INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Asignatura:
Ingeniería de Software

Profesor:
Dorantes González Marco Antonio

GRUPO: 3CM1

Herramienta de apoyo para la administración de la línea del trolebús: “Circuito Politécnico”

Integrantes:
Barbosa Peña Xavier Maristin
Cruz Cruz Miguel Ángel
Galicia Valadez Enrique
Morales Castellanos Adolfo Erik
Sanchez Huescas Jorge Luis

Índice

Capítulo I Introducción

La línea del trolebús “Circuito Politécnico” cuenta con una longitud de operación de 11 km y 46 paradas con origen y destino en la Unidad Profesional Adolfo López Mateos del IPN (Zacatenco). Dicho transporte es utilizado en su mayoría por estudiantes del IPN, los cuales diariamente se ven beneficiados debido a la rapidez con la que este transporte recorre la unidad.

En el área de Zacatenco el transporte colectivo trolebús es el más usado, pero esto lleva una problemática de concurrencia, al dificultar el traslado de personal docente, alumnos, así como visitantes, retardando el tiempo de traslado, espera y llegada a su destino. Esto causa inasistencias, retardos y otra clase de incidentes que afectan a los usuarios.

El objetivo del trolebús es satisfacer la demanda para garantizar un servicio de calidad, adecuado a las necesidades de los usuarios además de permitir al supervisor llevar un control de las corridas y operadores activos.

Marco Conceptual

La línea Circuito Politécnico (CP) del trolebús de la ciudad de México recorre de norte a sur el Instituto Politécnico Nacional en su campus de Zacatenco sobre las avenidas Othón de Mendizábal Ote., Juan de Dios Bátiz, Manuel de Anda y Barredo, Wilfrido Massieu, Luis Enrique Erro y Eje Central Lázaro Cárdenas. Tiene por origen y destino la Unidad Profesional "Adolfo López Mateos" del IPN (Zacatenco), y presta servicio en la delegación Gustavo A. Madero, en una ruta de 11 km.
Debido al recorrido que realiza el trolebús dentro del campus Zacatenco y la necesidad que presentan los estudiantes del IPN al transportarse a sus respectivas unidades académicas ubicadas dentro del campus Zacatenco. El transporte colectivo trolebús de la línea Circuito Politécnico es el más usado por los estudiantes y personal docente. Lamentablemente el trolebús de la línea CP también se encuentra inmerso en las típicas problemáticas que se presentan en todo el transporte público de la ciudad de México.

Los asaltos a mano armada a bordo de las unidades, es muy frecuente y cada vez deja mayores pérdidas, materiales y humanas. La seguridad a bordo de las unidades es responsabilidad del gobierno de la Ciudad de México, el mismo que requiere cubrir con cámaras de seguridad y botones de pánico dentro de las unidades del trolebús.

Mala planeación los horarios de salida del trolebús, la mayoría de los estudiantes del IPN unidades Zacatenco llegan tarde a sus clases debido la deficiente y pobre

movilidad del transporte público de hoy en día, todo esto lleva una problemática de concurrencia, al dificultar el traslado de personal docente, alumnos, así como visitantes, retardando el tiempo de traslado, espera y llegada a su destino. Esto causa inasistencias, retardos y otra clase de incidentes que afectan a los usuarios.

El objetivo del trolebús es satisfacer la demanda para garantizar un servicio de calidad, adecuado a las necesidades de los usuarios además de permitir al supervisor llevar un control de las corridas y operadores activos.

Definición del problema

Ámbito

Se planea desarrollar una herramienta de apoyo para la red de transporte trolebús “Circuito Politécnico” que conste de dos aplicaciones móviles, ambas orientadas a optimizar los procesos de administración y servicio que se ofrece, la primera estará dirigida a usuarios, esta permitirá obtener la ubicación, hora de llegada de la próxima unidad, enviar alerta, queja o sugerencias.

Para la parte de la administración, la aplicación permitirá la generación de horarios para cada unidad y tener un control de hora de llegada y salida de cada unidad, así como tener registro de los boletos de los viajes y un sistema de alarma en caso de emergencia, de igual forma, se pretende monitorear la ubicación de cada unidad del transporte trolebús.

Objetivos

Objetivo general:

Diseñar una herramienta de apoyo capaz de que permita facilitar la realización de ciertas tareas que actualmente son manuales e ineficientes dentro de la organización de transporte trolebús.

Objetivos específicos:

- Obtener un control real de los tiempos de viaje en cada unidad, permitiendo transparencia en cuanto a los datos proporcionados por los supervisores.
- Sistematizar la generación de horarios y asignación de unidades a cada conductor.
- Evitar retrasos en las unidades.
- Llevar a cabo un mejor registro del boletaje vendido por cada unidad.
- Facilitar él envió de alertas en caso de una avería de la unidad o algún otro percance.
- Saber la ubicación exacta de la unidad en cuestión.

Capítulo II Estado del Arte

El análisis del estado del arte que se realiza a continuación es sobre aplicaciones móviles para mejorar el transporte en CDMX no precisamente el trolebús, debido a que no existe una aplicación específicamente que sea dirigida a este.

Déjà-bus

Herramienta para que estudiantes puedan acceder a horarios del transporte público en tiempo real, y de esta manera conocer con exactitud las horas más pertinentes para estar a tiempo en clase. Esta aplicación permite mostrar horarios y las mejores rutas del transporte público de la ciudad que están cerca de ti.

Moovit

Herramienta que permite mostrar horarios y las mejores rutas del transporte público de la ciudad que están cerca de ti. Esta aplicación recientemente se ha aliado con un startup mexicano de nombre Sin Tráfico, que básicamente posee la red de datos en tiempo real más amplia de todo el país, esta vez para integrar los horarios de prácticamente todo transporte público de la Ciudad de México, incluyendo Metro, Metrobús e inclusive la disponibilidad de las estaciones de ecobici más cercanas. La app promete también, obtener una estimación en tiempo real del estado de aglomeración en que viene tu transporte.

Aplicación Ecobici para mejorar la movilidad en la Ciudad de México.

El sistema Ecobici funciona con una aplicación que permite a los usuarios tener más control de su acceso al transporte y un uso más eficiente de su tiempo.

Por medio de una aplicación para telefonía móvil (app), la red de transporte de bicicletas Ecobici busca mejorar la movilidad en la Ciudad de México.
Lo hacen a través del Sistema de Administrador de Movilidad (SAM), una app desarrollada por el Instituto Politécnico Nacional (IPN) y el Instituto de Políticas para el Transporte y el Desarrollo (ITDP por sus siglas en inglés), que permite conocer las tendencias de movilidad de los usuarios.

Estimación de costes con el modelo constructivo de costes (COCOMO)

Cálculo LDC

Tabla 2.1 “Calculo LDC”.

Categoría	Cantidad total	Simple	Media	Compleja	Puntos de Función
Entradas	3	2x3	1x7	0x15	6+7=14
Salidas	2	0x3	1x7	1x15	7+15=22
Interacciones	2	0x3	1x7	1x15	7+15=22
Interfaces externas	2	0x3	0x7	2x15	30
Archivos internos	3	0x3	1x7	2x15	7+15=22
			Cuenta total		110

Factores de complejidad técnica

Tabla 2.2 “Factores de complejidad técnica”.

Características generales del sistema	Nivel de influencia
¿Se requiere comunicación de datos?	5
¿Existen funciones o procedimientos distribuidos?	1
¿Es crítico el rendimiento?	3
¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado? ¿Hay restricciones de plataforma?	4
¿El sistema tendrá una carga transaccional alta o baja?	4
Nivel de Disponibilidad	3

Eficiencia del Usuario Final Requerida (Usabilidad)	5
Actualización en Línea	4
Complejidad del Procesamiento	2
¿El sistema debe estar diseñado e implementado para ser reutilizable?	3
¿El sistema debe ser diseñado para ser fácil de instalar y de portar?	5
Facilidad de Uso	4
¿El sistema debe soportar múltiples instalaciones en diferentes organizaciones?	1
¿El sistema debe estar diseñado e implantado para facilitar cambios?	5
Total (N)	49

CAF=0.65+0.01*49=1.14 (Factor de ajuste de complejidad)

AFP=110*1.14= 125.4 (Puntos de función ajustados)

LDC=125.4*67= 8401 (Líneas de código)

Tabla 2.3 “Conductores de costo”.

Conductores de coste	Valoración					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extr. alto
Fiabilidad requerida del software	0,75	0,88	1.00	1,15	1,40	--
Tamaño de la base de datos	--	0,94	1.00	1,08	1.16	--
Complejidad del producto	0,70	0,85	1.00	1,15	1,30	1,65
Restricciones del tiempo de ejecución	--	--	1.00	1,11	1,30	1,66
Restricciones del almacenamiento principal	--	--	1.00	1,06	1,21	1,56
Volatilidad de la máquina virtual	--	0,87	1.00	1,15	1,30	--
Tiempo de respuesta del ordenador	--	0,87	1.00	1,07	1,15	--
Capacidad del analista	1,46	1,19	1.00	0,86	0,71	--
Experiencia en la aplicación	1,29	1,13	1.00	0,91	0,82	--
Capacidad de los programadores	1,42	1,17	1.00	0,86	0,70	--
Experiencia en S.O. utilizado	1,21	1,10	1.00	0,90	--	--
Experiencia en el lenguaje de programación	1,14	1,07	1.00	0,95	--	--
Prácticas de programación modernas	1,24	1,10	1.00	0,91	0,82	--
Utilización de herramientas software	1,24	1,10	1.00	0,91	0,83	--
Limitaciones de planificación del proyecto	1,23	1,08	1.00	1,04	1,10	--

Total=0.71

KLDC= (125.4*67) / 1000=8.40

Esfuerzo= 3.2*(8.40) ^1.05*0.71=21.22 Personas/mes

Tiempo= 2.5*(21.22) ^.38=7.98 meses

Productividad= 8401/21.22=395.9 LDC/persona al mes

Personal promedio=21.22/7.98=2.65 personas

Personal promedio=21.22/4=5.30 personas

Cronograma

Tabla 3.1 “Cronograma de actividades”.

Actividad	Febrero				Marzo				Abril				Mayo				Junio			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1.- Comunicación																				
Entrevista con el cliente																				
Identificación de requisitos																				
Análisis de los requisitos																				
2.- Plan rápido																				
Construcción de prototipo rápido (diseño de interfaces)																				
3.- Modelado del diseño rápido																				
Modelado del sistema utilizando diagramas																				
Diseño de la interfaz de usuario																				
4.- Construcción de prototipo																				
Creación de la base de datos																				
Codificación de prototipo																				
Creación de la interfaz																				
Implementación del sistema																				
5.- Desarrollo, entrega y retroalimentación																				
Recolección, validación y alimentación del sistema																				
Evaluación y prueba del sistema																				
6.- Documentación																				
Manual técnico																				
Manual de usuario																				
7.- Entrega al cliente																				
Entrega final																				

Metodología

El diseño orientado al objeto, al igual que otras metodologías de diseño orientadas a la información, crea una representación del campo del problema del mundo real y lo hace corresponder con el ámbito de la solución, que es el software.

El diseño orientado al objeto produce un diseño que interconecta objetos de datos (elementos dato) y operaciones de una forma que modularizar la información y el procesamiento; por el contrario, otros métodos dejan aparte el procesamiento.

Ventajas

- Una buena abstracción de las clases, objetos y atributos nos brinda una implementación más detallada, puntual y coherente.
- Reutilización de código: Cuando se han diseñado adecuadamente las clases, se pueden usar en distintas partes del programa y en numerosos proyectos (herencia). Lo cual permite:
    - Rápido desarrollo.
    - Alta calidad del código.
    - Bajo costo en fases de desarrollo.
- Modificabilidad: La facilidad de añadir o quitar objetos nos permite hacer modificaciones de una forma muy sencilla.
- Encapsulamiento: Nos permite proteger la integridad de los datos.
- Fiabilidad: Al dividir el problema en partes más pequeñas podemos probarlas de manera independiente y aislar mucho más fácilmente los posibles errores que puedan surgir.

Desventajas

- Curva de aprendizaje: La necesidad de utilizar bibliotecas de clases obliga a su aprendizaje y entrenamiento.
- Se hereda código que no se usa en la clase hija cuando se extiende de una clase padre.
- Tamaño del programa.

Los objetivos clave del diseño orientado al objeto son:

1.- Aumentar la productividad: Según algunos estudios, el diseño orientado al objeto logra aumentar la productividad de un desarrollo en un 20 %

2.- Incrementar calidad: Cuando hacemos referencia al término calidad no solo nos estamos refiriendo a la ausencia de errores, sino también a otros aspectos quizás no tan fáciles de medir como son la facilidad de uso, la portabilidad o la facilidad de modificación.

Modelo de procesos

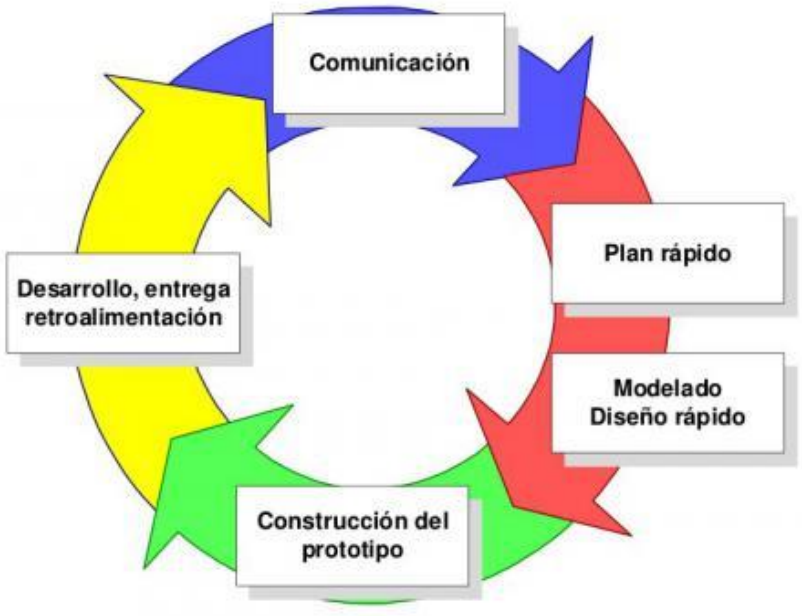
Modelo basado en Prototipos

El modelo seleccionado fue Prototipos, el cual es un modelo de desarrollo evolutivo construido en poco tiempo, al seguir este modelo se intenta tener un primer prototipo no funcional que será ajustado con base en las necesidades de los usuarios finales evaluando siempre que la aplicación sea intuitiva para el usuario.

Con el desarrollo de cada prototipo se identificarán requisitos detallados que facilitarán la interacción entre los usuarios y la aplicación, además de intentar cumplir con el principal objetivo del Sistema.

Con los prototipos los desarrolladores podrán identificar al menos desde el punto de vista del diseño los cambios y mejoras que se deban hacer a tiempo durante el desarrollo del Sistema.

Imagen 1.0 “Modelo de procesos (Prototipos”).



Capítulo III Análisis

Técnica para hallar datos

Entrevista realizada a personal del Trolebús

- ¿Cuál es el criterio para enviar un trolebús?

“Hay un programa que genera las nuevas corridas que hará un trolebús y que determina la hora en que debe salir cada unidad, se hace un estudio para sacar los horarios, para ver la demanda de que hora a qué hora se necesita el servicio. Entonces se hace un análisis de que hora a qué hora los estudiantes necesitan el servicio y luego de ahí, con base a eso, se generan horarios de servicio, se le asignan un número determinado de trolebuses. En este circuito andan 10 trolebuses, los cuales son los asignados. El tiempo del circuito son aproximadamente entre 45 y 50 minutos. Una vez mandados los 10 camiones, el intervalo es de 8 a 10 minutos. Durante este tiempo, debe de salir cada trolebús uno tras otro y obviamente tenemos operadores para los 10 turnos. Solo en esta línea trabajamos de lunes a viernes porque son los días que hay clases y hay dos turnos. Todas las unidades cuentan con 2 operadores cada día, desde las 6 de la mañana hasta las 11 de la noche”.

- ¿Cada operador se queda en un mismo trolebús?

“Puede variar el trolebús, pero si uno en un turno y otro en otro turno, pero son variables los trolebuses, nunca son fijos. Por lo que puede cambiar diario”.

- ¿Cuentan con la estadística de cuántos pasajeros tienen diario y en qué horarios?

“Por el promedio global, tenemos la venta diaria de cuánto recauda la unidad. Estos son datos que maneja la caja, sin embargo, tratamos de respetar el horario e itinerario que ya trae la unidad, el cual es el horario que maneja cada uno. Por las necesidades de este, nosotros hacemos regresos, lo que quiere decir que los sacamos de horario para poder atender la demanda.
Hay horarios pico aquí, entonces en la mañana lo que hacemos es crear el horario y metemos a toda la flota. O sea, no hace el recorrido completo, si no hace circuitos cortos, todo ingreso. Ahorita en la tarde nuevamente rompemos el esquema del horario porque, otra vez, en el segundo turno hay mucho ingreso, y en la noche es al revés: sacamos. Pero durante esos horarios, que para nosotros sería uno bastante atareado, respetamos el ya establecido. Después de que termine dicho horario pico, vemos dónde deben estar acomodadas las unidades de acuerdo con su itinerario para así continuar. Un trolebús en promedio da 8 vueltas, sin embargo, puede llegar a dar

más. Normalmente lo que marca el horario. El tiempo y el intervalo se calculan de acuerdo con el recorrido que tenemos y al número de unidades”.

- ¿Han llegado a tener varios percances con alguna falla en el trolebús? ¿Qué tan seguido es eso?

“Tenemos una incidencia alta de autobuses que están fallando. Alrededor de un 40% de unidades que presentan problemas. Aun así, por ahí ya hemos implementado varias acciones: al principio las unidades salían desde aquí y se iban hasta el depósito para ser reparadas y regresar, lo que era restar una unidad aquí; ahora tenemos el corredor cero emisiones en la terminal del norte, eje central. Esa unidad se manda ahí, que es menos el tiempo de recorrido, se repara y regresa. Cuando tiene una eventualidad el jefe de terminal, y por diversos motivos hay varias unidades, este método apenas lo implementamos ya que tuvimos una reunión con las autoridades de aquí. Por lo mismo que de alguna manera a veces no fallan varias unidades y nos llamaron la atención, tenemos el apoyo de eje central. Supongamos que el jefe de terminal dice “requiero de una, dos o tres unidades porque me estoy quedando sin ellas”, entonces se va y piden los camiones, vienen a hacer servicio y cuando se reparan las unidades hacemos el cambio. Posteriormente la integramos ahí en el circuito y esas se regresan al eje central. Por operación y para no dejar descubierto el circuito, le pedimos apoyo al eje central o a otra línea, para mantener siempre la flota de 10 unidades mínimas”.

“Hay ocasiones que el jefe de terminal, cuando es inicio de clases y hay mucho estudiante de nuevo ingreso, a veces nos rebasa y él puede hablar al norte para pedir que le manden dos aumentos. Así es como tratamos de mantener la operación ahí”.

“Aunque ya está establecida por un horario, el operador tiene uno. Con eso, ellos checan su papeleta y ahí vamos anotando todas las vueltas que dan de acuerdo con mismo. Cuando hay un desajuste por la eventualidad con la que tienen que mandarlos para sacar (en este caso las 2 de la tarde), no hacemos todo el circuito, si no que mandamos a meter con el fin de desajustarlo, pero a una vuelta tenemos que ajustar”.

- ¿Cuándo llega a fallar una unidad a medio camino tienen que reportarlo a los supervisores?

“Tenemos la dificultad de que los operadores se pueden quedar si no nos reportan”.

- ¿Cómo funciona la papeleta?

Un trolebús sale del depósito 5:36am a su primera vuelta, la cual es al metro “Politécnico”. Debe de llegar 6:46am, ya que es a la hora que yo lo empiezo a firmar, que es el mismo boletaje, y ahí es cuando ya empezamos la siguiente vuelta: 7:24am.

“Posteriormente, él está llegando 8:05am, y así sucesivamente. Yo le voy a poner en qué llegó, no en el horario que está programado porque en la mañana son puros rehiletes. Igual ahorita a las 2:00pm son puros rehiletes. El horario lo quebrantamos tantito en lo que se acomoda y para meter a la gente. Si te das cuenta, son 8 vueltas, entonces el día rinde 14:43pm en el metro “Politécnico”, es un programado de 8 a 9 vueltas”.

- ¿Les facilitaría tener la papeleta en digital?

“Claro, sería mucho más práctico. Con el mismo formato o puede ser variable”.

- ¿A dónde se debe de mandar la información de la papeleta?

“Sí sería importante, pero aquí nosotros entregamos la información de la papeleta y ya en la caja hacen todo un procedimiento. A mí lo que más me facilita es, por ejemplo, saber dónde las unidades andan en el circuito y decir “¿Sabes qué? ésta la mande y ya se quedó en este punto, no se mueve y el mismo puede hablar porque el operador se puede quedar ahí y no reportarse”. Se queda descompuesto y hasta la hora que quiere puede reportarse. Entonces él sí tiene el control de decir “¿Sabes qué? Ya no se mueve de este punto”.

- Como director de operación ¿aceptaría que los operadores tuvieran que instalar la aplicación y se les obligará a usar el GPS? Por ejemplo, como Waze, que a fuerzas tienes que saber su ubicación.

“Mira, ahí hay un tema: nosotros siempre hemos consignado a los operadores por usar el celular. Es un tema que hay que tocarlo porque si yo tengo una política de cero celulares y ahora le voy a poner un celular, entonces tendríamos un problema”.

“A mi si me gustaría hacerlo. No sé de qué forma poner una alerta donde sepamos que le estamos mandando mensaje y decir “Sabemos que estas parado”. Sería mejor para nosotros, porque qué tal si algún día él va hablando por teléfono y puede decir “Me habló el regulador y pues yo no tuve la culpa”, o puede tener un accidente. Por esto, nosotros tratamos de que ellos no utilicen el celular”.

“Entonces sería más físico la señal. Es decir, mandar una alerta y decir “¿Sabes qué? Te reportas o ya haríamos el lenguaje de cómo podríamos comunicarnos con la unidad, sin tener que estar con el celular, y que mande una alerta y haber encendido algo en el trolebús”. Eso sí podemos instalarlo en los camiones; un foco o algo, una luz”.

- La idea principal era como activará su GPS y sobre eso que trabajará la aplicación, sin necesidad de que utilizarán el celular.

“Nosotros tenemos algo así en los taxis eléctricos: ellos sí tienen celular, pero cuando no quieren que sean localizados ¿qué hacen?, lo apagan. Igual así los taxis eléctricos. Sí los traemos monitoreados, pero a veces, cuando se escapan, hay operadores que los mandamos a un servicio, desconectan el GPS y andan haciendo locales: Lo mandamos a Santa Fe, desconecta el GPS. Ya hizo dos o tres locales, se conecta y regresa”.

- ¿Los choferes tiene un número de Id para reconocerlos?

“Si. Todos los que trabajamos en transportes eléctricos nos identifican por un número de expediente”.

- Entonces ¿El GPS lo traen en su celular los conductores de los taxis eléctricos?

“Ellos tienen un celular igual que nosotros. Estos nos lo prestan el organismo. Son del organismo. Entonces ellos se comunican, tienen derecho a revisar, ir por rutas, todo. Siempre deben de hacer los reportes de dónde están, su ubicación. Terminan el servicio, si tienen otro asignado. Pero luego se me pierden; se desconectan y quitan los datos”.

- ¿Qué tanto encarece el servicio que les dieran celular a los radio taxis?

“No lo encarece porque esto nos lo da el organismo. Yo sé que hay aplicaciones donde a lo mejor van a tener un centro de recepción y ya podríamos monitorear a la unidad, pero eso ya es otra tecnología. Esto lo adaptamos porque sí hubo una propuesta de una empresa de ponerle GPS a la unidad, cámaras y todo ese rollo, pero ahí se nos fue por los costos. Como este nos lo paga gobierno, a cada uno le das uno, nosotros lo podemos controlar. Ya cuando no responde en cierto tiempo pues se le hace una amonestación”.

“Pero sí los traemos monitoreados. Esos los tenemos conectados al C5, eso sería muy importante para nosotros: tener una comunicación al C5 y un botón de pánico. ¿Por qué?, porque cuando hay un incidente, yo lo doy de alta en seguridad pública y seguridad pública rápidamente le toma la llamada. Así están nuestros taxis. Cuando un operador es asaltado, nos dieron una clave, un número de juego de botones para tocar, rápidamente se dispara la alerta a C5, y el C5 lo que hace es ubicarnos y de inmediato te llega apoyo. Lo que podríamos hacer es una tecnología similar con bajo costo, donde la señal de botón de alarma le avisa a C5 que hay un problema y de inmediato monitoree sus cámaras y llegue rápido en resumen un botón de pánico”.

- ¿Regularmente han sufrido muchos asaltos en el trolebús en esta línea?

“Normalmente no, pero últimamente se ha disparado. De hecho, hace 8 días tuvimos una alerta, pero por parte del operador: hablo “Me acaban de asaltar”, sin embargo,

la alerta ya fue muy tardía. Antes no nos asaltaban mucho porque somos un transporte rígido. Un transporte rígido es aquel que no lo puedes sacar de la ruta. Lo que hacen con un microbús, cuando lo asaltan lo desvían y lo asaltan”.

“A nosotros no nos hacían eso, pero últimamente bajo puente por ejemplo en Garibaldi, aunque sea en la misma línea pues si nos pasa, de hecho, nos mandaron varias fotos de asaltantes de la zona para hablar con los operadores y ver si reconocían a estas personas”.

---“El celular podría bloquearse para que tenga el GPS activado todo el tiempo*propuesta de nuestra parte, aceptada por el jefe de terminal*¿Aunque lo apagues? Si, estaría bloqueado”.

- ¿Tiene alguna aplicación para hacer los reportes?

“No, ahorita todo es a mano. Pero, sí que utilizamos un sistema ingles con las papeletas y no existe nada ahorita. Lo importante sería migrar a lo digital. Obviamente el gobierno no lo ha hecho, porque en estos momentos dependemos mucho del operador; si se reporta o no se reporta, si está malo o no está malo. La verdad hay fallas reales y fallas que no existen”.

Reglas de negocio

Tabla 4.1 “Descripción de las reglas del negocio”.

ID	Descripción
RDN1	Se utilizará el celular proporcionado por el sindicato.
RDN2	El conductor podrá hacer uso del celular si, y sólo si existe un accidente con la unidad, o emergencia como un asalto.
RDN3	El conductor de la unidad deberá poner al inicio de su turno el número de serie del boletaje y al final de su turno el último en el que se quedó.
RDN4	El conductor no podrá modificar el horario y unidad una vez asignados.
RDN5	El conductor deberá darse de alta para poder ver su horario generado y utilizar las demás opciones de la interfaz.
RDN6	El supervisor deberá darse de alta para poder ver los horarios y corroborar los datos en las opciones de la interfaz.
RDN7	El conductor en caso de emergencia solo podrá llamar a números registrados que pertenezcan a su supervisor directo.

RDN8	El supervisor podrá visualizar la información de horarios y confirmar si se cumplió con el horario de cada unidad.

Análisis de requerimientos

Requerimientos funcionales

Tabla 5.1 "Listado y descripción de los requerimientos funcionales".

ID	Descripción
RF1	Los empleados se registran en el sistema dependiendo de su cargo (Conductor o Supervisor).
RF2	Se deben registrar a los conductores guardando sus datos como: nombre, unidad que maneja, ID, etc.
RF3	Por medio de los tickets se hará una estimación de demanda del transporte, para identificar los horarios de mayor concurrencia.
RF4	La aplicación permitirá al supervisor visualizar la ubicación de los trolebuses que se encuentran dentro de la ruta Circuito Politécnico.
RF5	La aplicación deberá almacenar la hora de partida y de llegada de las unidades.
RF6	Se generará automáticamente la asignación de unidades a los operadores.
RF7	La aplicación deberá mandar una alerta en caso de un incidente (accidente, falla del servicio, intento de robo).

Requerimientos no funcionales

Tabla 5.2 "Listado y descripción de los requerimientos no funcionales".

ID	Descripción
RNF1	Se desarrollarán 2 aplicaciones móviles exclusivas para Android 7 Nougat.
RNF2	Se utilizará Java y el gestor de base de datos Firebase para el desarrollo de las interfaces.
RNF3	Las aplicaciones no deben tener un tiempo de respuesta mayor a 10 segundos en todas sus transacciones.

RNF4	Deben tener una conexión a internet mínima de 1 MB.
RNF5	Deben cifrar los datos sensibles antes de enviarlos

Gestión de riesgos

Tabla 5.3 “Gestión de riesgos”.

ID	RIESGO	CATEGORÍA	PROBABILIDAD	IMPACTO
01	Falta de conocimientos de la tecnología de desarrollo.	Riesgo técnico	70%	1
02	Planificación demasiado optimista.	Riesgo de gestión del proyecto	55%	2
03	Falta de capacidades por parte del usuario al manejar el software.	Riesgo externo	65%	3
04	Cambios de requerimientos.	Riesgo técnico	60%	3
05	Compatibilidad.	Riesgo técnico	10%	4

Capítulo IV Diseño

Diagramas

Diagrama de casos de uso

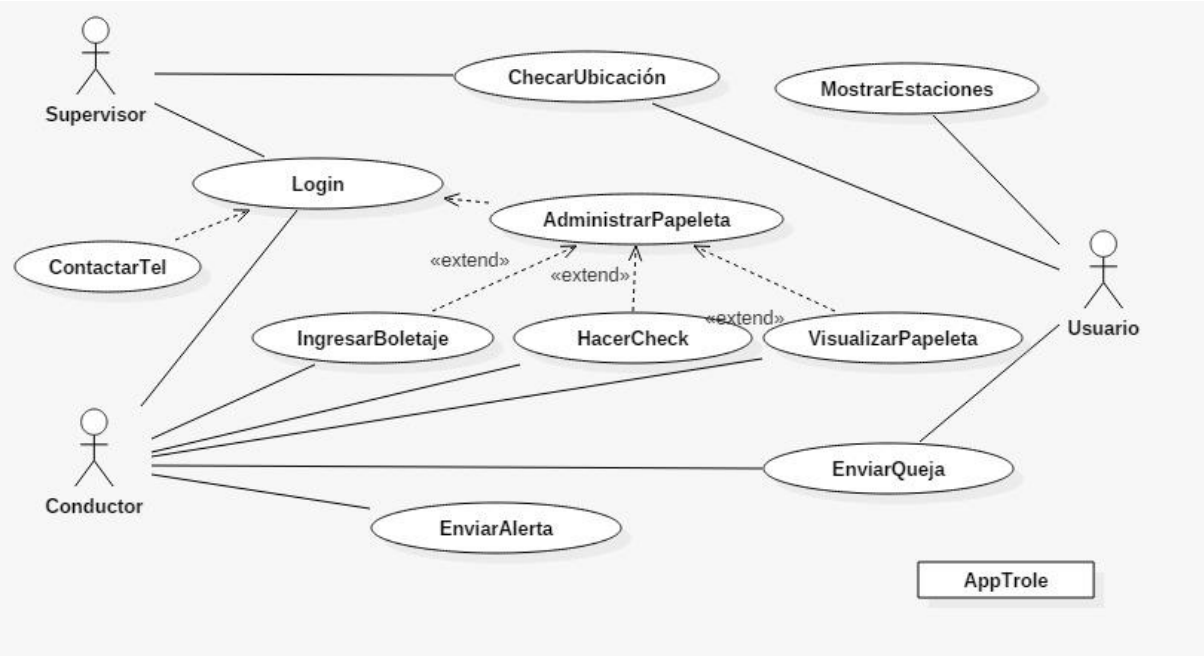


Imagen 2.1 “Diagrama de casos de uso”.

Casos de uso especificados

Tabla 6.1 “Descripción del caso de uso Checar ubicación”.

Identificador		CU1	
Nombre		Checar ubicación	
Descripción		El sistema enviará la ubicación actual de la unidad más cercana a la persona que se lo solicite.	
Actor Principal		Supervisor, Usuario.	
Responsable		Enrique Galicia Valadez.	
Revisión		Miguel Ángel Cruz Cruz.	
Fecha de creación	05/05/2019	**Fecha de modificación**	
Precondiciones			
Que existan unidades circulando.			
Postcondiciones			
La ubicación de la unidad más cercana será mostrada			
Trayectoria Principal			
1. El supervisor desea conocer la ubicación de alguna unidad. 2. El supervisor ingresa a la aplicación e inicia sesión 3. Se muestra la interfaz gráfica de inicio 4. El supervisor selecciona la opción de “Visualizar ubicación” 5. El sistema muestra la ubicación de las unidades			
Trayectorias Alternativas			
Trayectoria alternativa A: El usuario quiere conocer la ubicación A.1 El usuario desea conocer la ubicación de alguna unidad A.2 El usuario ingresa a la aplicación A.3 El sistema muestra la ubicación de las unidades			

Tabla 6.2 “Descripción del caso de uso Administrar papeleta ”.

Identificador		CU2	
Nombre		Administrar papeleta	
Descripción		El supervisor podrá generar horarios para cada unidad de trolebús, asignando un chofer y un horario de llegada aproximado.	
Actor Principal		Supervisor	
Responsable		Barbosa Peña Xavier Maristin	
Revisión		Galicia Valdez Enrique	
Fecha de creación	05/05/2019	**Fecha de modificación**	
Precondiciones			
El supervisor deberá estar logueado y con una conexión a internet.			
Postcondiciones			

Cada unidad tendrá un horario de llegada/salida con un chofer asignado para conducir.
Trayectoria Principal
1.- El supervisor desea generar un horario para una unidad específica. 2.- El supervisor inicia sesión en la aplicación. 3.- Se muestra la información del usuario en la interfaz principal de la aplicación. 4.- El supervisor selecciona “Administrar papeleta” del menú de opciones. 5.- El supervisor podrá generar o modificar un nuevo horario. 6.- El supervisor presiona el botón actualizar o en su defecto agregar horario. 7.- El sistema modifica la base de datos. 8.- Los cambios se reflejan automáticamente en las aplicaciones de los choferes.
Trayectorias Alternativas
A.4 El supervisor selecciona “modificar horario” del menú. A.5 El supervisor puede modificar o agregar un chofer al horario.

Tabla 6.3 “Descripción del caso de uso Visualizar papeleta”.

Identificador		CU3	
Nombre		Visualizar papeleta	
Descripción		En la papeleta, el chofer podrá visualizar su horario de salida y horario de llegada aproximado.	
Actor Principal		Chofer	
Responsable		Morales Castellanos Adolfo Erik	
Revisión		Sanchez Huescas Jorge Luis	
Fecha de creación	05/05/2019	**Fecha de modificación**	
Precondiciones			
El chofer deberá estar logueado y con una conexión a internet.			
Postcondiciones			
El chofer podrá ver su papeleta con éxito.			
Trayectoria Principal			
1.- El chofer desea ver su horario o papeleta. 2.- El chofer accede al menú principal. 3.- El chofer selecciona la opción “Visualizar papeleta”. 4.- El sistema despliega en pantalla la papeleta donde podrá visualizar su horario y otros datos sensibles.			
Trayectorias Alternativas			
A.1 El chofer podrá visualizar en la pantalla principal si tiene un horario asignado, pero no toda su papeleta.			

Tabla 6.4 “Descripción del caso de uso Contactar teléfono ”.

Identificador		CU4	
Nombre		Contactar teléfono	
Descripción		Contactar directamente al teléfono del supervisor en curso o a algún servicio en particular.	
Actor Principal		Conductor, Supervisor	
Responsable		Barbosa Peña Xavier Maristin	
Revisión		Morales Castellanos Adolfo Erik	
Fecha de creación	05/05/2019	**Fecha de modificación**	
Precondiciones			
Conductor logueado El conductor no debe de estar manejando Supervisor logueado			
Postcondiciones			
El supervisor recibirá la llamada por parte del conductor			
Trayectoria Principal			
1. El conductor desea contactar con el supervisor 2. El conductor selecciona la opción de “llamar supervisor” 3. El sistema realiza la llamada al supervisor 4. La hora de llamada se almacena en la base de datos			
Trayectorias Alternativas			

Tabla 6.5 “Descripción del caso de uso Ingresar boletaje”.

Identificador		CU5	
Nombre		Ingresar boletaje	
Descripción		Se le asigna cierto número de boletos al chofer al salir a cada viaje por el circuito que será registrado en el sistema.	
Actor Principal		Supervisor	
Responsable		Morales Castellanos Adolfo Erik	
Revisión		Sanchez Huescas Jorge Luis	
Fecha de creación	05/05/2019	**Fecha de modificación**	
Precondiciones			
El supervisor deberá estar logueado y con conexión a internet.			
Postcondiciones			
El supervisor ingreso de manera adecuada la cantidad de boletos asignados a cada chofer y tendrá un estimado de boletos al final de cada ruta.			
Trayectoria Principal			
1.- El supervisor desea agregar boletaje a una unidad. 2.- El supervisor selecciona una papeleta del chofer.			

3.- El supervisor selecciona editar la papeleta. 4.- El supervisor scrollea hasta parte del boletaje. 5.- El supervisor ingresa el rango de boletos que se le asigna a la unidad 6.- El supervisor seleccionar guardar papeleta para generar los cambios en la base de datos. 7.- El sistema actualiza los datos de la base de datos.
Trayectorias Alternativas

Tabla 6.6 “Descripción del caso de uso Mostrar estaciones ”.

Identificador		CU6	
Nombre		Mostrar estaciones	
Descripción		Mostrar todas las estaciones del circuito del trolebús	
Actor Principal		Usuario	
Responsable		Cruz Cruz Miguel Ángel	
Revisión		Morales Castellanos Adolfo Erik	
Fecha de creación	05/05/2019	**Fecha de modificación**	
Precondiciones			
Las estaciones estén registradas en el sistema			
Postcondiciones			
Las estaciones serán mostradas en pantalla			
Trayectoria Principal			
1. El usuario desea ver las estaciones que recorre el trolebús 2. El usuario ingresa a la aplicación 3. El sistema muestra la ubicación de las estaciones			
Trayectorias Alternativas			

Tabla 6.1 “Descripción del caso de uso Enviar queja ”.

Identificador		CU7	
Nombre		Enviar queja	
Descripción		Se podrá enviar una queja en caso de que exista alguna inconformidad durante el trayecto.	
Actor Principal		Usuario, Conductor	
Responsable		Sanchez Huescas Jorge Luis	
Revisión		Morales Castellanos Adolfo Erik	
Fecha de creación	05/05/2019	**Fecha de modificación**	
Precondiciones			
Conductor logeado El conductor no debe de estar manejando			

Postcondiciones
El supervisor recibirá la queja
Trayectoria Principal
1. El conductor desea enviar una queja 2. El conductor selecciona la opción de “enviar queja” 3. El conductor escribe su queja y envía el mensaje 4. La queja se almacena en la base de datos
Trayectorias Alternativas
A.1. El usuario desea enviar una queja A.2. El usuario selecciona la opción de “enviar queja” A.3. El usuario escribe su queja y envía el mensaje

Tabla 6.7 “Descripción del caso de uso Enviar alerta ”.

Identificador		CU8	
Nombre		Enviar alerta	
Descripción		Enviar un mensaje de alerta en caso de emergencia	
Actor Principal		Conductor	
Responsable		Galicia Valdez Enrique	
Revisión		Barbosa Peña Xavier Maristin	
Fecha de creación	05/05/2019	**Fecha de modificación**	
Precondiciones			
El conductor tiene una unidad asignada. El conductor necesita mandar mensaje de alerta.			
Postcondiciones			
El supervisor recibirá el mensaje de alerta.			
Trayectoria Principal			
1. El conductor desea mandar una alerta 2. El conductor selecciona la opción “Enviar alerta” 3. El supervisor recibe la alerta 4. La alerta se almacena en la base de datos			
Trayectorias Alternativas			

Diagrama de secuencia

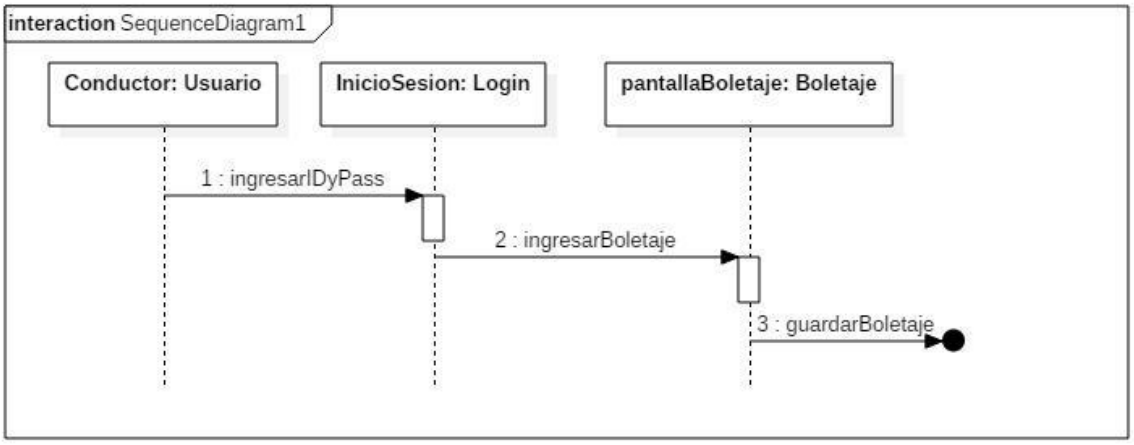


Imagen 3.1 “Diagrama de secuencia (Iniciar Sesión)”.

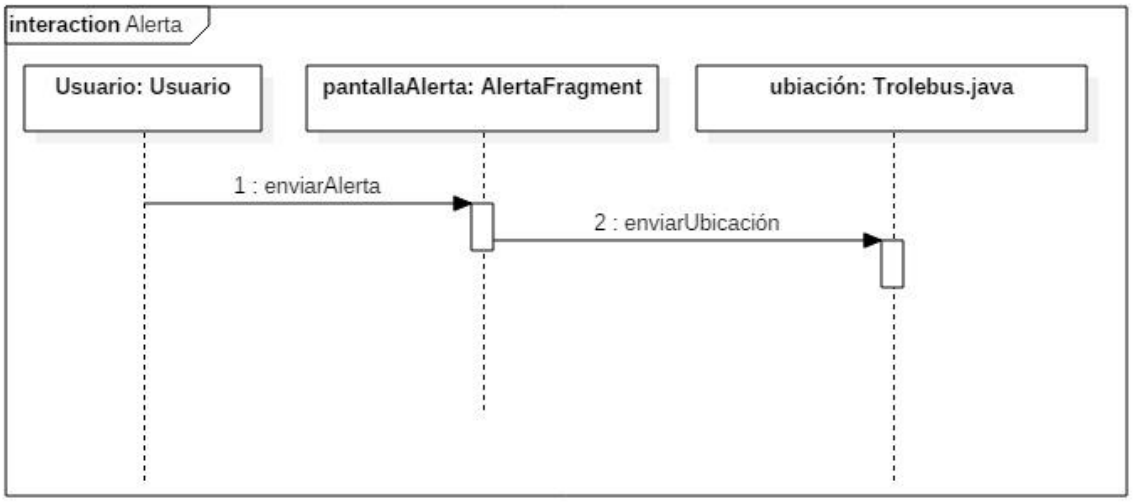


Imagen 3.2 “Diagrama de secuencia (Enviar Alerta)”.

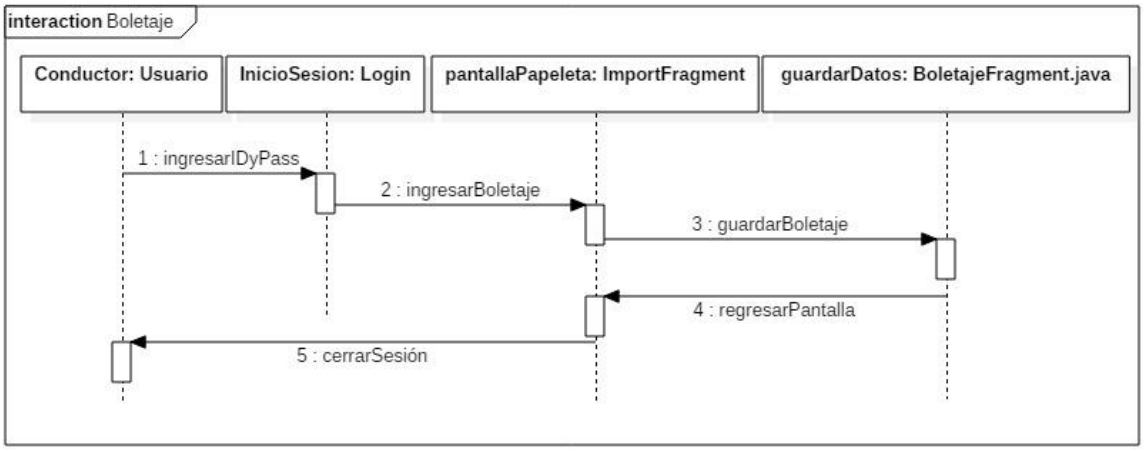


Imagen 3.3 “Diagrama de secuencia (Ingresar Boletaje)”.

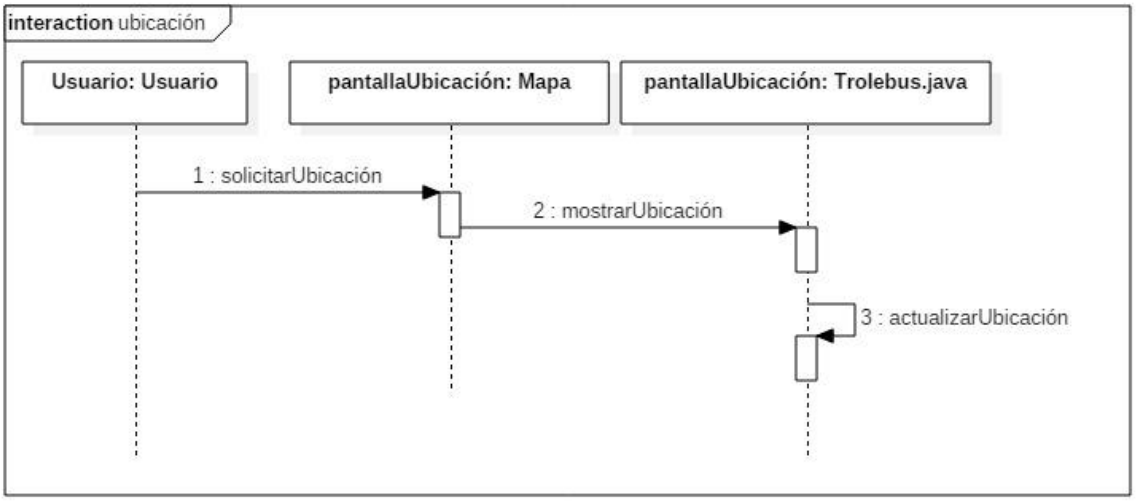


Imagen 3.4 “Diagrama de secuencia (Mostrar ubicación)”.

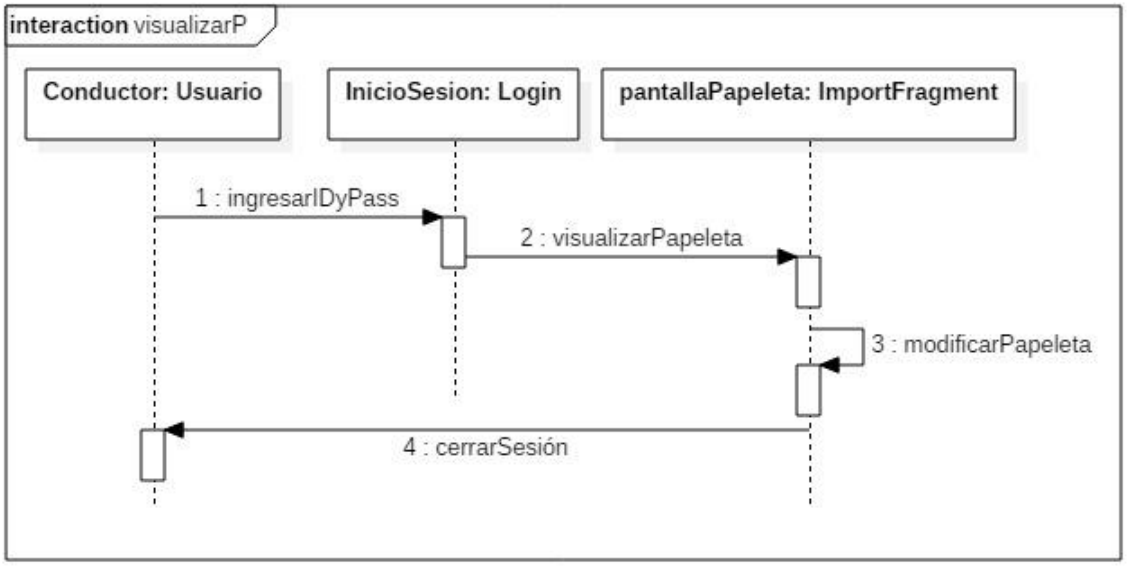


Imagen 3.5 “Diagrama de secuencia (Visualizar papeleta)”.

Diagrama de clases

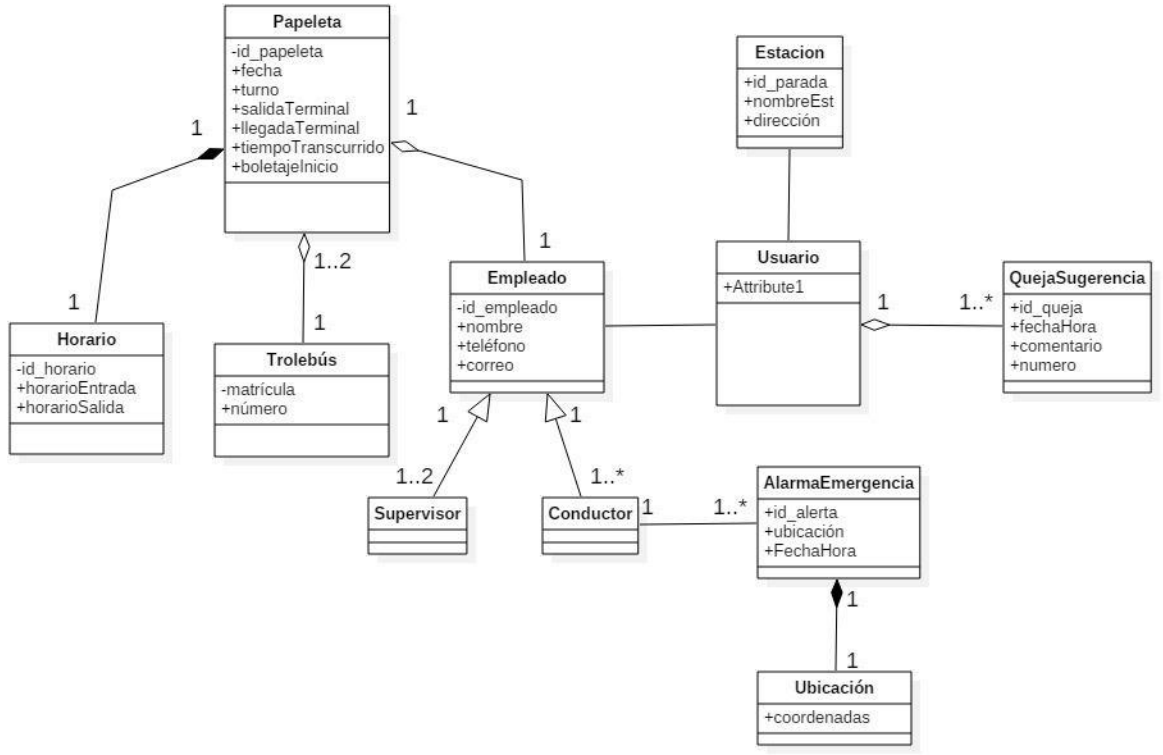


Imagen 4.0 “Diagrama de clases”.

Diagrama de estados

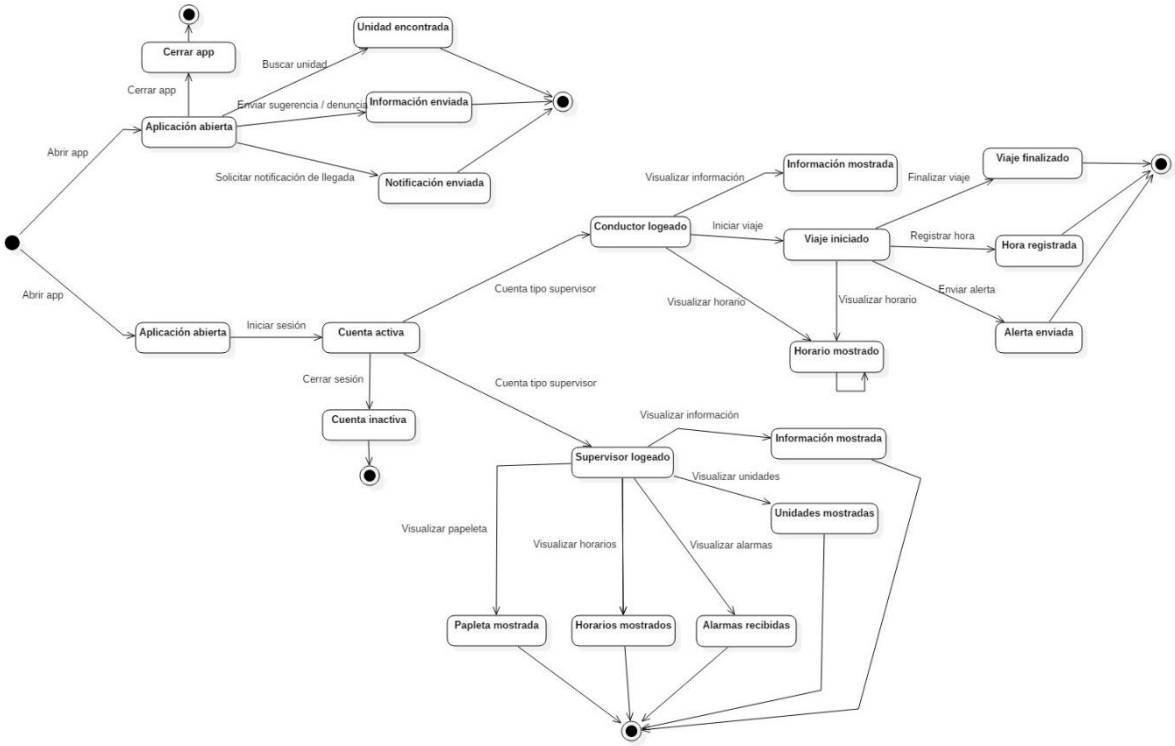


Imagen 5.0 “Diagrama de estados”.

Diagrama de actividades

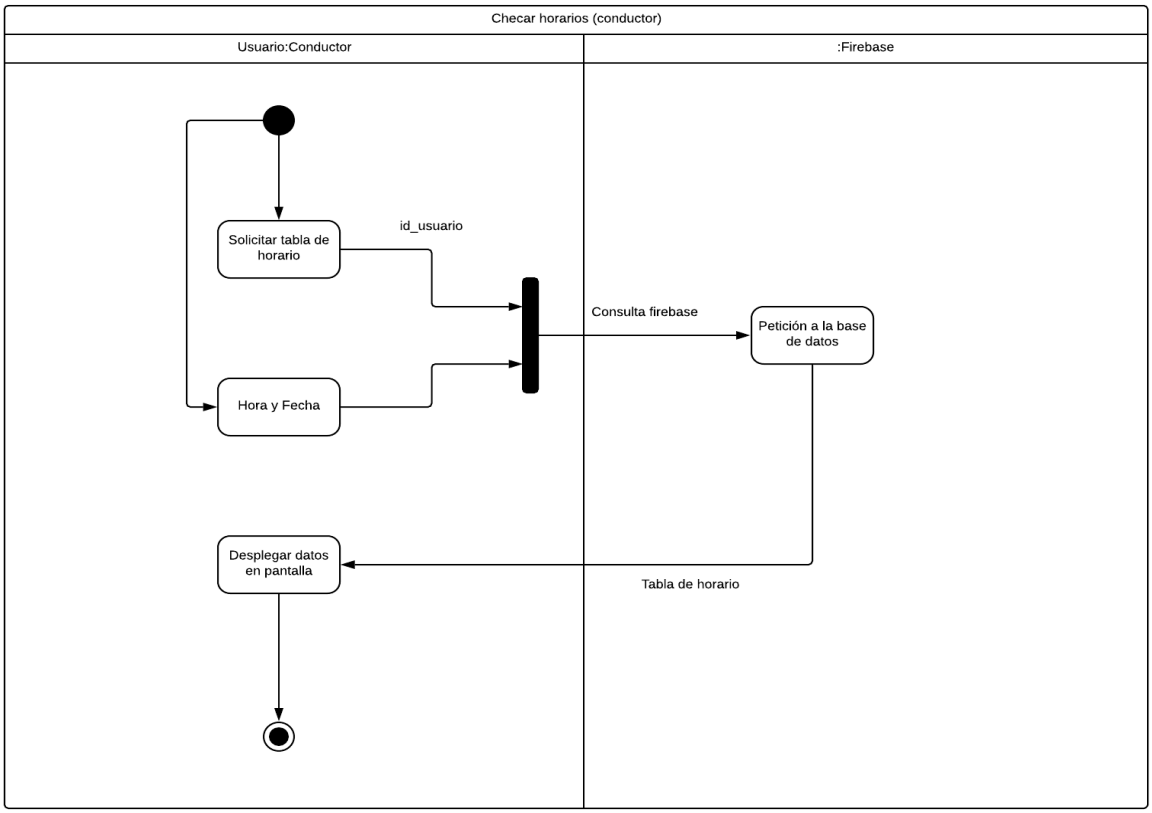


Imagen 6.1 “Diagrama de actividades (Checar horarios-conductor)”.

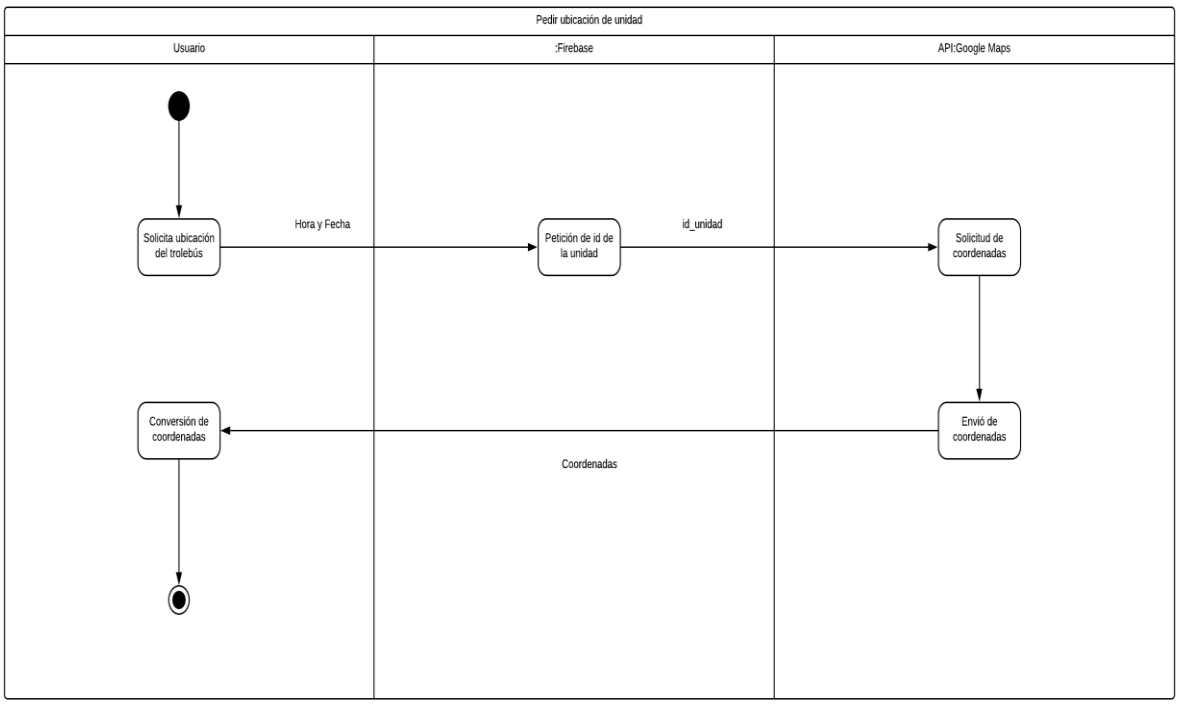


Imagen 6.2 “Diagrama de actividades (Pedir ubicación de unidad)”.

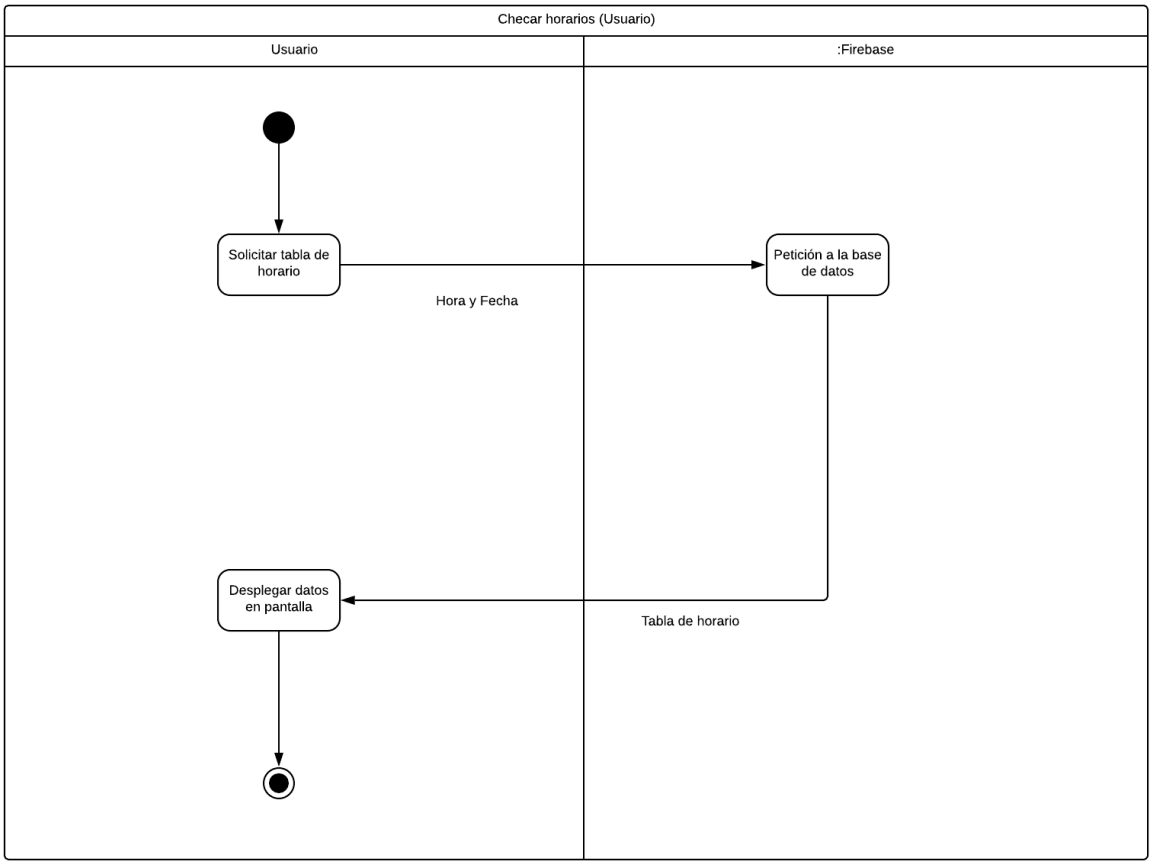


Imagen 6.3 “Diagrama de actividades (Checar horarios-usuario)”.

Capítulo V Codificación

Aplicación (Chofer)

Codificación del envió de alertas (Alerta.java y AlertaFragment.java)

```
1.  package com.example.myapplication;
2.
3.  public class Alerta {
4.
5.      double Lat, Lon;
6.      String fecha;
7.      int id;
8.
9.  }
10. //AlertaFragment
11. package com.example.myapplication;
12. import android.content.Intent;
13. import android.net.Uri;
14. import android.os.Bundle;
15. import android.support.annotation.NonNull;
16. import android.support.v4.app.Fragment;
17. import android.text.method.BaseKeyListener;
18. import android.view.LayoutInflater;
19. import android.view.View;
20. import android.view.ViewGroup;
21. import android.widget.ArrayAdapter;
22. import android.widget.Button;
23. import android.widget.ListView;
24. import android.widget.TextView;
25. import android.widget.Toast;
26.
27. import com.google.firebase.database.DataSnapshot;
28. import com.google.firebase.database.DatabaseError;
29. import com.google.firebase.database.DatabaseReference;
30. import com.google.firebase.database.FirebaseDatabase;
31. import com.google.firebase.database.ValueEventListener;
32.
33. import java.text.SimpleDateFormat;
34. import java.util.ArrayList;
35. import java.util.Date;
36. import java.util.Locale;
37.
38. public class AlertaFragment extends Fragment {
39.
40.     Button btn;
41.     String correo,user,aux;
42.     int i=1 ;
43.     private DatabaseReference mDatabase;
44.
45.     @Override
46.     public View onCreateView(LayoutInflater inflater, ViewGroup container,
47.                              Bundle savedInstanceState) {
48.         // Inflate the layout for this fragment
49.
50.         View view = inflater.inflate(R.layout.fragment_alerta, container, false);
51.
52.
53.         if(getArguments() != null) {
54.             correo = getArguments().getString("keybundle");
55.             user = getArguments().getString("userr");
56.         }
57.         else
58.             correo = "no entiendoooo";
59.
```

```
60.         btn = (Button)view.findViewById(R.id.Send);
61.
62.         btn.setOnClickListener(new View.OnClickListener() {
63.             @Override
64.             public void onClick(View view) {
65.
66.                 MandarAlerta(correo, user);
67.             }
68.         });
69.         return view;
70.     }
71.
72.     public void MandarAlerta(final String correo, final String user)
73.     {
74.         mDatabase = FirebaseDatabase.getInstance().getReference();
75.
76.         SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-
    dd HH:mm:ss", Locale.getDefault());
77.         Date date = new Date();
78.         String fecha = dateFormat.format(date);
79.         mDatabase.child("Usuarios").child(user).child("Alerta").child(fecha).child(
    "Fecha").setValue(fecha);
80.         mDatabase.child("Usuarios").child(user).child("Alerta").child(fecha).child(
    "ID").setValue(i);
81.         mDatabase.child("Usuarios").child(user).child("Alerta").child(fecha).child(
    "Ubicacion").child("Latitud").setValue(BackgroundService.Latitud);
82.         mDatabase.child("Usuarios").child(user).child("Alerta").child(fecha).child(
    "Ubicacion").child("Longitud").setValue(BackgroundService.Longitud);
83.         Toast.makeText(getContext(), "Alerta enviada", Toast.LENGTH_LONG).show();
84.
85.         try{
86.             Intent callIntent = new Intent(Intent.ACTION_CALL);
87.             callIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
88.             callIntent.setData(Uri.parse("tel:" + "5525762964"));
89.             getActivity().startActivity(callIntent);
90.         }catch (Exception e)
91.         {
92.             e.printStackTrace();
93.         }
94.     }
95. }
```

Implementación de la papeleta del conductor (Papeleta.java)

```
1.  package com.example.myapplication;
2.
3.  public class Papeleta {
4.
5.      String id_papeleta,fecha,turno;
6.      Check checks;
7.      String Boletaje;
8.
9.
10.
11.     public String getId_papeleta() {
12.         return id_papeleta;
13.     }
14.
15.     public void setId_papeleta(String id_papeleta) {
16.         this.id_papeleta = id_papeleta;
17.     }
18.
19.     public String getFecha() {
20.         return fecha;
21.     }
```

```
22.
23.     public void setFecha(String fecha) {
24.         this.fecha = fecha;
25.     }
26.
27.     public String getTurno() {
28.         return turno;
29.     }
30.
31.     public void setTurno(String turno) {
32.         this.turno = turno;
33.     }
34.
35.     public Check getChecks() {
36.         return checks;
37.     }
38.
39.     public void setChecks(Check checks) {
40.         this.checks = checks;
41.     }
42.
43.     public String getBoletaje() {
44.         return Boletaje;
45.     }
46.
47.     public void setBoletaje(String boletaje) {
48.         Boletaje = boletaje;
49.     }
50.
51. }
```

Programación del horario (HorarioFragment.java)

```
1.  package com.example.myapplication;
2.  import android.os.Bundle;
3.  import android.support.annotation.NonNull;
4.  import android.support.v4.app.Fragment;
5.  import android.view.LayoutInflater;
6.  import android.view.View;
7.  import android.view.ViewGroup;
8.  import android.widget.ArrayAdapter;
9.  import android.widget.ListView;
10. import android.widget.TextView;
11. import com.google.firebase.database.DataSnapshot;
12. import com.google.firebase.database.DatabaseError;
13. import com.google.firebase.database.DatabaseReference;
14. import com.google.firebase.database.FirebaseDatabase;
15. import com.google.firebase.database.ValueEventListener;
16. import java.util.ArrayList;
17.
18.
19. public class HorarioFragment extends Fragment {
20.     private String correo;
21.     TextView horario;
22.     ListView lista;
23.     ArrayList<String> Horario = new ArrayList<>();
24.     String [] Dias = {"Lunes","Martes","Miercoles","Jueves","Viernes"};
25.     private DatabaseReference mDatabase;// ...
26.
27.     @Override
28.     public View onCreateView(LayoutInflater inflater, ViewGroup container,
29.                              Bundle savedInstanceState) {
30.         // Inflate the layout for this fragment
31.
```

```
32.         View view = inflater.inflate(R.layout.fragment_horario, container, false);
33.
34.         if(getArguments() != null)
35.             correo= getArguments().getString("keybundle");
36.         else
37.             correo = "no entiendoooo";
38.
39.         horario = (TextView)view.findViewById(R.id.Horario);
40.
41.         //mail.setText(correo);
42.         //Toast.makeText(getContext(), "el correo es:  "+correo, Toast.LENGTH_LONG)
    .show();
43.
44.         lista = (ListView) view.findViewById(R.id.lista);
45.
46.         MostrarDatos(correo);
47.         return view;
48.
49.     }
50.
51.     public void MostrarDatos(final String correo)
52.     {
53.         mDatabase = FirebaseDatabase.getInstance().getReference();
54.         mDatabase.child("Usuarios").addValueEventListener(new ValueEventListener()
    {
55.             @Override
56.             public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
57.
58.                 int i=0;
59.                 for(DataSnapshot snapshot : dataSnapshot.getChildren()){
60.                     if(snapshot.child("Correo").getValue().toString().compareTo(cor
    reo)==0)
61.                     {
62.                         //Horario = snapshot.child("Horario").getValue().toString()
    ;
63.                         for (DataSnapshot horario : snapshot.child("Horario").getCh
    ildren()) {
64.                             Horario.add(Dias[i]+horario.getValue().toString());
65.                             i++;
66.                         }
67.
68.                     }
69.
70.                 }
71.                 i=0;
72.                 ArrayAdapter<String> adapter = new ArrayAdapter<String>(getContext(
    ), android.R.layout.simple_list_item_1,Horario);
73.                 lista.setAdapter(adapter);
74.
75.             }
76.             @Override
77.             public void onCancelled(@NonNull DatabaseError databaseError) {
78.
79.             }
80.         });
81.
82.     }
83. }
```

Aplicación (Usuario)

Programación de la visualización de los horarios (Usuario)

```
1.  package com.example.usuario;
2.  import android.os.Bundle;
3.  import android.support.annotation.NonNull;
4.  import android.support.v4.app.FragmentActivity;
5.  import android.widget.ArrayAdapter;
6.  import android.widget.ListView;
7.  import android.widget.TextView;
8.  import android.widget.Toast;
9.  import com.google.android.gms.maps.CameraUpdateFactory;
10. import com.google.android.gms.maps.model.LatLng;
11. import com.google.android.gms.maps.model.Marker;
12. import com.google.android.gms.maps.model.MarkerOptions;
13. import com.google.firebase.database.DataSnapshot;
14. import com.google.firebase.database.DatabaseError;
15. import com.google.firebase.database.DatabaseReference;
16. import com.google.firebase.database.FirebaseDatabase;
17. import com.google.firebase.database.ValueEventListener;
18. import java.util.ArrayList;
19. import static java.security.AccessController.getContext;
20.
21. public class HorariosActivity extends FragmentActivity {
22.     private DatabaseReference mDatabase;
23.     //textView2
24.     TextView hor;
25.     ListView lista;
26.     ArrayList<String> Horario = new ArrayList<>();
27.     String [] Dias = {"Unidad","Lunes","Martes","Miercoles","Jueves","Viernes"};
28.     @Override
29.     protected void onCreate(Bundle savedInstanceState) {
30.         super.onCreate(savedInstanceState);
31.         setContentView(R.layout.activity_horarios);
32.         hor= (TextView) findViewById(R.id.textView2);
33.         lista=findViewById(R.id.listhor);
34.         mDatabase =  FirebaseDatabase.getInstance().getReference();
35.       //  final String[] Horario = new String[1];
36.         mDatabase.child("Usuarios").addValueEventListener(new ValueEventListener()
    {
37.             @Override
38.             public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
39.                 int i=1;
40.                 for (DataSnapshot snapshot:dataSnapshot.getChildren()){
41.                     i=1;
42.                     Horario.add(Dias[0] + snapshot.child("Trolebus").child("Numero"
    ).getValue().toString() );
43.                     for (DataSnapshot horario : snapshot.child("Horario").getChildr
    en()) {
44.                         Horario.add(Dias[i]+horario.getValue().toString());
45.                         i++;
46.                     }
47.                 }
48.                 i=1;
49.                 ArrayAdapter<String> adapter = new ArrayAdapter<String>(HorariosAct
    ivity.this, android.R.layout.simple_list_item_1,Horario);
50.                 lista.setAdapter(adapter);
51.             }
52.             @Override
53.             public void onCancelled(@NonNull DatabaseError databaseError) {
54.             }
55.         });
56.     }
57. }
```

Programación de la visualización de la ubicación del trolebús (Usuario).

```
1.  package com.example.usuario;
2.  import android.support.annotation.NonNull;
3.  import android.support.v4.app.FragmentActivity;
4.  import android.os.Bundle;
5.  import android.widget.Toast;
6.  import com.google.android.gms.maps.CameraUpdate;
7.  import com.google.android.gms.maps.CameraUpdateFactory;
8.  import com.google.android.gms.maps.GoogleMap;
9.  import com.google.android.gms.maps.OnMapReadyCallback;
10. import com.google.android.gms.maps.SupportMapFragment;
11. import com.google.android.gms.maps.model.LatLng;
12. import com.google.android.gms.maps.model.Marker;
13. import com.google.android.gms.maps.model.MarkerOptions;
14. import com.google.firebase.database.DataSnapshot;
15. import com.google.firebase.database.DatabaseError;
16. import com.google.firebase.database.DatabaseReference;
17. import com.google.firebase.database.FirebaseDatabase;
18. import com.google.firebase.database.ValueEventListener;
19. import java.sql.DatabaseMetaData;
20. import java.util.ArrayList;
21.
22. public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

23.
24.     private GoogleMap mMap;
25.     private DatabaseReference mDatabase;
26.     private ArrayList<Marker>tmpRealTimeMarkers=new ArrayList<>();
27.     private ArrayList<Marker>realTimeMarkers=new ArrayList<>();
28.     private Double Latitud, Longitud;
29.     private ArrayList<Double> array = new ArrayList<>();
30.
31.     @Override
32.     protected void onCreate(Bundle savedInstanceState) {
33.         super.onCreate(savedInstanceState);
34.         setContentView(R.layout.activity_maps);
35.         // Obtain the SupportMapFragment and get notified when the map is ready to
    be used.
36.         SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentMan
    ager()
37.                 .findFragmentById(R.id.map);
38.         mapFragment.getMapAsync(this);
39.
40.         mDatabase =  FirebaseDatabase.getInstance().getReference();
41.     }
42.
43.
44.     /**
45.      * Manipulates the map once available.
46.      * This callback is triggered when the map is ready to be used.
47.      * This is where we can add markers or lines, add listeners or move the camera.
     In this case,
48.      * we just add a marker near Sydney, Australia.
49.      * If Google Play services is not installed on the device, the user will be pro
    mpted to install
50.      * it inside the SupportMapFragment. This method will only be triggered once th
    e user has
51.      * installed Google Play services and returned to the app.
52.      */
53.     @Override
54.     public void onMapReady(GoogleMap googleMap) {
55.         mMap = googleMap;
56.         mDatabase.child("Usuarios").addValueEventListener(new ValueEventListener()
    {
```

```
57.             @Override
58.             public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
59.
60.                 for(Marker marker:realTimeMarkers){
61.                     marker.remove();
62.                 }
63.                 for (DataSnapshot snapshot:dataSnapshot.getChildren()){
64.
65.                     for( DataSnapshot snap : snapshot.child("Ubicacion").getChi
    ldren()){
66.                     //MapsPojo mp = snap.getValue(MapsPojo.class);
67.                     //latitud = Double.parseDouble(snap.child("latitud").getVal
    ue().toString());//mp.getLatitud();
68.                     //longitud = Double.parseDouble(snap.child("longitud").getV
    alue().toString()); //mp.getLongitud();
69.                     array.add(Double.parseDouble(snap.getValue().toString()));

70.                     }
71.                     MarkerOptions markerOptions = new MarkerOptions();
72.                     markerOptions.position(new LatLng(array.get(0), array.get(1))).
    title("Camion 1");
73.                     tmpRealTimeMarkers.add(mMap.addMarker(markerOptions));
74.
75.                   //  mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(new LatLng(
    array.get(0), array.get(1)),13));
76.                     array.clear();
77.                 }
78.                     realTimeMarkers.clear();
79.                     realTimeMarkers.addAll(tmpRealTimeMarkers);
80.                 // Toast.makeText(MapsActivity.this, "lat:"+latitud+"  lon:"+longitu
    d, Toast.LENGTH_SHORT).show();
81.
82.             }
83.             @Override
84.             public void onCancelled(@NonNull DatabaseError databaseError) {
85.
86.             }
87.         });
88.         // Add a marker in Sydney and move the camera
89.        /* LatLng sydney = new LatLng(-34, 151);
90.         mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney
    "));
91.         mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));*/
92.     }
93. }
```

Implementación del apartado de quejas (Usuario)

```
1.  package com.example.usuario;
2.  import android.content.Intent;
3.  import android.os.Bundle;
4.  import android.support.annotation.NonNull;
5.  import android.support.v4.app.FragmentActivity;
6.  import android.view.View;
7.  import android.widget.Button;
8.  import android.widget.EditText;
9.  import android.widget.TextView;
10. import android.widget.Toast;
11. import com.google.firebase.database.DataSnapshot;
12. import com.google.firebase.database.DatabaseError;
13. import com.google.firebase.database.DatabaseReference;
14. import com.google.firebase.database.FirebaseDatabase;
15. import com.google.firebase.database.ValueEventListener;
16. import java.text.SimpleDateFormat;
17. import java.util.Date;
```

```
18. import java.util.Locale;
19.
20. public class Quejas extends FragmentActivity implements View.OnClickListener {
21.     private DatabaseReference mDatabase;
22.     TextView titulo;
23.     EditText numunidad;
24.     EditText quejasuge;
25.     Button enviar;
26.     String user;
27.     int ID=123;
28.     String numerounidad,quejaysugerencia;
29.     @Override
30.     protected void onCreate(Bundle savedInstanceState) {
31.         super.onCreate(savedInstanceState);
32.         setContentView(R.layout.quejas);
33.         mDatabase =  FirebaseDatabase.getInstance().getReference();
34.         titulo= (TextView) findViewById(R.id.textView3);
35.         numunidad=(EditText) findViewById(R.id.numuni);
36.         quejasuge=(EditText)findViewById(R.id.quejatxt);
37.         enviar=findViewById(R.id.btnenviar);
38.         enviar.setOnClickListener(this);
39.     }
40.
41.     @Override//55
42.     public void onClick(View v) {
43.         numerounidad=numunidad.getText().toString();
44.         quejaysugerencia=quejasuge.getText().toString();
45.         switch (v.getId()){
46.             case R.id.btnenviar:
47.                 mDatabase.child("Usuarios").addValueEventListener(new ValueEventLis
    tener() {
48.                     @Override
49.                     public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

50.                         int w=1,ID=123,i=0;
51.                         for(DataSnapshot snapshot : dataSnapshot.getChildren()){
52.                             w++;
53.                             for(DataSnapshot Numero:snapshot.child("Trolebus").getC
    hildren()) {
54.                                 if (Numero.getValue().toString().compareTo(numeroun
    idad) == 0) {
55.                                     String user = snapshot.getKey();
56.                                     ID++;
57.                                     SimpleDateFormat dateFormat = new SimpleDateFor
    mat("yyyy-MM-dd HH:mm:ss", Locale.getDefault());
58.                                     Date date = new Date();
59.                                     String fecha = dateFormat.format(date);
60.                                     mDatabase.child("Usuarios").child(user).child("
    Queja").child(fecha).child("Fecha").setValue(fecha);
61.                                     mDatabase.child("Usuarios").child(user).child("
    Queja").child(fecha).child("Numero").setValue(ID);
62.                                     mDatabase.child("Usuarios").child(user).child("
    Queja").child(fecha).child("Comentario").setValue(quejaysugerencia);
63.                                     Toast.makeText(Quejas.this, "Comentario enviado
    ", Toast.LENGTH_LONG).show();
64.                                     break;
65.                                 }
66.                             }break;
67.                         }
68.                     }
69.                     @Override
70.                     public void onCancelled(@NonNull DatabaseError databaseError) {

71.                     }
72.                 });
73.                 break; }}}
```

Capítulo VII Conclusiones

Este proyecto se realizó con éxito pues se alcanzaron los objetivos que anteriormente habíamos propuesto, se logró implementar la aplicación de manera correcta siguiendo los requisitos funcionales y los no funcionales.

Las pruebas que se realizaron en ambas aplicaciones y se fueron identificando problemas que se presentaron a la hora de implementar las dos aplicaciones fueron solucionados, pues como equipo se pudo identificar cual era la problemática y entre todos llegamos a una solución.

Bibliografía

[1] S. CDMX, «Linea circuíto Politecnico,» 1 Septiembre 2016. [En línea]. Available: https://www.ste.cdmx.gob.mx/linea-cp. [Último acceso: 20 abril 2019].

[2] Moovit, «Moovit,» [En línea]. Available: https://www.company.moovit.com/es. [Último acceso: 20 Abril 2019].

[3] sintesis, «Déjà-bus, una app con la que ya no llegarás tarde,» 13 julio 2017. [En línea]. Available: https://www.sintesis.mx/2017/07/13/deja-bus-una-app-la-ya-llegaras-tarde/. [Último acceso: 20 abril 2019].

[4] «Metodología orientada al objeto,» [En línea]. Available: ftp://www.dlsi.ua.es/people/jaime/apuntes/isi_tema3.1.pdf. [Último acceso: 10 marzo 2019].

[5] Google, «Firebase,» [En línea]. Available: https://firebase.google.com/?hl=es-419. [Último acceso: 20 marzo 2019].

[6] EcuRed, «Modelo de prototipos,» [En línea]. Available: https://www.ecured.cu/Modelo_de_prototipos. [Último acceso: 10 marzo 2019].

[7] getmonkey, «Ejemplo Estimación con el método de Cocomo,» [En línea]. Available: http://www.getmonkey.com/CDIM/finanzas/cocomo.pdf. [Último acceso: 10 abril 2019].

Anexos

Anexo A) Metodología estructurada Gane & Sarsons

Diagrama de contexto

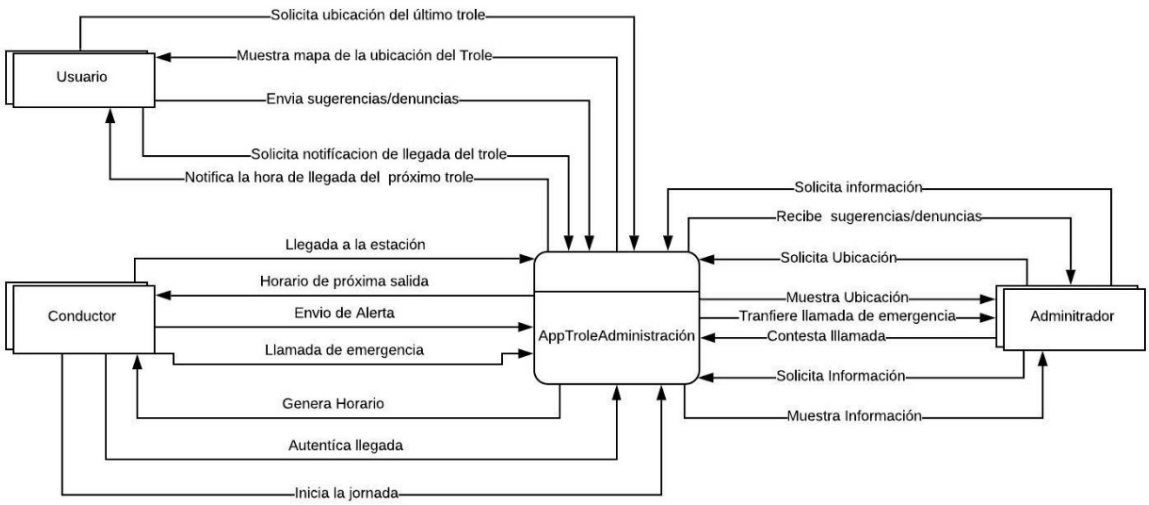


Imagen 7.1 “Diagrama de contexto”.

Diagrama de flujo de datos lógico

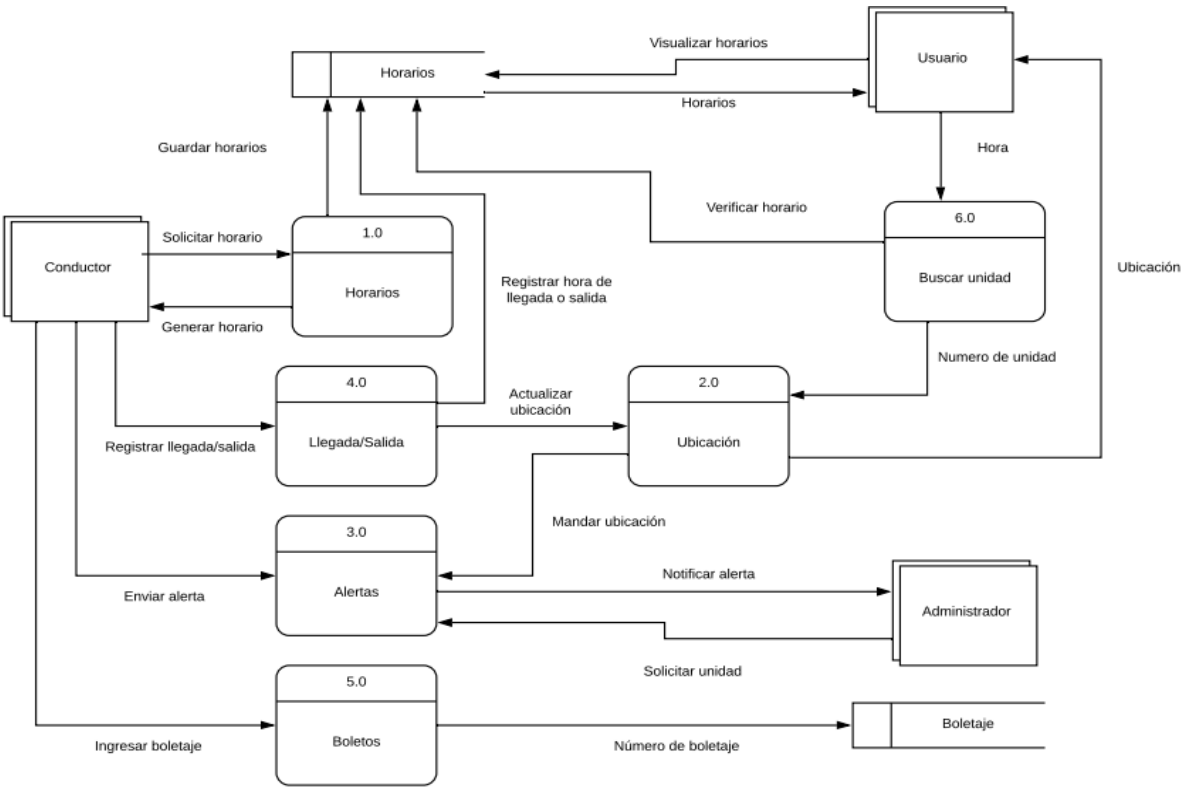


Imagen 7.2 “Diagrama de fujo de datos logico”.

Subproceso diagrama lógico-Horarios

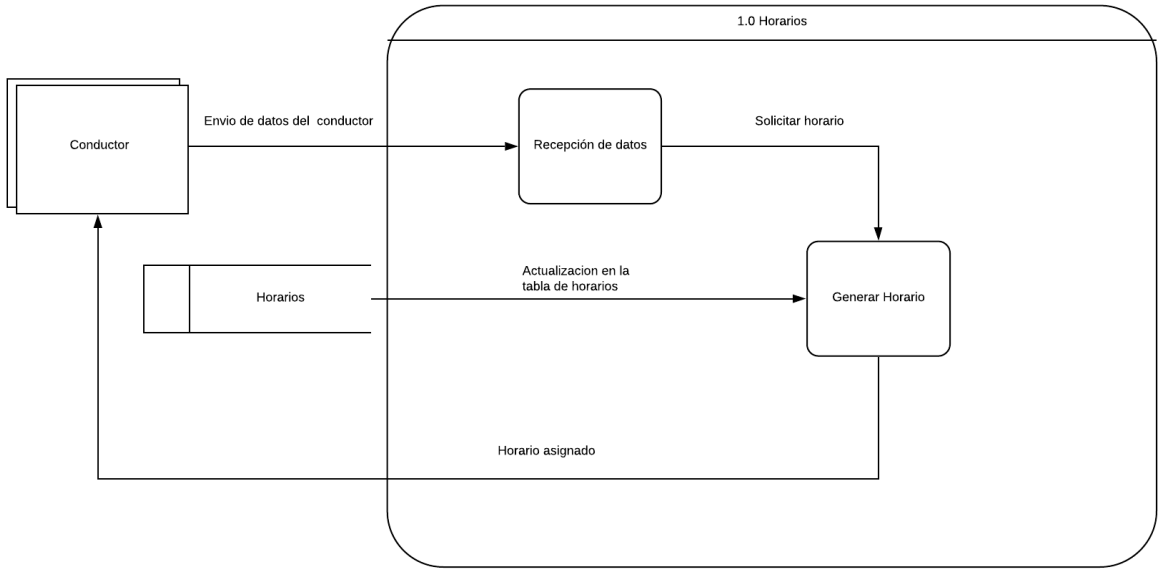


Imagen 7.3 “Diagrama de subprocesos lógico (Horarios)”.

Subproceso diagrama lógico-Ubicación

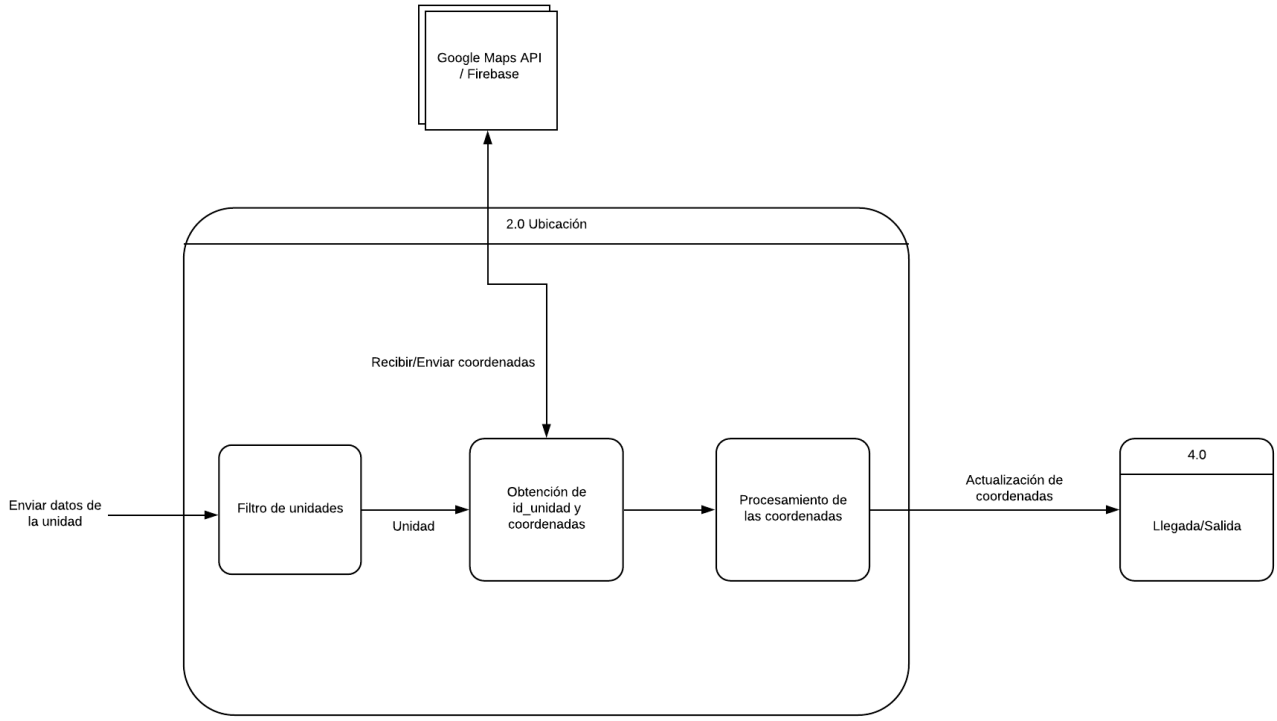


Imagen 7.4 “Diagrama de subprocesos lógico (Ubicación)”.

Subproceso diagrama lógico-Alertas

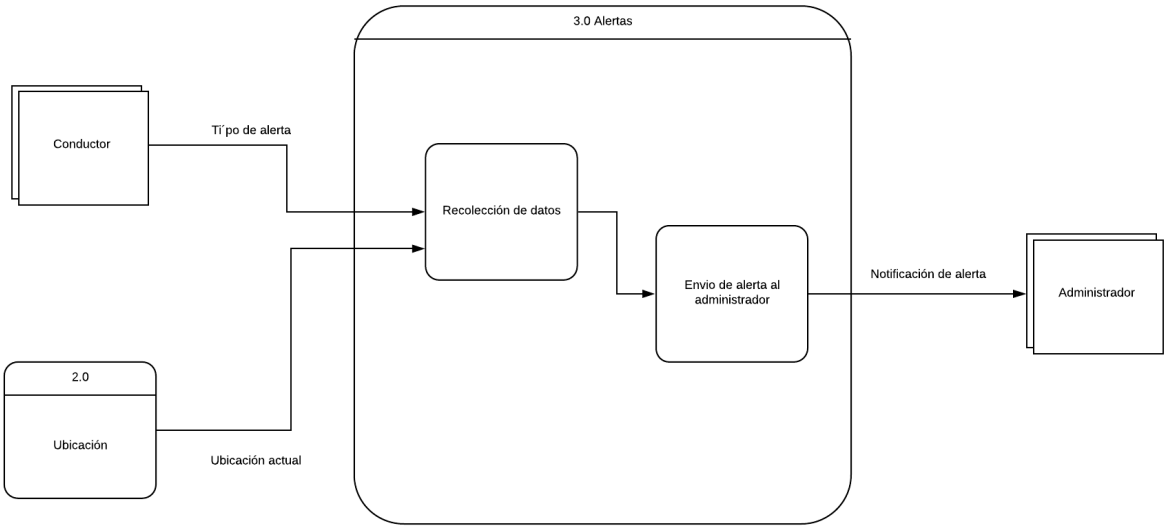


Imagen 7.5 “Diagrama de subprocesos lógico (Alertas)”.

Subproceso diagrama lógico-Llegada/Salida

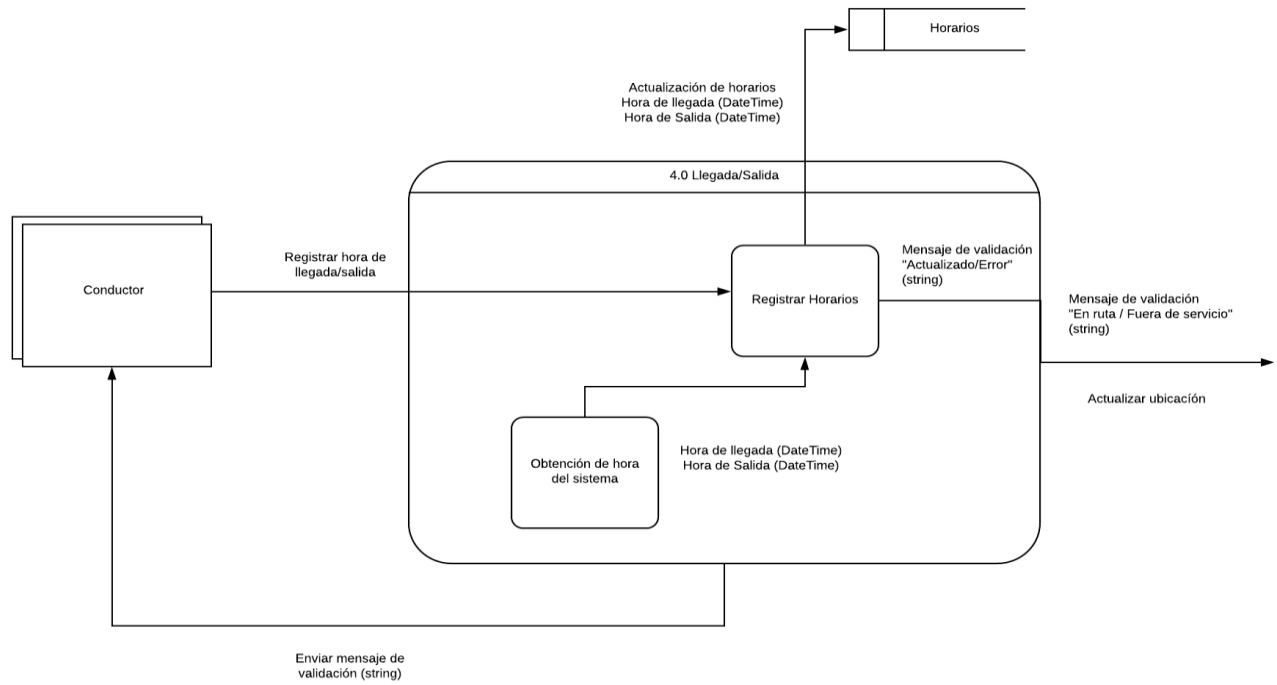


Imagen 7.6 “Diagrama de subprocesos lógico (Llegada-Salida)”.

Subproceso diagrama lógico-Buscar Unidad

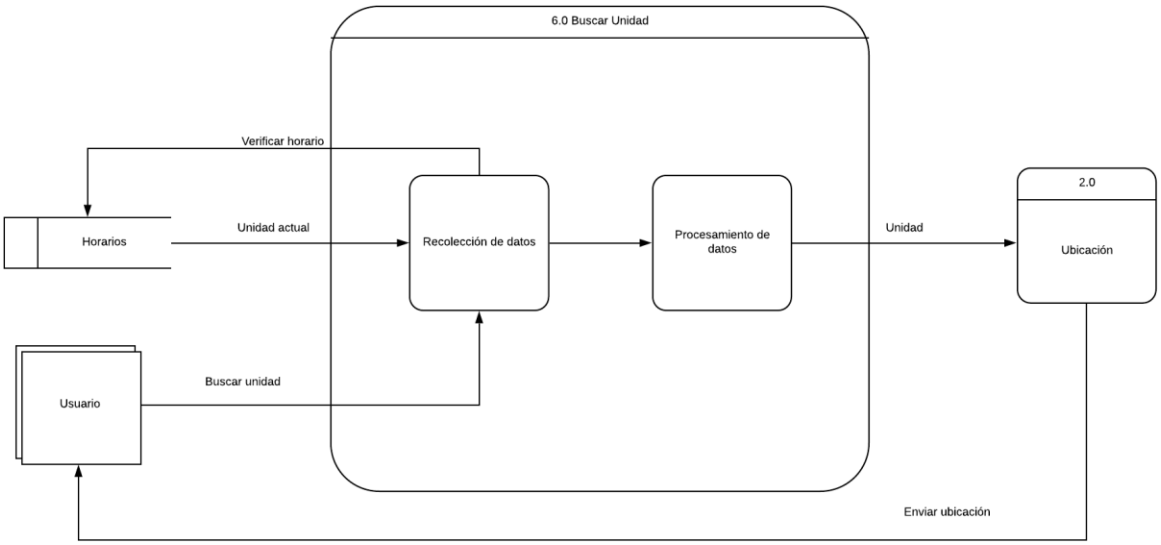


Imagen 7.7 “Diagrama de subprocesos lógico (Buscar unidad)”.

Diagrama de flujo de datos físico

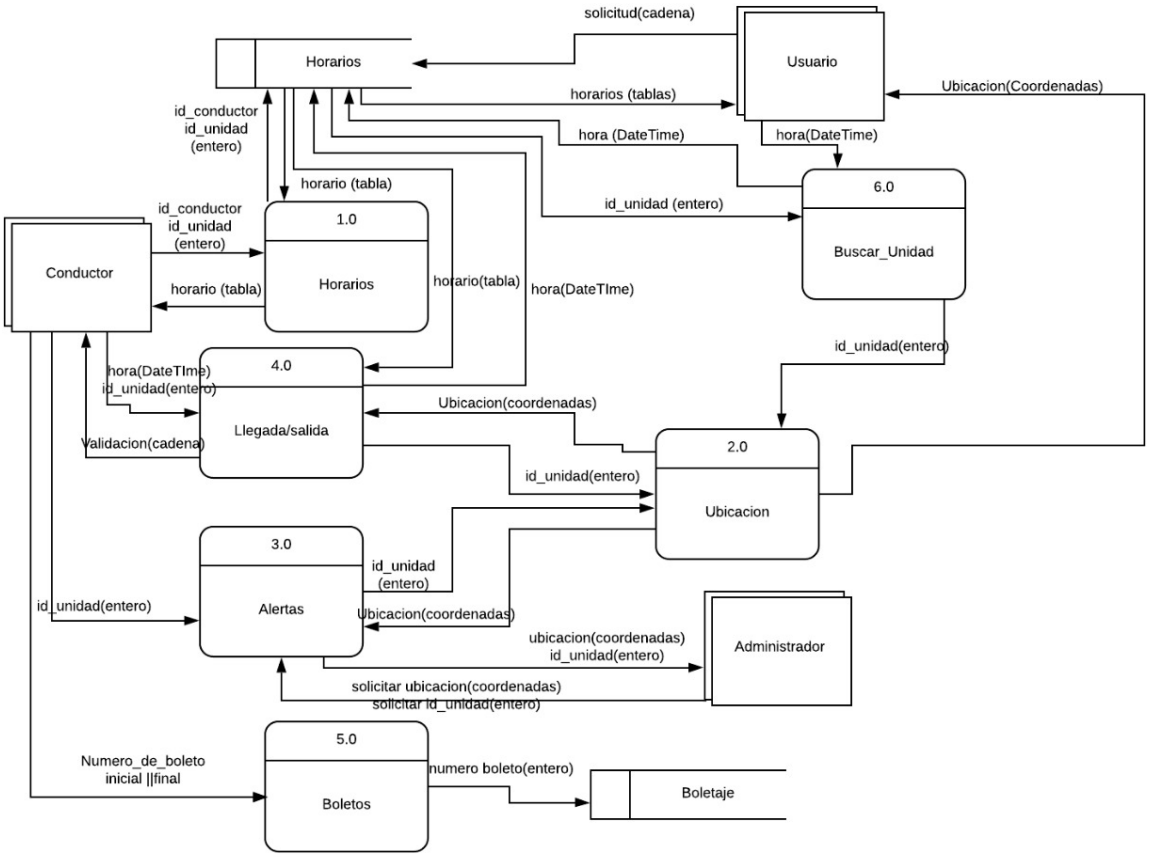


Imagen 7.8 “Diagrama de flujo de datos físico (Llegada-Salida)”.

Subproceso diagrama físico-Horarios

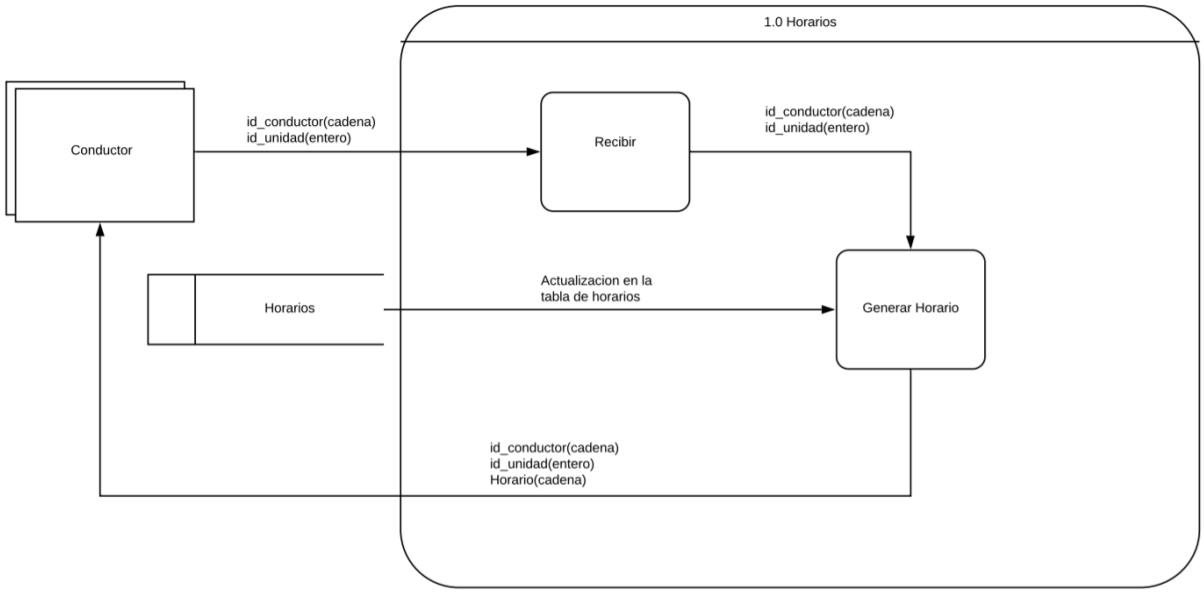


Imagen 7.9 “Diagrama de subproceso físico (Horario)”.

Subproceso diagrama físico-Ubicación

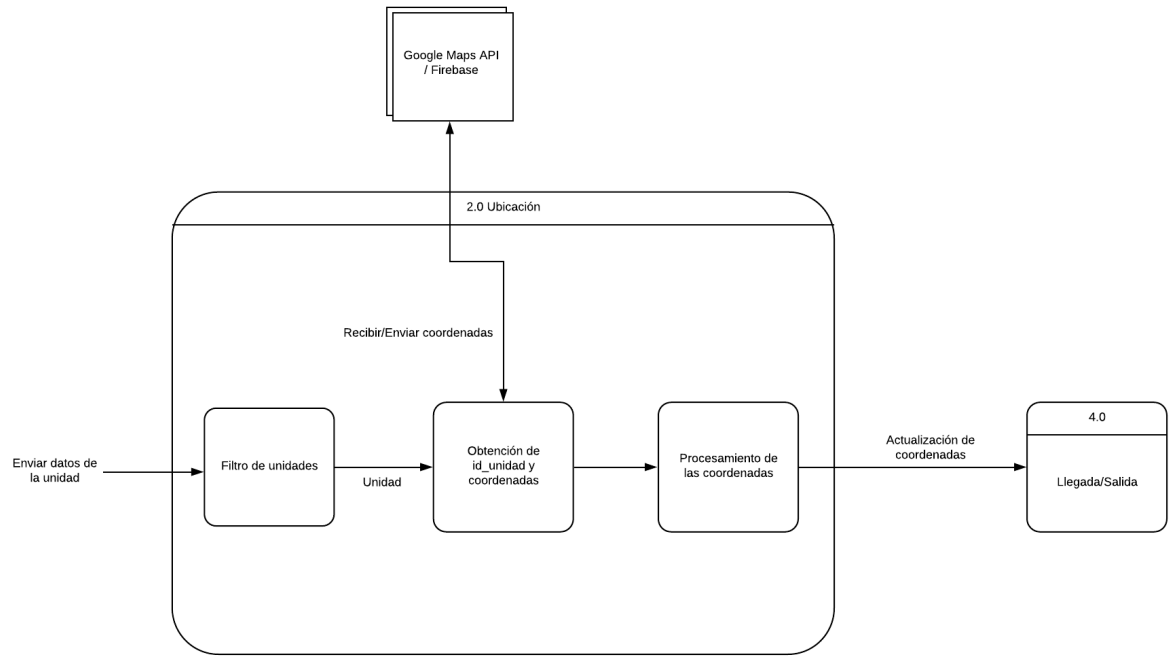


Imagen 7.10 “Diagrama de subproceso físico (Ubicación)”.

Subproceso diagrama físico-Alertas

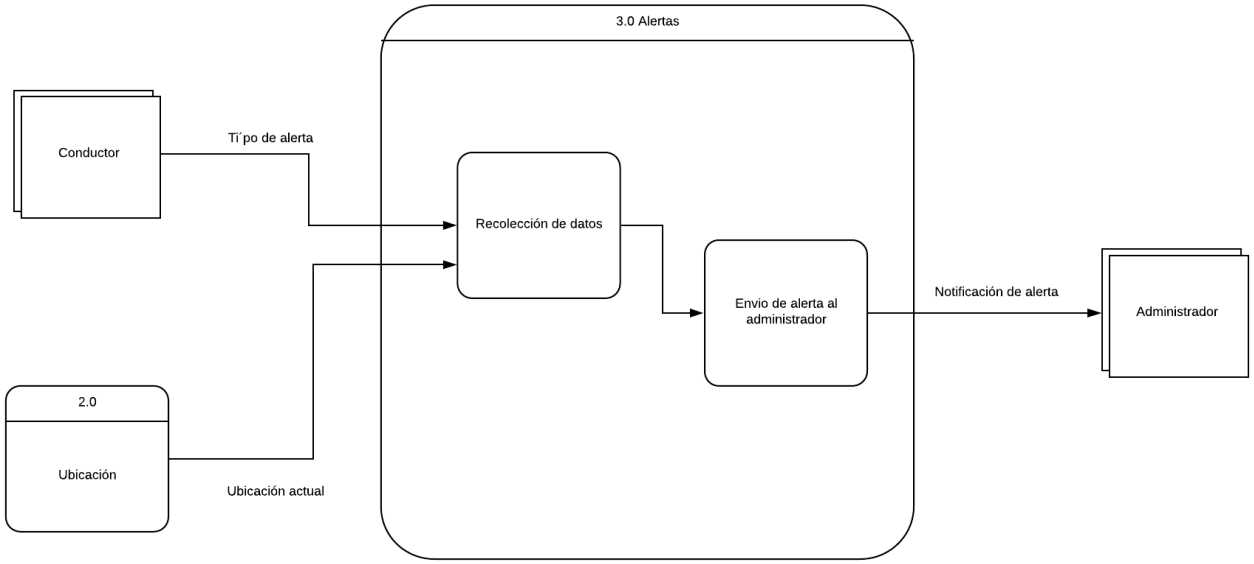


Imagen 7.11 “Diagrama de subproceso físico (Alertas)”.

Subproceso diagrama físico-Llegada Salida

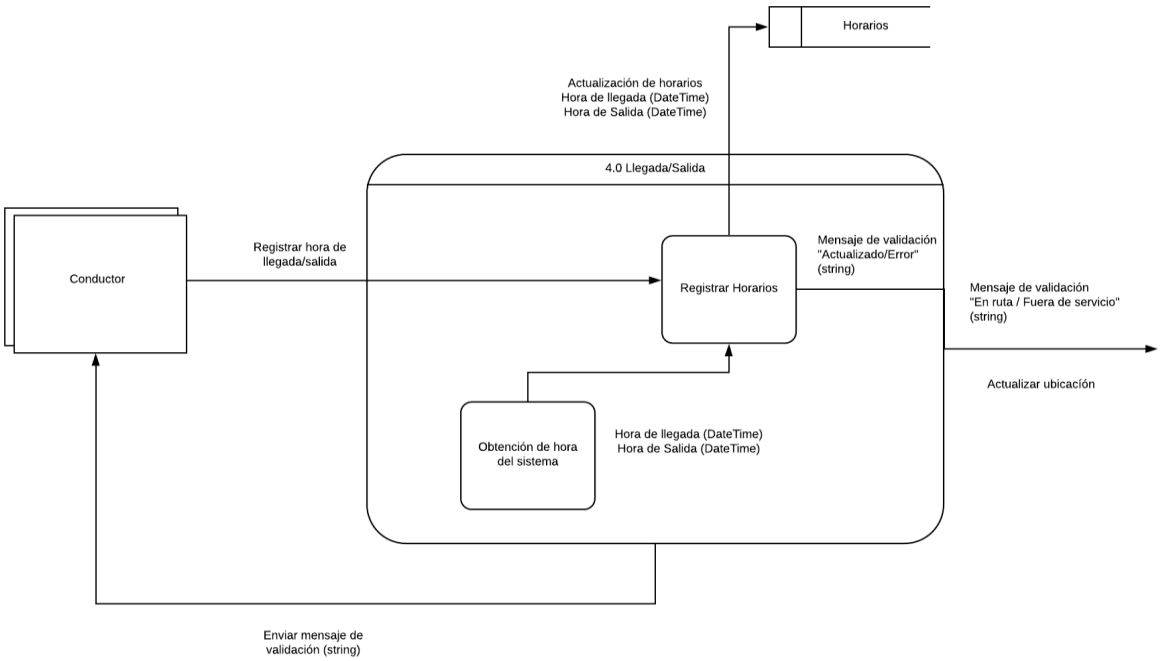


Imagen 7.12 “Diagrama de subproceso físico (Llegada-Salida)”.

Subproceso diagrama físico-Buscar Unidad

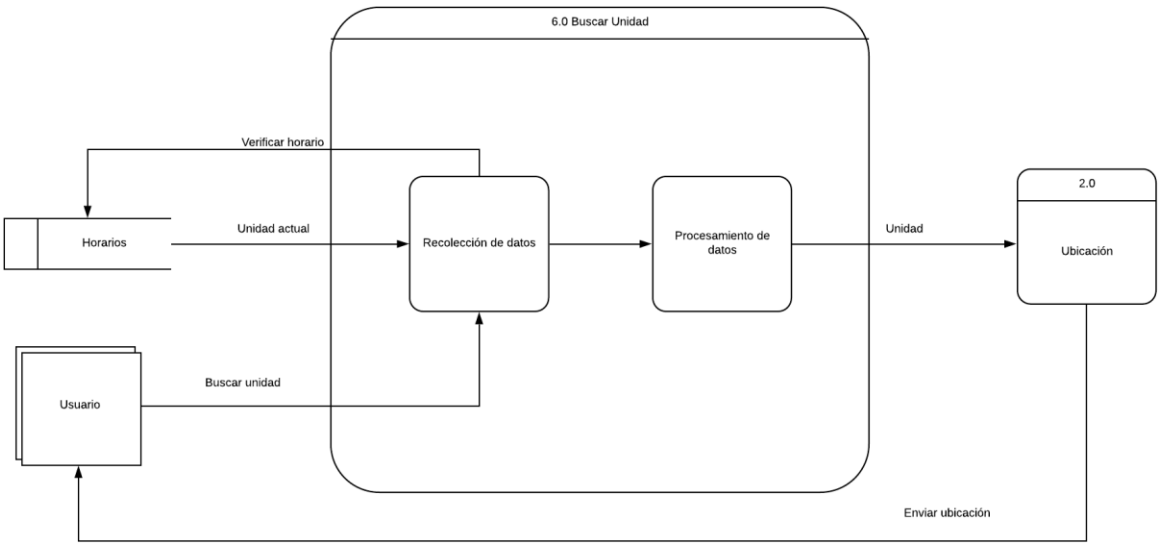


Imagen 7.13 “Diagrama de subproceso físico (Buscar unidad)”.

Anexo B) Metodología orientada a objetos Coud Yourdan

Narrativa

- La herramienta consistirá en dos aplicaciones móviles, una dirigida para los usuarios, y otra para la parte de administración.
- El usuario podrá obtener la ubicación de la unidad, hora de llegada de la próxima unidad, enviar quejas, sugerencias o denuncias.
- Por otro lado, la aplicación para la parte de administración; el administrador podrá visualizar los horarios de cada unidad, podrá visualizar su ubicación y recibirá las llamadas de emergencia por parte del conductor.
- Toda esta información estará almacenada en una “papeleta", esta contendrá la información de todas las unidades, como su hora de llegada y salida, el nombre del conductor, turno, fecha y tendrá un registro del boletaje.
- Para el conductor, cuando este se registre, el sistema le asignará un horario para trabajar, podrá enviar una alerta en caso de emergencia, le permitirá registrar su hora de llegada y salida y registrar el boletaje final después de cada trayecto

Definición de objetos

- Usuario: Podrá enviar quejas o sugerencias.
- Supervisor: Podrá visualizar los horarios y ubicación de cada unidad y recibirá las llamadas de emergencia por parte del conductor.
- Conductor: Estará registrado en una papeleta, se le asignará un horario, podrá mandar alertas, registrar su hora de llegada/salida y registrará el boletaje final.
- Papeleta: Tendrá la información de todas las unidades, conductores, horarios y un registro del boletaje.
- Horario: Se compone de un horario de entrada y horario de salida.
- Trolebús: Tendrá una matrícula y un número de Trolebus.
- Queja/Sugerencia: Se registrará la fecha y hora en la que se mandó la sugerencia y el comentario que el usuario quiera enviar.
- Alarma de Emergencia: Se registrará la fecha y hora en la que se emitió la alerta y se guardará su ubicación.

Características de los objetos

Tabla 7.0 “características de los objetos”.

Clase/Objeto	Características aplicables
Usuario	Aceptado (Se aplican 2,3,4,5,6)
Administrador	Aceptado (Se aplican todas)
Conductor	Aceptado (Se aplican todas)
Papeleta	Aceptado (Se aplican todas)
Horario	Aceptado (Se aplican todas)
Trolebús	Aceptado (Se aplican todas)
Alarma	Aceptado (Se aplican todas)
Ubicación	Aceptado (Se aplica 2,4,5,6)