obtener los datos y enviarlos a las librerías y las vistas, así como de manejar los posibles errores o mensajes que se muestran.

Todas las peticiones web son manipuladas por un solo controlador frontal que es el punto de entrada único de toda la aplicación en un entorno determinado.

Bajo Acoplamiento: el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Una clase con bajo acoplamiento no depende de muchas clases. El patrón propone que cada clase debe tener un bajo grado de dependencia con otras clases en la medida de lo posible.

Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

Patrones de la "Banda de los cuatro"

Los patrones "Banda de los Cuatro" (GoF, por sus siglas en inglés) describen las formas comunes en que diferentes tipos de objetos, pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases y la formación de estructuras de mayor complejidad. Además, permiten crear grupos de objetos para ayudar a realizar tareas complejas (52). Los patrones GoF pueden ser de tres tipos: de creación, comportamiento y estructurales.

Patrones de creación: se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de qué clase crear o cómo crearla.

Instancia única (*Singleton*): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Restringe la instanciación de una clase o valor de un tipo a un solo objeto.

Patrón de comportamiento: comprenden la asignación de responsabilidades entre objetos y algoritmos. Estos no solo conciernen a los objetos y las clases sino también la comunicación entre estas, además caracterizan flujos de control complejos que son difíciles de seguir en tiempo de ejecución.

Mediador (*Mediator*): se evidencia en las librerías, las cuales son mediadoras entre las clases controladoras y las modelos o acceso a datos.

2.4.5 Patrones de diseño de bases de datos

Los patrones de diseño de bases de datos son una plantilla que ya ha sido evaluada como la responsable de resolver un problema, es la guía para el trabajo, permiten crear una de base de datos más fortalecida por el usuario. Para el diseño y la construcción de una base de datos se requiere del mayor análisis posible pues a partir de este diseño se crea la base de datos, en la actualidad suelen ser muy grandes y a veces el trabajo con los patrones de diseño hacen que el trabajo sea más fácil, además asegura un resultado correcto

(53). A continuación se describen varios patrones de diseño de base de datos que se utilizan en la propuesta de solución.

Llaves subrogadas: este patrón es muy utilizado por facilitar la interacción con la base de dato en un futuro. El mismo plantea que se genere una llave primara única para cada entidad, en vez de usar un atributo identificador en el contexto dado. El uso del patrón permite que las tablas sean más fáciles de consultar a partir del identificador, pues los tipos de datos son iguales en cada una de las tablas. El patrón se manifiesta en la definición de la llave primaria de todas las tablas del modelo físico porque se generan llaves primarias independientes a los atributos de las entidades.

Árbol fuertemente codificado: cuando existe un conjunto de nodos conectados en la estructura de hijo a padre se está en presencia del patrón. Admite tantos niveles como requiera la jerarquía que se vaya a representar y es normalmente utilizado para representar jerarquías donde es bien conocida la estructura.

Esquema de copo de nieve: consta de una tabla de hechos que está conectada a muchas tablas de dimensiones, que pueden estar conectadas a otras tablas de dimensiones a través de una relación de muchos a uno. Entre sus principales ventajas está que permite eliminar redundancia de datos.

2.4.6 Modelo de la base de datos

Un modelo de datos es una colección de conceptos y reglas que se emplean para describir la estructura de una base de datos que incluye entidades, atributos y relaciones entre estos (54).

El modelo de la base de datos consta de tres fases: diseño conceptual, lógico y físico de la base de datos. La primera fase consiste en la producción de un esquema conceptual que es independiente de las consideraciones físicas. Luego el modelo se refina en un esquema lógico eliminando las construcciones que no se puede representar en el modelo de base de datos escogido. En la tercera fase, el esquema lógico se transforma en un esquema físico para el Sistema Gestor de Base de Datos elegido. La fase de diseño físico considera las estructuras de almacenamiento y los métodos de acceso necesarios para proporcionar un acceso eficiente a la base de datos.

Para diseñar la base de datos se elaboró el modelo físico de datos. Las clases persistentes que se muestran en la Figura 8 cubren las necesidades de la propuesta de solución.

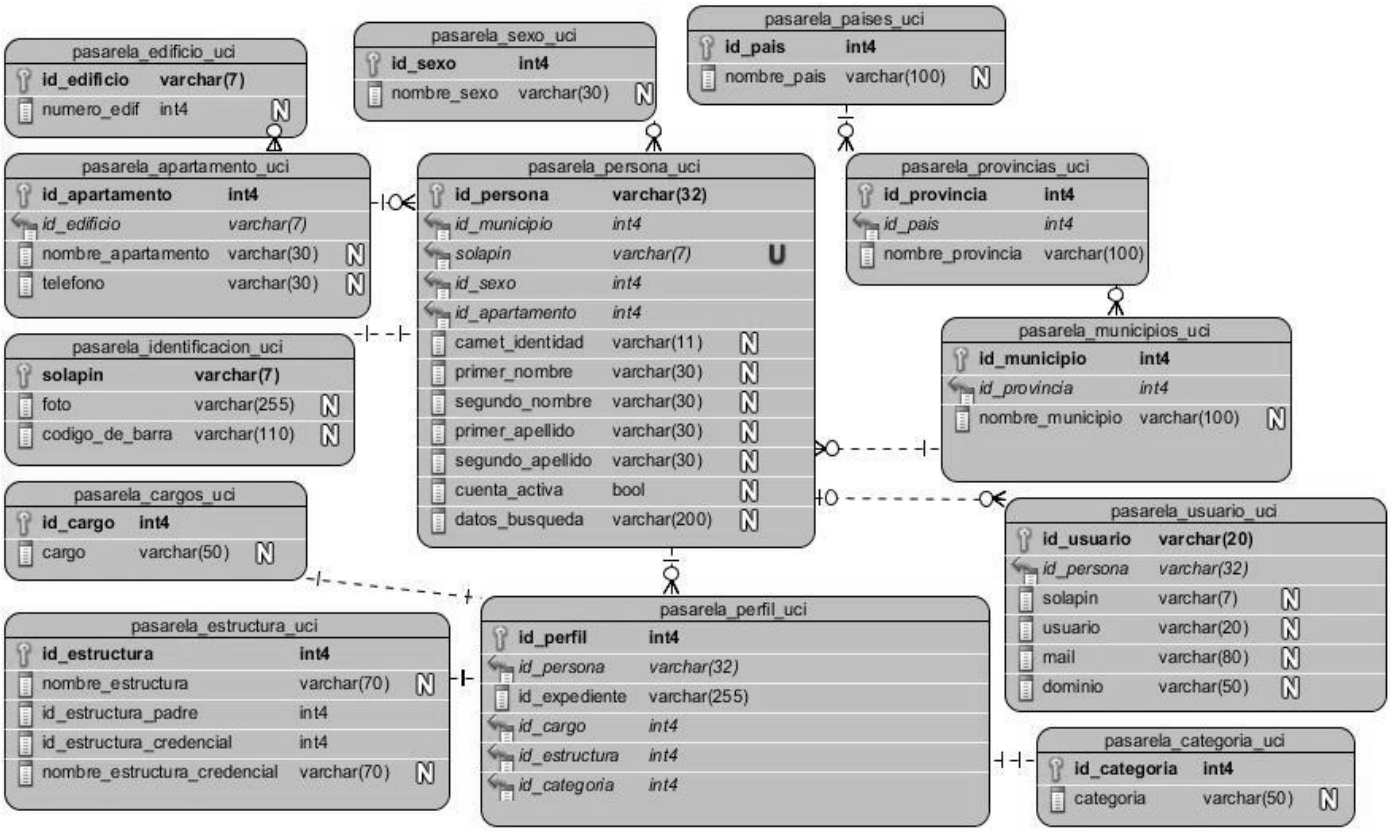


Figura 8: Modelo físico de datos

2.5 Conclusiones parciales

Luego del análisis realizado en el capítulo se arriban a las siguientes conclusiones:

La descripción del modelo de dominio permitió representar cómo se realiza el proceso de búsqueda de personas, teléfonos, aplicaciones y cumpleaños en la UCI, representados a través de las clases más importantes dentro del contexto del sistema, lo que permite desarrollar con mayor facilidad la solución propuesta.

El uso de técnicas para la obtención de requisitos proporcionó comprender, identificar y describir los requerimientos funcionales y no funcionales que deberá cumplir la solución.

La utilización de patrones de diseño y arquitectura permitieron diseñar una solución robusta, flexible y escalable. A partir del análisis, el diseño realizado y los artefactos generados quedan sentadas las bases para la implementación y validación de la solución propuesta.

Capítulo 3. Construcción y validación de la solución

Los artefactos generados a través del flujo de trabajo de análisis y diseño constituyen la base para las tareas de implementación y pruebas. Las actividades de implementación se contemplan en el desarrollo del sistema que se necesita y las pruebas que son un elemento vital para garantizar la calidad del software y representan una revisión final de las especificaciones, del diseño y la codificación. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado y constituyen una actividad en la cual un sistema y componentes son ejecutados bajo condiciones o requerimientos específicos y los resultados son observados y registrados para su posterior evaluación o corrección.

En el presente capítulo se describe la construcción de la solución propuesta teniendo en cuenta los estándares de codificación. Además, se definen y aplican las pruebas al producto, especificando las evaluaciones realizadas a la propuesta y los resultados obtenidos.

3.1 Diagrama de componentes

En un diagrama de componentes se representa un sistema de software dividido en componentes, estos a la vez son elementos del sistema que ofrecen un conjunto de funcionalidades o servicios; además, muestra su organización y dependencias representadas en una vista estática del sistema. En él se sitúan librerías, tablas, archivos, ejecutables y documentos que forman parte de la aplicación. En la Figura 9 se muestra el diagrama de componentes del sistema.

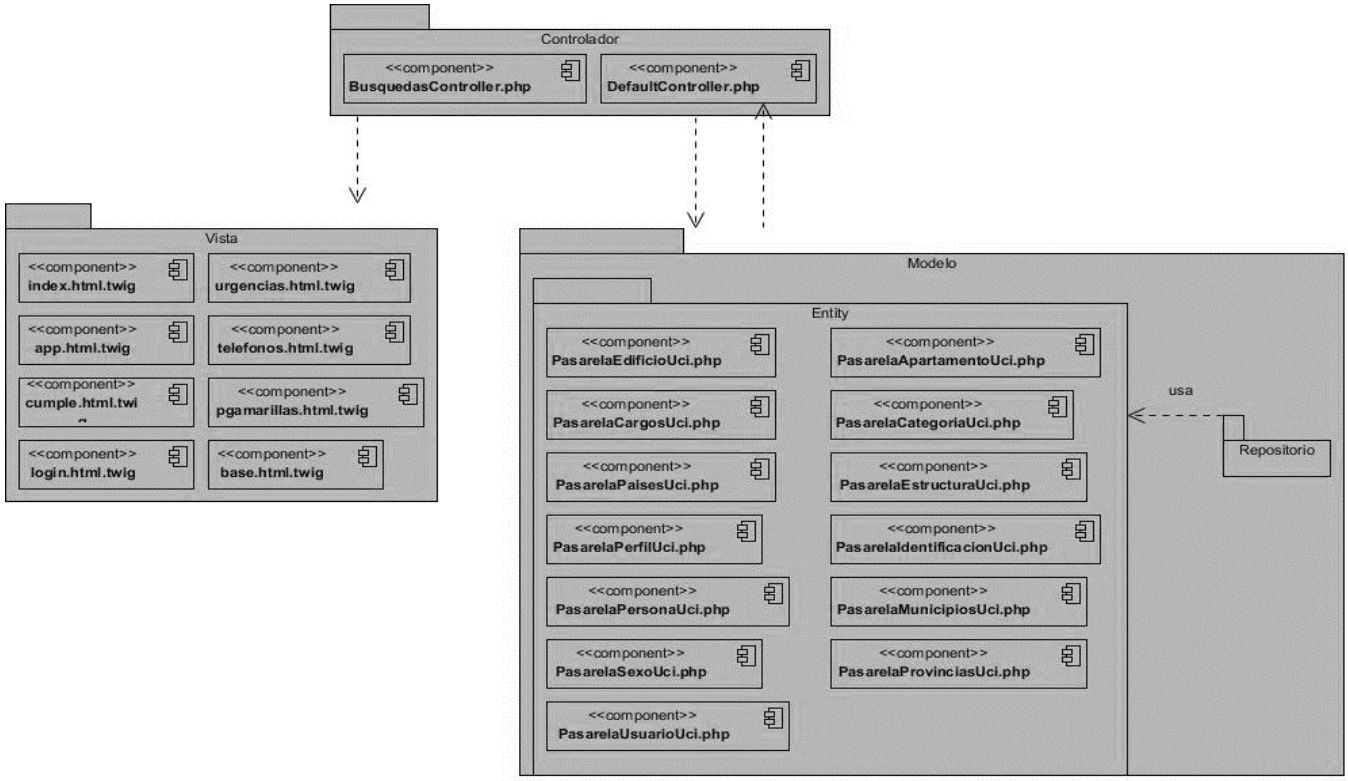


Figura 9: Diagrama de componentes

3.2 Estándares de codificación

Los estándares de codificación son un conjunto de reglas a seguir por los desarrolladores con el objetivo de establecer un orden y un formato común en el código fuente del software en desarrollo (55). Estos son de vital importancia, ya que permiten que el personal del proyecto pueda entender de forma fácil el código, garantizándose la organización y estructura del código fuente.

Symfony2 sigue estándares que son importantes en cualquier proyecto, principalmente si este involucra muchos desarrolladores. Esto asegura que el código sea de alta calidad, que contenga una cantidad baja de errores y sea fácil de mantener. Es importante que durante la codificación se consideren permanentemente los siguientes criterios de calidad (56):

- **Facilidad de Comunicación**: proporcionar al usuario entradas y salidas fácilmente asimilables.
- **Autodescripción**: proporcionar en el código, explicaciones sobre la implantación realizada.
- **Simplicidad**: la implantación realizada debe hacerse de la forma más comprensible posible.

3.2.1 Estándares de codificación usados en la solución

En el desarrollo de la solución los estándares utilizados fueron los siguientes:

Notación Camello *(Camel Case)*: el nombre se debe a que los identificadores recuerdan las jorobas de un camello. Este consiste en escribir los identificadores de variables o funciones con la primera letra en mayúscula y el resto en minúscula o viceversa en dependencia de la variante escogida. Un ejemplo de su utilización se evidencia a continuación en la Figura 10:

```
public function generalAction(Request $request ){
    $datos = json_decode($request->get('valor'));
    $valor=$request->get('valor');
    $em=$this->getDoctrine()->getManager();
    $personas=$em->getRepository('AppBundle:PasarelaPersonaUci')->findBy($valor);
    return new JsonResponse(array('personas' => $personas));
}
public function usuarioAction($user){
    $em=$this->getDoctrine()->getManager();
    $personas=$em->getRepository('AppBundle:PasarelaPersonaUci')->findBy($user);
    return $this->render(":default:index.html.twig", array("personnas"=>$personas));
}
```

Figura 10: Ejemplo de notación Camello

Notación Pascal: es una forma de definir las variables del entorno comenzando siempre en mayúscula la letra inicial de la variable, así como las siguientes palabras contenidas dentro de esta. Ejemplo de ello en el código de la aplicación se muestra a continuación en la Figura 11.

```
public function CumpleAction(){
      $fecha=date("Ymd");
      $fechap="";
      $em=$this->getDoctrine()->getManager();
      $personas=$em->getRepository('AppBundle:PasarelaPersonaUci')->findAll();
      $CumplenHoy=array();
      $cont=0;
      foreach($personas as $persona){
            for($i=0;$i<6;$i++)
               $fechap.=$persona->getCarnetIdentidad()[$i];
          if("{$fecha[4]}{$fecha[5]}"=="{$fechap[2]}{$fechap[3]}"){
            if("{$fecha[6]}{$fecha[7]}"=="{$fechap[4]}{$fechap[5]}"){
               $CumplenHoy [$cont++]=$persona;
            }
          }
      }
return $this->render("@Main/pages/cumple.html.twig",array("CumplenHoy "=>$CumplenHoy));
}
```

Figura 11: Ejemplo de notación Pascal

Otros estándares que se utilizaron son los siguientes, y se agrupan en las siguientes categorías:

Estructura:

- Utiliza un solo espacio después de cada delimitador (***coma).***
- Añade un solo espacio alrededor de los operadores *(**==, &&, ||**)*
- Utiliza una coma después de cada elemento del arreglo en un arreglo multilínea, incluso después del último.
- Añade una línea en blanco antes de las declaraciones ***return***, a menos que el valor devuelto solo sea dentro de un grupo de declaraciones ***(tal como una declaración if)***.
- Usa llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.

- Declara las propiedades de clase antes que los métodos.
- Declara primeramente los métodos ***public***, luego los ***protected*** y finalmente los ***private***.

Convenciones de nomenclatura:

- Utiliza mayúsculas intercaladas sin guiones bajos, en nombres de variable, función, método o argumentos.
- Usa guiones bajos para nombres de opción y nombres de parámetros.
- Utiliza espacios de nombres para todas las clases.
- Prefija las clases abstractas con *Abstract*.
- Sufija las interfaces con *Interface*.
- Sufija las características con *Trait*.
- Sufija las excepciones con *Exception*.
- Utiliza caracteres alfanuméricos y guiones bajos para los nombres de archive.
- Se hace uso de los comentarios utilizando las etiquetas /* */ para comentarios en varias líneas y // para comentarios de una única línea.
- Las estructuras de control deben cumplir un conjunto de normas para su correcto funcionamiento; las estructuras (if, while, for, entre otras.) y el primer paréntesis deben tener un espacio intermedio, para no confundirlas con la nomenclatura de las funciones; la llave de apertura ({) estará en la primera línea separada por un espacio y es recomendado usar las dos llaves ({ }) aunque el código lo permita; y las estructuras else y else if serán escritas en la línea siguiente de la llave de cierre anterior (}).
- Los valores dentro de un arreglo deben ser separados por un espacio después de la coma. El operador => debe separarse por un espacio a ambos lados. Cuando la línea de declaración del arreglo supera los 80 caracteres, cada elemento se debe escribir en una única línea, indentándolo[25] una vez (usando dos espacios).

Documentación:

- Añade bloques *PHPDoc* a todas las clases, métodos y funciones.

[25]Se define como el formato del código con espacios separándolo de la izquierda para lograr legibilidad del código generado por el programador.

- Omite la etiqueta *@return* si el método no devuelve nada.
- Las anotaciones *@package* y *@subpackage* no se utilizan.

A continuación, se muestra un ejemplo seleccionado del código donde se evidencian dichos estándares (ver Figura 12):

```
Class BusquedasController extends Controller
{
    /**
     * @Route("/devolver-busqueda", name="test")
     */
    public function generalAction(Request $request )
    {
        $datos = json_decode($request->get('valor'));
        $valor = $request->get('valor');
        $em = $this->getDoctrine()->getManager();
        $personas = $em->getRepository('AppBundle:PasarelaPersonaUci')->findBy($valor);
     return $this->render(":default:index.html.twig", array("personnas"=>$personas));
    }
}

/**
     * @Route("/acceso", name="acceso")
     */
    public function accesoAction(Request $request)
    {
        if ($request->attributes->has(SecurityContext::AUTHENTICATION_ERROR)) {
            $error = $request->attributes->has(SecurityContext::AUTHENTICATION_ERROR);
        } else {
            $error=$request->getSession()>get(SecurityContext::AUTHENTICATION_ERROR);
        }
        return $this->render(':seguridad:login.html.twig', array(
            'last_username' => $request->getSession(SecurityContext::LAST_USERNAME),
            'error' => $error
        ));
    }
```

Figura 12: Ejemplo de estándares de codificación

3.3 Validación de los requisitos del sistema

Para continuar el desarrollo del software a partir de los requisitos modelados y representados se hace necesaria la comprobación de su validez. Para ello se realizan actividades que generalmente se ejecutan una vez obtenida una primera versión de la documentación de requisitos; dicha actividad tiene como entrada el documento de requisitos, los artefactos relacionados y como salida se obtiene el documento de no conformidades y la opinión del cliente. Su objetivo es verificar todos los requerimientos que aparecen en el documento especificado, para asegurarse que representan una descripción aceptable del sistema que se debe implementar. Esto implica verificar que los requerimientos sean consistentes y que satisfagan las necesidades del cliente (57).

La validación de los requisitos en el Sistema Directorio v2.0 de la Universidad de las Ciencias Informáticas, se llevó a cabo mediante los métodos: revisión de requisitos, prototipado y generación de casos de pruebas.

3.3.1 Revisión de requisitos

La revisión de requisitos se convirtió en una de las tareas más importantes para llevar a cabo la validación de los mismos, mediante las reuniones y entrevistas planificadas con el grupo de desarrollo, se dieron a conocer los requerimientos del sistema y se anotaron los problemas detectados con la finalidad de mejorar el producto. En la Tabla 7 quedaron reflejados varios de estos errores. En el Anexo 7 se encuentra las preguntas que posibiltan la consistencia de cada requisito donde sus respuestas validan si el requisito se aprueba o no.

Tabla 7: Errores detectados en los requisitos funcionales

No.	Defectos detectados	Acciones recomendables
RFD1	Error de redacción, que lleva a ambigüedad	Modificar el texto del requisito de una forma más entendible. Especificar qué tipo de comunicación se realiza. Ejemplo: *"Notificar cumpleaños por correo electrónico".*
RFD47	Error de concepto	No se permiten agregar requisitos generales, como *Gestionar Aplicaciones,* la metodología DAC define que los requisitos tienen que describirse específicos. Ejemplo: *-Añadir aplicaciones* *-Modificar aplicaciones* *-Eliminar aplicaciones* *-Mostrar aplicaciones*
RFD11	Error de redacción	Debe especificar mejor la vía por la que se realizará la comunicación

3.3.2 Prototipado

El prototipado fue otra técnica utilizada, con el objetivo de dar a los usuarios interesados en el producto una imagen de lo que sería el futuro sistema a partir de los requisitos recogidos en la especificación, dando una versión del producto final. Su principal propósito fue obtener y validar los requerimientos esenciales, manteniendo abiertas, las opciones de implementación. Estos prototipos fueron evaluados por el equipo de trabajo y usuarios finales para comprobar que se estaba construyendo el sistema correcto, que la aplicación producía las salidas correctas y que cumplía con las necesidades y requisitos del cliente. En las reuniones realizadas se identificaron diferentes problemas que fueron corregidos posteriormente por el equipo de trabajo, logrando la completitud y validez de los mismos.

3.3.3 Generación de casos de pruebas

Este método tuvo como objetivo comprobar la verificabilidad de los requisitos funcionales. Para ello se construyeron 50 casos de pruebas, lo que permitio comprobar si las funcionalidades cumplían con las exigencias descritas. Esto permitió identificar los requisitos que debían ser modificados, eliminados y cuáles podían ser los más complejos en el desarrollo de la solución.

Después de realizados los métodos para la validación de los requisitos funcionales del sistema, de hacer varias reuniones con el equipo de trabajo y usuarios beneficiados con el producto, se corrigieron los errores antes mencionados quedando 100% validados y listos para ser usados en el desarrollo de la solución.

3.4 Pruebas

Una de las partes más importantes en el desarrollo del software son las pruebas, según Roger S. Pressman, son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Estas permiten validar y verificar el software, entendiendo como validación el proceso externo del equipo de desarrollo, que determina si el software satisface los requisitos y verificación, como el proceso interno que determina si los productos de una fase satisfacen las condiciones de dicha fase (58).

En la Tabla 8 se muestran las estrategias de pruebas diseñadas para aplicar al Directorio UCI versión 2.0.

Tabla 8: Estrategia de pruebas de software

Nombre de la prueba	Método	Técnica
Unidad	Caja Blanca	Camino básico
Aceptación	Caja Negra	Alfa
Sistema (carga y *estrés*)	Caja Negra	Automática
Sistema (funcionales)	Caja Negra	Partición de equivalencia

3.4.1 Pruebas de unidad

Las pruebas de unidad se centran en la verificación de los elementos más pequeños del software que se puedan probar examinando las estructuras de datos locales. Su objetivo es asegurar que las funcionalidades mantienen su integridad durante los pasos de ejecución de los algoritmos. Para la prueba se utiliza el método de caja blanca o estructural que se basa en un minucioso examen de los detalles procedimentales del código a evaluar. La técnica seleccionada en la investigación fue la del camino básico, que se basa en obtener una medida de la complejidad del diseño procedimental de un programa. La medida de complejidad del procedimiento es la complejidad ciclomática y representa un límite inferior para el número de casos de pruebas que se deben realizar para asegurar que se ejecuta cada camino del programa (58).

Tabla 9: Caso de prueba unitaria

Caso de prueba unitaria al método *cumpleAction*().

Programador(Estudiante): José Carlos Peñalver Peñalver

Código al que se le aplica:

```
public function cumpleAction()                                              1
{
    $fecha=date("Ymd");
    $fechap="";
    $em=$this->getDoctrine()->getManager();                                 2
    $personas=$em->getRepository('AppBundle:PasarelaPersonaUci')->findAll();
    $cumplenhoy=array();
    $cont=0;
 foreach($personas as $persona){                                            3
 for($i=0;$i<6;$i++)                                                        4
                $fechap.=$persona->getCarnetIdentidad()[$i];
 if("{$fecha[4]}{$fecha[5]}"=="{$fechap[2]}{$fechap[3]}"){                  5
  if("{$fecha[6]}{$fecha[7]}"=="{$fechap[4]}{$fechap[5]}"){
                $cumplenhoy[$cont++]=$persona;}
            }                                                               6
        }
   return $this->render("@Main/pages/cumple.html.twig",                     7
array("cumplenhoy"=>$cumplenhoy));
}                                                                           8
```

Figura 13: Ejemplo del código utilizado para calcular la complejidad ciclomática

Para el cálculo de la complejidad ciclomática existen tres fórmulas diferentes, se utilizan a continuación.

Fórmula 1:	**Fórmula 2:**	**Fórmula 3:**
V (G)=(A - N) +2	V (G)=P+ 1	V (G) = R
V (G)=9-8+2=3	V (G)=2+ 1=3	V (G) = 3

A: Cantidad de Aristas **N:** Cantidad de Nodos

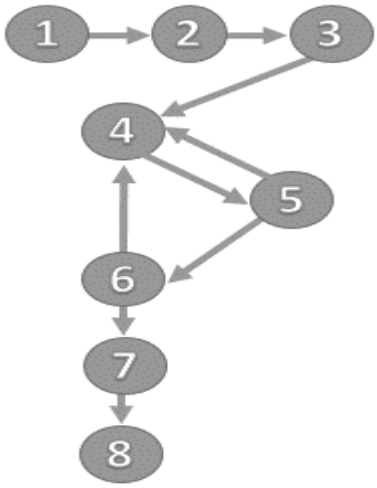
P: Número de nodos predicado en el grafo

R: Cantidad de regiones en el grafo

Caminos independientes:

1. **1.** 1→2→3→4→5→6→7→8→9
2. **2.** 1→2→3→4→5→4
3. **3.** 1→2→3→4→5→6→4

Representación del Grafo:



Prueba de Camino básico

Una vez calculada la complejidad ciclomática se define como límite superior 3, lo que indica que hay que realizarle al código tres pruebas, para garantizar que este se ejecute completamente al menos una vez. La funcionalidad cumpleaños del día posee poco riesgo debido a que el resultado arrojado por la métrica pertenece al intervalo entre 1-10.

3.4.2 Pruebas de aceptación

Las pruebas de aceptación son básicamente pruebas funcionales sobre el sistema completo, ya que tienen como objetivo obtener la aceptación final del cliente antes de la entrega del producto para su utilización. Su ejecución es facultativa del cliente, y en el caso de que no se realicen explícitamente, se dan por incluidas dentro de las pruebas del sistema. La ejecución de las pruebas de aceptación requiere un entorno de pruebas que represente el entorno de producción (58). Para el desarrollo de las pruebas se utiliza el método de caja negra que se centran en las funcionalidades que se espera obtener de un sistema, es decir, intentan encontrar casos en que el sistema no atiende alguna especificación. En estas pruebas el probador se limita a suministrarle datos al sistema y estudiar las salidas, sin preocuparse de lo que pueda estar realizando el sistema por dentro. La técnica alfa fue la escogida donde el cliente hace las pruebas al sistema en un ambiente controlado con el desarrollador como observador del usuario, registrando los errores y los

problemas de uso. El desarrollo de la prueba fue satisfactorio, quedando como constancia el acta de aceptación del cliente accesible en el Anexo 7.

3.4.3 Pruebas de sistema (carga y estrés)

Para la realización de las pruebas de carga y estrés se define la utilización de la herramienta *Jmeter*[26], la misma es una herramienta desarrollada en el lenguaje Java por el proyecto *Apache Software Foundation*. Esta aplicación brinda la posibilidad de realizar pruebas de rendimiento a través de la construcción de peticiones HTTP con cantidad de usuarios concurrentes variables, lo que posibilita obtener un resumen de los niveles de estrés del sistema y los límites de trabajo del mismo en condiciones extremas. El requerimiento no funcional que conforma el basamento de las pruebas realizadas es el siguiente:

RNFD5. El sistema debe responder en un tiempo menor a los 30 segundos.

Las pruebas de rendimiento son realizadas luego de que el sistema está completamente integrado, con el objetivo de observar el comportamiento del sistema bajo una cantidad de peticiones esperadas y comprobar que cumpla determinadas tareas en condiciones particulares de trabajo. Se encargan de demostrar que el sistema satisface sus requerimientos y monitorizar los comportamientos en cuanto a tiempo de respuesta de la petición y otros componentes que se vean afectados por la prueba.

Esta prueba fue realizada para 50 y 100 usuarios conectados en una 1ra iteración. Para ver los resultados de los 100 usuarios conectados consultar el Anexo 8.

A continuación, se muestran los resultados obtenidos para la 1ra iteración (ver Tabla 10):

Para una mejor comprensión de la tabla, a continuación, se detallan cada uno de los parámetros usados:

- **Muestras #:** indica la cantidad de usuarios haciendo peticiones de manera concurrente.
- **Media:** indica el máximo de tiempo de ejecución invertido para una petición.
- **Mediana:** significa que el 50% de las peticiones realizadas tardaron menos del valor reflejado.
- **Min:** indica el mínimo de tiempo de ejecución invertido para una petición.
- **Max:** indica el máximo de tiempo de ejecución invertido para una petición.
- **% Error**: indica la relación entre el total de peticiones y el número de peticiones que originaron errores.

[26] Herramienta que se utiliza para realizar pruebas de rendimiento y resistencia, normalmente contra aplicaciones web. Permite realizar simulaciones de gran carga en el servidor, red o aplicación para comprobar su "fuerza" y para analizar el rendimiento ante diferentes tipos de sobrecarga. Posee una interfaz gráfica que permite la interacción del usuario de una forma muy intuitiva.

- **Rendimiento:** hace referencia al número de peticiones que el servidor puede procesar en un segundo.
- **Kb/sec:** rendimiento medido en Kilobytes por segundo

Tabla 10: Resultados de rendimiento y resistencia para 50 usuarios

Etiqueta	# Muestras	Media	Mediana	Mín	Máx	% Error	Rendimiento	Kb/sec
30 /directori...	50	155	6	3	835	0,00%	19,3/min	19,2
34 /directori...	50	333	9	6	1163	0,00%	19,3/min	89,1
27 /directori...	50	333	12	7	1055	0,00%	19,4/min	110,2
35 /directori...	50	109	8	4	1107	0,00%	19,5/min	49,1
41 /directori...	50	200	5	3	1248	0,00%	19,6/min	,4
40 /directori...	50	152	5	2	828	0,00%	19,8/min	,4
39 /directori...	50	22	3	2	388	0,00%	19,9/min	,3
38 /directori...	50	94	4	2	1338	0,00%	19,9/min	,5
47 /directori...	50	197	4	2	1708	0,00%	20,0/min	1,4
43 /directori...	50	62	5	2	942	0,00%	20,0/min	,5
48 /directori...	50	309	5	2	1085	0,00%	20,0/min	,3
37 /directori...	50	46	8	2	658	0,00%	20,1/min	,6
46 /directori...	50	25	4	2	429	0,00%	20,2/min	,3
45 /directori...	50	17	4	2	276	0,00%	20,2/min	,3
42 /directori...	50	84	5	2	645	0,00%	20,2/min	1,1
44 /directori...	50	18	3	2	192	0,00%	20,3/min	,3
49 /directori...	50	19	3	2	246	0,00%	20,3/min	,3
51 /directori...	50	347	6	3	1301	0,00%	20,3/min	18,9
50 /directori...	50	74371	64098	32571	106510	0,00%	16,7/min	7,1
52 /directori...	50	28085	29194	2873	40396	0,00%	19,3/min	10,9
53 /directori...	50	26270	27491	2884	36804	0,00%	19,3/min	8,3
54 /directori...	50	26389	28162	2921	36495	0,00%	17,2/min	9,2
55 /directori...	50	23613	25423	3402	38814	0,00%	15,5/min	6,6
56 /directori...	50	27157	30264	2783	42140	0,00%	13,4/min	6,7
58 /directori...	50	7	5	2	37	0,00%	13,5/min	4,0
57 /directori...	50	21840	23847	2336	38969	0,00%	13,1/min	5,6
Total	2900	5732	20	0	138798	12,10%	7,8/sec	207,1

De manera general, en la Tabla 10 se puede observar que las peticiones son ejecutadas en tiempos inferiores a 30 segundos en la mayoría de los casos cumpliendo así con el requisito no funcional de eficiencia **RNFD5**. Para todas las muestras de usuarios la ocurrencia de errores se mantiene en 0.0%, lo que quiere decir que las peticiones hechas se ejecutan satisfactoriamente.

3.4.4 Pruebas de sistema (funcionales)

Las pruebas funcionales tienen por objetivo probar que los sistemas desarrollados cumplen con las funciones específicas para los que han sido creados. Se utiliza el método de caja negra para evaluar

funcionalmente la solución, donde los probadores se enfocan en el funcionamiento de la interfaz del sistema a partir del estudio de sus entradas y salidas (58).

Para confeccionar los casos de prueba de caja negra existen distintas técnicas entre la que se encuentra la de partición de equivalencia, que se basa en identificar las particiones para un sistema o componente.

Por cada requerimiento funcional del sistema se generó un caso de prueba donde se recogieron los datos necesarios para probarlo. Hay que tener en cuenta que en el diseño de caso de prueba **V** indica Válido, **I** Inválido y **NA** es irrelevante (no es necesario proporcionar un valor del dato). Las variables: Buscador, Nombre en pizarra, Teléfono, Área; significan los valores de entrada de datos para los casos de pruebas en cada escenario; la respuesta del sistema indica su comportamiento y el flujo central, los pasos que se deben ejecutar para completar cada escenario. Para ejecutar la prueba el usuario podía estar o no autenticado en el sistema.

A continuación, se describe un ejemplo de un diseño de caso de prueba que se le realizó a la solución (Tabla 11, 12 y 13).

Tabla 11: Diseño de caso de prueba del requisito funcional Búsqueda avanzada de teléfono(s). Parte 1

Escenario	**Descripción**	**Buscador (teléfono, área, ubicación, nombre en pizarra)**	**Nombre en pizarra**	**Teléfono**	**Área**
EC 1.1 Buscar datos de forma correcta.	Mediante el escenario se busca un teléfono correctamente.	V	V	V	V
		Docente 3	Laboratorio	2540	Dirección de Informatización
		V	NA	NA	NA
		2440			
EC 1.2 Buscar datos incorrectos.	Mediante el escenario se busca un teléfono agregando datos incorrectos.	I	I	I	NA
		“@”	333	dddd	
EC 1.3 Buscar datos sin introducir criterios de búsquedas.	Mediante el escenario de información sin introducir datos.	NA	NA	NA	NA

Tabla 12: Diseño de caso de prueba del requisito funcional Búsqueda avanzada de teléfono(s). Parte 2

Respuesta del sistema	**Flujo central**
El sistema realiza la búsqueda y muestra la	1. El usuario puede o no estar autenticado en el Sistema Directorio UCI.

información solicitada con los datos: *Teléfono, Nombre en pizarra, Área, Ubicación.*	2.Una vez dentro, selecciona la opción "Teléfonos" y se muestra una interfaz con un buscador que permite buscar por: *Teléfono, Área, Ubicación, Nombre en pizarra.* Además de los filtros de búsqueda: *Nombre en pizarra, Teléfono y Área.* 3. El usuario introduce los criterios que desea y presiona el botón Buscar. 4. El Sistema muestra los datos en un listado con los datos: *Teléfono, Nombre en pizarra, Área, Ubicación.*
El sistema muestra un mensaje: *"No hay resultados de búsqueda."*	1. El usuario puede o no estar autenticado en el Sistema Directorio UCI. 2. Una vez dentro selecciona la opción "Teléfonos" y se muestra una interfaz con un buscador que permite buscar por: *Teléfono, Área, Ubicación, Nombre en pizarra.* Además de los filtros de búsqueda: *Nombre en pizarra, Teléfono y Área.* 3. El usuario introduce los criterios de búsqueda incorrectamente y presiona el botón Buscar. 4. El sistema muestra un mensaje: *"No hay resultados de búsqueda."*
El sistema muestra todas las áreas de la Universidad en un listado con los datos: *Teléfono, Nombre en pizarra, Área, Ubicación.*	1. El usuario puede o no estar autenticado en el Sistema Directorio UCI. 2. Una vez dentro selecciona la opción "Teléfonos" y se muestra una interfaz con un buscador que permite buscar por: *Teléfono, Área, Ubicación, Nombre en pizarra.* Además de los filtros de búsqueda: *Nombre en pizarra, Teléfono y Área.* 3. El usuario no introduce criterios de búsqueda y presiona el botón Buscar. 4. El sistema muestra todas las áreas de la Universidad en un listado con los datos: *Teléfono, Nombre en pizarra, Área, Ubicación.*

Tabla 13: Diseño de caso de prueba del requisito funcional Búsqueda avanzada de teléfono(s). Parte 3

Nombre de la variable	Tipo de datos	Descripción
Buscador	Campo de texto	Permite buscar una persona por los criterios: Teléfono, Área, Ubicación, Nombre en pizarra.
Nombre en pizarra	Campo de texto	Permite introducir el nombre del local.
Teléfono	Campo de texto	Permite introducir un número de teléfono.

Área	Campo de selección	Permite seleccionar un área de la Universidad.

3.5 Resultados de las pruebas aplicadas al sistema

Las pruebas se realizaron en tres iteraciones y se detectaron errores de interfaz, ambigüedad y validación. En la primera iteración se identificaron 28 no conformidades para un total de 49 requisitos revisados, en la segunda 7 no conformidades de 54 requerimientos y en la última no se encontraron errores, cumpliendo con los requisitos funcionales expuestos y se demostró la calidad de la solución implementada.

La Figura 14 muestra los resultados de las pruebas realizadas al sistema, agrupando la cantidad de no conformidades encontradas por iteraciones.

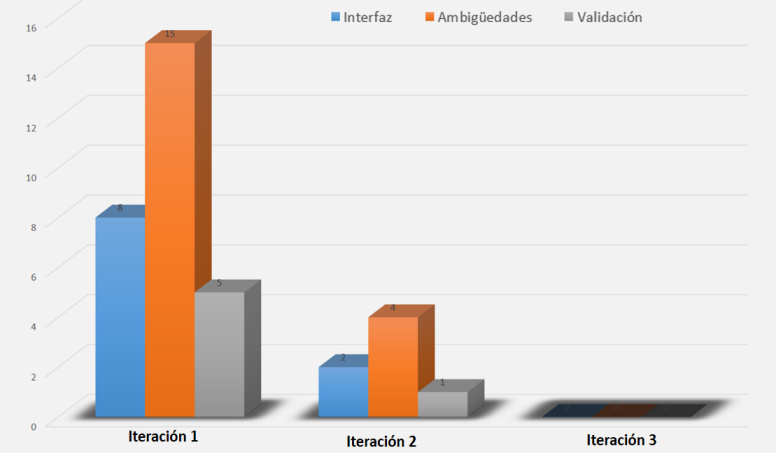


Figura 14: Resultados de las pruebas aplicadas al sistema.

3.6 Conclusiones parciales

Luego del análisis realizado en el capítulo se concluye:

La confección del diagrama de componentes ofreció una vista arquitectónica para ayudar al equipo de desarrollo en la implementación de la solución.

La definición de los estándares de codificación a tener en cuenta para la implementación de la solución, permitieron garantizar que el código posea calidad, menos errores y que pueda ser reutilizado por otros desarrolladores.

La realización de las pruebas de software al Sistema Directorio v2.0, facilitó identificar un conjunto de no conformidades, que fueron resueltas por el equipo de desarrollo, lo que posibilitó determinar y asegurar la calidad de la solución para que sea usada por la comunidad de la Universidad de las Ciencias Informáticas.

Conclusiones generales

Con la investigación realizada, el diseño y la implementación de una solución informática para la búsqueda de información de Personas, Teléfonos, Cumpleaños y Aplicaciones de la UCI se obtuvieron resultados que permiten arribar a las siguientes conclusiones:

- La revisión bibliográfica, el estudio de sistemas homólogos y el análisis de antecedentes encontrados en la literatura, permitieron adoptar una posición desde un enfoque informático e identificar metadatos[27] comunes y necesarios que debe contener el Sistema Directorio v2.0 de la UCI.
- El análisis del sistema que se encuentra en explotación permitió identificar y delimitar los problemas existentes, lo que posibilitó la definición de funcionalidades para la solución.
- Con la aplicación del proceso de desarrollo de software, se logró obtener y especificar los requisitos del sistema, así como describir los procesos relacionados a la búsqueda de información en la Universidad; lo que posibilitó comprender mejor los resultados a obtener en la solución y lo que sirvió de guía para la implementación del sistema.
- El desarrollo de la versión 2.0 del Sistema Directorio posibilitó la creación de una solución que se corresponde a las necesidades y exigencias de los usuarios de la UCI.
- Las pruebas realizadas al software permitieron determinar satisfactoriamente la calidad de la solución propuesta, se obtuvo una aplicación que cumple con los requisitos y que satisface las necesidades del cliente.

[27] Los metadatos son datos altamente estructurados que describen información, contenido, la calidad, la condición y otras características de los datos. Es "Información sobre información" o "datos sobre los datos".

Recomendaciones

Para el desarrollo de futuras versiones del Sistema Directorio v2.0 de la Universidad de las Ciencias Informáticas se recomienda:

- Implementación de un campo autogestionable que permita al usuario gestionar una localización alternativa.
- Para una nueva versión de la solución, el tiempo de respuesta sea menor a 15 segundos, mediante la optimización de los algoritmos de búsqueda.
- Establecer seguridad al API REST que se implementó en esta versión.

Bibliografía referenciada

1. Mena Aguilar, Adriana, Íncer Solís , Elia María y Acosta Sandoval, Silvia Elena. *Propuesta teórica-metodológica para evaluar los servicios de los archivos a partir de un estudio de usuario. Unidad de análisis: Los archivos municipales.* Costa Rica : Universidad de Costa Rica, Facultad de Ciencias Sociales., 2006.

2. Arvai, Peter; Halácsy , Péter; Adam Somlai-Fischer ;. Prezi Inc. *Manual de búsqueda documental y práctica bibliográfica.* [En línea] 2016. [Citado el: 10 de Octubre de 2015.] https://prezi.com/vb-b3rziyx1o/introduccion/.

3. Tiposde.org Portal educativo. *Tipos de sistemas de información.* [En línea] TiposDe.Org, 2012. [Citado el: 19 de Octubre de 2015.] http://www.tiposde.org/informatica/89-tipos-de-sistemas-de-informacion/#ixzz3zyzTNEiK.

4. EcuRed.cu. EcuRed. *Sistemas de Información.* [En línea] [Citado el: 19 de Octubre de 2015.] www.ecured.cu/Sistema_de_Informacion.

5. Duany Dangel, Armando. Econlink. *Los sistemas de información en las organizaciones.* [En línea] Centro de estudios de Desarrollo Agrario, 2000-2015. [Citado el: 22 de Octubre de 2015.] http://www.econlink.com.ar/sistemas-informacion/definiciones.

6. Monzalvo Serrano, Luciana. *Sistema de información de mercado.* México : Universidad Autónoma del Estado de Hidalgo.

7. Márquez Rodríguez, Julio. *Uso de la tecnología como recurso para la enseñanza. Sistema de búsqueda en Internet.* México : Universidad Autónoma del Estado de Hidalgo.

8. Padró del Pico, Manuel Alejandro. *Aplicación informática para la gestión de un servicio de directorio de personas.* Villa Clara, Cuba : Universidad central "Marta Abreu" de las Villas, 2013.

9. Creative Commons. CCM Benchmark. *Introducción a las Bases de Datos. ¿Qué es una base de datos ?* [En línea] Marzo de 2016. [Citado el: 11 de Noviembre de 2015.] http://es.ccm.net/contents/66-introduccion-bases-de-datos.

10. Blog Historia de la Informática. *Escuela Técnica Superior de Ingeniería Informática.* [En línea] Universidad Politécnica de Valencia, 4 de Enero de 2011. [Citado el: 16 de Noviembre de 2015.] http://histinf.blogs.upv.es/2011/01/04/historia-de-las-bases-de-datos/.

11. Calzada Prada, Rafael. *Introducción al Servicio de Directorio. .* España : Escuela Politécnica Superior, Universidad Carlos III de Madrid, 2002.

12. COMPUTERWORLD. Oracle Internet Directory, el directorio de base de datos para Internet de Oracle . [En línea] 1999-2016. [Citado el: 22 de Noviembre de 2015.] http://www.computerworld.es/economia-digital/oracle-internet-directory-el-directorio-de-base-de-datos-para-internet-de-oracle.

13. Directorio. [En línea] Universidad de Navarra, España. [Citado el: 26 de Noviembre de 2015.] http://www.unav.es/enlacesdirectos/.

14. Directories. [En línea] Universidad de Harvard. [Citado el: 26 de Noviembre de 2015.] http://www.harvard.edu/about-harvard/directories .

15. —. Directories Stanford University. [En línea] Universidad de STANFORD, 2013. [Citado el: 2016 de Febrero de 2015.] http://www.stanford.com/.

16. Directorio. [En línea] Universidad de Costa Rica, 2016. [Citado el: 6 de Febrero de 2016.] http://www.ucr.ac.cr/directorio/.

17. ETECSA. *Directorio telefónico.* [En línea] Cuba, 2012. [Citado el: 14 de Enero de 2016.] http://www.etecsa.cu/?page=directorio_telefonico.

18. Facultad de Ingeniería Industrial, Instituto Superior Politécnico, Jose Antonio Echeverría. *Directorio Telefónico.* [En línea] Cujae. Higher Polytechnic Institute José Antonio Echeverría, 2 de Marzo de 2012. [Citado el: 6 de Febrero de 2016.] http://cujae.edu.cu/en/industrial/directorio-telefónico.

19. Universidad de las Ciencias Informáticas, Dirección de Informatización. Portal de la Universidad de las Ciencias Informáticas. [En línea] Cuba, 2012. [Citado el: 15 de Febrero de 2016.] http://www.uci.cu/lista-de-directorios.

20. Universidad de las Ciencias Informáticas, Dirección de Informatización. Directorio UCI. [En línea] Cuba, 4 de Mayo de 2016. http://directorio.uci.cu/.

21. Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A. *Metodologías tradicionales vs. Metodologías agilés.* Ecuador : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

22. Sánchez Méndez, Alelí. *Proceso de desarrollo de software DAC, una alternativa para desarrollar ágil con CMMI nivel 2.* Perú : Congreso Internacional de Ingeniería de Software. CIIS, Universidad La Salle, Arequipa, 2013.

23. Nacho, Pacheco. *Manual de Symfony2, Release 2.0.1.* 2011.

24. ClubEnsayos. *Introducción a la Computación. Historias de los lenguajes de computación.* [En línea] ClubEnsayos.com, 17 de Mayo de 2014. [Citado el: 5 de Marzo de 2016.] https://www.clubensayos.com/Tecnolog%C3%ADa/Introducci%C3%B3n-A-La-Computaci%C3%B3n/1715803.html.

25. Larman, Craig. *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.* New York : PRENTICE HALL, 1999. ISBN 970-17-0261-1.

26. Lamarca Lapuente, María Jesús. *Hipertexto: El nuevo concepto de documento en la cultura de la imagen.* Madrid : Tesis doctoral. Universidad Complutense de Madrid, 2013.

27. Wium Lie, Håkon y Bos, Bert. *Cascading Style Sheets, designing for the Web. Segunda Edición.* s.l. : New York, 1999. ISBN 0-201-59625-3. .

28. Mehdi Achour, Daniel Beckham, Friedhelm Betz, Victor Boivie, Jesus M. Castagnetto, Nicolas Chaillan, Ron Chmara, Sean Coates, James Cox, Vincent Gevers, Sara Golemon, Zak Greant, Szabolcs Heilig, Oliver Hinckel, Hartmut Holzgraefe, Etienne Kneuss, Rasmus. PHP. *Manual de PHP.* [En línea] My PHP.net, 2001-2016. [Citado el: 30 de Marzo de 2016.] https://secure.php.net/manual/es/preface.php.

29. LIBROSWEB. *Introducción al JavaScript.* [En línea] LibrosWeb.es, 2006-2016. [Citado el: 30 de Marzo de 2016.] http://librosweb.es/libro/javascript/.

30. PostgreSQL-es. *Portal en español sobre PostgreSQL.* [En línea] 2 de Octubre de 2010. [Citado el: 28 de Marzo de 2016.] http://www.postgresql.org.es/sobre_postgresql.

31. Mastermagazine. [En línea] 2012. [Citado el: 30 de Marzo de 2016.] http://mastermagazine.info/termino/5234.php.

32. Visual Paradigm. *ULM tool, business process modeler and database designer for software development team.* [En línea] 2010. [Citado el: 15 de Marzo de 2016.] http://www.visual- paradigm.com..

33. Pencil Project. MariFont 2.0. *Carcaterísticas de Pencil Project.* [En línea] Evolus, 2 de Abril de 2012. [Citado el: 15 de Marzo de 2016.] http://www.merifont.com/pencil-project-crea-tus-prototipos-web/2012/04/02/.

34. NetBeans. NetBeans . *The NetBeans Platform.* [En línea] Oracle Corporation, 2016. [Citado el: 30 de Marzo de 2016.] https://netbeans.org/downloads/.

35. PgAdmin PostgreSQL Tools. PgAdmin. [En línea] [Citado el: 24 de Marzo de 2016.] http://www.pgadmin.org/translation/.

36. Martínez, Rafael. PostgreSQL-es. *Portal en español sobre PostgreSQL.* [En línea] 1 de Septiembre de 2015. [Citado el: 5 de Marzo de 2016.] http://www.postgresql.org.es.

37. Hopkins , Ben. Datamation. *Open Source BI: Pentaho 6.0 combina datos de ambientes virtuales y físicos.* [En línea] 5 de Octubre de 2015. [Citado el: 1 de Abril de 2016.] http://www.datamation.com.ar/open-source-bi-pentaho-6-0-combina-datos-de-ambientes-virtuales-y-fisicos-7044.

38. Adrian Jiménez Blanes. *Estudio del servidor web: Nginx.* [En línea] 23 de Enero de 2015. [Citado el: 17 de Abril de 2016.] http://adrianjimenezb.wordpress.com/2015/01/23/estudio-de-servidor-web-ngx.

39. jQuery. *Introducción a jQuery.* [En línea] 2010. [Citado el: 5 de 4 de 2016.] http//:www.desarrolloweb.com/articulos/introduccion-jquery.html.

40. Rodríguez, Txema. GENBETA:dev. [En línea] 12 de Junio de 2012. [Citado el: 16 de Febrero de 2015.] http://www.genbetadev.com/frameworks/bootstrap.

41. Lázaro, Pablo. ¿Qué es AngularJS? Una breve introducción. [En línea] 20 de Mayo de 2013. [Citado el: 16 de Febrero de 2016.] http://pablolazarodev.blogspot.com/2013/05/que-es-angularjs-una-breve-introduccion.html.

42. Larman, Craig. *UML y Patrones. Una introducción al análisis y diseño orientado a objeto y al proceso unificado. Segunda edición.* s.l. : Universisas de Murcia, 2003.

43. Metodología de Gestión de Requerimientos. *Técnicas para identificar requerimientos funcionales y no funcionales.* [En línea] [Citado el: 23 de Abril de 2016.] https://sites.google.com/site/metodologiareq/system/app/pages/sitemap/hierarchy.

44. Guerra, César Arturo. SG Buzz. *Obtención de Requerimientos. Técnicas y Estrategia.* [En línea] 2007. [Citado el: 18 de Abril de 2016.] http://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

45. CeO2 Software. *Sistemas Clientes Servidor.* [En línea] 2010-2016. [Citado el: 30 de Abril de 2016.] http://www.co2soft.com.ar/sistemas-cliente-servidor/.

46. Aplicaciones Web. *Arquitectura cliente-servidor.* [En línea] [Citado el: 19 de Abril de 2016.] https://sites.google.com/site/4appweb/tarea/2-4-arquitectura-cliente-servidor.

47. Álvarez, Miguel Angel. *Manual de CodeIgniter. Modelo - Vista - Controlador en CodeIgniter.* [En línea] 23 de Diciembre de 2009. http://www.desarrolloweb.com/articulos/modelo-vista-controlador-codeigniter.html.

48. *Cap 7. Symfony 1.4, la guía definitiva.*

49. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000. ISBN 74-7829-036-2.

50. Pressman, Roger S. *Ingeniería del Software, un enfoque práctico.* Madrid : Concepción Fernández, 2002. ISBN: 0-07-709677-0.

51. Craig, Larman. *UML y Patrones.* ISBN-84-205-3438-2.

52. *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web.* Guerrero, Carlos A., Suárez, Johanna M y Gu, Luz E. . 2013.

53. *Blaha, Michael.* Patterns of data modeling. Segunda Edición. : CRC Press Taylor & Francis Group, 2010. ISBN 1-4398-1989-0. .

54. *Diseño y explotación de almacenes de datos.* Trujillo, Juan Carlos, Mazón, José Norberto y Pardillo, Jesús. s.l. : Club Universitario. ISBN-978-84-9948-546-1.

55. MICROSOFT. *Revisiones de código y estándares de codificación.* [En línea] 2016. [Citado el: 30 de Abril de 2016.] https://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

56. Symfony en Español. *Estándares de codificación.* [En línea] 2013. [Citado el: 25 de Abril de 2016.]

57. *Master de Ingeniería de Software, Módulo II.Requisitos de Sistemas. Unidad 11. Validación de Requisitos.* s.l. : http://is.ls.fi.upm.es/docencia/masterTI/ARS/docs/Manual_M2C1U11.pdf.

58. Pressman, Roger S. . *Ingeniería de Software, un enfoque práctico. Séptima edición.* 2013. ISBN: 978-607-15-0314-5.

Bibliografía consultada

1. Mena Aguilar, Adriana, Íncer Solís , Elia María y Acosta Sandoval, Silvia Elena. *Propuesta teórica-metodológica para evaluar los servicios de los archivos a partir de un estudio de usuario. Unidad de análisis: Los archivos municipales.* Costa Rica : Universidad de Costa Rica, Facultad de Ciencias Sociales., 2006.

2. Arvai, Peter; Halácsy , Péter; Adam Somlai-Fischer ;. Prezi Inc. *Manual de búsqueda documental y práctica bibliográfica.* [En línea] 2016. [Citado el: 10 de Octubre de 2015.] https://prezi.com/vb-b3rziyx1o/introduccion/.

3. Márquez Rodríguez, Julio. *Uso de la tecnología como recurso para la enseñanza. Sistema de búsqueda en Internet.* México : Universidad Autónoma del Estado de Hidalgo.

4. Padró del Pico, Manuel Alejandro. *Aplicación informática para la gestión de un servicio de directorio de personas.* Villa Clara, Cuba : Universidad central "Marta Abreu" de las Villas, 2013.

5. Creative Commons. CCM Benchmark. *Introducción a las Bases de Datos. ¿Qué es una base de datos ?* [En línea] Marzo de 2016. [Citado el: 11 de Noviembre de 2015.] http://es.ccm.net/contents/66-introduccion-bases-de-datos.

6. Blog Historia de la Informática. *Escuela Técnica Superior de Ingeniería Informática.* [En línea] Universidad Politécnica de Valencia, 4 de Enero de 2011. [Citado el: 16 de Noviembre de 2015.] http://histinf.blogs.upv.es/2011/01/04/historia-de-las-bases-de-datos/.

7. Calzada Prada, Rafael. *Introducción al Servicio de Directorio. .* España : Escuela Politécnica Superior, Universidad Carlos III de Madrid, 2002.

8. Sánchez Méndez, Alelí. *Proceso de desarrollo de software DAC, una alternativa para desarrollar ágil con CMMI nivel 2.* Perú : Congreso Internacional de Ingeniería de Software. CIIS, Universidad La Salle, Arequipa, 2013.

9. Symfony en Español. *Estándares de codificación.* [En línea] 2013. [Citado el: 25 de Abril de 2016.].

10. Pressman, Roger S. . *Ingeniería de Software, un enfoque práctico. Séptima edición.* 2013. ISBN: 978-607-15-0314-5.

11. Larman, Craig. *UML y Patrones. Una introducción al análisis y diseño orientado a objeto y al proceso unificado. Segunda edición.* s.l. : Universisas de Murcia, 2003.

12. Metodología de Gestión de Requerimientos. *Técnicas para identificar requerimientos funcionales y no funcionales.* https://sites.google.com/site/metodologiareq/system/app/pages/sitemap/hierarchy.

13. Guerra, César Arturo. SG Buzz. *Obtención de Requerimientos. Técnicas y Estrategia.* [En línea] 2007. [Citado el: 18 de Abril de 2016.] http://sg.com.mx/revista/17/obtencion-requerimientos-tecnicas-y-estrategia.

14. *Cap 7. Symfony 1.4, la guía definitiva.*

15. Pressman, Roger S. *Ingeniería del Software, un enfoque práctico.* Madrid : Concepción Fernández, 2002. ISBN: 0-07-709677-0.

16. Craig, Larman. *UML y Patrones.* ISBN-84-205-3438-2.