SAD

para

Proyecto 2

Versión 1.0

Preparado por

Marlon Reyes Montero
Gerald Morales Alvarado
Jose Villalobos Martinez

23 de Noviembre del 2018

Revision History

Date	Version	Description	Author
<07/10/18>	<1.0>	Finalización del desarrollo del sistema.	Marlon Reyes Montero Gerald Morales Alvarado Jose Villalobos Martinez

Table of Contents

Software Architecture Document

1. Introduction

Esta sección del proyecto tiene como fin explicar brevemente como está diseñado el proyecto, qué elementos se van a utilizar para explicar los diferentes apartados y lógica del sistema así como diagramas de arquitectura, casos de uso, paquetes y demás para proporcionar una visión clara de la arquitectura del sistema a implementar.

1.1 Purpose

Este documento proporciona una descripción arquitectónica completa del sistema, utilizando una serie de vistas arquitectónicas diferentes para representar diferentes aspectos del sistema. Está pensado para capturar y transmitir las importantes decisiones arquitectónicas que se han tomado en el sistema.

1.2 Scope

El presente documento aplica para el proyecto 2, solicitado para el curso de diseño de software impartido por la Prof.Ing.Ericka Solano Fernández, que consiste en la actividad de construcción del modelo y programación de una arquitectura para la propuesta presentada.

Como herramientas representativas, se va a contar con diagramas de casos de uso, de paquetes, de arquitectura de software y de proceso; esto para proporcionar información visual comprensible del sistema.

Además, el desarrollo del sistema propuesto se ve influenciado por los requerimientos necesarios para utilizar buenas prácticas de diseño de software, más la aplicación de patrones de diseño para crear un aplicación con cada característica propuesta por la docente.

1.3 Definitions, Acronyms, and Abbreviations

DTO: “ Data Transfer Object”. Patrón que utiliza un objeto que transporta información entre procesos de manera que se reduzca la cantidad de métodos.

DAO: “Data Access Object”. Patrón utilizado para separar datos con un acceso de bajo nivel de servicios de negocio de alto nivel.

MVC: Model - View -Controller: Metodología o patrón de diseño que busca relacionar eficientemente la interfaz de usuario con los modelos de datos inferiores.

POO: Programación Orientada a Objetos: Modelo de lenguaje de programación organizado alrededor de objetos en vez de acciones y en datos en vez de lógica.

Objeto: En programación orientada a objetos, son unidades de código que son derivadas del proceso de diseñar un programa.Cada objeto es una instancia de una clase particular.

Clase: En POO, es una definición por plantilla de los métodos y variables de un objeto particular.

Método: En POO, un método es un procedimiento programado definido como parte de una clase e incluido en cualquier objeto de esa clase.

Variable: Valor que puede cambiar dependiendo de la información o condiciones que se

presenten en el programa.

S.O.L.I.D: Principios de diseño que buscan diseños más entendibles, flexibles y mantenibles.

GRASP: En POO, serie de lineamientos que busca asignar responsabilidad a clases.

Iterador: Patrón de diseño que permite al programador recorrer un contenedor.

Decorador: Patrón de diseño que permite añadir comportamiento a un objeto.

Observador: Patrón de diseño donde un objeto llamado Sujeto mantiene una lista de sus objetos dependientes(observadores) notificándolos automáticamente de cualquier cambio en el estado del sujeto.

IBM WATSON: Sistema de respuesta de preguntas capaz de responder preguntas en lenguaje natural, desarrollado por IBM.

1.4 References

1-Plantilla de Software Architecture Document proporcionada por Rational Unified Process.

2-Especificación del proyecto 2 proporcionado por Prof.Ing.Ericka Solano Fernández.

1.5 Overview

El el documento ofrece una serie de diagramas que explicarán internamente el sistema, desde el punto de vista lógico. Estos diagramas siguen la notación de UML. El documento contiene los siguientes puntos importantes que se van a analizar:

- Diagrama de representación de la arquitectura.
- Objetivos y limitaciones de la arquitectura.
- Una vista de los Casos de Uso con los que cuenta el sistema.
- Una vista lógica en forma de paquetes.
- Proceso de carga de vistas del sistema.
- Vista de despliegue.
- Visión de estructura de la Base de Datos.
- Tamaño y Desempeño del sistema.

2. Architectural Representation

La arquitectura que se propuso para desarrollar el proyecto como punto inicial es utilizar el

modelo-vista-controlador, representado con el siguiente diagrama:

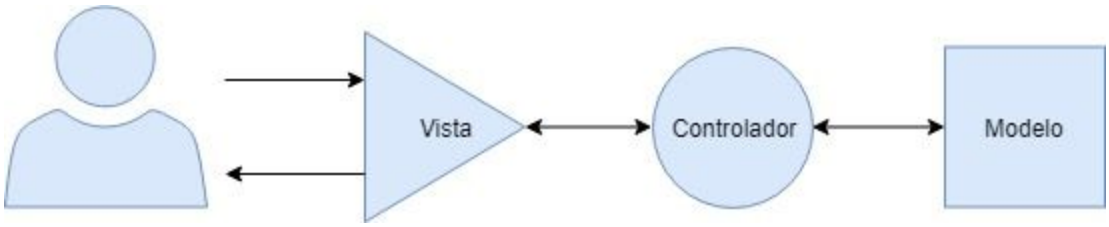


Figura 1

Debido que el sistema desarrollado es para dispositivos móviles, donde puede ser utilizado desde cualquier zona, los datos se encuentran almacenados en un Backend, donde los clientes a través de cada aplicación realizan diferentes solicitudes con las consultadas deseadas hacia el servidor de WebHosting.

Entonces la aplicación se ve representada con el siguiente diagrama:

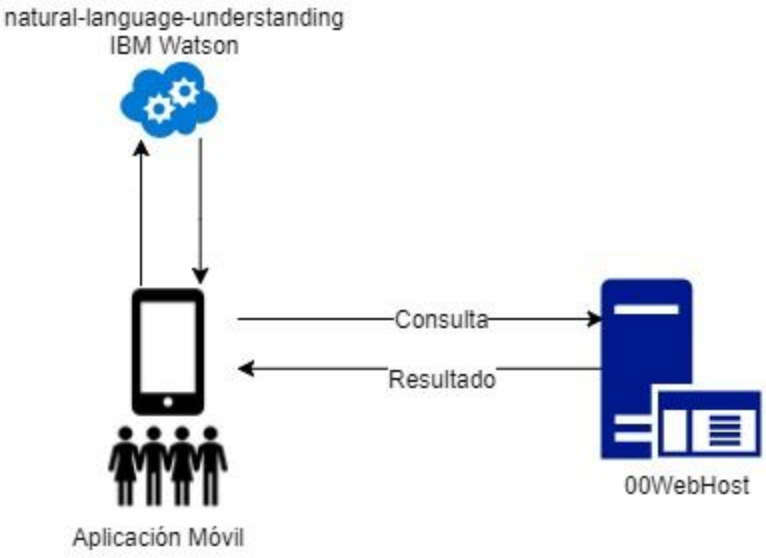


Figura 2

3. Architectural Goals and Constraints

Mantenibilidad: Uno de los objetivos clave de esta propuesta es lograr que mantener y extender el sistema sea sencillo.

Accesibilidad: el sistema debe garantizar que los clientes puedan generar consultas desde su dispositivo móvil.

Portabilidad: La aplicación debe estar disponible para ser utilizado en cualquier dispositivo Android Móvil.

4. Use-Case View

Para el Aplicación existe un caso de uso relacionado; es decir, cada funcionalidad representa un evento en el cual es sistema debe de proporcionar una funcionalidad, para lo cual se tiene el siguiente diagrama:

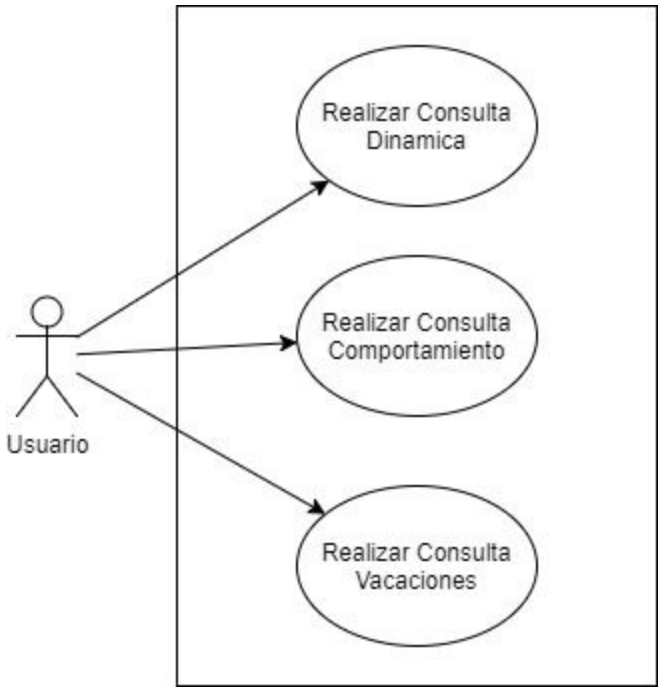


Figura 3

5. Logical View

Este apartado muestra cómo es la comunicación dentro de la aplicación; por lo cual es de mucha importancia porque es donde se ve referenciado el diseño de software. El sistema fue desarrollado a través de los requerimientos otorgados por la profesora, donde se desarrollaron 2 partes a alto nivel:

● Servidor de Base de Datos:

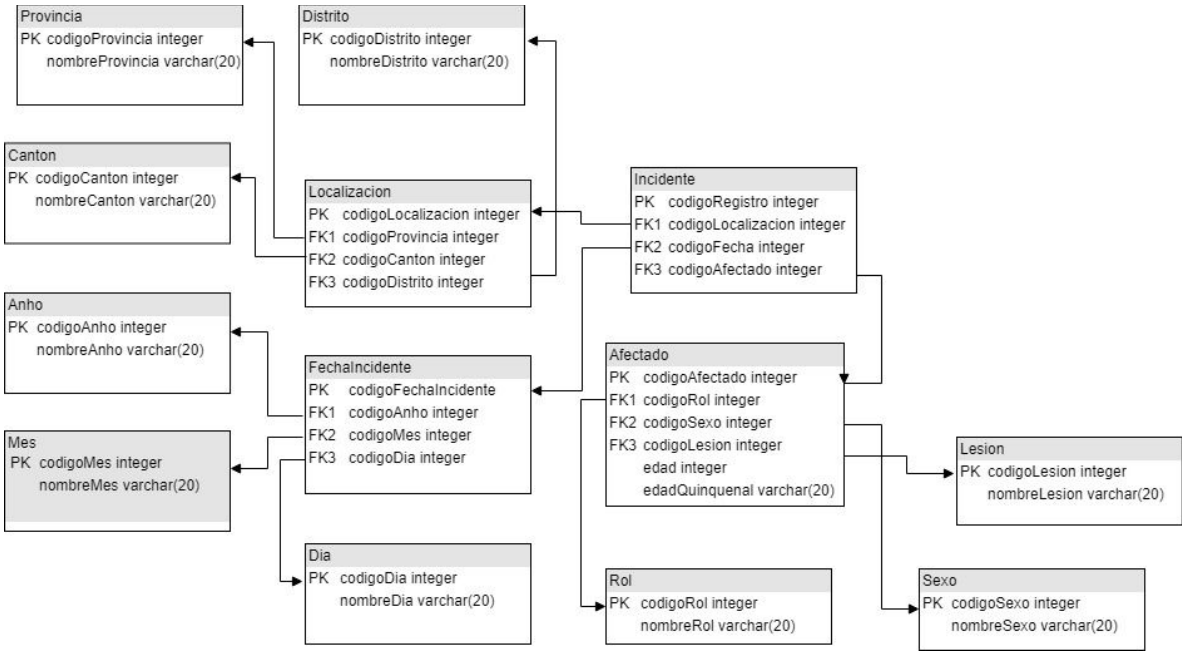


Figura 4

● Aplicación Móvil:
Enlace:

https://www.draw.io/?state=%7B%22ids%22:%5B%221cPcPcHUOATV841eiBzWjBoO-xyNksl90%22%5D,%22action%22:%22open%22,%22userId%22:%22103897335034874625196%22%7D#G1cPcPcHUOATV841eiBzWjBoO-xyNksl90

Diagrama Completo:

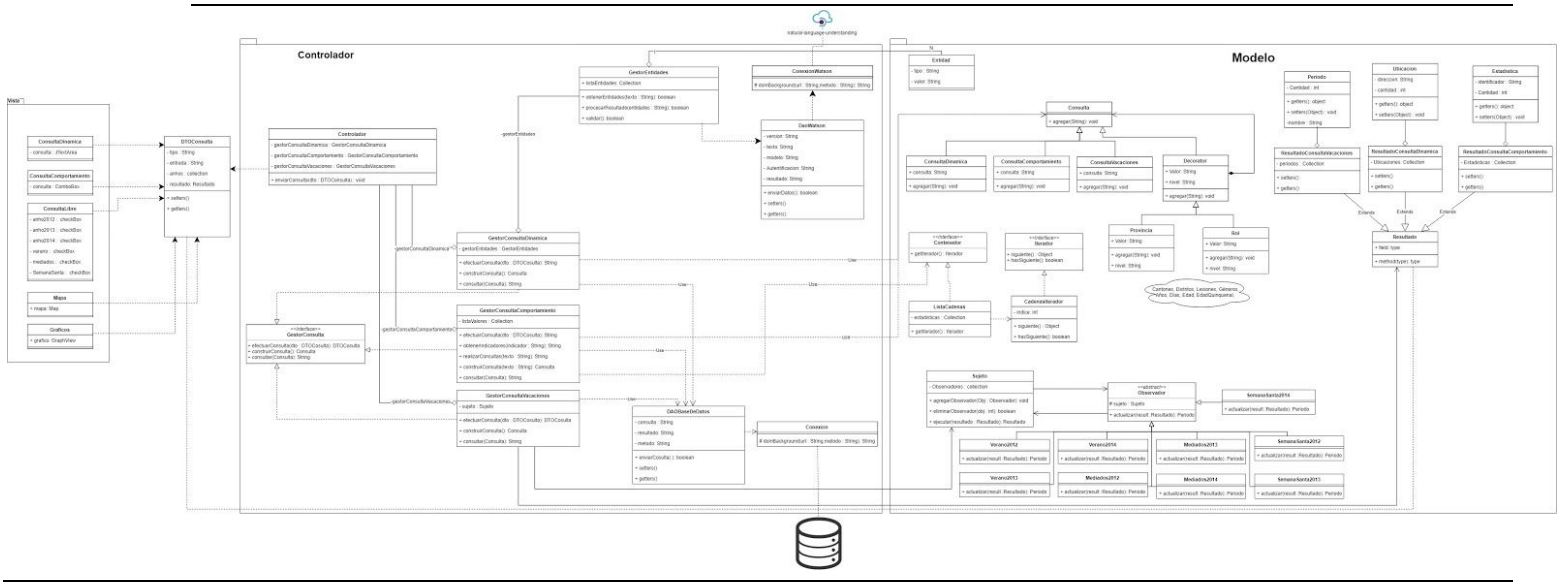


Figura 5

Vista:
Modelo:
Controlador:

5.1 Overview

Esta implementación puede ser descompuesta en sus tres grandes paquetes, de acuerdo al modelo MVC implementado: modelo, vista y controlador.

El paquete de modelo contiene todas las clases relacionadas directamente con el manejo de datos que son necesarios para dar por satisfecha una consulta generada por un usuario. Estos elementos permiten tener un control de los datos para garantizar el éxito.

El paquete controlador conecta todas las solicitudes de consulta generadas por los usuarios durante el uso de la aplicación. Estos elementos del controlador permiten manejar de buena manera la administración de la información extraída, de manera que los elementos de interfaz gráfica no tengan un acceso a directo a la capa de modelo de la arquitectura.

El paquete de vista contiene todos los elementos de interfaz gráfica, de manera que el usuario pueda interactuar con la aplicación para poder realizar las consultas a la base de datos.

5.2 Architecturally Significant Design Packages

En cuanto al ordenamiento de paquetes, internamente el proyecto cuenta con el siguiente esquema de ordenamiento bastante separado debido que se deben guardar muchos archivos para mantener el buen funcionamiento:

- Controlador: Se encuentran las clases que son utilizadas para el control a más alto nivel del sistema como que permiten el manejo de las interacciones generadas por los usuarios.
- Modelo: Contiene todas las clases que son parte del modelo.
- Vista: Utilizado para manejar la interfaz y mostrar los diferentes datos de consulta solicitados.
- Res/layout: Contiene archivos XML de configuración a nivel visual de cada una de las vistas de la aplicación.

6. Process View

Para entender el flujo de los procesos se debe comprender como funcionaba desde las dos partes:

- Aplicación:
    - Con la ejecución de la aplicación se le permite al usuario enviar una solicitud de consulta.
    - El usuario selecciona la consulta que le interesa ingresando los datos correspondientes, la cual es procesada por el administrador de la base de

datos.
    - Cuando la consulta es procesada se devuelve la información para ser mostrada en pantalla al usuario.
- Administrador de Base de Datos:
    - El servidor recibe un solicitud la cual corresponde a una consulta de SQL que es ejecutada en la base de datos, y luego envía esos datos a la aplicación que realizó la consulta.

7. Deployment View

Como se puede observar en la figura 2, el funcionamiento completo del sistema tiene involucrado tres partes principales la aplicación móvil, backend o servidor de datos y el servicio de IBM Watson, los cuales realizan la comunicación a través de consultas HTTP, para envio y solicitud de información.

Además, debido que en la arquitectura propuesta se cuentan con DTOs, estos son utilizados de igual manera para la conexión, para respetar el modelo MVC propuesto donde las solicitudes llegan con una forma establecida y son retornadas con la respuesta.

8. Implementation View

8.1 Overview

Se implementó la arquitectura de diseño de 3 capas MVC para la aplicación móvil, donde los objetos que se utilizan para manejar la persistencia se encuentra en la capa de Modelo.Las clases que se encargan de toda la lógica que tienen estas dos aplicaciones, se maneja por medio de un controlador que se encuentra en la capa de Controller. Y por último la que se encarga de la comunicación con el usuario y de mostrar la información de las aplicaciones es la capa de View.

8.2 Capas

Model

Para la capa de la aplicación tienen las siguientes clases:

- Resultado
- ResultadoConsultaVacaciones
- ResultadoConsultaDinamica
- ResultadoConsultaComportamiento
- Periodo
- Ubicacion
- Estadistica
- Consulta
- ConsultaDinamica
- ConsultaComportamiento
- ConsultaVacaciones
- Decorator

- Provincia
- Cantones
- Distritos
- Lesiones
- Generos
- Anhos
- Dias
- Edad
- EdadQuinquenal
- Rol
- Iterator
- CadenasIterator
- ListaCadenas
- Contenedor
- Sujeto
- Observador
- SemanaSanta2014
- SemanaSanta2013
- SemanaSanta2012
- Mediados2014
- Mediados2013
- Mediados2012
- Verano2014
- Verano2013
- Verano2012
- Entidad

Controller

Para la capa de la aplicación se tienen las siguientes clases:

- ConexionWatson
- DaoWatson
- GestorEntidades
- GestorConsultaDinamica
- GestorConsultaComportamiento
- GestorConsultaVacaciones
- DAOBaseDeDatos
- Conexion
- GestorConsulta
- Controlador
- DTOConsulta

View

Para la capa de la aplicación se tienen las siguientes clases:

- ConsultaDinamica
- ConsultaComportamiento

- ConsultaLibre
- Mapa
- Grafico

9. Size and Performance

A continuación se mostrarán las condiciones de rendimiento y restricciones con las que cuentan las dos aplicaciones.

- La aplicación móvil debe contar con servicio de internet para poder enviar las solicitudes.
- El administrador de datos permite que múltiples usuarios realizan peticiones, es decir , múltiples aplicaciones ejecutándose a la vez y realizando consultas.
- El administrador de datos siempre se mantiene pendiente a recibir peticiones.
- Se necesita el sistema operativo Android 5.0 para poder utilizar la aplicación.

10. Quality

- Performance: La aplicación ejecutará la consulta que introduzca el usuario de una manera rapido y mostrará los puntos en el mapa. El tiempo en el que se realiza es menor a 2 segundos. Además para las consultas en la que se realiza por comportamiento y por atributos de vacaciones, se grafica en un tiempo no mayor a 1 segundo.

- Scalability: El sistema es altamente capaz de crecer y adaptarse a ese crecimiento. La herencia aplicada en diferentes niveles permite que agregar nuevos módulos al sistema sea extremadamente sencillo.

- Scalability: La aplicación tiene la cualidad de que es de fácil escalabilidad debido a que las funcionalidades se implementaron utilizando patrones de diseño como el decorator, observer, entre otros. Esto ayudaría en el caso de que se necesite agregar otro tipo de consulta para una nueva funcionalidad, sea más sencillo de realizar.

- Maintainability: Se da debido a la baja dependencia entre las clases de la aplicación debido a diseño del mismo.

- Security: Se valida que los datos como keys para utilizar la api para el mapa entre otros valores, se guardar en una parte específica de la aplicación para evitar tener estos datos al descubierto en el código. Además para realizar peticiones a la plataforma de IBM cloud Watson, se necesita realizar una autenticación, debido a esto, se evita que cualquier persona acceda a los recursos de una manera no deseada.

11. Anexos

11.1. Funcionalidades

- Consultas dinámicas solicitadas en el dashboard: 100%.

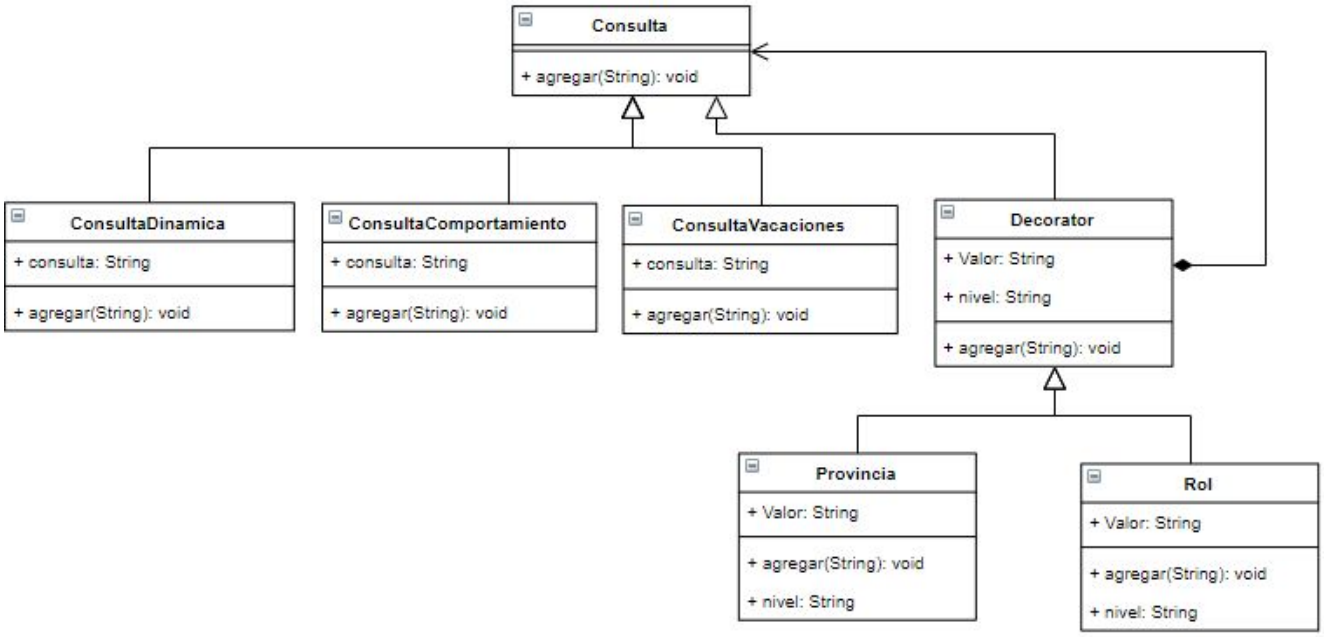
-Consulta de comportamiento de un indicador: 100%.

-Consulta libre (De vacaciones): 100%.

11.2. Justificación del uso del patrón estructural: Decorador

El uso de este patrón estructura en el desarrollo de la implementación del sistema es muy útil, debido que se está trabajando en una aplicación donde se debe de crear una consulta que puede tener una gran cantidad de elementos pero al final es una consulta. Entonces este patrón permite asignarle responsabilidades a objetos como Provincia, Cantón, Distritos, Mes, entre otros, de otorgar responsabilidad a cada objeto de darle la forma correcta a la consulta, donde la forma correcta corresponde a la sentencia de sql que es correcta para el atributo.

También, porque el decorador permite que se bajan agregando elementos dentro del decorador, entonces permite la escabilidad de la consulta.

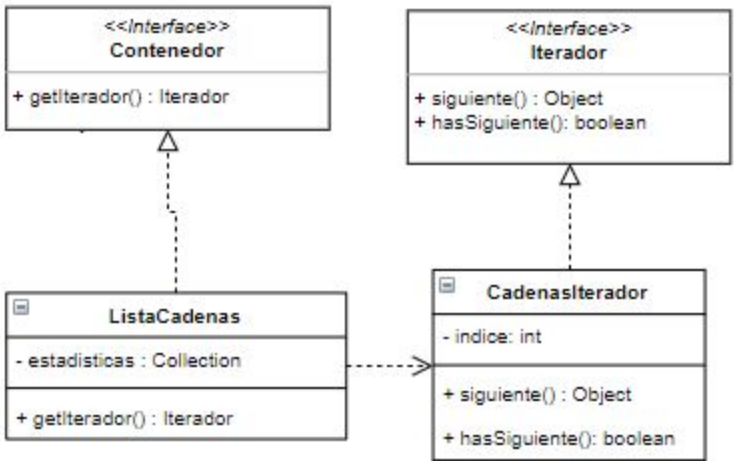


11.3. Justificación del uso del patrón de Comportamiento: Iterador

Se seleccionó el patrón Comportamiento **Iterador** debido que como la consulta 2, la cual en el sistema se conoce como Comportamiento, necesita obtener la información de todos los valores disponibles para un indicador (como Provincia, Cantón, Distrito, entre otros). Entonces se tiene que consultar en la Base de datos los accidentes que son igual al valor del indicador, entonces se pueden se crea una estructura que almacena la cantidad de accidentes que cumplen esa característica entonces a través del iterador se pueden obtener la cantidad que cumple, sin importar la información que se encuentra en cada accidente.

Entonces a través del iterador se puede recorrer el contenedor que contiene los resultados de las consultas generadas y extraer la información para generar los resultados que son mostrados

al usuario de la aplicación.



11.4. Localización de puntos en el mapa

Para esta parte lo que se planteó fue utilizar la api de google maps para poder mostrar los puntos en el mapa con al cantidad de accidentes que ocurren en dicha zona. Estos puntos se mostrarán con base a la consulta realizada por el usuario. Para hacer que el uso de la aplicación sea mejor para los usuarios, se muestra el mapa en la totalidad de la pantalla a excepción de la barra de herramientas.

11.5. Análisis e investigación para el diseño de la interfaz gráfica

Como primer análisis que se realizó para la aplicación, fue elegir una paleta de colores que fuera acorde al tema de la aplicación. No se podían elegir colores que muy alegres debido a que el tema era accidentes, entonces se eligieron colores que no resaltan mucho, pero que dieran un entorno amigable para el uso de la aplicación. Luego de la selección de colores, se tuvo que elegir la distribución de las ventanas y lo que se iba a agregar en cada una. Para esto se realizó desde cero el estilo de la interfaz gráfica. Como fueron la distribución de los elementos en las ventanas, los tamaños de letra, estilos de letra, iconos para los botones, iconos para el menú, entre otros.

11.6. Aspectos de UX considerados

Primero que todo se consideró utilizar la misma paleta de colores para todas las ventanas de la aplicación para brindar un entorno agradable para el uso de la misma. Por otra parte la elección de los iconos se realizó utilizando los mismos tipos de iconos para toda la aplicación. Este punto ayuda a que los usuarios se familiaricen con el estilo que tiene la aplicación. Además se tiene la distribución de las ventanas en orden de importancia con la que el usuario deberá utilizar la aplicación. También se eligió un tamaño de letra para el texto de la información de ayuda para la aplicación, que fuera visible para todos los usuarios posibles que utilicen la aplicación, sin importar la edad. Por otra parte con la estructura que

tiene cada ventana, se va guiando a los usuarios para que utilicen esta de una forma sencilla.