UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 1

Título: Aplicación para dispositivos móviles sobre los recorridos de ómnibus de la Universidad de las Ciencias Informáticas

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

Autor:

Adriana Osuna Miranda

Tutores:

Ing. Laritza González Marrero

Ing. Eliodanis Maceo Rosales

Ing. Ivelisse Montero Jiménez

La Habana, mayo de 2022

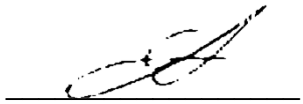
"Año 63 de la Revolución"

DELARACIÓN DE AUTORÍA

Declaro por este medio que yo Adriana Osuna Miranda, con carné de identidad 98071002493 soy la autora principal del trabajo de diploma “Aplicación para dispositivos móviles sobre los recorridos de ómnibus de la Universidad de las Ciencias Informáticas”, y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

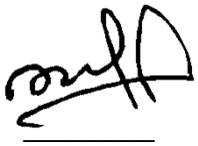
Y para que así conste, firmamos la presente en La Habana a los ____ días del mes de _____ del año _____.

Firma del autor
Adriana Osuna Miranda



Firma de tutora
Ing. Laritza González Marrero

Firma de tutor
Ing. Eliodanis Maceo Rosales



Firma de tutora

Ing. Ivelisse Montero Jiménez

"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica".

Aristóteles

AGRADECIMIENTOS

Primeramente, quiero agradecer a esta universidad por regalarme tantos momentos inolvidables en estos años de mi vida, a todas las personas que, con mucha dedicación, paciencia, y esmero me bridaron los conocimientos para formarme como profesional en las ciencias informáticas, a todas esas personitas que estuvieron conmigo en los momentos malos y buenos. Agradecer a mi Soslayo Capcioso por ser lo más lindo y sincero que he sentido en mucho tiempo y es Alexis Edgardo a quien quería agradecerle. A ti, por estar a mi lado todo este tiempo, por brindarme tu amistad, amor, apoyo incondicional, gracias por presentarme a la bella familia que tienes, sabes que los adoro. Agradecer también a una persona la cual con toda su alegría, amor, locura y malcriadez siempre me sacaba una sonrisa en los momentos más difíciles, gracias Dairon Canel, por estar ahí cada vez que te llamaba, por siempre compartir conmigo tus sentimientos, aprovecho para agradecerle a mi Pichy madre ¨Mami Maritza¨. Gracias a todas esas personas que siempre estuvieron dándome apoyo en especial a Kilmer, tan lindo conmigo, gracias al machi (Enier) tantos momentos en tan poco tiempo, Gracias a Mis coquetas, por todos los momentos vividos juntas, en especial a Daniela por su dedicación en cada momento de esta etapa… no me alcanzaría el tiempo para mencionarlos a todos, pero lo importante es que sepan que les estoy muy agradecida por todo su apoyo y de dedicación por prepararme para este momento tan especial y para la vida, ya que a todos los tengo en un lugarcito de mi corazón como mis mejores amigos. Gracias a toda mi familia por estar siempre pendiente de mí y mis estudios. Por último, quisiera agradecer a Daniel Pedroso, un millón de gracias, ya que me presentaste a

una persona la cual llego a mi vida para hacerla mejor. Se trata de Snayder Zarate. Gracias a ti amor por apoyarme en todo es te tiempo, por tus consejos, por aguantar mis malcriadeces, por sacarme una sonrisa todos los días, Te amo.

¿El por qué no mencioné específicamente a mis padres en los agradecimientos? Bueno, no los mencioné porque en realidad este trabajo es dedicado a ellos. Le dedico este trabajo a Florito y Nayansy, se los dedico por haber confiado en mí, por apoyarme incondicionalmente, por hacer lo posible y lo imposible porque su niñita tuviera lo necesario para poder terminar su carrera profesional. Por estar al tanto de cada cosa que me hiciera falta para mis estudios, por dedicar horas de trabajo y esfuerzo para que nunca me faltara nada y por todo el amor del mundo que me han dado. Gracias por hacerme una mejor persona. Los amo.

Resumen

Cada día, millones de personas utilizan los sistemas de información geográfica (SIG) en el gobierno, la industria y el mundo académico. Incluso las organizaciones más pequeñas contratan a profesionales de SIG con el fin de mejorar la calidad y la exactitud del trabajo realizado, y las ventajas de llevar esto a cabo son inconmensurables. El presente trabajo lleva a cabo una investigación sobre la utilización de los sistemas de información geográfica en el mundo de la telefonía móvil, con la finalidad de desarrollar una aplicación sobre la plataforma Android capaz de informar a los trabajadores de la universidad sobre los recorridos de los ómnibus de la misma, para así solucionar el problema existente respecto a la falta de información de los obreros a la hora de abordar el ómnibus que los trasladará desde y hacia la institución. La aplicación que lleva como nombre “Ómnibus UCI”, también será capaz de proporcionarle a los viajeros, la información referente a las paradas de cada ruta existente, lo cual les proporciona una mejor ubicación de su destino. El desarrollo de esta aplicación posibilita además un mejor análisis de los trabajadores en su decisión al abordar el ómnibus correcto. Para su desarrollo fue utilizada la metodología XP, utilizando la herramienta Visual Paradigm para el modelado y los lenguajes de programación Java y Python para la implementación. Se emplearon métodos de investigación teóricos y empíricos y para evaluar los resultados de la investigación; se realizaron pruebas unitarias y de aceptación.

Palabras clave: GPS, Sistemas de Información Geográfica, tecnología.

ÍNDICE DE CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

El sustancial despliegue que han tenido las Tecnologías de la Información y la Comunicación (TIC) hoy en día trae consigo una percepción diferente de la forma en que el ser humano se comunica con su entorno social, evidenciando que la distancia o ubicación geográfica ya no es un impedimento. Con el creciente desarrollo de las tecnologías, es posible que se tenga al alcance cualquier información que quizás en tiempos anteriores era inasequible.

Un sistema operativo para dispositivos móviles se define como el conjunto de programas cuya misión fundamental es la de gestionar los recursos del dispositivo y, en consecuencia, constituirá el soporte lógico que controla el funcionamiento del equipo físico. Los sistemas operativos para dispositivos móviles se enfocan principalmente en la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes formas de manejar la información en ellos (Alegsa, 2018).

En la actualidad, se utilizan diferentes sistemas operativos móviles como Windows, Linux, MacOS y Android. Uno de los últimos estudios realizados por Statista, revela que en 2019 la cuota de móviles con sistema operativo Android ascendía hasta el 83%. Apple se quedaba solo con el 13,9% y el otro 3,1% era para sistemas operativos móviles casi desconocidos (ProAndroid 2021). Android está basado en Linux, que es un núcleo de sistema operativo libre, gratuito y multiplataforma (Sacristán et.al, 2012).

Diseñado en un principio para dispositivos móviles, Android permite controlar dispositivos por medio de bibliotecas desarrolladlas o adaptados por Google mediante el lenguaje de programación Java. Es una plataforma de código abierto. Esto quiere decir, que cualquier desarrollador puede crear y desarrollar aplicaciones escritas con lenguaje C u otros lenguajes y compilarlas a código nativo de ARM[1] [API de Android] (Vilchez, 2009).

En el desarrollo de las aplicaciones para dispositivos móviles, una de las categorías que ha ido revolucionando en los últimos años es la de los Sistemas de Información Geográfica (SIG), ya que estos constituyen una herramienta utilizada para la toma de decisiones, permitiendo al usuario decidir cómo manejar el territorio analizado (Aleph, 2021).

Disímiles son las aplicaciones que se han desarrollado en los últimos años relacionadas con el estudio de los sistemas de información geográficas en Cuba, debido a que estos sistemas son una integración organizada de *hardware*, *software* y datos geográficos diseñada para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográfica referenciada, con el fin de resolver problemas complejos de planificación y gestión geográfica.

[1] (Advanced RISC Machine), por sus siglas en inglés, es una arquitectura "liviana" y económica de procesamiento.

El Sistema de Posicionamiento Global (GPS) se utiliza para ubicar dispositivos móviles en cualquier punto de la tierra, con una precisión de pocos metros, 15 metros en el 95% del tiempo. En caso de que el sistema WAAS/EGNOS/MSAS[2] esté activado, la precisión del sistema asciende de 1 a 2 metros. Para obtener la posición del receptor, éste debe estar al aire libre y, a su vez, debe rastrear las señales de 4 o más satélites para que estos estimen su posición mediante la técnica de la *trilateración*[3] (Legarretaetxebarria, 2011).

Por la necesidad existente en Cuba de una revolución informática, así como el bloqueo económico impuesto por los Estados Unidos y los elevados precios en la compra de software privativos; Cuba plantea la necesidad imperiosa de informatizar la sociedad cubana. Es muy importante que todas las empresas y procesos sean sustentables para la economía del país, en concordancia con el desarrollo de este proceso de informatización y como parte de la ejecución de una política orientada a alcanzar la seguridad, invulnerabilidad e independencia tecnológica (Manso et.al 2016).

Es importante también, tener en cuenta, que actualmente el tema de la movilidad de las personas cada vez toma más fuerza y a su vez, las aplicaciones informáticas que le permiten al usuario realizar disímiles operaciones.

Situación problemática

Las tecnologías y sistemas de localización han avanzado en los últimos años debido a la manifestación del uso de los dispositivos móviles. Con los sistemas de localización actuales se ha podido llevar a cabo diversas herramientas para empresas en numerosos sectores profesionales, siendo el sector del transporte protagónico, tanto para el desarrollo socioeconómico de la nación cubana, como para la defensa del país, subrayó el presidente de los Consejos de Estado y de Ministros, Miguel Díaz-Canel Bermúdez, en el balance anual del Ministerio de Transporte, correspondiente al año 2018 (Granma, 2020).

Actualmente la Universidad de las Ciencias Informáticas tiene una plantilla de más de 1200 trabajadores externos. Para transportar a los empleados hacia y desde la entidad se cuenta con 40 ómnibus que cubren 17 rutas[4]. A causa de la disminución en la asignación de combustible

[2] Los satélites WAAS, EGNOS y MSAS son satélites geoestacionarios. No cambian su posición relativa en el espacio, como ocurre con el resto de los satélites GPS.

[3] Solo podrán ser empleados los métodos de trilateración si son conocidas las coordenadas de tres posiciones cardinales de tres estaciones bases cercanas al usuario, que normalmente serán: la estación base a la que el usuario está conectado y las dos estaciones base vecinas con el nivel de potencia recibida más alto. Mediante el uso de esta técnica se obtendrá la posición del usuario.

4 Encuesta realizada en la Dirección de Transporte de la Universidad.

ocasionada por la situación presentada por el país en el año 2018 fue necesaria la combinación de rutas y la transportación en días alternos para los empleados.

Debido a los cambios mencionados se originan ciertas inconformidades en los trabajadores, lo que repercute negativamente en su persona, puesto que pasan períodos extensos de espera por el ómnibus, desconociendo la mayor parte del tiempo cual ruta tomar, lo cual provoca que estos lleguen con deseos de descansar, ofuscados por equivocarse y dirigirse a un lugar lejano a su residencia teniendo como consecuencia un desánimo por ir a trabajar. A partir de lo planteado anteriormente sobre el tema del transporte en la universidad, se decide aplicar una encuesta a 80 trabajadores externos que hacen uso de dicho transporte para detectar posibles inconvenientes a la hora de trasladarse. Al finalizar la encuesta se pudo concluir que el ordenamiento del transporte presenta dificultades evidenciándose fundamentalmente en:

- La insuficiente información sobre las rutas e itinerarios.
- Falta de información sobre la disponibilidad de salida de los ómnibus.
- Escasos conocimientos sobre los horarios de salida/llegada de cada autobús.

Estas dificultades persisten en la actualidad lo que provoca confusión en los trabajadores sobre su decisión al abordar el ómnibus correspondiente. El contexto antes descrito origina el desarrollo de esta investigación que tiene como **problema de la investigación:** ¿Cómo facilitar la información sobre los itinerarios y paradas de los ómnibus de la Universidad de las Ciencias Informáticas a los trabajadores que utilizan móviles que operan sobre la plataforma Android?

Para llevar a cabo la investigación se tiene como **objeto de estudio:** Los Sistemas de Información Geográfica para dispositivos móviles, quedando enmarcada la investigación en el **campo de acción:** Los Sistemas de Información Geográfica para la gestión de los recorridos de los ómnibus y sus paradas sobre dispositivos móviles con sistema operativo Android.

Se define como **objetivo general:** Desarrollar una aplicación para dispositivos móviles que permita a los trabajadores de la Universidad de las Ciencias Informáticas conocer el recorrido de los ómnibus y las paradas establecidas. Con el fin de alcanzar el objetivo planteado se decidió establecer como **objetivos específicos:**

1. Elaborar el marco teórico de la investigación sobre soluciones informáticas asociadas al conocimiento de la información referente a los Sistemas de Información Geográfica para dispositivos móviles.
2. Diseñar una aplicación que permita la gestión de los recorridos y las paradas de las rutas de la universidad.
3. Implementar una aplicación que permita la gestión de los recorridos y las paradas de las

rutas de la universidad.

4. Evaluar la aplicación desarrollada.

Se propone desarrollar las **tareas investigativas** que se plantean a continuación:

1. Elaboración del marco teórico de la investigación para definir conceptos fundamentales que ayuden a la comprensión de la misma.
2. Realización del análisis del estado del arte de los sistemas de información geográfica para establecer semejanzas con las soluciones existentes e identificar características o funcionalidades similares a las del sistema a implementar.
3. Descripción de la metodología de desarrollo a utilizar para la descripción de los artefactos que se generan durante la implementación de la aplicación Ómnibus UCI para dispositivos que operan con sistema operativo Android.
4. Selección de las tecnologías necesarias para el desarrollo de la aplicación Ómnibus UCI para dispositivos que operan con sistema operativo Android.
5. Identificar los requisitos funcionales y no funcionales que la aplicación Ómnibus UCI debe cumplir.
6. Definir la arquitectura de la aplicación para la aplicación Ómnibus UCI.
7. Implementación de la aplicación.
8. Validación de la aplicación.

La investigación se rige por la siguiente **hipótesis:** El desarrollo de una aplicación para dispositivos móviles con Sistema Operativo Android facilitará un mejor acceso a la información sobre los itinerarios y paradas de los ómnibus de la Universidad de las Ciencias Informáticas.

Para obtener como **posibles resultados:**

1. Una aplicación para la gestión de los recorridos y las paradas de las rutas de la universidad.
2. Documentación asociada a la aplicación.
3. Un informe de investigación de ciclo completo de desarrollo de la aplicación.

Para llevar a cabo la elaboración de esta aplicación se tuvo presente los siguientes métodos científicos de investigación:

Métodos empíricos: Describen y explican las características fenomenológicas del objeto, representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional. (Hernández et.al 2020).

- **Análisis documental:** Permite recopilar y valorar todos los conocimientos ya establecidos para conformar el marco teórico.
- **Modelación:** Se emplea para la realización de los artefactos que requiere la metodología seleccionada.
- **Observación:** Se realiza una observación de las diferentes aplicaciones móviles existentes tanto en Cuba como a nivel mundial. Específicamente en aquellas que se ejecutan sobre la plataforma Android, designadas a ofrecer información al usuario sobre como trasladarse por las ciudades, haciendo uso del transporte público para estipular si algunas de las soluciones existentes darían respuesta al problema en cuestión.
- **Encuesta:** Se aplica ante la necesidad de probar una hipótesis o descubrir una solución a un problema, e identificar e interpretar, de la manera más metódica posible, un conjunto de testimonios que puedan cumplir con el propósito establecido. Se ejecuta con la finalidad de recoger información de los sujetos a partir de la formulación de preguntas a través de una entrevista personal, por correo o por teléfono, para hacer estimaciones de las conclusiones para la población a partir de los resultados obtenidos en una muestra (Torrado, 2004).

La encuesta fue aplicada a 80 trabajadores con el objetivo de conocer los criterios de los mismos sobre las deficiencias y dificultades que actualmente presenta el transporte de la universidad a la hora de trasladarse hacia y desde el centro. Para conocer el contenido de la encuesta ver **Anexo 1: Encuesta**.

- **Entrevista:** Consiste en la comunicación verbal entre el entrevistador y entrevistado con el fin de obtener datos. Debe ser previamente diseñada en función al tema de estudio, a la vez de ser planteada por el entrevistador, “Según Kerlinger (1997), la entrevista del tipo estructurada será mejor que los cuestionarios autoadministrados para sondear el comportamiento de las personas, sus intenciones, sus emociones, sus actitudes y sus programas de comportamiento” (Quispe, 2011).

Se realizaron entrevistas no estructuradas a 5 expertos que trabajan en la base de transporte UCI para obtener información actualizada sobre los procesos que se estaban llevando a cabo para el mejoramiento del transporte en la universidad. La utilización de este método permite la obtención de información necesaria para el desarrollo de los requerimientos del sistema. Para conocer el contenido de la entrevista ver **Anexo 2: Entrevista**.

Métodos teóricos: Son aquellos que permiten revelar las relaciones esenciales del objeto de investigación, son fundamentales para la comprensión de los hechos y para la formulación de la hipótesis de investigación (Ocaña 2009).

- **Analítico – Sintético:** Hace posible el análisis de los procesos, documentos y teorías para la extracción de los elementos más importantes que se relacionan con los sistemas de información geográfica enfocados al desarrollo en Android. Además, para buscar y analizar la información acerca de las tecnologías, metodologías y herramientas a utilizar en el desarrollo del sistema.
- **Inducción-deducción:** Permite analizar indistintamente las funcionalidades de las aplicaciones móviles, a nivel mundial como nacional desarrolladas para el transporte, para poder determinar que ninguna satisface en su totalidad, las necesidades de la población a la que se le realizó la encuesta.

La presente investigación se encuentra estructurada en tres capítulos de la siguiente forma:

Capítulo 1: Fundamentación teórica sobre los Sistemas de Información Geográfica para la gestión de los recorridos de los ómnibus y sus paradas sobre dispositivos móviles.

En el desarrollo del capítulo se aborda en detalle todo lo relacionado con la fundamentación teórica que sustenta la presente investigación. Se evidencia el estudio realizado de los Sistemas de Información Geográfica (SIG) existentes tanto a nivel nacional como internacional. Además, se analiza la metodología de desarrollo, herramientas y tecnologías que se utilizan en el proceso de desarrollo de un SIG, se conforma la propuesta de tecnologías y herramientas a utilizar y se exponen los elementos analizados, como criterios de selección para poder elegir entre todas las opciones posibles.

Capítulo 2: Análisis y diseño de la aplicación Ómnibus UCI

En el capítulo se describe la propuesta de solución de acuerdo a la metodología ágil XP en sus fases de Exploración y Planificación y se especifican los requisitos funcionales y no funcionales, así como las historias de usuario, generando una visión para la creación de la aplicación y realizando un correcto análisis y diseño del módulo.

Capítulo 3: Implementación y evaluación de la aplicación Ómnibus UCI.

En el capítulo se describen los artefactos relacionados con la implementación de la aplicación, se diseñan y ejecutan las pruebas a realizar, con el objetivo de comprobar las funcionalidades y el código de la aplicación en los diferentes escenarios.

CAPÍTULO 1: Fundamentación teórica sobre los Sistemas de Información Geográfica para la gestión de los recorridos de los ómnibus y sus paradas sobre dispositivos móviles.

En el presente capítulo se dan a conocer los principales conceptos referentes al problema de investigación. Se evidencia el estudio realizado de los Sistemas de Información Geográfica existentes tanto a nivel nacional como internacional. Además, se analiza la metodología de desarrollo para el sistema y se examinan las tecnologías que se utilizan en el proceso de desarrollo de un SIG.

1.1 Definición de conceptos asociados a la investigación

A continuación, se enuncian los principales conceptos relacionados con un sistema de información geográfica para ayudar a una mejor comprensión del caso de estudio.

Tecnología

Se asocia particularmente con la innovación, la transformación de una idea en un producto, en un proceso productivo, o en nuevo enfoque o procedimiento para la organización social y que transcurre por una serie de etapas científicas, técnicas, comerciales y financieras necesarias para su desarrollo y comercialización con éxito (Guerrero Pupo et.al, 2004).

Teléfonos inteligentes (Smartphone)

Los dispositivos móviles actuales (Smartphone, tablets, phablets, etcétera) constituyen parte de la tecnología con la que se interactúa en la actualidad. Dispositivos como el Smartphone o 'teléfono inteligente' permiten acceder a Internet desde cualquier lugar. Entre las principales definiciones acerca del Smartphone, Quicios, Sevillano y Ortega (2013) señalan que se trata de un teléfono móvil que cuenta con un sistema de gestión de la información y características técnicas similares a una laptop; Yu y Conway (2012) indican que un dispositivo móvil como el Smartphone tiene las funciones básicas de un teléfono y las mismas capacidades de un computador, con el agregado de la movilidad (Figueroa, 2016).

Geolocalización

También denominada como georreferenciación. La geolocalización implica el posicionamiento que define la localización de un objeto en un sistema de coordenadas determinado, este proceso es generalmente empleado por los sistemas de información geográfica (ABC, 2016).

1.2 Sistema de información geográfica

Un SIG es una integración organizada de *hardware, software* y datos geográficos, diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas, la información geográficamente referenciada con el fin de solucionar dificultades complejas relacionadas con los procesos de planificación y gestión (Santovenia et.al 2009). Dicha tecnología puede ser utilizada para investigaciones científicas, la gestión de los recursos, gestión de activos, la arqueología, la evaluación del impacto ambiental, la planificación urbana, la cartografía, la sociología, la geografía histórica, el *marketing*[5], etcétera.

1.2.1 Componentes de un SIG

Para lograr el funcionamiento de un SIG se debe contar con ciertos componentes como son:

Datos

Los datos son la materia prima para trabajar con los Sistemas de Información Geográfica (SIG). Sin ellos, no es posible construir productos de información o mapas que nos ayuden a hacer el análisis y tomar las decisiones en la organización. Esos datos podrán venir de diferentes fuentes: sensores remotos, GPS, fotografías aéreas, archivos formatos *shapefile*[6], archivos CAD[7], archivos Excel, etcétera (Santovenia et.al 2009).

Software

Para el correcto análisis e interpretación de la información geográfica es necesaria la participación de un software SIG que tenga la potencia y funcionalidad de trabajar con información de este tipo. Hoy en día existen numerosos *softwares* SIG en el mercado que nos

[5] Conjunto de técnicas y estudios que tienen como objetivo mejorar la comercialización de un producto.
[6] Formato de archivo informático para el intercambio de información geográfica.
[7] Diseño asistido por ordenador (CAD), sistema de hardware y software utilizado por los diseñadores profesinales para diseñar y documentar objetos del mundo real.

ponen a disposición herramientas SIG para el tratamiento de la información geográfica (Santovenia et.al 2009).

Hardware

Para poder utilizar algunos de los *softwares* es necesario un ordenador o *hardware*. Dependiendo de las características de esta máquina, se obtiene un mayor o menor rendimiento a la hora de realizar el análisis. Dentro de las características del *hardware* a tener en cuenta para análisis de información geográfica con *software* SIG se deben incluir las siguientes (Santovenia et.al 2009):

- Sistema operativo: Windows, Mac, Linux.
- RAM[8]
- Disco duro
- CPU[9]: 64 o 32 bits.
- Tarjeta gráfica (para visualizaciones 3D)

Personas

Una vez que se tienen los datos y con qué analizarlos, se necesita saber cómo. Aquí es donde entran en juego los profesionales SIG. Y es que el profesional SIG, es un perfil muy cuestionado (y demandado) en los últimos años, ya que existen muchas tareas dentro de un análisis SIG, las cuales necesitan de uno o varios profesionales, incluso profesionales temáticos (Santovenia et.al 2009).

Procesos

Un SIG exitoso opera de acuerdo a un buen diseño de reglas de implementación y de negocios, que son los modelos y prácticas de operación únicas para cada organización. Al igual que en todas las organizaciones relacionadas con la tecnología sofisticada, las nuevas herramientas sólo se pueden utilizar con eficacia si se integran adecuadamente en toda la estrategia empresarial de la organización. Para hacer esto correctamente, se requiere no sólo de las inversiones necesarias en *hardware* y *software*, sino también en el reciclaje y / o contratación de

[8] Memoria de acceso aleatorio.
[9] Unidad central de procesamiento

personal para utilizar la nueva tecnología en el contexto de la organización adecuada (Santovenia et.al 2009).

En la actualidad, los Sistemas de Información Geográficos (SIG) han sido una herramienta esencial que ha abordado la naturaleza de la información geográfica. En un pequeño período, los SIG han tenido una buena aceptación tanto dentro de la propia geografía como en el conjunto de estudios y aplicaciones en el cual, el componente espacial y territorial es un aspecto fundamental en el análisis. El campo de aplicación de los SIG crece cada día más, utilizándose en numerosos escenarios de la vida cotidiana. La planificación urbana, la arqueología, la cartografía, la evaluación de las redes de servicio y el transporte son algunas de las ramas donde pueden ser fructíferos estos sistemas.

1.2.2 Sistema de Posicionamiento Global

El sistema de posicionamiento global (GPS) es un sistema global de navegación por satélite (GNSS) que permite determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión hasta de centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión. Fue desarrollado, instalado y actualmente operado por el Departamento de Defensa de los Estados Unidos, determinado como “Sistema de Navegación mediante Tiempo y Distancia” (*Navigation Satellite Timing and Ranging*; NAVSTAR, por sus siglas en inglés) y necesita, para su funcionamiento, de cuatro satélites.

El GPS funciona mediante una red de, como mínimo, 24 satélites alrededor del planeta, a una altura de aproximadamente 20 000 kilómetros, con órbitas distribuidas para que en todo momento haya al menos cuatro satélites visibles en cualquier punto de la geografía global (Porto, 2020).

El sistema utiliza el principio de triangulación a partir de la posición de cada uno de los satélites que forman la constelación de NAVSTAR (efemérides) y del tiempo requerido por una señal de radio emitida por el satélite en alcanzar un receptor en Tierra. El GPS fue diseñado para estimar posición (lat., long. y elevación) en mar, tierra y aire; velocidad y tiempo; así como navegar de un sitio a otro. En términos cuantitativos, esto se interpretó como un Sistema capaz de tener una raíz del error medio cuadrático en posición de 10m, en velocidad de 0.1m/s y en tiempo de 100 nanosegundos (Paredes, 2016).

1.3 Sistemas de información geográficos para dispositivos que operan sobre Android

El anuncio del sistema Android se realizó en 2007 junto con la creación de la *Open Handset Alliance*, un consorcio de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles. Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto (Rivera et.al, 2012) y avaló principalmente su desarrollo tras la compra de Android Inc. en 2005) y entre las compañías se pueden encontrar: de *software*, operadores, fabricantes de móviles (Motorola, Samsung, LG, HTC, Huawei y otros) o fabricantes de *hardware* (Torres, 2014).

Este sistema operativo permite programar aplicaciones empleando una variación de Java llamada Dalvik (o ART a partir de la versión 5.0 de Android) y proporciona todas las interfaces necesarias para desarrollar fácilmente aplicaciones que acceden a las funciones del teléfono (como el GPS, las llamadas, la agenda, etcétera) utilizando el lenguaje de programación Java. Su sencillez principalmente, junto a la existencia de herramientas de programación gratuitas, es la causa de que existan cientos de miles de aplicaciones disponibles, que extienden la funcionalidad de los dispositivos y mejoran la experiencia del usuario (Mentor, 2014).

1.4 Estudio de sistemas homólogos

Actualmente, con la avanzada repercusión de las tecnologías informáticas, existen varios sistemas a nivel internacional que poseen entre otras funcionalidades, brindar la información necesaria al usuario para transportarse en las ciudades, ya sea a través de la utilización del transporte público o de medios particulares con los que cuenta la misma, en dispositivos que operan sobre la plataforma Android.

El uso de estos sistemas ha aumentado exponencialmente en los últimos años, y como consecuencia, han pasado del total desconocimiento a la práctica cotidiana en el mundo de los negocios, en las universidades y en los organismos gubernamentales. Actualmente existe una gran variedad de aplicaciones encargadas de proporcionar a los usuarios la información necesaria para trasladarse por las ciudades. Por esta razón se hace necesario analizarlas, con el objetivo de definir si cubren las necesidades planteadas en la problemática descrita y si pueden ser adaptadas a la Universidad. En caso contrario, este estudio permitirá analizar todos los aspectos que puedan ser de utilidad para el desarrollo de una propuesta de solución, ajustada a las necesidades de la institución.

1.4.1 Andariego

Es una aplicación para teléfonos móviles con sistema operativo Android, posee referencias de la cartografía de toda Cuba y permite a los usuarios acceder a un programa localizador que proporciona, entre otros, la distancia entre La Habana y todos los municipios del país, además, ubica los centros de salud que se soliciten, lugares de alojamiento, tiendas, paradas de ómnibus, agencias de viajes, entre otras informaciones. Fue creada por GeoSí, una empresa estatal cubana del Grupo Empresarial GEOCUBA perteneciente al Ministerio de las Fuerzas Armadas, responsable de la creación y mantenimiento de la Cartografía Digital a nivel nacional, y del desarrollo y comercialización de soluciones informáticas aplicadas al medio geográfico (Cubadebate, 2016).

Principales características (Cubadebate, 2016).

- ✓ Selecciona las provincias que se deseen visitar, disponible por el momento solo para La Habana.
- ✓ Busca, localiza y visualiza la información de lugares de interés (breve descripción, datos de contactos, servicios, horarios, imágenes, videos y más).
- ✓ Permite localizar la posición del usuario y muestra el recorrido mientras se mueve haciendo uso del servicio brindado por el GPS.
- ✓ Gestiona rutas de vehículos entre dos o más lugares contenidos en la base de datos.
- ✓ Muestra la descripción del recorrido entre los lugares o puntos seleccionados, expresando las distancias entre los puntos y la distancia total.
- ✓ Disponible para anuncios estatales y de cuentapropistas, con precios competitivos.
- ✓ Muestra algunas paradas de los diferentes tipos de ómnibus con los que cuenta el transporte público en la capital.

1.4.2 Osmand

Es una aplicación, que permite el uso de un mapa en cualquier lugar que lo necesite sin tener una conexión de datos. La misma cuenta con el enrutamiento en línea y la navegación guiada, es de código abierto y está siendo desarrollada activamente. Todo el mundo puede contribuir a la aplicación para informar fallos, mejorar las traducciones, o la codificación de nuevas características. El proyecto se encuentra en un estado muy animado, de mejora continua por

todas estas formas de desarrollo y la interacción del usuario. El avance del proyecto también se basa en las contribuciones financieras para financiar el desarrollo, codificación, y prueba de nuevas funcionalidades. OsmAnd ofrece enrutamiento, con el guiado óptico y la voz, un coche, bicicleta y peatones. Las principales funcionalidades operan tanto en línea como fuera de línea. Los datos fuera de línea se basan en *OpenStreetMap* - el mapa wiki libre (OSM) (APPSAPK. Osmand, 2016).

Principales características (APPSAPK. Osmand, 2016).

- ✓ Realiza búsquedas de lugares de dirección, según el tipo (restaurante, hotel, gasolinera, museo), o mediante coordenadas geográficas.
- ✓ Permite visualizar la posición y orientación en el mapa.
- ✓ Muestra diferentes superposiciones como pistas GPX[10] gira / navegación y mapas adicionales con la transparencia personalizable.
- ✓ Visualiza opcionalmente el límite de velocidad, en caso de que se excedan notifica de esto al usuario.
- ✓ Los mapas incluyen a pie, senderismo y rutas en bicicleta, ideal para actividades al aire libre.
- ✓ Encaminamiento y modos de visualización especiales para ciclistas y peatones.
- ✓ Muestra de forma opcional las paradas de transporte público (autobús, tren), incluyendo los nombres de las líneas.
- ✓ Calcula la distancia y el tiempo aproximado existente entre puntos determinados por el usuario.
- ✓ Tiene cobertura de mapas muy aproximadas y con buena calidad en muchos países del mundo principalmente en Europa occidental.

1.4.3 SIG-Rutas

SIG-Rutas es un SIG sobre las rutas que brindan servicio de transportación a los trabajadores de la comunidad universitaria de la UCI. Este sistema está diseñado y desarrollado para que funcione sobre una plataforma web. Entre las funcionalidades que tiene se pueden mencionar la navegación (acercar, alejar, mover, recentrar, etcétera.), cálculo de áreas, medir distancias,

[10] Formato de intercambio GPS.

visualización de capas y personalización de rutas. Además, cuenta con un sistema de alerta y permite conocer la localización de direcciones, municipios, paradas y rutas. Además, brinda a los usuarios la posibilidad de conocer las paradas más cercanas a partir de diferentes criterios introducidos en el sistema (Ramirez et.al, 2011).

Principales características (Ramirez et.al, 2011)

- ✓ La velocidad de procesamiento de la información, la actualización y la recuperación dependen de la cantidad de información que tenga que procesar el sistema.
- ✓ La construcción de la aplicación funcionó bajo los conceptos de arquitectura cliente/servidor. Por tanto, el usuario final debe tener como requerimientos mínimos de software un navegador que cumpla con los estándares de la Word Wide Web Consortium (W3C, por sus siglas en inglés), se recomienda utilizar el Mozilla Firefox en su versión 3 o superior.
- ✓ Para los Servidores los sistemas operativos GNU/Linux, servidor web Apache v2.0 o superior, con módulo PHP 5 configurado con la extensión pgsql incluida.
- ✓ Postgre SQL v3.0 como Sistema Gestor de Base de Datos.
- ✓ En cuanto al hardware las PC's utilizadas por los clientes deben tener tarjeta de red, al menos 128 MB de memoria RAM, se requiere al menos 100 MB de disco duro y procesador 512 MHz.
- ✓ En el caso de las PC's servidoras deben contar con tarjeta de red, para el Servidor de Mapas mínimo de 2 GB de RAM y 2 GB de disco duro, para el Servidor de Base de Datos, 2 GB de RAM y 10 GB de disco duro en conjunto de un procesador 3 GHz.

1.4.4 Metro de Paris

Metro de Paris es una buena opción para moverse de un punto a otro, ya que hay numerosas líneas y bocas de metro por toda la ciudad (¡más de 300!). Las líneas de metro en París se diferencian por colores y están numeradas. Esta es la tercera red de metro más larga de Europa, después del Metro de Londres y de Metro de Madrid (PARIS ANDO, 2020).

Visitar París en Metro es la aplicación oficial de la RATP[11]. Se encuentra totalmente traducida a varios idiomas (ruso, brasileño, chino, inglés, alemán, neerlandés, italiano, español y japonés),

[11] Administración autónoma de Transportes Parisinos.

está diseñada para todos los usuarios que deseen visitar París y transitar la ciudad en metro. La RATP incluye el metro, el ferrocarril, el tranvía y el autobús. La misma da acceso a una multitud de servicios para desplazarse fácilmente por París y descubrir los lugares más hermosos de la capital. La mayoría de las funcionalidades son accesibles incluso en modo sin conexión para que pueda disfrutar de la capital sin consumir los megas de Internet de su móvil (Jacobson et.al, 1999).

Principales características (Marroquín et.al, 2013)

- ✓ Calcula itinerarios desde y hacia las estaciones de metros y los lugares turísticos catalogados.
- ✓ Planifica fácilmente traslados desde las estaciones y los aeropuertos de París, gracias a la información disponible: mapas e itinerarios, estaciones comunicadas por los transportes públicos, horarios, tarificación, entre otros.
- ✓ Encuentra los planos de la red férrea sin conexión una vez haya sido descargado el mapa de París.
- ✓ La última versión (1.6.12.4), determina ubicación aproximada según la red y una ubicación precisa a partir del GPS.
- ✓ Lee la configuración del servicio de Google.
- ✓ Impide que el dispositivo entre en modo suspensión.
- ✓ Requiere de una versión de Android 2.2.3 o superior a esta.
- ✓ Tamaño de la misma es de 32 MB.

1.4.5 Metro de Madrid

Es el principal sistema de transporte de la ciudad y el más popular. Se compone de 12 líneas y otras tres líneas adicionales de Metro Ligero, un tranvía que va a nivel de calle. En total, el metro de Madrid suma 301 estaciones y 294 kilómetros de vías.

Metro de Madrid es una aplicación de uso sencillo e inmediato a través de la cual se puede descubrir: el mapa actualizado de la totalidad de la red, el plano turístico, la información de las líneas con las correspondencias y servicios de la estación y el estado de la circulación en tiempo real, toda la información de servicios de la estación que selecciones, y además, al poder localizar por GPS, proporciona información sobre la estación más próxima a la ubicación, permitiendo elegir una estación de destino o la estación más próxima al destino elegido recomendándote el

trayecto más idóneo con menos estaciones o con menos trasbordos. También indica el tiempo aproximado del trayecto y permite planificar tus desplazamientos introduciendo directamente una estación de Origen y otra de Destino pudiendo seleccionar la opción de menos estaciones o de menos trasbordos e indicándote el tiempo que se tarda en realizarlo. Permite consultar abonos, tarifas y horarios, así como la información de contacto (Metro Madrid, 2016).

Principales características (Google Play, 2016)

- ✓ Obtiene ubicación aproximada a partir de la red.
- ✓ Obtiene ubicación precisa a partir del GPS y de la red.
- ✓ Realiza llamadas directamente a números de teléfonos.
- ✓ Impide que el dispositivo entre en modo suspensión mientras se ejecuta en primer plano.
- ✓ Requiere de una versión de Android 2.2 o superior.
- ✓ En su versión 1.13 actualizada el 7 de abril del 2015 su tamaño es de 54 MB.

1.4.6 GPS Navigation & Maps Sygic

GPS Navigation & Maps Sygic es la aplicación de navegación sin conexión más instalada del mundo y se basa en la tecnología de TomTom Maps. La misma, brinda la posibilidad de disfrutar de acceso gratuito permanente a: mapas de TomTom sin conexión, puntos de interés (PDI), planificador de rutas y actualizaciones de mapas gratuitas. Los mapas de alta calidad de TomTom y otros proveedores se almacenan en su dispositivo Android o en la tarjeta de memoria, para que puedan ser usados sin conexión a Internet. Además, cuenta con una versión *Premium* que permite disfrutar siempre de: mapas 3D, navegación con instrucciones de voz giro a giro, instrucciones de carriles, advertencias de límite de velocidad y visión de los cruces con flechas indicadoras de carriles en los cruces complicados (Sygic, 2016).

Principales características (Sygic, 2016)

- ✓ Servicio de tráfico para evitar retrasos en los desplazamientos diarios.
- ✓ Avisador *Premium* de radares online con 300 000 ubicaciones de cámaras móviles cada mes.
- ✓ El Head Up Display (HUD) proyecta la navegación GPS en el parabrisas.
- ✓ Muestra zonas urbanas y rurales en 3D para facilitar la orientación.
- ✓ Cuenta con gráficos optimizados para tabletas y pantallas de alta definición.

- ✓ Permite la integración con el sistema de audio del vehículo ya sea a través del cable o mediante una conexión entre el dispositivo y el vehículo por bluetooth.
- ✓ Se encuentra disponible, en su versión 15.6.7 para Android 4.0 o superior (*Ice Cream Sandwich*, API[12] 14), la cual fue actualizada por última vez el 11 de febrero de 2016.

1.5 Análisis de las soluciones similares.

A modo general, estas aplicaciones informáticas son de uso liberado para los usuarios, pero no ocurre así con el código fuente de las mismas, por lo que no pueden ser reutilizadas y adaptadas al problema existente.

Es necesario destacar que los sistemas analizados, dotan de información al usuario en cuanto a: planificación de traslados dentro de la ciudad, ubicación actual en el mapa, visualización de puntos de interés, entre otras. En su totalidad brindan un amplio volumen de información acerca de las ciudades y del transporte, pero aún no existe ningún sistema de este tipo, que mediante el uso específicamente de los mapas de Cuba pueda brindar toda la información requerida a los trabajadores de la universidad que se trasladan en La Habana. En consecuencia, hacer uso de uno de los sistemas anteriormente mencionados, no facilitaría la toma de decisiones al usuario respecto a la travesía a seguir para llegar a un sitio determinado una vez que salga de la institución o para poder llegar a ella, haciendo uso de las redes de ómnibus de la UCI y paradas de la ciudad.

A pesar de que varias de las aplicaciones mencionadas con anterioridad, tengan ya actualizadas las paradas y rutas en las que hacen estacionamiento los ómnibus, no son capaces de ofrecer información real y actualizada sobre la red del transporte de la universidad, con excepción del SIG-Rutas, que es el único que se dedica al propio problema de la UCI brindando servicio de transportación a los trabajadores de dicha comunidad. Aun así, fue diseñado y desarrollado para que funcione solamente sobre una plataforma web y en la red del campus universitario, lo cual en la realidad objetiva del trabajador fuera de la universidad este sistema no está al alcance. Sin embargo, cada una de ellas contribuyó a una mejor comprensión de las principales funcionalidades que debe cumplir un aplicativo de este tipo.

[12] Interfaz de programación de aplicaciones.

A continuación, se muestra una tabla resumen de las principales características que se tomaron en cuenta para el estudio de las soluciones similares. Las características a evaluar son:

SO Android: Se analiza de las soluciones encontradas si son operan sobre el sistema operativo Android o no.
Conectividad: Se analiza si necesitan conexión a Internet.
Disponibilidad: Se analiza si están disponibles para los usuarios actualmente.
Código accesible: Se analiza si se puede acceder al código fuente.
Gestión de rutas y paradas: Se analiza si gestionan información relacionada a las rutas y paradas de los ómnibus.
Transporte UCI: Se analiza si brindan información sobre los recorridos de los ómnibus de la UCI.

Tabla 1: Características de las soluciones analizadas

Características / Sistemas	SO Android	Conectividad	Disponibilidad	Código accesible	Gestión de rutas y paradas	Transporte UCI
Andariego	Si	No	Si	No	Si	No
Osmand	Si	No	Si	No	Si	No
SIG-Rutas	No	Si	No	No	Si	Si
Metro de Paris	Si	Si	Si	No	Si	No
Metro de Madrid	Si	No	Si	No	Si	No
GPS Navigation & Maps Sygic	Si	No	Si	No	Si	No

1.6 Metodología de software

Para asegurar el éxito durante el desarrollo de software no es suficiente contar con notaciones de modelado y herramientas, hace falta un elemento importante: la metodología de desarrollo. Una metodología de desarrollo de software tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. En la actualidad hay dos grupos de metodologías para el desarrollo de software. Las metodologías tradicionales, que establecen un rigor en el proceso de desarrollo de software, dándole importancia al seguimiento y la planificación predictiva, herramientas, documentación extensiva y negociación contractual. Luego surgen las metodologías ágiles como reacción de la filosofía utilizada en las metodologías tradicionales. Los métodos ágiles están basados en entregas frecuentes de versiones de software funcionales, con mayor relevancia a la planificación adaptativa, colaboración con el cliente y respuestas ante los cambios inherentes al desarrollo de software. (Boaventura José et al., 2016).

La **Tabla 1** muestra las principales diferencias entre las metodologías ágiles y las tradicionales (Gaitan, 2016).

Tabla 2: Diferencias entre Metodologías Ágiles y Tradicionales

Metodologías Ágiles	Metodologías Tradicionales
Especialmente preparadas para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos

Pocos artefactos	Más artefactos
Pocos roles	Más roles

1.6.1 Metodología ágil

Una vez analizada la comparación anterior, se decide utilizar una metodología ágil para la presente investigación. Existen dos razones principales que justifican esta selección. En primer lugar, la flexibilidad de la metodología, lo que posibilita su adaptación de acuerdo a la realidad de cada equipo y proyecto. En segundo lugar, la poca documentación que genera. Este elemento es fundamental para un equipo de desarrollo de una persona, donde esta debe hacer todo el flujo de trabajo ingenieril, además del desarrollo del componente de visualización propuesto. La mayoría de los equipos ágiles exitosos han adaptado prácticas ágiles de distintas metodologías para generar un proceso de desarrollo propio que se ajusta a sus necesidades (Orjuela Duarte & Rojas C, 2008).

1.6.2 Metodología XP

Entre las principales metodologías ágiles se encuentra la **PROGRAMACIÓN EXTREMA (XP).** Según Bautista Q (2012) la Programación Extrema o *Extreme Programing*, es un enfoque de la ingeniería de software formulado por Kent Beck, se considera el más destacado de los procesos ágiles de desarrollo de software. Al igual que estos, la programación extrema se diferencia de los métodos tradicionales principalmente en que presenta más énfasis en la adaptabilidad que en la previsibilidad. El proceso de Programación Extrema (XP por sus siglas en inglés) se basa en la perspectiva orientada a objetos como paradigma y contiene un conjunto de reglas y prácticas que suceden en el contexto de cuatro actividades estructurales: planeación, diseño, codificación y pruebas (Ceballos, 2015).

Como resultado del análisis realizado sobre las metodologías y la profundización sobre XP se decide que para el desarrollo de este proyecto se utilice una versión reducida de la metodología ágil XP. El uso de esta metodología supone, para muchos teóricos, una aproximación a la calidad óptima del producto, pues durante el ciclo de vida del software, ocurren cambios naturales. Mientras más cambios haya, puede que más cerca estemos del mejor resultado que nuestro

cliente espera. Teniendo en cuenta lo anteriormente planteado, este cambio constante en el proyecto se llega a considerar como favorable. Y si logramos aplicar una manera dinámica de gestionarlos, mejor. Esta forma llega a ser conocida como metodología XP.

- **Actividades estructurales de XP**

Planeación: El primer paso de cualquier proyecto que utiliza la metodología XP es definir las historias de usuario (HU). La actividad comienza escuchando al cliente para obtener requerimientos que permiten que los miembros técnicos del equipo entiendan el contexto del negocio y crear las historias del usuario (HU).

Diseño: La metodología XP sugiere los diseños simples y sencillos para representar tareas con elevado nivel de complejidad. XP estimula el uso de las tarjetas Clase Responsabilidad Colaborador, estas identifican y organizan las clases relevantes para los requerimientos del sistema, mecanismo eficaz para pensar en el software en un ámbito orientado a objetos. Además, es el único producto del trabajo de diseño que se genera como parte del proceso de desarrollo de software, según lo planteado por la metodología XP.

Codificación: Después de elaboradas las HU y realizado el trabajo de diseño preliminar, el equipo no comienza la codificación, sino que desarrolla una serie de pruebas unitarias a cada una de las Historias de Usuario que se van a incluir en la entrega en curso. La prueba unitaria es una manera de probar el correcto funcionamiento de un módulo de código. Esto ayuda para que cada parte o módulo funcione correctamente e independientemente. La codificación del proyecto debe respetar un estándar de codificación. Esto se considera una buena práctica que mantiene el código consistente y facilita su comprensión y escalabilidad.

Pruebas: Uno de los pilares de la metodología XP es el uso de pruebas para comprobar el funcionamiento del código a medida que se va implementando. Las pruebas unitarias que se crean deben contemplar el uso de estructuras que permitan su automatización. Las llamadas pruebas de aceptación son definidas por el cliente y orientadas a las características y funcionalidades que debe cumplir el sistema.

1.7 Tecnologías utilizadas para desarrollar la aplicación Ómnibus UCI

El proceso de desarrollo de software se apoya en el uso de diferentes lenguajes y herramientas, las cuales, unidas a la metodología seleccionada, conforman el ambiente de desarrollo de un sistema. A continuación, se detallan las seleccionadas en la presente investigación.

1.7.1 Lenguaje y herramienta para el modelado de la solución

Para modelar las entidades se emplea el **Lenguaje Unificado de Modelado** (UML)**,** por sus siglas en inglés, *Unified Modeling Language* en la versión 8.0. UML es un lenguaje utilizado para especificar, visualizar, construir y documentar artefactos de un sistema de software. Está constituido por un conjunto de diagramas y es necesario contar con todas esas representaciones dado que cada una se dirige a cada tipo de persona implicada en el sistema. Un modelo UML indica qué es lo que se supone que deba hacer el sistema, no como lo hará (Visual Paradigm, 2015).

UML facilita el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan. Por las características antes mencionadas y debido a que el equipo de desarrollo cuenta con experiencia en la utilización del lenguaje, se decide emplearlo.

Herramienta CASE

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE) por sus siglas en inglés, *Computer Aided Software Engineering,* tienen como función aumentar la productividad en el desarrollo de software reduciendo los costos en términos de tiempo y dinero (Franky, 2010).

Visual Paradigm for UML en la versión 5.0, como herramienta de modelado, ayuda a los equipos de desarrollo de softwares a capturar los requisitos correctos y transformarlos en diseños precisos, lo que ayuda a los desarrolladores a crear el software adecuado según los requisitos (Capterra, 2002).

Principales características de Visual Paradigm for UML (Pressman, 1998).

- Entorno de creación de diagramas UML 8.0.
- Lenguaje estándar, común a todo el equipo de desarrollo.

- Existen varias versiones compatibles con diferentes plataformas, usadas para necesidades específicas del equipo de desarrollo.
- Permite la integración con Android Studio.

Como demuestran las características antes mencionadas la herramienta Visual Paradigm for UML es un software potente por lo que es utilizada a nivel mundial. Además, posibilita la realización satisfactoria de los diagramas en dependencia del flujo de trabajo de la metodología que se utilice. Por tanto, en la presente investigación Visual Paradigm en la versión 5.0, será la herramienta case que permita el modelado de los artefactos ingenieriles de la metodología ágil XP, como es el caso de la confección del diseño de la base de datos y el diagrama del modelo del dominio.

1.7.2 Tecnologías para la implementación

Un **lenguaje de programación** es un lenguaje formal que permite controlar el comportamiento físico y lógico de un ordenador mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos de hardware y software existentes (Gregory, 1987).

Java versión 11

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995, desarrollado por James Gosling y sus compañeros de Sun Microsystems al principio de la década de los 90 (Meza, 2021).

Luego de realizar un examen se determinó utilizar como lenguaje de programación Java en la versión 11 para la implementación del sistema sobre los recorridos de ómnibus de la Universidad de las Ciencias Informáticas para dispositivos móviles que operan sobre la plataforma Android, teniendo en cuenta que es un lenguaje orientado a objetos de gran potencialidad, robustez, versátil, seguro y multiplataforma (corre en cualquier sistema operativo moderno).

Además, se puede obtener Java y gran cantidad de herramientas para trabajar con él de forma gratuita, siendo la mayor parte de su código libre y abierto. También por ser este el lenguaje nativo en el desarrollo de aplicaciones Android permitiendo la instalación de un plugin llamado *Android Development Tools* (ADT). Este plugin permite una configuración rápida de proyectos,

exportación de ficheros de instalación, depuración mediante herramientas del Android SDK (*Android Software Development Kit*) siendo esta herramienta utilizada por Android Studio.

Java también dispone de una extensa librería de clases en forma de paquete, estas librerías son concebidas como un conjunto de clases, que poseen una serie de métodos y atributos las cuales facilitan muchas operaciones (Meza, 2021). Permiten realizar la programación mucho más rápida y libre de errores a través de la reutilización de código (hacer uso de los métodos, clases y atributos que componen la librería). Además, pueden ser creadas según las necesidades del programador para hacer uso de ellas en el interior del proyecto. Muchas son las librerías y facilidades que proporciona Java, especialmente para el trabajo y manipulación de mapas, permitiendo emplear los servicios de navegación GPS por medio de Mapsforge[13].

Python versión 3.7

Python es un lenguaje de programación orientado a objetos, multiplataforma y es software libre. Fue creado a finales de los años 80 y principio de los 90 por Guido van Rossum, un programador holandés. Guido, como creador de este lenguaje, dictaminó que el código debe ser limpio y legible, haciéndolo simple sin ser limitado (Gutiérrez, 2016).

Tras el estudio realizado se decide usar Python en la versión 3.7 como lenguaje para el desarrollo del lado del servidor (Backend) ya que este es un lenguaje de alto nivel que contiene implícitas algunas estructuras de datos como listas, diccionarios, conjuntos y tuplas, que permiten realizar algunas tareas complejas en pocas líneas de código y de manera legible. Python no solo se creó para el desarrollo web, sino que también es compatible con el desarrollo completo y además, dispone, por medio del uso de bibliotecas, de herramientas para aprovechar las posibilidades concretas que le brinda cada plataforma, pero también es posible escribir programas evitando el uso de estas bibliotecas específicas posibilitando que esos programas funcionen indistintamente en cualquier ordenador.

[13] Proveedor de herramientas libres y abiertas que permiten la gestión de mapas.

1.7.3 Entorno de desarrollo integrado (IDE)

Un Entorno de Desarrollo Integrado, traducido del inglés *Integrated Development Environment* (IDE) es una herramienta informática que permite maximizar la productividad como programador, proporcionando facilidades para el desarrollo de software.

Un IDE usualmente está compuesto por un editor de código, herramientas de automatización y un depurador o *debugger*. La mayoría de los IDEs modernos cuentan con un completador de código inteligente. A veces se integra un sistema de control de versiones (como GIT), o varias herramientas para simplificar la construcción de una interfaz gráfica de usuario (GUI), por sus siglas en inglés *Graphical User Interface* (UNIR, 2021).

Android Studio versión 2020.3.1.22

Android Studio es un entorno de desarrollo integrado (IDE) multiplataforma de código abierto. Cuenta con capacidad de completamiento de código, ambiente flexible y rápido. Es un producto singular por la versatilidad que ofrece, y todas estas características están determinadas por la arquitectura con la cual fue diseñado. Android Studio utiliza una licencia de software libre Apache 2.0, programado en Java. Ha sido ratificado por Google como el IDE de programación oficial para desarrollar aplicaciones para su sistema operativo (Academia Android, 2016).

Principales características que incluye Android Studio (Academia Android, 2016)

- Integración de la herramienta Gradle encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de pruebas, compilación o empaquetado.
- Nueva interfaz específica para el desarrollo en Android.
- Permite la creación de nuevos módulos dentro de un mismo proyecto, sin necesidad de estar cambiando de espacio de trabajo para el manejo de proyectos.
- Posibilita el control de versiones accediendo a un repositorio desde el que se puede descargar Mercurial, Git, Github o Subversión.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
- Vista previa en diferentes dispositivos y resoluciones.

El entorno de desarrollo es imprescindible en la producción de un software y Android Studio en su versión 2020.3.1.22 nos brinda disímiles funcionalidades que pueden ser muy bien aprovechadas en el desarrollo de la aplicación Android, puesto que es puramente Android, o sea, creado para programar en Android, se actualiza constantemente, el rendimiento es mejor y más rápido que en otro entorno de desarrollo. También es más intuitivo, más fácil de usar (es el futuro). Otra característica que propició la utilización de Android Studio como IDE, es que el proyecto se hace menos engorroso y el código más ordenado, estructurado y con mejores sugerencias. Como se menciona anteriormente, es mejor para diseñar Interfaces y tiene integrada la herramienta Gradle y las mejores plantillas para empezar proyectos. Es importante mencionar que exporta .APK más fácil y tiene mayor facilidad para la creación de diferentes versiones de la misma aplicación, entre otras.

Visual Studio Code (VS Code) versión 1.44.2

Visual Studio Code es un editor de código fuente anunciado en abril de 2015 que permite trabajar con diversos lenguajes de programación, admite gestionar los propios atajos de teclado y refactorizar el código. Es gratuito, de código abierto y proporciona una utilidad para descargar y gestionar extensiones las cuales permiten personalizar y potenciar esta herramienta (Aitana, 2018).

Diversas son las funcionalidades que VS Code proporciona, desde herramientas de *Debug*[14] hasta opciones para actualización en tiempo real de nuestro código en la vista del navegador y compilación en vivo de los lenguajes que lo requieran. Características como las antes mencionadas son las que determinan la selección de este IDE en su versión 1.44.2 para desarrollar el servidor web que permita la gestión de los diferentes ómnibus y rutas que estos poseen.

[14] Depuración

1.7.4 Framework de desarrollo

Un *framework web*[15] según Gutiérrez, (2017) es un conjunto de componentes software que construyen un diseño reutilizable que facilita y agiliza el desarrollo de una aplicación Web robusta.

Se puede considerar como una aplicación web incompleta y configurable a la que se le pueden añadir nuevas funcionalidades para conseguir el comportamiento deseado por el grupo de programadores. Estas herramientas y funcionalidades pueden ser: librerías de clases, funciones, scripts, etcétera.

Django versión 3.1

Django, es un *Framework Web* de alto nivel escrito en Python, el cual nació en el año 2003, con un equipo de desarrolladores Web del diario *Lawrence Journal-World*, en Lawrence, Kansas (Estados Unidos.). Dicha unidad de programadores dejó de lado *PHP* para empezar a utilizar Python para desarrollar sus aplicaciones Web (Caldera, 2017). Puesto que Django fue diseñado en una empresa del ámbito de publicación de noticias en la web, es muy común para ese tipo de aplicaciones disponer de un sitio de administración en el cual se lleven las labores propias de dicha tarea como puede ser la gestión de usuarios, añadir, eliminar o modificar entradas de un determinado sitio, entre otras.

En cuanto a la seguridad, Django tiene activados mecanismos incluidos para proteger la base de datos, formularios y JavaScript. Es escalable, es decir, se puede utilizar el *framework* para un desarrollo sencillo, hasta uno mucho más complejo, ambos casos funcionarán de manera estable y con rapidez. Su interfaz para acceso a la base de datos y hacer consultas es sumamente buena. Además, es portable.

Al estar escrito en Python, se puede ejecutar en muchas plataformas como Windows, OS X, entre otras, dándole muchísima libertad al programador al momento de ejecutar las aplicaciones. Teniendo en cuenta dichas características antes mencionadas, se selecciona Django en la versión 3.1 como *framework* de desarrollo.

[15] Marco de trabajo

1.7.5 Sistema de gestión de base de datos

Un sistema gestor de bases de datos (SGBD) se puede definir como una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. Esta colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. (Burton et.al, 2012).

SQLite versión 3.0

Es un manejador de código abierto de bases de datos que combina una interfaz limpia de SQL y permite trabajar con poca memoria y con una velocidad bastante rápida, peculiaridades que son necesarias cuando se habla de entornos móviles. Soporta las características estándar de las bases de datos relacionales, como la sintaxis basada en SQL, transacciones y la elaboración de consultas. Debido a esto, cualquier desarrollador que haya trabajado con bases de datos sin importar el entorno, no encontrará una dificultad especial en trabajar con bases de datos locales en Android. Git tiene pequeña memoria y necesita de una biblioteca única para acceder a bases de datos, lo que lo hace ideal para aplicaciones de bases de datos incorporadas. También es multiplataforma, es decir, se ejecuta en varias plataformas y sus bases de datos pueden ser fácilmente portadas sin ninguna configuración o administración (Martín, 2015).

Teniendo en cuenta las características antes mencionadas y que Android incorpora todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, y entre ellas una completa API para llevar a cabo de manera sencilla todas las tareas necesarias se decide usar SQLite en la versión 3.0 como gestor de base de datos.

1.7.6 Herramienta de control de versiones

El control de versiones, también conocido como "control de código fuente", es la práctica de rastrear y gestionar los cambios en el código de *software*. A medida que los entornos de desarrollo se aceleran, los sistemas de control de versiones ayudan a los equipos de *software* a trabajar de forma más rápida e inteligente.

Git versión 2.32.0

Git es un sistema de control de versiones distribuido de código abierto desarrollado por Linus Torvalds, el creador de Linux. También permite a los desarrolladores descargar un software, realizar cambios y subir la versión que han modificado. No depende de un repositorio central y permite el movimiento, similar a un puntero en el tiempo, por todas las revisiones de código y desplazarse de una manera muy ágil. (Rubio, 2019).

Se recurre a Git en la versión 2.32.0 como software de control de versiones para estar al tanto de los cambios en el código fuente y contar con una copia de seguridad del mismo para poder retroceder ante cualquier imprevisto. También, porque permite realizar comparaciones entre versiones de una misma aplicación, y en caso de que haya alguna modificación, conocer quién ha sido el responsable y cuándo la ha realizado. Además, para poder observar la evolución del proyecto con el paso del tiempo.

1.7.7 Herramienta para la automatización de tareas

En cuanto a la productividad y el rendimiento laboral es importante optimizar los procesos y automatizar las tareas más repetitivas que pueden consumir mucho tiempo de la jornada de trabajo.

Gradle

Gradle es una herramienta que permite la automatización de compilación de código abierto, la cual se encuentra centrada en la flexibilidad y el rendimiento. Gradle tiene una gran flexibilidad y permite hacer usos de otros lenguajes y no solo de Java**,** también cuenta con un sistema de gestión de dependencias muy estable. Gradle es altamente personalizable y rápido ya que completa las tareas de forma rápida y precisa reutilizando las salidas de las ejecuciones anteriores, sólo procesar las entradas que presentan cambios en paralelo, además es el **sistema de compilación oficial para Android** y cuenta con soporte para diversas tecnologías y lenguajes (Muradas, 2020).

Maven: Es una herramienta de gestión de proyectos, es decir, estandariza la configuración de un proyecto en todo su ciclo de vida, como por ejemplo en todas las fases de compilación y empaquetado y la instalación de mecanismos de distribución de librerías, para que puedan ser

utilizadas por otros desarrolladores y equipos de desarrollo (Yagüe, 2019).

A partir de las características planteadas anteriormente con relación a Gradle y Maven, la **Tabla 2** muestra a modo de resumen las principales diferencias entre estas dos herramientas.

Tabla 3: Diferencias entre herramientas Gradle y Maven.

Gradle	Maven
El tiempo de construcción de Gradle es corto y rápido	El rendimiento de Maven es lento en comparación con Gradle
Los scripts de Gradle son mucho más cortos y limpios	Los scripts de Maven son un poco largos en comparación con Gradle
Utiliza lenguaje específico de dominio (DSL)	Utiliza XML
Se basa en la tarea mediante la cual se realiza el trabajo	En Maven se definen objetivos vinculados al proyecto
Admite compilaciones incrementales de la clase java	No admite compilaciones incrementales
Soporte en la mayoría de herramientas de Integración continua	Soporte en la mayoría de herramientas de Integración continua

Una vez analizados los datos anteriores y a modo de conclusión, Gradle y Maven son herramientas para la automatización de tareas, donde, serán usadas para compilar proyectos en Java. Otra de las grandes ayudas que brindan es la integración con dependencias externas al proyecto. En este caso, se puede afirmar que ambas realizan las mismas funciones, sin embargo, la implementación en Gradle es más fácil de leer y conservar. Eso se debe a que Maven utiliza *XML* y Gradle *Groovy*, haciendo a Gradle mucho más limpio en su sintaxis y fácil de modificar para, por ejemplo: instalar un nuevo *plugin* o instalar nuevas dependencias. Por las razones antes mencionadas, es Gradle la herramienta seleccionada para esta tarea.

1.7.8 Bibliotecas de clases para la visualización de mapas

Para facilitar el desarrollo de aplicaciones sobre la plataforma Android, Google proporciona un SDK muy completo. El mismo contiene las clases necesarias para visualizar mapas utilizando el servicio de mapas de Google directamente desde Internet. Existen otras alternativas que permiten que la información cartográfica se encuentre almacenada en el dispositivo móvil. A continuación, se presentan dos de las más utilizadas:

Mapsforge

El proyecto Mapsforge fue iniciado en 2008 en el Instituto de Ciencias de la Computación Frei en la universidad de Berlín. Mapsforge es una librería libre, offline y de código abierto que permite gestionar la visualización de mapas de *OpenStreetMap* en los dispositivos Android. Cuenta con una API fácil de usar, además con ella los desarrolladores pueden crear sus propias aplicaciones de mapeo en menos de 30 líneas de código. Fue diseñada y desarrollada con el objetivo de proveer herramientas libres y abiertas que permitan a la comunidad la creación de aplicaciones basadas en OpenStreetMap de una forma sencilla. Estas herramientas y APIs proporcionadas incluyen soluciones para representación en mapas, planificación y navegación de rutas, indexado y búsqueda de POI, capas en los mapas entre otras opciones (Wei, 2013).

Durante el análisis de la documentación referente a las librerías que existen para realizar el mapeo, haciendo uso de los mapas que brinda OpenStreetMap se encontró que no solo Mapsforge es la única librería desarrollada para esto. También existen otras como OsmDroid, la cual permite personalizar el estilo de renderización de los mapas y gestionar mapas sin conexión, pero a diferencia de Mapsforge los almacena como tramas y a la vez ocupa mucho espacio en el dispositivo, siendo esto una gran desventaja para esta librería, por lo tanto, se selecciona Mapsforge para realizar la confección del mapa.

1.8 Conclusiones del capítulo

Con el estudio de los elementos presentes en este capítulo, se les da cumplimiento a las tareas de la investigación 1, 2, 3 y 4, lo que se evidencia en:

- ✓ El estudio de los conceptos asociados a la investigación, permitió obtener un mayor dominio del problema a resolver.

- ✓ El análisis de los Sistemas de información geográficos existentes, específicamente, aquellos que se enfocan en brindar información al usuario a través de una aplicación móvil, sobre el uso de los medios del transporte público, permitió conocer de forma general el funcionamiento y desarrollo de los mismos y demostró que estos no satisfacen las necesidades de Ómnibus UCI debido a que no cumplen en su totalidad los requerimientos que necesita la aplicación. Sin embargo, se obtuvieron de ellos características de funcionamiento que sirven de guía para el desarrollo de la investigación.
- ✓ La caracterización de la metodología Ágil XP, como metodología seleccionada, permitió guiar el proceso de desarrollo del software de una manera productiva y eficaz y trabajar en equipo de manera organizada.
    - La selección de las herramientas y tecnologías mayormente adecuadas para dar solución al problema planteado, teniendo en cuenta las características de estas y las necesidades descritas con anterioridad permitieron conocer las ventajas que ofrecen las mismas para el desarrollo de la aplicación.

CAPÍTULO 2: Análisis y diseño de la aplicación Ómnibus UCI

En el presente capítulo se describen las principales características de la propuesta de solución. Siguiendo las características que plantea la metodología ágil XP en su fase de Planificación, la cual conduce los pasos a seguir para lograr mayor formación en la distribución de las actividades, tiempo y artefactos a desarrollar. Se realiza la especificación de los requisitos funcionales y no funcionales, y se confeccionan las historias de usuario correspondientes a dichos requisitos y los prototipos de diseño de interfaz de usuario.

2.1 Descripción del Modelo de Dominio

El presente epígrafe describe el modelo del dominio realizado durante la fase de planeación de la metodología seleccionada. Este artefacto según Larman es uno de los más importantes en el ciclo del desarrollo de software.

2.1.1 Modelo de dominio

El modelo de dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina modelos conceptuales, modelos

de objetos del dominio y modelos de objetos de análisis (Larman, 2016). Se realiza el modelo de dominio ya que este es el mecanismo fundamental para comprender el dominio del problema y para establecer conceptos comunes. Es un diccionario visual del dominio del problema. Teniendo en cuenta todo lo explicado anteriormente, se decide realizar el Modelo de Dominio. Para una mejor comprensión del mismo ver **Ilustración 1**.

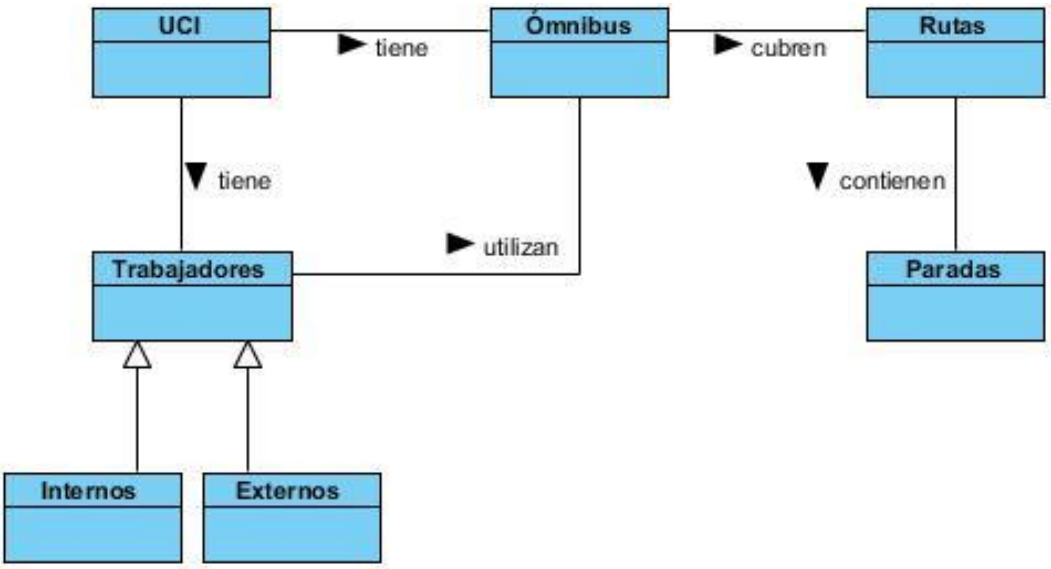


Ilustración 1: Modelo del Dominio. Elaboración propia.

Como se evidencia en el Diagrama de Modelo de Dominio, la universidad tiene trabajadores, los cuales son internos o externos. Estos trabajadores hacen uso de los ómnibus que la universidad tiene destinados para transportar al personal que no reside en la institución. Los ómnibus cubren disimiles rutas, las cuales están asociadas al destino de los trabajadores y cada ruta contiene varias paradas.

2.1.2 Descripción de conceptos del dominio

A continuación, se brinda una breve descripción de los conceptos fundamentales identificados en el ámbito del problema, con la finalidad de proporcionar una mejor comprensión del Diagrama del Modelo del Dominio.

UCI: Universidad de las Ciencias Informáticas.

Trabajadores: Recursos humanos que mantienen una relación contractual con la institución.

Internos: Trabajadores que tienen asignada una residencial en la institución.

Externos: Trabajadores que no tienen asignada una residencial en la institución y viven en zonas aledañas al campus universitario.

Ómnibus: Medio de transporte a través del cual se transportan los trabajadores.

Rutas: Vías que permiten transitar desde un lugar a otro.

Paradas: Elementos urbanos caracterizados por ser un espacio público, lugar de encuentro entre buses y pasajeros.

2.2 Requisitos

La educción de requisitos es la actividad de la ingeniería de requisitos definidas por Pressman para la identificación de los requisitos funcionales y no funcionales (Pressman, 2010).

En la educción de requisitos se tuvo como fuentes de obtención de requisitos los sistemas estudiados en el capítulo anterior, la entrevista realizada a expertos que laboran en la base de transporte de la universidad y el resultado de la encuesta ejecutada a varios trabajadores externos. La información obtenida de estas fuentes se transformó en los requisitos apropiados para el diseño de la aplicación Ómnibus UCI.

Una vez procesada toda la información obtenida de las técnicas y métodos aplicados, se formalizaron por escrito, estableciendo los servicios que la aplicación debe ser capaz de ofrecer y las condiciones bajo las cuales debe operar. Los requisitos identificados se clasificaron en funcionales y no funcionales.

2.2.1 Requisitos funcionales

Los requisitos funcionales (RF) definen las acciones que debe realizar el sistema. Son capacidades o condiciones que el sistema debe cumplir, cómo debe comportarse en situaciones específicas. En algunos casos también pueden plantear explícitamente qué no debe hacer el sistema (Sommerville, 2011).

Descripción de requisitos funcionales

La descripción de requisitos se encarga de mostrar para cada requisito su descripción, donde se explica qué se espera que el sistema pueda hacer una vez que un usuario requiera darle inicio al requisito. La prioridad de los requisitos se estableció a partir del factor importancia (Alta, Media, Baja).

Tabla 4: Listado de requisitos funcionales.

Identificador	Requisito	Descripción	Prioridad
RF 1	Navegar en el mapa.		Media
RF 2	Mostrar ubicación en el mapa		Media
RF 3	Mostrar posibles rutas a tomar.		Media
RF 4	Localizar parada más cercana a la posición del usuario.		Media
RF 5	Localizar parada más cercana al punto de destino.		Media
RF 6	Visualizar puntos de interés en el mapa.		Media
RF 7	Mostrar paradas por rutas.		Media
RF 8	Autenticar usuario.		Alta
RF 9	Insertar usuario.		Alta
RF 10	Mostrar usuario.		Alta
RF 11	Modificar usuario.		Alta
RF 12	Eliminar usuario.		Alta

RF 13	Insertar ómnibus.		Alta
RF 14	Mostrar ómnibus.		Alta
RF 15	Modificar ómnibus.		Alta
RF 16	Eliminar ómnibus.		Alta
RF 17	Insertar ruta.		Alta
RF 18	Mostrar ruta.		Alta
RF 19	Modificar ruta.		Alta
RF 20	Eliminar ruta.		Alta
RF 21	Insertar parada.		Alta
RF 22	Mostrar parada.		Alta
RF 22	Modificar parada.		Alta
RF 24	Eliminar parada.		Alta

2.2.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) detallan las propiedades o cualidades que el producto debe tener, aumentándole funcionalidad al sistema, pues hacen al producto atractivo, fácil de usar, rápido y confiable (Pressman, 2010).

Tabla 5: Listado de requisitos no funcionales.

Requisitos no funcionales	
Interfaz de usuario	**RNF 1:** La aplicación debe presentar un diseño adaptable a las dimensiones del contenido y muestre los elementos de una forma ordenada y optimizada independientemente de la

	resolución del dispositivo. **RNF 2:** La aplicación debe poseer interfaz de usuario basada en *Material Design*[16].
Diseño e implementación	**RNF 3:** Se utiliza el lenguaje de programación Java en su versión 11. **RNF 4:** Se emplea como IDE Android Studio en su versión 2020.3.1.22 junto al SDK 23.0.0 para el desarrollo de las funcionalidades.
Portabilidad	**RNF 5:** La aplicación debe funcionar correctamente en dispositivos móviles con sistema operativo Android en su versión 4.0 o superior.
Hardware	**RNF 6:** El dispositivo debe tener mínimo 50 MB de almacenamiento disponible. **RNF 7:** Microprocesador Dual Core a 2.0 GHz o superior. **RNF 8:** Memoria RAM de 512 MB o superior. **RNF 9:** Debe contar con sistema de posicionamiento global (GPS) integrado. **RNF 10:** Ordenador con sistema operativo Windows. **RNF 11:** Procesador Intel Celeron con velocidad de 1.6 GHz. **RNF 12:** Memoria RAM de 2 GB de y 500 MB de disco duro.
Software	**RNF 13:** El dispositivo móvil debe tener el sistema operativo Android en su versión 4.0 o superior.
Disponibilidad	**RNF 14:** La aplicación debe estar disponible en todo momento que el usuario necesite acceder y manejar la información contenida en la misma. **RNF 15:** Debe utilizar Internet cada vez que surja algún

[16] Normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma, este debe brindar una interfaz amigable y comprensible para el usuario.

	cambio en el servidor.

2.3 Especificación de los requisitos

La especificación de requisitos en el proceso de desarrollo del software es de vital importancia. Tener los requisitos bien claros y definidos permite comprender desde un inicio la línea a seguir en el desarrollo y así garantizar la eficiencia y calidad del software (Pressman, 2010). Para realizar la especificación de requisitos de la aplicación que le posibilitará a los usuarios acceder a la información referente a los recorridos de los ómnibus de la Universidad de las Ciencias Informáticas, se hace uso de las historias de usuario.

Las historias de usuario (HU) constituyen una forma de administración de requisitos sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las mismas son escritas utilizando el lenguaje común del usuario (Jeffries et.al, 2001).

Para la descripción de requisitos funcionales de la propuesta de solución se define una historia de usuario para cada uno de ellos, para un total de 24 historias de usuario. A continuación, se muestra la HU correspondiente al requisito funcional **Localizar parada más cercana a la posición del usuario**. En el **Anexo 3: Historias de Usuario** pueden encontrarse las HU restantes implementadas en el desarrollo de la aplicación.

En el marco de la investigación fueron definidos los siguientes parámetros a tener en cuenta en las HU:

- **Número:** Número asignado a la HU.
- **Nombre:** Nombre de la HU.
- **Programador:** Nombre del programador responsable de la HU.
- **Iteración asignada:** Iteración a la que pertenece la HU en el plan de iteraciones.
- **Prioridad:** Nivel de prioridad de la HU señalado por el cliente (Alta, Media, Baja).
- **Complejidad:** Nivel de complejidad a la hora de programar (Alta, Media, Baja)**.**

- **Tiempo estimado:** Estimación que hace el equipo de desarrollo del tiempo necesario para implementar la HU. El 1 equivale a una semana ideal de trabajo – 5 días hábiles por 8 horas diarias. Por lo que 0.5 equivale a 2 días y medio, para un total de 20 horas.
- **Tiempo real:** El tiempo real en el que se realizó la HU.
- **Descripción:** Breve descripción de la HU.
- **Observaciones:** Aspectos importantes de interés para el cliente.
- **Prototipo de interfaz:** Prototipo visual del diseño esperado.

Prioridad en el negocio:

- **Alta**: Son aquellas HU que constituyen funcionalidades fundamentales en el desarrollo el sistema y el cliente define como principales para el control integral del sistema.
- **Media**: Son las funcionalidades a tener en cuenta por el cliente, sin que estas tengan una afectación sobre el sistema que se esté desarrollando.
- **Baja:** Es otorgada a las HU que son funcionalidades que sirven de ayuda al control de elementos asociados al equipo de desarrollo.

2.3.1 Descripción de las HU

Tabla 6: HU-Localizar parada más cercana a la posición del usuario.

Historia de Usuario		
Número: HU_4	**Nombre:** Localizar parada más cercana a la posición del usuario.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 2	
Complejidad: Alta	**Tiempo real:**	

Descripción:

1. **Objetivo:**

-El sistema debe permitirle al usuario visualizar la parada más cercana a su posición actual.

2. **Acciones para lograr el objetivo:**

Para localizar la parada más cercana a la posición del usuario hay que:

-Cargar el mapa en la Base de Datos.

3. **Flujo de la acción a realizar:**

- El sistema debe permitir la localización de la parada más cercana a la posición del usuario.

- Cuando el usuario accede a la aplicación, oprime sobre su posición actual y se muestra una ventana en la cual selecciona la opción Paradas Cercanas.

Observaciones:

Prototipo de interfaz:

2.3.2 Plan de Estimación de esfuerzo

Determinar el esfuerzo requerido para llevar a cabo una tarea es parte de todo proceso de desarrollo de software, ya que permite un monitoreo y control factible para garantizar que los proyectos finalizan de acuerdo a lo planeado. Para efectos de planificación, las historias pueden ser de una a tres semanas de tiempo de programación (para no superar el tamaño de una iteración) (Jeffries et.al, 2001). Cuando una HU tiene tiempo de duración inferior a una semana significa que está muy bien descrita y que puede ser combinada con otra sin muchas dificultades (Beck, 1999).

A continuación, se muestra mediante una tabla la planificación de las diferentes HU para cada iteración teniendo en cuenta su prioridad.

Tabla 7: Estimación del esfuerzo por Historias de Usuarios.

Número	Nombre de la Historia de Usuario	Tiempo estimado en días / semanas
HU_1	Navegar en el mapa.	1
HU_2	Mostrar ubicación en el mapa.	1
HU_3	Mostrar posibles rutas a tomar.	1
HU_4	Localizar parada más cercana a la posición del usuario.	0.5
HU_5	Localizar parada más cercana al punto de destino.	2
HU_6	Visualizar puntos de interés en el mapa.	0.5
HU_7	Mostrar paradas por rutas.	2
HU_8	Autenticar usuario.	0.5
HU_9	Insertar usuario.	0.5
HU_10	Mostrar usuario.	0.5

HU_11	Modificar usuario.	0.5
HU_12	Eliminar usuario.	0.5
HU_13	Insertar ómnibus.	0.5
HU_14	Mostrar ómnibus.	0.5
HU_15	Modificar ómnibus.	0.5
HU_16	Eliminar ómnibus.	0.5
HU_17	Insertar ruta.	0.5
HU_18	Mostrar ruta.	0.5
HU_19	Modificar ruta.	0.5
HU_20	Eliminar ruta.	0.5
HU_21	Insertar parada.	0.5
HU_22	Mostrar parada.	0.5
HU_23	Modificar parada.	0.5
HU_24	Eliminar parada.	0.5
Total	**24**	**78 / 16**

2.3.3 Plan de duración de iteraciones

La implementación se ejecuta en dos iteraciones teniendo en cuenta la complejidad de las historias y las dependencias entre historias de usuario. A continuación, se muestra el plan de iteraciones.

Tabla 8: Plan de duración de iteraciones

Iteración	Historia de usuario asignada	Duración (semanas)
1	8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 y 24	8
2	1, 2, 3, 4, 5, 6 y 7	8
Total	**24**	**16**

2.4 Diseño de la propuesta de solución

Durante el desarrollo de la propuesta de solución se modela el sistema, su forma y su arquitectura para que soporte todos los requisitos, incluyendo los requisitos no funcionales (Rodríguez, 2015). Propiciando que dichos elementos sirvan de base a la etapa de implementación.

2.4.1 Arquitectura de software

Según (Pressman, 2010) en su forma más simple, la arquitectura del software es la estructura u organización de los componentes del programa, la manera en que estos interactúan y la estructura de datos que utilizan. El objetivo principal de la arquitectura del software es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los miembros de un proyecto. Para conseguirlo, la arquitectura del software construye abstracciones, materializándolas en forma de diagramas comentados. (Casanova, 2004)

2.4.2 Estilo arquitectónico

La aplicación está planteada con una arquitectura basada en el Modelo Vista Controlador (MVC) como la mayoría de los proyectos Android. Esto permite manejar de forma independiente las actividades encargadas de las vistas y las clases controladoras que como su nombre da a relucir están encargadas de controlar el funcionamiento interno del sistema. En el caso específico de Android el modelo vista-controlador tiene como principal bondad separar los datos de una aplicación, la interfaz de usuario y la lógica de negocios en tres componentes distintos que se relacionarán para tener como resultado la aplicación final (Buschmann, 1996).

- Modelo: El componente maneja lo referente a la persistencia de datos de la aplicación, las clases entidades, el acceso a la red y los elementos necesarios para manejar dichos elementos (paquete Model).
- Vista: El componente representa la interfaz gráfica para la interacción con el usuario. Dentro se ubican todos los componentes que intervienen en la visualización del resultado de la comunicación con el Controlador (paquete View).
- Controlador: Componente que contiene las clases que interactúan con la Vista recibiendo las solicitudes de eventos de los usuarios y con el Modelo registrando los cambios realizados por el mismo (paquete Controller).

Se representa a continuación, en la **Ilustración 2**, una vista lógica de la arquitectura MVC utilizando el diagrama de paquetes.

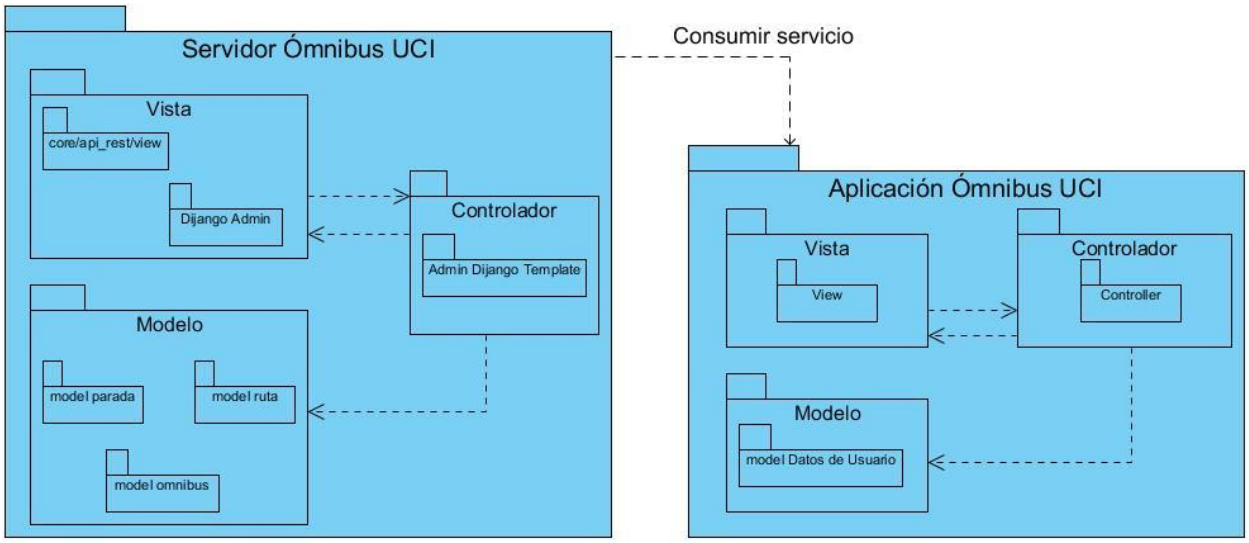


Ilustración 2: Arquitectura del sistema. Elaboración propia.

En cambio, el Sistema de Gestión Ómnibus UCI está planteado con una arquitectura basada en el Modelo Vista Plantilla (MVP). Según Infante (2012), Django es un marco de trabajo MTV del inglés Model-Template-View que es una modificación de MVC (Modelo-Vista-Controlador) debido a que los desarrolladores del marco de trabajo no tuvieron la intención de seguir algún

patrón de desarrollo sino hacerlo lo más funcional posible. Para comenzar a entender Django se debe tener en cuenta su analogía con MVC de la siguiente forma:

- ✓ El modelo sigue siendo Modelo (M) en Django
- ✓ La vista pasa a llamarse Plantilla (P) en Django
- ✓ El controlador pasa a llamarse Vista (V) en Django

Por tanto, se asumirá la arquitectura Modelo-Plantilla-Vista (MPV) para Django, de la cual se explica su funcionamiento a continuación, el cual se puede observar en la **Ilustración 3**.

1. El navegador envía una solicitud.
2. La vista interactúa con el modelo para obtener datos.
3. La vista llama a la plantilla.
4. La plantilla renderiza la respuesta a la solicitud del navegador.

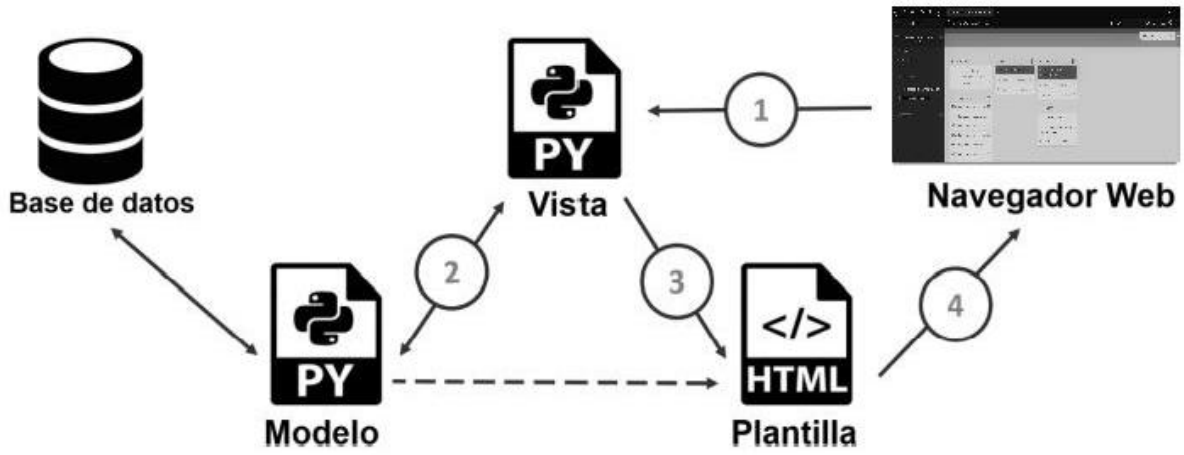


Ilustración 3: Funcionamiento del MPV en Django aprehendido de Infante (2012).

A partir de lo planteado anteriormente, se deriva lo siguiente (Infante, 2012):

- **Modelo:** Define lo datos almacenados, es representado en forma de clases de Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros, posee métodos también. Todo esto permite indicar y controlar el comportamiento de los datos.
- **Plantilla:** recibe los datos de la vista y luego los organiza para la presentación al navegador web. Básicamente es una página HTML (HyperText Markup Language) con

algunas etiquetas extras que son propias del Django, dichas etiquetas permiten flexibilidad para los desarrolladores del *frontend*[17].

- **Vista:** su propósito es determinar qué datos serán visualizados, es representado en forma de funciones.

2.4.3 Tarjetas Clase-Responsabilidad-Colaborador

En la metodología XP el diseño de las clases se realiza a través de las tarjetas CRC (Clase-Responsabilidad-Colaborador), para de esta forma ayudar al refinamiento de las clases. De forma organizada cada tarjeta representa una clase, donde se describen las responsabilidades que tiene y las clases colaboradoras que se relacionan con la misma. Además, ayudan a diseñar el sistema en conjunto entre todo el equipo de desarrollo, aunque su principal objetivo es propiciar el enfoque orientado a objetos y reducir el modo de pensar procedimental.

Las tarjetas CRC son el único producto del trabajo de diseño que se genera como parte del proceso XP (Pressman, R. 2010). El diseño XP sigue rigurosamente el principio MS (mantenlo sencillo). Un diseño sencillo siempre se prefiere sobre una representación más compleja. En esta etapa se busca describir las responsabilidades que tiene que cumplir cada clase y las colaboraciones entre ellas para poder implementar las historias de usuario (Cevallos, 2015). A continuación, se describen algunas de las tarjetas CRC diseñadas para la implementación del sistema Ómnibus UCI.

Tabla 9: Tarjeta CRC de la clase Ruta.

Nombre de la clase: Ruta	
Descripción: Contiene toda la información referente a una ruta en específico.	
Responsabilidad	**Colaboradores**
Describir el origen, destino y las paradas que tiene una ruta.	Ómnibus Parada

Tabla 10: Tarjeta CRC de la clase Parada.

Nombre de la clase: Parada

[17] Es la parte del desarrollo web que se dedica de la parte frontal de un sitio web, en pocas palabras, se encarga del diseño de un sitio web, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos.

Descripción: Define los detalles de una parada.	
Responsabilidad	**Colaboradores**
Describe el nombre, la dirección la latitud y la longitud de una parada determinada.	Ruta

Tabla 11: Tarjeta CRC de la clase Ómnibus.

Nombre de la clase: Ómnibus	
Descripción: Contiene toda la información referente a un ómnibus en específico.	
Responsabilidad	**Colaboradores**
Definir el ómnibus que realizara la ruta en cuestión.	Ruta

2.4.4 Diagrama de clases

Para un mejor entendimiento de cómo se representa la parte estática del sistema, las clases y sus relaciones, se decide realizar el diagrama de clases representado en la **ilustración 4**. Los diagramas de clases del diseño con estereotipos web, describen gráficamente las especificaciones del modelo, la vista y la plantilla de las historias de usuario descritas en el sub-epígrafe 2.3.3. Estas representaciones contienen información acerca de las clases, asociaciones, atributos, métodos y dependencias (Sommerville, 2011).

Para la modelación de este diagrama se emplearon los patrones GRASP (del inglés General Responsibility Assignment Software Patterns, Principios Generales para Asignar Responsabilidades), que constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objetos esenciales y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Este enfoque para la comprensión y utilización de los principios de diseño se basa en los patrones de asignación de responsabilidades (Bermúdez et.al, 2012).

Patrones GRASP

Los patrones de asignación de responsabilidades describen los principios fundamentales del diseño de objetos para la asignación de responsabilidades (Gamma et.al, 2004). Teniendo en cuenta el patrón arquitectónico seleccionado, los patrones de diseño que se aplican durante el desarrollo del Sistema de Gestión Ómnibus UCI son los siguientes:

Experto: Se usa más que cualquier otro al asignar responsabilidades, es un principio básico que suele utilizarse en el diseño orientado a objeto. Consiste en la asignación de una responsabilidad a la clase que cuenta con la información necesaria para llevarla a cabo. El uso de este patrón da pie a un bajo acoplamiento y una alta cohesión, lo que favorece tener sistemas más robustos y de fácil mantenimiento. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos (Larman, 2004). En la aplicación este patrón se evidencia en la mayoría de las clases ya que a cada una le fue asignada la responsabilidad en dependencia de la información que esta posee. Por ejemplo, las clases del modelo de la Base de Dato (Ómnibus, Parada y Ruta) las cuales representan las entidades de la base de datos lo que permite extraer los datos de la misma.

Alta Cohesión: En la perspectiva del diseño orientado a objetos, la cohesión o cohesión funcional es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme, clases con responsabilidades moderadas en un área funcional que colaboran con las otras para llevar a cabo las tareas (Larman, 2004). Este patrón se evidencia en el proyecto en la clase Autenticación del *LDAP*[18] ("*ClientUserManager*") el cual tiene como única responsabilidad autenticar mediante el LDAP a los usuarios autorizados.

Bajo Acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo o débil acoplamiento no depende de muchas otras. El bajo acoplamiento soporta el diseño de clases más independientes y reutilizables, lo cual reduce el impacto de los cambios y acrecienta la oportunidad de una mayor productividad (Larman, 2004). Este patrón se observa en la clase

[18] (Protocolo de Acceso Liviano al Directorio). Es un protocolo basado en la conexión entre cliente y servidor.

("*ReadOnlyAdmin*") la cual se usa como *mixin*[19] en Python.

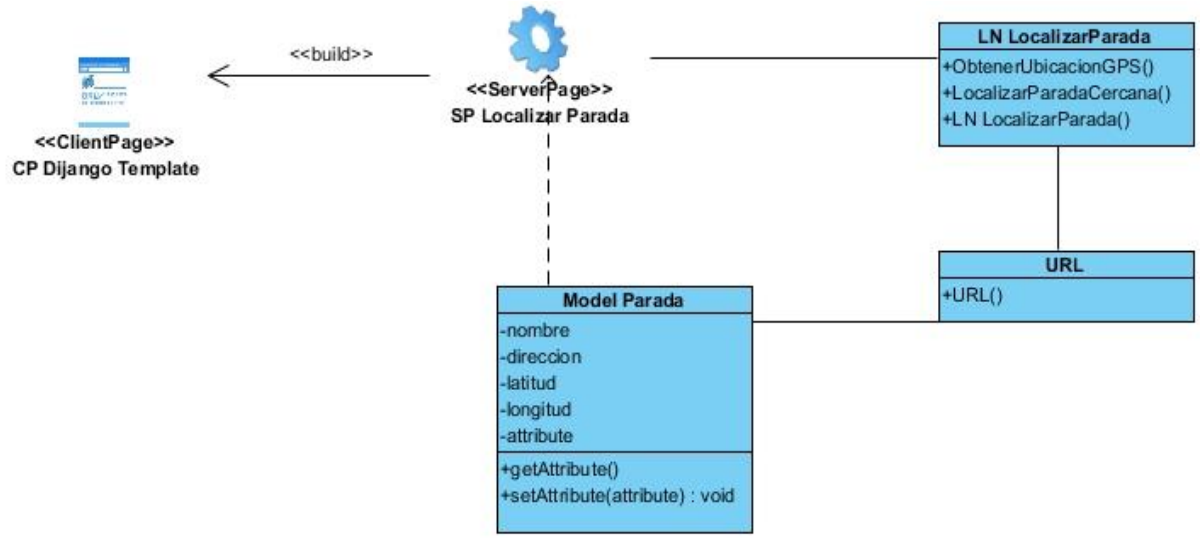


Ilustración 4: Diagrama de clases de la HU-Localizar parada más cercana a la posición del usuario.

2.4.6 Modelo de la Base de Datos

El modelo de datos determina la estructura lógica de una base de datos y el modo de almacenar, organizar y manipular los datos. Dentro de sus propósitos se encuentra definir los datos persistentes que serán almacenados (Pressman, 2010).

Con el objetivo de definir las clases persistentes se identifican los conceptos, en el dominio del negocio, que persisten en el tiempo. La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo, por lo que no todos los conceptos definidos son transformados en clases de este tipo (Pressman, 2010). Se generó el siguiente diagrama entidad-relación, el cual posee un conjunto de tablas correspondientes a cada componente, como se puede observar en la **Ilustración 5**.

[19] Clase intermedia que se utiliza para poder asignarle responsabilidades a otra sin que necesariamente estén vinculadas.

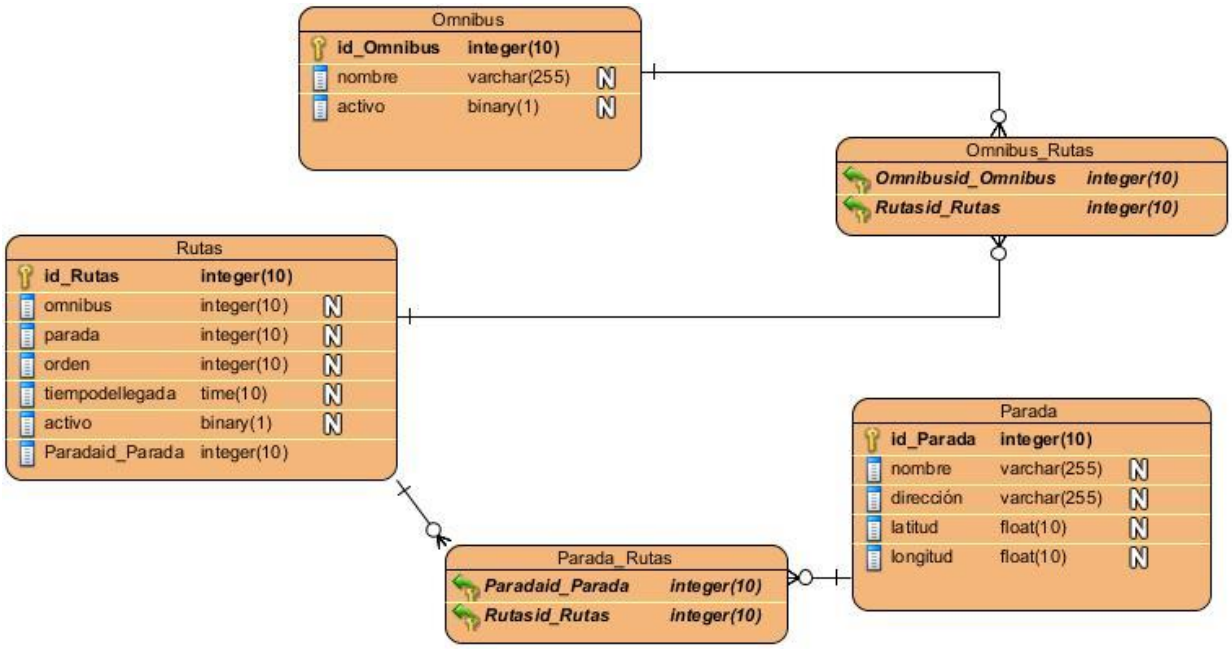


Ilustración 5: Modelo físico de la Base de Datos. Elaboración propia.

2.5 Modelo de despliegue

Un modelo de despliegue consiste en una representación estructural de la arquitectura del sistema desde el punto de vista de la distribución de los artefactos del software en los destinos de despliegue; definiendo a los artefactos como representaciones de elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo (Sarmiento, 2016).

A pesar de que este artefacto no es generado por la metodología seleccionada, a continuación, en la **Ilustración 6** se muestra el diagrama de despliegue propuesto para el Sistema de Gestión Ómnibus UCI. El mismo, muestra la disposición física de los nodos que componen el sistema y el reparto de los componentes en dichos nodos.

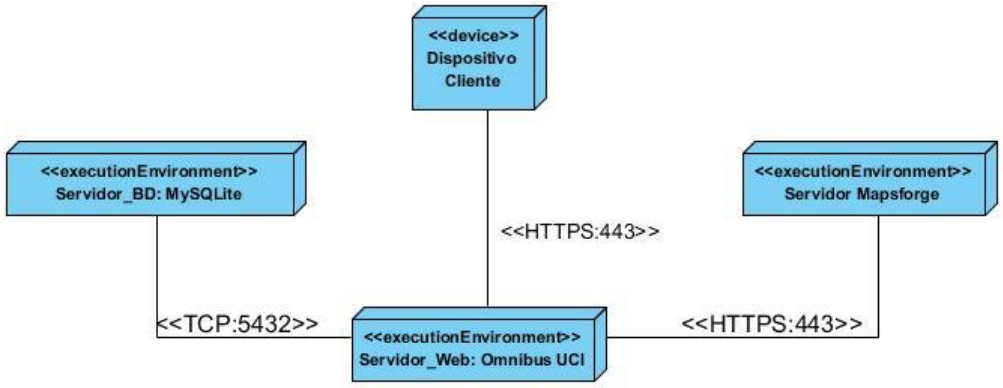


Ilustración 6: Modelo de despliegue del Sistema de Gestión Ómnibus UCI.

En este diagrama se definen las estaciones de trabajo (Dispositivo Cliente: computadora personal, tableta, teléfono celular) que el usuario utilizará para conectarse, vía *HTTPS*[20], con el servidor de aplicaciones web (Ómnibus UCI) en el que se encuentra alojado el Sistema de Gestión Ómnibus UCI. Este servidor debe disponer de un ordenador con sistema operativo Windows y cuyas propiedades mínimas sean un procesador Intel Celeron con velocidad de 1.6 GHz, 2 GB de memoria RAM y 500 MB de disco duro. Se observa, además, la conexión vía HTTPS del servidor web con el servidor Mapsforge, necesario para gestionar la visualización del mapa que se descargará de este servidor; y el uso del protocolo *TCP* [21] para conectarse con la BD del mismo, la cual estará en servidor de Base de Datos PostgreSQL con capacidad mínima de 500 MB de disco duro.

2.6 Conclusiones del capítulo

En este capítulo se manifestaron las principales características de la aplicación Ómnibus UCI, cumpliendo las tareas de la investigación 5 y 6, lo que se evidencia en:

- ✓ La especificación de los requisitos funcionales y no funcionales; lo cual ha contribuido a una mejor organización del sistema, permitiéndole al desarrollador identificar sus funcionalidades básicas.
- ✓ La selección de la arquitectura de la aplicación, empleándose el patrón arquitectónico Modelo-Vista-Controlador, facilitando la separación de los datos de la aplicación, la

[20] Del inglés HyperText Transfer Protocol Secure, Protocolo Seguro de Transferencia de Hipertexto (Definición .DE, 2014).
[21] Del inglés Transmission Control Protocol, Protocolo de Control de Transmisión (Definición .DE, 2014).

interfaz de usuario y la lógica de negocios en tres componentes distintos; garantizando un bajo acoplamiento y una buena escalabilidad a la aplicación.

- ✓ Se identificaron los patrones de diseño GRASP y GoF utilizados en el desarrollo de la aplicación, proporcionando la asignación de responsabilidades logrando un diseño de software que sirva de apoyo a la implementación de la aplicación.

CAPÍTULO 3: Implementación y evaluación de la aplicación

En el capítulo se presentan los diseños de casos de prueba que permiten realizar pruebas al sistema, con el objetivo de verificar que responde a las necesidades de los *stakeholders*[22]. Además, se presenta la disciplina de codificación y las pruebas a realizar referente a la disciplina de prueba.

3.1 Diagrama de componentes

El diagrama de componentes muestra un conjunto de elementos de un modelo, tales como: componentes, subsistemas de implementación y sus relaciones, el cual se utilizan para modelar la vista estática de un sistema. En éste se muestra la organización y las dependencias lógicas entre un conjunto de componentes software, sean estos componentes de código fuente, librerías, binarios o ejecutables. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema (Urquiza, 2013).

A pesar de que la metodología seleccionada no genera este artefacto, a continuación, la **Ilustración 7**, muestra el diagrama de componentes de la aplicación Ómnibus UCI; cuya organización se encuentra acorde con el estilo arquitectónico MVP propuesto por el marco de trabajo Django y descrita en el capítulo anterior de este trabajo.

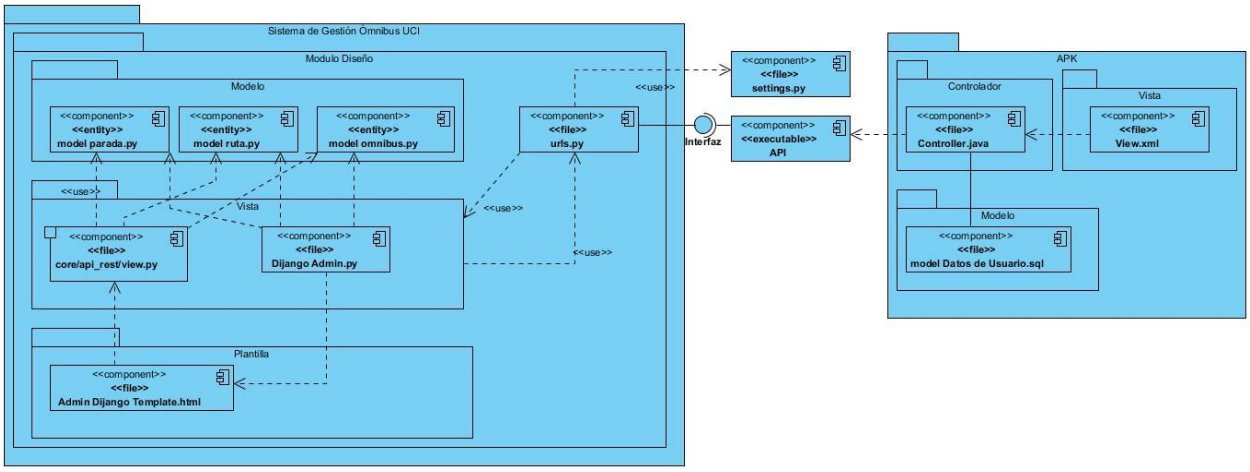


Ilustración 7: Diagrama de componentes.

[22] Involucrados o interesados (Tomas, 2020).

A continuación, se describen los elementos que componen el diagrama de componentes mostrado.

Tabla 12: Descripción de los elementos del diagrama de componentes.

Componentes		Descripción
Vista	core/api_rest/view.py	API REST que provee los datos de los ómnibus, las paradas y las rutas, a la aplicación web.
Modelo	core/models.py	Clase entidad que almacena los modelos ómnibus, ruta y parada.
Plantilla	No se creó ningún componente ya que la aplicación de administración de Django usa los modelos para construir automáticamente el área deseada dentro del sitio. **Django Admin**, es un panel de control que permite administrar todos los modelos, por tanto, Django genera todo el sistema administrativo, es decir, el código ya está hecho y los formularios también.	

3.2 Codificación

Es la única actividad de la que no podremos prescindir. Sin código fuente no hay programa (Solís, 2003). La codificación debe hacerse ateniendo a estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión y escalabilidad (André et.al, 2006).

3.2.1 Estándares de codificación

Los estándares de codificación son un conjunto de convenciones (denominaciones, formatos) establecidas para la escritura de código. Estos estándares varían en dependencia del lenguaje de programación y son necesarios para los programadores y miembros del equipo de trabajo. Generalmente el software no es mantenido por los autores por ende debe ser revisado y/o manejado por otros. Los estándares posibilitan una mejor lectura e interpretación del software y su empleo permite aplicar un conjunto de lineamientos (André et.al, 2006). A continuación, se definen los estándares de codificación a utilizar.

Sangría:

- Utilice 4 espacios para la sangría.

Grosor de línea:

- Cada línea de código no debe exceder los 80 caracteres en la medida de lo posible (en circunstancias especiales, puede exceder ligeramente los 80, pero la más larga no puede exceder los 120)

Razón:

- Esto es útil al ver una diferencia de lado a lado.
- Conveniente para ver el código debajo de la consola
- Demasiado tiempo puede ser un diseño defectuoso

Comillas:

- En pocas palabras, el lenguaje natural usa comillas dobles y las etiquetas de máquina usan comillas simples, por lo que la mayor parte del código debe usar comillas simples
- El lenguaje natural usa comillas dobles"..."

  Por ejemplo, mensajes de error; en muchos casos sigue siendo Unicode, utilice "Hola mundo"
- ID de máquina Utilice comillas simples'...' Por ejemplo, la clave en el dict
- Las expresiones regulares usan comillas dobles nativas"..."
- La cadena de documentos usa tres comillas dobles"""....."""" (Programador Clic).

En la siguiente figura se evidencia la utilización de los estándares anteriormente mencionados.

```
from django.http import JsonResponse
from core.models import Omnibus, Parada, Ruta


def download_data(request):
    """Endpoint para la extraccion de los datos de las paradas y los omnibus"""
    data = {
        'omnibus': [x.to_json() for x in Omnibus.objects.filter(active=True)],
        'paradas': [x.to_json() for x in Parada.objects.all()],
    }
    return JsonResponse(data, safe=False)
```

Ilustración 8: Empleo de los estándares de codificación.

3.3 Pruebas

Las pruebas son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Por esta razón se debe definir en el proceso de la ingeniería del software. Todo esto contribuye a elevarla calidad de los productos desarrollados y a la seguridad de los programadores a la hora de introducir cambios o modificaciones (Joskowicz, 2008).

Uno de los pilares de la Programación Extrema es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones (Joskowicz, 2008).

La metodología XP divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente (Joskowicz, 2008).

En este epígrafe se muestran los resultados de la estrategia de prueba diseñada para la propuesta de solución, en función de garantizar y validar la calidad de este.

Tabla 13: Selección de pruebas. Elaboración propia.

Tipo de prueba	Método (técnica) de prueba	Validación
Pruebas de unidad	Basado en clases. TestCase (Caja blanca)	Valida las funcionalidades diseñadas para el sistema
Pruebas de aceptación	Prueba de tipo alfa	Valida el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique.

3.3.1 Pruebas de unidad

Primero se realizan pruebas de unidad que enfoca los esfuerzos de verificación en la unidad más pequeña del diseño de software: el componente o módulo de software. Para esto se va a realizar el método de caja blanca que es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba.

Django proporciona un marco de prueba con una pequeña jerarquía de clases que se basan en la librería *unittest* [23] estándar de Python. Este módulo es adecuado para pruebas unitarias y las define usando un enfoque basado en clases.

Unit Testing es el primer nivel de prueba de software en el que se prueban las partes comprobables más pequeñas de un software. Esto se utiliza para validar que cada unidad del software funcione según lo diseñado. Para esto se va a realizar el método de caja blanca que es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba (GeeksforGeeks, 2021).

Un ejemplo que subclases de *django.test.TestCase* , que es una subclase de *unittest.TestCase* que ejecuta cada prueba dentro de una transacción para proporcionar aislamiento:

```
from django.test import TestCase // Importar clase para probar.
from core.models import Ómnibus // Importar la clase modeloOmnibus.
```

Hay tres tipos de posibles resultados de la prueba:

- OK: esto significa que se aprobaron todas las pruebas.
- FAIL: esto significa que la prueba no pasó y se genera una excepción AssertionError.
- ERROR: esto significa que la prueba genera una excepción distinta de AssertionError.

```
import unittest

class SimpleTest(unittest.TestCase):


    def test(self):
        self.assertTrue(True)

if __name__ == '__main__':
    unittest.main()
```

[23] Infraestructura de tests unitarios (Python, 2021).

```
prueba (__main __. SimpleTest) ... ok
---------------------------------------------- --------------------
Ejecutó 1 prueba en 0.000 s

Okay
```

Al ejecutar las pruebas, el comportamiento predeterminado de la utilidad de la misma, fue encontrar todos los casos de prueba (es decir, subclases de unittest.TestCase) en cualquier archivo cuyo nombre comenzara con *test* y que construyera automáticamente un conjunto de pruebas a partir de esos casos de prueba, y ejecutara esa *suite*. Al finalizar la prueba, se determinó que se lograron ambos objetivos.

3.3.2 Pruebas de aceptación

Según Huaraca (2013), el uso de cualquier producto de software tiene que estar justificado por las ventajas que ofrece. Sin embargo, antes de empezar a usarlo es muy difícil determinar si sus ventajas realmente justifican su uso. El mejor instrumento para esta determinación es la llamada prueba de aceptación. En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique.

Se decide realizar las pruebas de aceptación ya que estas se realizan con el objetivo de evaluar el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique. Con su empleo se persigue evaluar la disposición del sistema para su despliegue y posterior uso. Las pruebas de aceptación se derivan de las historias de los usuarios que se han implementado como parte de la liberación del software (Pressman, 2010).

En la realización de las pruebas se emplean casos de prueba que permiten detectar el mayor número de no conformidades y corregirlas antes de la entrega del software.
La mayoría de los realizadores de productos de software usan un proceso llamado prueba alfa y prueba beta para descubrir errores que al parecer sólo el usuario final es capaz de encontrar (Pressman, 2010).

La prueba alfa se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El software se usa en un escenario natural con el desarrollador “mirando sobre el hombro” de los usuarios y registrando los errores y problemas de uso. Las pruebas alfa se realizan en un ambiente controlado (Pressman, 2010).

La prueba beta se realiza en uno o más sitios del usuario final. A diferencia de la prueba alfa, por lo general el desarrollador no está presente. Por tanto, la prueba beta es una aplicación “en vivo”

del software en un ambiente que no puede controlar el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que se encuentran durante la prueba beta y los reporta al desarrollador periódicamente (Pressman, 2010).

Se decide utilizar la prueba alfa ya que el desarrollador está presente en la prueba del software y puede el mismo detectar los errores. A continuación, se presenta el caso de prueba (CP) correspondiente al requisito **Autenticar usuario**. En el **Anexo 4: Diseños de casos de prueba de aceptación,** pueden encontrarse los CP restantes llevados a cabo en las pruebas de aceptación realizadas a la aplicación.

Tabla 14: Diseño de caso de prueba RF: Autenticar usuario.

CP-8	**Historia de Usuario 8**
Nombre: Autenticar usuario.	
Descripción: Permitir que el usuario se autentique en el Sistema de Gestión. Para autenticarse hay que: - Ingresar los datos necesarios (usuario y contraseña).	
Condiciones de ejecución: Usuario registrado en el dominio UCI.	
Pasos de ejecución: - Cuando el usuario entra al sistema se muestra una ventana donde debe ingresar sus datos (usuario y contraseña), luego selecciona la opción aceptar y si son correctos inicia sesión en el sistema. - Si los datos están incompletos o incorrectos se señalarán los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción. - Si selecciona la opción Cancelar regresará a la vista previa.	
Resultado esperado: Que el usuario acceda al Sistema de Gestión.	

Para las pruebas de aceptación se diseñaron veinticuatro casos de prueba de los cuales en la primera iteración se obtuvieron tres no conformidades de cinco casos de prueba, las cuales fueron corregidas satisfactoriamente. En la segunda iteración se obtuvieron dos casos de pruebas, y no se identificaron no conformidades. Los resultados son mostrados en la gráfica siguiente:

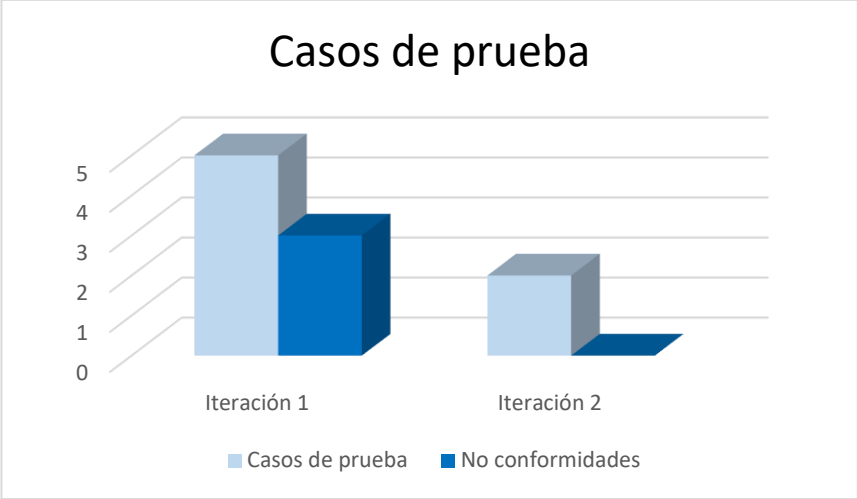


Ilustración 9: Casos de prueba por iteración. Elaboración propia.

3.4 Conclusiones del capítulo

En este capítulo se definieron temas relacionados con los procesos de implementación y prueba de la aplicación Ómnibus UCI, cumpliendo las tareas de la investigación 7 y 8, lo que se evidencia en:

- ✓ La utilización de los estándares de codificación, la cual permitió el mejoramiento de la comunicación entre los equipos de desarrollo futuros, reduciendo los errores de programación y mejorando la calidad del software.
- ✓ Las pruebas realizadas verificaron y validaron el correcto funcionamiento de la aplicación.

CONCLUSIONES GENERALES

Con el presente trabajo de diploma se logró dar cumplimiento a las tareas de investigación propuestas, logrando obtener como resultado una aplicación para dispositivos móviles sobre los recorridos de ómnibus de la Universidad de las Ciencias Informáticas. La investigación permitió arribar a las siguientes conclusiones:

- ✓ La caracterización y revisión del estado del arte de los Sistemas de información geográficos para dispositivos que operan sobre la plataforma Android; permitió afirmar que las soluciones que existen hoy, tributan a la investigación, pero no resuelven la problemática planteada, expresándose la necesidad del desarrollo de la propuesta de solución.
- ✓ Las herramientas utilizadas permitieron conocer las características que ofrecen las mismas para el desarrollo de la aplicación Ómnibus UCI.
- ✓ La utilización del patrón arquitectónico posibilitó la separación del Modelo y la Vista, lo cual logró separar los Datos de su representación visual.
- ✓ Se realizó la implementación de la aplicación Android Ómnibus UCI para dispositivos móviles, la cual facilita la información a los trabajadores de la Universidad de las Ciencias Informáticas sobre el recorrido de los ómnibus y las paradas establecidas.
- ✓ Las pruebas realizadas ayudaron en la detección de errores existentes en la propuesta de solución y para probar si esta cumplía con los requisitos del cliente.

RECOMENDACIONES

Realizar el completo desarrollo de la aplicación debido a las ventajas que ofrece a los trabajadores de la institución.

REFERENCIAS BIBLIOGRÁFICAS

- (Entrevista método) E, Agazzi. El bien, el mal y la ciencia... Madrid, España.: Tecnos S.A, .(1996).
- ABC. (23 de Abril de 2016). Definición ABC. Obtenido de Definición ABC: http://www.definicionabc.com/geografia/geolocalizacion.php
- Aitana. Visual Studio Code: Funcionalidades y extensiones. [Publicado el: 16 de octubre de 2018].
- Alegsa, Leandro. Diccionario de Informática y Tecnología. www.alegsa.com.ar. [En línea] [Citado el: 2015 de Noviembre de 18.] http://www.alegsa.com.ar/Dic/software.php.
- Alegsa, Leandro. Diccionario de informática y tecnologías. alegsa.com.ar. [En línea] 12 de Mayo de 2010. [Citado el: 22 de Febrero de 2016.]
- Aleph.org.mx, 2021.
- Anacleto, V. A. 2006. MDA Reusabilidad Orientada al Negocio.
- André, M., & López, Y. (2006). Creando un profesional con disciplina en el proceso de desarrollo de software. Ingeniería Industrial. Tomado de: https://www.redalyc.org/articulo.oa?id=360433560004
- Aplicaciones oficiales del metro de Madrid. Metro Madrid. [En línea] 7 de Abril de 2015. [Citado el: 22 de Febrero de 2016.] https://www.metromadrid.es/es/viaja_en_metro/APP/index.html.
- Bautista Q, J. M. (2012). Programación Extrema XP. Bolivia: Unión Bolivariana.
- Bermúdez, Gabriela Salazar; JIMÉNEZ, Isaac Montenegro; RODRÍGUEZ, Luis Rodríguez. USO DE PATRONES DE DISEÑO: UN CASO PRÁCTICO. *Ingeniería. Revista de la Universidad de Costa Rica*, 2012, vol. 22, no 2, p. 45-59.
- Boaventura José, C., Peña Herrera, E., Verdecia Vicet, P., & Fustiel Alvarez, Y. (2016). Elección entre una metodología ágil y tradicional basado en técnicas de soft computing. Revista Cubana de Ciencias Informáticas, 10, 145-158.
- BR. GAITAN María Elizabeth, BR. Meléndez Valladarez, Sintya Milena y BR. Pérez Reyes Neldin Noel. Metodología Ágil de Desarrollo de Software. Programación Extrema. [Citado el: 28 ENERO 2016.]. https://repositorio.unan.edu.ni/1365/1/62161.pdf
- Burton, Michael y Felker, Donn. Android™ Application Development For Dummies 2ND EDITION. Hoboken, New Jersey : John Wiley & Sons, Inc., 2012 .
- BUSCHMANN, F. 1996. Model–view–controller.
- Caldera Vergara, Roberto, et al. Estudio del framework de desarrollo web Django. 2017.

- Capterra. Visual Paradigm, 2002. Tomado a partir de: https://www.capterra.mx/software/145716/visual-paradigm.
- Carlos del Porto Blanco. De los sistemas de posicionamiento satelital. [Accedido 28mayo de 2020]. Recuperado a partir de: http://www.granma.cu/ctrl-f/2020-05-28/de-los-sistemas-de-posicionamiento-satelital-28-05-2020-01-05-02
- Casanova, Josep. 2004. desarrolloweb.com. Usabilidad y Arquitectura del Software. [En línea 2004]. [Citado el: 17 de febrero de 2016]. http://www.desarrolloweb.com/1622.php.
- Ceballos, Karla. Ingeniería de Software. Mayo de 2015.
- Change, Extreme Programming Explained*.* Beck, Kent y Wesley, Addison. S.L: Embrace, 1999.
- Cubadebate. [En línea] 30 de Enero de 2015. [Citado el: 12 de Enero de 2016.] http://www.cubadebate.cu/noticias/2015/01/30/descargue-en-su-movil-el-andariego-un-servicio-de- localizacion-para-cuba/.
- Definición .DE, 2014
- Digital Learning SL. Academia Android. [En línea] 11 de Diciembre de 2014. [Citado el: 3 de Marzo de 2016.] Tomado de: http://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/.
- Digital Millennium Copyright Act (DMCA). APPSAPK.Osmand. [En línea] [Citado el: 17 de Febrero de 2016.] http://www.appsapk.com/osmand-maps-navigation/.
- Figueroa Portilla, Carlos Saussure. Educación Set 2016, Volume 25 Nº 49 Páginas 29 – 44
- Franky, M. C. 2010. MDA: Arquitectura Dirigida por Modelos.
- Gamma, Erich, et al. Design Patterns. Elements of Reusable Software. USA : Addison-Wesley, 1995. s.n.
- GeeksforGeeks. Prueba unitaria en Python – Unittest. Febrero de 2021.
- Google Play. Metro de Madrid. [En línea] 1.13, Google Inc., 7 de Abril de 2015. [Citado el: 22 de Febrero de 2016.] 5000
- Granma, 2019. Realizado Balance de Trabajo del 2018 en el Ministerio del Transporte. Presidencia de Cuba [en línea]. 20 marzo 2019. [Accedido 24 junio 2021]. Recuperado a partir de: https://www.presidencia.gob.cu/es/noticias/realizado-balance-de-trabajo-del-2018-en-el-ministerio-del-transporte/
- Granma, ÓRGANO OFICIAL DEL COMITÉ CENTRAL DEL PARTIDO COMUNISTA DE CUBA. 12 de marzo de 2020

- Gregory, S. Parallel. Logic Programming in PARLOG. The language and its implementation. s.l.: Addison Wesley, 1987
- Gutiérrez J. J., "¿Qué es un Framework Web?", [En línea] Disponible en: http://www.cssblog.es/guias/Framework.pdf.
- Gutiérrez, Ángel Pablo Hinojosa. *Python paso a paso*. Grupo Editorial RA-MA, 2016.
- Huaraca, Abner Valdez. 2013. Pruebas de Sistemas y Pruebas de Aceptación. s.l. : FIS-UNICA, 2013.
- Infante, S. 2012. Curso Django para perfeccionistas con deadlines. 2012.
- Jacobson, Ivar y Rumbaugh., Grady Booch y James. El Proceso Unificado de Desarrollo de Software. . s.l. : Addison Wesley, 1999
- Jeffries, R., Anderson, A., Hendrickson, C. "Extreme Programming Installed". Addison-Wesley. 2001
- JMPergar. Androcode. MapsForge: OpenStreetMap en Android. [En línea] Diciembre de 2012. [Citado el: 15 de Febrero de 2016.] http://code.google.com/p/mapsforge.
- Joskowicz, J. (2008). Reglas y prácticas en eXtreme Programming. *vol. 22*.
- Kerlinger, F. Investigación del comportamiento. México, D.F.: McGraw-Hill. 1997: 502
- Larman, C. 2004. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. s.l. : ISBN 0-13-148906-2, 2004.
- Larman, Craig. UML y Patrones. 2da Edición 9 de marzo de 2016. Tomado a partir de: https://mega.nz/#!JYoCnRTD
- Legarretaetxebarria, A. (2011). Sistema de localización y seguimiento de personas en interiores mediante cámara PTZ basado en las tecnologías Kinect y Ubisense.Donostia - San Sebastián.
- León, Rolando Alfredo Hernández y González, Sayda Coello, 2020. El proceso de investigación científica. Editorial Universitaria (Cuba). ISBN 978-959-16-1307-3.
- Manso Guerra, Yerandy; González, Roxana Cañizares; Febles, Juan Pedro. Revista Cubana de Ciencias Informáticas vol.10 no.2 La Habana abr.-jun. 2016.
- Marroquín, Mónica Lucía Tapia. Estudio y desarrollo de aplicaciones para dispositivos móviles android. Ibarra : s.n., 2013.
- Martín Maldonado, Daniel. Empresa&Economía. [En línea] 1 de Julio de 2008. [Citado el: 2 de Diciembre de 2015.] http://empresayeconomia.republica.com/aplicaciones-para-empresas/sqlite-el- motor-de-base-de-datos-agil-y-robusto.html.

- Martínez, Francisco Javier. Guía de construcción de software en java con patrones de diseño. Oviedo: s.n., 2011.
- Mentor, Aula; PROGRAMACI, S. Desarrollo de aplicaciones para Android I. D. Robledo, Desarrollo de aplicaciones para android I, 2014.
- Meza González, Juan David. ProgramarYa [Citado en: agosto de 2021]. Tomado de: https://www.programarya.com/Cursos/Java/Librerias
- Muradas, Yanina. Qué es Gradle: La herramienta para ser más productivo desarrollando 25 de Febrero de 2020. Tomado a partir de: https://openwebinars.net/blog/que-es-gradle/
- Ocaña, Alexander Ortiz, 2009. Temas pedagógicos, didácticos y metodológicos. Alexander Ortiz Ocaña. ISBN 978-958-33-8560-5.
- Orjuela Duarte, A., & Rojas C, M. (24 de Mayo de 2008). Las Metodologías de desarrollo Ágil como una oportunidad para la Ingeniería del software educativo. Recuperado el 2015, de http://www.bdigital.unal.edu.co/15430/1/10037-18216-1-PB.pdf
- Paredes. M.I.M.A.V. «Universidad del Valle - Bolivia,». [En línea] 24 de Junio de 2012. [Citado el: 2016 de Febrero de 17.] http://www.univalle.edu/publicaciones/brujula/brujula19/pagina15.htm.
- PARIS ANDO. Metro de Paris (Actualizado 2020) - Lineas, mapas, planos, pases y precios. https://www.parisando.com/metro/
- PMT “Project Management Team”. Gestión de Proyectos. 30 de mayo de 2019
- Portafolio Digital. Karla Cevallos. Metodología de Desarrollo Ágil: XP y Scrum. 4-8 de Mayo del 2015
- POZO-RUZ, A., et al. Sistema de posicionamiento global (GPS): descripción, análisis de errores, aplicaciones y futuro. ETS ingenieros de Telecomunicaciones. Universidad de Malaga, 2000.
- Pressman, Roger S. Ingeniería del Software : Un Enfoque Práctico. 6ta Edición. 1998.
- Pressman, Roger. 2010. Ingeniería de Software. Un enfoque Práctico. 2010.
- PROANDROID, 2021. Android es el sistema operativo preferido para smartphones. Pro Android - Mejores aplicaciones, juegos, tutoriales y reviews [en línea]. 6 junio 2021. [Accedido 23 junio 2021]. Recuperado a partir de: https://www.proandroid.com/android-sistema-operativo-preferido-para-smartphones/
- Programador Clic Tomado de: https://programmerclick.com/article/88171495957/
- Python. Development Tools. 2021. Tomado de: https://docs.python.org/3/library/unittest.html

- Quispe Parí, Doris Joselin; Sánchez Mamani, Griselda. Encuestas y entrevistas en investigación científica. Revista de actualización clínica investiga, 2011, p. 490.
- Rivera, Yeicy Juliana Molina; CARDONA, Jonathan Sandoval; FRANCO, Santiago Alberto Toledo. Sistema operativo Android: características y funcionalidad para dispositivos móviles. 2012. Tesis Doctoral. Universidad Tecnológica de Pereira. Facultad de Ingenierías Eléctrica, Electrónica, Física y Ciencias de la Computación. Ingeniería de Sistemas y Computación.
- Rodríguez, Tamara. 2015. Metodologia de desarrollo para la actividad productiva de la UCI. La Habana : s.n., 2015.
- Rubio, Juan Carlos el 25 de Febrero de 2019. Metodologías y Herramientas. Que es GIT y para qué sirve. Tomado a partir de: https://openwebinars.net/blog/que-es-git-y-para-que-sirve/
- Sacristán. Programación en Android. Ministerio de Educación
- Sánchez, T. R. 2015. Metodología de desarrollo para la Actividad productiva de la UCI. La Habana : s.n., 2015.
- Santovenia Díaz, Javier; Tarragó Montalvo, Consuelo; Cañedo Andalia, Rubén. Sistemas de información geográfica para la gestión de la información. Acimed, 2009, vol. 20, no 5, p. 72-75. Recuperado a partir de: https://geoinnova.org/cursos/componentes-sistema-informacion-geografica-sig/
- Sarmiento, J. 2016. UML: Diagrama de despliegue. Obtenido de Visión general de los diagramas de despliegue. [En línea] Febrero de 2016. http://umldiagramadespliegue.blogspot.com/
- SIG-Rutas. Ramirez Martín, Carlos Enrique, García Avila, Adrian y Pérez Vazquez, Maridalia. La Habana: s.n., 12 de diciembre de 2011, Serie Científica, pág. 12.
- Solís, Manuel Calero. Una explicación de la programación extrema (XP). Willi. Net, 2003.
- Sommerville, Ian. 2011. Ingeniería de Software. México: Pearson Educación de México, 2011. ISBN: 978-607-32-0603-7.
- Sygic. apkpure.com. [En línea] 12 de Febrero de 2016. [Citado el: 17 de Febrero de 2016.] https://apkpure.com/gps-navigation-maps-sygic/com.sygic.aura
- Tomas David, ¿Qué son los stakeholders y cómo afectan a tu empresa? Publicado en junio de 2020.
- Torrado Fonseca, Mercè. Estudio de encuesta. 2004.
- Torres, Cristina. la-verdadera-historia-de-android-nacimiento-del-sistema-operativo-

2003/. AndroidSIS. [En línea] AB Internet Networks 2008 S.L, 6 de Junio de 2014. [Citado el: 17 de Febrero de 2016.] http://www.androidsis.com/la-verdadera-historia-de-android-nacimiento-del-sistema- operativo-2003/.

- UNIR. ¿Qué es un IDE en programación? Ingeniería y Tecnología. Accedido el [6 de julio de 2021]. Tomado a partir de https://www.unir.net/ingenieria/revista/ide-programacion/
- Urquiza, Jorge Reinier. 2013. Aplicación Web para estudiantes aventajados en la asignatura Idioma Extranjero III de la UCI. La Habana: s.n., 2013.
- Visual Paradigm Copyright. UML, BPMN and Software Design Tools for Agile Teams. [En línea] [Citado el: 2015 de Diciembre de 2.] https://www.visual-paradigm.com/.
- Wei, Meng Lee. Android™ Application Development. Indianapolis, Indiana : John Wiley & Sons, Inc., 2013. ISBN: 978-1-118-17767-9.
- Yagüe, Carlos. Qué es Apache Maven. 29 de Abril de 2019. Tomado a partir de: https://openwebinars.net/blog/que-es-apache-maven/

ANEXOS

Anexo 1: Encuesta.

Hola. Estamos realizando una encuesta para que usted nos de su opinión acerca del funcionamiento del transporte de los trabajadores de la Universidad, marque con una (x) su respuesta:

1- ¿Conoce todas las rutas de los ómnibus?

_SI _NO

2- ¿Conoce todas las paradas de los ómnibus?
_SI _NO

3- ¿El ómnibus que usted acostumbra a coger llega a la hora que le corresponde?

_SI _NO

4- ¿Alguna vez se ha confundido de ómnibus?

_ dos veces _más de cinco veces _nunca

5- ¿Sabe los horarios de salida y entrada de los ómnibus a la universidad?

_SI _NO

6- ¿Alguna vez ha fallado algún ómnibus y usted no se ha enterado?

_SI _NO

7- ¿Utilizaría usted una aplicación para estar informado(a) sobre las paradas, las rutas y los horarios de los ómnibus?

_SI _NO

8- ¿Por qué? _____.

Muchas gracias por su tiempo.

Anexo 2: Resultados de la entrevista

De la encuesta realizada se arrojó que el 50 % de las personas no tienen conocimiento acerca de las paradas y rutas de los ómnibus, y al menos un 20 % alguna vez no ha conocido que un ómnibus no saldrá, por lo que se han quedado en la parada. Por ello es necesario la realización de una aplicación para dispositivos móviles que contenga toda la información referente a los recorridos de ómnibus de la Universidad de las Ciencias Informáticas.

Anexo 3: Historias de Usuario.

Tabla 15: HU-Navegar en el mapa.

Historia de Usuario		
Número: HU_1	**Nombre:** Navegar en el mapa.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 1	
Complejidad: Alta	**Tiempo real:**	
Descripción: **1. Objetivo:** -El sistema debe permitir que el usuario pueda navegar en el mapa. **2. Acciones para lograr el objetivo:** Para navegar en el mapa hay que: - Cargar el mapa en la Base de Datos. **3. Flujo de la acción a realizar:** -El sistema debe permitir navegar en el mapa. - Acceder a la aplicación.		

Observaciones:
Prototipo de interfaz: 

Tabla 16: HU-Mostrar ubicación en el mapa.

Historia de Usuario		
Número: HU_2	**Nombre:** Mostrar ubicación en el mapa.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 1	
Complejidad: Alta	**Tiempo real:**	
Descripción: 1. **Objetivo:** - El sistema debe mostrarle al usuario su ubicación en el mapa, una vez activada la ubicación geográfica en el dispositivo.		

2. Acciones para lograr el objetivo: Para mostrar la ubicación en el mapa hay que: - Asegurarse de que el GPS debe estar activado. **3. Flujo de la acción a realizar:** - El sistema debe permitir mostrar la ubicación en el mapa. - Cuando el usuario accede a la aplicación, selecciona el ícono que hace referencia a la posición actual del usuario.
Observaciones:
Prototipo de interfaz: 

Tabla 17: UH-Mostrar posibles rutas a tomar.

Historia de Usuario		
Número: HU_3	**Nombre:** Mostrar posibles rutas a tomar.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 1	

Complejidad: Alta	**Tiempo real:**

Descripción:

1. **Objetivo:**

- El sistema debe permitirle al usuario visualizar las posibles rutas que pueda tomar.

2. **Acciones para lograr el objetivo:**

Para visualizar las posibles rutas a tomar hay que:

- Cargar el mapa en la Base de Datos.

3. **Flujo de la acción a realizar:**

- El sistema debe permitir mostrar las posibles rutas a tomar.

- Cuando el usuario accede a la aplicación, oprime sobre su posición actual y se muestra una ventana en la cual selecciona la opción “Rutas cercanas”.

Observaciones:

Prototipo de interfaz:

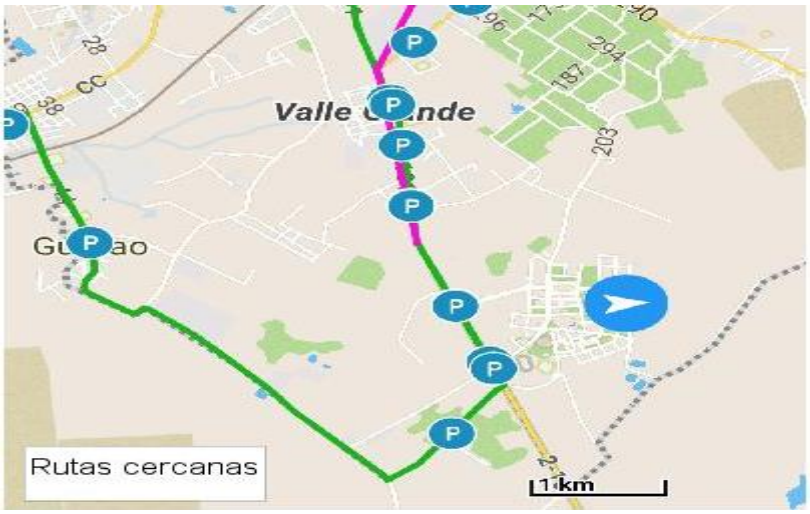


Tabla 18: HU-Localizar parada más cercana al punto de destino.

Historia de Usuario		
Número: HU_5	**Nombre:** Localizar parada más cercana al punto de destino.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 2	
Complejidad: Alta	**Tiempo real:**	

Descripción:

1. **Objetivo:**

- El sistema debe permitirle al usuario visualizar la parada más cercana a su destino.

2. **Acciones para lograr el objetivo:**

Para localizar la parada más cercana al punto de destino hay que:

- Seleccionar el punto de destino.

3. **Flujo de la acción a realizar:**

- El sistema debe permitir localizar la parada más cercana al punto de destino.

- Cuando el usuario accede a la aplicación, oprime sobre el lugar correspondiente a su destino y se muestra una ventana en la cual selecciona la opción “Parada más cercana”.

Observaciones:

Prototipo de interfaz:

Tabla 19: HU- Visualizar puntos de interés en el mapa.

Historia de Usuario	
Número: HU_6	**Nombre:** Visualizar puntos de interés en el mapa.
Programador: Adriana Osuna Miranda	**Iteración asignada:** 2
Prioridad: Media	**Tiempo estimado:** 0.5
Complejidad: Media	**Tiempo real:**
Descripción: 1. **Objetivo:**	

- El sistema debe permitirle al usuario crear, cambiar de posición, modificar el nombre y eliminar los puntos de interés creados por él.

2. Acciones para lograr el objetivo:

Para visualizar los puntos de interés en el mapa hay que:

- Cargar el mapa en la Base de Datos.

3. Flujo de la acción a realizar:

- El sistema debe permitir visualizar los puntos de interés en el mapa.

- Cuando el usuario accede a la aplicación, oprime sobre el lugar que desee marcar como “lugar favorito” y le dará un nombre al mismo. También podrá modificar y eliminar el nombre cuando lo prefiera.

Observaciones:

Prototipo de interfaz:

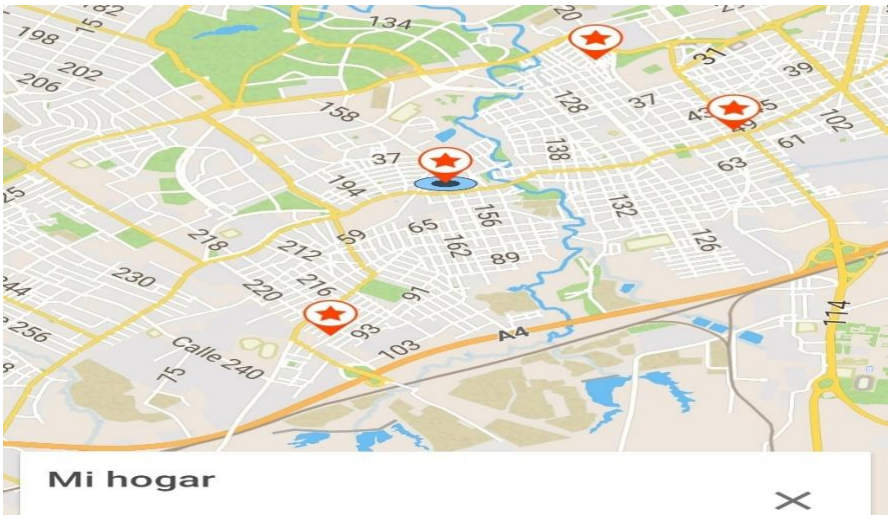


Tabla 20: HU-Mostrar paradas por rutas.

Historia de Usuario

Número: HU_7	**Nombre:** Mostrar paradas por rutas.
Programador: Adriana Osuna Miranda	**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 2
Complejidad: Alta	**Tiempo real:**

Descripción:

1. **Objetivo:**

- El sistema debe permitirle al usuario seleccionar una ruta para consultar las paradas asociadas a la misma.

2. **Acciones para lograr el objetivo:**

Para mostrar las paradas por rutas hay que:

- Selecciona la ruta deseada.

3. **Flujo de la acción a realizar:**

- El sistema debe permitir visualizar los puntos de interés en el mapa.

- Cuando el usuario accede a la aplicación, oprime sobre el lugar que desee marcar como “lugar favorito” y le dará un nombre al mismo. También podrá modificar y eliminar el nombre cuando lo prefiera.

Observaciones:

Prototipo de interfaz:

Tabla 21: HU-Autenticar usuario.

Historia de Usuario		
Número: HU_8	**Nombre:** Autenticar usuario.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	
Complejidad: Media	**Tiempo real:**	

Descripción:

1. **Objetivo:**

Permitir que el usuario se autentique en el sistema.

2. **Acciones para lograr el objetivo:**

Para autenticarse hay que:

- Ingresar los datos necesarios (usuario y contraseña).

3. Flujo de la acción a realizar:

- El sistema debe permitir autenticarse.

- Cuando el usuario entra al sistema se muestra una ventana donde debe ingresar sus datos (usuario y contraseña), luego selecciona la opción “Aceptar”. Si son correctos sus datos, inicia sesión en la aplicación.

- Si los datos están incompletos o incorrectos se señalarán los campos en cuestión dando la posibilidad al usuario de realizar nuevamente la acción.

Observaciones:

Prototipo de interfaz gráfica de usuario:

Tabla 22: HU-Insertar usuario.

Historia de Usuario	
Número: HU_9	**Nombre:** Insertar usuario.

Programador: Adriana Osuna Miranda	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5
Complejidad: Media	**Tiempo real:**

Descripción:

1. **Objetivo:**

- Permitir insertar un usuario en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para insertar un usuario en el sistema hay que:

- Ingresar los datos necesarios (usuario y contraseña).

- Usuario registrado en el dominio UCI, con permiso de Administrador.

3. **Flujo de la acción a realizar:**

- El sistema debe permitir autenticarse en el Sistema de Gestión.

- Cuando el usuario entra al sistema y accede a la sección Autenticación y Autorización > Usuarios; en el botón “Añadir Usuario” se muestra una ventana donde se aprecian los campos necesarios para insertar un usuario, luego selecciona la opción “Guardar”.

- Si selecciona la opción “Guardar y añadir otro” tendrá la posibilidad de guardar los datos anteriormente agregados y seguir añadiendo usuarios al Sistema de Gestión.

- Si selecciona la opción “Guardar y continuar editando” podrá guardar los datos insertados y continuar con la edición de los mismos.

Observaciones:

Prototipo de interfaz:

Tabla 23: HU-Mostrar usuario.

Historia de Usuario		
Número: HU_10	**Nombre:** Mostrar usuario.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	
Complejidad: Media	**Tiempo real:**	
Descripción: 1. **Objetivo:** - Permitir mostrar los datos de un usuario en el Sistema de Gestión. 2. **Acciones para lograr el objetivo:**		

Para mostrar un usuario en el sistema hay que:

- Haber al menos un usuario existente en el sistema.

3. Flujo de la acción a realizar:

- El sistema debe permitir mostrar los datos de un usuario en el Sistema de Gestión.

- Cuando el usuario entra al sistema y accede a la sección Autenticación y Autorización > Usuarios; obtendrá toda la información referente a estos.

Observaciones:

Prototipo de interfaz:

Tabla 24: HU-Modificar usuario.

Historia de Usuario	
Número: HU_11	**Nombre:** Modificar usuario.
Programador: Adriana Osuna Miranda	**Iteración asignada:** 1

Prioridad: Alta	**Tiempo estimado:** 0.5
Complejidad: Media	**Tiempo real:**

Descripción: Funcionalidad que permitirá modificar un usuario en la aplicación.

1. **Objetivo:**

- Permitir modificar un usuario en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para modificar un usuario en el sistema hay que:

- Seleccionar un usuario y modificar los datos existentes.

- Estar registrado en el dominio UCI, con permiso de Administrador.

3. **Flujo de la acción a realizar:**

- El sistema debe permitir modificar un usuario en el Sistema de Gestión.

- Cuando el usuario entra al sistema y accede a la sección Autenticación y Autorización > Usuarios; selecciona el nombre de usuario a modificar y apararecerá la vista de “Edición de Usuario”.

Observaciones:

Prototipo de interfaz:

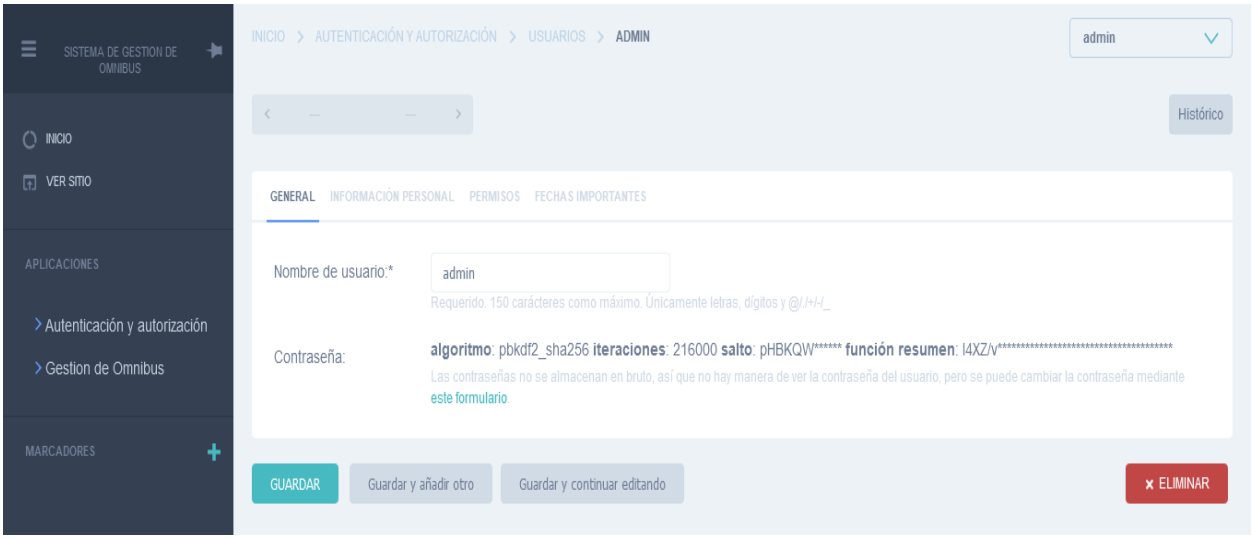


Tabla 25: HU-Eliminar usuario.

Historia de Usuario		
Número: HU_12	**Nombre:** Eliminar usuario.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	
Complejidad: Media	**Tiempo real:**	
Descripción: 1. **Objetivo:** - Permitir que se elimine un usuario en el Sistema de Gestión. 2. **Acciones para lograr el objetivo:** Para eliminar un usuario en el sistema hay que:		

- Haber al menos un usuario existente en el sistema y poseer permiso de Administración.

3. Flujo de la acción a realizar:

- Cuando el usuario entra al sistema y accede a la sección Autenticación y Autorización > Usuarios; elige el usuario a eliminar y selecciona la opción “Eliminar usuarios seleccionados”.

Observaciones:

Prototipo de interfaz:

Tabla 26: HU-Insertar ómnibus.

Historia de Usuario	
Número: HU_13	**Nombre:** Insertar ómnibus.
Programador: Adriana Osuna Miranda	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5
Complejidad: Media	**Tiempo real:**

Descripción:

1. **Objetivo:**

- Permitir que se inserte un ómnibus en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para insertar un ómnibus en el sistema hay que:

- Ingresar los datos necesarios (nombre, activo).

- Poseer permiso de Administración.

3. **Flujo de la acción a realizar:**

- Cuando el usuario entra al sistema y accede a la sección Gestión de Ómnibus > Ómnibus; en el botón “Añadir Ómnibus” se muestra una ventana que proporciona los campos necesarios para insertar un ómnibus, luego selecciona la opción Guardar.

Observaciones:

Prototipo de interfaz:

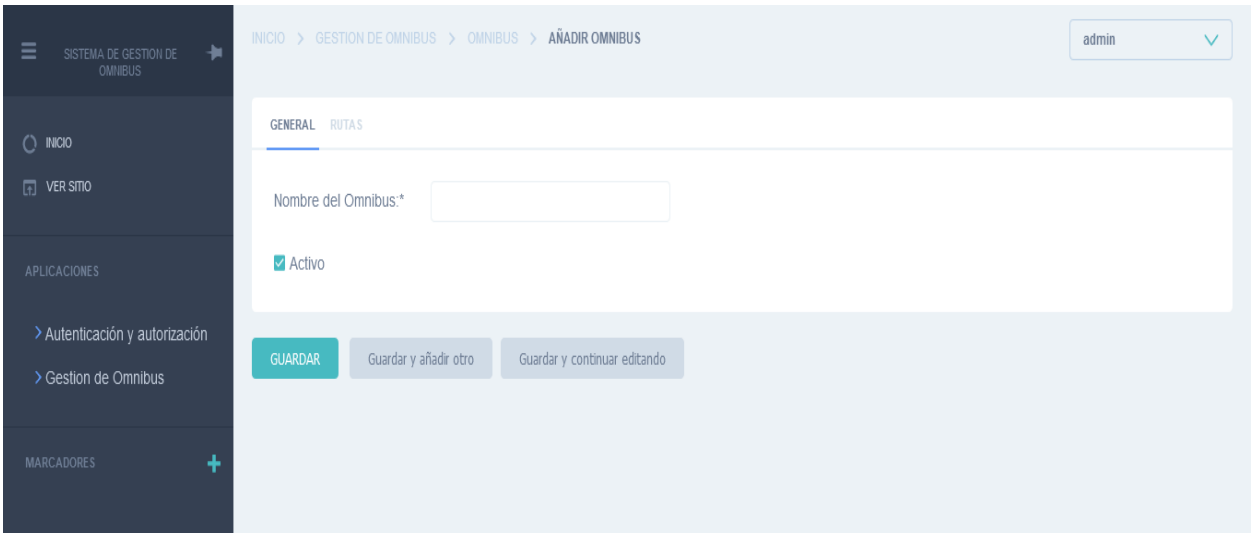


Tabla 27: HU-Mostrar ómnibus.

Historia de Usuario

Número: HU_14	**Nombre:** Mostrar ómnibus.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	
Complejidad: Media	**Tiempo real:**	

Descripción: Permitir mostrar los ómnibus en la aplicación.

1. **Objetivo:**

- Permitir mostrar los ómnibus en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para mostrar los ómnibus en el sistema hay que:

- Estar registrado en el dominio UCI, con permiso de Administrador.

- Haber al menos un ómnibus existente en el sistema.

Flujo de la acción a realizar:

- Cuando el usuario entra al sistema y accede a la sección Gestión de Ómnibus > Ómnibus; obtendrá toda la información referente a estos.

Observaciones:

Prototipo de interfaz:

Tabla 28: HU-Modificar ómnibus.

Historia de Usuario		
Número: HU_15	**Nombre:** Modificar ómnibus.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	
Complejidad: Media	**Tiempo real:**	

Descripción:

1. **Objetivo:**

- Permitir modificar los ómnibus en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para modificar los ómnibus en el sistema hay que:

- Ingresar los datos necesarios (nombre, activo).

- Haber al menos un ómnibus existente en el sistema.

Flujo de la acción a realizar:

- Cuando el usuario entra al sistema y accede a la sección Gestión de Ómnibus > Ómnibus; selecciona el ómnibus a modificar y apararecerá la vista de “Edición de Ómnibus” en la cual se modificarán los datos existentes.

Observaciones:

Prototipo de interfaz:

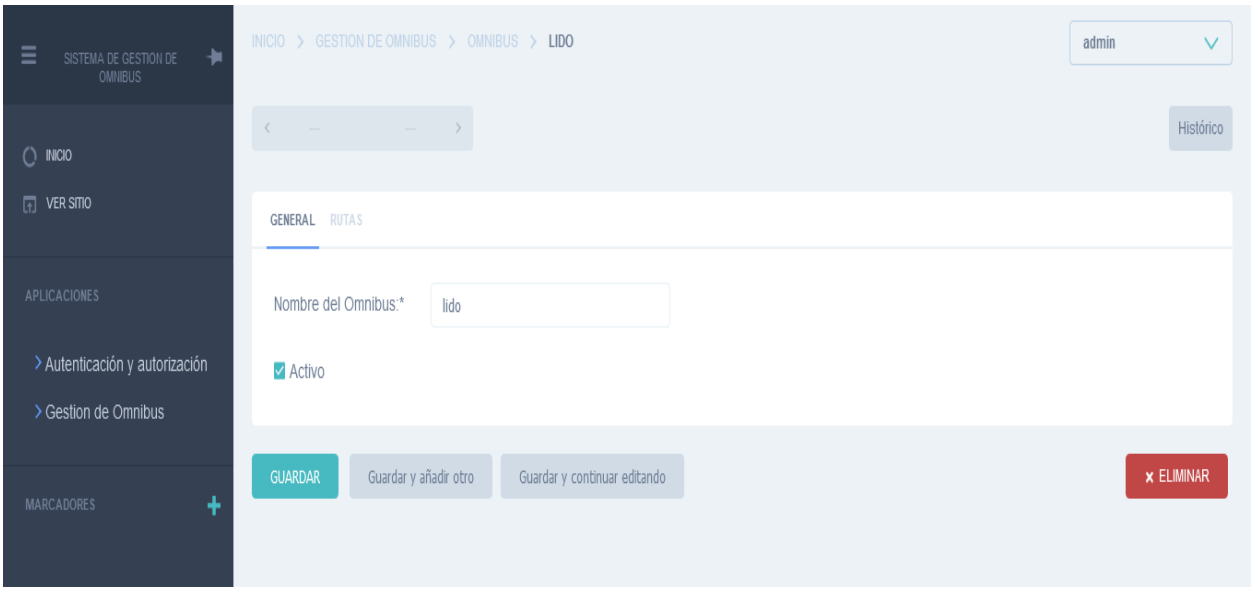


Tabla 29: HU-Eliminar ómnibus.

Historia de Usuario		
Número: HU_16	**Nombre:** Eliminar ómnibus.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	

Complejidad: Media	**Tiempo real:**

Descripción:

1. **Objetivo:**

- Permitir eliminar los ómnibus en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para eliminar los ómnibus en el sistema hay que:

- Haber al menos un ómnibus existente en el sistema.

- Poseer permiso de Administración.

Flujo de la acción a realizar:

- Cuando el usuario entra al sistema y accede a la sección Gestión de Ómnibus > Ómnibus; elige el ómnibus a eliminar y selecciona la opción “Eliminar Ómnibus seleccionados”.

Observaciones:

Prototipo de interfaz:

Tabla 30: HU-Insertar ruta.

Historia de Usuario		
Número: HU_17	**Nombre:** Insertar ruta.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	
Complejidad: Media	**Tiempo real:**	

Descripción:

1. **Objetivo:**

- Permitir insertar las rutas en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para insertar las rutas en el sistema hay que:

- Ingresar los datos necesarios (ómnibus, parada, orden, tiempodellegada, activo).

- Estar registrado en el dominio UCI, con permiso de Administrador.

Flujo de la acción a realizar:

- Cuando el usuario entra al sistema y accede a la sección Gestión de Ómnibus > Ómnibus; selecciona el ómnibus al cual se le insertará la ruta. En la pestaña Rutas se selecciona “Añadir otra Ruta”.

Observaciones:

Prototipo de interfaz:

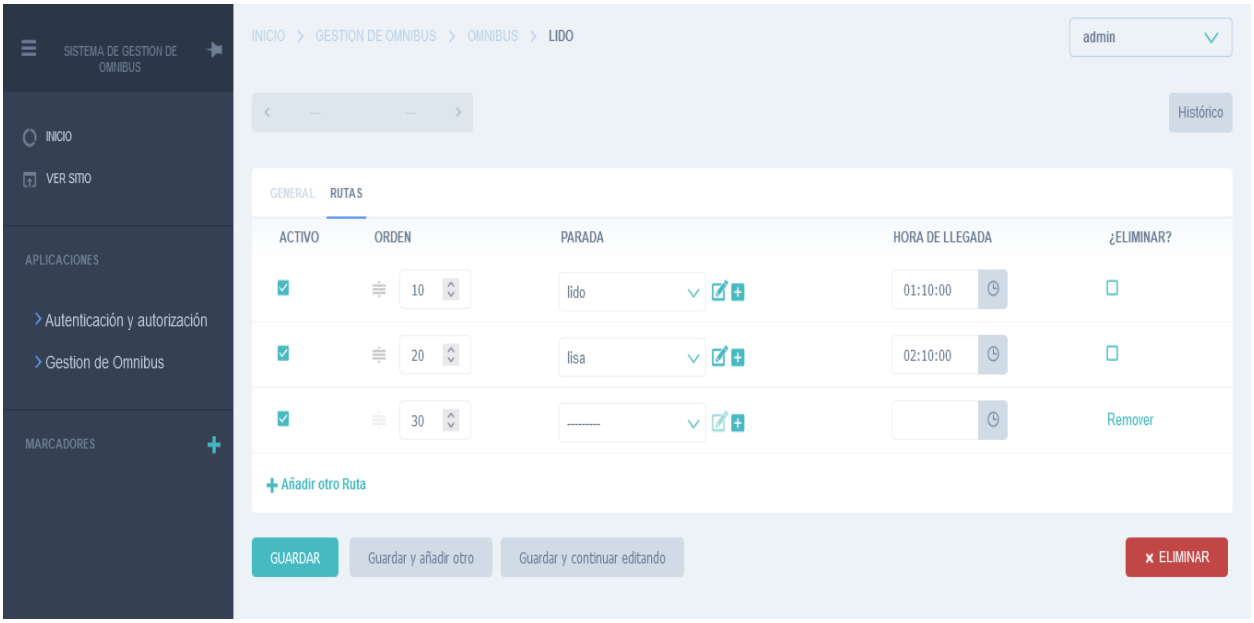


Tabla 31: HU-Mostrar ruta.

Historia de Usuario		
Número: HU_18	**Nombre:** Mostrar ruta.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	
Complejidad: Media	**Tiempo real:**	

Descripción:

1. **Objetivo:**

- Permitir mostrar los datos de una ruta en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para mostrar las rutas en el sistema hay que:

- Haber al menos una ruta existente en el sistema.

Flujo de la acción a realizar:

- Cuando el usuario entra al sistema y accede a la sección Gestionar Ómnibus; en el apartado Ruta se muestra una ventana con los datos de estas.

Observaciones:

Prototipo de interfaz:

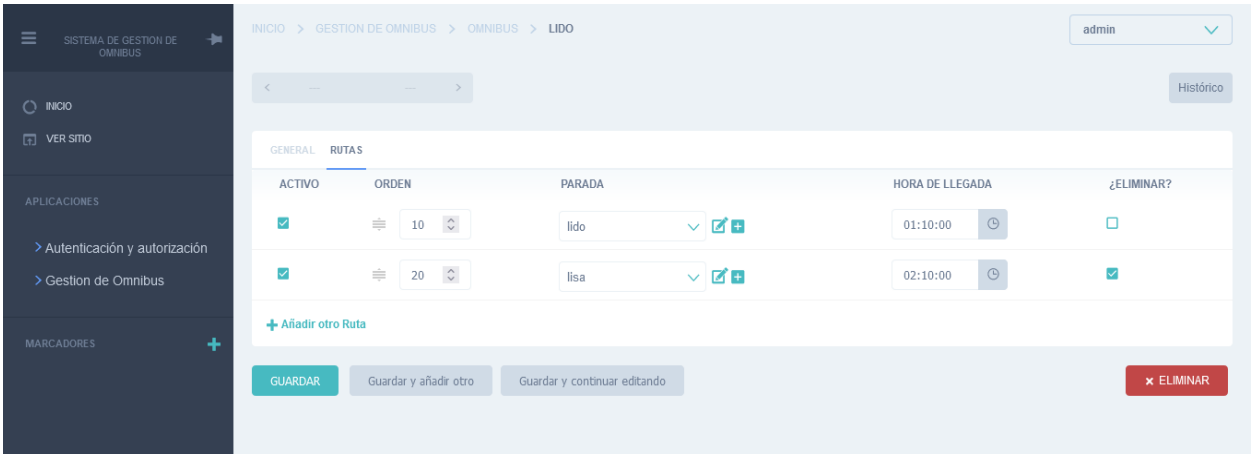


Tabla 32: HU-Modificar ruta.

Historia de Usuario	
Número: HU_19	**Nombre:** Modificar ruta.
Programador: Adriana Osuna Miranda	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5
Complejidad: Media	**Tiempo real:**

Descripción:

1. **Objetivo:**

- Permitir modificar los datos de una ruta en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para modificar las rutas en el sistema hay que:

- Haber al menos una ruta existente en el sistema.

- Ingresar los datos necesarios (ómnibus, parada, orden, tiempodellegada, activo).

Flujo de la acción a realizar:

- Cuando el usuario entra al sistema y accede a la sección Gestión de Ómnibus > Ómnibus; selecciona el ómnibus al cual se le va modificar la ruta. En la pestaña Rutas se modifica la misma.

Observaciones:

Prototipo de interfaz:

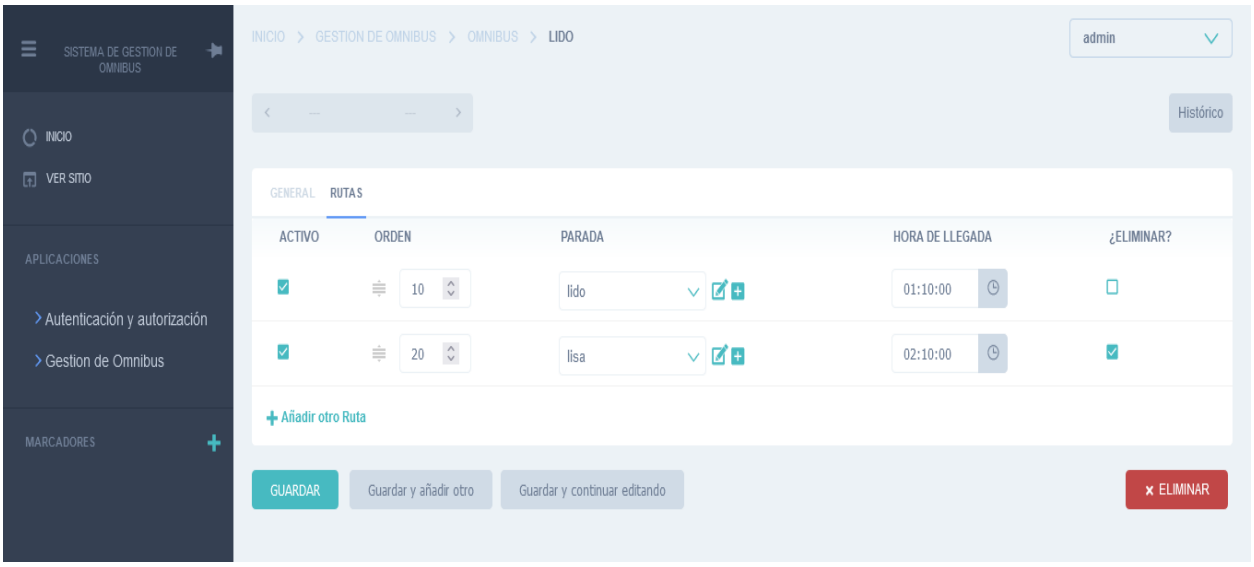


Tabla 33: HU-Eliminar ruta.

Historia de Usuario

Número: HU_20	**Nombre:** Eliminar ruta.
Programador: Adriana Osuna Miranda	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5
Complejidad: Media	**Tiempo real:**

Descripción:

1. **Objetivo:**

- Permitir eliminar los datos de una ruta en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para eliminar una ruta en el sistema hay que:

- Haber al menos una ruta existente en el sistema.

- Estar registrado en el dominio UCI, con permiso de Administrador.

3. **Flujo de la acción a realizar:**

- Cuando el usuario entra al sistema y accede a la sección Gestión de Ómnibus > Ómnibus; elige el ómnibus al que se le va a eliminar la ruta y elimina la seleccionada.

Observaciones:

Prototipo de interfaz:

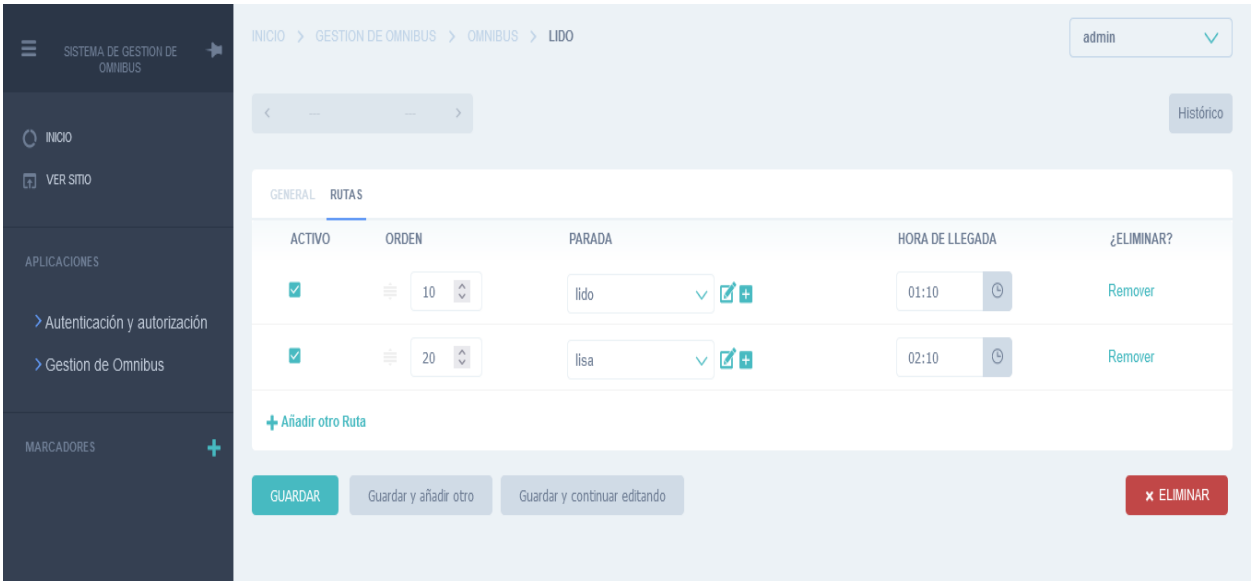


Tabla 34: HU-Insertar parada.

Historia de Usuario		
Número: HU_21	**Nombre:** Insertar parada.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	
Complejidad: Media	**Tiempo real:**	

Descripción:

1. **Objetivo:**

- Permitir insertar paradas en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para insertar una parada en el sistema hay que:

- Ingresar los datos necesarios (nombre, dirección, latitud, longitud).

- Estar registrado en el dominio UCI, con permiso de Administrador.

3. Flujo de la acción a realizar:

- Cuando el usuario entra al sistema y accede a la sección Gestión de Ómnibus > Parada; en el botón "Añadir Parada" se muestra una ventana que proporciona los campos necesarios para insertar una parada, luego selecciona la opción "Guardar".

Observaciones:

Prototipo de interfaz:

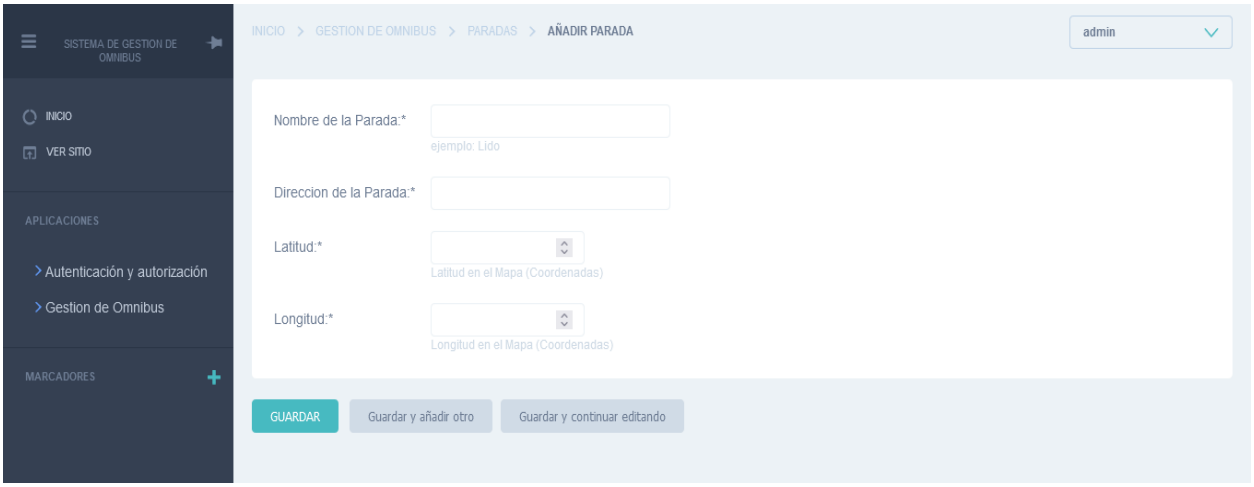


Tabla 35: HU-Mostrar parada.

Historia de Usuario		
Número: HU_22	**Nombre:** Mostrar parada.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	

Complejidad: Media	**Tiempo real:**

Descripción:

1. **Objetivo:**

- Permitir mostrar las paradas en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para mostrar las paradas en el sistema hay que:

- Estar registrado en el dominio UCI, con permiso de Administrador.

- Haber al menos una parada existente en el sistema.

3. **Flujo de la acción a realizar:**

- Cuando el usuario entra al sistema y accede a la sección Gestionar Ómnibus; en el apartado Parada se muestra una ventana con los datos de la parada.

Observaciones:

Prototipo de interfaz:

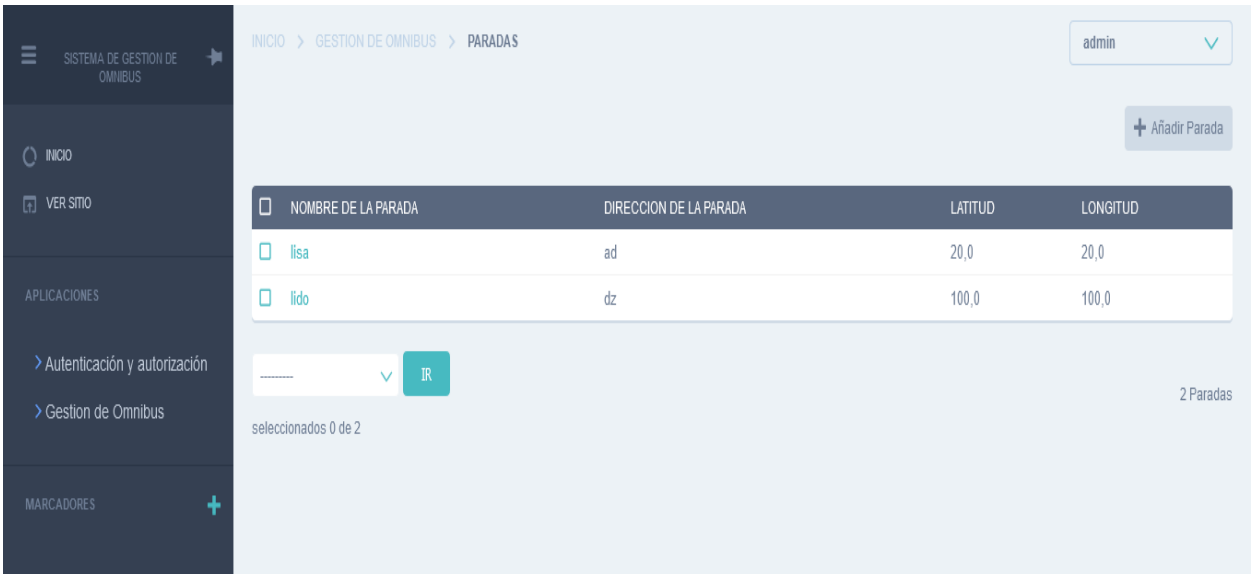


Tabla 36: HU-Modificar parada.

Historia de Usuario		
Número: HU_23	**Nombre:** Modificar parada.	
Programador: Adriana Osuna Miranda		**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5	
Complejidad: Media	**Tiempo real:**	

Descripción:

1. **Objetivo:**

- Permitir modificar las paradas en el Sistema de Gestión.

2. **Acciones para lograr el objetivo:**

Para modificar las paradas en el sistema hay que:

- Estar registrado en el dominio UCI, con permiso de Administrador.

- Haber al menos una parada existente en el sistema.

3. **Flujo de la acción a realizar:**

- Cuando el usuario entra al sistema y accede a la sección Gestionar Ómnibus; en el apartado Modificar Parada se muestra una ventana con los datos para modificar una parada y selecciona el botón “Aceptar”.

Observaciones:

Prototipo de interfaz:

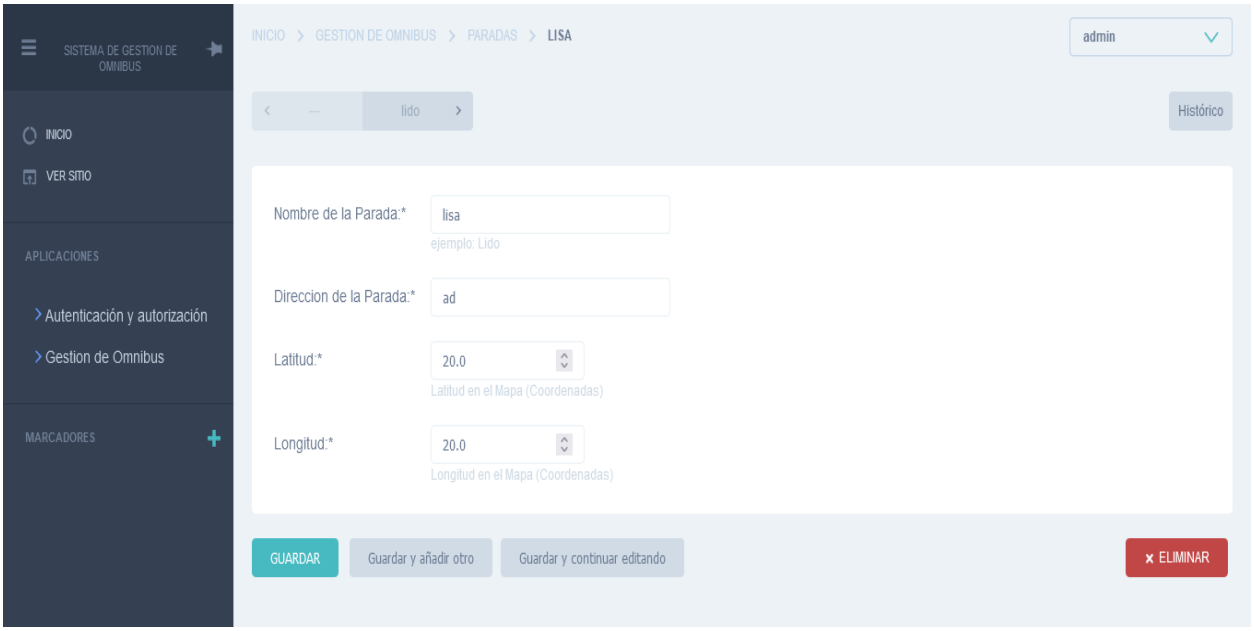


Tabla 37: HU-Eliminar parada.

Historia de Usuario	
Número: HU_24	**Nombre:** Eliminar parada.
Programador: Adriana Osuna Miranda	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 0.5
Complejidad: Media	**Tiempo real:**
Descripción: 1. **Objetivo:** - Permitir eliminar paradas en el Sistema de Gestión. 2. **Acciones para lograr el objetivo:**	

Para eliminar las paradas en el sistema hay que:

- Estar registrado en el dominio UCI, con permiso de Administrador.

- Haber al menos una parada existente en el sistema.

3. **Flujo de la acción a realizar:**

- Cuando el usuario entra al sistema y accede a la sección Gestión de Ómnibus > Parada; elige la parada que va a eliminar y selecciona la opción “Eliminar paradas seleccionadas”.

Observaciones:

Prototipo de interfaz:

Anexo 4: Diseños de casos de prueba de aceptación

Tabla 38: Diseño de caso de prueba RF: Insertar usuario.

CP-9	**Historia de Usuario 9**
Nombre: Insertar usuario.	
Descripción: Permitir que se inserte un usuario en el Sistema de Gestión.	

Para insertar hay que: -Ingresar los datos necesarios (nombre y contraseña).
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador.
Pasos de ejecución: Acceder a la sección Autenticación y Autorización > Usuarios. En el botón Insertar Usuario se muestra una ventana donde se muestran los campos necesarios para insertar un usuario, luego selecciona la opción Guardar.
Resultado esperado: Que el usuario sea añadido al sistema.

Tabla 39: Diseño de caso de prueba RF: Mostrar usuario.

CP-10	**Historia de Usuario 10**
Nombre: Mostar usuario.	
Descripción: Permitir mostrar los datos de un usuario.	
Condiciones de ejecución: Tiene que haber al menos un usuario en el sistema.	
Pasos de ejecución: Acceder a la sección Autenticación y Autorización > Usuarios.	
Resultado esperado: Que se muestren los datos de un usuario en el sistema.	

Tabla 40: Diseño de caso de prueba RF: Modificar usuario.

CP-11	**Historia de Usuario 11**
Nombre: Modificar usuario.	
Descripción: Permitir que se modifique un usuario en el sistema. Para modificar hay que seleccionar un usuario y modificar los datos existentes.	
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador.	

Pasos de ejecución: Acceder a la sección Autenticación y Autorización > Usuarios. Seleccionar el nombre de usuario a modificar y apararecerá la vista de Edición de Usuario.
Resultado esperado: Permitir que se modifique un usuario en el sistema.

Tabla 41: Diseño de caso de prueba RF: Eliminar usuario.

CP-12	**Historia de Usuario 12**
Nombre: Eliminar usuario.	
Descripción: Permitir que se elimine un usuario en el sistema.	
Condiciones de ejecución: Tiene que haber al menos un usuario en el sistema y poseer permiso de Administración.	
Pasos de ejecución: Acceder a la sección Autenticación y Autorización > Usuarios. Elegir el usuario a eliminar y escoger la opción de Eliminar usuarios seleccionado	
Resultado esperado: Que se elimine un usuario en el sistema.	

Tabla 42: Diseño de caso de prueba RF: Insertar ómnibus.

CP-13	**Historia de Usuario 13**
Nombre: Insertar ómnibus.	
Descripción: Permitir que se inserte un ómnibus en el sistema. Para insertar hay que: -Ingresar los datos necesarios (nombre, activo).	
Condiciones de ejecución: Poseer permiso de Administración.	
Pasos de ejecución: Acceder a la sección Gestión de Ómnibus > Ómnibus. En el botón Añadir Ómnibus se muestra una ventana donde que proporciona los campos necesarios para insertar un ómnibus, luego selecciona la opción Guardar.	

Resultado esperado: Que el ómnibus sea añadido al sistema.

Tabla 43: Diseño de caso de prueba RF: Mostar ómnibus.

CP-14	**Historia de Usuario 14**
Nombre: Mostar ómnibus.	
Descripción: Permitir mostrar los datos de un ómnibus.	
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador. Tiene que haber al menos un ómnibus en el sistema.	
Pasos de ejecución: Acceder a la sección Gestión de Ómnibus > Ómnibus.	
Resultado esperado: Que se muestren los datos de un ómnibus en el sistema.	

Tabla 44: Diseño de caso de prueba RF: Modificar ómnibus.

CP-15	**Historia de Usuario 15**
Nombre: Modificar ómnibus.	
Descripción: Permitir que se modifique un ómnibus en el sistema. Para modificar hay que: -Ingresar los datos necesarios (nombre, activo).	
Condiciones de ejecución: Acceder a la base de datos.	
Pasos de ejecución: Acceder a la sección Gestión de Ómnibus > Ómnibus. Seleccionar el nombre de ómnibus a modificar y apararecerá la vista de Edición de Ómnibus.	
Resultado esperado: Que el ómnibus sea modificado en el sistema.	

Tabla 45: Diseño de caso de prueba RF: Eliminar ómnibus.

CP-16	**Historia de Usuario 16**

Nombre: Eliminar ómnibus.
Descripción: Permitir que se elimine un ómnibus en el sistema.
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador. Tiene que haber al menos un ómnibus en el sistema.
Pasos de ejecución: Acceder a la sección Gestión de Ómnibus > Ómnibus Elegir el ómnibus a eliminar y escoger la opción de Eliminar Ómnibus seleccionado.
Resultado esperado: Que el ómnibus sea eliminado del sistema.

Tabla 46: Diseño de caso de prueba RF: Insertar ruta.

CP-17	**Historia de Usuario 17**
Nombre: Insertar ruta.	
Descripción: Permitir que se inserte una ruta en el sistema. Para insertar hay que: -Ingresar los datos necesarios (ómnibus, parada, orden, tiempodellegada, activo).	
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador.	
Pasos de ejecución: Acceder a la sección Gestión de Ómnibus > Ómnibus. Se selecciona el ómnibus al cual se le insertará la ruta. En la pestaña Rutas se selecciona Añadir otra Ruta.	
Resultado esperado: Que la ruta sea añadida al sistema.	

Tabla 47: Diseño de caso de prueba RF: Mostrar ruta.

CP-18	**Historia de Usuario 18**
Nombre: Mostar ruta.	
Descripción: Permitir mostrar los datos de una ruta.	

Condiciones de ejecución: Tiene que haber al menos una ruta en el sistema.
Pasos de ejecución: Acceder a la sección Gestionar Ómnibus. En el apartado Ruta se muestra una ventana con los datos de una ruta.
Resultado esperado: Que se muestren los datos de una ruta en la aplicación.

Tabla 48: Diseño de caso de prueba RF: Modificar ruta.

CP-19	**Historia de Usuario 19**
Nombre: Modificar ruta.	
Descripción: Permitir que se modifique una ruta en el sistema. Para modificar hay que: -Ingresar los datos necesarios (ómnibus, parada, orden, tiempodellegada, activo).	
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador.	
Pasos de ejecución: Acceder a la sección Gestión de Ómnibus > Ómnibus. Se selecciona el ómnibus al cual se le va modificar la ruta. En la pestaña Rutas se modifica la Ruta.	
Resultado esperado: Que la ruta sea modificada en el sistema.	

Tabla 49: Diseño de caso de prueba RF: Eliminar ruta.

CP-20	**Historia de Usuario 20**
Nombre: Eliminar ruta.	
Descripción: Permitir que se elimine una ruta en l sistema.	
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador. Tiene que haber al menos una ruta en el sistema.	

Pasos de ejecución: Acceder a la sección Gestión de Ómnibus > Ómnibus. Elegir el ómnibus al que se le va a eliminar la ruta y eliminar la ruta.
Resultado esperado: Que la ruta sea eliminada del sistema.

Tabla 50: Diseño de caso de prueba RF: Insertar parada.

CP-21	**Historia de Usuario 21**
Nombre: Insertar parada.	
Descripción: Permitir que se inserte una parada en el sistema. Para insertar hay que: -Ingresar los datos necesarios (nombre, dirección, latitud, longitud).	
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador.	
Pasos de ejecución: Acceder a la sección Gestión de Ómnibus > Parada. En el botón Añadir Parada se muestra una ventana donde que proporciona los campos necesarios para insertar una parada, luego selecciona la opción Guardar.	
Resultado esperado: Que la parada sea añadida al sistema.	

Tabla 51: Diseño de caso de prueba RF: Mostrar parada.

CP-22	**Historia de Usuario 22**
Nombre: Mostar parada.	
Descripción: Permitir mostrar los datos de una parada.	
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador. Tiene que haber al menos una parada en el sistema.	
Pasos de ejecución: Acceder a la sección Gestionar Ómnibus. En el apartado Parada se muestra una ventana con los datos de una parada.	

Resultado esperado: Que se muestren los datos de una parada en el sistema.

Tabla 52: Diseño de caso de prueba RF: Modificar parada.

CP-23	**Historia de Usuario 23**
Nombre: Modificar parada.	
Descripción: Permitir modificar los datos de una parada.	
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador. Tiene que haber al menos una parada en el sistema.	
Pasos de ejecución: Acceder a la sección Gestionar Ómnibus. En el apartado Modificar Parada se muestra una ventana con los datos para modificar una parada y se selecciona el botón Aceptar.	
Resultado esperado: Que se modifiquen los datos de una parada en el sistema.	

Tabla 53: Diseño de caso de prueba RF: Eliminar parada.

CP-24	**Historia de Usuario 20**
Nombre: Eliminar parada.	
Descripción: Permitir que se elimine una parada en el sistema.	
Condiciones de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador. Tiene que haber al menos una parada en el sistema.	
Pasos de ejecución: Acceder a la sección Gestión de Ómnibus > Parada. Elegir la parada que se le va a eliminar y eliminarla.	
Resultado esperado: Que la parada sea eliminada del sistema.	

Tabla 54: Diseño de caso de prueba RF: Navegar en el mapa.

CP-1	**Historia de Usuario 1**

Nombre: Navegar en el mapa
Descripción: Permitir al usuario navegar en el mapa.
Condiciones de ejecución: El mapa debe estar cargado en la Base de Datos.
Pasos de ejecución: Usuario registrado en el dominio UCI, con permiso de Administrador.
Resultado esperado: Que el usuario pueda navegar en el mapa.

Tabla 55: Diseño de caso de prueba RF: Mostrar ubicación en el mapa.

CP-2	**Historia de Usuario 2**
Nombre: Mostrar ubicación en el mapa	
Descripción: Permitir al usuario conocer su ubicación en el mapa.	
Condiciones de ejecución: El GPS debe estar activado.	
Pasos de ejecución: Acceder a la aplicación.	
Resultado esperado: Que el usuario pueda conocer su ubicación en el mapa.	

Tabla 56: Diseño de caso de prueba RF: Mostrar posibles rutas a tomar.

CP-3	**Historia de Usuario 3**
Nombre: Mostrar posibles rutas a tomar.	
Descripción: Permitir al usuario visualizar las posibles rutas que pueda tomar.	
Condiciones de ejecución: El mapa debe estar cargado en la Base de Datos.	
Pasos de ejecución: Acceder a la aplicación.	
Resultado esperado: Que el usuario pueda conocer su ubicación en el mapa.	

Tabla 57: Diseño de caso de prueba RF: Localizar parada más cercana a la posición del usuario.

CP-4	Historia de Usuario 4
Nombre: Localizar parada más cercana a la posición del usuario.	
Descripción: Permitir al usuario localizar la parada más cercana a su posición.	
Condiciones de ejecución: El mapa debe estar cargado en la Base de Datos.	
Pasos de ejecución: Acceder a la aplicación. Oprimir sobre su posición actual. Se muestra una ventana en la cual se selecciona la opción Paradas Cercanas.	
Resultado esperado: Que el usuario pueda conocer su ubicación en el mapa.	

Tabla 58: Diseño de caso de prueba RF: Localizar parada más cercana al punto de destino.

CP-5	Historia de Usuario 5
Nombre: Localizar parada más cercana al punto de destino.	
Descripción: Permitir al usuario localizar las posibles rutas más cercanas al punto de su destino.	
Condiciones de ejecución: Seleccionar el punto de destino.	
Pasos de ejecución: Acceder a la aplicación.	
Resultado esperado: Que el usuario pueda localizar las posibles rutas más cercanas al punto de su destino.	

Tabla 59: Diseño de caso de prueba RF: Visualizar puntos de interés en el mapa.

CP-6	Historia de Usuario 6
Nombre: Visualizar puntos de interés en el mapa.	
Descripción: Permitir al usuario visualizar puntos de su interés en el mapa.	
Condiciones de ejecución: El mapa debe estar cargado en la Base de Datos.	

Pasos de ejecución: Acceder a la aplicación.
Resultado esperado: Que el usuario pueda visualizar sus puntos de interés en el mapa.

Tabla 60: Diseño de caso de prueba RF: Mostrar paradas por rutas.

CP-7	**Historia de Usuario 7**
Nombre: Mostrar paradas por rutas.	
Descripción: Permitir al usuario visualizar las paradas de cada ruta según la ruta que haya seleccionado.	
Condiciones de ejecución: Selecciona la ruta deseada,	
Pasos de ejecución: Acceder a la aplicación.	
Resultado esperado: Que el usuario pueda localizar las posibles rutas más cercanas al punto de su destino.	