
**Facultad de Ciencias y Tecnologías Computacionales**

# Título: Sistema de gestión de contactos para la Plataforma de Telecomunicaciones PLATEL.

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

**Autor(es):** Dayelin Alayo Roll

**Tutores:** MSc. Yaneisis Pérez Heredia

Ing. Maday Segreo García

**Cotutor:** Ing. Ronney Zulueta Esquivel

La Habana, noviembre de 2023

"Año 65 de la Revolución"

# DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo Dayelin Alayo Roll, soy el autor del Trabajo de Diploma Sistema de gestión de contactos para la Plataforma de Telecomunicaciones PLATEL y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

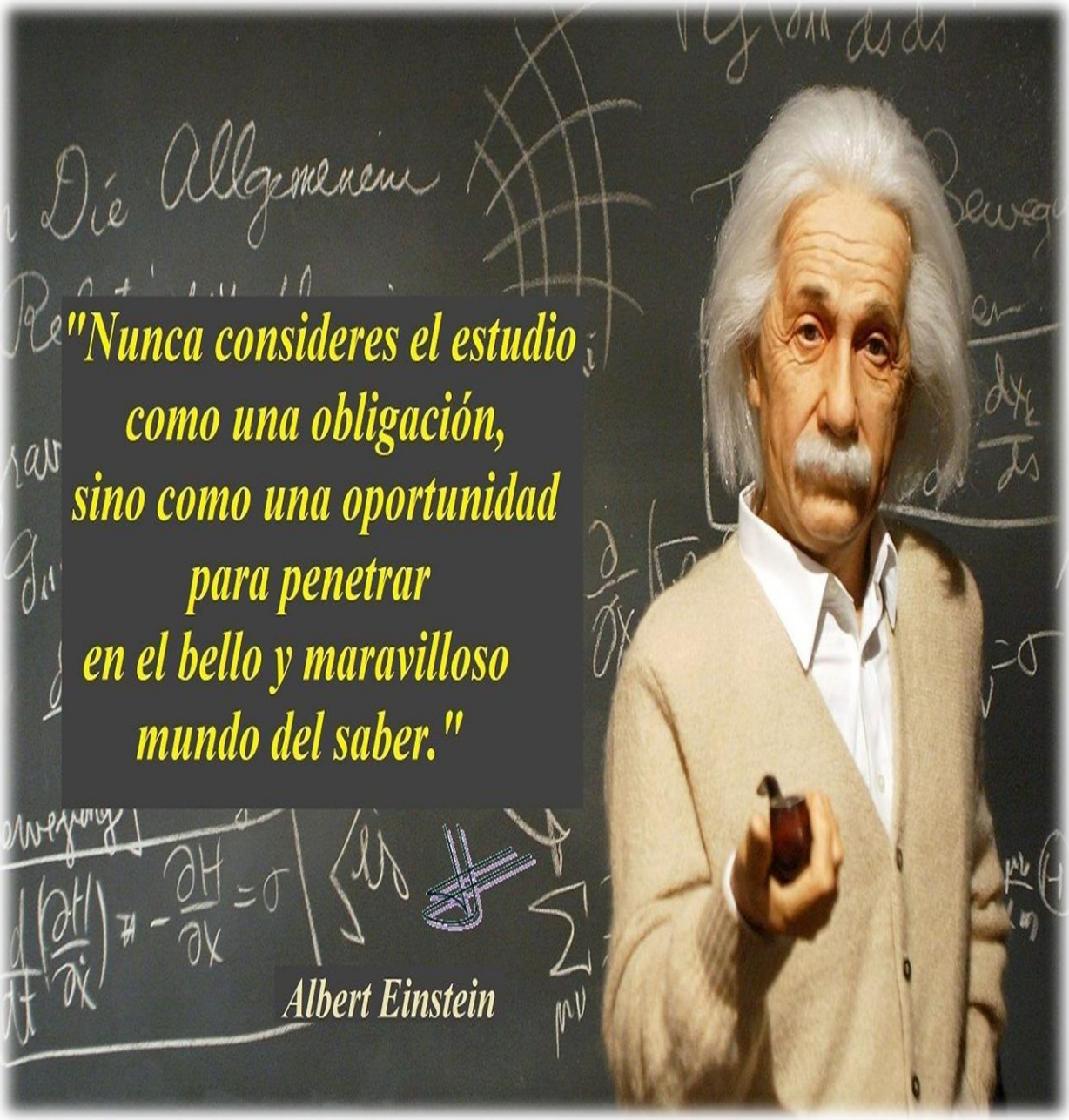
Y para que así conste, firmo la presente declaración de autoría en La Habana a los 15 días del mes de noviembre del año 2023.

_____

**Dayelin Alayo Roll**

**Autor**

| | | |
|---|---|---|
| _____ | _____ | _____ |
| MSc. Yaneisis Pérez Heredia | Ing. Maday Segreo García | Ing. Ronney Zulueta Esquivel |
| **Tutor** | **Tutor** | **Cotutor** |


"Nunca consideres el estudio
como una obligación,
sino como una oportunidad
para penetrar
en el bello y maravilloso
mundo del saber."
Albert Einstein

# DEDICATORIA

*Esta tesis va dedicada a mis padres y a mi familia en general ya que siempre han estado conmigo apoyándome en todos los momentos buenos y malos de mi vida.*

# AGRADECIMIENTOS

*Quiero expresar mi más sincero agradecimiento a todas las personas que han contribuido de manera significativa a la realización de esta tesis. Sin su apoyo, este logro no habría sido posible.*

*En primer lugar, agradecerles a mis padres por su amor incondicional, paciencia y constante aliento me han sostenido durante todo este proceso. Gracias por creer en mí y por ser mi fuente de inspiración.*

*A mis familiares quienes siempre han estado ahí para escucharme, motivarme y celebrar mis triunfos. Su apoyo ha sido fundamental. En especial a mi prima Yemila que me ha apoyado desde que comencé la carrera.*

*A mis amistades por las conversaciones, las risas compartidas y el apoyo moral en los momentos más desafiantes. Gracias por ser mi red de apoyo y por celebrar cada pequeño avance. principalmente a Laura, Geilis, Mardelis, Fernando, mis amistades de Santiago de Cuba.*

*A mis profesores y tutoras por su orientación, conocimientos compartidos y valiosos comentarios. Cada interacción ha enriquecido mi aprendizaje y mi perspectiva.*

*Muchas gracias a todos.*

# RESUMEN

Los sistemas de gestión de contactos son una herramienta que permiten a las empresas almacenar y organizar información sobre sus clientes, también pueden incluir detalles sobre las interacciones pasadas con los clientes, como llamadas telefónicas, correos electrónicos y reuniones. En este sentido la empresa Xetid opta por la creación de un centro de contacto para uno de sus proyectos, la plataforma de telecomunicaciones Platel, para gestionar todo lo referente a la información de los clientes.

La presente investigación se centra en el desarrollo de un componente para la administración de un centro de contactos para la plataforma de telecomunicaciones PLATEL. El objetivo principal es mejorar la gestión y operación de los contactos, brindando una solución integral y escalable. Se realizó un análisis exhaustivo de las necesidades y requisitos de los centros de contacto actuales, identificando las funcionalidades clave que deben ser implementadas. El desarrollo de la propuesta de solución fue guiado por la metodología de desarrollo de software AUP variación UCI de acuerdo con las políticas de informatización de la Universidad de las Ciencias Informáticas. Para la implementación de este se utilizó como marco de trabajo Django con su lenguaje de programación Python. Para validar que la solución cumpliera con los requisitos definidos por el cliente, se aplicaron pruebas unitarias, pruebas funcionales y pruebas de rendimiento.

**Palabras clave:** plataforma de telecomunicaciones Platel, sistemas de gestión de contactos

# *ABSTRACT*

Contact management systems are a tool that allow companies to store and organize information about their customers, they can also include details about past interactions with customers, such as phone calls, emails and meetings. In this sense, the company Xetid opts for the creation of a contact center for one of its projects, the Platel telecommunications platform, to manage everything related to customer information.

This research focuses on the development of a component for the administration of a contact center for the PLATEL telecommunications platform. The main objective is to improve the management and operation of contacts, providing a comprehensive and scalable solution. An exhaustive analysis of the needs and requirements of current contact centers was carried out, identifying the key functionalities that must be implemented. The development of the solution proposal was guided by the AUP UCI variation software development methodology in accordance with the computerization policies of the University of Computer Sciences. To implement this, Django with its Python programming language was used as the framework. To validate that the solution met the requirements defined by the client, unit tests, functional tests and performance tests were applied.



Keywords: Platel telecommunications platform, contact management systems

# ÍNDICE

# ÍNDICE DE TABLAS

# ÍNDICE DE FIGURAS

# INTRODUCCIÓN

El avance de la tecnología ha sido uno de los mayores impulsores del progreso humano en los últimos años. Desde la invención de la rueda y el fuego, hasta la creación de la electricidad y la informática, la tecnología ha transformado la forma en que se vive, trabaja y se relaciona. En el siglo XXI, los avances tecnológicos han sido más rápidos que nunca, y han dado lugar a una serie de innovaciones que han cambiado el mundo.

Los avances en la tecnología han permitido la creación de nuevos productos y servicios, así como la mejora de los procesos de producción y la eficiencia en la gestión empresarial. Estas tecnologías están transformando la manera en que las empresas operan, cómo los gobiernos prestan servicios públicos y cómo la gente se comunica

En este contexto, los sistemas de gestión de contactos surgieron como una herramienta para ayudar a las empresas a organizar sus listas de contactos, actualizarlas y permitir que todos los departamentos puedan acceder a ellas de forma remota (*Las diez mejores plataformas de software para la administración de contactos* 2023). Permiten a las empresas almacenar y organizar información sobre sus clientes, como nombres, direcciones, números de teléfono y correos electrónicos. También pueden incluir detalles sobre las interacciones pasadas con los clientes, como llamadas telefónicas, correos electrónicos y reuniones. El software también le permite organizar y filtrar contactos fácilmente para ayudarlo a encontrar subconjuntos de clientes específicos. También puede filtrar la lista de contactos por industria, cargo, relaciones con su empresa, etc. Además, puede administrar las comunicaciones utilizando el software de administración de contactos e integrarlo con otras herramientas como calendarios para programar reuniones y recibir notificaciones de participantes y horarios (Kumar 2021).

Es por eso que muchos sistemas de administración de contactos ya están integrados en otras soluciones prácticas de software comercial. Por ejemplo, el software de gestión de proyectos y el CRM (gestión de la relación con el cliente). Una plataforma CRM hace lo mismo que las herramientas de administración de contactos, pero con más funciones.

En la XETID unos de sus proyectos, Plataforma Integral de Telecomunicaciones PLATEL es una plataforma de comunicaciones que ofrece diversos servicios como voz, datos, video, mensajería unificada, seguridad, supervisión y notificaciones. Para ello, emplea protocolos estándares e interfaces flexibles, con la capacidad de interoperar con otros sistemas empresariales (XETID 2023).

Ha contribuido al desarrollo de empresas, gobiernos y entidades presupuestadas con el despliegue de aplicativos como Platel-PBX, Platel-Videoconferencia y Platel-Call Center. Platel-Videoconferencia ha sido desplegado en centros de datos de clientes en muchos casos con arquitecturas de despliegue y configuraciones diferentes para adaptarse a las características de las redes de los usuarios. El impacto ha sido muy favorable, ha facilitado el teletrabajo y trabajo a distancia de muchas organizaciones, también ahorrar tiempo, dinero y combustible, el uso de los servicios de PLATEL fue la alternativa más viable para mantener niveles de producción en tiempos de pandemia por lo que ha aumentado la demanda.

En la actualidad se brinda el servicio del uso de PLATEL desde la Nube y un aspecto que resulta deficiente es la gestión de los contactos, ya que la plataforma no cuenta con una solución que le permita administrar eficazmente los contactos, pues no disponen de datos de contactos organizados, esto trae como consecuencia la pérdida de los mismos. La gestión adecuada de los contactos permite a las empresas construir y mantener relaciones sólidas con los clientes. Al registrar detalles como nombres, correos electrónicos, números de teléfono y preferencias, puedes personalizar las interacciones y brindar un servicio excepcional. Esto fomenta la lealtad del cliente y aumenta las posibilidades de retención.

Con un aumento continuo de la preferencia de los clientes por los canales digitales para conectarse con las empresas es de suma importancia contar con un sistema de gestión de contactos en la Plataforma PLATEL.

Sobre lo antes expuesto se plantea como **problema a resolver:** ¿Cómo gestionar los contactos de la Plataforma de Telecomunicaciones PLATEL que permita almacenar y mejorar la información de los contactos?

Para dar solución al problema identificado se deriva como **objeto de estudio:** los sistemas de gestión de contactos, enmarcados en el **campo de acción:** el centro de contactos para la plataforma PLATEL.

Se define como **objetivo general** Desarrollar un sistema de gestión para la administración de un centro de contactos que garantice una mejor gestión de los mismos en la Plataforma de Telecomunicaciones Platel.

**Objetivos específicos:**

- Analizar los Sistemas de Gestión de Contactos existentes en el mundo.
- Definir las tecnologías y las herramientas a usar durante el desarrollo del componente.

- Modelar el sistema de Gestión de Contactos.
- Implementar el sistema de Gestión de Contactos.
- Validar la solución.

Para dar cumplimiento a los objetivos expuestos se proponen las siguientes **tareas de investigación:**

- Caracterizar los sistemas de gestión de contactos en Cuba y en el mundo.
- Conceptualizar los principales términos para profundizar en el conocimiento del objeto de estudio.
- Determinar las herramientas y tecnologías a emplear en el desarrollo del sistema.
- Seleccionar la metodología a emplear.
- Especificar los requisitos funcionales y no funcionales del sistema.
- Especificar la arquitectura de sistema y de datos.
- Obtener los artefactos generados a partir del uso de la metodología.
- Implementar el sistema.
- Realizar las pruebas de caja blanca.
- Realizar las pruebas de rendimiento.
- Realizar las pruebas funcionales.

Para el análisis y estudio del proyecto propuesto se emplearon los siguientes **métodos científicos**:

**Métodos teóricos:**

- **Histórico-Lógico:** este método permite seguir la evolución de los centros de contactos existentes en el mundo, desde su surgimiento y tendencias más actuales lo que servirá para profundizar el conocimiento sobre estos.
- **Analítico-sintético:** posibilita el análisis de las teorías, documentos y materiales, permitiendo así la extracción de los elementos más importantes sobre el proceso de gestión de contactos.
- **Modelación:** es utilizado para realizar una representación del proceso estudiado que sirva de guía en el desarrollo del sistema, y mediante este, identificar las características y relaciones fundamentales que den cumplimiento a los requisitos funcionales de la solución propuesta.

**Métodos empíricos:**

- **Entrevista:** se emplea en encuentros con el cliente para definir las funcionalidades del sistema de administración de contactos, identificando a la vez particularidades necesarias para su desarrollo.
- **Observación:** este método permite obtener conocimiento acerca del comportamiento de los sistemas de gestión de contactos y realizar una investigación directa e inmediata sobre los procesos, fenómenos u objetos relacionados. Estimula la curiosidad, impulsa el desarrollo de nuevos hechos que pueden tener interés científico.

**Estructuración de los capítulos de la investigación:**

El trabajo de diploma se divide en tres capítulos, los cuales estarán estructurados de la siguiente forma:

- **Capítulo 1: "Fundamentación teórica".** Se describen los conceptos por los cuales se rige el dominio de la investigación y se realiza un análisis de los sistemas conocidos en el ámbito nacional e internacional. Se analizan las herramientas, metodologías y técnicas utilizadas para la solución del problema planteado en la investigación.
- **Capítulo 2: "Propuesta solución".** En este capítulo se diseña la propuesta de solución según la metodología empleada. Se describen los requisitos funcionales y no funcionales con las historias de usuarios de los requerimientos que debe cumplir el sistema, se define el diseño de la arquitectura, los patrones del diseño, el diagrama de clase del diseño.
- **Capítulo 3: "Pruebas o validación".** En este capítulo se realiza la implementación del sistema. Se efectúan los diseños de casos de pruebas y se aplican las pruebas al sistema.

# CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

El presente capítulo tiene como fin lograr una mayor comprensión sobre que es un sistema de gestión de contactos. Para ello serán definidos los conceptos asociados al tema, para un mayor entendimiento. Además, se realiza un estudio sobre las diferentes soluciones homólogas existentes en el marco nacional e internacional. Luego se fundamenta la metodología a emplear junto con las tecnologías y herramientas.

## 1.1 Conceptos asociados al problema

**Plataforma de telecomunicaciones Platel:** Platel es una plataforma integral de comunicaciones que ofrece diversos servicios como voz, datos, video, mensajería unificada, seguridad, supervisión y notificaciones. Para ello, emplea protocolos estándares e interfaces flexibles, con la capacidad de interoperar con otros sistemas empresariales.

Esta herramienta cuenta con una arquitectura de software segura, robusta y flexible, que permite configurar sus funciones y servicios de acuerdo con las características y necesidades de cada cliente, para optimizar la administración de los servicios de comunicación. Es importante destacar que esta soporta una amplia gama de tecnologías IP, interfaces para la conexión a la red telefónica pública conmutada y a la internet(XETID 2023).

**Centro de contacto:** el Centro de Contacto como servicio (Contact Center as a Service, CCaaS) es una aplicación de atención al cliente, basada en la nube que administra y realiza un seguimiento de los recorridos del cliente, las interacciones de los empleados con los clientes y cualquier otra comunicación entrante o saliente con los clientes. Se hace un seguimiento de estas comunicaciones a través de canales de voz y digitales, como correo electrónico, chat en web y mensajes de texto.

Las plataformas de CCaaS se administran y se acceden a través de la nube. No se necesita una infraestructura independiente, lo que permite a las empresas de CCaaS impulsar automáticamente la actualización de capacidades en tiempo real sin tiempo de inactividad del software ni interrupciones del negocio, de manera similar a como Facebook actualiza su aplicación(Romano 2022).

**Software de gestión de contactos:** es una herramienta que le ayuda a almacenar la información de contacto de sus clientes, prospectos y proveedores y rastrear la comunicación entre su empresa y ellos. La información puede incluir direcciones, números de teléfono, correos

electrónicos, identificadores de redes sociales, pedidos, cotizaciones abiertas, historial de ventas, empresas asociadas, etc.

El software también le permite organizar y filtrar contactos fácilmente para ayudarlo a encontrar subconjuntos de clientes específicos. También puede filtrar la lista de contactos por industria, cargo, relaciones con su empresa, etc.

Además, puede administrar las comunicaciones utilizando el software de administración de contactos e integrarlo con otras herramientas como calendarios para programar reuniones y recibir notificaciones de participantes y horarios.(Kumar 2021)

### 1.2 Análisis de soluciones similares

Para la investigación, se realiza un estudio, sobre las soluciones existentes que se relacionen con el objeto de estudio. En el contexto **internacional** se identifican:

**HubSpot:** el CRM (Customer Relationship Management) de HubSpot, entre muchas de sus funcionalidades, te ofrece el software para call center, el cual te permite brindar una experiencia personalizada a cada uno de tus clientes. Al registrar todas las comunicaciones en un lugar, tus clientes no tendrán que gastar tiempo repitiendo detalles; te ayuda a priorizar las llamadas de los leads calificados y se encarga de priorizar las tareas diarias para una máxima productividad.

Sus llamadas con tecnología VoIP facilitan la interacción con tus clientes, puedes grabar las llamadas y crear informes de métricas. Su sistema está basado en la nube para administrar grandes volúmenes de datos. Gracias a sus integraciones te permite programar llamadas y videollamadas, enviar correos, ver los chats de redes sociales y del sitio web.

El CRM gratuito de HubSpot incluye las siguientes características:

- **Gestión de contactos** y tareas
- Email tracking compatible correo corporativo de Gmail y Outlook
- Gestión de pipeline de ventas
- Plantillas de correo electrónico
- Buzones de correo compartidos para equipos
- Programación de citas y reuniones
- Chat en vivo y chatbots

- Cotizaciones en línea de ventas

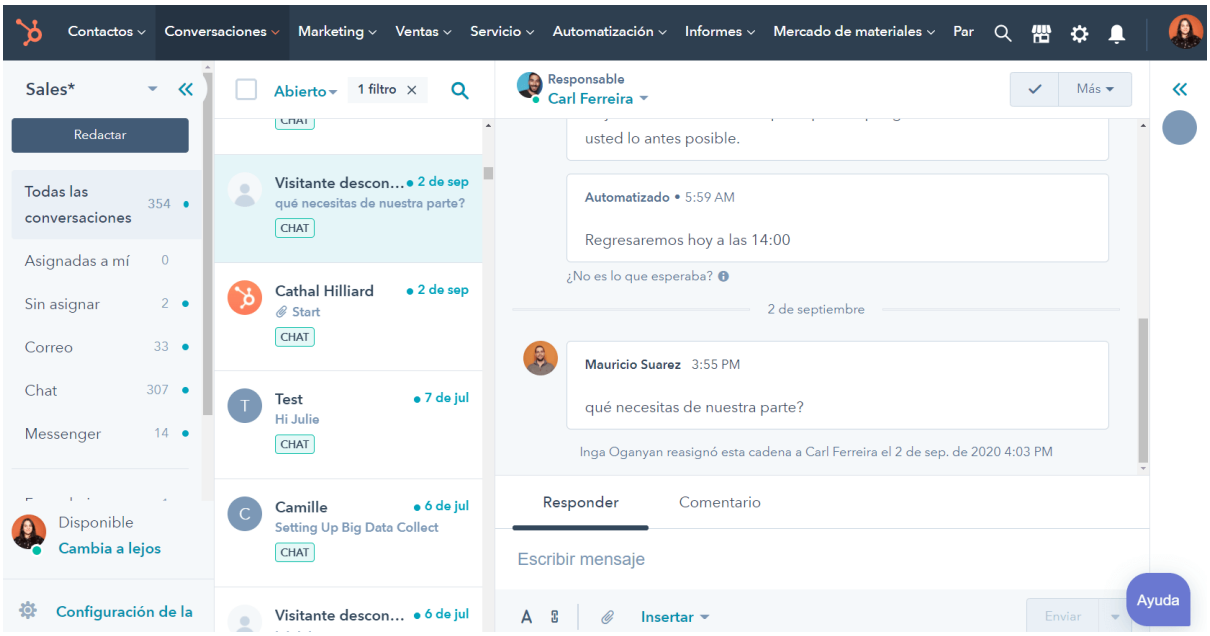



*Ilustración 1: HubSpot.*

**Contact Boss:** es un software de contactos empresariales que permite a los agentes introducir nuevos contactos rápidamente y rellenar después el resto de la información. También cuenta con funciones para organizar y buscar contactos, así como para eliminar los duplicados erróneos. La función “Legacy Data Support” permite que los usuarios suban bases de datos de contactos existentes para reducir la introducción manual de datos.

Contact Boss ofrece funciones de auditoría, informes y seguridad para que los datos estén protegidos. Se integra con Gmail, Outlook, Apple Mail y otras aplicaciones conocidas de marketing, como MailChimp, iContact y Constant Contact.

Funciones

- Base de datos de contactos
- Gestión de leads
- Segmentación
- Importación y exportación de datos
- Buscar/filtrar
- Contactos compartidos

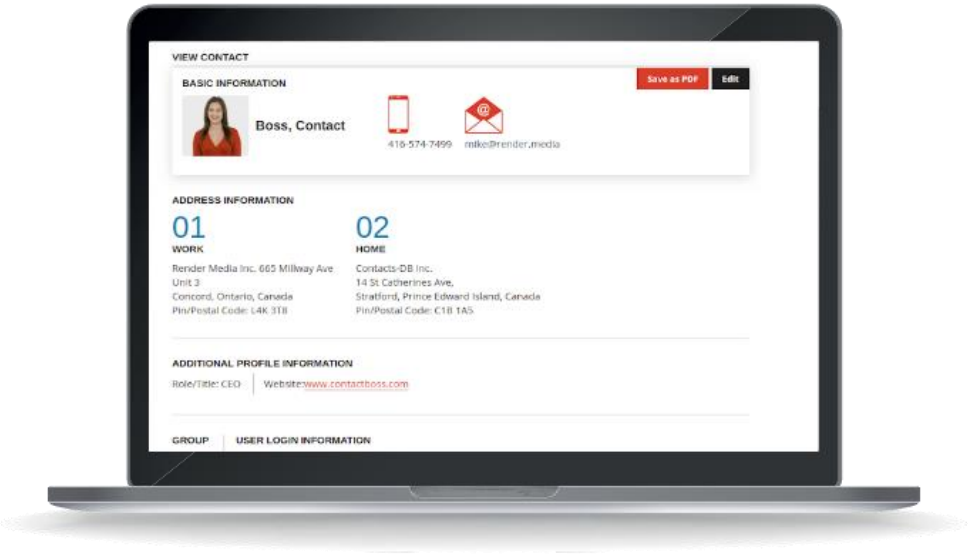



*Ilustración 2: Contact Boss.*

**EngageBay**: es uno de los mejores softwares de gestión de contactos con integración de ventas y CRM para gestionar acuerdos, tareas y contactos. Le permite conectarse con los clientes a través de varios canales mientras centraliza toda la información que le da poder a su equipo de ventas para tener una comunicación, un compromiso y un contexto significativos con los clientes potenciales.

Tendrá acceso completo a los perfiles de sus clientes, incluidas interacciones pasadas, tasas de respuesta, patrones de participación, intereses, detalles de contacto, notas de llamadas, historial de tareas, presencia social y mucho más. Con integraciones como Office 365, Google y otras plataformas, puede recopilar información e importar los datos fácilmente con unos pocos clics.

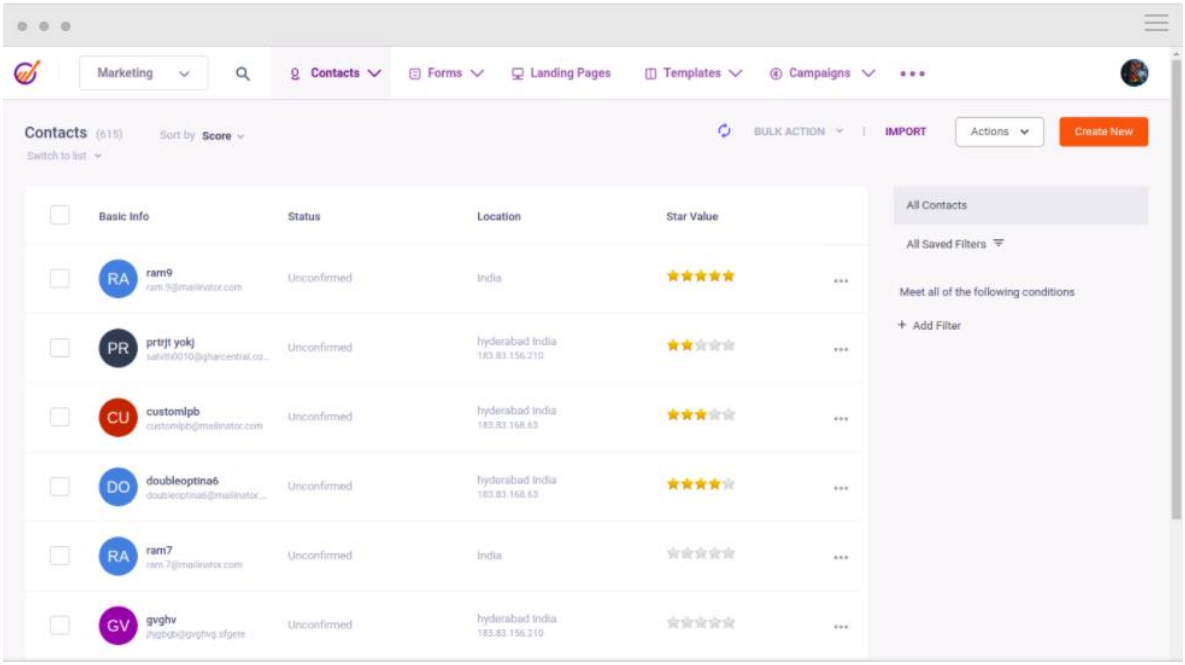



*Ilustración 3: EngageBay.*

Hubspot, Contact Boss y EngageBay son soluciones de gestión de contactos que ofrecen una amplia gama de funciones y características. Sin embargo, estas soluciones no satisfacen plenamente las necesidades de Xetid.

En primer lugar, estas soluciones son privativas por lo que sería demasiado costoso para Xetid, pues es una empresa pequeña y no puede permitirse el lujo de pagar las tarifas mensuales o anuales de estas soluciones, además no están disponibles para Cuba. En segundo lugar, estas soluciones no son lo suficientemente flexibles para adaptarse a las necesidades específicas de la empresa, por lo que necesita una solución que pueda personalizarse para satisfacer sus necesidades.

En el contexto **nacional** actualmente se desconoce de la existencia de un sistema de gestión de contactos desarrollado en Cuba.

Dadas las limitaciones de las soluciones de gestión de contactos existentes, Xetid necesita crear un contact manager personalizado. Esto le permitirá a Xetid:

- Ajustar el precio a su presupuesto.
- Personalizar las funciones y características para satisfacer sus necesidades únicas.

### 1.3 Metodología de desarrollo de software

La metodología de desarrollo de software es el conjunto de técnicas y métodos que se utilizan para diseñar una solución de software informático. Trabajar con una metodología es imprescindible por una cuestión de organización. No en vano, los factores tienen que estar ordenados y saber cómo se van a utilizar. Por otra parte, las metodologías también sirven para controlar el desarrollo del trabajo. Esto sirve para minimizar los márgenes de errores y anticiparse a esa situación.

El **Proceso Unificado Ágil** de Scott Ambler o **Agile Unified Process (AUP)** en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (test driven development - TDD en inglés)
- Modelado ágil
- Gestión de Cambios ágil
- Refactorización de Base de Datos para mejorar la productividad.

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI.

*Tabla 1: Descripción de las fases de la variación AUP-UCI.*

| **Fases AUP** | **Fases AUP-UCI** | **Objetivos de las fases (variación AUP-UCI)** |
|---|---|---|
| Inicio | Inicio | Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de |

| | | |
|---|---|---|
| | | tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto |
| Elaboración<br>Construcción<br>Transición | Ejecución | En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. |
| | Cierre | En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto. |

La metodología AUP-UCI propone para el ciclo de vida de los proyectos 7 disciplinas (modelado de negocio, requisitos, análisis y diseño, implementación, pruebas internas, de liberación y de aceptación) y surgen cuatro escenarios para modelar el sistema en los proyectos:

- Escenario No 1: proyectos que modelen el negocio con CUN (Casos de uso del negocio) solo pueden modelar el sistema con CUS (Casos de uso del sistema).
- Escenario No 2: proyectos que modelen el negocio con MC (Modelo conceptual) solo pueden modelar el sistema con CUS.
- Escenario No 3: proyectos que modelen el negocio con DPN (Descripción de proceso de negocio) solo pueden modelar el sistema con DRP (Descripción de requisitos por proceso).
- Escenario No 4: proyectos que no modelen negocio solo pueden modelar el sistema con HU (Historias de usuario).

Luego de analizar estas características se decide utilizar como guía de desarrollo de este proyecto la metodología AUP-UCI en su escenario4, pues se adapta al ciclo de vida de los proyectos desarrollados en la UCI, es aplicable a proyectos no muy extensos y el cliente va a estar presente en el proceso de desarrollo.

## 1.4 Tecnologías y herramientas

**1.4.1 Herramienta CASEVisual Paradigm v15.1** es una herramienta CASE que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software(*Guión Visual Paradigm for UML* sin fecha).

### 1.4.2 Lenguaje de modelado

El **Lenguaje de Modelado Unificado (UML) v2.0** es una herramienta que ayuda a capturar mediante un conjunto de símbolos y diagramas a comunicar la idea de un sistema (software orientado a objetos), a quien esté involucrado en su proceso de desarrollo sirviendo de apoyo en los procesos de análisis y diseño de un problema. El objetivo es capturar las partes esenciales del sistema mediante notaciones gráficas, a esto se le conoce como “modelado visual”, el cual es independiente del lenguaje de implementación (el lenguaje que se usará para codificar)( *ITESRC - Portal Académico - UML : Lenguaje de Modelado Unificado* , s. f.).

### 1.4.3 Gestor de base de datos

**PostgreSQL v14.1** es uno de los sistemas de gestión de bases de datos relacionales más empleados en la actualidad. Está orientado a objeto, contando con versiones para una amplia gama de sistemas operativos, entre ellos: Linux, Windows, Mac OS X y Solaris. Se destaca por su robustez, escalabilidad y cumplimiento de los estándares SQL(Bermúdez 2022).

Es un sistema de código abierto. Esto ha permitido que una gran comunidad de desarrolladores crezca para respaldarlo y continuar mejorándolo. Es gratuito, un software multiplataforma, es decir, es un software que puede correr bajo distintos entornos y sistemas operativos, y es compatible con muchos de los servidores web más populares como Apache, Nginx y LiteSpeed por mencionar algunos. Puede manejar un gran volumen de datos

### 1.4.4 Marco de trabajo para el desarrollo de la solución informática

**Django v4.2.2** es un framework de desarrollo para Python. Se trata de una herramienta de código abierto y gratuita que cuenta con una comunidad amplia y que comparte recursos constantemente. Además, Django también cuenta con funciones de pago que pueden facilitar más el trabajo de los desarrolladores. Es una herramienta que se puede usar para el desarrollo

full-stack de aplicaciones y páginas web, así como para el desarrollo de servidores. Está considerado como el mejor framework para el desarrollo de aplicaciones web con Python y es uno de los marcos de desarrollo más demandados por los programadores que trabajan con este lenguaje en el desarrollo web.

**Bootstrap v5.3.1** se trata de un framework de CSS utilizado en aplicaciones front-end. Es decir, una biblioteca de estilos genéricos utilizada para diseñar la pantalla de interfaz con el usuario Cuenta con múltiples recursos para configurar los distintos elementos de cada página de una manera simple y eficiente. Además, facilita la construcción de aplicaciones para todo tipo de soportes, ya que sus páginas están adaptadas tanto para la web como para dispositivos móviles.(Benito 2022)

El framework Bootstrap combina los lenguajes de programación CSS y JavaScript para estilizar los elementos de una página HTML. Por eso, ofrece muchas más funcionalidades que, simplemente, cambiar la forma y el color de botones y enlaces.

**Visual Studio Code (VS Code) v1.78.2** es uno de los editores de código más populares y potentes en la actualidad. Desarrollado por Microsoft, este entorno de desarrollo integrado (IDE) se ha convertido en una herramienta indispensable para los programadores y desarrolladores de software. Con una interfaz intuitiva y un conjunto excepcional de características, VS Code ha revolucionado la forma en que los profesionales interactúan y crean código(*Características clave de Visual Studio Code - TecnoBits* 2023).

Entre las características principales se encuentra su capacidad para trabajar con múltiples lenguajes de programación. Además, cuenta con una amplia comunidad de desarrolladores que contribuyen constantemente con extensiones y paquetes que agregan funcionalidades extra al editor. Tiene la capacidad de personalización, pues se puede adaptar el editor a tus necesidades y preferencias, cambiando el tema de color, ajustando el tamaño y tipo de letra, y agregando atajos de teclado personalizados. También permite la integración con herramientas de control de versiones como Git, lo que facilita el trabajo en equipo y la colaboración en proyectos. Estas características claves lo hacen una herramienta indispensable para desarrolladores de todo tipo convirtiéndolo en una opción versátil y eficiente para desarrolladores de todos los niveles.

### 1.4.5 Lenguajes de programación

**Python v3.10** es un lenguaje de programación creado por Guido Van Rossum. Se trata de un código que se basa en C y es un lenguaje de alto nivel que permite crear operaciones simples y

complejas. Además, cuenta con multitud de librerías y extensiones que hacen que soporte otros códigos como Java, C, C++ o JSON (School 2022). Es un lenguaje de programación orientado a objetos y de código abierto. Python es un código muy amable para todo tipo de desarrolladores, desde los que ya tienen experiencia con otros lenguajes como para los que están aprendiendo a programar desde cero.

**JavaScript** es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, utilizado principalmente en el desarrollo web, pensado para agregar potencial de interacción y dinamismo a las páginas web. Es compatible con todos los navegadores modernos y se ejecuta del lado del cliente, lo que significa que se ejecuta en el navegador web del usuario final. Además de su uso en el desarrollo web, JavaScript también se utiliza en aplicaciones de servidor (con tecnologías como Node.js) y en el desarrollo de aplicaciones móviles y de escritorio.

JavaScript proporciona una amplia gama de funcionalidades, como manipulación del DOM (Document Object Model) para interactuar con elementos de una página web, manejo de eventos, comunicación con servidores a través de AJAX, creación de animaciones, validación de formularios y mucho más. Es un lenguaje flexible y dinámico que permite a los desarrolladores crear experiencias interactivas y ricas en contenido(Coppola 2023).

**Conclusiones del capítulo**

Durante el desarrollo de este capítulo se realiza el estudio de los conceptos relacionados al tema para una mejor comprensión. Se realiza un análisis sobre los sistemas de gestión de contactos tanto en el contexto internacional como nacional arribando a que ninguno satisface las necesidades planteadas en la situación problemática por sus características tan particulares. El estudio de las herramientas, tecnologías y metodología de desarrollo de software permite crear la base tecnológica adecuada para el desarrollo de este tipo de aplicación. Se selecciona como metodología a utilizar AUP versión UCI para el modelado la herramienta Visual Paradigm, como lenguaje de programación se decide usar Python con su respectivo framework Django para el lado del servidor y para el lado del cliente Javascript junto al framework Bootstrap, utilizando como IDE de desarrollo el Visual Studio Code y como gestor de bases de datos PostgresSQL.

# CAPÍTULO 2: PROPUESTA SOLUCIÓN

En el presente capítulo se realiza el análisis y diseño de la propuesta de solución, mediante la realización del modelo de dominio para identificar la relación que existe entre los principales conceptos del sistema, así como los requisitos funcionales y no funcionales. Además, se desarrollan los artefactos pertinentes, según la metodología seleccionada.

## 2.1 Modelo de dominio.

El modelo de dominio se crea con el fin de representar los conceptos clave del dominio del problema. El modelo de dominio también identifica las relaciones entre todas las entidades comprendidas en el ámbito del dominio del problema, y comúnmente identifica sus atributos. Un modelo de dominio que encapsula los métodos dentro de las entidades se asocia más bien con modelos orientados a objetos. El modelo de dominio proporciona una visión estructural del dominio que puede ser complementado con otros puntos de vista dinámicos, como el modelo de casos de uso. En la figura se representa el modelo de dominio para la aplicación web.


*Ilustración 4: Modelo de dominio. Fuente: elaboración propia.*

### 2.1.1 Descripción de Conceptos.

**Usuario:** persona que interactúa con el sistema.

**Directorio:** lugar donde se encuentra la información de los usuarios registrados en el sistema

**Contactos:** contiene la información de los usuarios registrados en el sistema.

## 2.2 Requisitos de Software.

La ingeniería de requerimientos proporciona el mecanismo apropiado para entender lo que desea el cliente, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable, especificar la solución sin ambigüedades, validar la especificación y administrar los requerimientos a medida que se transforman en un sistema funcional. Es importante notar que algunas de estas tareas ocurren en paralelo y que todas se adaptan a las necesidades del proyecto(*Presman* 2010).

### 2.2.1 Requisitos Funcionales

Los requerimientos funcionales para un sistema refieren lo que el sistema debe hacer. Tales requerimientos dependen del tipo de software que se esté desarrollando, de los usuarios esperados del software y del enfoque general que adopta la organización cuando se escriben los requerimientos. Al expresarse como requerimientos del usuario, los requerimientos funcionales se describen por lo general de forma abstracta que entiendan los usuarios del sistema.(*Sommerville 2011*). A partir de las entrevistas realizadas se definieron los requisitos funcionales que a continuación aparecen:

*Tabla 2: Requisitos Funcionales. Fuente: Elaboración propia.*

| No. | Nombre | Descripción | Prioridad | Complejidad |
|---|---|---|---|---|
| RF1. | Autenticar usuario | El sistema debe permitir al usuario autenticarse mediante el usuario y contraseña. | Alta | Baja |
| RF2 | Cerrar sesión | El sistema debe permitir al usuario salir del mismo | Alta | Baja |
| RF3 | Crear contacto | El sistema debe permitir adicionar contactos | Alta | Alta |
| RF4 | Listar contactos | El sistema debe permitir mostrar los contactos. | Alta | Media |
| RF5 | Editar contacto | El sistema debe permitir editar los contactos. | Alta | Alta |

| | | | | |
|---|---|---|---|---|
| RF6 | Eliminar contacto | El sistema debe permitir eliminar contactos. | Alta | Baja |
| RF7 | Buscar contacto | El sistema debe permitir al usuario buscar contactos. | Alta | Alta |
| RF8 | Mostrar datos del contacto. | El sistema debe permitir al usuario mostrar la información completa del contacto | Baja | Baja |
| RF9 | Importar lista de contactos. | El sistema debe permitir al usuario importar una lista de contactos. | Media | Alta |
| RF10 | Exportar lista de contactos. | El sistema debe permitir al usuario exportar la lista de contactos en un archivo csv. | Alta | Alta |
| RF11 | Marcar contacto como favorito. | El sistema debe permitir al usuario marcar contactos como favorito. | Baja | Baja |
| RF12 | Listar contactos favoritos | El sistema debe permitir listar los contactos seleccionados como favoritos. | Baja | Baja |
| RF 13 | Crear etiquetas | El sistema debe permitir crear etiquetas. | Media | Media |
| RF14 | Mostrar etiquetas creadas | El sistema debe mostrar las etiquetas creadas. | Baja | Baja |
| RF15 | Asignar etiquetas a los contactos | El sistema debe permitir asignarle una etiqueta a cada contacto. | Media | Alta |
| RF16 | Mostrar papelera de contactos eliminados. | El sistema debe permitir mostrar los contactos eliminados. | Media | Baja |

| | | | | |
|---|---|---|---|---|
| RF17 | Recuperar contacto eliminado. | El sistema debe permitir recuperar los contactos eliminados. | Media | Media |
| RF18 | Vaciar papelera | El sistema debe permitir vaciar la papelera de reciclaje. | Media | Alta |
| RF19 | Añadir grid de información de contacto | El sistema debe permitir añadir grid con información de contactos. | Media | Alta |
| RF20 | Añadir tarea | El sistema debe permitir agendar tareas. | Alta | Alta |
| RF21 | Editar tarea | El sistema debe permitir editar tareas. | Alta | Alta |
| RF22 | Eliminar tarea | El sistema debe permitir eliminar tareas. | Alta | Media |
| RF23 | Mostrar tareas en el calendario | El sistema debe permitir mostrar tareas en el calendario. | Alta | Alta |

### 2.2.2 Requisitos No Funcionales.

Los requerimientos no funcionales, como indica su nombre, son requerimientos que no se relacionan directamente con los servicios específicos que el sistema entrega a sus usuarios. Pueden relacionarse con propiedades emergentes del sistema, como fiabilidad, tiempo de respuesta y uso de almacenamiento. De forma alternativa, pueden definir restricciones sobre la implementación del sistema, como las capacidades de los dispositivos I/O o las representaciones de datos usados en las interfaces con otros sistemas.(*Sommerville* 2011)

**Usabilidad**

RNF1. Se garantizará la uniformidad de las interfaces de usuario, teniendo en cuenta que las operaciones comparables siempre se activen de la misma forma.

**Interfaz**

RNF2. La apariencia del producto será sencilla, cómoda, amigable y de fácil navegación para cualquier tipo de usuario.

**Eficiencia**

RNF3. El tiempo de demora del sistema en cada transición debe ser el menor posible.

**Requisitos de Software:**

RNF4. Navegador web en el pc cliente: Internet Explorer, Mozilla, etc.

RNF5. Entorno de ejecución sistema operativo Windows.

**Requisitos de Seguridad:**

RNF6: El sistema debe ser seguro, mediante un sistema con autenticación.

**Confidencialidad**

RNF7. La información que se maneje en la herramienta no deberá ser conocida por personal no autorizado y estará prohibida su divulgación a personas externas.

**Disponibilidad**

RNF8. La información se encontrará disponible en todo momento para aquellos usuarios autorizados a acceder al sistema.

### 2.2.3 Historias de usuario

Una historia de usuario (HU) describe una funcionalidad que, por sí misma, aporta valor al usuario. En el escenario 4 de la metodología de desarrollo de software AUP-UCI las HU constituyen el artefacto utilizado para describir las funcionalidades del sistema. Las historias de usuario constituyen las bases para las pruebas funcionales ya que se utilizan para verificar si la propuesta desarrollada cumple con lo que especifica en ellas. A continuación, se representan las HU de los requisitos funcionales del componente.

*Tabla 3: Historia de usuario Crear contacto. Fuente: elaboración propia.*

| Historias de usuario | |
|---|---|
| Número:3 | Usuario: Cliente |
| Nombre de Historia: Crear contacto | |
| Prioridad del negocio: Alta | |

| | |
|---|---|
| Riesgo en desarrollo: Alta | Iteración asignada:1 |
| Tiempo estimado:5 días | Tiempo Real:5 días |
| Programador responsable: Dayelin Alayo Roll | |
| Descripción: Le permite al usuario introducir los datos del contacto para luego ser almacenados en la Base de Datos. Entre los campos que el usuario debe llenar están:<br>• Foto<br>• Nombre(obligatorio)<br>• Apellidos<br>• Teléfono(obligatorio)<br>• Correo<br>• Empresa<br>• País<br>• Ciudad<br>• Dirección<br>• Fecha de nacimiento<br>• Redes sociales<br>• Agregar más información<br>. | |
| Prototipo: | |

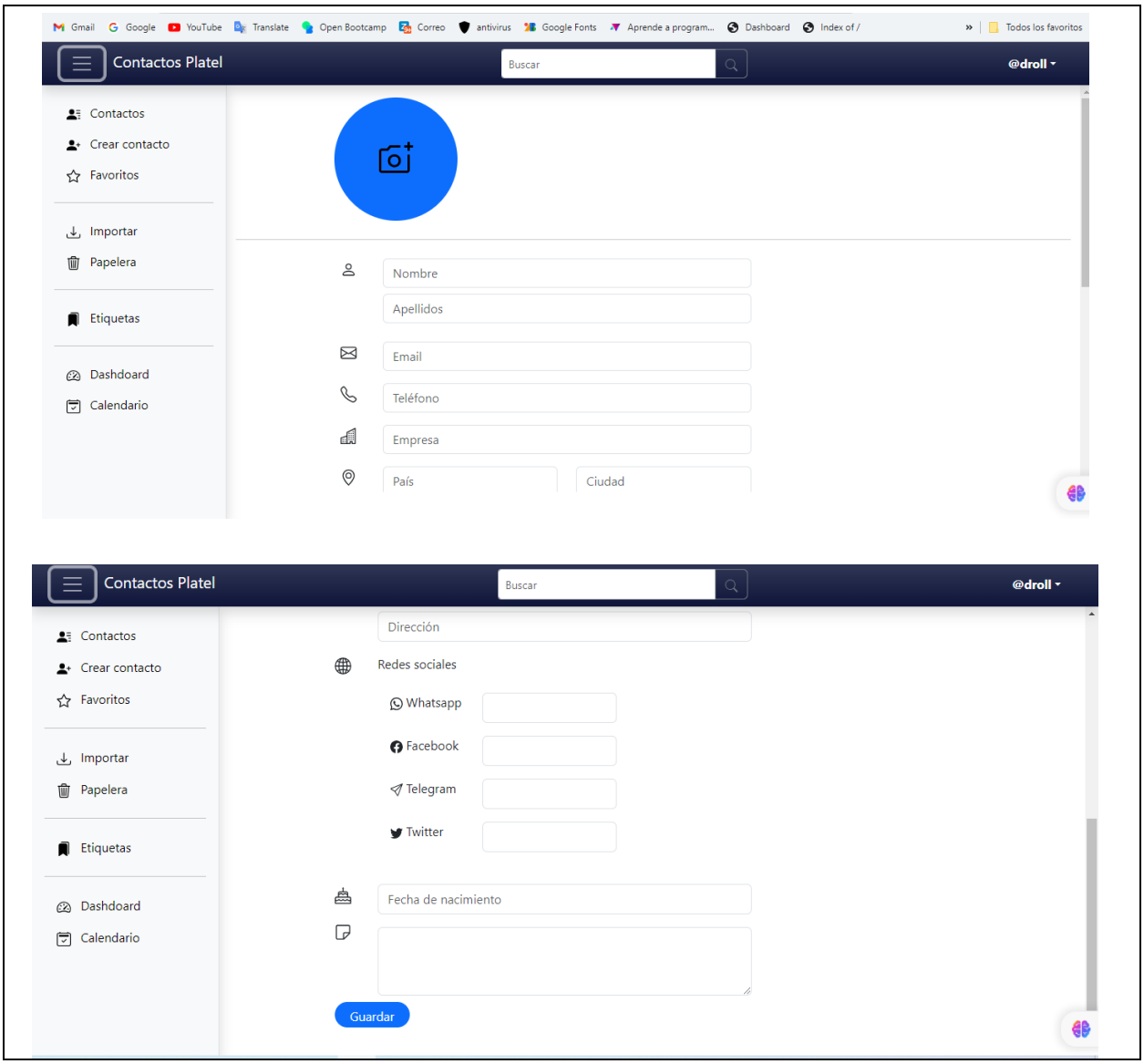
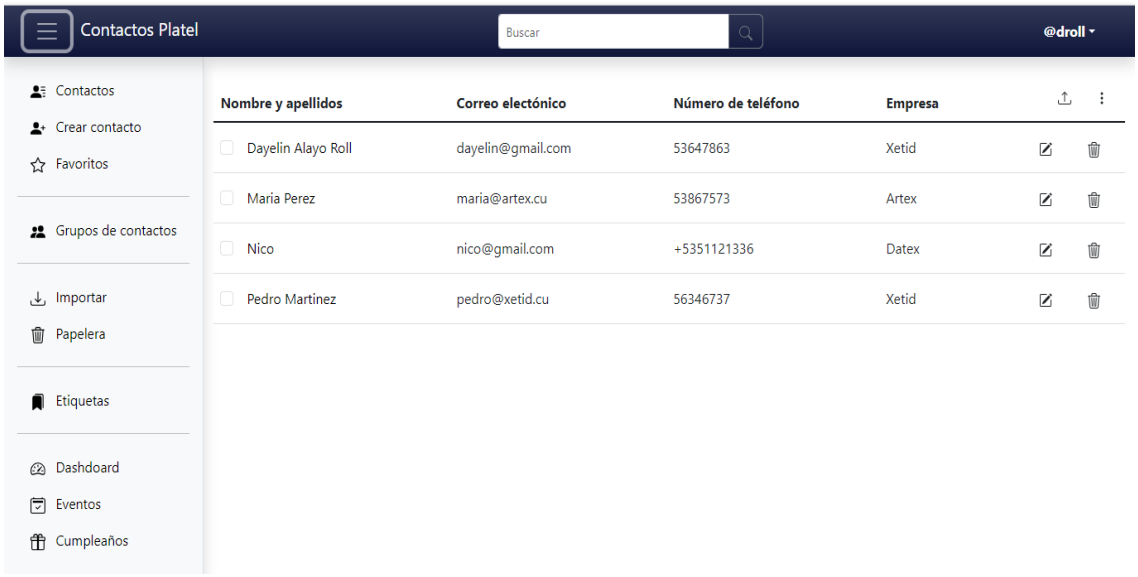



*Tabla 4: Historia de usuario Listar contactos. Fuente: elaboración propia*

| Historias de usuario | |
|---|---|
| Número:4 | Usuario: Cliente |
| Nombre de Historia: Listar contactos. | |
| Prioridad del negocio: Alta | |
| Riesgo en desarrollo: Baja | Iteración asignada:1 |
| Tiempo estimado:5 días | Tiempo Real: 5 días |

| |
|---|
| Programador responsable: Dayelin Alayo Roll |
| Descripción: Le permite al usuario visualizar la lista de contactos. |
| Prototipo: |




## 2.3 Arquitectura de software.

La arquitectura de software es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores del software compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación. Es considerada el nivel más alto en el diseño de la arquitectura de un sistema puesto que establecen la estructura, funcionamiento e interacción entre las partes del software. (*Revista Digital - Arquitectura del Software - Alfonso Arias M. - Sistemas I by C3R3VR0 - Issuu* 2021)

Se trata de una serie de decisiones basadas en una amplia gama de factores, y cada una de estas decisiones pueden tener un impacto considerable en la calidad, rendimiento, facilidad de mantenimiento, y en general el éxito de la aplicación.

### 2.3.1 Patrones arquitectónicos

Los patrones arquitectónicos se abocan a un problema de aplicación específica dentro de un contexto dado y sujeto a limitaciones y restricciones. El patrón propone una solución arquitectónica que sirve como base para el diseño de la arquitectura (*Presman.2010*).

Para la realización de este sistema se utiliza el patrón arquitectónico Modelo-Vista-Plantilla (MTV), el cual, constituye un patrón de arquitectura de software utilizado por el framework del lenguaje de programación Python, Django. Es una variante al Modelo-Vista-Controlador.

El patrón MTV consta de tres componentes principales: Modelo (Model), Plantilla (Template) y Vista (View)

- **Modelo:** es la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- **Plantilla:** la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.
- **Vista:** la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre el modelo y las plantillas.

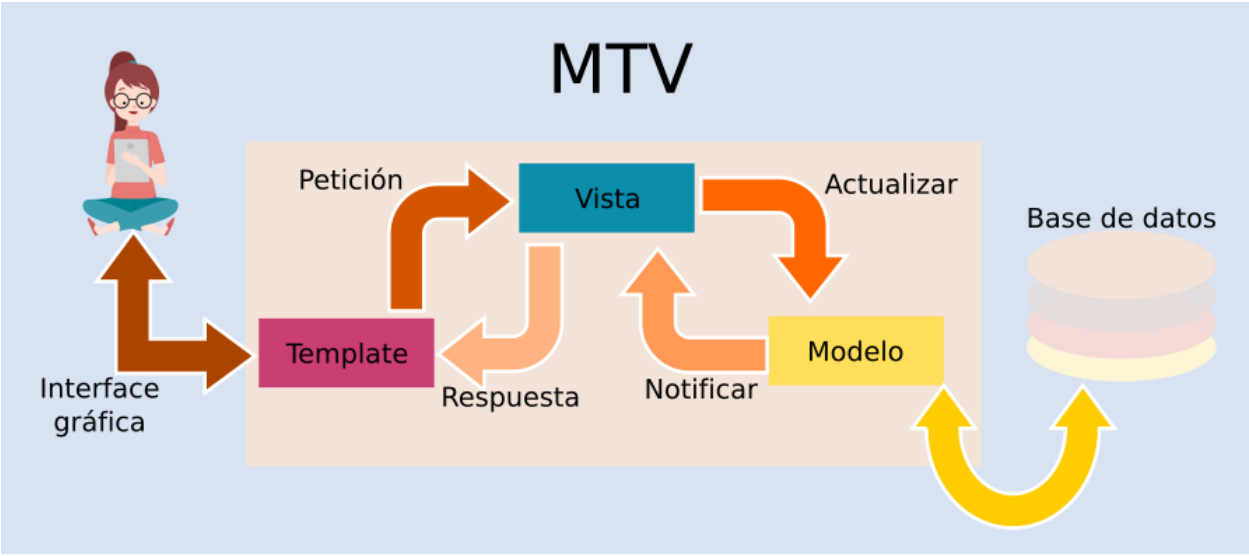



*Ilustración 5: Arquitectura MVT*

### 2.3.2 Diagrama de clases

Un diagrama de clases es una estructura estática que se usa para mostrar los tipos de relaciones entre los objetos que se están programando. También es una buena manera de mostrar la estructura de clases en un sistema. Un diagrama de clases es especialmente útil para comunicar las jerarquías de clases y las colaboraciones de clases con las partes interesadas o un equipo (Fonseca 2022). A continuación, se muestra el diagrama de clases para el crud Gestionar contactos.

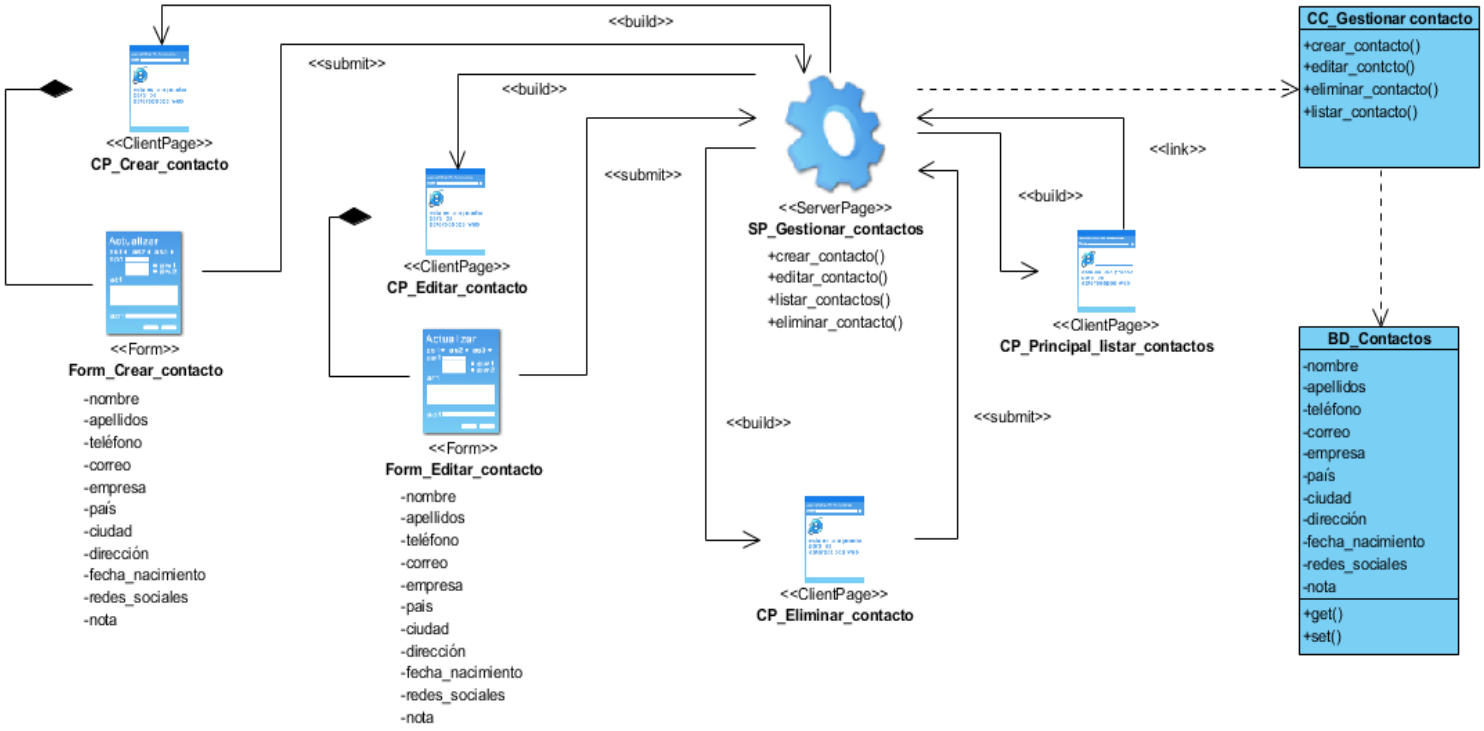



*Ilustración 6: DCD. Gestionar contactos. Fuente: elaboración propia.*

### 2.4 Patrones de diseño

Los patrones de diseño son una solución general, reutilizable y aplicable a diferentes problemas de diseño de software. Se trata de plantillas que identifican problemas en el sistema y proporcionan soluciones apropiadas a problemas generales a los que se han enfrentado los desarrolladores durante un largo periodo de tiempo, a través de prueba y error.

Estos patrones proporcionan un enfoque estructurado y reutilizable para resolver situaciones recurrentes en el desarrollo de software. Los patrones de diseño ayudan a los desarrolladores a comunicarse y compartir soluciones eficientes y efectivas que han demostrado ser exitosas en el pasado (Canelo 2020).

### 2.4.1 Patrones GRASP

Los patrones GRASP son patrones generales de software para asignar responsabilidades. Son guías, buenas ideas y buenos motivos para definir a qué clases asignarle ciertas responsabilidades. No son reglas a seguir con los ojos cerrados, pero si nos vamos a desviar es bueno tener en claro el motivo y las ventajas/desventajas de no seguir estos patrones (*(25) GRASP con ejemplos | LinkedIn* 2017).

**Creador.** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos (una tarea muy común). La intención básica del patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. La ventaja de utilizar este patrón es eliminar la dependencia entre las clases logrando una mayor mantenibilidad y reutilización. Dentro del sistema se evidencia por sus acciones en la clase controladora “view.py”, las cuales crean objetos para sus entidades.

**Bajo acoplamiento.** El acoplamiento es una medida de la fuerza con que un elemento está conectado a, tiene conocimiento de, confía en, otros elementos. Un elemento con bajo (o débil) acoplamiento no depende demasiado de otros elementos. El patrón Bajo Acoplamiento es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño y soporta clases más independientes. Las URLconfs de Django son un claro ejemplo de este principio en la práctica. En una aplicación de Django, la definición de la URL y la función de vista que se llamará están débilmente acopladas.

```
urlpatterns = [
    path('', views.contact_list, name='list'),
    path('export/', views.exportar, name='exportar'),
    path('add/', views.contactos_add, name='add'),
    path('<int:pk>/edit/', views.contactos_edit, name='editar'),
    path('<int:pk>/mover/', views.mover_papelera, name='papelera'),
    path('<int:pk>/detalle/', views.detalles, name='detalles'),
]
```

*Ilustración 7: Captura de pantalla de urls del proyecto.*

**Experto:** Este patrón se basa en la asignación de responsabilidades, las cuales se asignan a las clases que posean la información necesaria para llevarlas a cabo. Es utilizado en la capa de abstracción del modelo de datos. Con el uso del Object Relational Mapping (ORM) de Django, se crean las clases que representan las entidades del modelo de datos. Asociado a cada una de estas clases, son generadas un conjunto de funcionalidades que las relacionan de forma directa con la entidad que representan. Las clases contienen toda la información necesaria de la tabla que representan en la base de datos.

```python
class Contacto(models.Model):
    avatar = models.ImageField(upload_to='contact', null=True, blank=True)
    nombre=models.CharField(max_length=50)
    apellidos=models.CharField(max_length=100, null=True, blank=True,)
    empresa=models.CharField(max_length=200, null=True, blank=True,)
    puesto=models.CharField(max_length=200, null=True, blank=True,)
    correo=models.EmailField(null=True, blank=True,)
    telefono=models.CharField(max_length=20, null=True, blank=True)
    cumpleaños=models.DateField(null=True, blank=True, verbose_name='Fecha de nacimiento')
    nota=models.TextField(null=True, blank=True,)
    mover_papelera = models.BooleanField(default=False)
    creado_por = models.ForeignKey(User, related_name='contactos', on_delete=models.CASCADE)



    class Meta:
        ordering = ('nombre',)


    def __str__(self) -> str:
        return self.nombre
```

*Ilustración 8: Captura de pantalla del modelo de contactos del proyecto.*

### 2.4.2 Patrones GoF.

Un patrón de diseño provee un esquema para refinar los componentes de un sistema de software, o las relaciones entre ellos. Estos brindan soluciones a los problemas que pueda haber en el diseño de un software

**Decorador:** es un patrón de tipo estructura, ya que permite que clases y objetos sean utilizados para componer estructuras de mayor tamaño. Este patrón añade dinámicamente nuevas responsabilidades a un objeto. Django contienen un decorador que permite agregar funcionalidades dinámicamente a las aplicaciones desarrolladas bajo sus principios. Cada una de las vistas generadas hereda su diseño de la plantilla “base.html”, siendo esta la plantilla contenedora de la estructura y el diseño básico de las plantillas.

## Conclusiones del capítulo

En este capítulo se presentan los artefactos generados en la fase de análisis y diseño en la metodología AUP versión UCI escenario 4 como son el modelo de dominio, las historias de usuario y el diagrama de clases de diseño. También se realiza el levantamiento de requisitos de software, se define como patrón arquitectónico Modelo-Vista-Plantilla y se explica cómo se ponen de manifiesto los patrones GRASP y GOF en la implementación de la solución.

# CAPÍTULO 3: PRUEBAS O VALIDACIÓN

En el capítulo se aborda las actividades que se llevan a cabo durante la fase de implementación y pruebas. En la fase de implementación se genera el diagrama de componentes y el diagrama de despliegue, para visualizar la organización de los componentes del sistema, las relaciones de dependencia entre ellos y la arquitectura de ejecución del sistema. También se generan las pruebas seleccionadas para la validación de la implementación, verificando en cada una de ellas que los resultados sean los esperados.

## 3.1 Diagrama de despliegue

Un diagrama de despliegue es un tipo de diagrama UML que muestra la arquitectura de ejecución de un sistema, incluyendo nodos como entornos de ejecución de hardware o software, y el middleware que los conecta. Los diagramas de despliegue se utilizan normalmente para visualizar el hardware y el software físico de un sistema, para una mayor comprensión de cómo el sistema se desplegará físicamente en el hardware(Siriwardhana 2020a). A continuación, se muestra el diagrama de despliegue de la aplicación de contactos:

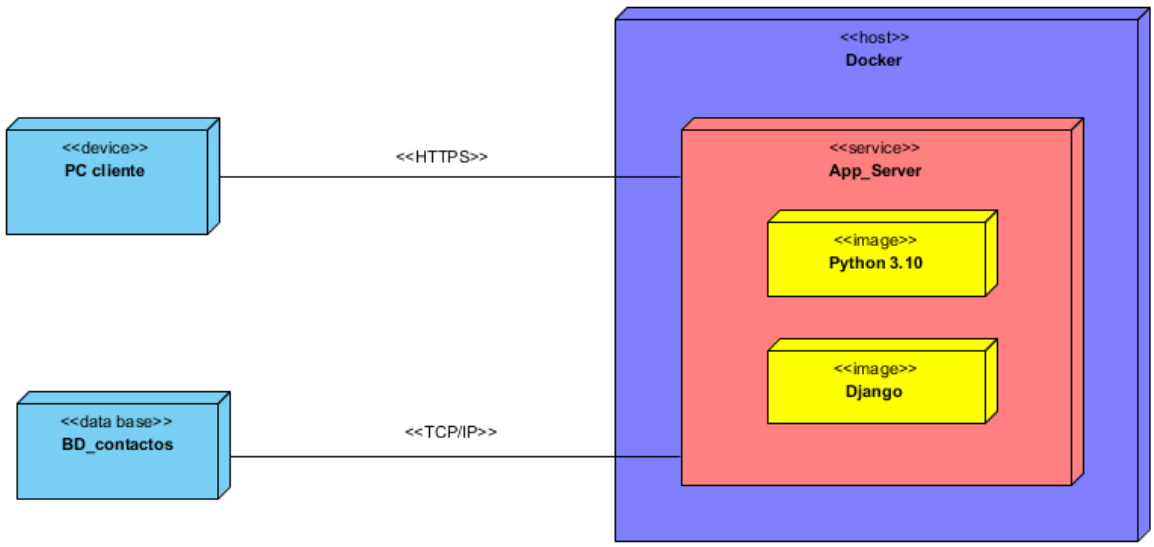



*Ilustración 7: Diagrama de despliegue. Fuente: elaboración propia.*

- **PC cliente:** este nodo representa una computadora personal que se utiliza para acceder a la aplicación web.
- **Docker:** host servidor donde será alojado en contenedor de Docker que virtualizará y modularizará el sistema entre su código, librerías y dependencias (Python, Django) para permitir un despliegue más rápido y facilitar el posterior mantenimiento. En cuanto a requerimientos de hardware, estos pueden ser satisfechos sin necesidad de superar los mismos establecidos para las máquinas clientes.
- **Base de datos:** este nodo representa un sistema de almacenamiento de datos que contiene los datos de la aplicación.

### 3.2 Diagrama de componentes

Los diagramas de componentes se utilizan para visualizar la organización de los componentes de un sistema y las relaciones de dependencia entre ellos. Proporcionan una visión de alto nivel de los componentes de un sistema. Los componentes pueden ser un componente de software como;  una base de datos o una interfaz de usuario o un componente de hardware; como un circuito, un microchip o un dispositivo; o una unidad de negocio, como un proveedor, una nómina o un envío (Siriwardhana 2020b).

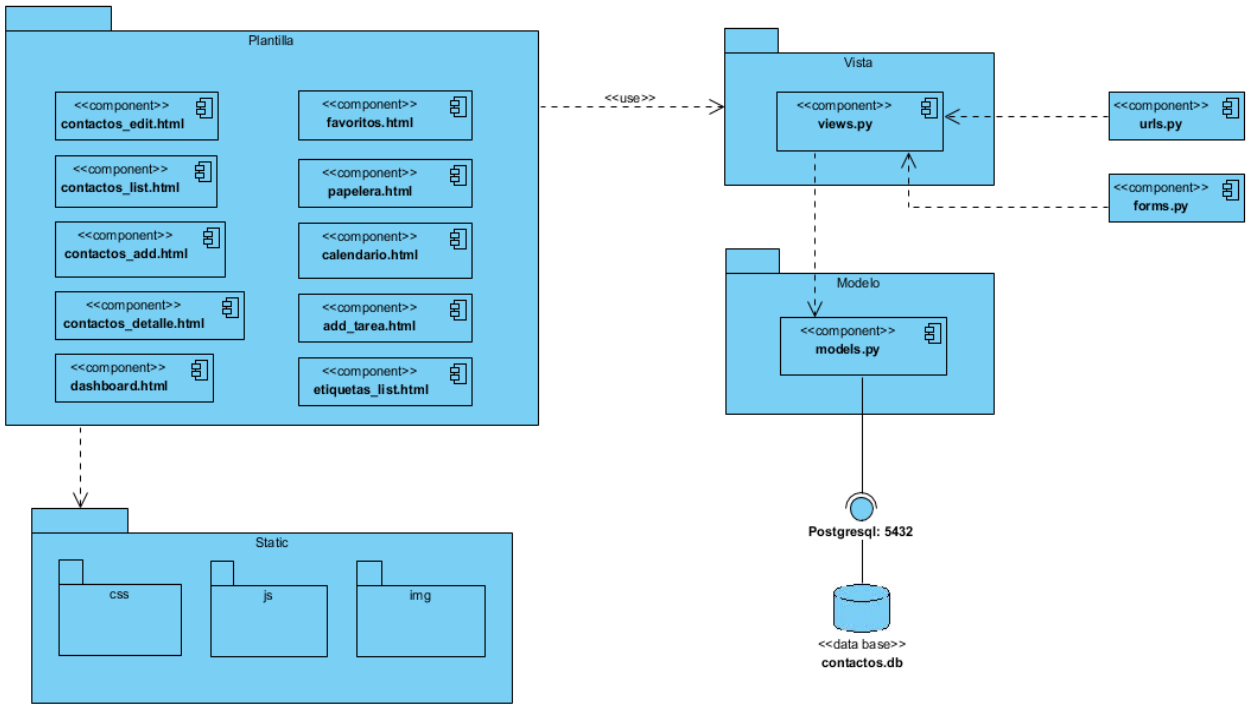


*Ilustración 8: Diagrama de componente. Fuente: elaboración propia.*

### 3.3 Pruebas de software

Las pruebas de software son una parte integral del ciclo de vida del desarrollo de software (SDLC). Las pruebas son la forma en que puede estar seguro acerca de la funcionalidad, el rendimiento y la experiencia del usuario (Lee 2020).

Las pruebas de software se pueden dividir en dos tipos diferentes: pruebas funcionales y no funcionales. Diferentes aspectos de una aplicación de software requieren diferentes tipos de pruebas, como pruebas de rendimiento, pruebas de escalabilidad, pruebas de integración, pruebas unitarias y muchos más. Cada uno de estos tipos de pruebas de software ofrece una excelente visibilidad de la aplicación, desde el código hasta la experiencia del usuario.

#### 3.3.1 Estrategia de pruebas

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos se requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados, además, debe ser suficientemente flexible para promover un uso personalizado de la prueba. Al mismo tiempo, debe ser suficientemente rígida para alentar la planificación razonable y el seguimiento de la gestión conforme avanza el proyecto (Presman 2010).

A continuación, se muestra la estrategia de prueba diseñada para la propuesta solución, en función de validar y garantizar la calidad de este.

*Tabla 5: Estrategia de pruebas. Fuente: elaboración propia.*

| Pruebas | Método | Alcance |
|---|---|---|
| Unitaria | Caja Blanca (Técnica de camino básico). | Para garantizar el correcto funcionamiento de los métodos. |

| | | |
|---|---|---|
| Rendimiento | Prueba de carga y estrés. | Se valida el comportamiento del sistema con distintos niveles de usuarios concurrentes y el consumo excesivo de sus recursos. |
| Funcional | Caja Negra (Partición de equivalencia). | Se valida el funcionamiento del sistema. |

### 3.3.2 Pruebas unitarias

Las pruebas unitarias son una parte esencial del proceso de desarrollo de software que pone a prueba los componentes individuales de la aplicación o programa de software para detectar el error fácilmente. El objetivo principal de las pruebas unitarias es comprobar que cada parte individual funciona según los requisitos del cliente (Acharya 2023).

**Prueba de caja blanca.**

Son pruebas estructurales, conociendo el código y siguiendo su estructura lógica, se pueden diseñar pruebas destinadas a comprobar que el código hace correctamente lo que el diseño de bajo nivel indica y otras que demuestren que no se comporta adecuadamente ante determinadas situaciones (pincay 2017).

Entre las técnicas existentes de caja blanca se encuentran: prueba del camino básico, prueba de condición, prueba de flujo de datos y prueba de bucles. Para la investigación se utiliza la prueba del camino básico, la misma permite: generar el grafo de flujo, calcular la Complejidad Ciclomática, determinar los caminos linealmente independientes y diseñar los casos de prueba para forzar la ejecución de cada camino del conjunto básico.

Para calcular la Complejidad Ciclomática, se tiene tres opciones:

- Restar las aristas menos los nodos y sumar 2:

  V(G) = Aristas – Nodos + 2

- Sumar 1 al número de nodos predicados:

  V(G) = Nodos predicados + 1

- Contar el número de regiones (espacios «encerrados entre nodos y aristas», también se tiene en cuenta el espacio «exterior» a todos los nodos y aristas).

V(G) = Regiones

Método:

*Ilustración 9: Código añadir contactos.*

```
def contactos_add(request):                                  1
    if request.method == 'POST':                             2
        form = ContactForm(request.POST)

        if form.is_valid():                                  4
            contacto = form.save(commit=False)
            contacto.creado_por = request.user
            contacto.save()

            return redirect('contactos:list')                5

    else:                                                    3
        form = ContactForm()

    return render(request, 'contacto/contactos_form.html', {  5
        'form': form,
    })
```

Grafo resultante:

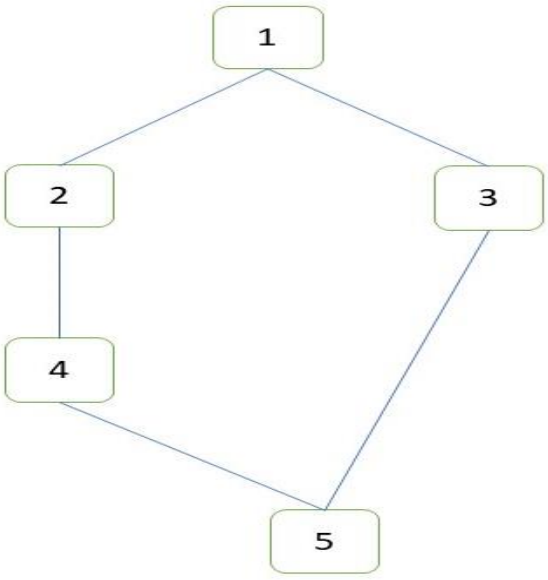


*Ilustración 10: Grafo de flujo. Fuente elaboración propia.*

**Complejidad Ciclomática:**

*Tabla 6: Complejidad Ciclomática. Fuente: elaboración propia.*

| | | |
|---|---|---|
| *V(G) = # de regiones*<br>*V(G) = 2* | *V(G) = A – N + 2*<br>*V(G) = 5 – 5 + 2*<br>*V(G) = 2* | *V(G) = P + 1*<br>*V(G) = 1 + 1*<br>*V(G) = 2* |

La complejidad ciclomática es igual a 2, lo que significa que existen 2 posibles caminos linealmente independientes y representa el mínimo número de casos de prueba para el algoritmo, a continuación, se muestran los caminos existentes.

*Tabla 7: Caminos. Fuente: elaboración propia.*

| # | **Caminos** |
|---|---|
| 1 | 1245. |
| 2 | 135. |

El próximo paso es ejecutar los casos de pruebas para cada camino y se compara con los resultados esperados, verificando que las instrucciones se ejecuten por lo menos una vez. A continuación, se muestran los casos de prueba para los caminos básicos identificados.

*Tabla 8: Caso de prueba para el camino 1. Fuente: elaboración propia.*

| Caso de prueba para el camino básico 1 | |
|---|---|
| Descripción | Añadir un nuevo contacto. |
| Condición de ejecución | Estar autenticado en el sistema. |
| Entrada | Datos correctos del contacto nuevo. |
| Resultado esperado | • Adicionar un nuevo contacto.<br>• Mostrar la lista de contactos |

*Tabla 9: Caso de prueba para el camino 2. Fuente: elaboración propia.*

| Caso de prueba para el camino básico 2 | |
|---|---|
| Descripción | Añadir un nuevo contacto. |
| Condición de ejecución | Estar autenticado en el sistema. |
| Entrada | Datos incorrectos del nuevo contacto. |
| Resultado esperado | • No se añade el contacto.<br>• Mostrar formulario. |

### 3.3.3 Pruebas de rendimiento

Las pruebas de rendimiento ayudan a mantener los sistemas correctamente y a corregir los defectos antes de que lleguen a los usuarios del sistema. Ayudan a mantener la eficacia, la capacidad de respuesta, la escalabilidad y la velocidad de las aplicaciones cuando se comparan con los requisitos empresariales (martinekuan 2023).

Las **pruebas de carga** miden el rendimiento del sistema a medida que aumenta la carga de trabajo. Identifica dónde y cuándo se interrumpe la aplicación para que pueda corregir el problema antes de enviarlo a producción. Para ello, prueba el comportamiento del sistema en cargas típicas y pesadas.

A diferencia de las pruebas de carga, que garantizan que un sistema puede controlar lo que se ha diseñado que controle, las **pruebas de estrés** se centran en sobrecargar el sistema hasta que se interrumpa. Una prueba de estrés determina la estabilidad de un sistema y su capacidad de resistir los aumentos extremos de la carga.

**Resultados de la prueba:**

Para aplicar estas pruebas se tomó como muestra una cantidad de 30 usuarios. Las pruebas se desarrollaron con el apoyo de la herramienta Apache JMeter 5.5, simulando el entorno de producción para obtener los datos más cercanos al comportamiento y resultado real.

**Hardware:**

- Microprocesador: Intel(R) Celeron(R) CPU 1.60 GHz

- Memoria: 4GB RAM
- Tipo de Red: Ethernet 10/100 Mbps

**Software:**

- Sistema Operativo: Windows 10, Arquitectura de 64 bit

La prueba define treinta hilos de concurrencia, los cuales simulan treinta usuarios accediendo concurrentemente. Se simularon un total de 120 peticiones a cuatro direcciones del módulo en el servidor. En la Tabla 10 se puede observar los resultados obtenidos por el sistema.

*Tabla 10: Reporte resumen de JMeter.*

| Etiqueta | # Muestras | Media | Mín | Máx | Desv. Estándar | % Error | Rendimiento | Kb/sec | Sent KB/sec | Media de Byt... |
|---|---|---|---|---|---|---|---|---|---|---|
| login | 30 | 609 | 29 | 3009 | 660,16 | 0,00% | 30,4/min | 1,43 | 0,13 | 2893,0 |
| login_post | 30 | 5289 | 2459 | 8424 | 2061,40 | 0,00% | 27,9/min | 218,97 | 0,27 | 482809,6 |
| Insertar cont... | 30 | 723 | 243 | 1621 | 412,86 | 0,00% | 29,1/min | 14,72 | 0,10 | 31055,0 |
| Insertar cont... | 30 | 1946 | 616 | 4317 | 1090,72 | 0,00% | 28,9/min | 230,80 | 0,32 | 491032,2 |
| Total | 120 | 2142 | 29 | 8424 | 2255,51 | 0,00% | 1,8/sec | 446,38 | 0,79 | 251947,5 |

**Muestras:** cantidad de hilos utilizados para la URL.

**Media:** tiempo promedio en milisegundos para un conjunto de resultados.

**Min:** tiempo mínimo que demora un hilo en acceder a una página.

**Max:** tiempo máximo que demora un hilo en acceder a una página.

**%Error:** porciento de error de las respuestas de las peticiones.

**Rendimiento:** rendimiento medido en los requerimientos por segundo / minuto / hora. Kb/s

**Recibidos:** rendimiento medido en Kbyte por segundo.

### 3.3.4 Pruebas de caja negra

Las pruebas de caja negra, es una técnica de pruebas de software en la cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software.

En las pruebas de caja negra, se tienen en cuenta solamente las entradas y salidas del sistema, sin tener conocimientos de la estructura interna del programa de software. Para obtener el detalle de cuáles deben ser esas entradas y salidas, se basa únicamente en los requerimientos de software y especificaciones funcionales (pmoinformatica.com 2017).

**Partición de equivalencia**

La partición de equivalencia es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos de los que pueden derivarse casos de prueba. Un caso de prueba ideal descubre de primera mano una clase de errores (por ejemplo, procesamiento incorrecto de todos los datos carácter) que de otro modo podrían requerir la ejecución de muchos casos de prueba antes de observar el error general ( *Presman 2010*).

El diseño de casos de prueba para la partición de equivalencia se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana (Presman 2010).

*Tabla 11: Variables empleadas en el caso de prueba basado en el requisito "Añadir tarea".*

| **No** | **Nombre del campo** | **Clasificación** | **Valor Nulo** | **Descripción** |
|---|---|---|---|---|
| 1 | Nombre | Campo texto | No | Se inserta una cadena de texto, puede contener valores alfanuméricos. |
| 2 | Fecha | Campo fecha | No | Se selecciona la fecha en que se realizará la tarea. |
| 3 | Hora | Campo hora | No | Se selecciona la hora en que se realizará la tarea. |
| 4 | Descripción | Campo texto | Si | Se proporciona una breve descripción de la tarea. |

*Tabla 12: Caso de prueba correspondiente al requisito "Añadir tarea".*

| **Escenario** | **Descripción** | **V1** | **V2** | **V3** | **V4** | **Respuesta del sistema** | **Flujo central** |
|---|---|---|---|---|---|---|---|
| | | V | V | V | V | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| EC 1.1 Introducir datos correctos | El usuario introduce los datos correspondientes a una tarea | Reunión | 11-12-2023 | 11:00 | Reunión sobre el proyecto Platel | El sistema guarda la tarea y redirecciona al calendario. | El usuario rellena los campos correctamente y da clic en el botón aceptar. |
| EC 1.2 Campos vacíos | El usuario deja uno o varios campos vacíos | V<br>Vacío | V<br>11-11-2023 | V<br>Vacío | V<br>Vacío | El sistema muestra un mensaje anunciando que debe llenar los campos nombre y hora. | El usuario completa el formulario quedando campos vacíos. |
| EC 1.3 Introducir datos incorrectos. | El usuario introduce uno o varios valores incorrectos. | V<br>Visitar cliente | V<br>31-11-20033 | V<br>16:30 | V<br>Visitar cliente del nuevo proyecto | El sistema muestra un mensaje anunciando “Debe ingresar un valor válido. El campo está incompleto o contiene una fecha no válida.” | El usuario introduce uno o varios valores incorrectos y da clic en el botón aceptar. |

**Resultados de la prueba:**

Con el objetivo de validar los requisitos funcionales se realizaron tres iteraciones de pruebas al sistema. En la Ilustración 11 se muestran los resultados obtenidos en cada iteración de prueba, así como la corrección de cada uno de los errores:

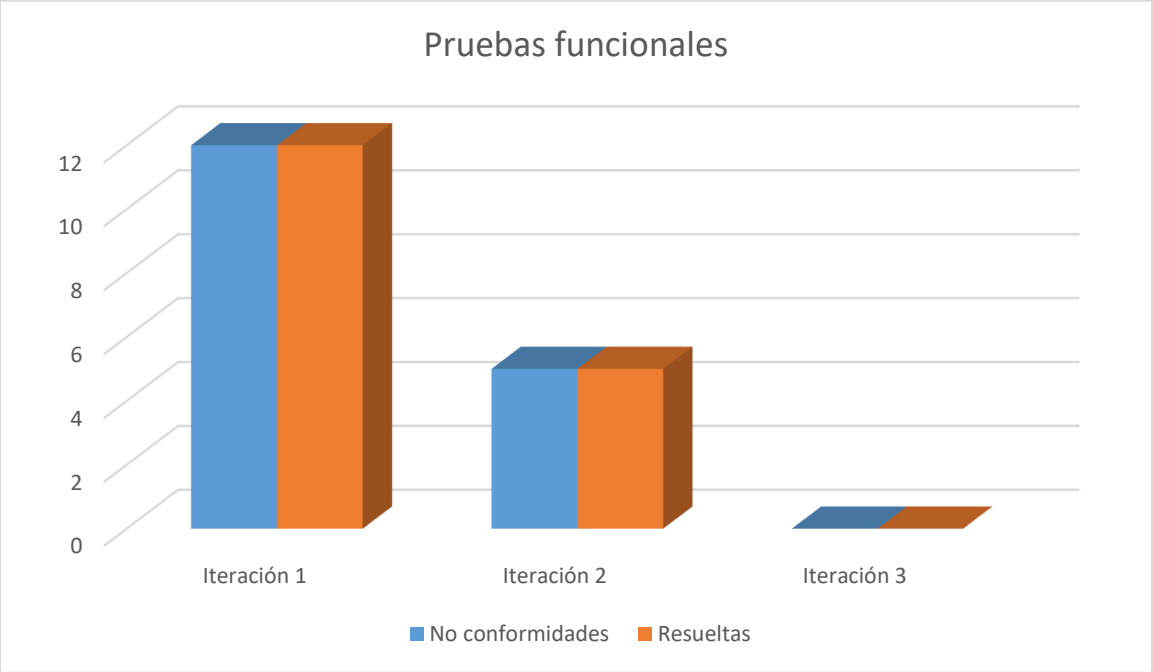



*Ilustración 11: Gráfico de los resultados obtenidos de las pruebas funcionales. Fuente: elaboración propia.*

Se detectaron en las 3 iteraciones un total de 17 no conformidades. En la primera iteración se arrojaron 12 no conformidades las cuales fueron resueltas, en la segunda se obtuvieron 5 no conformidades que fueron solucionadas y en la última iteración no se encontró ninguna no conformidad.

Entre las principales no conformidades detectadas se encontraron errores ortográficos, errores en validaciones de los campos de los formularios, no se mostraban los datos en el calendario, problemas con el campo fecha.

**Conclusiones del capítulo**

En el presente capítulo se realiza el diagrama de despliegue el cual muestra la arquitectura del sistema y el diagrama de componente con el propósito de visualizar los componentes del sistema y sus relaciones. Se establece una estrategia de prueba para validar las funciones de la aplicación, donde se efectuaron pruebas a nivel de unidad, de rendimiento y las funcionales.

# CONCLUSIONES GENERALES

Una vez culminada la investigación, considerando los resultados descritos en el informe, la necesidad y el objetivo planteado por la investigación se arriba a las siguientes conclusiones:

- El análisis de diferentes conceptos y elementos teóricos relacionados con el objeto de estudio permitieron adquirir el conocimiento necesario sobre los centros de contactos y así identificar la metodología de desarrollo, las herramientas y tecnologías seleccionadas para darle cumplimiento al objetivo general propuesto.
- El diseño de la propuesta de solución, guiado por la metodología de desarrollo AUP-UCI en su escenario 4, posibilitó identificar los requisitos y características del sistema, para lograr los resultados deseados.
- La implementación y el análisis de las pruebas unitarias, funcionales y de rendimiento realizadas al sistema desarrollado permitieron garantizar la correcta funcionalidad del sistema, además de comprobar que los requisitos planteados fueron cumplidos y que las necesidades del cliente fueron satisfechas.

# RECOMENDACIONES

Para la realización de proyectos futuros que permitan enriquecer la solución propuesta se recomienda:

- Incorporar una opción que permita sincronizar el sistema de gestión de contactos a la plataforma Platel y obtener la información de contacto de otros usuarios registrados.

# REFERENCIAS BIBLIOGRÁFICAS

:: ITESRC - Portal Académico - UML : Lenguaje de Modelado Unificado ::, sin fecha [en línea]. Recuperado a partir de : https://www.itesrc.edu.mx/portal/articles.php?id_art=1 [accedido 16 junio 2023].

(25) GRASP con ejemplos | LinkedIn, 2017 [en línea]. Recuperado a partir de : https://www.linkedin.com/pulse/grasp-con-ejemplos-guillermo-ojeda/?originalSubdomain=es [accedido 15 septiembre 2023].

ACHARYA, Durga Prasad, 2023. Explicación de las Pruebas Unitarias: Qué es, por qué es importante y cómo empezar. *Geekflare* [en línea]. 14 septiembre 2023. Recuperado a partir de : https://geekflare.com/es/unit-testing-guide/ [accedido 11 octubre 2023].

BACH, John, 2021. *Programación JavaScript: Javascript para Principiantes*. Independently Published. ISBN 9798716341081. Google-Books-ID: A7FUzgEACAAJ

BENITO, Maria, 2022. ¿Qué es bootstrap y cuáles son sus características? *FP Online* [en línea]. 26 enero 2022. Recuperado a partir de : https://fp.uoc.fje.edu/blog/que-es-bootstrap-y-cuales-son-sus-caracteristicas/ [accedido 15 septiembre 2023].

BERMÚDEZ, Yolagny Díaz, 2022. PostgreSQL, sistema de bases de datos relacional. *Revista TINO* [en línea]. 7 julio 2022. Recuperado a partir de : https://revista.jovenclub.cu/postgresql-sistema-de-bases-de-datos-relacional/ [accedido 17 junio 2023].

CANELO, Miriam Martínez, 2020. Qué son los Patrones de Diseño de software / Design Patterns. *Profile Software Services* [en línea]. 24 junio 2020. Recuperado a partir de : https://profile.es/blog/patrones-de-diseno-de-software/ [accedido 15 septiembre 2023].

Características clave de Visual Studio Code - TecnoBits ▷→, 2023 [en línea]. Recuperado a partir de : https://tecnobits.com/caracteristicas-clave-de-visual-studio-code/ [accedido 4 octubre 2023].

COPPOLA, Maria, 2023. Qué es JavaScript, para qué sirve y cómo funciona. [en línea]. 2023. Recuperado a partir de : https://blog.hubspot.es/website/que-es-javascript [accedido 3 octubre 2023].

Diagramas de clases del diseño, 2013*Cursos gratis* [en línea]. Recuperado a partir de : https://conocimientosweb.net/dcmt/ficha16595.html [accedido 6 noviembre 2023].

FONSECA, Letícia, 2022. Cómo crear un diagrama de clases [+Ejemplos]. *Venngage Blog* [en línea]. 27 junio 2022. Recuperado a partir de : https://es.venngage.com/blog/diagrama-de-clases/ [accedido 6 noviembre 2023].

Guión Visual Paradigm for UML, sin fecha*studylib.es* [en línea]. Recuperado a partir de : https://studylib.es/doc/7046646/guión-visual-paradigm-for-uml [accedido 16 junio 2023].

PRESSMAN, R. S. Ingeniería del Software. Un enfoque práctico, Séptima edición ed. 2010.

SOMMERVILLE, Ian, et al. Edition: Software Engineering. Instructor, 2011.

KUMAR, Chandan, 2021. 19 El mejor software de gestión de contactos para pequeñas y medianas empresas. *Geekflare* [en línea]. 1 septiembre 2021. Recuperado a partir de : https://geekflare.com/es/best-contact-management-software/ [accedido 13 julio 2023].

Las diez mejores plataformas de software para la administración de contactos | Herramientas para 2023, 2023 [en línea]. Recuperado a partir de : https://www.zendesk.es/sell/crm/contact-management-software/ [accedido 15 septiembre 2023].

LEE, Glenn, 2020. Tipos de pruebas de software: diferencias y ejemplos. *LoadView* [en línea]. 16 octubre 2020. Recuperado a partir de : https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/ [accedido 5 octubre 2023].

MARKETING, Yggdrasil, sin fecha. OCM SOFTWARE. *OCM SOFTWARE* [en línea]. Recuperado a partir de : https://www.ocmsoft.com/ [accedido 17 junio 2023].

MARTINEKUAN, 2023. Pruebas de rendimiento - Microsoft Azure Well-Architected Framework. [en línea]. 1 junio 2023. Recuperado a partir de : https://learn.microsoft.com/es-es/azure/well-architected/scalability/performance-test [accedido 8 octubre 2023].

MAZA, Miguel Ángel Sánchez, 2012. *Javascript*. Innovación Y Cualificación. ISBN 978-84-95733-18-4. Google-Books-ID: 3x09sewjaHIC

PINCAY, cesar javier perez, 2017. Ingenieria del software: prueba de la caja blanca y camino básico. *Monografias.com* [en línea]. 28 mayo 2017. Recuperado a partir de : https://www.monografias.com/docs113/ingenieria-software-prueba-caja-blanca-y-camino-basico/ingenieria-software-prueba-caja-blanca-y-camino-basico [accedido 11 octubre 2023].

PMOINFORMATICA.COM, 2017. Pruebas de caja negra: Ejemplos. [en línea]. 2017. Recuperado a partir de : http://www.pmoinformatica.com/2017/02/pruebas-de-caja-negra-ejemplos.html [accedido 26 octubre 2023].

¿Qué es Django y para qué se utiliza? ¡Descúbrelo! | Tokio, sin fecha [en línea]. Recuperado a partir de : https://www.tokioschool.com/noticias/que-es-django/ [accedido 16 junio 2023].

¿Qué es la prueba de caja blanca? Técnicas y ejemplos - Historiadelaempresa.com, 2022 [en línea]. Recuperado a partir de : https://historiadelaempresa.com/que-es-el-white-box-testing [accedido 11 octubre 2023].

Quiénes Somos - EasyCall - Software online para su Call Center, sin fecha*EasyCall* [en línea]. Recuperado a partir de : https://www.easycallcloud.com/es/quienes-somos/ [accedido 17 junio 2023].

*Rodríguez - Actualización de los roles de la metodología..pdf*, sin fecha. .

ROMANO, Natalie, 2022. ¿Qué es el Centro de Contacto como servicio (CCaaS)? | Blog de Avaya. *Avaya* [en línea]. 12 septiembre 2022. Recuperado a partir de : https://www.avaya.com/es/blogs/2021/09/what_is_ccaas/ [accedido 14 octubre 2023]. Locale: es

SCHOOL, Tokio, 2022. Estas son las características de Python: ¡domina el código! | Tokio. *Tokio School* [en línea]. 7 marzo 2022. Recuperado a partir de : https://www.tokioschool.com/noticias/caracteristicas-principales-de-python/ [accedido 15 septiembre 2023].

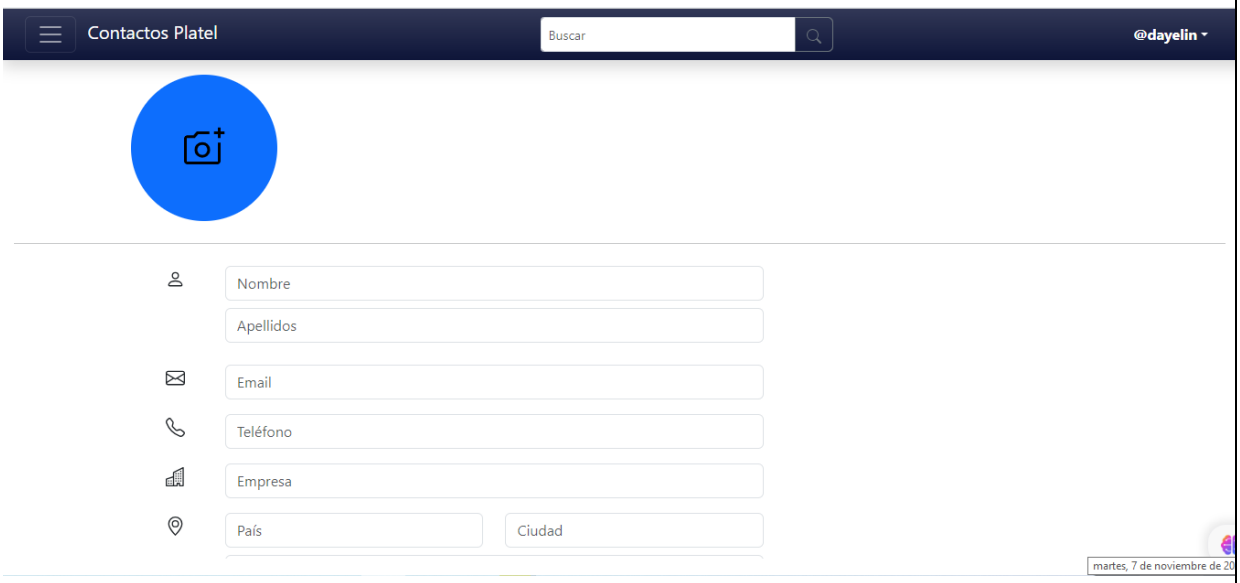
SIRIWARDHANA, Shalin, 2020a. Tutorial de Diagrama de Despliegue | ¿Qué es un Diagrama de Despliegue. *Blog de Creately* [en línea]. 26 octubre 2020. Recuperado a partir de : https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-despliegue/ [accedido 5 octubre 2023].

SIRIWARDHANA, Shalin, 2020b. Tutorial del Diagrama de Componentes | Guía Completa con Ejemplos. *Blog de Creately* [en línea]. 26 octubre 2020. Recuperado a partir de : https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-componentes/ [accedido 5 octubre 2023].

XETID, 2023. Portal Web de la XETID. *Portal Web de la XETID* [en línea]. 15 junio 2023. Recuperado a partir de : https://www.xetid.cu/es/novedades/platel-plataforma-de-videoconferencias-que-mejora-la-comunicacion-empresarial [accedido 15 junio 2023].

# ANEXOS

*Tabla 13: Historia de usuario Editar contacto. Fuente: elaboración propia.*

<table>
<tr><td colspan="2">Historias de usuario</td></tr>
<tr><td>Número:5</td><td>Usuario: Cliente</td></tr>
<tr><td colspan="2">Nombre de Historia: Editar contacto</td></tr>
<tr><td colspan="2">Prioridad del negocio: Alta</td></tr>
<tr><td>Riesgo en desarrollo: Alta</td><td>Iteración asignada: 1</td></tr>
<tr><td>Tiempo estimado: 4 días</td><td>Tiempo Real: 4 días</td></tr>
<tr><td colspan="2">Programador responsable: Dayelin Alayo Roll</td></tr>
<tr><td colspan="2">Descripción: Le permite al usuario editar la información de un contacto.</td></tr>
<tr><td colspan="2">Prototipo:<br> 
</td></tr>
</table>

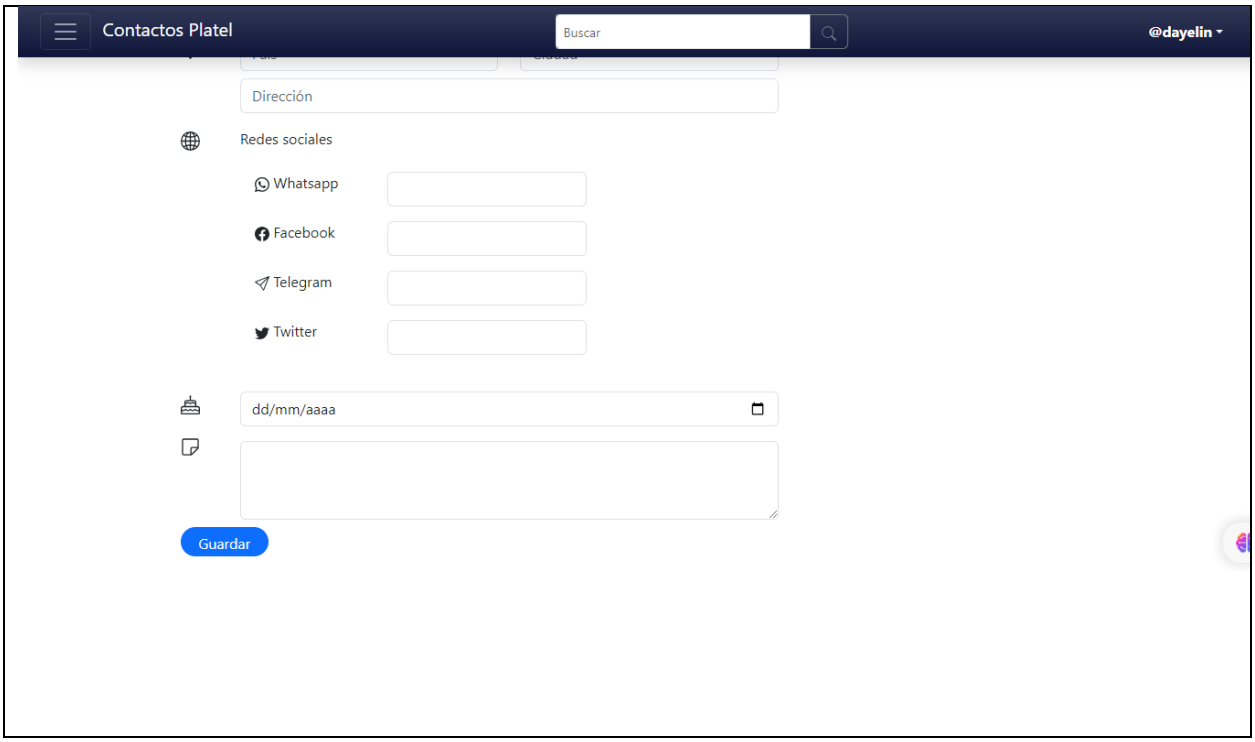



*Tabla 14: Historia de usuario Eliminar contacto. Fuente: elaboración propia.*

| Historias de usuario | |
|---|---|
| Número:6 | Usuario: Cliente |
| Nombre de Historia: Eliminar contacto. | |
| Prioridad del negocio: Alta | |
| Riesgo en desarrollo: Alta | Iteración asignada:1 |
| Tiempo estimado:5 días | Tiempo Real: 5 días |
| Programador responsable: Dayelin Alayo Roll | |
| Descripción: Le permite al usuario eliminar los contactos. | |

*Tabla 15: Historia de usuario Mostrar datos del contacto. Fuente: elaboración propia.*

| Historias de usuario | |
|---|---|
| Número:8 | Usuario: Cliente |
| Nombre de Historia: Mostrar datos del contacto. | |
| Prioridad del negocio: Alta | |
| Riesgo en desarrollo: Baja | Iteración asignada:1 |
| Tiempo estimado:3 días | Tiempo Real: 3 días |
| Programador responsable: Dayelin Alayo Roll | |
| Descripción: Le permite al usuario ver la información los contactos. | |
| Prototipo:  | |