Facultad de Ciencias y Tecnologías Computacionales

Título:

Sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI (Universidad de Ciencias Informáticas)

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

Autor(es):

Sindy Nuñez Medina

Emilio Muñoz Monterrey

Tutor(es):

Dr.C. Omar Mar Cornelio

La Habana, <noviembre> de <2022>

DECLARACIÓN DE AUTORÍA

Los autores del trabajo de diploma con título **"Sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI (Universidad de Ciencias Informáticas)"**, concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara a Sindy Nuñez Medina y a Emilio Muñoz Monterrey como únicos autores de su contenido. Para que así conste firma(n) la presente a los <día> días del mes de <mes> del año <año>.

Sindy Nuñez Medina

_____
Firma del Autor

Emilio Muñoz Monterrey

_____
Firma del Autor

Dr.C. Omar Mar Cornelio

_____
Firma del Tutor

DATOS DE CONTACTO

Autores:

Autor: Sindy Nuñez Medina.

Correo: sindynm@uci.cu

Autor: Emilio Muñoz Monterrey

Correo: emiliomm@uci.cu

Tutor:

Tutor: Dr.C. Omar Mar Cornelio

Correo: omarmar@uci.cu

Síntesis del tutor: Licenciatura en la Especialidad de Informática en el año 2006, diplomado para la Formación de Investigador en el año 2012, Máster en Informática Aplicada en el año 2013, Doctor en Ciencias Técnicas 2020, Profesor Auxiliar 2017 e Investigador Auxiliar en el año 2018.

Se vinculó al Proceso Docente en la Educación Superior desde el 2007 hasta la fecha como profesor del Departamento de Sistemas Digitales de la Facultad de Ciencias y Tecnologías Computacionales, impartiendo asignaturas como Máquinas Computadoras, Redes, Arquitectura de Computadoras, Hardware de Computadoras, Internet de las Cosas entre otras, donde se evidenció un crecimiento profesional.

Se encuentra vinculado a las líneas de investigación de Automática, Inteligencia Artificial y Matemática Computacional. Sus resultados investigativos se evidencian en publicaciones en

eventos internacionales y revistas referenciadas de alto impacto. Obtuvo distinción como mejor ponencia en el 3er y 4to congreso internacional de Innovación, Tecnología y Educación ¨Civitec 2015 y 2016¨.

Obtuvo la condición de destacado municipal del trabajo en el año 2007, Obtuvo el Premio del Rector en la Categoría de trabajador no docente en el año 2011, obtuvo el Premio del Rector en la Categoría de profesor joven más destacado en el año 2016, obtuvo el Premio del Rector en la Categoría de profesor universitario integral más destacado en el año 2017

Cargo: Metodólogo en el Centro de Estudio de Matemática Computacional

Agradecimientos

Sindy

Agradezco a mis padres Arelis y Alberto por todos los sacrificios que hicieron por mí, por ser guías y brindarme su confianza y cariño incondicional, porque sin ellos no hubiese podido cumplir este sueño.

A mi hermano Albertico por ser mi amigo incondicional, compañero, confidente y por querer-me como soy.

A mi amigo, compañero y esposo Emilio por acompañarme en toda esta experiencia, por siempre estar a mi lado, por tu amor, cariño, paciencia y apoyo en todos estos años de rela-ción.

A mis compañeros de grupo, por todos los momentos de alegrías y tristezas que pasamos juntos, a mis amigos: Javier, Jesús, Melissa, Zoe, Liset, Suset, Xavi, Yaneysi, Nayalys, Fer-nando, Karel, Claudia, Thaylin, Naide etc.

A mi tutor Omar por todo su apoyo, ayuda y dedicación en la realización de este sueño.

A todos aquellos profesores que contribuyeron a mi formación como profesional.

En fin, le agradezco a todas las personas que estuvieron junto a mi en este largo camino por la UCI, a las que conocí, a las que están cerca y a las que están lejos.

Emilio

A mi tutor Omar por apoyarnos en todo este tiempo y brindarnos su gran experiencia.

Agradecer a mi colega Alain por tomar parte de su tiempo para ayudarme en este proyecto.

Agradecer a mi esposa por ser mi compañera en esta tesis

Dedicatoria

Sindy

Le dedico mi tesis a mi familia por siempre creer en mí, a mis padres por apoyarme incondicionalmente, por amarme cada día de mi vida, por cuidarme, por siempre estar ahí para mí y ser unos padres ejemplares.

A mi hermano Albertico por ser mi amigo, mi confidente, por darme siempre ánimos para ser cada día una mejor persona y por ser tan especial en mi vida.

A mi esposo Emilio, mi compañero de tesis y de la vida, por su apoyo, dedicación, amistad, comprensión, amor y cariño por soportarme durante estos 5 años y por formar parte de este sueño.

Me dedico esta tesis, por los esfuerzos realizados durante estos 5 años, por las noches que pasé despierta estudiando y los sacrificios hechos para poder cumplir este sueño.

Emilio

Dedico esta tesis a mi familia y amigos , en especial a mis tías y a mi prima que me apoyaron mucho en momentos difíciles.

A mi madre materna y de crianza , que me dieron valores y una buena educación

A mis suegros por quererme como un hijo más .

Y a mi esposa por acompañarme siempre en las buenas y en las malas , sin ella seguramente no estuviera aquí ahora.

RESUMEN

La presente investigación propone el diseño y la implementación de un sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI (Universidad de Ciencias Informáticas), que garantice la eficiencia y eficacia de este proceso, debido a que no se encuentra ningún mecanismo establecido que posibilite que se desarrolle de una manera óptima. Para el cumplimiento de este objetivo se realiza un estudio preliminar sobre las soluciones existentes relacionadas con los procesos de optimización de rutas y algoritmos de optimización para el transporte. Finalmente se verifica y valida el resultado generado mediante pruebas funcionales y no funcionales, con el fin de determinar el cumplimiento de las expectativas de los objetivos sobre la solución informática propuesta, obteniéndose como resultado un sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI.

Palabras clave: sistema, algoritmo, ruta, optimización

ABSTRACT

This research proposes the design and implementation of a route management system using optimization algorithms for the transportation of solid waste at the UCI (University of Informatics Sciences), which guarantees the efficiency and effectiveness of this process, since there is no established mechanism that allows it to be developed in an optimal way. For the fulfillment of this objective, a preliminary study is carried out on the existing solutions related to the processes of route optimization and optimization algorithms for transportation. Finally, the result generated is verified and validated through functional and non-functional tests, in order to determine the fulfillment of the expectations of the objectives of the proposed computer solution, obtaining as a result a route management system using optimization algorithms for the transportation of solid waste in the UCI.

Keywords: system, algorithm, routing, optimization

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las Tecnologías de la Información y Comunicación (TIC) se han convertido en uno de los agentes más eficaces en relación al favorecimiento de cambios y avances en la sociedad actual. Su papel como medio de comunicación y de socialización, así como sus funciones en busca de mejorar procesos en campos de la economía, la salud o el ocio, han convertido a las TIC en un elemento fundamental de cambio, incluso en aspectos cotidianos de nuestro día a día. La introducción de las TIC está transformando nuestra sociedad en todos los ámbitos, también nuestra cultura científica, base sobre la cual se instaura el desarrollo de la sociedad moderna.[CITATION Bra183 \l 1033]

Según la Asociación Americana de las Tecnologías de la Información (*Information Technology Association of America,* ITAA), las TIC son: “el estudio, el diseño, el desarrollo, el fomento, el mantenimiento y la administración de la información por medio de sistemas informáticos”. Esto incluye no solamente los teléfonos celulares, uno de los medios más usados, sino también la computadora, la televisión, la radio, los periódicos digitales, etcétera.[CITATION Cor13 \l 1033]

Las TIC tienen una gran influencia en muchos sectores de la sociedad, uno de ellos donde tiene grandes repercusiones es en los sistemas realizados para la informatización de los medios de transporte, los cuales nos permiten la circulación de bienes y personas, logrando una integración social que favorece el desarrollo[CITATION daS12 \l 1033] .Por este motivo, con el paso del tiempo se ha visto una mejora en la eficiencia de estos, con servicios renovados y una utilización de recursos menor. En la actualidad existe un verdadero interés en lograr que los medios de transporte utilicen menor cantidad de energía, circunstancia en parte relacionada con los problemas que pueden existir en el futuro en lo que respecta a la provisión de combustible.

La informatización de los sistemas dedicados a la gestión de rutas de la transportación, ha sido una estrategia adoptada por gobiernos y administraciones para hacer un uso racional del combustible y de todos los recursos relacionados en este complejo tema de movilizar personal o recursos materiales.[CITATION Jor212 \l 1033]

En la actualidad, la planificación de rutas es uno de los principales problemas en la optimización de las operaciones logísticas de transporte, cuyo objetivo principal es reducir los costos

de esta actividad[CITATION Mos15 \l 1033] .El diseño y optimización de rutas es tremendamente ventajoso en cualquier ámbito y situación para cualquier tipo de usuario, en especial para las empresas de transporte cuyas pérdidas y ganancias se basan en la distribución óptima tanto del tiempo, como del combustible, que están directamente relacionadas con la distancia recorrida.

Un factor fundamental para la optimización de rutas son los algoritmos que durante las últimas décadas ha habido un creciente interés en algoritmos basados en el principio de la evolución. Entre los algoritmos más conocidos se incluyen los algoritmos genéticos, programación evolutiva, estrategias evolutivas y programación genética. El conjunto de estas técnicas se agrupan bajo el nombre de computación evolutiva. Los algoritmos evolutivos son métodos robustos de búsqueda, que permiten tratar problemas de optimización donde el objetivo es encontrar un conjunto de parámetros que minimizan o maximizan una función de adaptación. [CITATION And131 \l 1033]

En los últimos años en las grandes ciudades y en aquellas en proceso de crecimiento, el incremento de la producción de residuos sólidos urbanos (RSU) su recolección, transporte, tratamiento y disposición final, constituye un problema difícil de gestionar. Los residuos, también conocidos como basura, son aquellos productos no intencionados derivados de las actividades individuales y colectivas de la población, en la actualidad, por el incremento de las actividades productivas, comerciales, turísticas y de servicios se diversifican enormemente; aumentan en producción y en muchos casos las municipalidades no logran recolectarlos en su totalidad.[CITATION Msc171 \l 1033]

El crecimiento acelerado de la población en los últimos años, así como el proceso de industrialización han aumentado la generación de residuos sólidos, haciendo que la logística de recolección sea más compleja, aunque este problema siempre ha existido en la actualidad se ha convertido en un aspecto crítico, debido a que la recolección y el transporte son las actividades más costosas del servicio de aseo urbano, en la mayoría de los casos representa entre el 60% y 70% del costo total de este servicio.[CITATION Rod18 \l 1033]

En Cuba, la gestión de la optimización del transporte de la recogida de residuos sólidos está teniendo un gran auge por ello se solicita fomentar la expansión de proyectos que agilicen y hagan más eficientes y eficaz los recorridos que realizan los mismos. Por tal razón la

Universidad de las Ciencias Informáticas (UCI) desea realizar la informatización del proceso de optimización de rutas del transporte de residuos sólidos utilizando el algoritmo dijkstra para que optimice las rutas disponibles, buscando así una solución práctica a este problema y gestionar de forma adecuada todos los eslabones de la cadena, ofreciéndole a la población universitaria un mejor servicio, mejorar el entorno, cuidar el medio ambiente y disminuir los diferentes costos ocasionados en el desarrollo de esta actividad.

Es precisamente aquí donde se evidencia la necesidad de continuar este proyecto para lograr optimizar el servicio de recolección de RSU en la Universidad de las Ciencias Informáticas (UCI), lo cual implica desarrollar una investigación al respecto, constituyendo como el **problema central a resolver**: ¿Cómo gestionar la ruta utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI?

Para dar solución al problema se concretó como **objeto de estudio** los sistemas de gestión de rutas mediante algoritmos de optimización

Con la finalidad de enmarcar la investigación se define el siguiente **campo de acción**: los procesos de gestión de rutas utilizando algoritmos de optimización para el transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI).

Se plantea como **objetivo general** desarrollar un sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI (Universidad de Ciencias Informáticas)

Objetivos específicos:

1. Analizar los principios teóricos de la investigación enfocados a los algoritmos de optimización de rutas.
2. Diseñar la propuesta del sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI).
3. Validar la solución desarrollada a partir de las pruebas de software.

A partir del problema general se derivan las siguientes **Tareas de investigación**

1. Estudio del estado de las soluciones de software destinados a los sistemas de gestión de algoritmo de optimización de rutas
2. Estudio de los procesos de la gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos de la Universidad de las Ciencias Informáticas (UCI).
3. Elaboración de los fundamentos teóricos producto de la revisión bibliográfica e investigación del estado del arte.
4. Caracterización del proceso de desarrollo, las herramientas y tecnologías a utilizar en el desarrollo del sistema.
5. Modelación de los procesos de negocio.
6. Diseño de la solución informática.
7. Descripción de la arquitectura
8. Definición de los patrones de diseños más adecuados para la construcción la propuesta de solución.
9. Presentación del estándar de codificación y explicación de las técnicas de programación a utilizar.
10. Implementación del sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI).
11. Validación de los resultados obtenidos.

Fueron utilizados varios métodos de la investigación científica los cuales según Hernández León [CITATION Med14 \l 1033] , tienen el objetivo de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones.

De los **Métodos Teóricos** se emplean los siguientes

1. Analítico–Sintético. Se empleó para determinar las generalidades y especificidades en el objeto de estudio y el campo de acción.

2. Histórico–Lógico. Se utilizó para determinar antecedentes, tendencias y regularidades del objeto de estudio y el campo de acción. Específicamente para estudiar la forma en que han evolucionado las tecnologías para el desarrollo de sistemas informáticos relacionados con el tema tratado en la investigación.

3. Modelación. Se utilizó con el objetivo de obtener los productos de trabajo necesarios para estudiar y transformar el proceso de análisis de datos de la concepción del sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI).

De los **Métodos Empíricos** se emplean los siguientes:

Entrevista. En la investigación se aplicó esta técnica para obtener los requisitos funcionales del sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI).

Observación: Se utiliza para obtener una información más precisa de cómo se realiza proceso de gestión de rutas utilizando algoritmos de optimización para el transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI).

Encuesta: Se emplea para la recopilación de información sobre el proceso de gestión de rutas utilizando algoritmos de optimización para el transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI).

El presente trabajo está estructurado en tres capítulos, tal y como se describe a continuación:

Capitulo1:

Fundamentos y referentes teórico-metodológicos sobre el sistema de gestión de rutas mediante algoritmos de optimización

Fundamentación teórica sobre la gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos. Se realiza un estudio del estado del arte que permite conocer la situación mundial del tema. Además, se hace referencia a los principales conceptos referentes al mismo. Se describen los sistemas informáticos que existen para evaluar la satisfacción de los clientes. Por último, se caracterizan las metodologías de desarrollo, tecnologías y herramientas posibles a utilizar, seleccionando las que más se ajustan a las necesidades del sistema a implementar, con el fin de desarrollar un sistema gestión de rutas utilizando algoritmos de optimización para el transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI).

Capítulo 2: Diseño del sistema de gestión de rutas mediante algoritmos de optimización

Análisis, planificación y diseño del sistema de gestión de rutas utilizando algoritmos de optimización del transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI). En este capítulo se hace un análisis y descripción general de la propuesta de desarrollo del sistema, así como la realización del levantamiento de los requisitos funcionales y no funcionales, entrada de la implementación del software. También se especifican los patrones del diseño aplicados, así como los artefactos derivados de la metodología de desarrollo de software seleccionada para esta etapa del proceso de desarrollo del sistema propuesto. Se lleva a cabo una explicación de la organización del sistema expuesta en los diagramas de clase para el correcto funcionamiento de este.

Capítulo 3: Validación del sistema de gestión de rutas mediante algoritmos de optimización

Implementación y validación del sistema de gestión de rutas: En este capítulo se detalla la propuesta de solución al problema planteado. Se realiza el diagrama de despliegue y se es-

pecifican los estándares de codificación a utilizar. Se realizan las estrategias de pruebas definidas para el sistema y se muestran interfaces como parte del resultado final.

Por último, se presentan las conclusiones, las recomendaciones, la bibliografía y el conjunto de anexos para una mejor comprensión de lo expuesto en la investigación.

Capítulo I: Fundamentos y referentes teórico-metodológicos sobre el sistema de gestión de ruta mediante algoritmos de optimización

En el presente capítulo se realiza un análisis del estado del arte nacional e internacional relacionado al tema de estudio. Además, se enuncian y fundamentan los principales conceptos, tecnologías, herramientas y metodologías a utilizar para el diseño e implementación de la solución propuesta caracterizando la metodología de desarrollo de software, los métodos, el lenguaje de programación y las distintas herramientas seleccionadas para darle solución al problema de la investigación.

I.1 Conceptos generales asociados al dominio del problema

Según *Van Gigch,* un **sistema** se define como una unión de partes o componentes, conectados en una forma organizada. Las partes se afectan por estar en el sistema y se cambian si lo dejan. La unión de partes hace algo (muestra una conducta dinámica como opuesto a permanecer inerte). Además, un sistema puede existir realmente como un agregado natural de partes componentes encontradas en la naturaleza, o puede ser un agregado inventado por el hombre, una forma de ver el problema que resulta de una decisión deliberada de suponer que un conjunto de elementos está relacionado.[CITATION Gig871 \l 1033]

De acuerdo al cibernetista *Stafford Beer,* un **sistema** se define como un conjunto de ítems que están dinámicamente relacionados. Sumemos a esta definición una serie de elementos que le otorgan mayor precisión y riqueza:

- Se trata de un conjunto de elementos que son las partes u órganos del sistema.
- Dinámicamente relacionados en una red de comunicaciones resultante de la interacción de los elementos.
- Formando una actividad, que es la operación (o procesamiento) del sistema.
- Para alcanzar un objetivo o propósito.

- Operando sobre datos/energía/materia, que son los insumos o entradas de recursos para que el sistema pueda operar.
- En una referencia dada de tiempo, que constituye el ciclo de actividad del sistema.
- Para suministrar información/energía/materia, que son los resultados de la actividad del sistema.[CITATION Leo94 \l 1033]

Analizando las definiciones anteriores se puede afirmar que un sistema es un conjunto de elementos relacionadas entre sí ordenadamente para lograr un fin determinado. (elaboración propia)

Gestión

Para *Robbins y Coulter* la gestión o administración se refiere a la coordinación de actividades de trabajo, de modo que se realicen de manera eficiente y eficaz con otras personas y a través de ellas, lo cual se convierte en el objetivo principal de toda gestión.[CITATION Sya14 \l 1033]

Según *Taylor Wilson* la gestión es el arte de saber lo que se quiere hacer y a continuación, hacerlo de la mejor manera y por el camino más eficiente.[CITATION Sýk05 \l 1033] De acuerdo a estos conceptos se puede plantear que la gestión es la capacidad de una institución para definir, alcanzar y evaluar sus propósitos, administrando y dirigiendo de manera adecuada sus recursos disponibles. (elaboración propia)

Sistema de Gestión

Según los autores *Camisón, Cruz y González* un Sistema de Gestión es el conjunto de elementos mediante el cual la dirección planifica, ejecuta y controla todas sus actividades para el logro de los objetivos preestablecidos.[CITATION Cam06 \l 1033]

Es decir, un sistema de gestión es un conjunto de elementos mutuamente relacionados o que interactúan, para establecer la política y los objetivos, y para lograr dichos objetivos. Un sistema de gestión integrada, posibilita y simplifica la implantación de un único sistema de gestión eficaz, adecuado para la empresa. (elaboración propia)

Gestor de rutas

Un gestor de rutas es una aplicación informática que, a partir de un conjunto de puntos, entre ellos un punto de destino final prefijado, selecciona un punto inicial y una ruta que

recorra los restantes puntos intermedios de manera óptima, es decir, minimizando el tiempo y la distancia de dicha ruta.[CITATION And132 \l 1033]

Según Vidal [CITATION Vid13 \l 1033] la **GIRSU** es la disciplina asociada al adecuado manejo de los residuos y constituye la cadena del ciclo de los residuos, que incluye la reducción en la fuente, reúso, reciclaje, barrido, almacenamiento, recolección, transferencia, tratamiento y disposición final, unido al cambio de actitud y comportamiento de todos quienes integran el proceso productivo y de consumo.

La Dirección General de Saneamiento Ambiental *(DIGESA),* define los residuos como un producto no intencionado derivado de las actividades individuales y colectivas cuya peligrosidad se evidencia para la sociedad cuando su manejo compromete la salud, el ambiente y el bienestar de la persona.[CITATION Mac17 \l 1033]

1.1.2 Algoritmos de optimización

Durante las últimas décadas han existido una serie de algoritmos dirigidos a la optimización, algunos de ellos son: los algoritmos evolutivos, entre los más conocidos se incluyen los algoritmos genéticos[CITATION Dav91 \l 1033], programación evolutiva[CITATION Fog95 \l 1033].El conjunto de estas técnicas se agrupa bajo el nombre de computación evolutiva. Los algoritmos evolutivos son métodos robustos de búsqueda, que permiten tratar problemas de optimización donde el objetivo es encontrar un conjunto de parámetros que minimizan o maximizan una función de adaptación.
Estos algoritmos operan con una población de individuos Pt x x t n t ( ) { ,..., } =1 , para la iteración t, donde cada individuo xi representa un punto de búsqueda en el espacio de las soluciones potenciales a un problema dado. El desempeño de un individuo xi se evalúa según una función de adaptación f(xi). Esta función permite ordenar del mejor al peor los individuos de la población en un continuo de grados de adaptación.
La población inicial evoluciona sucesivamente hacia mejores regiones del espacio de búsqueda mediante procesos probabilísticos de:
a) selección de los individuos más adaptados en la población (a mayor grado de adaptación mayor probabilidad de dejar descendencia)
b) modificación por recombinación y/o mutación de los individuos seleccionados.
Para encontrar los óptimos globales, los algoritmos de optimización hacen uso de dos técnicas:
a) explorar áreas desconocidas en el espacio de búsqueda,

b) explotar el conocimiento obtenido de puntos previamente evaluados. Los algoritmos evolutivos combinan ambas técnicas de un modo eficiente.

Otros Algoritmos de optimización

Algoritmo de árbol de expansión mínima

Un árbol de expansión es un mínimo conjunto de enlaces de E que conectan todos los nodos en V y por lo tanto al menos un árbol de expansión puede ser encontrado en un grafo G. El mínimo árbol de expansión denotado por T* es un árbol de expansión cuyo peso total de todos los enlaces es mínimo.[CITATION May14 \l 1033]

Pasos del algoritmo del árbol de expansión mínima

Los pasos del procedimiento se dan como sigue: Digamos que N = {1,2, ..., n} es el conjunto de la red y defina:

C_k = Conjunto de nodos que se han conectado permanentemente en la iteración k del algoritmo.

$\overline{C_k}$ = Conjunto de nodos que todavía se deben conectar k permanentemente.

Paso 0. Establecer$C$o=$\phi y C$o =N.

Paso 1. Empiece con cualquier nodo i, en el conjunto no conectado Co y establezca C1 = {i}, lo que nos da automáticamente$C$ =N−{i}.

Se establece k = 2.

Paso general k. seleccione un nodo j ∗, en conjunto no conectado C k−1 que produce la rama más corta hacia un nodo en el conjunto conectado Ck−1. Una permanente j* con Ck−1 y elimínelo de Ck−1, es decir: C_k = C_{k-1} + {j *}, C_k=C_{k-1}−{j*}

Algoritmo de la ruta más corta

- Algoritmo de Dijkstra.
- Algoritmo de Floyd.

Algoritmo de Dijkstra. También llamado **algoritmo de caminos mínimos**, es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de los vértices en un grafo con pesos en cada arista. Explorando todos los caminos más cortos que parten del vértice, cuando se obtiene el camino más corto desde el vértice de origen, al resto de vértice que componen el grafo, el algoritmo se detiene.[CITATION Tor12 \l 1033]

Los cálculos del algoritmo avanzan de un nodo i a un nodo inmediatamente siguiente j, utilizando un procedimiento especial de clasificación.

Digamos que u_i es la distancia más corta del nodo 1 del punto de origen al nodo i y defina $d_{ij}(\geq 0)$ como longitud del arco (i, j) entonces la clasificación para el nodo j se define como: $(u,i)=[u+d,i],d \geq 0$.

Pasos del algoritmo:

- **Paso 0**

Clasificar el punto de origen (nodo 1) con la clasificación permanente [0, −] quedetermine i = 1.

- **Paso1**

(a) Calcule las clasificaciones temporales $[u_i + d_{ij} , i]$ para cada nodo j al que se puede llegar desde el nodo i, siempre y cuando j no esté clasificado permanentemente. Si el nodo j ya está clasificado con $[u , k]$ a través de otro j nodo y si $u+d<u$,se reemplaza$[u,k]$ con: $[u_i+d_{ij},i]$.

(b) Si todos los nodos tienen clasificaciones permanentes, existe un error. De lo contrario, seleccione la clasificación $[u_r, s]$ con la distancia más corta $(= u_r)$ entre todas las clasificaciones temporales (rompa los empates arbitrariamente). Sea $i = r$ y repita el paso i. [CITATION JAt12 \l 1033]

Algoritmo de FLOYD. - En el 2014, **TAHA** manifiesta que “El algoritmo de Floyd es más general que el de Dijkstra, porque determina la ruta más corta entre cualesquiera dos nodos en la red.” [CITATION Tah14 \l 1033]

El algoritmo representa una red de n nodos como una matriz cuadrada con n renglones y n columnas. La entrada (i, j) de la matriz de la distancia d_{ij} del nodo i al nodo j, que es finito si i esta eslabonado directamente a j; de los contrarios es infinito.

La idea del algoritmo de Floyd es directa. Dados tres nodos i, j, k donde las distancias de conexión se muestran en los tres arcos, es más corto llegar a k desde i a través de j si: $d_{ij} + d_{jk} < d_{ik}$.

En este caso, es óptimos reemplazar la ruta directa de $i \rightarrow k$ con la ruta indirecta $i \rightarrow j \rightarrow k$.

Tabla 1 Cuadro comparativo entre Algoritmos propios de la teoría de redes.

Fuente: [Requejo.J.(2018)]

RUTA MÁS CORTA	ÁRBOL DE EXPANSIÓN MÍNIMA	FLUJO MÁXIMO	FLUJO DE COSTO MÍNIMO
Se considera una red no dirigida y conexa	Se considera una red no dirigida y conexa.	Se considera una red conexa.	Se considera una red conexa.
Se incluye una medida de longitud positiva: (distancia, tiempo, costo.)	Se incluye una medida de longitud positiva: (distancia, tiempo, costo.).	Se incluye una medida de longitud positiva: (distancia, tiempo, costo, etc.).	Se incluye una medida de longitud positiva: (distancia, tiempo, costo, etc.).
Cada medida está asociada a una unión (ligadura).	Cada medida está asociada a una unión (ligadura).	Cada medida está asociada a una unión.	Cada medida está asociada a una unión.
Se seleccionan un conjunto de ligaduras con la longitud total más corta entre todos los conjuntos de trayectorias.	Se seleccionan un conjunto de ligaduras con la longitud total más corta entre todos los conjuntos de trayectorias.	La selección de la longitud total, estará establecida cuando se maximice la cantidad total de flujo del nodo de inicio al	La selección de la longitud total, estará establecida cuando se minimice el costo total de enviar el suministro a los nodos intermedios y llegar al nodo final.

			nodo final.	
	Para el problema de la ruta más corta la propiedad requerida es que la ligadura seleccionada debe proporcionar una trayectoria entre el origen y el destino.	Para el problema del árbol de expansión mínima la propiedad requerida es que las ligaduras seleccionadas deben proporcionar una trayectoria entre cada par de nodos.	Se consideran redes conexas, pero estas son dirigidas. Cada red dirigida tiene un número de transiciones determinadas.	-Se consideran redes conexas, pero estas son dirigidas. Cada red dirigida tiene un número de transiciones determinadas.

Luego de analizar el cuadro comparativo podemos observar que todos los problemas de transportes están orientados al mismo fin: concluir toda una trayectoria y basándose por ciertos parámetros. Como nuestro parámetro inicial para la mejora de nuestro problema es la mejora de tiempos de recogida de los residuos sólidos se decidió analizar más a fondo los algoritmos: ruta más corta y árbol de expansión mínima por adecuarse más con la mejora de la problemática planteada.

Al analizar ambos algoritmos se decidió realizar el sistema mediante el algoritmo Dijkstra porque mediante el mismo se puede calcular realmente la distancia mínima entre cualquier nodo, es decir con la misma complejidad podemos hallar la mínima distancia de un nodo a todos los demás.

I.2 Sistemas homólogos

En el proceso de búsqueda de sistemas informáticos existentes con propósitos similares a los deseados se definieron los siguientes criterios a evaluar: sistemas de gestión, búsque-

das personalizadas, gestión del transporte de residuos sólidos, gestión de instituciones, generación de reportes y multiplataforma. Estos criterios permitieron identificar sistemas Informáticas con características similares a las necesidades del sistema de gestión de optimización de rutas para el transporte de residuos sólidos, permitiendo conocer el objetivo y las funcionalidades que poseen los mismos.

En la actualidad existe una amplia gama de sistemas que se encargan de la Gestión de optimización de rutas, estos han sido desarrollados tanto por entidades internacionales como por empresas de la red nacional.

1.2.1 Sistemas internacionales

Existen diferentes autores que han enfrentado problemas relacionados en la planificación de rutas en escenarios diferentes, como se menciona en (Zhang, Chen & Li, 2015) [CITATION Zha15 \l 1033] que para empresas grandes, donde sus vehículos recorren grandes distancias, la optimización de rutas es importante en cuestión de logística, es por ello que implementaron el algoritmo de dijkstra para resolver el problema de la ruta más corta, utilizaron el método del mapa etiquetado para visualizar el estado de cada iteración (Xin, Yan, & Taoying, 2015)[CITATION Xin15 \l 1033]. En el proyecto de investigación (Bernal, Juan Manuel Anton, et al.), utilizaron los algoritmos Dijkstra, para planificar las rutas de la recogida de los residuos sólidos.[CITATION Ber21 \l 1033]

Tabla 2 Casos de proyectos donde se utiliza la optimización
Fuente: [Elaboración propia]

Referencias	Metodologías utilizadas	Resultados	Lugar de la implementación
1-(Amal et al., 2019)	Sistemas de información Geográfica Arc- GIS_Network analyst y modelos para optimización multicriterio.	Optimización del consumo de combustible, las emisiones de gases, la confiabilidad del servicio de recolección, la longitud del camino y la compatibilidad con la	Sfax, Tunez

		política energética nacional	
2-(Cavdar et al., 2016)	Sistema inteligente de recogida de residuos sólidos(SSWCS)con sistemas de identificación por radio frecuencia(RFID)	Se demostró que con la implementación se reducirían entre 50 a 80% de los costos en servicios de sanidad	Turquía
3-(Ferronato et al., 2020)	SIG: QGIS3.8 y clasificación de residuos reciclables en la fuente generadora	Reducción de gastos operativos en un 10%, reducción de distancias recorridas por los camiones en un 7%, generación de fuentes de empleo	La paz, Bolivia
4-(Lozano Álvaro, Villarrubia Gabriel, López Alberto, Hernández	Internet de las cosas (IOT)	Evaluaron una plataforma tecnológica en un ambiente real, logrando optimizar la recolección de RSM, mejoró el consumo de combustible de la flota vehicular e incrementó su vida útil, además se generó información que puede ayudar a determinar las necesidades de la ciudad.	Málaga, España

5-(Rada et al., 2013)	Sistemas Web GIS	Optimizaron el sistema de recolección de RSU en Italia en un 80% y comprueban la viabilidad de su aplicación en otros entornos con casos resueltos en China y Malasia	Trento, Italia
6-(Rızvanoğlu et al., 2020)	Sistemas de información Geográfica y optimización mediante programación lineal	Optimización de hasta un 33% en los costos de transporte, y cronogramas óptimos de rutas	Haliliye, Turquía
7-(Titrik, 2016)	Sistema de información y comunicación basado en tiempo real y tecnología GIS	Reducción del vaciado innecesario de contenedores, vaciado de los contenedores a tiempo, reducción del impacto ambiental y mejora de la seguridad vial, mejoras en la ubicación de estaciones de recogida selectiva de residuos.	Győr, Hungría

Adicionalmente se debe mencionar al trabajo realizado por Rossit et al, en cuanto a los beneficios visuales que brinda la aplicación de un SIG en la resolución de problemas de optimización de rutas vehiculares y que son sin duda un factor clave para determinar su aceptación e implementación definitiva.[CITATION ros19 \l 1033]

Sin embargo, en la implementación de estos sistemas existen algunos factores que afecta la viabilidad del sistema como son los **altos costos de la inversión** inicial para adquirirlos y muchas veces es necesario implementar, también **complejas redes** de **transmisión de**

datos para el posicionamiento de los vehículos recolectores en tiempo real y no muestran la ruta más óptima en su totalidad.

1.2.2 Sistemas nacionales

El sistema actual de gestión de los Residuos Sólidos Urbanos en la ciudad de Santa Clara (sistema 8) está caracterizado por etapas simples que comprenden la generación y recolección. Para el servicio de recolección, Comunales que es la empresa responsable de esta actividad dividió la ciudad en diez zonas. Tomando como criterio el número de habitantes y la cantidad de RSU generados. En cada zona la recolección tiene una frecuencia de cuatro días a la semana, exceptuando las zonas centrales, comercial, edificios de apartamentos e instituciones hospitalarias, donde el servicio se realiza diariamente en jornada diurna y nocturna.[CITATION Cár19 \l 1033]

El servicio de recogida y transportación en las condiciones actuales cubre el 72,51 % de la demanda, prestándole este servicio al 92.8% de la población. El almacenamiento de los residuos sólidos se realiza en condiciones inadecuadas sin segregación en el origen y en lugares alternativos presentes en edificios multifamiliares, comercios instituciones educacionales y de salud. Los métodos de recolección empleado actualmente son: el método de Esquina o de Parada Fija, el método de Acera. Y la existencia de sitios de disposición ilegales producto de la indisciplina social generados por las fallas del sistema de recogida

Principales deficiencias en la actividad de recogida

No existe uniformidad en los envases de recolección domiciliaria, mal almacenamiento primario de la basura la cual se encuentra toda mezclada, inadecuada estrategia de recogida, inestabilidad en la frecuencia de recogida y por último las rutas de recolección no tienen un procedimiento logístico.

Principales deficiencias en el servicio de transportación

1. No uniformidad en los carros donde se realiza la recogida domiciliaria.
2. Deficiente aprovechamiento de la capacidad de recogida de los carros.
3. Falta de control de los viajes al vertedero.
4. El recorrido de la recogida se realiza sin un sistema logístico de transportación.

5. Mala estrategia de decisión con respecto a las zonas afectadas.

6. Disposición final

7. La disposición final de los residuos en la ciudad de Santa Clara se realiza en 4 vertederos. Uno convencional mecanizado y cuatro a cielo abierto, con tratamiento parcial, por la falta de equipamiento.

Tabla 3 Análisis de los indicadores en los sistemas estudiados.
Fuente: [Elaboración propia]

Sistemas	Bajos Costos de inversión	Sencillas redes de transmisión de datos	Óptimo
1	×	×	✓
2	✓	×	×
3	✓	×	×
4	×	×	✓
5	✓	×	✓
6	✓	×	×
7	×	×	✓
8	✓	×	✓

Después de analizar los sistemas anteriores se pudo constatar que ninguno de los sistemas anteriores es satisfactorio en su totalidad, por lo que se realizó un estudio con mayor profundidad para determinar la ruta del transporte más óptima en su totalidad. No se tiene en cuenta ninguno de los sistemas antes investigados para la solución de nuestro problema, debido a los resultados obtenidos antes expuesto en la figura 3.

I.3 Tecnologías, herramientas y metodologías del sistema

La solución de software, se desarrolla utilizando las tecnologías y herramientas definidas por la Dirección de Informatización de la UCI, más específicamente las utilizadas para el desarrollo del sistema de gestión de ruta del transporte, a la que se desea integrar. Esta aplicación establece utilizar como lenguaje de programación Dart, Flutter y Hive como marco de trabajo y se hace uso de la metodología AUP_UCI en su escenario 4.

1.3.1Tecnologías

Dart es un lenguaje open source desarrollado en Google con el objetivo de permitir a los desarrolladores utilizar un lenguaje orientado a objetos y con análisis estático de tipo. Desde la primera versión estable en 2011, Dart ha cambiado bastante, tanto en el lenguaje en sí como en sus objetivos principales. Con la versión 2.0, el sistema de tipo de Dart pasó de opcional a estático, y desde su llegada, Flutter se ha convertido en el principal objetivo del lenguaje.[CITATION MEN18 \l 1033]

A diferencia de muchos lenguajes, Dart se diseñó con el objetivo de hacer el proceso de desarrollo lo más cómodo y rápido posible para los desarrolladores. Por eso, viene con un conjunto bastante extenso de herramientas integrado, como su propio gestor de paquetes, varios compiladores/transpiladores, un analizador y formateador. Además, la máquina virtual de Dart y la compilación Just-in-Time hacen que los cambios realizados en el código se pueden ejecutar inmediatamente. Una vez en producción, el código se puede compilar en lenguaje nativo, por lo que no es necesario un entorno especial para ejecutar. En caso de que se haga desarrollo web, Dart se transpila a JavaScript.

En cuanto a la sintaxis, la de Dart es muy similar a lenguajes como JavaScript, Java y C ++, por lo que aprender Dart sabiendo uno de estos lenguajes es cuestión de horas.

Dart es un lenguaje de propósito general, y lo puedes utilizar casi para cualquier cosa:

- En aplicaciones web, utilizando la librería de arte: html y el transpilador para transformar el código en Dart en JavaScript, o utilizando frameworks como AngularDart.
- En servidores, utilizando las librerías de arte: http y arte: io. También hay varios frameworks que se pueden utilizar, como por ejemplo Aqueduct.
- En aplicaciones de consola.

- En aplicaciones móviles gracias a Flutter.

1.3.2 Herramientas

Marco del trabajo

- Framework utilizado: Flutter.

Flutter es un *framework* de Dart para crear aplicaciones multiplataforma con un único código. A diferencia de otros frameworks multiplataforma como por ejemplo Ionic, el código de una aplicación de Flutter se compila a código nativo, por lo que el rendimiento alcanzado es superior a aplicaciones basadas en *web-view*s. Además, a diferencia de React Native, flutter no utiliza componentes nativos, sino que viene con sus propios componentes, llamados *widgets*, por lo que la misma aplicación se verá igual en cualquier dispositivo, independientemente de su sistema operativo o la versión. Gracias a ello, el desarrollador no tiene que preocuparse por que el diseño de su aplicación se vea mal en dispositivos antiguos.[CITATION BOU19 \l 1033]

Además de aplicaciones móviles, con Flutter también se pueden hacer páginas web y aplicaciones de escritorio, aunque el soporte para páginas web está en beta, y por aplicaciones de escritorio en technical preview, por lo que quien los quiera usar habrá que esperar un tiempo más a que sea estable.
A continuación, tiene varios ejemplos de aplicaciones desarrolladas con Flutter:

- Google Ads
- AppTree
- Reflectly
- inLapp Coffee
- SSHButtons

Lista con varios recursos que pueden ser útiles :

- Página web de Dart, donde podrá encontrar toda la documentación del lenguaje y varios tutoriales. En particular, el Language Tour es muy útil para ver una breve introducción a la sintaxis del lenguaje.

- Dartpad, que le permite experimentar con el lenguaje desde el navegador sin necesidad de instalar nada (también se puede escribir aplicaciones de Flutter).

- Página web del repositorio de paquetes, donde podrá encontrar todos los paquetes de la comunidad.

- Páginas de Medium de Dart y Flutter, con muchos artículos que pueden ser de ayuda.

- Página web de Flutter, donde se encuentra la documentación del framework, y cómo empezar a usarlo.

- Canal de YouTube de Flutter, con una gran cantidad de vídeos que pueden ser útiles. En particular, la serie de videos de Widget of the Week, una serie de breves vídeos de un par de minutos cada uno explicando un widget, bastante útil si estás empezando y no tienes muy claro qué widgets existen y como los puedes usar, y la serie Flutter in Focus, con videos más extensos explicando conceptos clave tanto de Dart como de Flutter.

➢ Programación del lado del servidor: ***Hive Flutter***

Hive es una base de datos clave-valor ligera y ultra rápida escrita en puro Dart
Características:

- Multiplataforma: móvil, escritorio, navegador
- Gran rendimiento
- API simple, potente e intuitiva
- Cifrado fuerte incorporado
- Sin dependencias nativas
- Pilas incluidas

➢ Entorno de desarrollo: Visual Studio Code (VS CODE).

Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity).[CITATION RAS21 \l 1033]

- Visual Paradigm for UML v8.0

Visual Paradigma for UML (UML, *Unified Modeling Language* por sus siglas en inglés, Lenguaje de Modelamiento Unificado) es una herramienta CASE (Ingeniería de Software Asistida por Computadora) profesional que soporta el ciclo de vida completo del desarrollo de software, análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases y documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas y proyectos.[CITATION Par13 \l 1033]

Dentro de sus principales características se aprecian:

- Soporte para Modelo y Notación de Procesos de Negocio (BPMN - Business Process Model and Notation, por sus siglas en inglés) y UML.
- Diagramas de Procesos de Negocio.
- Modelado colaborativo con CVS (Control de versión) y Subversión.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas.
- Reorganización de las figuras y conectores de los diagramas UML.
- Dibujo de diagramas UML con plantillas (stencils) de MS Visio.

Metodología de desarrollo de software

Metodología

Una metodología de software es un marco de trabajo que lo integran varios procedimientos agrupados con el objetivo de organizar, controlar y planear el proceso de desarrollo de un sistema. La misma está compuesta por técnicas, herramientas y artefactos que crean toda la base documental necesaria en el desarrollo de un software.[CITATION Jua13 \l 1033]

Metodología AUP

Creada por Scott Ambler, la metodología AUP es considerada una versión simplificada del Proceso Unificado Racional (RUP). AUP describe, de una manera simple y fácil de entender, la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Entre las técnicas ágiles que incluye se encuentra el Desarrollo Dirigido por Pruebas, el Modelado ágil, la Gestión de Cambios y la Refactorización de Base de Datos para mejorar la productividad. La estructura de esta metodología se organiza en cuatro fases y define 7 disciplinas, de ellas cuatro a los procesos ingenieriles y tres dedicados a la Gestión de Proyectos.

Metodología AUP-UCI

Proceso ágil unificado, por sus siglas en inglés *Agil Unified Process*, en su versión UCI. Esta metodología propone cuatro fases (inicio, elaboración, construcción, transición) para el ciclo de vida de los proyectos de la UCI mantiene la fase de inicio, pero modificando el objetivo de la misma, se unifican las restantes tres fases de AUP en una sola, a la que se llamó ejecución y se agrega la fase de cierre.[CITATION SAL16 \l 1033]

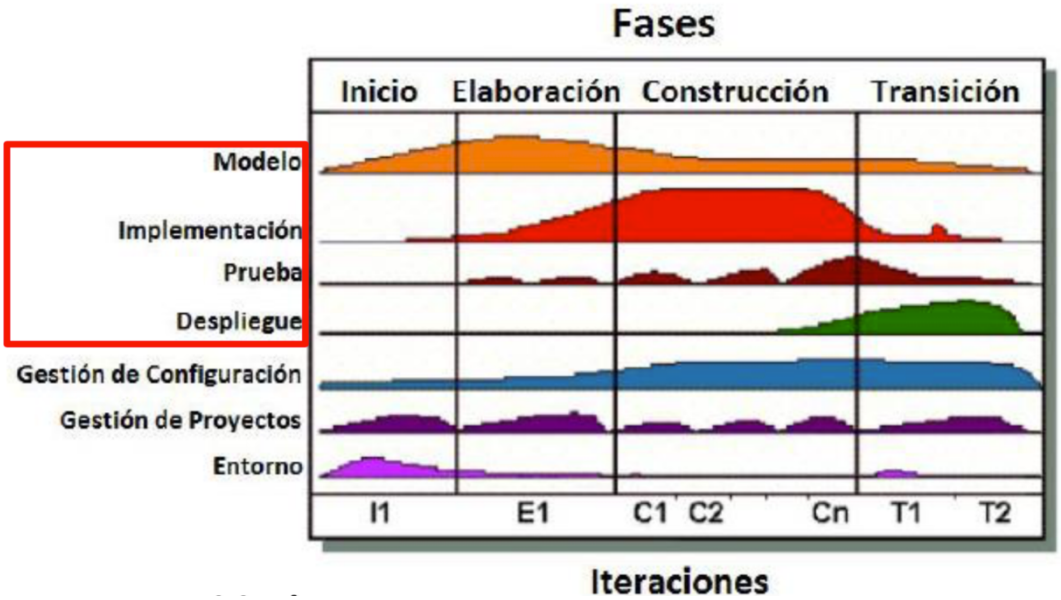


Figura 1 Fases de la metodología AUP

Fuente: [Scott W. Amber]

AUP propone siete disciplinas (modelado, implementación, prueba, despliegue, gestión de configuración, gestión de proyecto y entorno), se decide para el ciclo de vida de los proyectos de la UCI tener ocho disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: modelado de negocio, requisitos y análisis y diseño en AUP están unidos en la disciplina modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina: Implementación, en el caso de prueba se desagrega en tres disciplinas: pruebas internas, de liberación y aceptación y la disciplina despliegue se considera opcional. Las restantes tres disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV (Capacidad de Integración de Madurez del Modelo, por su significado en español) v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto).[CITATION Sán151 \l 1033]

AUP propone nueve roles (administrador de proyecto, ingeniero de procesos, desarrollador, administrador de BD (Base Datos), modelador ágil, administrador de la configuración, Stakeholder (experto en dominio), administrador de pruebas, probador). Se decide para el ciclo de vida de los proyectos de la UCI tener once roles, manteniendo algunos de los propuestos por AUP y unificando o agregando otros.

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Con la adaptación de AUP que se propone para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de software.[CITATION Sán15 \l 1033]

Por tanto, la metodología utilizada como guía para el desarrollo de la presente investigación fue la AUP-UCI, pues es por la que se rigen los procesos productivos en el centro de soporte de tecnologías.

Conclusiones del capítulo

A partir del análisis teórico de la investigación fue posible comprender el objetivo planteado, teniendo como partida el campo de acción donde se aplica. El estudio de los conceptos asociados permitió comprender términos usualmente tratados, facilitó la búsqueda bibliográfica y aportó vocablos concretos que acercaran al autor a la temática.

Como parte del análisis de las tecnologías de desarrollo fueron usadas las definidas por el cliente, las cuales los autores pudieron corroborar su utilidad, las facilidades que aportan al desarrollo, además de ser las más usadas en el mercado. Finalmente, para guiar el desarrollo de la presente investigación se utilizó la metodología AUP en su variación UCI por su adecuación a los procesos productivos de la universidad. Todo esto permitió un desarrollo sólido para obtener un sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI (Universidad de Ciencias Informáticas)

Capítulo II: Diseño de la solución propuesta al sistema de gestión de ruta mediante algoritmos de optimización.

En este capítulo se hace un análisis y descripción general de la propuesta de desarrollo del sistema, así como la realización del levantamiento de los requisitos funcionales y no funcionales, entrada de la implementación del software. También se especifican los patrones del diseño aplicados, así como los artefactos derivados de la metodología de desarrollo de software seleccionada para esta etapa del proceso de desarrollo del sistema propuesto. Se lleva a cabo una explicación de la organización del sistema expuesta en los diagramas de clase para el correcto funcionamiento de este.

II.1 Propuesta de solución

Se propone como solución a la problemática planteada, un sistema para la gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos. Dicho sistema brinda la funcionalidad de gestionar los choferes. Una vez incluido los choferes ofrece la posibilidad de visualizar mediante otra página la tarea que le corresponde a cada chofer según se registran en el sistema mediante su usuario y contraseña. Luego de registrarse y rellenar una serie de información y de observar las tareas asignada tendrán la opción de visualizar el mapa en el cual se mostrarán las rutas más cortas disponibles según la zona asignada a los choferes. Al terminar la jornada de los trabajadores, los mismos deben enviar un reporte al director con las actividades realizadas y las que quedaron pendiente.

El sistema contiene otro rol que es el del director que al registrarse en el sistema mediante su usuario y contraseña puede asignar las tareas a sus trabajadores y además puede observar un reporte por días de cada trabajador al terminar o no su jornada de trabajo asignada. El sistema aspira ser una herramienta útil para agilizar la búsqueda de rutas cortas para el transporte de residuos sólidos en la UCI.

II.2 Modelo de dominio

Para lograr una mayor comprensión del contexto en que se ubica la extensión y la posterior representación de la solución, se recurre al empleo del modelo de dominio. Es un subconjunto del modelo de negocio y se realiza cuando no están claros los procesos o no se

identifican claramente los actores y trabajadores del negocio. Además, el modelo de dominio captura los tipos más importantes de objetos que existen, se identifican conceptos, se definen y se relacionan en un diagrama de clases UML.[CITATION Mil11 \l 1033]

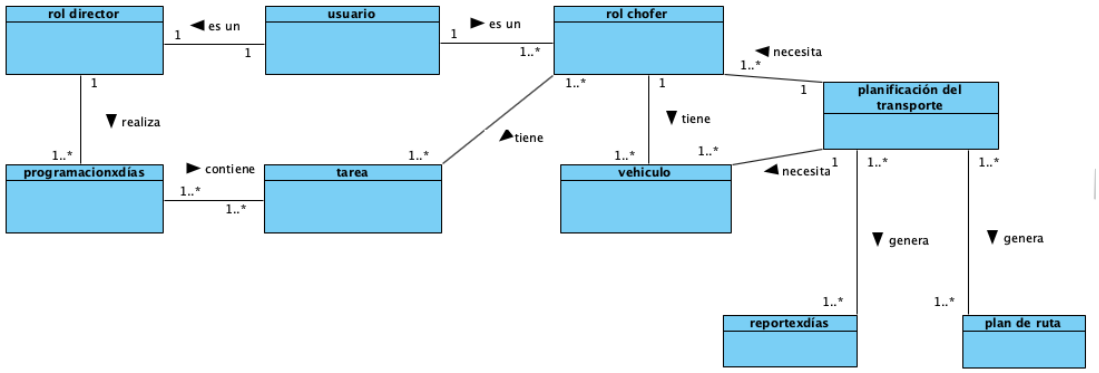


Figura 2 Diagrama de dominio del sistema de gestión para la optimización de rutas del transporte de los residuos sólidos

Fuente: [Elaboración propia]

II.2.1 1Descripción de los conceptos

En la siguiente tabla se describen los conceptos asociados al modelo conceptual:

Tabla 4 Conceptos y descripciones asociados al diagrama del dominio

Fuente: [Elaboración propia]

Conceptos	Descripción
Director	Persona que dirige o guía algo
Chofer	Persona cuyo oficio es conducir automóviles
Tarea	Labor o trabajo que realiza alguien

Vehículo	Aparato con o sin motor que se mueve sobre el suelo, en el suelo o en el aire y sirve para transportar cosas o personas.
Planificación de rutas	Herramienta capaz de organizar en forma automática las rutas diarias de la flota de vehículos.
Reporte	Informe o noticia que pretende transmitir una información
Planificación	Proceso sistemático en el que primeramente se establece una necesidad, y acto seguido, se desarrolla la mejor manera de enfrentarse a ella, dentro de un marco estratégico que permite identificar las prioridades y determina los principios funcionales.

II.2.3 3 Mapa de navegabilidad

Los sistemas de navegación son estructuras básicas de un producto multimedia los cuales permiten que el contenido sea interpretado y distribuido adecuadamente, cumpliendo los conceptos de navegabilidad e interactividad usuario/producto.

Un mapa de navegación es la representación gráfica de la organización de la información de una estructura web. Expresa todas las relaciones de jerarquía y secuencia y permite elaborar escenarios de comportamiento de los usuarios. [CITATION Lar10 \l 1033]

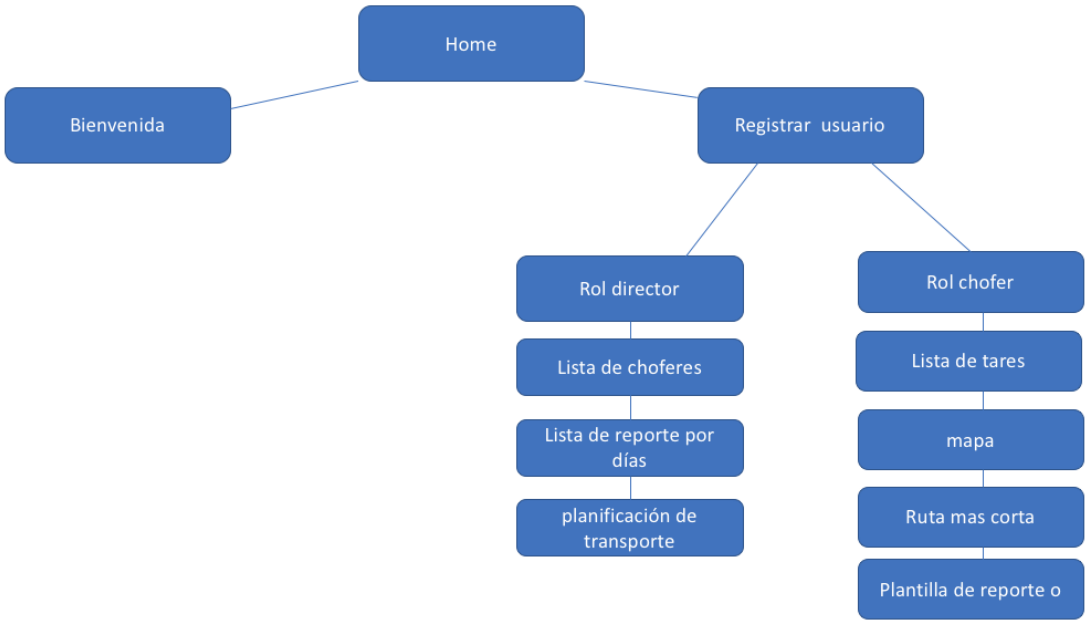


Figura 3 Mapa de navegabilidad

Fuente: [Elaboración propia]

II.3 Especificación de requisitos

Para lograr un entendimiento entre clientes y el equipo de proyecto a lo largo del proceso de desarrollo del sistema, se deben establecer los objetivos del producto a desarrollar. Con el fin de llegar a un consenso entre ambas partes, surge la especificación de requisitos[CITATION GUT19 \l 1033], donde se analizan las necesidades exactas de los usuarios. Luego son traducidas a precisas funciones y acciones que serán usadas en el desarrollo del sistema. Una correcta identificación de requisitos permite posteriormente la construcción de un software de calidad. Estos pueden clasificar por categorías en: funcionales y no funcionales, además permiten comprender desde un inicio, la línea a seguir en el desarrollo y así garantizar la calidad del sistema.

II.3.1 Requisitos funcionales

Un Requisito Funcional (RF) es una capacidad o condición que el sistema debe cumplir. [CITATION Lar15 \l 1033]. La técnica utilizada para la obtención de requisitos fue el modelo del dominio, el análisis de este permitió identificar requisitos. A continuación, se muestra el listado de requisitos identificados.

Tabla 5 Requisitos funcionales del sistema

Fuente: [Elaboración propia]

Requisitos funciona-les	Prioridad para el cliente	Complejidad
RF1: Autenticar usuario	Alta	Media
RF2: Registrar usuario	Media	Media
RF3: Gestionar usuario	Alta	Alta
RF4: Crear ruta	Alta	Alta
RF5: Modificar ruta	Alta	Alta
RF6: Eliminar ruta	Alta	Alta
RF7: Mostrar ruta	Alta	Alta
RF8: Adicionar vehículo	Media	Baja
RF9: Modificar vehículo	Media	Media
RF10: Mostrar vehículo	Media	Media
RF11: Listar vehículo	Baja	Baja
RF12: Eliminar vehículo	Media	Baja
RF13: Crear reporte	Alta	Media
RF14: Modificar reporte	Alta	Media

RF15: Mostrar reporte	Baja	Media
RF16: Eliminar reporte	Baja	Media
RF17: Listar reporte	Baja	Media
RF18: Asignar tarea	Alta	Alta
RF19: Modificar tarea	Alta	Alta
RF20: Mostrar tarea	Alta	Baja
RF21: Listar tarea	Media	Baja
RF22: Eliminar tarea	Media	Media
RF23: Registrar reporte por días	Baja	Media
RF24: Modificar reporte por día	Baja	Baja
RF25: Eliminar Reporte por días	Media	Media
RF26: Ver reporte por días	Alta	Media
RF27: Listar reporte por días	Media	Baja
RF28: Ver rutas disponi-	Alta	Alta

bles		
RF29: Eliminar rutas disponibles	Media	Alta
RF30: Listar rutas disponibles	Media	Medias

II.3.2 Requisitos no funcionales

Los Requisitos No Funcionales (RNF) son propiedades o cualidades que el producto debe tener. Son restricciones de los servicios o funciones ofrecidas por el sistema, y generalmente se aplican al sistema en su totalidad. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto. Están vinculados a requisitos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer se puede determinar como ha de comportarse, qué cualidades debe tener o cuan rápido debe ser [CITATION Mol19 \l 1033]. Los RNF tienen varias clasificaciones, la utilizada sigue lo que propone la plantilla de especificación de requisitos de la metodología AUP-UCI. Los requisitos identificados son:

Tabla 6 Requisitos no funcionales del sistema
Fuente: [Elaboración propia]

Usabilidad	El sistema puede ser utilizado por cualquier usuario que posea conocimientos informáticos básicos. El software tiene una curva de aprendizaje baja, que permite al usuario familiarizarse rápidamente con los elementos del sistema y operarlo de forma correcta en poco tiempo de uso. - El sistema será utilizado por todo el personal del centro que tenga acceso a la red del centro y necesite hacer uso del mismo
Fiabilidad	-El sistema brinda la posibilidad de establecer permisos sobre acciones, garantizando que solo acceda quien esté autorizado. -El sistema muestra las funcionalidades de acuerdo a quien esté autenticado en el mismo. -El sistema debe garantizar la protección ante acciones no autorizadas.
Disponibilidad	-Los usuarios deben tener acceso a la información desde cualquier dispositivo sin que los mecanismos utilizados para la seguridad de los datos retrasen la obtención de los mismos.
Restricciones de diseño e implementación	Framework de desarrollo Flutter v3.3 Lenguaje de programación: Dart
Software	- El servidor de aplicación debe poseer un Procesador Intel Core i3 o superior y una memoria RAM de 4GB o superior. -Un navegador como Mozilla Firefox

II.3.3 3Historias de usuario

La metodología AUP-UCI, en su escenario 4 para la disciplina requisitos, genera las Historias de Usuario (HU), estas se utilizan para especificar los requisitos de software en la aplicación. Las HU se descomponen en tareas de programación y describen las características que el sistema debe cumplir, están escritas en un formato legible por el cliente, sin necesidad de sintaxis técnicas (Rodríguez, 2015). [CITATION Rod15 \l 1033]

A continuación, se muestra en las siguientes tablas varias historias de usuarios las cuales fueron generadas a partir de los requisitos funcionales y las mismas se estructuran de la siguiente forma:

Número: A cada HU se le asigna un número para facilitar su identificación por parte del equipo de desarrollo.

Nombre: Nombre descriptivo de la HU.

Prioridad: Grado de prioridad que le asigna el cliente a la HU en dependencia del valor en el negocio. Los valores que puede tomar son (Alta, Media o Baja).

Iteración Asignada: Número de la iteración en la cual será implementada la HU.

Programador: Nombre del programador encargado de desarrollar la HU.

Tiempo estimado: Esfuerzo estimado por el equipo de desarrollo para darle cumplimiento a la HU.

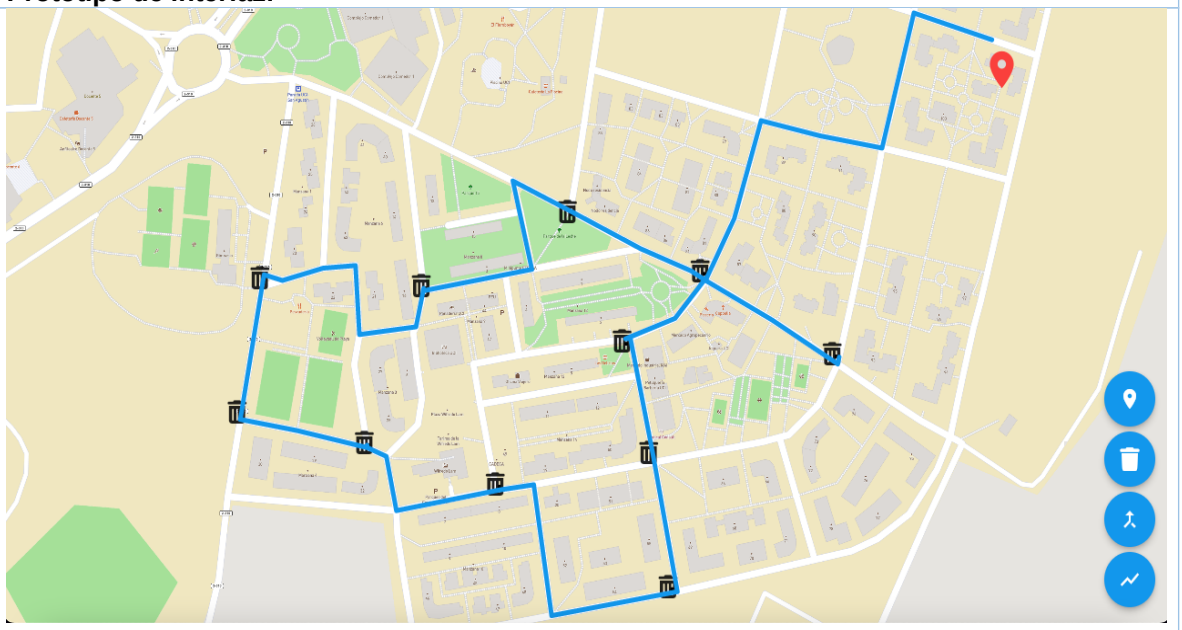
Tiempo real: Plazo real que el equipo de desarrollo tiene para dar cumplimiento a la HU.

Descripción: Descripción simple sobre lo que debe hacer la funcionalidad a la que se hace referencia.

Riesgo en Desarrollo: Grado de complejidad que le asigna el equipo de desarrollo a la HU luego de analizarla. (Alto, Medio o Bajo).

Tabla 7 Historia de Usuario “Gestionar ruta”
Fuente: [Elaboración propia]

Historia de Usuario	
Número: HU_2	Nombre Historia Usuario: Gestionar ruta
Modificación de Historia de Usuario Número: ninguna	
Usuario: Emilio Muñoz Monterrey y Sindy Nuñez Medina	Iteración Asignada:1
Prioridad en Negocios: Muy alta	Puntos Estimados: 8 días
Riesgo en Desarrollo: Alto	Puntos Reales: 8 días
Descripción: Permite adicionar, modificar, eliminar y mostrar una ruta asignada al usuario.	
Observaciones: **-Los usuarios deben estar autenticados y contar con los permisos pertinentes para gestionar la ruta.** **-Para realizar las acciones de Eliminar o Modificar debe existir al menos una ruta creada**	
Prototipo de interfaz:	



II.4 Diseño del sistema de gestión y arquitectura

II.4.1 Arquitectura del sistema

Al usar Flutter como framework de desarrollo, se utiliza necesariamente la arquitectura Modelo-Vista-Modelo de vista por ser sobre el cual se encuentra desarrollado el marco del trabajo. Se trata de un patrón de la arquitectura de software que se caracteriza por tratar de desacoplar lo máximo posible la interfaz de usuario de lógica de la aplicación facilitando las pruebas, mantenimiento y la escalabilidad de los proyectos.[CITATION Lou16 \l 1033]

Componentes del patrón MVVM

El modelo

Representa la capa de datos y/o la lógica de negocio, también denominado como el objeto del dominio. El modelo contiene la información, pero nunca las acciones o servicios que la manipulan. En ningún caso tiene dependencia alguna con la vista.

La vista

La misión de la vista es representar la información a través de los elementos visuales que la componen. Las vistas en MVVM son activas, contienen comportamientos, eventos y enlaces a datos que, en cierta manera, necesitan tener conocimiento del modelo subyacente.

Modelo de vista

El modelo de vista es un actor intermediario entre el modelo y la vista, contiene toda la lógica de presentación y se comporta como una abstracción de la interfaz. La comunicación entre la vista y el modelo de vista se realiza por medio los enlaces de datos.

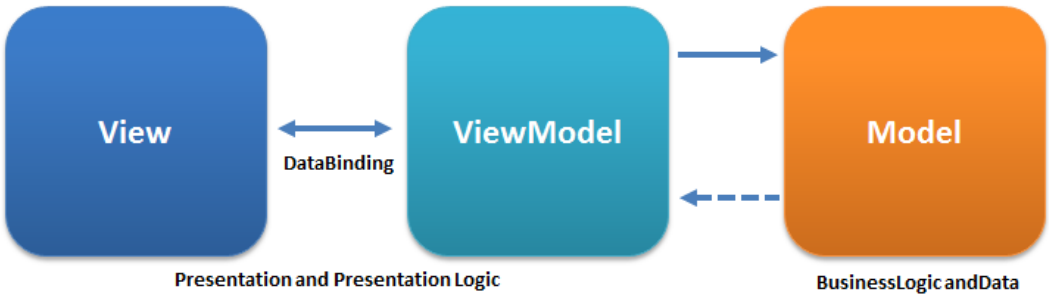


Figura 4 Patrón de Modelo-Vista-Modelo (MVVM)

Fuente: [Elaboración propia]

Ventajas y desventajas de MVVM (31)

Ventajas

- La lógica de negocio está desacoplada de la interfaz de usuario.
- Es más fácil de mantener y probar. Se puede hacer pruebas unitarias para el modelo y para vista-modelo, sin necesidad de hacer referencia a la vista.
- Los componentes pueden ser reutilizados
- El mantenimiento de los sistemas es simplificado.

Desventajas

- La curva de aprendizaje para nuevos desarrolladores es un poco superior a los otros modelos que son más simples.
- La distribución de componentes nos obliga a la creación y mantenimiento de un mayor número de ficheros.
- Debes de adaptarte a una estructura predefinida y eso incrementa la complejidad del sistema.

II.4.2 Patrones del diseño

El objetivo de los patrones es crear un cúmulo de bibliografía que ayude a los desarrolladores de software a resolver problemas recurrentes que surgen a lo largo del desarrollo. Los patrones ayudan a crear un lenguaje compartido para comunicar perspectiva y experiencia acerca de dichos patrones y sus soluciones. La codificación formal de estas soluciones y sus relaciones permite acumular con éxito el cuerpo de conocimientos que define nuestra comprensión de las buenas arquitecturas que satisfacen las necesidades de sus usuarios [CITATION Gam00 \l 1033]

Existen varios patrones de diseño utilizados en el proceso de desarrollo del software que describen los principios fundamentales del diseño de objetos. Estos son agrupados fundamentalmente por dos grandes grupos conocidos como Patrones de Software para la Asignación General de Responsabilidad (GRASP - *General Responsibility Assignment Software Patterns*) y “La Banda de los cuatro” (GOF - *Gang of Four*).

II.4.2.1 Patrones Generales de Asignación de Responsabilidades de Sistemas

Patrones GRASP

Los patrones GRASP constituyen un apoyo para entender el diseño de objetos y aplican el razonamiento para el diseño de una forma sistemática, racional y explicable. En consecuencia, los patrones GRASP son una codificación de principios básicos ampliamente utilizados (Larman, 2003). Los patrones GRASP empleados en el desarrollo de la extensión son descritos a continuación.[CITATION Gro11 \l 1033]

A continuación, se ejemplifican los patrones utilizados en la propuesta de solución.

Patrón Experto

La responsabilidad de asignar una labor a la clase que tiene o puede tener los datos necesarios para cumplir determinada responsabilidad, es la solución que pretende dar este patrón ante el problema de cómo realizar la asignación de la forma más eficiente posible. Si estas asignaciones de responsabilidades se hacen adecuadamente, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, lo que nos ofrece la garantía de poder reutilizar los componentes en futuras aplicaciones. Se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos.[CITATION Lar031 \l 1033]

Este patrón se utiliza en esta librería para realizar su capa de abstracción en el modelo, encapsula toda la lógica de los datos y generar las clases con todas las funcionalidades

que están relacionados directamente con la entidad que representan.

```dart
import 'package:flutter/material.dart';
import 'package:hive_flutter/hive_flutter.dart';

part 'hive_chofer_model.g.dart';

@HiveType(typeId: 0)
class Chofer extends HiveObject{
  @HiveField(0)
  final String ci;

  @HiveField(1)
  final String name;

  @HiveField(2)
  final String job;

  Chofer( {required this.ci, required this.name, required this.job} );

}
```

Figura 5 Ejemplo de patrón experto

Fuente:[Elaboración propia]

Patrón Creador

Este patrón guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Con esto se favorece el bajo acoplamiento. [CITATION Lar16 \l 1033]

Este patrón ayuda a identificar quien debe ser el responsable de la creación de nuevos objetos. Es donde se asigna la responsabilidad de que una clase B cree un objeto de la clase A.

```dart
import 'package:flutter/material.dart';
import 'package:hive_flutter/hive_flutter.dart';
import 'package:proyecto_definitivo/models/hiveModels/hive_chofer_model.dart';


class ChoferOperations extends HiveObject{
 ChoferOperations();

   Future<int> saveChofer(Chofer chofer)async{
    final Box<Chofer> box = await Hive.openBox<Chofer>('choferes');
    return box.add(chofer);
  }
```

Figura 6 Ejemplo de patrón creador
Fuente:[Elaboración propia]

II.4.2.2 Los patrones GOF se revelan como una forma indispensable de enfrentarse a la programación. Los patrones "Banda de los Cuatro" (Gang-of-Four) describen las formas comunes en que diferentes tipos de objetos, pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases y la formación de estructuras de mayor complejidad. Además, permiten crear grupos de objetos para ayudar a realizar tareas complejas. [CITATION Gam98 \l 1033]

Patrón Método de Fabricación (Factory Method): Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar; su objetivo es devolver una instancia de múltiples tipos de objetos, que normalmente provienen de una misma clase padre y sólo se diferencian entre ellos por algún aspecto de comportamiento. En el proyecto se puede ver que las clases creadoras provienen de una misma clase padre HiveObject.

```dart
class ChoferOperations extends HiveObject
```

Figura 7 Ejemplo de patrón método de fabricación
Fuente:[Elaboración propia]

Patrón Inyección de dependencias: Este patrón es utilizado en los contenedores de servicios (o contenedores de inyección de dependencias), los cuales son objetos DART que gestionan la creación de instancias de servicios, es decir, objetos.

```dart
final authProvider = Provider.of<AuthProvider>(context);
```

Figura 8 Ejemplo de patrón inyección de dependencias

Fuente:[Elaboración propia]

El patrón Iterador es de tipo comportamiento a nivel de objetos. A través de su utilización es posible acceder de forma secuencial a cada uno de los elementos de un objeto agregado sin exponer su representación interna. Además, permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos. Su utilización tributa al incremento de la flexibilidad porque es posible utilizar nuevas formas de recorrer una estructura con solo modificar el iterador en uso, cambiarlo por otro o definir uno nuevo. También facilita el paralelismo y la concurrencia, pues es posible que dos o más iteradores recorran una misma estructura simultánea o solapadamente [CITATION Gam95 \l 1033]

En la extensión propuesta, es aplicado este patrón directamente en la implementación de una clase de la interfaz para mostrar en un listado todos los datos referentes a la entidad chofer.

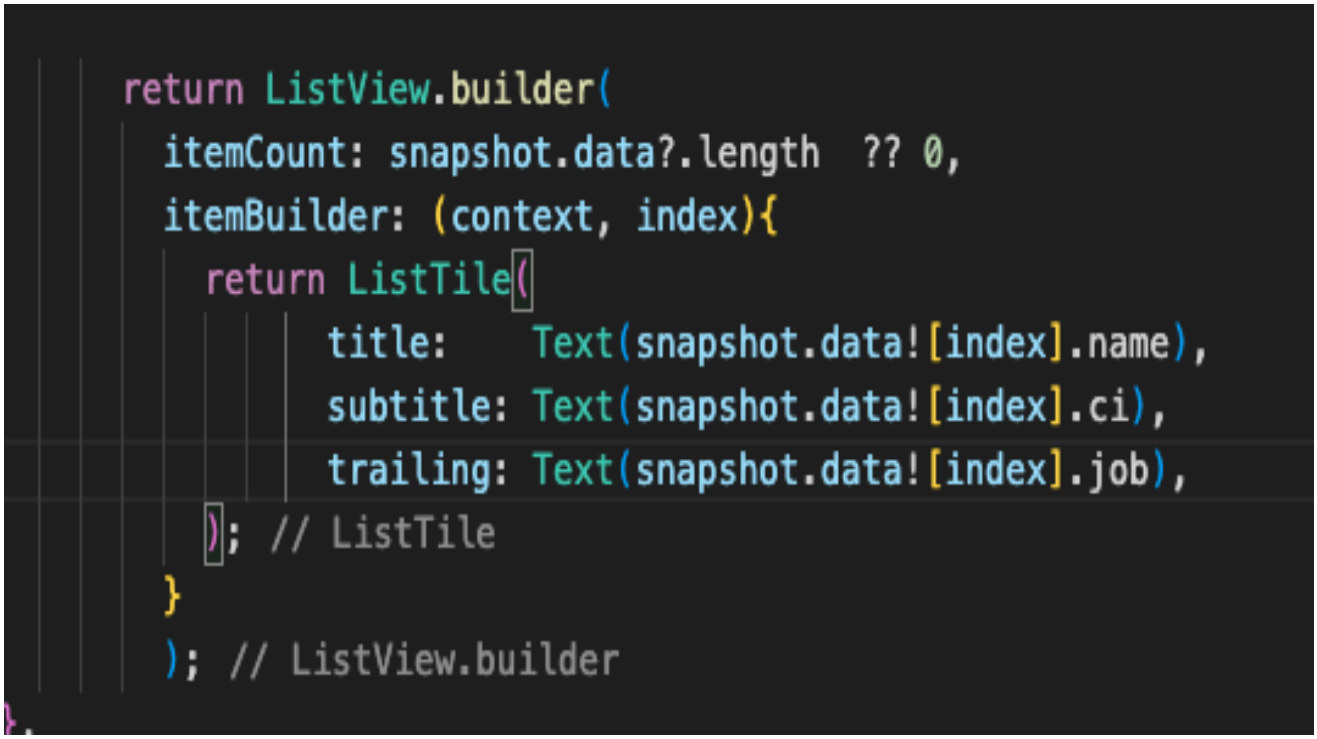


Figura 9 Ejemplo de patrón iterador

Fuente:[Elaboración propia]

II.4.3 Diagrama de clases de diseño

Los diagramas de clases representan la estructura del sistema que será implementado, estos contienen clases, atributos, métodos y relaciones.[CITATION Ken16 \l 1033]

Las clases del diseño representan una abstracción de una o varias clases en la implementación del sistema. El lenguaje utilizado para especificar una clase del diseño es el mismo que el lenguaje de programación que se emplea en la solución, los métodos tienen corres-

pondencia directa con el debido método de la implementación de clase. En la figura #3 se muestra el Diagrama de Clases del Diseño de gestionar ruta.

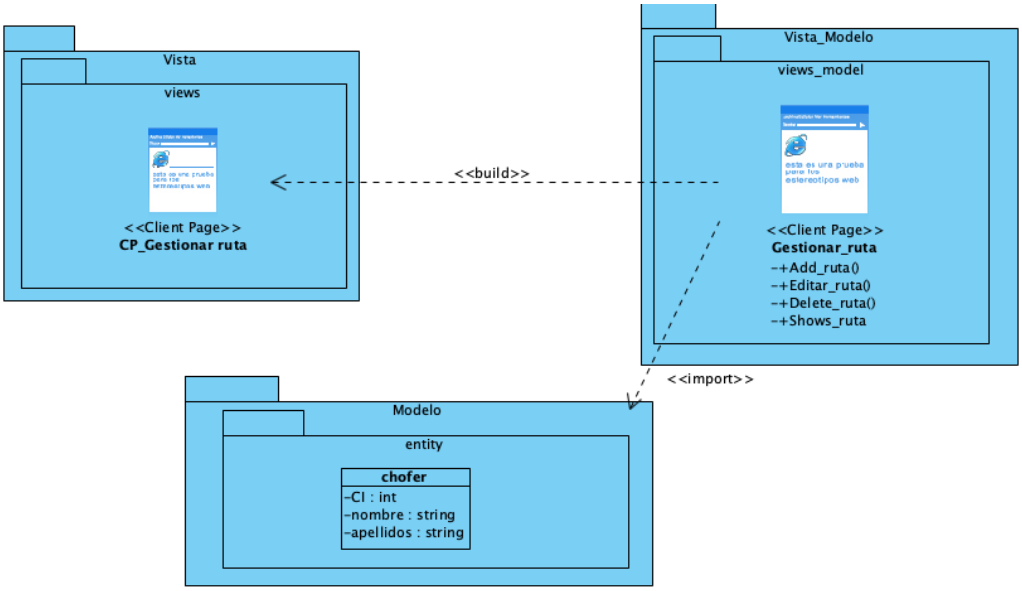


Figura 10 Diagrama de Clases del Diseño de Gestionar Ruta.

Fuente:[Elaboración propia]

Conclusiones del capítulo

A partir del desarrollo del sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI (Universidad de Ciencias Informáticas)**,** el modelo del dominio nos permitió conocer y comprender los principales conceptos asociados al producto. Además, se realizó una propuesta y concepción del sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI (Universidad de Ciencias Informáticas)**.** Una vez definidas las directivas del desarrollo se realizó el levantamiento de los requisitos, funcionales y no funcionales, con los cuales debía de cumplir el sistema. La descripción de los requisitos funcionales fue apoyada en las historias de usuario, definiendo las funcionalidades del sistema, lo que permitió al programador una mayor claridad a la hora de codificar. El diseño del diagrama de clases, la definición de la arquitectura de desarrollo, la representación de los principales patrones de diseño (GRASP y GOF) permitieron obtener una idea concisa del sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI (Universidad de Ciencias Informáticas) los objetivos que persigue y los elementos fundamentales a tener en cuenta en el desarrollo de la solución.

Capítulo III: Validación mediante pruebas del sistema de gestión de ruta utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI (Universidad de Ciencias Informáticas)

Las pruebas de software consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados , por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo. Lo cual no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software. Debido a la importancia que posee esta fase, el siguiente capítulo estará orientado a la realización de pruebas a las distintas funcionalidades del software para verificar que funcionen correctamente.

III.1 Diagrama de despliegue

Este tipo de diagrama se realiza con el objetivo de mostrar las relaciones físicas de los distintos nodos que componen un sistema y la distribución de los componentes sobre dichos nodos. Además, se modela la arquitectura en tiempo de ejecución de un sistema. [CITATION Cil17 \l 1033]

Figura 11 Diagrama de despliegue
Fuente:[Elaboración propia]

Descripción de los Nodos

Servidor de Aplicaciones: Es el nodo encargado de tener instalado el sistema al que tendrán acceso los usuarios desde las estaciones de trabajo, como servidor web Apache 2.4.10 .

PC Cliente: Es el nodo que representa la estación de trabajo que permite al usuario mediante el protocolo HTTPS y el puerto 8080 acceder a la aplicación, utilizando los navegadores Mozilla Firefox 48.0, Google Chrome 20.0 o Internet Explorer 15.0.

III.1.2 Diagrama de componente

Un componente es un módulo de software que puede ser código fuente, código binario, un ejecutable, o una librería con una interfaz definida. Una interfaz establece las operaciones externas de un componente, las cuales determinan una parte del comportamiento del mismo. Además, se representan las dependencias entre componentes o entre un componente y la interfaz de otro, es decir uno de ellos usa los servicios o facilidades del otro.

Estos diagramas pueden incluir paquetes que permiten organizar la construcción del sistema de información en subsistemas y que recogen aspectos prácticos relacionados con la secuencia de compilación entre componentes, la agrupación de elementos en librerías, etc.[CITATION Cil171 \l 1033]

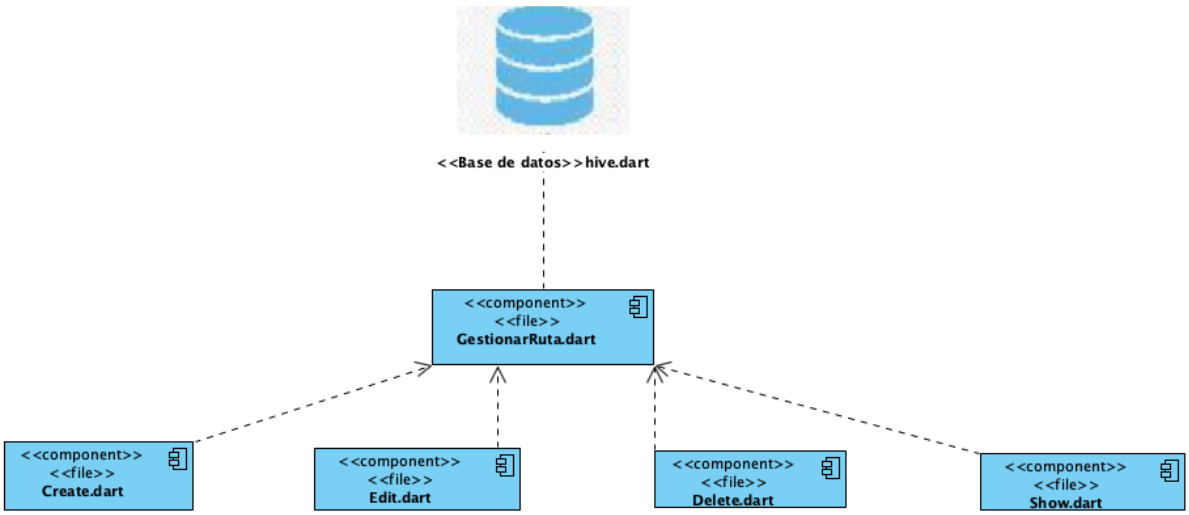


Figura 12 Diagrama de Componente

Fuente:[Elaboración propia]

III.2 Estándar de codificación

Un estándar de codificación son reglas que se siguen para la escritura del código fuente. Estos permiten que otros programadores puedan identificar las variables, las funciones o métodos al leer los códigos de otras personas. Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto para un lenguaje de programación permite que todos los participantes lo puedan entender en menos tiempo y que cualquier persona que se desempeñe como codificador de dicho lenguaje pueda interpretar de manera eficiente.[CITATION ÁL10 \l 1033]

A continuación se exponen, más detalladamente, los estándares de codificación seguidos en el desarrollo del sistema:

III.2.1 Indentación, llaves de apertura y cierre

El código debe usar cuatro espacios para la indentación en vez de usar el tabulado, esto minimiza problemas con otras herramientas de desarrollo.

Las llaves de apertura deben ir en la siguiente línea y la llave de cierre debe ir en la siguiente línea después del cuerpo.

Los paréntesis en las estructuras de control, no deben usar espacios antes ni después. Se añade un solo espacio después de cada limitador de coma y alrededor de los operadores (==, &&, ...).

```
ElevatedButton.icon(onPressed: ()async{
  // Buscar en hive el chofer por CI y eliminar
 for (var i = 0; i < listaChoferes.length; i++) {
   if (listaChoferes[i].ci == delete_controller.text) {
     await listaChoferes[i].delete();
     getData();
     delete_controller.text = '';

   }
 }

},
```

Figura 13 Indentación, llaves de apertura y cierre, y tamaño de líneas

Fuente:[Elaboración propia]

III.2.2 Convención de nomenclatura

Variables: Se rigen por la nomenclatura camelCase[1] para declarar. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

Se muestra en el método cronogramaAction (Request $request) un ejemplo donde se emplean las variables:

Clases: Se rigen por la nomenclatura StudlyCaps[2]. El patrón que sigue para declarar las clases es que siempre comienzan con mayúscula y en caso de nombre compuesto cada palabra comienza en mayúscula, sin espacios o guion bajo, como se muestra a continuación:

Funciones: Se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa.

---

1 camelCase: Es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en camelCase se asemejan a las jorobas de un camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula.

2 Studlycaps es una forma de notación de código en el que la capitalización de letras varía según un patrón, o arbitrariamente, por lo general también omitiendo espacios entre las palabras y, a menudo omitiendo algunas letras.

```
static Handler login = Handler(
  handlerFunc: ( context, params ) {
    final authProvider = Provider.of<AuthProvider>(context!);
    if (authProvider.authStatus == AuthStatus.notAuthenticated) {
      return LoginView();

    }else{
      return DashboardView();
    }
  }
```

Figura 14 Convención de nomenclatura: función: camelCase

Fuente:[Elaboración propia]

III.2.3Estructuras de control

Las estructuras de control incluyen if, for, for each, while, switch, entre estas estructuras y los paréntesis que encierran la condición debe de existir un espacio. Es recomendable utilizar en cualquier caso llaves de apertura y cierre al comienzo de una nueva línea, incluso en situaciones en las que técnicamente son opcionales.

```
if (authProvider.authStatus == AuthStatus.notAuthenticated) {
  return LoginView();

}else{
  return DashboardView();
}
```

Figura 15 Estructuras de control

Fuente:[Elaboración propia]

III.2.4 Comentarios en las funciones.

Todas las funciones deben tener un comentario, antes de su declaración, explicando que hacen. Ningún programador debería tener que analizar el código de una función para conocer su utilidad. Tanto el nombre como el comentario que acompañe a la función deben bastar para ello.

```
//Despues de insertar deja los campos te texto en blanco
  void clear() {
    name_controller.text = '' ;
    ci_controller.text = '';
    job_controller.text = '';
    setState(() {

    });
  }
```

Figura 16 Comentarios en funciones

Fuente:[Elaboración propia]

III.3 Validación de requisitos

La validación de requisitos examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos y que el resultado del trabajo se ajusta a los estándares establecidos para el proyecto y el producto.[CITATION Alv10 \l 1033]

III.3.1Criterios para validar los requisitos

Para validar los requisitos del sistema según Pressman, se pueden chequear mediante un cuestionario guiado por un conjunto de interrogantes, con el objetivo de descubrir la mayor cantidad de errores posibles.[CITATION McG13 \l 1033]

Interrogantes para la validación de requisitos:

¿Está el requisito claramente definido?
¿Puede interpretarse mal?
¿Está identificado el origen del requisito (por ejemplo: persona, norma, documento)?
¿El planteamiento final del requisito ha sido contrastado con la fuente original?
¿El requisito está delimitado en términos cuantitativos?
¿Qué otros requisitos hacen referencia al requisito estudiado?
¿El requisito incumple alguna restricción definida?
¿El requisito es verificable? Si es así, ¿se pueden efectuar pruebas para verificar el requisito?

¿Se puede seguir el requisito en el sistema que se ha desarrollado?
¿Está el requisito asociado con los rendimientos del sistema o con su comportamiento?
¿El requisito está implícitamente definido?
¿El requisito es modificable?
¿El requisito está completo?
¿El requisito puede ser implementado?
¿El requisito puede ser probado?
¿El resultado de la evaluación de impacto es positivo?

Resultado de aplicar los criterios de validación

Luego de aplicar el conjunto de interrogantes para validar los requisitos definidos para el desarrollo del sistema, se obtuvo el 100 % de aprobación por parte del cliente.

III.3.2 Técnicas de validación de requisitos

Con el objetivo de obtener una mayor calidad y demostrar que los requisitos definidos realmente describen el sistema que el cliente necesita; se utilizaron las técnicas para la validación de requisitos siguientes[CITATION Pad12 \l 1033]:

Prototipado de interfaz: permite al cliente entender fácilmente la propuesta de solución, al brindar la representación aproximada de la interfaz de usuario que tendrá el sistema. Existen dos tipos principales de prototipo de interfaz:

Desechables: se utilizan solo para la validación de los requisitos y posteriormente se desechan. Pueden ser prototipos en papel o en software.

Evolutivos: una vez utilizados para la validación de los requisitos, se mejora su calidad y se convierten progresivamente en el producto final. El tipo utilizado finalmente para la propuesta de solución fue Evolutivos pues de esta forma se reutiliza el prototipo diseñado en todo el proceso de desarrollo del sistema.

Generación de casos de prueba: se realizaron los diseños de casos de pruebas para cada uno de los requisitos obtenidos, permitiendo verificar que todos se pudieran probar y determinar en la medida de la complejidad del diseño de caso de prueba, identificando requi-

sitos que deberían ser reconsiderados y cuales pueden ser los más difíciles de implementar.

Resultado de aplicar las técnicas de validación

Como resultado de este proceso se identificaron inconsistencias en las especificaciones tales como la falta de concordancia entre la complejidad de la especificación y la registrada en el documento de evaluación de requisitos. Descripciones de requisitos poco detalladas o ambiguas. Se encontraron además numerosos errores ortográficos.

III.4 Pruebas realizadas a la solución

Para evaluar la calidad del sistema que se está desarrollando y verificar el cumplimiento de los objetivos trazados, se aplicaron un conjunto de pruebas definidas por Pressman en su libro de Ingeniería del software.

“Un enfoque práctico”, en su quinta edición. A continuación, se muestra la estrategia de prueba diseñada para aplicar en la solución desarrollada:

Tabla 8 Pruebas realizadas a la solución
Fuente:[Elaboración propia]

Pruebas	Método	Técnica
Prueba de unidad	Caja blanca	Camino básico
Prueba de Aceptación	Caja negra	Partición de equivalencia

III.4.1 Pruebas internas

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas.[CITATION San15 \l 1033]

Prueba de unidad

Las pruebas de unidad tienen como objetivo verificar la unidad más pequeña del diseño del software. Estas pruebas se concentran en la lógica del procesamiento interno y en las estructuras de datos tales como: código fuente, archivos binarios, archivos de datos, entre otros. Este tipo de prueba se puede aplicar en paralelo a varios componentes. [CITATION Pre10 \l 1033]

Las pruebas de unidad se realizan mediante el método de caja blanca o estructural, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. La técnica de prueba de caja blanca utilizada en la investigación es el camino básico, que permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.[CITATION Cas14 \l 1033]

Técnica de camino básico

La técnica de camino básico es empleada en el método de caja blanca, su objetivo es comprobar que cada camino se ejecute independiente de un componente o programa, obteniendo la complejidad lógica del diseño. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática.[CITATION Pre101 \l 1033]

Pressman propone como estrategia para aplicar camino básico, realizar un análisis de la complejidad ciclomática de cada procedimiento que componen las clases del sistema; una vez concluido este paso se selecciona el método con valor de contener errores, además de que ofrece una medida del número de pruebas que deben diseñarse para validar la correcta implementación de una determinada función.

Luego se determina la complejidad ciclomática V(G) del grafo resultante, el cual indica el número de caminos independientes que existen en un grafo, es cualquier camino dentro del código que introduce por lo menos un nuevo conjunto de sentencias de procesos o una nueva condición. La complejidad ciclomática se puede calcular de 3 formas:

1. V (G) = (A - N) + 2

2. V (G)=P+ 1

3. V (G) = R

Donde:

A: es la cantidad de aristas.

N: la cantidad de nodos.
P: es el número de nodos predicado contenidos en el grafo de flujo G
R: representa la cantidad de regiones en el grafo

Realizando los cálculos correspondientes se obtiene por cualquiera de las variantes el siguiente resultado:

1.V (G) = (A- N) + 2
V (G) = 4 - 54+ 2

V (G) = 2

2.V (G) = P + 1

V (G) = 1+1

V (G) = 2

3.V (G) = R = 2

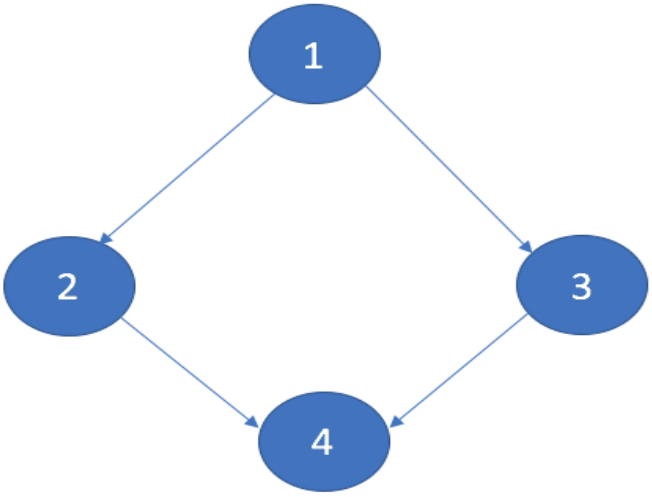


Figura 17 Grafo de flujo

Fuente:[Elaboración propia]

```
ElevatedButton.icon(onPressed: () async{

if ( formKey.currentState!.validate()) {

 await choferOperations.saveChofer(Chofer(ci: ci_controller.text, name: name_controller.text, job: job_controller.text));
}
else{
 print('Datos de chofer incorrectos');
}
await getData();
```

Figura 18 Ejemplo de código utilizado para calcular la complejidad ciclomática

Fuente:[Elaboración propia]

La siguiente tabla muestra el análisis de la complejidad ciclomática en un software: [CITATION Mar12 \l 1033]

Tabla 9 Análisis de riesgo de la Complejidad Ciclomática

Fuente:[Elaboración propia]

Complejidad ciclomática.	Evaluación del riesgo.
1 – 10	Programa simple, sin mucho riesgo.
11 – 20	Más complejo, riesgo moderado.
21 – 50	Complejo, programa de alto riesgo.
50 en adelante.	Programa no testeable, muy alto riesgo.

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que la complejidad ciclomática del código es de 2. Existen 2 posibles caminos por donde el flujo puede circular, coincidiendo con el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Tabla 10 Trayectorias básica obtenida en el grafo
Fuente:[Elaboración propia]

No.trayectorias	Trayectorias.
1	1-2-4
2	1-3-4

Casos de prueba por trayectoria

Tabla 11 Caso de prueba para la trayectoria básica 1
Fuente:[Elaboración propia]

Descripción	Se inserta un chofer
Camino	1-2-4
Entrada	Se introduce el nombre, CI y la zona asignada al chofer
Resultados	Se debe mostrar una alerta especificando que el chofer se ha creado correctamente
Condiciones	El chofer debe tener nombre, CI y la zona asignada

Tabla 12 Caso de prueba para la trayectoria básica 2
Fuente:[Elaboración propia]

Descripción	Se inserta un chofer
Camino	1-3-4
Entrada	Se introduce el nombre, CI y la zona asignada al chofer
Resultados	Se debe mostrar una alerta especificando que el chofer se ha creado correctamente y otra alerta si se ha creado incorrectamente.
Condiciones	El chofer debe tener nombre, CI y la zona asignada

A partir de la ejecución los casos de pruebas obtenidos a través de la aplicación de la técnica camino básico, se concluye que los mismos fueron probados satisfactoriamente demostrando que el código generado no se encontraron ciclos infinitos y no existe código innecesario en el sistema desarrollado.

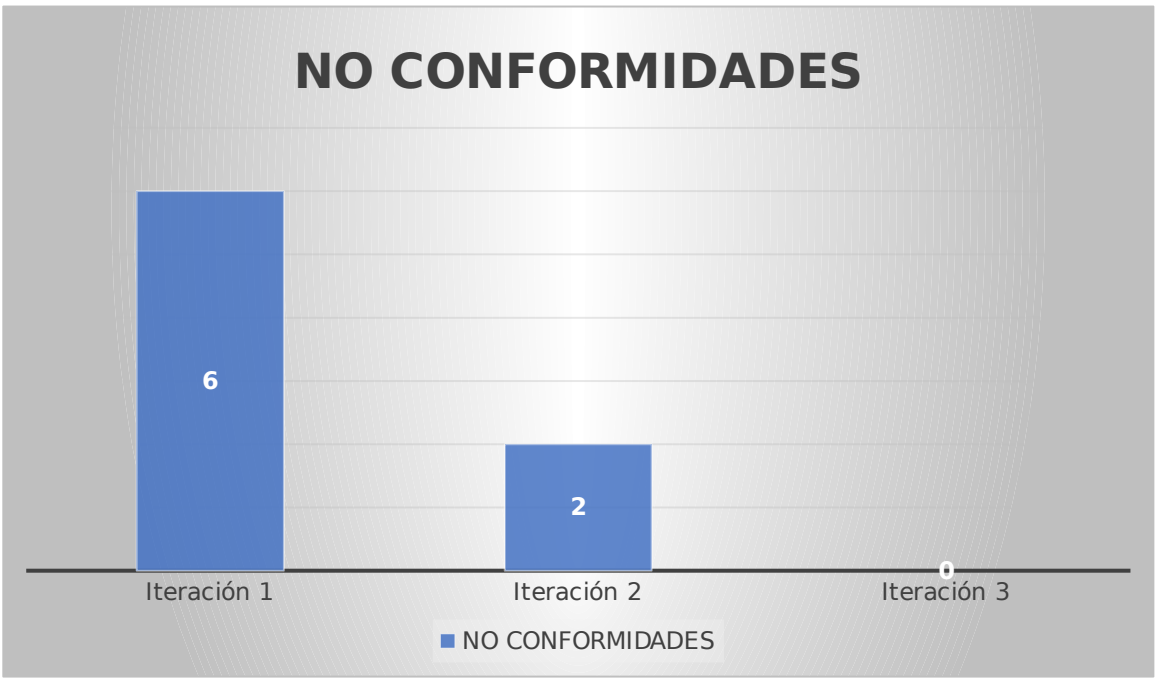


Figura 19 No conformidades en el método de caja blanca

Con el objetivo de comprobar que las funcionalidades de la herramienta se realizaron correctamente y responden a las necesidades del cliente, el método se aplicó en tres iteraciones como se muestra en la Figura 19. En la primera se detectaron un total de 6 No Conformidades (NC). En la segunda iteración los resultados mejoraron al disminuir a 2 NC, ya en la tercera iteración se logró resolver las mismas obteniéndose cero NC. La imagen anterior ilustra los resultados de aplicar el método de caja negra, teniendo en cuenta las NC.

III.4.2 Pruebas de aceptación

Las pruebas de aceptación funcionan con la participación del cliente para ayudar a definir los criterios de pruebas pues para el que realiza el software será difícil comprobar que está incorrecto. El cliente, junto a todos los miembros del equipo de desarrollo, define los escenarios de pruebas que describen lo que debe hacer el sistema y cómo debe hacerlo. [CITATION Hal091 \l 1033]

En la ingeniería de software, las pruebas de aceptación (PA) se realizan para establecer el grado de confianza en un sistema, partes del mismo o en sus características no funcionales. La confianza en el sistema estará determinada por su grado de adherencia a las necesidades, requerimientos y procesos de negocio solicitados por el usuario o cliente. Es en función a estos que el usuario debe decidir si acepta o no el sistema que le está siendo entregado.

“A grandes rasgos se puede decir que están basadas más en cómo comprobar que el sistema en desarrollo cumple los requisitos del cliente y menos en reducir el número de errores en el código. En otras palabras, las pruebas de aceptación no son acerca de las pruebas de código, sino sobre lo que desea el cliente con el sistema o de su negocio” [CITATION Hal14 \l 1033]

Tabla 13 Prueba de aceptación de la HU Gestionar ruta

Fuente:[Elaboración propia]

Caso de Prueba de Aceptación

Código Caso de Prueba: HU_1-1	**Nombre Historia de Usuario:** Gestionar ruta
Nombre de la persona que realiza la prueba: Sindy Nuñez Medina y Emilio Muñoz Monterrey	
Descripción de la Prueba: Consiste en evaluar el requisito Gestionar ruta desde el sistema de gestión de ruta del transporte de residuos sólidos.	
Condiciones de Ejecución: Debe existir un código escrito en la sección correspondiente del sistema.	
Entrada / Pasos de ejecución: Para evaluar el requisito es necesario: Se debe registrar como chofer en el sistema utilizando el usuario y la contra-seña correcta, luego en la siguiente página se tiene un botón mapa en donde se puede gestionar la ruta	
Resultado Esperado: El usuario con rol chofer puede adicionar, eliminar, editar y listar rutas	
Evaluación de la Prueba: Satisfactoria.	

Método de Caja Negra

Las pruebas de caja negra se centran en los requisitos funcionales del software, es decir, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerci-ten completamente todos los requisitos funcionales de un programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías [CITATION Pre103 \l 1033]

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.
- Se utiliza la técnica de partición equivalente para llevar a cabo el método de caja negra, a continuación, se describe.

Técnica de prueba: Partición equivalente

Esta es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba se basa en la evaluación de las clases de equivalencia para una condición de entrada, así mismo, una condición de entrada es un valor numérico, un rango de valores, un conjunto de valores relacionados o una condición lógica. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada [CITATION Est131 \l 1033]

Para aplicar esta técnica, se deben primeramente realizar el diseño de casos prueba (DCP) [3]con el objetivo de obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software.

Tabla 14 Caso de prueba del escenario Adicionar chofer

Fuente:[Elaboración propia]

Descripción	Variable 1	Respuesta del sistema	Flujo central
Permite adicionar chofer	V (Introducir va-	El sistema	1. Seleccionar el botón Crear chofer 2. Introducir valores en el cam-
	I (introducir ca-	El sistema	
	I (Dejar campos	El sistema	

3 Un DCP es, en ingeniería del software, un conjunto de condiciones o variables bajo las cuales un analista determinará si una aplicación o una característica de estos es parcial o completamente satisfactoria

Figura 20 No conformidades del método Adicionar chofer

Fuente:[Elaboración propia]

Con el objetivo de comprobar que las funcionalidades de la herramienta se realizaron correctamente y responden a las necesidades del cliente, el método se aplicó en tres iteraciones como se muestra en la Figura 20. En la primera se detectaron un total de 4 No Conformidades (NC). En la segunda iteración los resultados mejoraron al disminuir a 2 NC, ya en la tercera iteración se logró resolver las mismas obteniéndose cero NC. La imagen anterior ilustra los resultados de aplicar el método de caja negra, teniendo en cuenta las NC.

III.4.3 Funcionalidades obtenidas

El sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de los residuos sólidos en la UCI está compuesto por 34 funcionalidades, una por cada requisito funcional establecido durante la fase de análisis. La implementación de estas se realizó de acuerdo a lo establecido por el cliente, ejecutando los pasos necesarios para obtener los resultados esperados.

A partir de que un usuario se registra y se autentica en el sistema obtiene distintos permisos. Un usuario con permisos de director cuenta con la capacidad de gestionar todos los datos. El usuario que no tiene acceso a este permiso, mediante el rol de chofer, puede entrar al sistema obtener información asignada por el director y visualizar mediante un mapa las rutas más óptimas según el destino inicial y final.

El fin del sistema permite que el usuario chofer pueda entrar al sistema y poder modificar tanto las rutas óptimas disponible como cambiar su destino.

Dentro de los beneficios que tiene el software se encuentra:

El principal beneficio del sistema es que les permite a los usuarios consultar rutas, como llegar desde un punto hasta el otro de la manera más óptima posible.

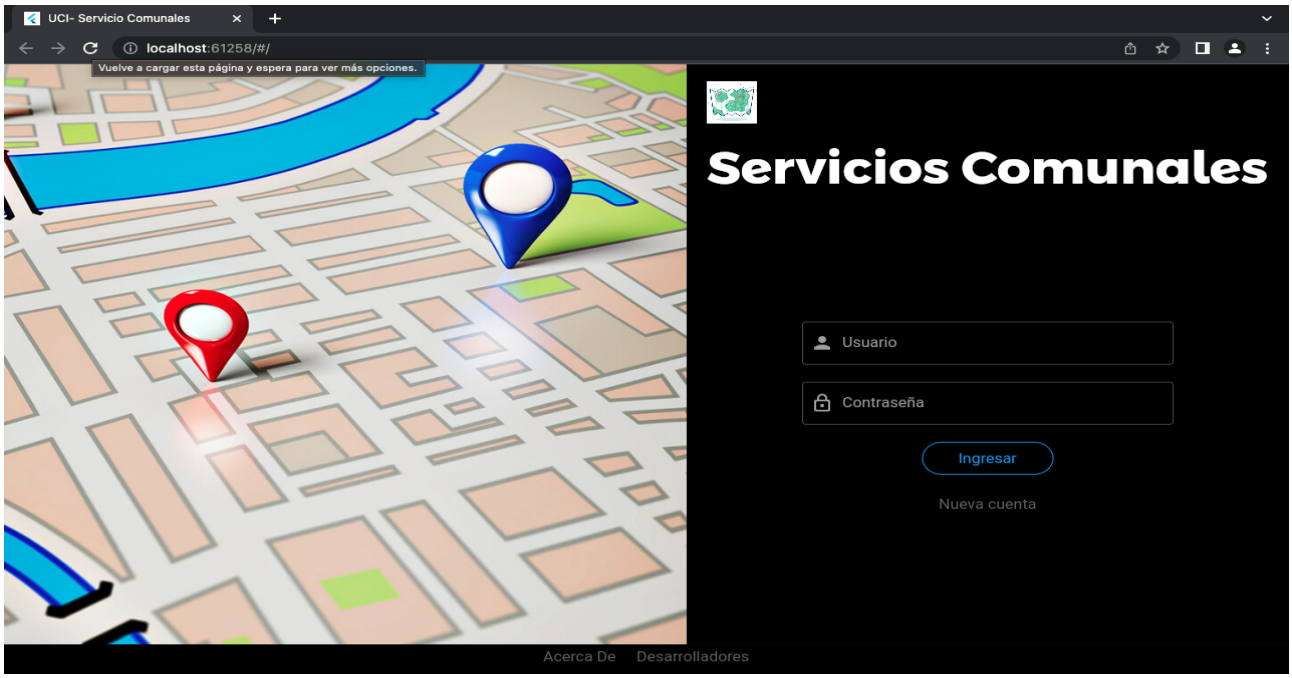


Figura 21 :Login del sistema realizado

Fuente:[Elaboración propia]

Figura 22:Mapa donde se visualiza la ruta más corta

Fuente: [Elaboración propia]

Conclusiones del capítulo

En este capítulo se detallaron los estándares de diseño y codificación empleados en la implementación del sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI), permitiendo una mejor organización y comprensión del código. Con los resultados de la validación de los requisitos identificados se logró obtener un listado de requisitos correctamente redactados , la descripción de las pruebas utilizadas para asegurar la calidad del software, como fueron las pruebas de unidad mediante la utilización de la técnica del camino básico lo que permitió corregir errores en el código de manera independiente, disminuyendo la dificultad de las pruebas funcionales, también se realizaron las pruebas de aceptación para probar cada uno de los requisitos propuestos en las HU, logrando verificar el cumplimiento de los requisitos funcionales establecidos.

CONCLUSIONES FINALES

- (1) Se definieron los principales conceptos asociados a la investigación, mediante los cuales de establecieron las bases de la fundamentación teórica. Se realizó el estudio del software homólogo, evidenciándose la necesidad de desarrollar un nuevo sistema de gestión de rutas utilizando algoritmos de optimización para el transporte de residuos sólidos de la Universidad de las Ciencias Informáticas (UCI). Además, se describieron las principales características de las tecnologías definidas para la implementación de la solución y se seleccionó la metodología de desarrollo a utilizar.

- (2) Basándose en las pautas establecidas por la metodología, se realizó el análisis de la propuesta de solución junto al cliente, permitiendo a los autores conocer el negocio asociado a su desarrollo e identificar los requisitos funcionales del sistema. A partir de esto se diseñaron un conjunto de artefactos para fundamentar y describir el desarrollo de la solución, como fueron las historias de usuario, el diagrama de clases del diseño y el modelo del dominio, permitiendo guiar al equipo de desarrollo durante la fase de implementación y pruebas.

- (3) Se aplicaron pruebas de caja blanca, específicamente la técnica de camino básico a partes del código del sistema. Se realizaron también pruebas de funcionalidad una vez terminado el software, permitió evaluar el desarrollo de los requisitos definidos e implementados en el sistema. También se realizaron pruebas de caja negra mediante la técnica partición de equivalencia. El desarrollo del software se realizó de acuerdo a los requisitos establecidos, obteniéndose un sistema funcional que realiza la gestión de rutas optimizando el transporte de los residuos solidos en la UCI.

RECOMENDACIONES

Luego de realizar el presente trabajo de investigación, analizar la realidad problemática, presentar una solución para mejorarla, y realizar pruebas, se recomienda:

1. Por ser un proyecto de tesis, que tiene como característica la escalabilidad, puede ser antecedente para futuras investigaciones o para la mejora de la misma, integrando módulos a la aplicación ya propuesta.

2. Se recomienda que el sistema sea trasladado a una aplicación móvil para optimizar la experiencia del usuario.

BIBLIOGRAFÍA REFERENCIADA

1. Incidencia de las TIC en la enseñanza en el sistema educativo español: Una revisión de la investigación. Bravo, María Pilar Colás, Juan de Pablos Pons, y Javier Ballesta Pagán. 26 de marzo de 2018, Revista de Educación a Distancia (RED), Vol. 56.

2. Correa Espinal, Alexander, and Rodrigo Andres Gomez Montoya. "Information technologies in supply chain management.". 2013. págs. 37-48.

3. da Silva, R. B., Shimoishi, J. M., & Mariano, A. M. Revisión sistemática de la bibliografía sobre tecnologías de información y comunicación (tic) aplicadas al transporte público (tp) a partir del enfoque meta analítico. 2012.

4. Jorge Rohan Requejo Falla, Juan Antonio Torres Benavides. Aplicación web utilizando geolocalización en tiempo real y aplicando la teoría de redes, para mejorar el proceso de trazabilidad de rutas y la distribución de insumos de panadería en la empresa Dipropan Sac. 2021.

5. Moscardó, Celia Mas. Planificación de rutas y operaciones de transporte por carretera. s.l. : Editorial Elearning, SL,, 2015.

6. Andrés Aguado arando, Javier Jiménez de Vega Dirigido. Optimización de rutas de transporte. 2012/2013.

7. Alarcón, Msc. Paquita Lourdes Velásquez. gestión de residuos sólidos urbanos:factores que limitan su adecuada implementación. 2017.

8. Rodriguez. 2018.

9. Medina-León, A., Nogueira-Rivera, D., Hernández-Nariño, A., & Díaz-Navarro, Y. Consideraciones y criterios para la selección de procesos para la mejora: Procesos Diana. Ingeniería Industrial,. 2014. págs. 272-281.

10. "Teoría general de sistemas.". Gigch, John P. van. México : Editorial Trillas, 1987.

11. Leonard, A., & Beer, S. The systems perspective: Methods and models for the future. 1994. AC/UNU Project.

12. Influence organizational commitment on the quality of accounting information system." . Syaifullah, Muhammad. 3,9, 2014, nternational Journal of Scientific & Technology Research, pág. 305.

13. "Classification of huminite—ICCP System. Sýkorová, I., et al. 2005, International Journal of Coal Geology, págs. 85-106.
14. Gestión de la Calidad. Conceptos, enfoques, modelos y sistemas. Camisón, C., Cruz, S. y González, T. Madrid, España, : Pearson Educación, S.A., 2016.
15. Andrés Aguado Arando, Javier Jiménez de Vega Dirigido. Optimización de rutas de transporte. 2014/2015.
16. Vidal. 2013.
17. Macedo Rojas, Yolaina Malí. "Programa de sensibilización sobre norma técnica de salud para la mejora del manejo de residuos sólidos hospitalarios en el Centro de Salud Palmira, Independencia-Huaraz, 2017.". 2017. MINSA/DIGESA.
18. Davis, l.,. Handbook of genetic algorithms, van nostrand reinhold, . 1991.
19. Fogel, d.,. Evolutionary computation, ieee press,. 1995.
20. Mayta, Rosmeri. "Algoritmo evolutivo para el problema de árbol de expansión mínima (MST).". 2014. págs. 64-67.
21. Torrubia, G., and V. Terrazas. "Algoritmo de Dijkstra. Un tutorial interactivo.". s.l. : VII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2001), 2012.
22. J. A torres Benavides. «investigación de operaciones y toma de decisiones,». Chiclayo-Perú : s.n., 2012.
23. Taha. 2014.
24. Optimizing FPGA-based accelerator design for deep convolutional neural networks. Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., & Cong, J. February de 2015, ACM/SIGDA international symposium on field-programmable gate arrays (, págs. 161-170.
25. Xin, Yan y Taoying. 2015.
26. Bernal, Juan Manuel Anton, et al. "Planificador de rutas para recojo de desechos sólidos utilizando el algoritmo de dijkstra.". 2021, vOL. 8, PÁGS. 92-99.
27. Rossit, Vigo, Tohmé, & Frutos. 2019.
28. Cárdenas-Ferrer, Teresa Margarita, et al. "Propuesta metodológica para el sistema de gestión de los residuos sólidos urbanos en Villa Clara." . s.l. : Ecnología Química 39.2 , 2019.
29. MENESES CORTÉS, Matías Ignacio. Desclasificación basada en tipos en DART: Implementación y elaboración de herramientas de inferencia. 2018.

30. BOUKHARY, Shady y COLMENARES, Eduardo. A clean approach to flutter development through the flutter clean architecture package. En 2019 International Conference on Computational Science and Computational Intelligence. 2019, págs. 1115-1120.

31. RASK, Jonas Kjær, et al. Visual studio code vdm support. John Fitzgerald, Tomohiro Oda, and Hugo Daniel Macedo (Editors). 2021, pág. 35.

32. 2013. Paradigm, visual. visual paradigm for uml. visual paradigm for uml-uml tool for software application development, Vol. 72.

33. Miguel, Juan. 2013.

34. SALVADOR, JUAN. Academia.edu. [En línea] 20 de January de 2016. http://www.academia.edu/7894130/METODOLOGIAS_AGILES_AUP.

35. Sánchez, Tamara Rodríguez. Programa de mejora. Metodología de desarrollo para la actividad productiva de la UCI. La Lisa: s.n : s.n., 2015.

36. Alfonso Benítez, Drymon. Herramienta para generar productos de trabajos de la metodología variación AUP-UCI. s.l. : BS thesis. Universidad de las Ciencias Informáticas. Facultad 3., 2017.

37. Milián. 2011.

38. Larman, Craig. UML y patrones Introducción al análisis y diseño orientado a objetos. 2010.

39. "Ingenieria de software i-introduccion a la ingenieria del software.-especificacion de requisitos." . gutierrez terceros, mauricio ernesto. 2019.

40. Larman, Craig. UML y patrones Introducción al análisis y diseño orientado a objetos. S.L: McGraw-Hill, : s.n., 2015.

41. "Los requisitos no funcionales de software. Una estrategia para su desarrollo en el Centro de Informática Médica. Molina Hernández, Yenisel, Ailec Granda Dihigo, and Alionuska Velázquez Cintra. s.l. : Revista Cubana de Ciencias Informáticas, 2019, Vol. 13.

42. Rodriguez. 2015.

43. Lou, Tian. A comparison of Android native app architecture MVC, MVP and MVVM." . 2016. Master's Thesis, Eindhoven: Eindhoven University of Technology.

44. Pressman. Elements of Reusable Object-Oriented Software. [aut. libro] Addison Wesley. s.l. : s.l., 2010.

45. Larman. Practicas de software: Patrones GRASP. 2003.

46. "Los patrones de diseño como herramienta para guiar la práctica del profesorado.". Gros Salvat, Begoña, Anna Maria Escofet Roig, and Marta Marimon Martí. s.l. : RELATEC: revista latinoamericana de tecnología educativa, 2016.

47. Ortega, Gilberto Andrés Vargas. "Lineamientos para el diseño de aplicaciones web soportados en patrones GRASP.". s.l. : Ciencia e Ingenieria, 2021. Vol. 8.

48. Gamma. 1995.

49. Prieto, Félix. "Patrones de diseño." España: Departamento de Informática. . s.l. : Universidad de Valladolid , 2012.

50. Kenneth E. kendall, Julie kendall. Análisis y diseño de sistemas. 6ta edición. s.l. : s.n, 2016.

51. Cillero. Diagrama de despliegue,. 2017.

52. —. Diagrama de Clases del Diseño,. 2017.

53. ÁLVAREZ., DANIEL JOSÉ SALAS. Estándares de codificación Java. [En línea] 2010. [Citado el: 20 de septiembre de 2022.] Available from: http://www.aves.edu.co/ovaunicor/recursos/view/265..

54. Alvarado, Jose Manuel. Bases de datos. Unidad 2. Modelo de datos. Universidad Nacional de Ingenieria. Estelí : s.n., 2010.

55. McGraw Hill. Ingeniería del Software. Un enfoque práctico. VI Edición. 2013.

56. Procedimiento para la validacion de requisitos de software. BS thesis. Padrón Blanco, Geidys, Ailec Rodríguez Iglesias, and Diana Valdés González. 2012.

57. Sanchez. 2015.

58. BARRIENTOS, Pablo Andrés. Enfoque para pruebas de unidad basado en la generación aleatoria de objetos. 2014. Tesis Doctoral. Facultad de Informática.

59. "Procedimiento para la realización de pruebas de unidad de software orientado por objetos a nivel de clases.". Castro, Mauricio Alba. s.l. : Revista Avances en Sistemas e Informática, 2014, Vol. 8, págs. 165-175.

60. Pressman, R. Ingenieria de software : un enfoque practico. 2010.

61. Martínez. 2012.

62. Hall. 2009.

63. Pardo Matos, José Manuel, and Ailyn Febles Estrada. Proceso de Pruebas de Aceptación de Software. 2014.

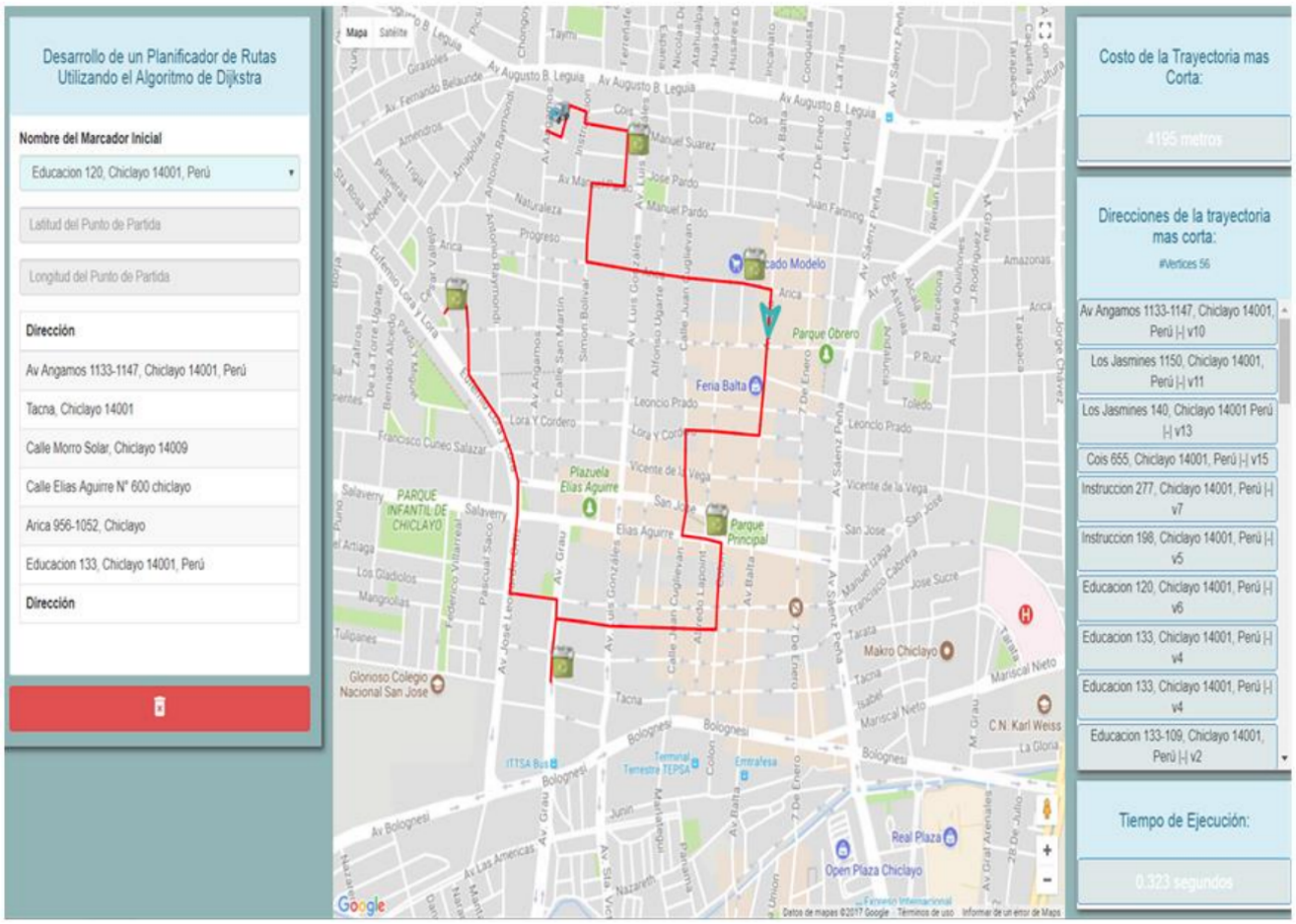
64. Zapata, J. Niveles de pruebas del software. 2013.

65. Estrategia de prueba de software. Prueba de caja negra. Ingenieria del software. 2013, Vol. Capitulo 17.

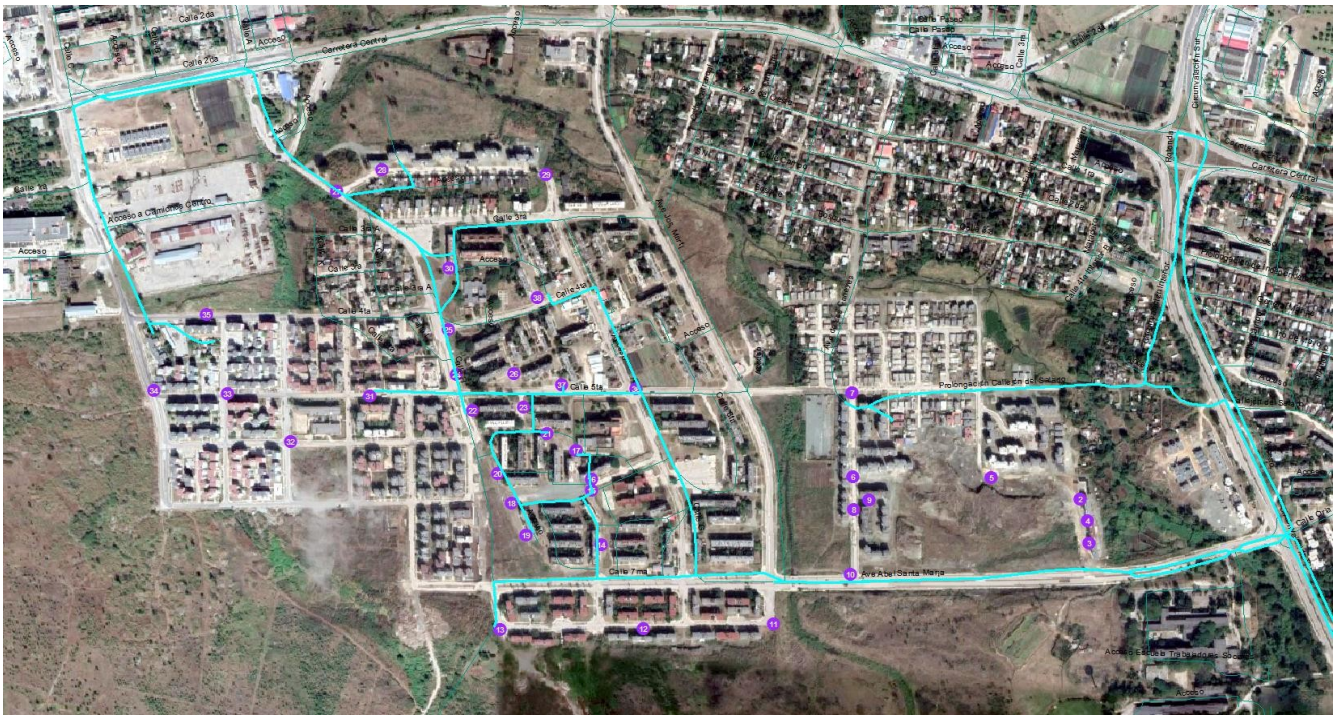
ANEXOS

Anexo 1: Sistemas homólogo (Sistema basado en mapa etiquetado para optimizar rutas

(Xin, Yan, & Taoying, 2015)

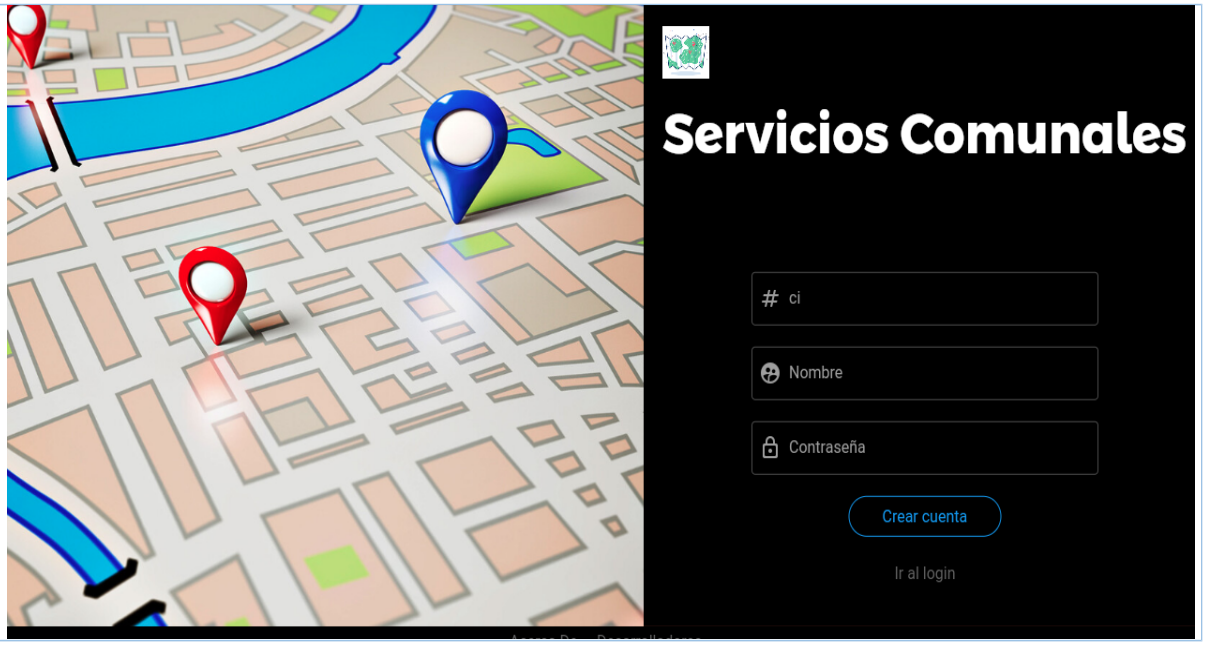


Anexo 2: Sistema homólogo (El sistema actual de gestión de los Residuos Sólidos Urbanos en la ciudad de Santa Clara)



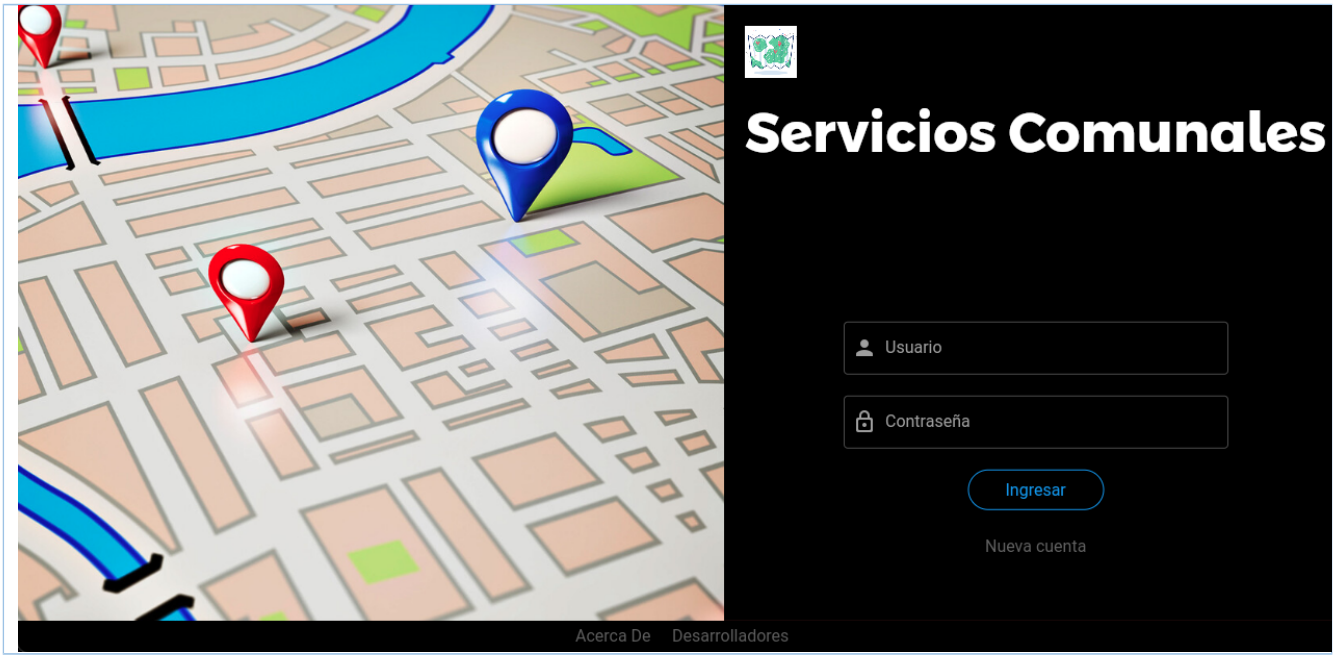
Anexo 3: Historia de usuario: Registrar usuario

Historia de Usuario	
Número: HU_2	Nombre Historia Usuario: Registrar usuario
Modificación de Historia de Usuario Número: ninguna	
Usuario: Emilio Muñoz Monterrey y Sindy Nuñez Medina	Iteración Asignada:1
Prioridad en Negocios: Media	Puntos Estimados: 2 días
Riesgo en Desarrollo: Media	Puntos Reales: 2 días
Descripción: Permite a los trabajadores del centro registrarse en el sistema	
Observaciones: - El cliente desde el que se accede debe estar conectado a la red. - El usuario debe rellenar todos los campos obligatorios para poder registrarse.	
Prototipo de interfaz:	



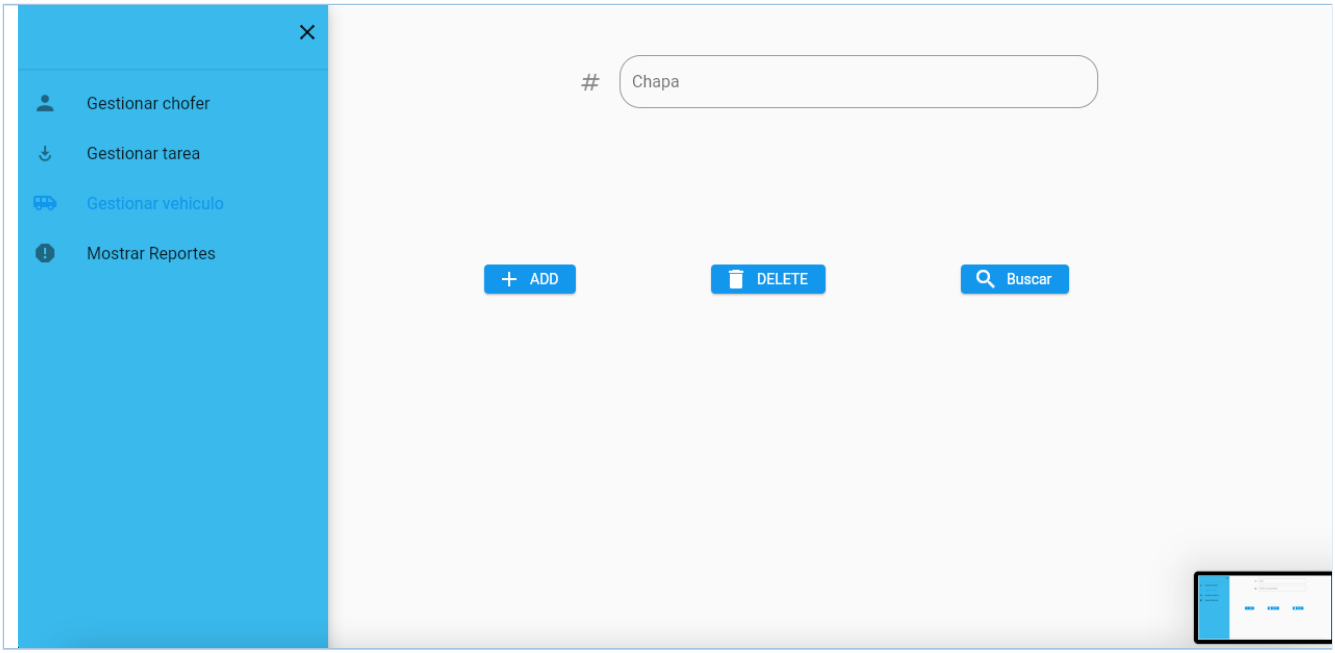
Anexo 4: Historia de usuario: Autenticar Usuario

Historia de Usuario	
Número: HU_3	Nombre Historia Usuario: Autenticar Usuario
Modificación de Historia de Usuario Número: ninguna	
Usuario: Emilio Muñoz Monterrey y Sindy Nuñez Medina	Iteración Asignada:1
Prioridad en Negocios: Muy alta	Puntos Estimados: 2 días
Riesgo en Desarrollo: Alto	Puntos Reales: 2 días
Descripción: Permite el acceso de los usuarios del centro de soporte al sistema.	
Observaciones: - El sistema debe estar iniciado. - El dispositivo desde el que se accede debe estar conectado a la red. - El usuario autenticado debe ser un usuario del sistema.	
Prototipo de interfaz:	



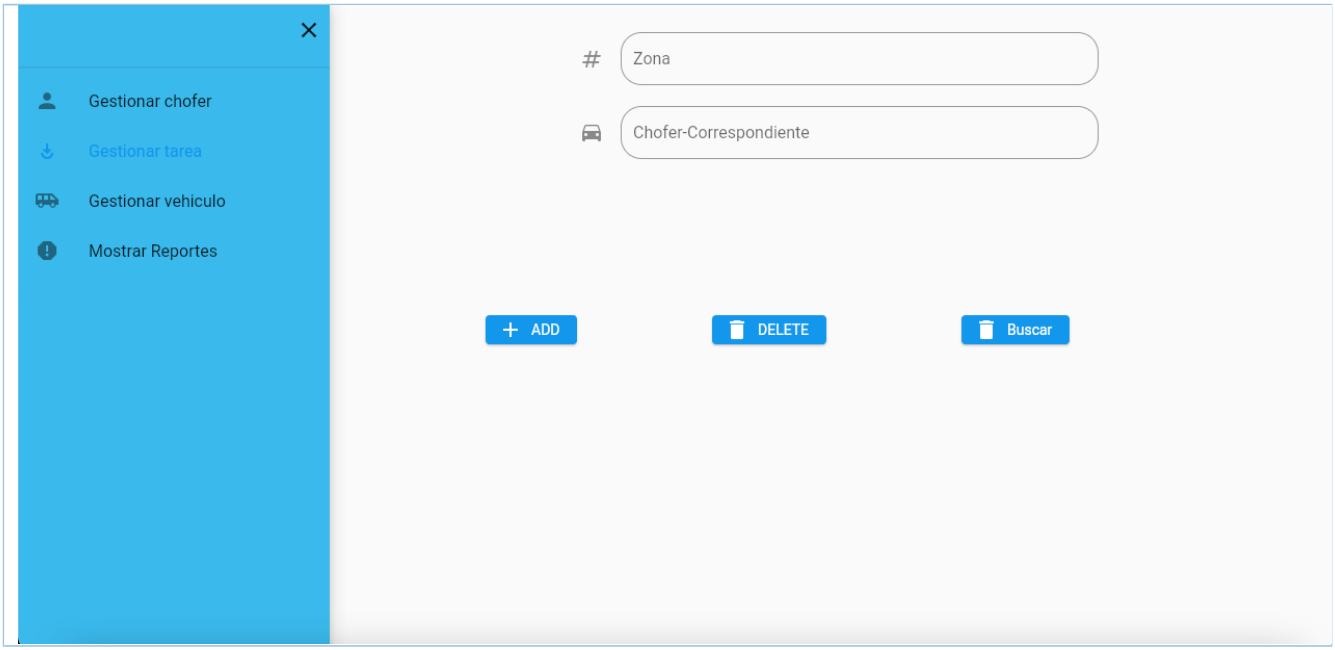
Anexo 5: Historia de usuario: Gestionar Vehículo

Historia de Usuario	
Número: HU_4	Nombre Historia Usuario: Gestionar vehículo
Modificación de Historia de Usuario Número: ninguna	
Usuario: Emilio Muñoz Monterrey y Sindy Nuñez Medina	Iteración Asignada:1
Prioridad en Negocios: Muy alta	Puntos Estimados:4 días
Riesgo en Desarrollo: Alto	Puntos Reales: 4 días
Descripción: Permite adicionar, modificar, eliminar y listar un vehículo	
Observaciones: Para gestionar un vehículo el director debe estar registrado al sistema y para modificar y eliminar un vehículo debe existir al menos un vehículo creado.	
Prototipo de interfaz:	



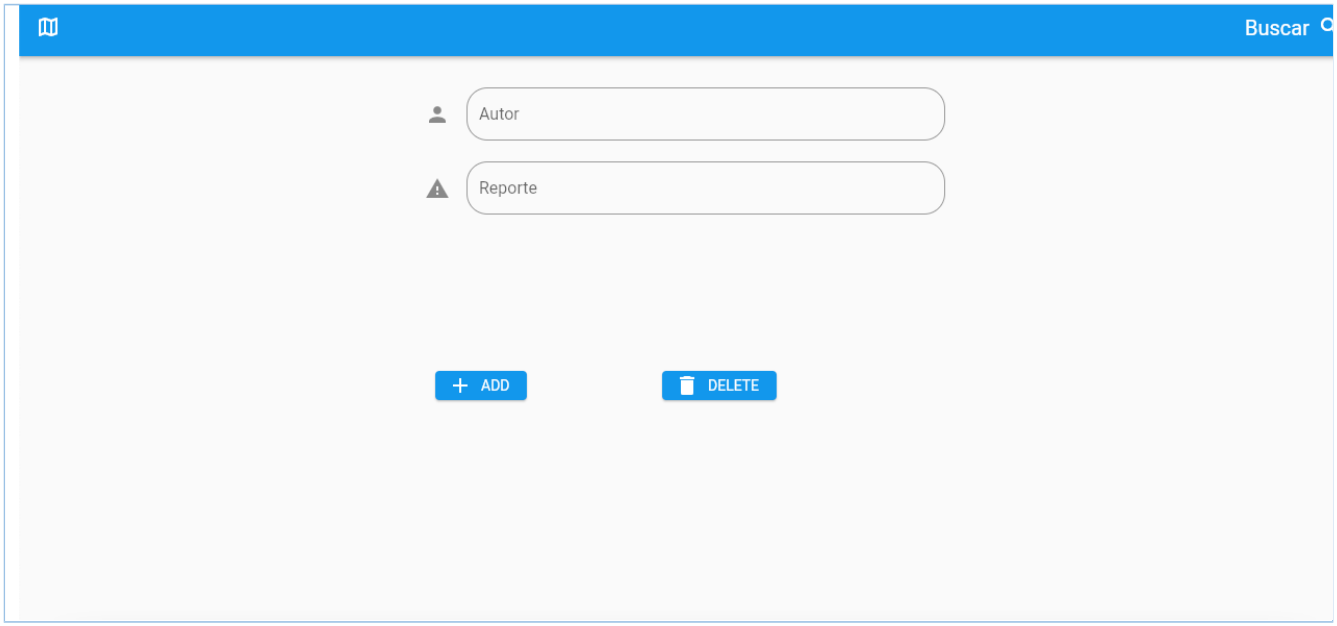
Anexo 6: Historia de usuario: Gestionar Tarea

Historia de Usuario	
Número: HU_5	Nombre Historia Usuario: Gestionar tarea
Modificación de Historia de Usuario Número: ninguna	
Usuario: Emilio Muñoz Monterrey y Sindy Nuñez Medina	Iteración Asignada:1
Prioridad en Negocios: Muy alta	Puntos Estimados:5días
Riesgo en Desarrollo: Alto	Puntos Reales: 5 días
Descripción: Permite adicionar, modificar, eliminar y listar una tarea	
Observaciones: Para gestionar una tarea el director debe estar registrado al sistema y para modificar y eliminar una tarea debe existir al menos una tarea creado.	
Prototipo de interfaz:	



Anexo 6: Historia de usuario: Gestionar Reporte

Historia de Usuario	
Número: HU_6	Nombre Historia Usuario: Gestionar reporte
Modificación de Historia de Usuario Número: ninguna	
Usuario: Emilio Muñoz Monterrey y Sindy Nuñez Medina	Iteración Asignada:1
Prioridad en Negocios: Muy alta	Puntos Estimados:6días
Riesgo en Desarrollo: Alto	Puntos Reales: 6días
Descripción: Permite adicionar, modificar, eliminar y listar un reporte	
Observaciones: Para gestionar un reporte el director debe estar registrado al sistema y para modificar y eliminar un reporte debe existir al menos un reporte creado.	
Prototipo de interfaz:	


Buscar
Autor
Reporte
ADD
DELETE