

Gestion de Parqueaderos

Luis Alejandro Ocampo - 20172020050
Daniel Esteban Rodriguez - 20172020120
Camilo Andres Torres - 20172020067

8 de julio de 2020

Índice general

I PROYECTO	7
1. Proyecto	9
1.1. Introducción	9
1.2. Descripción General	10
1.3. Objetivos	11
1.4. Alcances y Límites	12
1.5. Cronograma	13
2. Metodología	15
2.1. Análisis de Requerimientos	15
2.2. Procesos de Software	16
II DISEÑO	17
3. UML[4, 1, 2, 3]	19
3.1. Introducción	19
3.2. Historia	20
3.3. Estructura	21
3.3.1. Elementos de UML	21
3.3.2. Relaciones de UML	23
3.3.3. Diagramas de UML	25
4. Casos de Uso	27
4.1. Introducción	27
5. Interacciones	35
5.1. Introducción	35
5.2. Diagramas de Secuencia	36
5.3. Diagramas de Comunicación	44

6. Clases	49
6.1. Introduccion	49
6.2. Descripción	50
6.3. Diagrama de Clase UML	51
6.4. Diagrama de Clase UML: Patrones de Diseño	52
6.4.1. Patrón: Estrategia	52
6.4.2. Patrón: Método Fábrica	53
6.4.3. Patrón: Comando	54
7. Estados	55
7.1. Introduccion	55
7.2. Diagrama de Estados	56
7.2.1. Diagrama de Clase	57
7.2.2. Diagrama de Clase: Patron Estado	58
7.2.3. Diagrama de Secuencia	59
7.3. Anexo 1:	60
7.3.1. Anexo 1: Diagrama de clase	61
7.3.2. Anexo 1: Diagrama de Clase del Patrón Estado	62
7.3.3. Anexo 1: Diagrama de Secuencia Extendido	63
8. Actividades	65
8.1. Introduccion	65
8.2. WorkFlow	66
9. Componentes	67
9.1. Introduccion	67
10. Sistemas	69
10.1. Introduccion	69
11. Nodos	71
11.1. Introduccion	71
III CIERRE	73
12. Conclusiones	75

Índice de figuras

1.1.	Cronograma	13
2.1.	Cascada	16
2.2.	SCRUM	16

Parte I

PROYECTO

Capítulo 1

Proyecto

1.1. Introducción

En este proyecto se pretende desarrollar una aplicación web de tipo economía colaborativa, en la que se gestionen los parqueaderos en la ciudad y los usuarios puedan solicitar un servicio de alquiler de los mismos, de igual manera, los mismos usuarios pueden ofrecer en alquiler un parqueadero en su propiedad, todo basándose en, como se mencionó antes, un consumo colaborativo, en el cual los consumidores pueden actuar como proveedores de recursos o receptores de recursos.

1.2. Descripción General

La plataforma esta pensada para generar un comercio virtual entre arrendatario/arrendador de manera simple y practica, para poder alquilar u ofrecer en alquiler un espacio donde aparcar cualquier tipo de vehiculo, ya sean carros, camionetas, motos, e incluso bicicletas y vehiculos especiales, todo funcionando como una economia colaborativa donde cualquier persona pueda obtener un beneficio al usar la plataforma, se va a utilizar un sistema de valoraciones tanto para quien ofrece un espacio como para quien decide tomar éste en alquiler, todo a manera de guia para los usuarios.

1.3. Objetivos

El proyecto tiene como objetivo principal facilitar el transporte en la ciudad para usuarios de carros, motos, bicicletas, etc. sin tener que preocuparse por donde van a estacionar su vehículo o por el precio que van a pagar, esto con el afán de agilizar y facilitar el proceso de apartado de un espacio para aparcar un vehículo, además de combatir los arbitrarios precios de parqueaderos en sectores de mucha concentración laboral o estudiantil.

1.4. Alcances y Limites

Alcances

El presente proyecto explorara el sector vehicular en la ciudad, apuntando sobretodo, al sector de la poblacion que se presenta constantemente a espacios donde se pagan unos muy altos precios de estacionamiento, o donde es muy limitado el espacio para aparcar, ya sea una zona de alta densidad laboral, cerca a un centro comercial, o una zona universitaria.

Limites

Las limitaciones del proyecto van mas orientadas a los sectores donde el sistema de parqueo no presente fallos criticos, es decir, sectores con baja concurrencia vehicular no se contemplan dentro del proyecto por su baja probabilidad tanto de potenciales arrendatarios como de potenciales arrendadores, aunque esto va directamente ligado al desarrollo del proyecto.

1.5. Cronograma

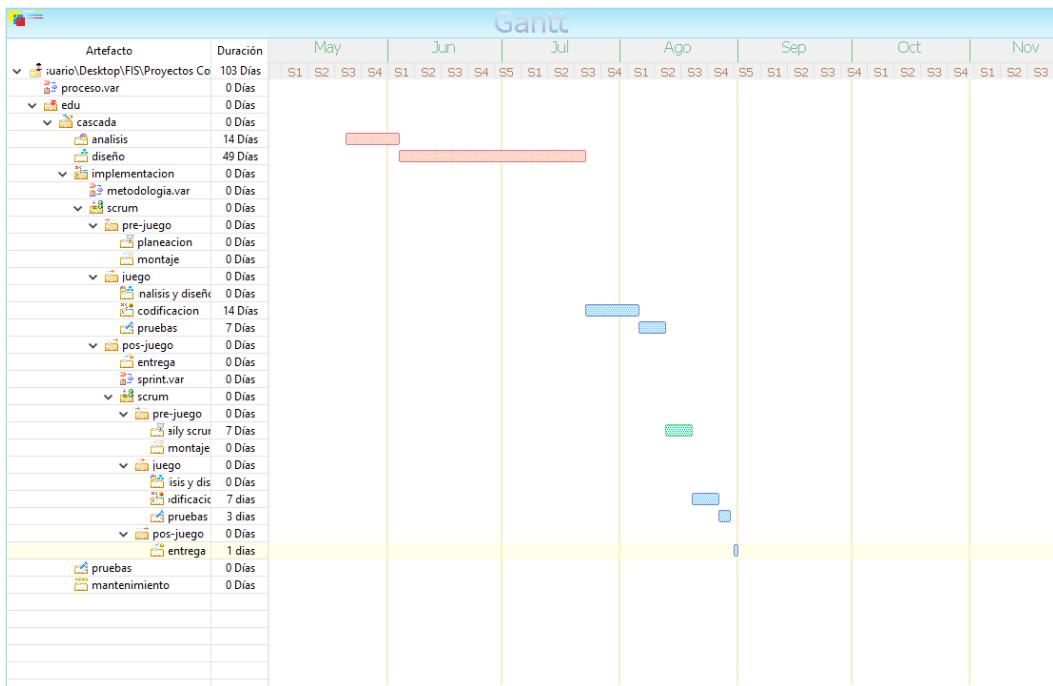


Figura 1.1: TimeLine del Cronograma

Descripción

El Cronograma consta de una duración de aproximadamente 14 semanas donde se hace un especial énfasis en la etapa de diseño, correspondiente al proceso que vamos a incorporar (Modelo de Cascada) se debe enfocar una gran parte de los esfuerzos del equipo en esta etapa de diseño, esto, después de haber realizado el correspondiente análisis de requerimientos, con todo lo que conlleva esta etapa.

Una vez finalizada la etapa de diseño, se incorpora una metodología agilista para enfrentar la etapa de implementación, la Metodología SCRUM, donde únicamente tomamos de esta metodología, como ya se mencionó, la fase de implementación y prueba, que consta de dos iteraciones del proceso tanto de codificación como de sus respectivas pruebas, este proceso en total suele llevar entre 2 a 4 semanas, donde

en este periodo, ademas de realizar los procesos ya mencionados, internamente contiene una fase de daily Scrum, cuya finalidad es relizar reuniones diarias para retroalimentar el trabajo hasta ese punto, ver los posibles fallos y compartir las ideas del equipo, tal y como lo describe el modelo.

Y para finalizar, con los timpos justos del semestre y esperando que no haya ningun tipo de contratiempo, se espera realizar la correspondiente entrega del proyecto.

Capítulo 2

Metodología

2.1. Análisis de Requerimientos

El proyecto de gestión de parqueaderos basado en el principio de economía colaborativa deberá contar con una serie de especificaciones y requisitos para su correcto desarrollo e implementación, para comenzar será una aplicación desarrollada en entorno web para su fácil acceso desde cualquier dispositivo y esta deberá contar con la posibilidad de creación y manejo de cuentas personales para la administración de sus objetos vinculados ya sean automóviles o parqueaderos como tal permitiendo añadir estos, de igual manera permitirá modificar la tarifa que desea manejar el usuario y medios de pago, cada automóvil registrado deberá contar con modelo y número de placa y cada parqueadero con la ubicación, número de plazas disponibles, y si es abierto o cerrado.

Se deberá poder consultar en un mapa la ubicación de los parqueaderos registrados disponibles y poder consultar la tarifa que manejan estos y los medios de pago disponibles, por ende este mapa debe estar actualizado a tiempo real.

2.2. Procesos de Software

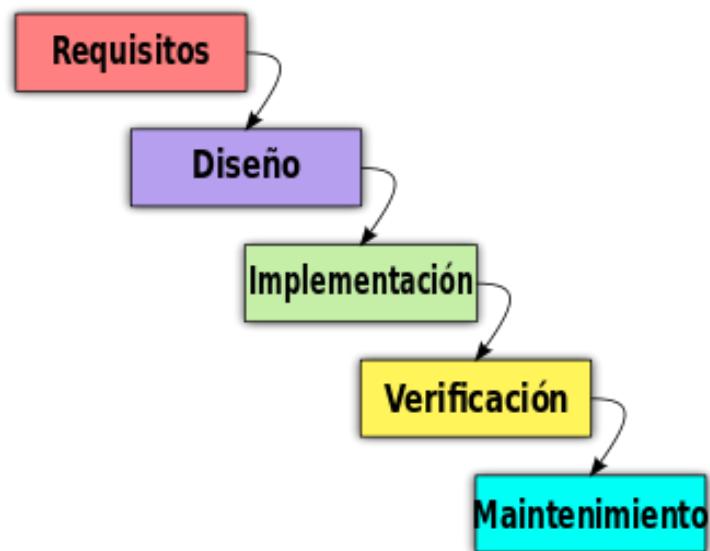


Figura 2.1: Proceso Cascada



Figura 2.2: Metodología SCRUM

Parte II

DISEÑO

Capítulo 3

UML[4, 1, 2, 3]

3.1. Introducción

UML es una notación estándar para el modelado de software para la arquitectura, diseño e implementación de software, ya sea en su estructura como en comportamiento siendo este bastante visual, al ser un lenguaje que usa diagramas este no se limita a usarse únicamente en el desarrollo de software. UML no es un proceso de desarrollo, es decir, no describe los pasos sistemáticos a seguir para desarrollar software. UML sólo permite documentar y especificar los elementos creados mediante un lenguaje común describiendo modelos. Es un lenguaje de propósito general para el modelado orientado a objetos, que combina notaciones provenientes desde: Modelado Orientado a Objetos, Modelado de Datos, Modelado de Componentes, Modelado de Flujos de Trabajo (Workflows).

3.2. Historia

El lenguaje UML comenzó a gestarse en octubre de 1994, cuando Rumbaugh se unió a la compañía Rational fundada por Booch (dos reputados investigadores en el área de metodología del software). El objetivo de ambos era unificar dos métodos que habían desarrollado: el método Booch y el OMT (Object Modelling Tool). El primer borrador apareció en octubre de 1995. En esa misma época otro reputado investigador, Jacobson, se unió a Rational y se incluyeron ideas suyas. Estas tres personas son conocidas como los “tres amigos”. Además, este lenguaje se abrió a la colaboración de otras empresas para que aportaran sus ideas. Todas estas colaboraciones condujeron a la definición de la primera versión de UML. Figura 1: Logo de UML Se necesitaba por tanto un lenguaje no sólo para comunicar las ideas a otros desarrolladores sino también para servir de apoyo en los procesos de análisis de un problema. Con este objetivo se creó el Lenguaje Unificado de Modelado (UML: Unified Modeling Language). UML se ha convertido en ese estándar tan ansiado para representar y modelar la información con la que se trabaja en las fases de análisis y, especialmente, de diseño. Esta primera versión se ofreció a un grupo de trabajo para convertirlo en 1997 en un estándar del OMG (Object Management Group <http://www.omg.org>). Este grupo, que gestiona estándares relacionados con la tecnología orientada a objetos (metodologías, bases de datos objetuales, CORBA, etc.), propuso una serie de modificaciones y una nueva versión de UML (la 1.1), que fue adoptada por el OMG como estándar en noviembre de 1997. Desde aquella versión han habido varias revisiones que gestiona la OMG Revision Task Force. La última versión aprobada es la 1.4. En estos momentos se está desarrollando una nueva versión en la que se incluirán cambios importantes (principalmente añadir nuevos diagramas) que conducirán a la versión 2.0 planificada para fines del 2002.

3.3. Estructura

3.3.1. Elementos de UML

Elemento Estructurales

Elemento	Definicion	Notacion
Caso de uso	Un caso de uso representa la forma en como un Cliente (Actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso).	
Colaboracion	La colaboración permite que una clase se comunique con otra clase con el propósito de utilizar los métodos de esta ultima para mejorar algún servicio o mantener una relación lógica de dependencia.	
Clase	Es un elemento estructural en el que se modelan atributos y operaciones de un conjunto de objeto.	
Interface	Esta especifica qué se debe hacer pero no su implementación. Serán las clases que implementen estas interfaces las que describen la lógica del comportamiento de los métodos.	
Componente	Un componente, también denominado componente simple o atómico, es un objeto que tiene una representación gráfica, que se puede mostrar en la pantalla y con la que puede interactuar el usuario.	
Nodo	un nodo es cada uno de los elementos de una lista enlazada , un árbol o un grafo en una estructura de datos. Cada nodo tiene sus propias características y cuenta con varios campos; al menos uno de éstos debe funcionar como punto de referencia para otro nodo.	
Artefacto	En términos generales de software, un artefacto es algo producido por el proceso de desarrollo de software, ya sea una documentación relacionada con el software o un archivo ejecutable.	

Elemento de Comportamiento

Elemento	Definicion	Notacion
Estado	El diagrama de estado se usa para dar forma al comportamiento de un objeto, de una clase. Se representa la secuencia de estados que un objeto de la clase tiene durante su vida, según las acciones que van sucediendo.	
Actividad	Básicamente se puede decir que el diagrama de actividades modela el flujo de actividades. Estos pueden ser procesos dentro de un sistema informático, procesos de casos de uso o procesos comerciales. Estas actividades pueden descomponerse en muchas actividades secundarias más pequeñas: las acciones, que pueden llevarse a cabo cronológicamente.	
Transicion	Es un comportamiento que comprende un conjunto de mensajes intercambiados entre un conjunto de objetos, dentro de un contexto particular para alcanzar un propósito específico.	

Elemento Agrupación

Elemento	Definicion	Notacion
Sistema	Es un mecanismo de propósito general para organizar elementos en grupos. En estos se pueden agrupar los elementos estructurales, de comportamiento e incluso otros elementos de agrupación, son los elementos de agrupación básicos con los cuales se puede organizar un modelo UML.	
Frame	Similar al Sistema, usado mas en diagramas dinamicos	

Elemento Anotación

Elemento	Definicion	Notacion
Nota	<p>Son comentarios que se pueden aplicar para describir, clasificar y hacer observaciones sobre cualquier elemento de un modelo.</p> <p>El tipo principal de anotación es la nota que simplemente es un símbolo para mostrar restricciones y comentarios junto a un elemento o un conjunto de elementos.</p>	

3.3.2. Relaciones de UML

Relaciones Estructurales

Elemento	Definicion	Notacion
Dependencia	Es una relación de significado entre dos elementos, donde cualquier cambio a un elemento independiente, puede afectar el significado de otro elemento dependiente.	
Asociacion	Las asociaciones representan las relaciones más generales entre clases, es decir, las relaciones con menor contenido semántico. Para UML una asociación va a describir un conjunto de vínculos entre las instancias de las clases.	
Agregacion	Es muy similar a la relación de Asociación solo varía en la multiplicidad ya que en lugar de ser una relación uno a uno. es de uno a muchos".	

Composicion	<p>Aporta documentación conceptual ya que es una relación de vida”, es decir, el tiempo de vida de un objeto está condicionado por el tiempo de vida del objeto que lo incluye.</p>	<pre> classDiagram class Empresa { addEmpleado():void } class Empleado { name age } Empresa "1" *-- "*" Empleado </pre>
-------------	---	---

Relaciones de Comportamiento

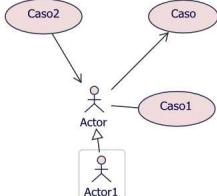
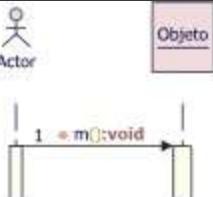
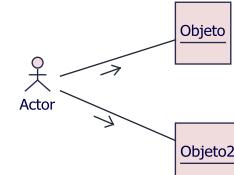
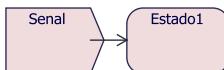
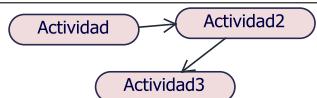
Elemento	Definicion	Notacion
Generalizacion	<p>Esta hace referencia a la relación de una súper clase o clase padre con una subclase o clase hija. La generalización significa que los objetos hijos se pueden emplear en cualquier clase donde pueda aparecer el padre, pero no a la inversa.</p>	<pre> classDiagram class Vehiculo { mover():void } class Bus class Auto Vehiculo < -- Bus Vehiculo < -- Auto </pre>
Realizacion	<p>Es una relación de contrato con otra clase. Se la utiliza para implementar una interfaz.</p>	<pre> classDiagram interface Ventas class Venta { total():void } Ventas --> Venta </pre>

3.3.3. Diagramas de UML

Diagramas Estáticos (o Estructurales)

Elemento	Definicion	Notacion
Clases	Describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o metodos), y las relaciones entre los objetos.	<pre> classDiagram class Clase { <<atributo>> <<operacion>> } class Clase2 class Clase3 interface Interface1 interface Interface2 interface Interface3 Clase < -- Clase2 Clase < -- Clase3 Clase --> Interface1 Clase --> Interface2 Clase --> Interface3 </pre>
Objetos	Los diagramas de objetos modelan las instancias de elementos contenidos en los diagramas de clases. Un diagrama de objetos muestra un conjunto de objetos y sus relaciones en un momento concreto.	<pre> objectDiagram object ClaseA a() : String end object ObjetoA --> ClaseA : a() end object ObjetoA2 --> ClaseA : a() end </pre>
Componentes	Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes.	<pre> componentDiagram component Componente1 <<interface>> <<interface>> <<interface>> end component Componente2 <<interface>> <<interface>> end interface Interface1 interface Interface2 interface Interface3 interface Interface4 Componente1 < --> Interface1 Componente1 < --> Interface2 Componente1 < --> Interface3 Componente2 < --> Interface4 </pre>
Sistemas	Muestra como un proyecto esta dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones.	<pre> systemDiagram system Sistema1 system Sistema2 system Sistema3 Sistema1 --> Sistema2 Sistema2 --> Sistema3 </pre>
Estructura Compuesta	Muestra la estructura interna de una clase y las colaboraciones que esta estructura hace posibles.	<pre> compositeStructureDiagram class Clase1 { <<internalCompartment>> <<internalCompartment>> } class Clase2 interface Interface Clase1 < --> Clase2 Clase2 < --> Interface Clase1 ..> Colaboracion </pre>

Diagramas dinámicos (o de Comportamiento)

Elemento	Definicion	Notacion
Casos de uso	Los diagramas de caso de uso modelan la funcionalidad del sistema usando actores y casos de uso.	
Secuencias	Un diagrama de secuencia es un tipo de diagrama de interacción porque describe cómo (y en qué orden) un grupo de objetos funcionan en conjunto.	
Comunicacion	Un diagrama de comunicación modela las interacciones entre objetos o partes en términos de mensajes en secuencia.	
Estados	Es un diagrama que muestra transiciones entre diversos objetos.	
Actividades	Representa los flujos de trabajo paso a paso de negocio y operacionales de componentes en un sistema.	
Vista Conjunta de Interaccion	Es una representación gráfica de una interacción, este se distingue fuertemente de los diagramas de secuencia y de comunicación, así como de los otros diagramas de interacción.	

Capítulo 4

Casos de Uso

4.1. Introducción

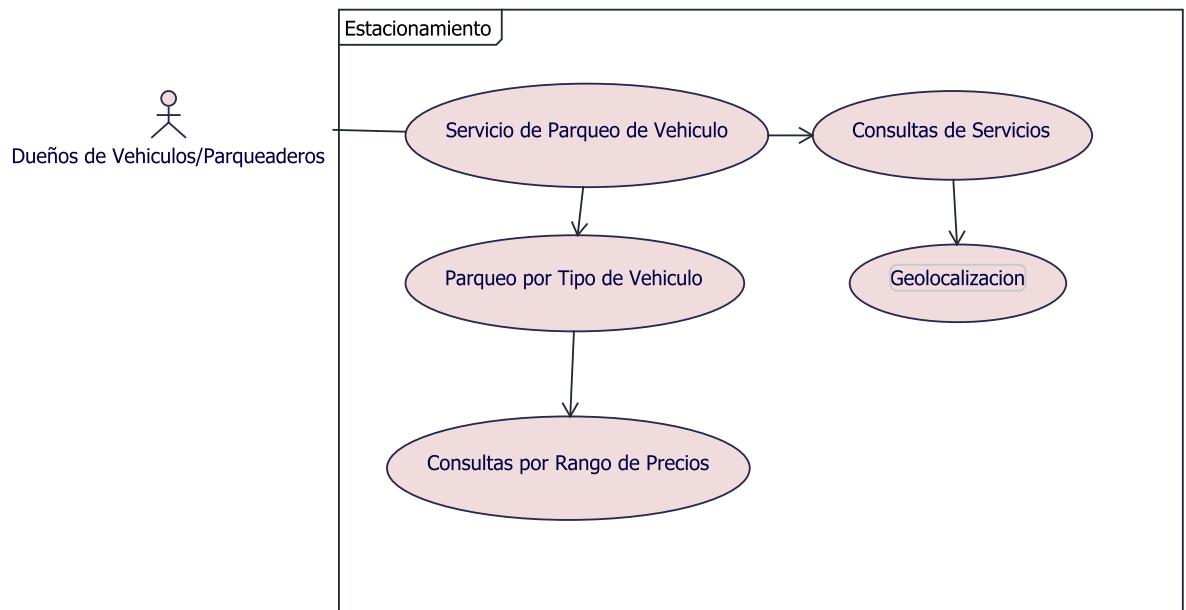
En base a lo que hemos revisado anteriormente nos damos cuenta que existen diferentes metodologías que nos permiten desarrollar software de calidad enfocadas a las necesidades que se tengan.

Dentro de un proyecto de software existen diferentes etapas, una de estas independientemente de la metodología que se esté utilizando es definir de manera concisa y general qué es lo que se quiere, ya que es fundamental para definir los requerimientos de software porque muchas veces lo que se plantea no es lo que realmente se ve en el proyecto terminado, es por esto que se definen formas de presentar una perspectiva primitiva de lo que será el software una vez finalizado.

Existen diferentes formas de representar la funcionalidad del software sin estar terminado, una de estas es el Lenguaje Unificado de Modelado “UML”, que es el sistema de modelado de software más conocido y utilizado en la actualidad.

Dentro de UML se pueden encontrar diversos diagramas que permiten representar las diferentes perspectivas de un sistema, ademas, apoyados en la herramienta Coloso se consiguió el correcto modelado de dichos diagramas.

Los Casos de Uso son diagramas que permiten representar que hará el sistema pero no como funciona, a continuación se analiza su implementación en nuestro proyecto de software.



A continuacion, se realizaron una serie de tablas con descripciones de los casos de uso del diagrama anterior.

Caso uso: Servicio de Parqueo del Vehículo	
<pre> graph LR User[User con Vehiculo] --> SPV[Service de Parqueo de Vehiculo] subgraph Estacionamiento [Estacionamiento] SPV end </pre>	
Descripción	Ubicar el vehículo en alguna plaza disponible.
Actores	Usuario con vehículo, usuario con parqueadero.
PreCondición	Ambos usuarios deben estar registrados, y el parqueadero debe tener el estado disponible.
PosCondición	Se agrega el vehículo en el parqueadero y empieza a contabilizar el tiempo.
Escenarios	
Primario	<ul style="list-style-type: none"> - Selecciono un parqueadero disponible en el área. - El dueño del parqueadero recibe una notificación de un nuevo vehículo. - Una vez llega el vehículo ambos usuarios confirman la llegada de este.
Secundario	<ul style="list-style-type: none"> - Selecciono un parqueadero no disponible. - Mostrará un aviso diciendo que este no se encuentra disponible
Excepcionales	<ul style="list-style-type: none"> -No hay parqueaderos disponibles. - Le notificara que en esa área no es posible parquear su vehículo.

Figura 0.0. Descripcion de caso de uso Servicio de Parqueo de Vehiculo”

Caso uso: Parquear por tipo de vehículo	
<pre> graph LR Actor((User)) --- UC([Parquear por tipo de vehículo]) subgraph System [Estacionamiento] UC end </pre>	
Descripción	Clasificar los parqueaderos por tipo de vehículo soportado.
Actores	Usuario con vehículo, usuario con parqueadero.
PreCondición	Ambos usuarios deben estar registrados, el usuario con vehículo debe tener el tipo de vehículo registrado de igual manera el parqueadero el tipo de vehículo que pueden usarlo
PosCondición	Se mostrara si el parqueadero tiene plazas para el tipo de vehículo.
Escenarios	
Primario	<ul style="list-style-type: none"> - Selecciono un parqueadero disponible en el área. - Se mostrara si se puede usar este para su tipo de vehículo.
Secundario	<ul style="list-style-type: none"> - Selecciono un parqueadero no compatible con el vehículo - No permitirá registrar el parqueo y volverá a la búsqueda.
Excepcionales	<ul style="list-style-type: none"> - No hay parqueaderos disponibles para su tipo de vehículo. - Le notificara que en esa área no es posible parquear su vehículo.

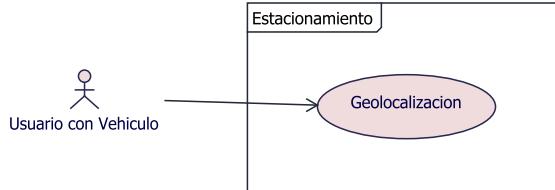
Figura 0.0. Descripcion de caso de uso ”Parqueo por Tipo de Vehiculo”

Caso uso: Consulta por Rango de Precios	
	<pre> graph LR Actor((User)) --- UC([Consultar por Rango de Precios]) subgraph System [Estacionamiento] UC end </pre>
Descripción	Mostrar solo los parqueaderos que se encuentren en el rango de precios seleccionado por el usuario
Actores	Usuario con vehículo
PreCondición	El usuario debe estar en el menú de búsqueda
PosCondición	Solo se verán los parqueaderos filtrados por precio.
Escenarios	
Primario	<ul style="list-style-type: none"> - Selecciono un rango de precios en el menú - El mapa se actualizara mostrando solo los parqueaderos que entran en ese rango
Secundario	<ul style="list-style-type: none"> - Selecciono in rango de precios en el menú -No aparece ningún parqueadero en ese rango. - Se mostrara un mensaje diciendo que en esa área no hay parqueaderos dentro de ese rango de precios
Excepcionales	<ul style="list-style-type: none"> - - El rango de precios seleccionado es de 0. - No se mostrara ningún parqueadero en el área.

Figura 0.0. Descripción de caso de uso Consulta por rango de precios

Caso uso: Consultas de Servicios	
<pre> graph TD Actor[Usuario con Vehículo] --> UseCase{Consultas de Servicios} subgraph System [Estacionamiento] UseCase end </pre>	
Descripción	Un sistema de consultas es aquel sistema digital que interactúa activamente con el usuario con dinámica conocida en relación con sus entradas y salidas permitiéndole hacer busquedas de todo tipo, en función de su necesidad a partir de un servicio de parqueo adquirido.
Actores	Usuario con vehículo
PreCondición	El Usuario debe haber adquirido un servicio de parqueo estar
PosCondición	Se podrá ver el resultado o la salida de la consulta hecha, relacionada con el servicio adquirido.
Escenarios	
Primario	<ul style="list-style-type: none"> - Selección de un servicio adquirido - Recibo un tipo de consulta de un servicio adquirido - Se recibe una consulta directa al usuario que presta el servicio
Secundario	<ul style="list-style-type: none"> - Se seleccionó una opción no disponible
Excepcionales	<ul style="list-style-type: none"> - Se seleccionó una opción sin haber solicitado previamente un servicio

Figura 0.0. Descripción de caso de uso Consulta de Servicios

Caso uso: Geolocalizacion real	
 <pre> graph LR UV[User con Vehiculo] --> G((Geolocalizacion)) subgraph Estacionamiento [Estacionamiento] G end </pre>	
Descripción	La Geolocalización es un sistema que determina la ubicación de un objeto en un entorno físico o virtual (Internet). A menudo, el objeto es una persona que quiere utilizar un servicio basado en la ubicación, mientras mantiene su privacidad.
Actores	Usuario con vehículo
PreCondición	<ul style="list-style-type: none"> - El Usuario debe haber registrado su vehículo - Haber seleccionado un servicio previamente - Haber dado los requeridos permisos de ubicacion
PosCondición	<ul style="list-style-type: none"> - Se podra seleccionar o visualizar directamente desde el mapa la ubicacion deseada con las opciones previamente seleccionadas.
Escenarios	
Primario	<ul style="list-style-type: none"> - Se genera un mapa de la zona con los requerimientos seleccionados - Se gestiona una solicitud de servicio
Secundario	<ul style="list-style-type: none"> - Se seleccionó una zona no disponible para visualizacion geo- espacial - No hay disponibilidad para el servicio solicitado
Excepcionales	<ul style="list-style-type: none"> - Se selecciono la geolocalizacion sin haber solicitado previamente un servicio - No fueron autorizados los permisos (ubicacion) requeridos para el funcionamiento de la herramienta

Capítulo 5

Interacciones

5.1. Introducción

El diagrama de secuencia es un tipo de diagrama de interacción cuyo objetivo es describir el comportamiento dinámico del sistema de información haciendo énfasis en la secuencia de los mensajes intercambiados por los objetos.

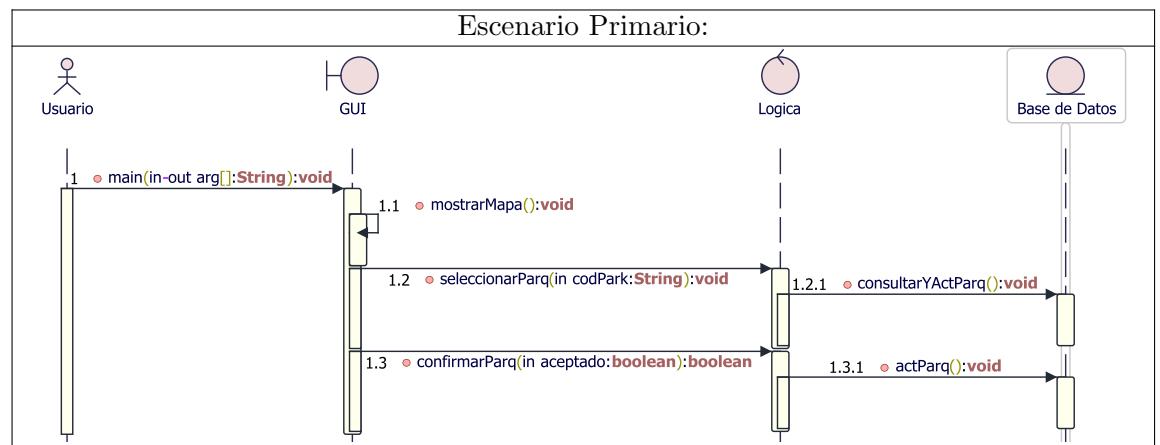
En este caso se va a representar cada caso de uso planteado anteriormente en un diagrama de secuencia, y cada caso de uso con sus respectivos escenarios primario, secundario y excepcional que también fueron expresados anteriormente.

Los diagramas fueron elaborados con ayuda de la herramienta Coloso y están basados en las descripciones de los casos de uso propuestos en la sección anterior.

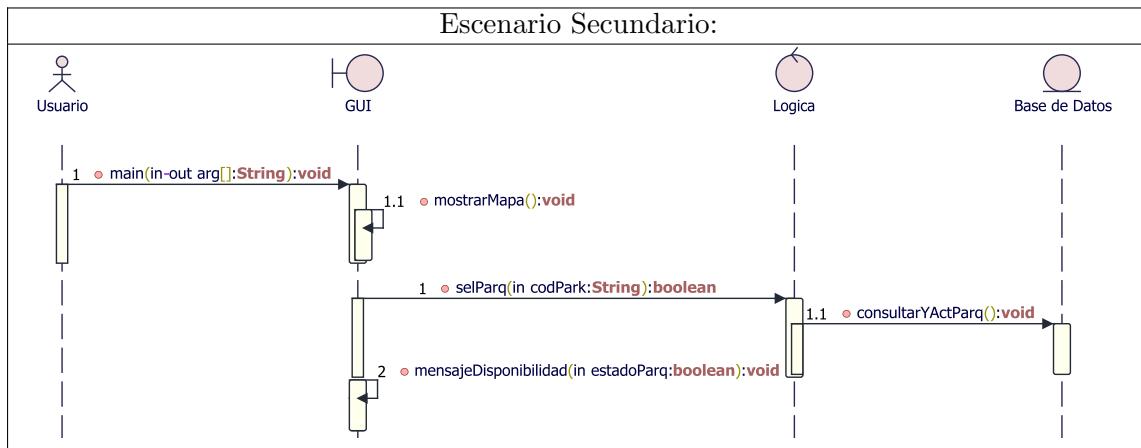
5.2. Diagramas de Secuencia

CASO DE USO: Servicio de Parqueo de Vehiculo

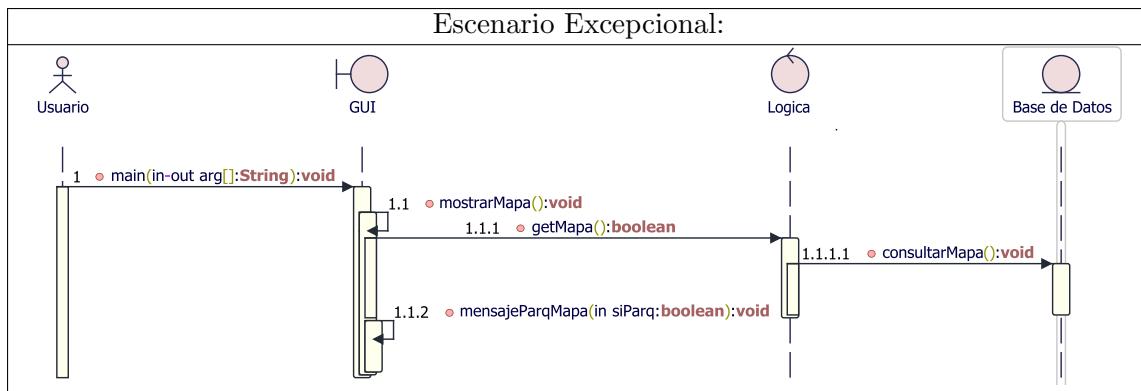
El siguiente diagrama inicia con el usuario accediendo al programa en esta fase este ya ha iniciado sesión, por el primer mensaje se procede a mostrarle el mapa al usuario y una vez seleccione un parqueadero en el mapa el cual causa que se envíe un mensaje a la parte lógica con el código del parqueadero seleccionado, bien ahora la parte logica envia el mensaje a la base de datos solicitando información del parqueadero. Ahora el usuario confirma el parqueadero lo cual de igual manera envia un mensaje a la lógica para acceder y modificar la base de datos respecto al parqueadero.



Ahora para el caso secundario necesitamos un retorno para que el usuario conozca el estado del parqueadero seleccionado, el cual siguiendo el paso de mensajes que consulta la información del parqueadero almacenada en la base de datos, este retornara un dato que se tomará como booleano para por medio de un mensaje cíclico muestre al usuario si está disponible este parqueadero.

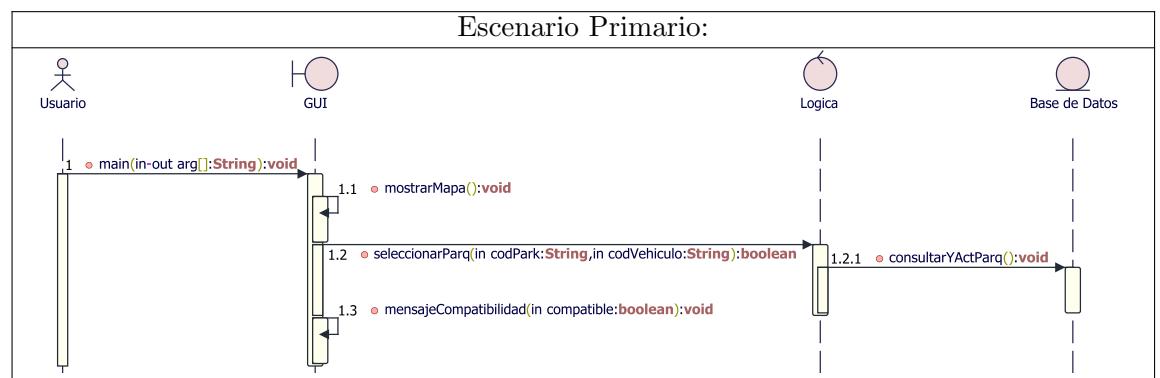


En el caso excepcional se parte de que antes de mostrar el mapa al usuario este se obtiene por un mensaje enviado a la lógica para obtenerlo de la base de datos el cual obtendrá el estado de los parqueaderos ubicados en el mapa para que el usuario sepa cuales están disponibles.

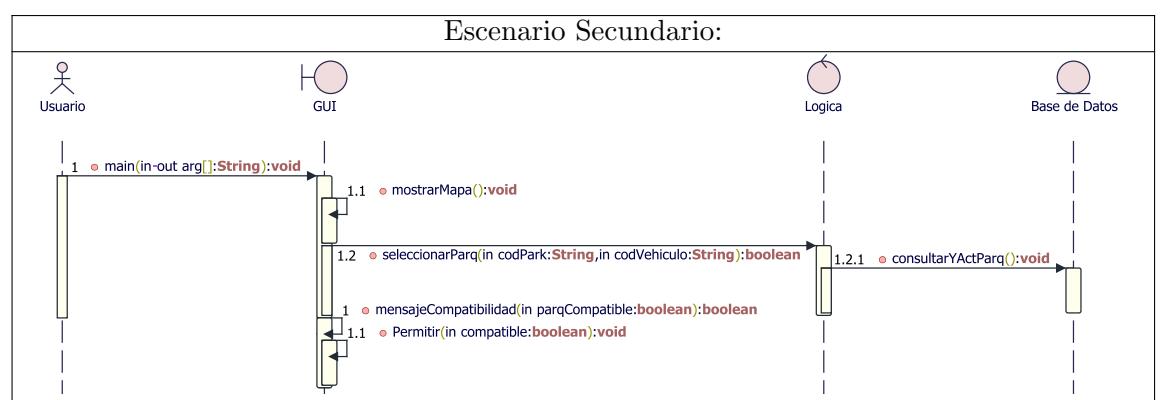


CASO DE USO: Parqueo por tipo de Vehículo

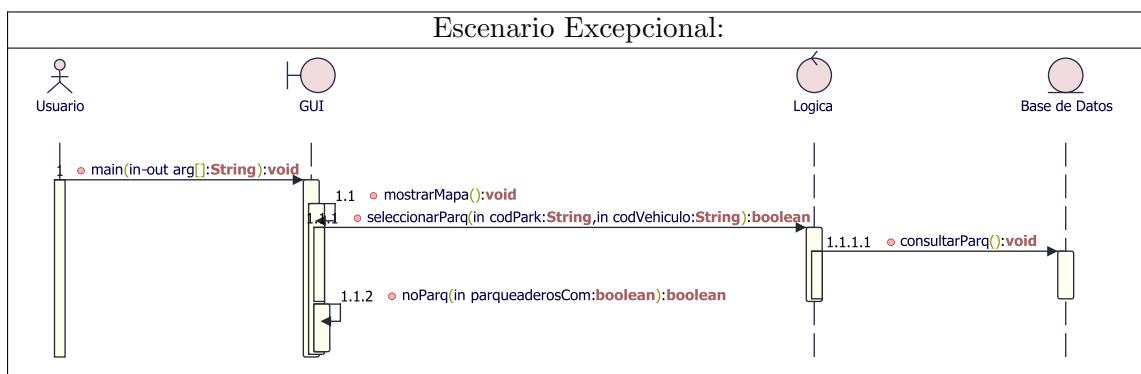
Este parte del mismo punto que el caso anterior por ende el paso de mensajes es el mismo, ahora bien teniendo en cuenta que el usuario necesita saber si el parqueadero seleccionado es compatible con el tipo de vehículo esta información se obtendrá de la base de datos la cual me verificará por medio de un retorno booleano si es compatible lo cual se mostrará al usuario por medio de un mensaje cíclico.



Teniendo en cuenta la secuencia anterior el paso de mensajes no es muy diferente lo que se agrega para tratar este caso es por medio de un mensaje cíclico que el usuario verá en forma de ventana emergente que impide que se pueda registrar en el parqueadero en caso de que este no sea compatible.

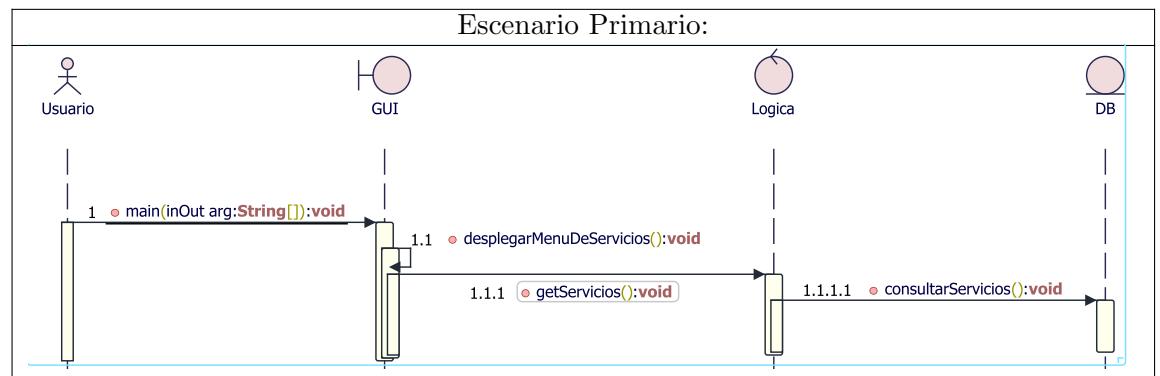


Ahora bien para este caso se debe tratar desde que se muestra el mapa al usuario, este antes de mostrar el mapa de la base de datos pasa por un filtro que se da por el paso de mensajes el cual envia un código que es el tipo de vehículo del usuario para determinar su compatibilidad, y en caso de que retorne 0 parqueaderos que sean compatibles se enviará un mensaje cíclico el cual le notificará al usuario que no se encuentran parqueaderos compatibles en el área.

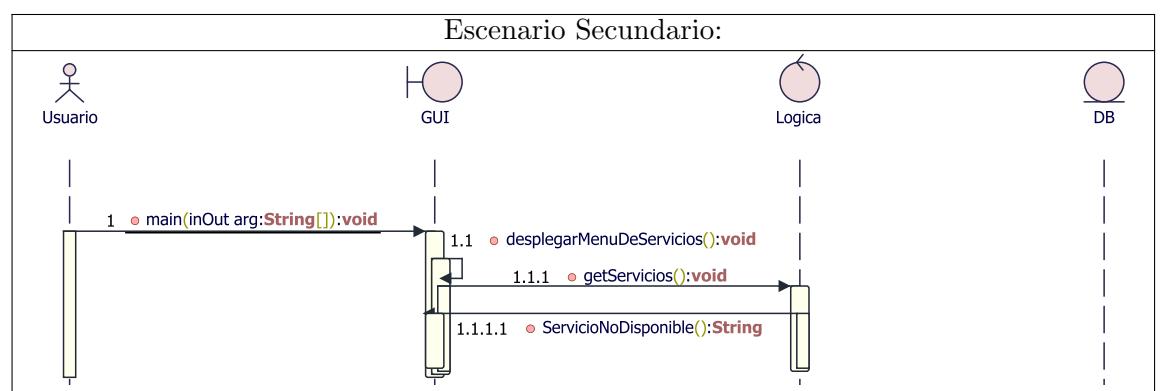


CASO DE USO: Consulta de Servicios

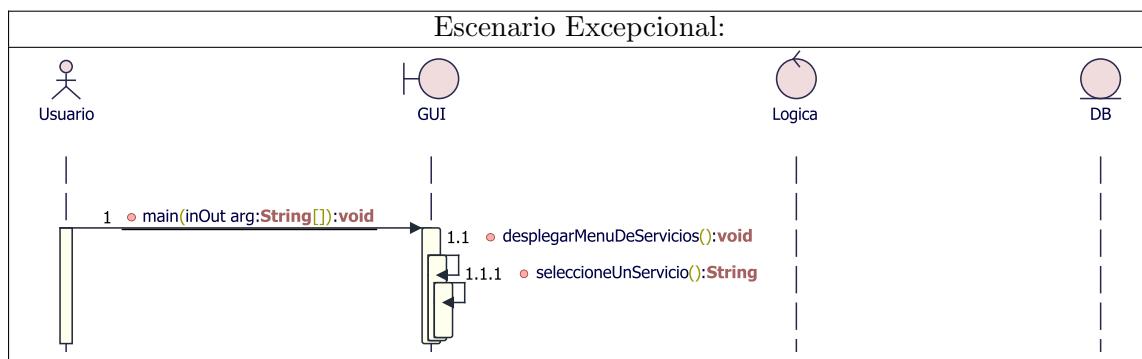
En el Escenario Primario del caso de uso de Consultas de Servicios, se plantea una precondicion en la cual el usuario ya debe haber adquirido un servicio de parqueo, por lo que podra desplegar un menu de servicios desde la Interfaz del programa, y podra consultar cualquiera de estos servicios que se encuentren disponibles para el parqueo que ya ha sido adquirido.



En el Escenario Secundario, se plantea la posibilidad de que el usuario selecciono un servicio que es posible que el espacio que adquirio antes ofrecia, pero en este momento ya no se encuentra disponible, como una renovacion de alquiler un dia que el parqueo ya se encuentra ocupado.

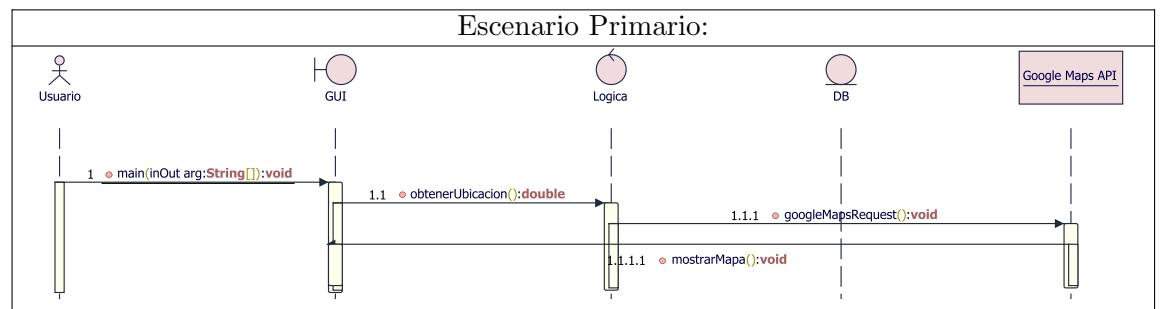


En el Escenario Excepcional, es posible que un usuario ya registrado y logeado, haya podido seleccionar la opcion de consulta de servicios sin previamente haber adquirido un servicio de parqueo.

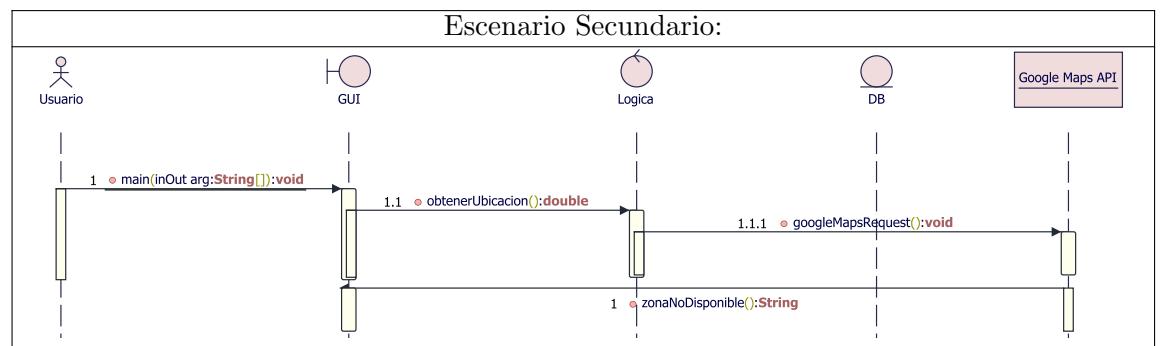


CASO DE USO: Geolocalizacion

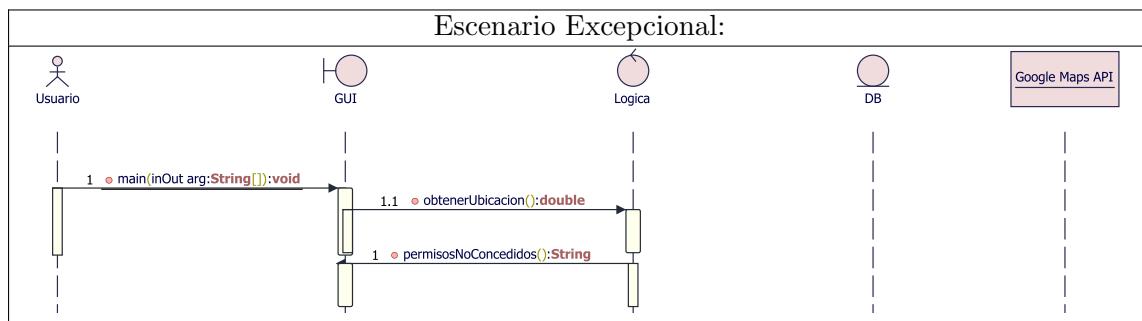
En el Escenario Primario del caso de uso de Geolocalizacion, se tiene en cuenta que el usuario previamente se ha registrado y logeado, y ha concedido los permisos necesarios para poder desplegar la interfaz de geolocalizacion, interfaz en la que el usuario podra solicitar la ubicacion en tiempo real de un espacio que quiera alquilar, y la API de google maps se lo muestre.



En el Escenario Secundario se plantea una situacion en la que un servicio especifico seleccionado por el usuario no se encuentra disponible en la zona solicitada.

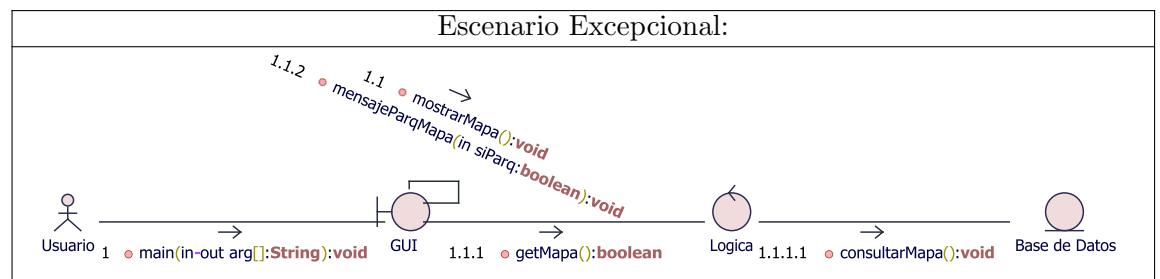
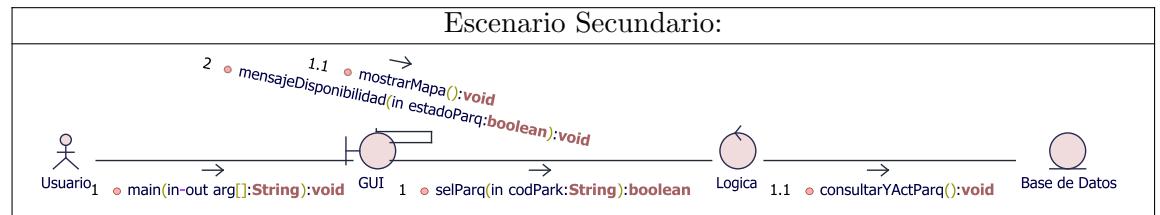
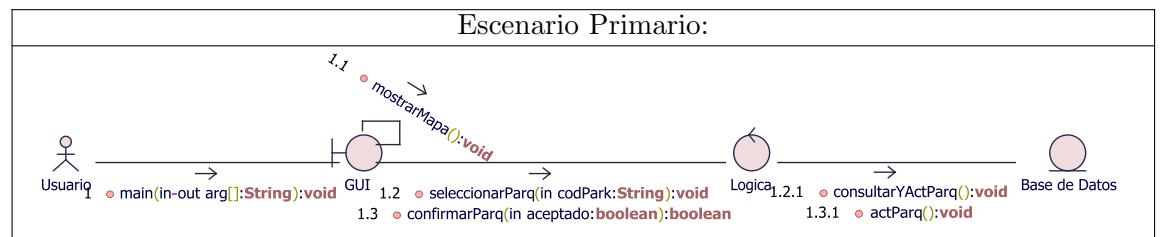


En el Escenario Excepcional es posible que el usuario no haya concedido los permisos necesarios para el correcto funcionamiento de la API de google maps

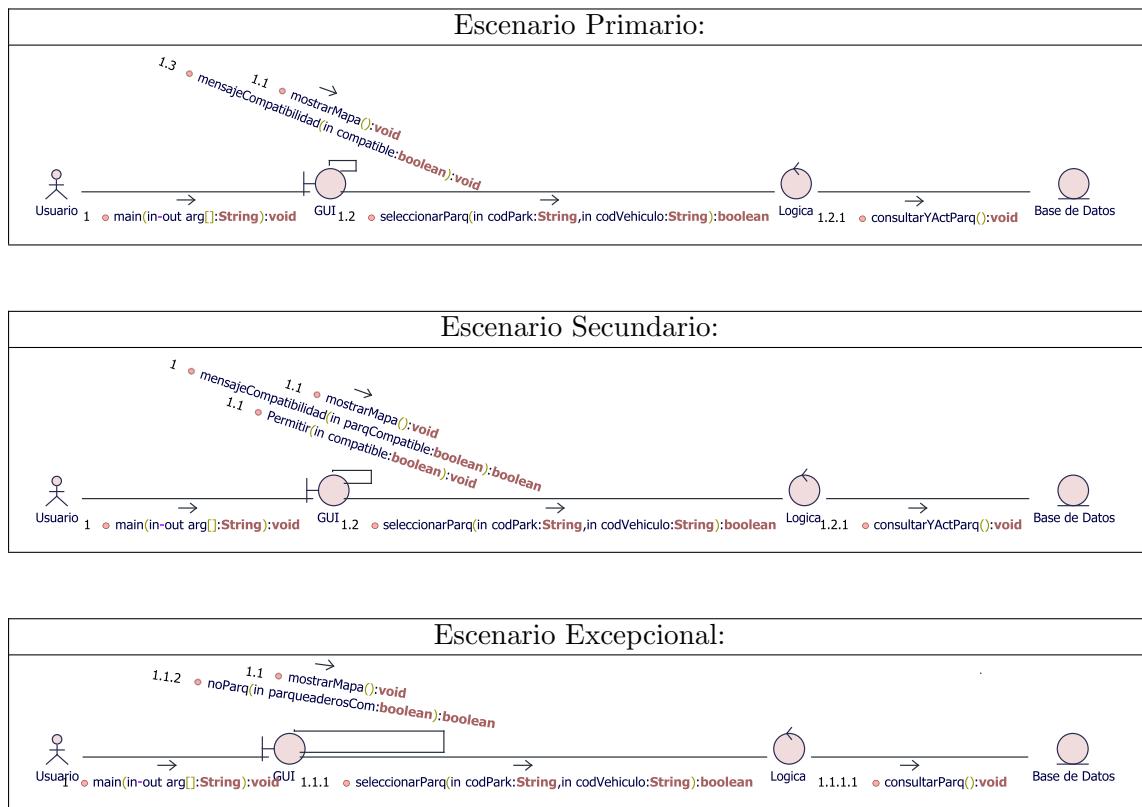


5.3. Diagramas de Comunicación

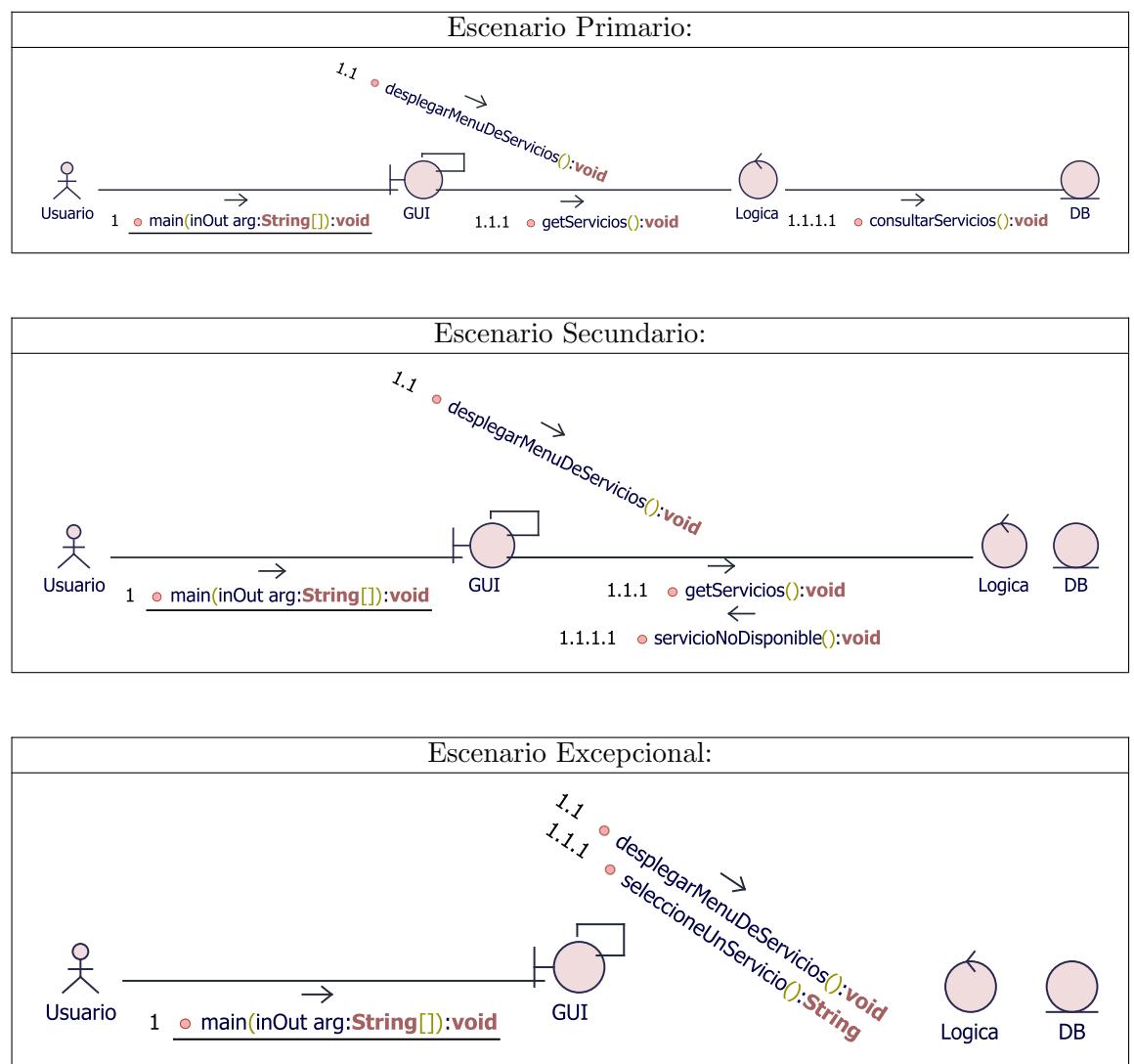
CASO DE USO: Servicio de Parqueo de Vehiculo



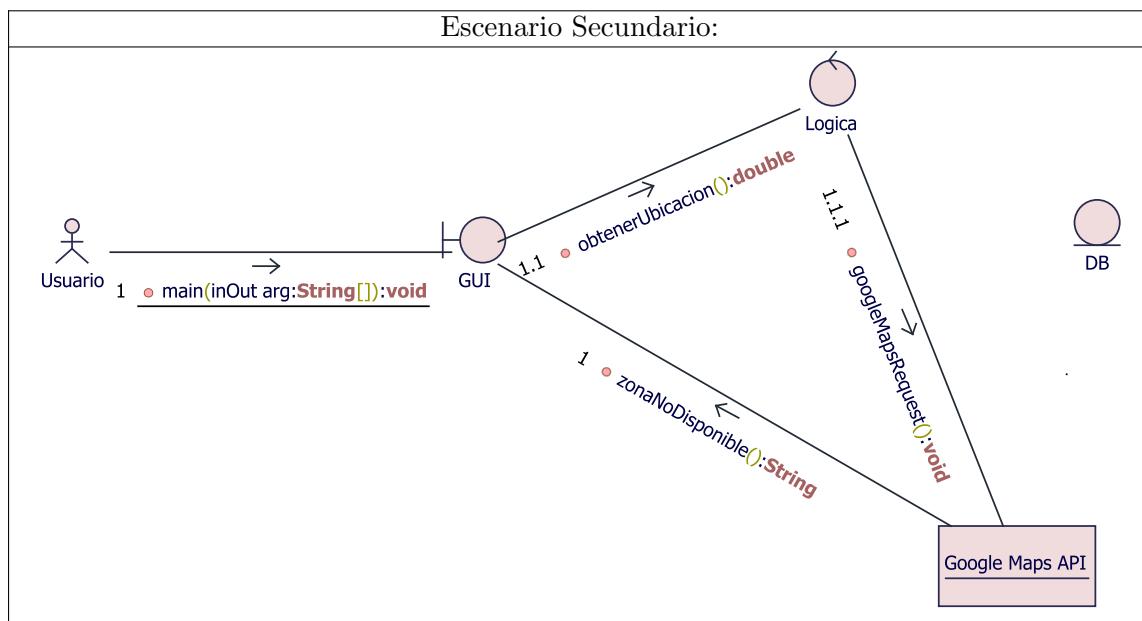
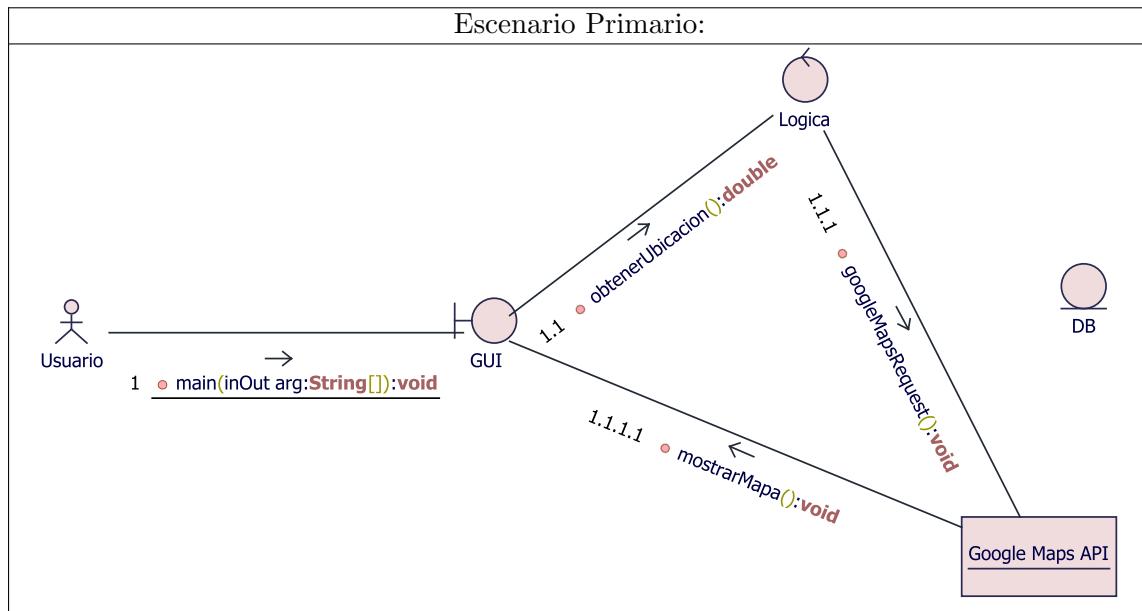
CASO DE USO: Parqueo por tipo de Vehículo

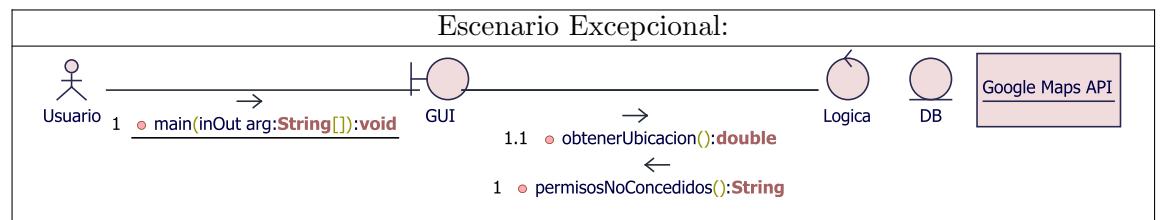


CASO DE USO: Consulta de Servicios



CASO DE USO: Geolocalizacion





Capítulo 6

Clases

6.1. Introducción

Introducción

Los diagramas de clases son diagramas estructurales del lenguaje unificado de modelado o UML (unified modeling language). Este lenguaje de modelado visual es un estándar ISO para visualizar los sistemas de la programación orientada a objetos, aunque también permite visualizar procesos de negocio. Con ayuda de elementos gráficos, UML muestra los estados de los sistemas y describe las interacciones entre los elementos que lo componen.

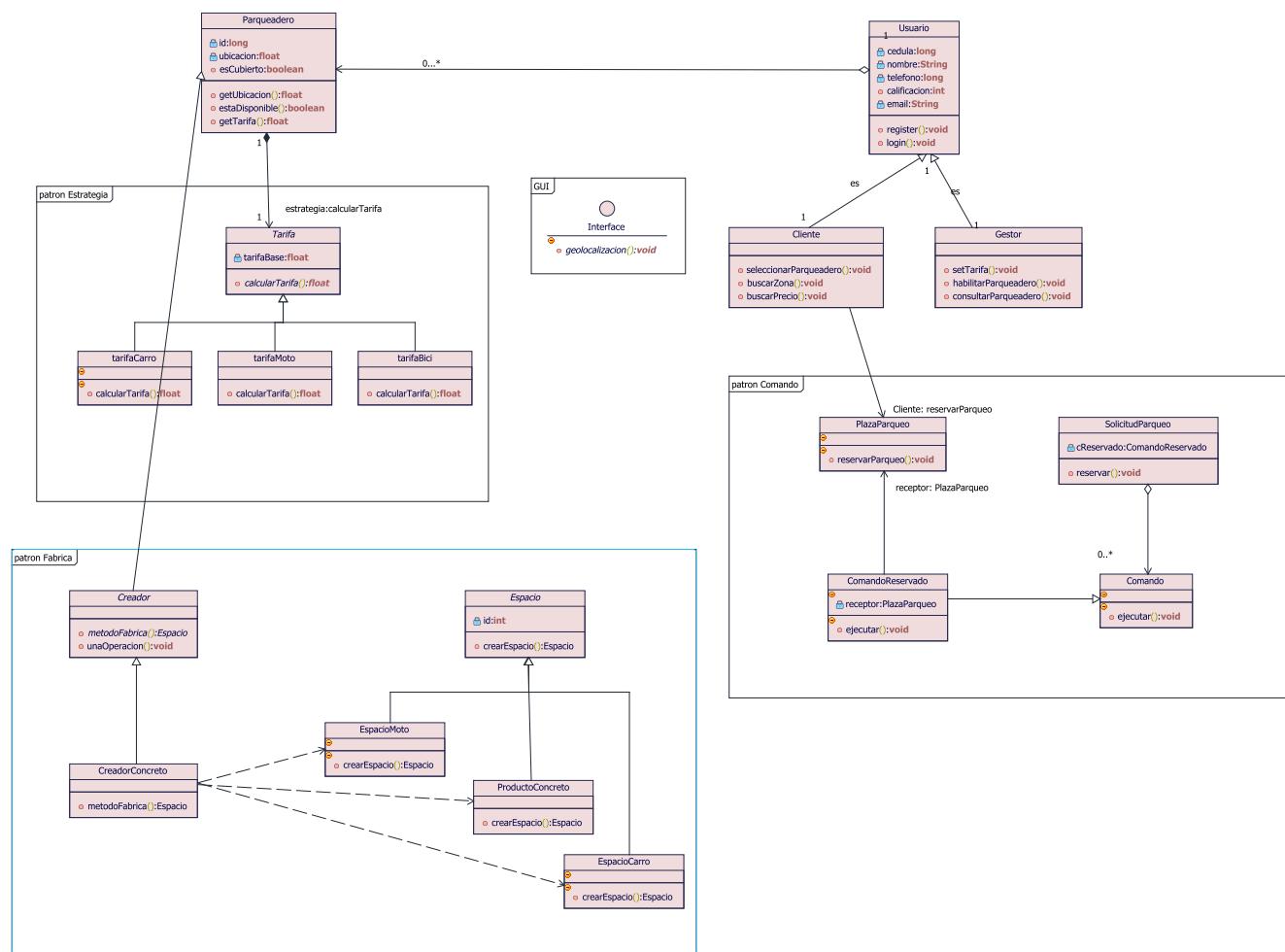
Los diagramas de clases son diagramas estructurales del lenguaje unificado de modelado o UML (unified modeling language). Este lenguaje de modelado visual es un estándar ISO para visualizar los sistemas de la programación orientada a objetos, aunque también permite visualizar procesos de negocio. Con ayuda de elementos gráficos, UML muestra los estados de los sistemas y describe las interacciones entre los elementos que lo componen.

6.2. Descripción

En esta oportunidad diseñamos un diagrama de clase para nuestro proyecto utilizando la herramienta Coloso, donde diseñamos una estructura de lo que seria nuestro proyecto, se crearon las entidades base de cliente y gestor, que heredan de una clase Usuario, al igual se creo una entidad Parqueadero, con ciertos atributos y operaciones. Se implementaron tres patrones de diseño Gof, que son, Strategy, Command, y Factory Method, el strategy esta pensado para calcular una tarifa designada al parqueo, dependiendo del tipo de vehiculo que se vaya a estacionar, donde el algoritmo base es “calcularTarifa”, el patron Factory Method esta pensado para la creacion de los distintos espacios diferenciados por el tipo de vehiculo que van a alojar, por lo tanto, se crea una factoria de espacio de parqueo, y de ella heredan los distintos tipos de espacios que se van a crear, ya que cada uno tiene unas caracteristicas distintas, y por ultimo se penso en el patron command como una forma de verificar si un espacio de parqueo se encuentra actualmente disponible, o , ocupado, y ya terminando con el diseño se agrego una entidad interface que tiene una operacion que activa o muestra la posicion en un mapa de geolocalizacion (con una API de google) correspondiente a la busqueda del usuario.

6.3. Diagrama de Clase UML

Para este diagrama se separaron cuatro frames en particular para una mejor interpretación, los tres frames más grandes representan los tres patrones de diseño que se planean utilizar en el proyecto, y que van a ser explicados en la siguiente sección, y un cuarto frame que representa la GUI del proyecto, que va a contar con una interfaz gráfica principal del presente proyecto.



6.4. Diagrama de Clase UML: Patrones de Diseño

6.4.1. Patrón: Estrategia

Este patrón define un conjunto de algoritmos, encapsula cada uno de ellos y los hace intercambiables. Permite que el algoritmo pueda variar independientemente de los clientes que lo utilicen.

Esto quiere decir que nuestros objetos deben estar preparados para afrontar diversos contextos sin cambiar las interacciones de estos con el cliente.

Para este caso concreto, nuestro contexto base va a ser la entidad parqueadero, y nuestro algoritmo principal es calcularTarifa, así el cliente, dependiendo el tipo de parqueo que seleccione, va a pedir calcular una tarifa determinada para el tipo de vehículo que halla seleccionado.

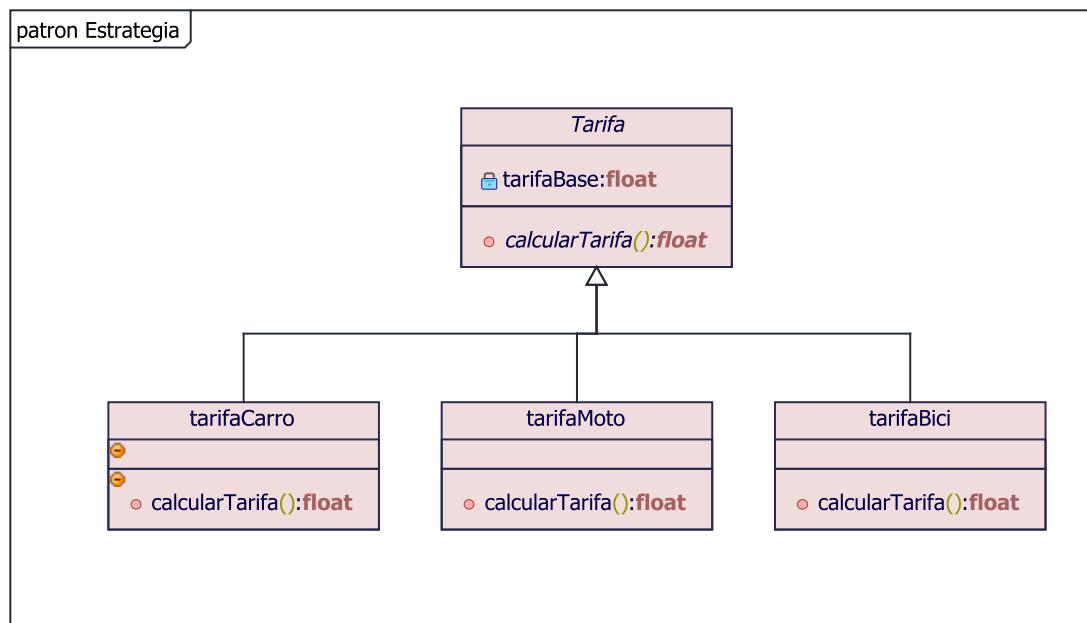


Figura 0.0. En este caso nuestro contexto va a ser "Parqueadero" (Se puede ver en el diagrama de clases general) y nuestra Interface del Algoritmo va a ser *calcularTarifa*", el cual es un algoritmo que varia en función del vehículo que se vaya a estacionar

6.4.2. Patrón: Método Fábrica

Factory Method permite la creación de objetos de un subtipo determinado a través de una clase Factory. Esto es especialmente útil cuando no sabemos, en tiempo de diseño, el subtipo que vamos a utilizar o cuando queremos delegar la lógica de creación de los objetos a una clase Factory. El patron factory method nos sirve para la creacion de los espacios teniendo en cuenta si el vehiculo es una bicicleta, creara un espacio para una bicicleta, si entra un carro un espacio de carro y asi dependiendo el requerimiento solicitado.

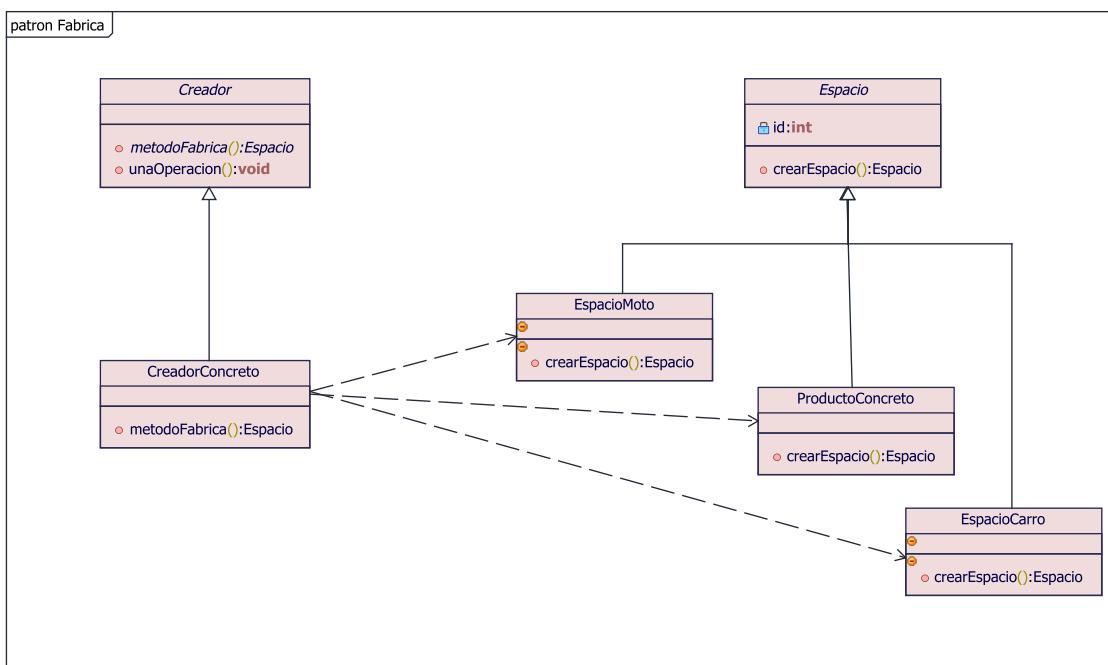


Figura 0.0. Patrón Factory Method para la creacion de distintos espacios de estacionamiento

6.4.3. Patrón: Comando

El patrón de diseño Command es muy utilizado cuando se requiere hacer ejecuciones de operaciones sin conocer realmente lo que hacen, estas operaciones son conocidas como comandos y son implementadas como una clase independiente que realiza una acción muy concreta, para lo cual únicamente recibe un conjunto de parámetros para realizar su tarea. Se usa el patrón command para recibir las peticiones de los usuarios denominados clientes de manera encapsulada, esto con el objetivo de separar la lógica de la interfaz, en este patrón con el comando reservado se envía la petición de reservar una plaza en el parqueadero previamente seleccionado, esto permitirá controlar las peticiones y decidir si es disponible aparcar o no en el sitio seleccionado.

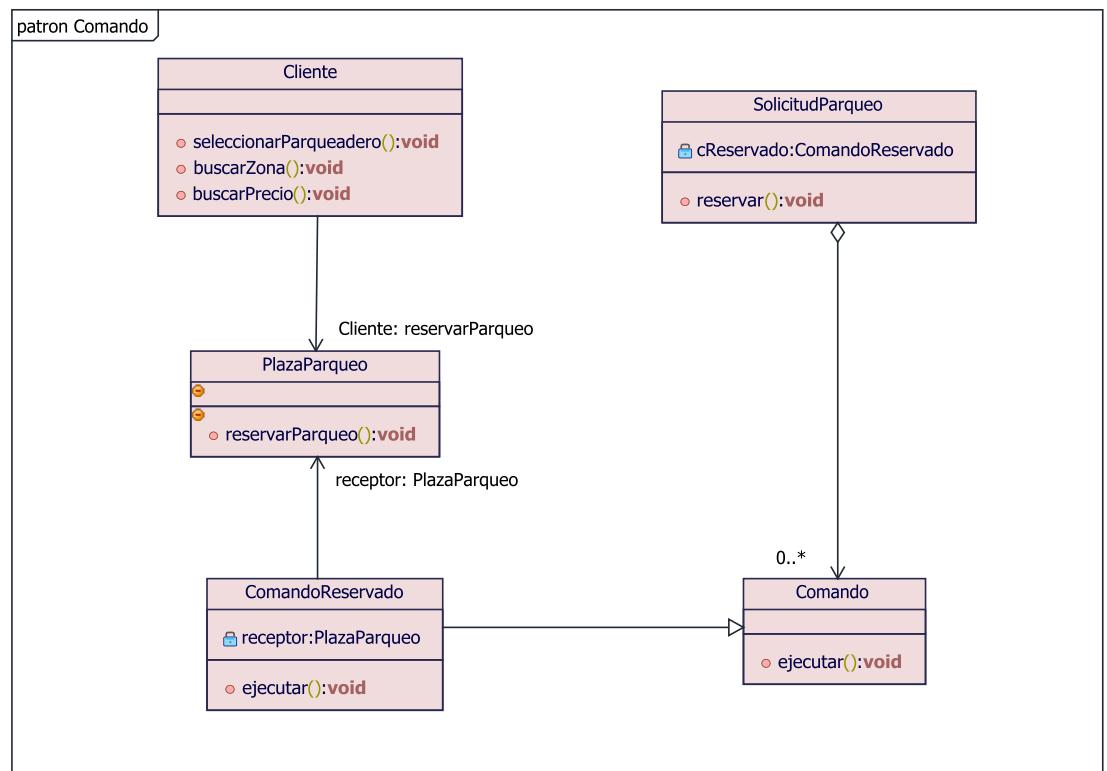


Figura 0.0. Patrón Comando para la validacion de disponibilidad de un espacio de parqueo concreto

Capítulo 7

Estados

7.1. Introducción

El diagrama de estado se usa para dar forma al comportamiento de un objeto, de una clase. Se representa la secuencia de estados que un objeto de la clase tiene durante su vida, según las acciones que van sucediendo.

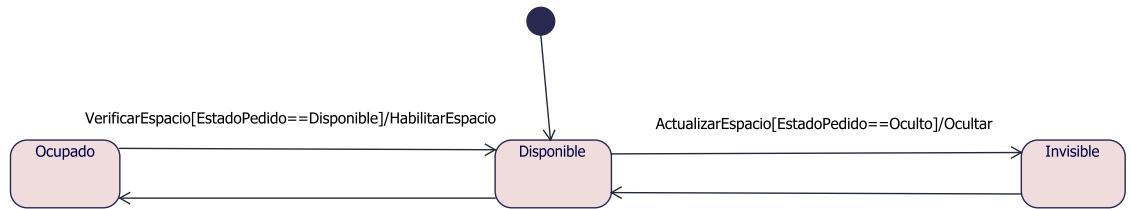
Las características de los diagramas de estados se puede resumir en estas tres:

- Un estado es la representación de un objeto en los diferentes espacios de tiempo que le van sucediendo. Cuando hablamos de estado estamos hablando de los diferentes estados que puede tener un objeto.
- Cada evento representa algo que hace que nuestro objeto pueda cambiar.
- Existen unas líneas que llamamos líneas de transición, su finalidad es describir el movimiento de un estado a otro. A estas líneas le ponemos el nombre del evento que origina la transición.

7.2. Diagrama de Estados

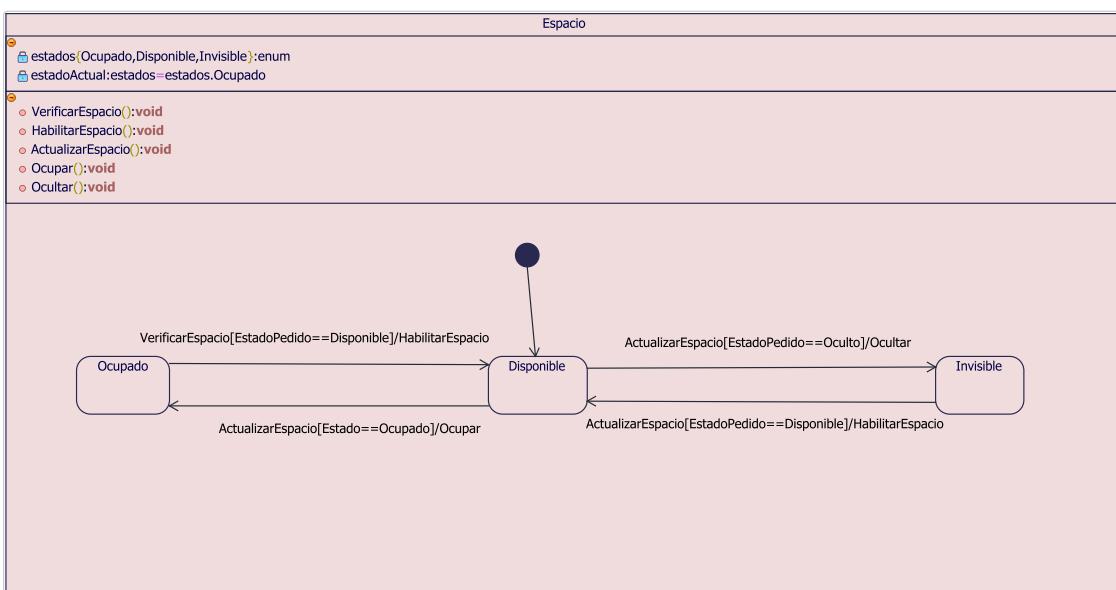
A continuacion, esta la ejemplificacion de un diagrama de estados aplicado a nuestro proyecto de parqueos, donde se van alterando los distintos estados de un parqueo, que puede tener unicamente tres estados en concreto, el estado base de un parqueadero es el estado "Disponible", que indica que el espacio de parqueo es apto para ser ocupado por un vehiculo de un tipo particular, es decir, el espacio puede ser tomado en alquiler por un usuario. Un segundo estado que es el de ".cupado", le señala al usuario que el espacio de parqueo no puede ser tomado en uso por un tiempo determinado. Y, por ultimo, esta el estado "Invisible", que señala que un parqueadero no esta ni Disponible, ni Ocupado, simplemente el dueño de este espacio decide no mostrarlo como espacio seleccionable, por lo que sus datos tampoco son visibles al resto de usuarios, en otras palabras, el dueño puede ocultar su parqueadero si simplemente no quiere alquilarlo por un tiempo.

A continuación esta el diagrama de estados de nuestro proyecto de parqueos, donde, como se menciono anteriormente, solo se incluyen 3 estados, que corresponde a cualquier tipo de espacio de parqueo:



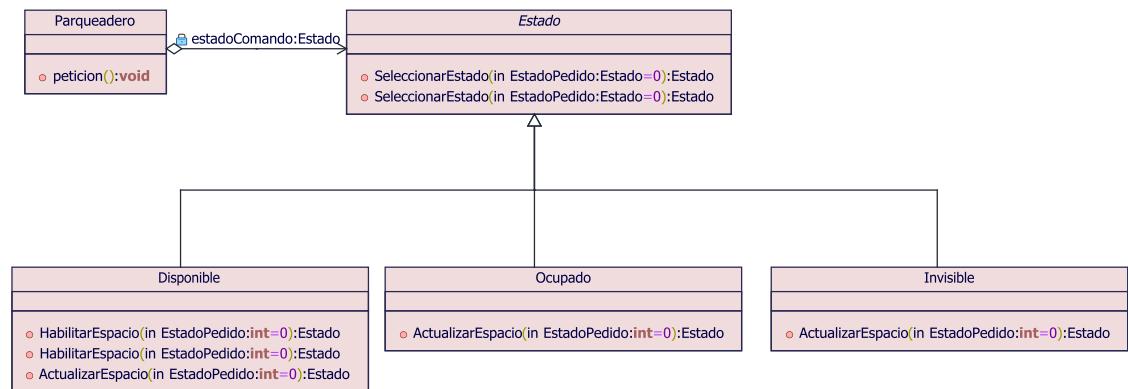
7.2.1. Diagrama de Clase

Tanto el diagrama anterior, como los siguientes, fueron elaborados gracias a la herramienta de diseño Coloso, la cual, al momento de construir el diagrama de estados con su respectivo Disparador (Metodo que da accion al cambio de estados), condicional (sentencia que se tiene que cumplir para efectuar el cambio) y efecto (metodo que realiza el cambio de estado como tal). La herramienta coloso ademas de construir el diagrama de estados con estos elementos, genera un diagrama de clase con los metodos y atributos correspondientes a dicho diagrama:



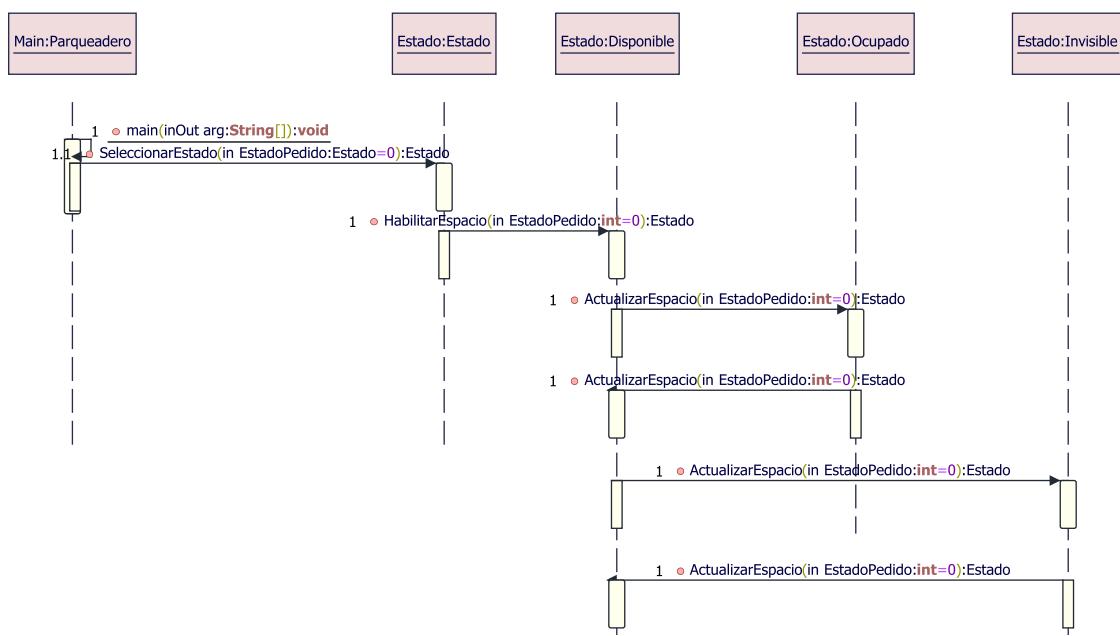
7.2.2. Diagrama de Clase: Patrón Estado

Para una mejor conceptualización de lo que representa nuestro diagrama de estados, se diseño un diagrama de clase con el patrón Estados, que posee una clase “contexto”, que es la que requiere dichos cambios de estados, y que en nuestro caso es la clase “Parqueadero”, una clase abstracta “Estados” que representa la entidad que maneja los estados, con sus respectivos métodos y atributos, correspondientes al diagrama anterior, y por último, los tres estados que se van a manejar y que son nuestros “estados concretos”, que heredan los métodos de la clase Estados.



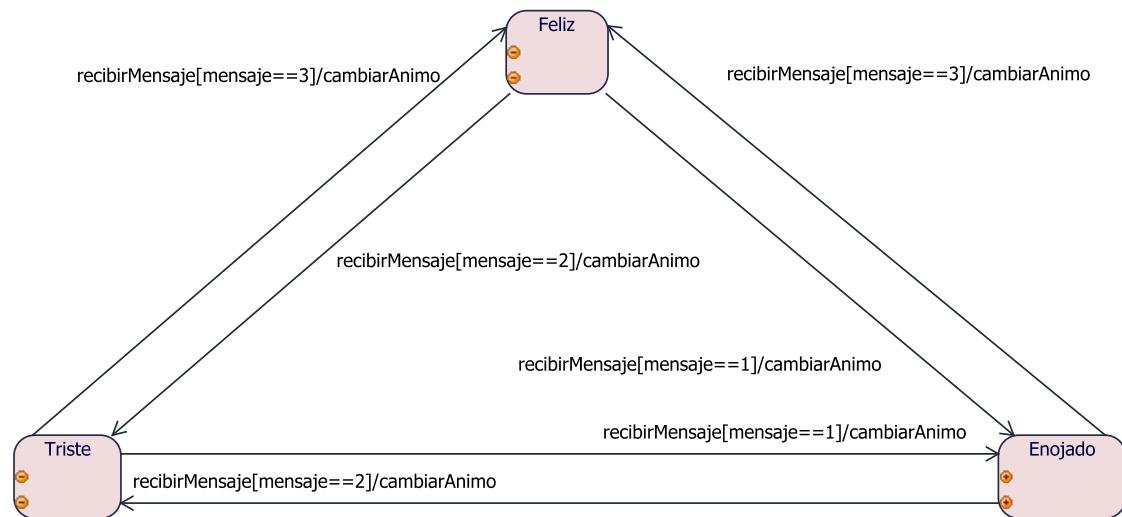
7.2.3. Diagrama de Secuencia

Por ultimo, para realizar nuestros cambios de estados de una manera concreta, y sin cometer errores, se deben seguir una serie de pasos en el programa, por lo tanto, se diseña el siguiente diagrama de secuencia que representa la serie de pasos que debe seguir el programa para ir de un estado a otro, ademas, como se puede apreciar en el diagrama de estados, no todos los estados estan conectados por una transicion, es decir, no se puede pasar de un estado a otro arbitrariamente, esto se puede ver tambien en el diagrama de secuencia:



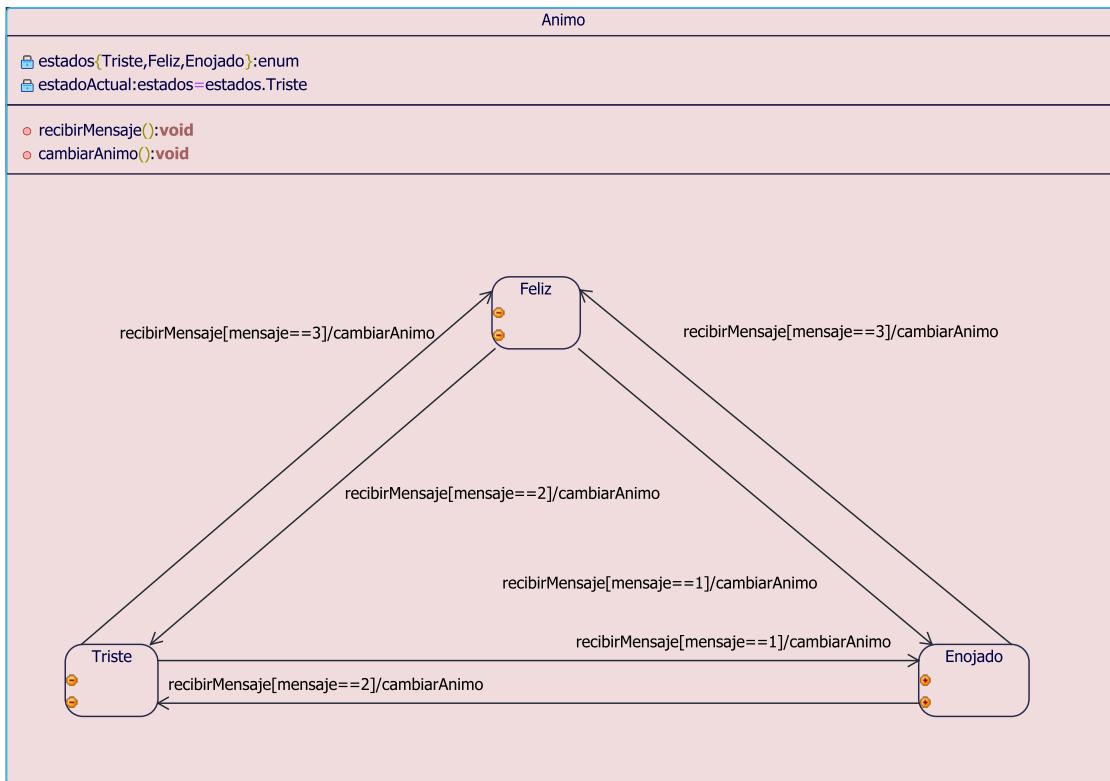
7.3. Anexo 1:

Una persona puede tener varios estados de animo en general, pero solo puede tener uno a la vez, es decir, debe haber un cambio de estado en la persona cada vez que quiera experimentar uno diferente. Para ver un ejemplo de lo anterior enfocado a la Ingeniería de Software, se creó el siguiente diagrama de estados de una persona con 3 estados de animo:



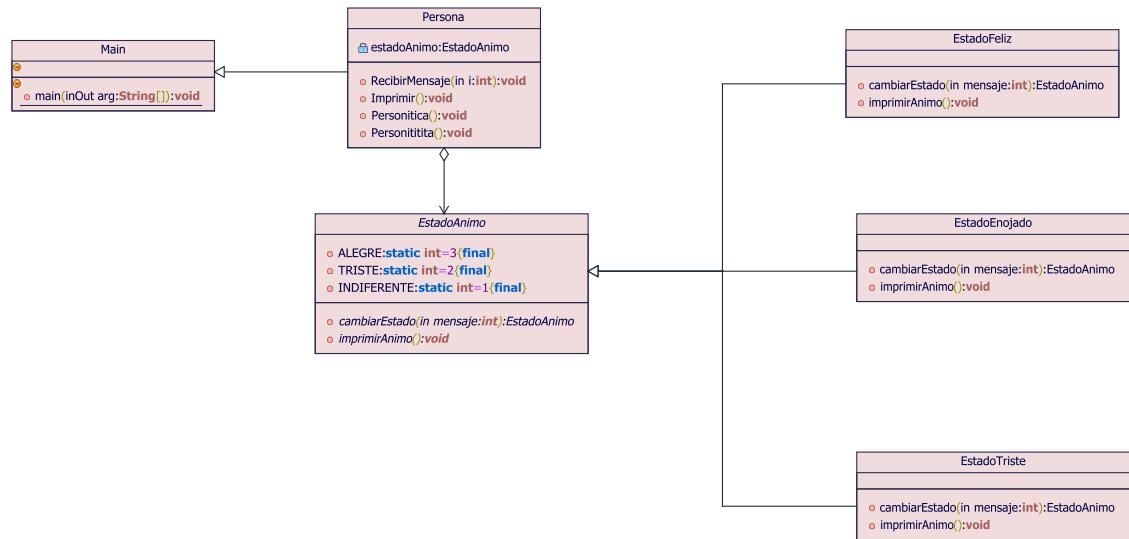
7.3.1. Anexo 1: Diagrama de clase

Este diagrama de estados con su respectivo diagrama de clases, con sus métodos y atributos correspondientes:



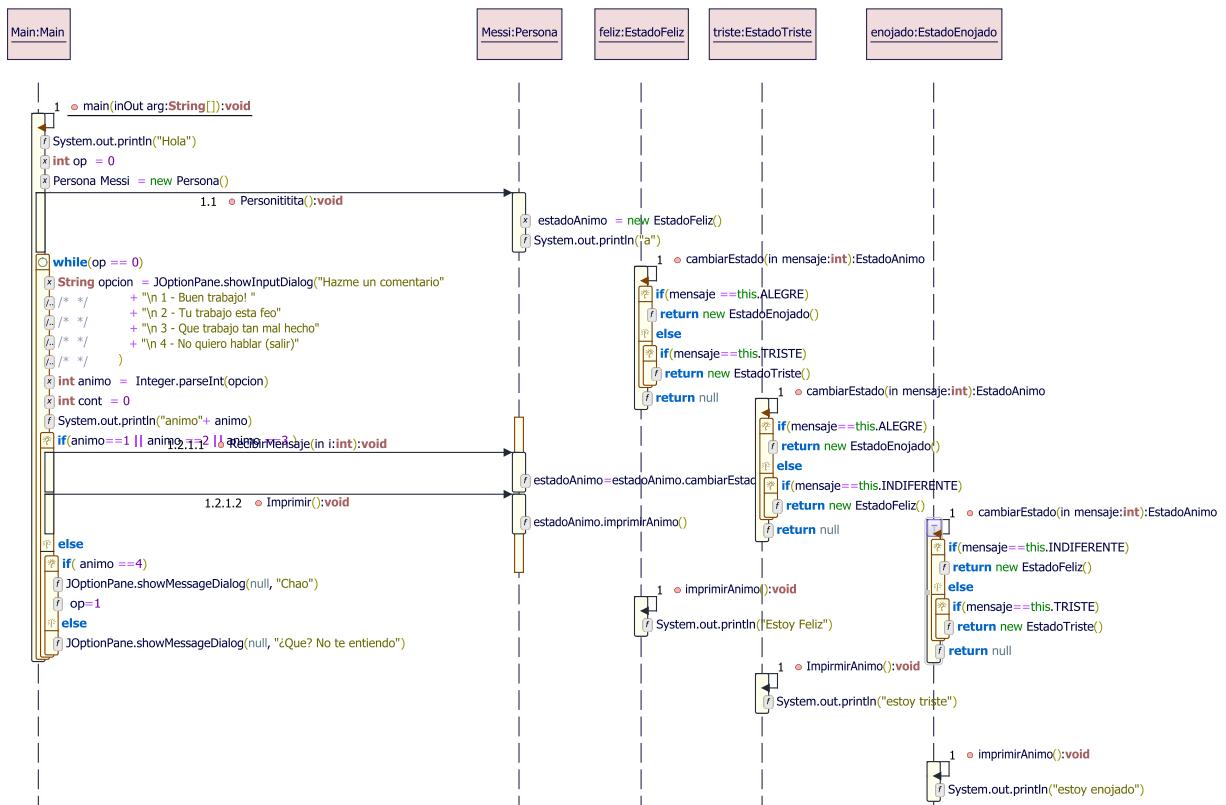
7.3.2. Anexo 1: Diagrama de Clase del Patrón Estado

El patron Estado nos ayuda a entender mejor el funcionamiento de un diagrama de estados, nosotros hemos diseñado uno para el ejemplo de los estados de animo de una persona:



7.3.3. Anexo 1: Diagrama de Secuencia Extendido

Una vez ponemos a prueba lo aprendido, y partiendo de nuestro diagrama de clase del Patrón Estado, hemos construido un diagrama de secuencia extendido en la herramienta de diseño y programación Coloso, donde es posible ejecutar dicho diagrama de secuencia, que principalmente es el diseño de un software que varía los estados de ánimo de una persona, entre triste, feliz y enojado, tal y como se ha visto en los anteriores diagramas:



Capítulo 8

Actividades

8.1. Introduccion

contenido.....

8.2. WorkFlow

Capítulo 9

Componentes

9.1. Introducción

contenido.....

Capítulo 10

Sistemas

10.1. Introducción

contenido.....

Capítulo 11

Nodos

11.1. Introducción

contenido.....

Parte III

CIERRE

Capítulo 12

Conclusiones

Bibliografía

- [1] R. Iakovlev, I. Vatamaniuk, and D. Malov. Architecture transformation of the corporate information providing system for a scientific organization. In *2019 12th International Conference on Developments in eSystems Engineering (DeSE)*, pages 873–878, 2019.
- [2] S. Iram, D. Al-Jumeily, and J. Lunn. An integrated web-based e-assessment tool. In *2011 Developments in E-systems Engineering*, pages 271–275, 2011.
- [3] T. Rais Castro and S. Nice Alves de Souza. Graphic logical desing for ordb. *IEEE Latin America Transactions*, 11(4):1097–1103, 2013.
- [4] Booch G Rumbaugh J, Jacobso I. *The Unified Modeling Language Reference Manual, Second Edition*.