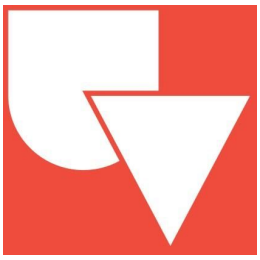


UNIVERSIDAD DEL VALLE

ULTIMA ENTREGA
PROYECTO FINAL

BASES DE DATOS

INTEGRANTES:

CARLOS ESTEBAN MURILLO
ANDRÉS MAURICIO MONTENEGRO
CHRISTIAN DAVID MILLAN

PRESENTADO A:

CASTILLO ROBLES ANDRÉS MAURICIO

Santiago de Cali, Marzo de 2019

Escuela de Ingeniería en Sistemas
Facultad de Ingeniería

CONTENIDO

1.INTRODUCCIÓN

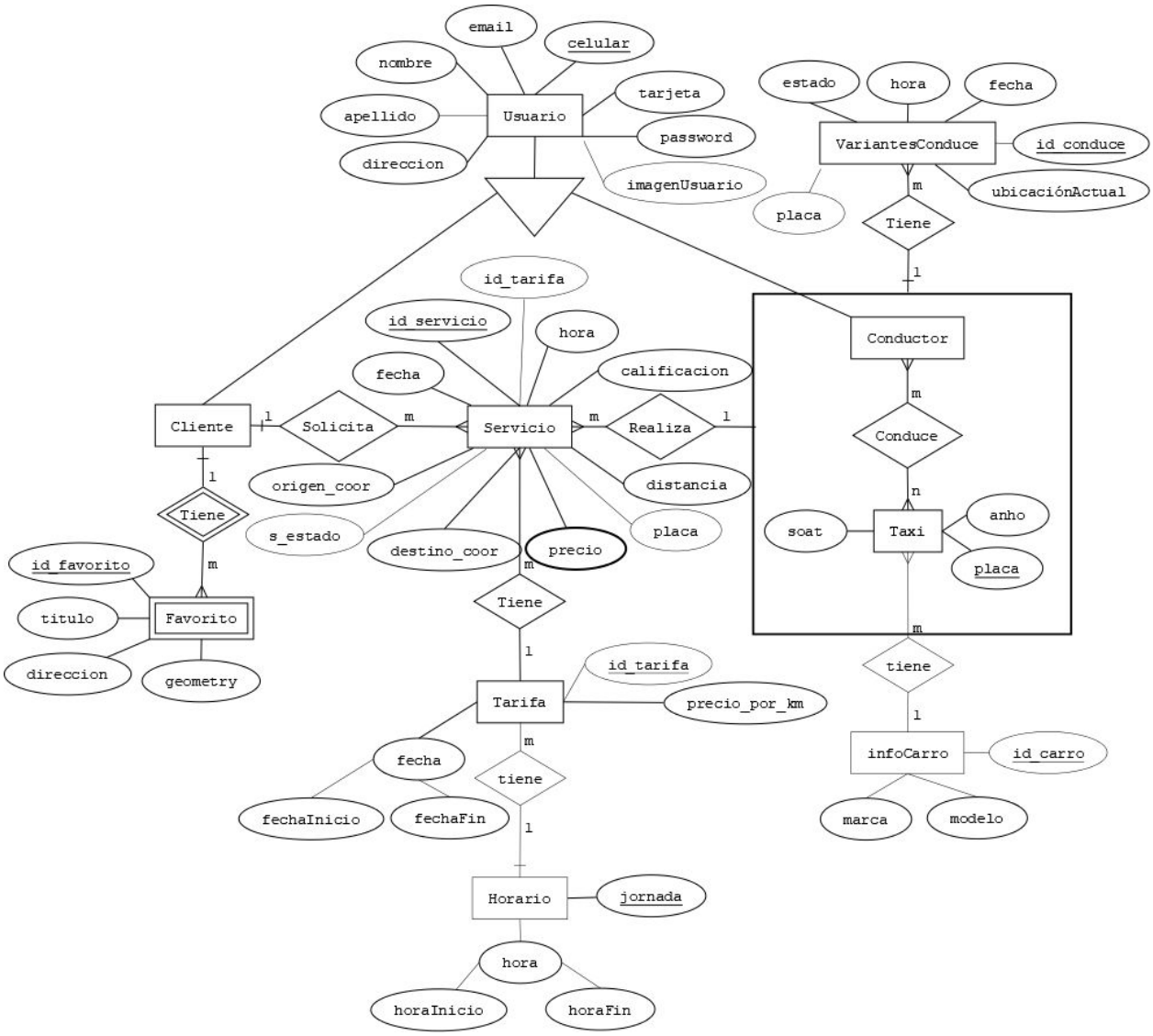
Debido al gran problema de movilidad en Cali y al creciente uso de aplicaciones telefónicas para llevar a cabo viajes de un lugar a otro, hemos creado una aplicación que busca ayudar a solventar dicho problema haciendo uso de tecnologías web, esto permite a una empresa de taxis (medio de transporte legal en la ciudad) registrar conductores y clientes para que interactúen juntos en un servicio el cual permite planear un viaje de principio a fin conociendo el valor de este de antemano como también su tiempo aproximado y la información de los involucrados, algo fundamental ya que genera confianza al momento de usarla y al momento de concretar un viaje. Es flexible con el pago ya que permite el uso tarjeta de credito o debito. Consideramos que la aplicación tiene un buen desempeño y que satisface las necesidades de un público específico de la ciudad.

2. OBJETIVOS

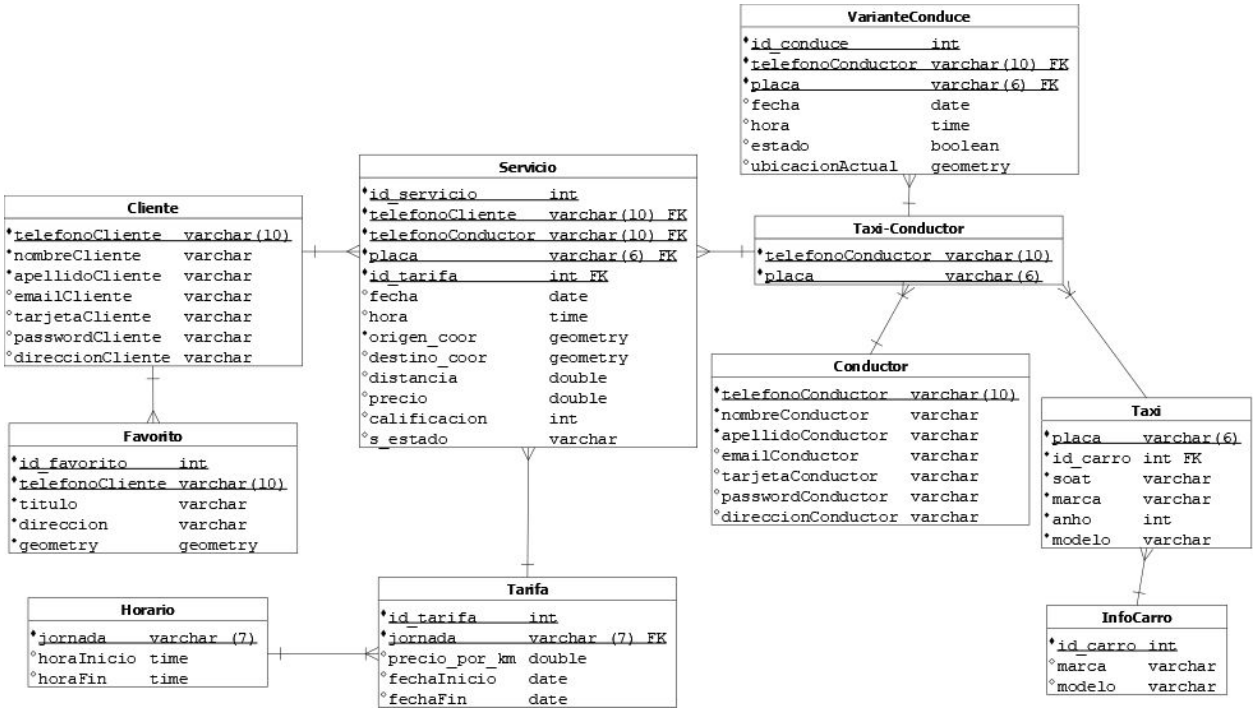
1. Ofrecer una aplicación de tipo servidor que sirva de API para manejar las peticiones de los clientes y procesarlas de manera correcta.
2. La API provee una conexión con la base de datos para consultar y editar información.
3. Las aplicaciones del cliente (Conductor y Pasajero) pueden realizar peticiones a la API y a su vez tienen la responsabilidad de procesar la respuesta que esta les envía.
4. Las aplicaciones cliente proveen una interfaz amigable la cual es fácil de usar para un usuario final. Estas permiten a su vez la opción de poder crear una cuenta o bien loguearse en la plataforma.
5. La aplicación del cliente Pasajero además de poder mostrar y permitir modificar la información personal, ofrece las funcionalidades de poder visualizar los conductores disponibles en el mapa, crear, modificar y eliminar posiciones marcadas como favoritas y solicitar servicios.
6. La aplicación del cliente Conductor ofrece consultar y editar su información personal, asignar y eliminar automóviles para trabajar. También puede alterar su estado entre ocupado o disponible para ser incluido o no en las búsquedas. Por último la responsabilidad de recibir notificaciones por parte de algún cliente pasajero que solicite el servicio.

3. DISEÑO DE MODELO DE DATOS

Para el diseño de la base de datos, se implementó los modelos entidad-relación y relacional siguientes:



Modelo Entidad-Relación



Modelo Relacional

3.1 Cardinalidades de cada entidad.

Se creó una entidad Usuario para especializarse en **Cliente** y en **Conductor**, debido a que comparten los mismos atributos y sería redundante repetirlos de cliente a conductor.
Conductor está en una cardinalidad muchos conductores pueden conducir muchos taxis, que se representa como una entidad fuerte taxi.
Cliente tiene una relación con **favoritos** donde, esta entidad favoritos es débil, debido a que si no existe un cliente no hay favoritos
Cliente tiene relación de uno a muchos con **servicio**, se puede ver como una agregación [**conductor-conduce-taxi]** también con cardinalidad de muchos a 1 de la agregación a servicios, se quiso poner aquí una agregación para hacer una abstracción de esta relación que es fundamental en todo el diagrama, muchos **conductores** conducen muchos **taxis**.

A su vez esta agregación tiene una relación con infocarro su cardinalidad es agregación por 1 solo infocarro e infocarro para muchos vehiculos , infocarro que es la información total del vehículo,y también tiene una relación con **variantesconduce**, esta entidad tiene la función de darnos la ubicación actual del conductor en cualquier tiempo y su cardinalidad es 1 **conductor-taxi(agregacion)** muchas **variantesconduce**(posicion) y muchas posiciones(variantesconduce) 1 solo conductor.

La entidad **tarifa** tiene una relacion con **servicio**, su cardinalidad es un servicio una tarifa, y una tarifa muchos servicios.

Hasta acá se han descrito cada una de la entidades que participan en el modelo relacional.
Los datos de entrada ya se definieron en META DATA

4. ARQUITECTURA DE LA APLICACIÓN

Estructura Física de la Aplicación
Descripción individual de los directorios de app Conductor y Cliente:
Nuestro software tiene una estructura basada de VUEJS los siguientes directorios se sujetan a esta estructura
Src: directorio principal donde se encuentra gran parte de la funcionalidad de la aplicación.
Este contiene:
Assets, componentes, router, store, static un archivo, App.vue, y otro archivo main.js además de un archivo index.html y archivos package.json, módulo con librerías y frameworks usados en la app, módulo build y config.

4.1 Descripción General y Propósito: ¿Qué contiene cada directorio y cuál es su función?

Assests es un directorio que contiene imágenes estáticas, éstas son usadas dentro de la aplicación llamadas desde la parte interna del código y su ruta.
Componentes, este directorio es fundamental, ya que contiene módulos. vue, cada uno de estos contiene una parte html llamada template, una parte java script y una parte de style de css, cada módulo componente se comunica entre sí y entre otros archivos html y. vue para implementarse en el navegador.

Router: El módulo router contiene un archivo index.js, que en ruta cada componente dentro de la aplicación.

Store: Este directorio se encarga de contener un módulo store.js el cual contiene todas las variables con las que el programa trabaja, además de los métodos que las modifican y las implementan.
Node modules: Contiene todas las librerías de cada framework con el que trabaja la app.

4.2 Metadatos

Cliente					
Llave	**Nombre**	**Tipo**	**Longitud**	**Usuario Responsable**	**Descripción**
PK	telefonoCliente	varchar	10	Cliente	número telefónico de un cliente en particular
	nombreCliente	varchar	40	Cliente	Nombre de un cliente en particular
	apellidoCliente	varchar	40	Cliente	Apellido de un cliente en particular
	direccionCliente	varchar	40	Cliente	dirección de un cliente en particular
	emailCliente	varchar	60	Cliente	email de un cliente en particular
	tarjetaCliente	varchar	10	Cliente	tarjeta de crédito de un cliente
	passwordCliente	varchar	40	Cliente	password de un cliente
Conductor					
Llave	**Nombre**	**Tipo**	**Longitud**	**Usuario Responsable**	**Descripción**
PK	telefonoConductor	varchar	10	Conductor	número de un conductor en particular
	nombreConductor	varchar	40	Conductor	Nombre de un conductor en particular
	apellidoConductor	varchar	40	Conductor	Apellido de un conductor en particular
	direccionConductor	varchar	40	Conductor	dirección de un conductor en particular
	emailConductor	varchar	60	Conductor	email de un conductor en particular
	tarjetaConductor	varchar	10	Conductor	tarjeta de crédito de un conductor
	passwordConductor	varchar	40	Conductor	password de un conductor
Favoritos					
Llave	**Nombre**	**Tipo**	**Longitud**	**Usuario Responsable**	**Descripción**

	titulo	varchar	40	Cliente	título de un lugar favorito
	geometry	geometry point	40	Cliente	coordenada espacial del sitio
	direccion	varchar	40	Cliente	dirección del sitio
PK	id_favorito	int	10	Cliente	identificador del lugar
FK	celular	int	10	Cliente	número telefónico de un cliente
Vehiculo					
Llave	**Nombre**	**Tipo**	**Longitud**	**Usuario Responsable**	**Descripción**
PK	placa	varchar	6	Conductor	identificador del vehículo
	marca	varchar	40	Conductor	marca del vehículo
	modelo	varchar	40	Conductor	modelo del vehículo
	soat	int	40	Conductor	número de soat del vehículo
	anho	int	4	Conductor	año de fabricación del vehículo
Tarifa					
Llave	**Nombre**	**Tipo**	**Longitud**	**Usuario Responsable**	**Descripción**
PK	Id_tarifa	int	10	Sistema	identificador del tipo de tarifa
	precio_por_km	double	5	Sistema	valor por km
	jornada	varchar	20	Sistema	nombre de la jornada
	fechaInicio	date	default	Sistema	fecha de inicio de la jornada
	fechaFinal	date	default	Sistema	fecha final de la jornada
	horaInicial	hour	default	Sistema	hora inicial de la jornada
	horaFinal	hour	default	Sistema	hora final de la jornada
Servicio					
Llave	**Nombre**	**Tipo**	**Longitud**	**Usuario Responsable**	**Descripción**
PK	id_servicio	int	10	Conductor - Cliente	identificador de un servicio a realizar
FK	telefonoCliente	varchar	10	Cliente	teléfono del cliente que solicita el servicio
FK	telefonoCliente	varchar	10	Conductor	teléfono del conductor que va a realizar el servicio

FK	placa	varchar	6	Conductor	placa del automóvil que se usará para prestar el servicio
FK	id_tarifa	int	10	Conductor - Cliente	identificador de la tarifa correspondiente al servicio
	fecha	date	default	Conductor - Cliente	fecha en que se presta el servicio
	hora	hour	default	Conductor - Cliente	hora en que se presta el servicio
	origen_coor	geometry point	default	Conductor - Cliente	coordenada de origen del servicio
	destino_coor	geometry point	default	Conductor - Cliente	coordenada del destino del servicio
	distancia	double	5	Conductor - Cliente	distancia desde el origen al destino en Km
	precio	double	5	Conductor - Cliente	precio del servicio
	calificación	int	1	Conductor - Cliente	Calificación del servicio
Taxi - Conductor (conduce)					
Llave	**Nombre**	**Tipo**	**Longitud**	**Usuario Responsable**	**Descripción**
PK	telefonoConductor	int	10	Conductor	teléfono del conductor como llave foránea
PK	placa	varchar	6	Conductor-vehiculo	placa del vehículo como llave foránea
VarianteConduce					
Llave	**Nombre**	**Tipo**	**Longitud**	**Usuario Responsable**	**Descripción**
PK	id_conduce	int	10	Conductor - vehiculo	identificador de las variantes de conducción
FK	telefonoConductor	varchar	10	Conductor - vehiculo	teléfono del conductor
FK	placa	varchar	6	Conductor - vehiculo	identificador del vehículo

	fecha	date	default	Conductor - vehiculo	fecha en que se está conduciendo cierto vehículo
	hora	hour	default	Conductor - vehiculo	hora en que se está conduciendo cierto vehículo
	estado	boolean	1	Conductor - vehiculo	estado actual del conductor y su vehículo
	ubicacionActual	geometry point	default	Conductor - vehiculo	ubicación actual del vehículo que maneja cierto conductor

5. ESTRUCTURA FUNCIONAL DE LA APLICACIÓN

Ambas aplicaciones tanto la del cliente y la del conductor poseen una estructura similar, por lo tanto vamos a generalizar la explicación de los mecanismos. Partamos de los archivos .vue dentro del directorio **componentes**, estos archivos(módulos) son un pedazo de la aplicación que se ejecuta sobre un archivo app.vue que a su vez se ejecuta sobre un archivo HTML, cada componente, debe ser llamado desde un archivos .js llamado router, el cual tiene la funcionalidad de darle un nombre reconocible a cada componente para llamarlo desde cualquier otro componente con la función .push de vuejs, la parte importante de estos componentes .vue es que son reactivos y se pueden pasar información entre sí, utilizando el directorio y el módulo store.
El archivo llamado **store.js**, es el que atrapa por decirlo así a los datos con los que se trabaja en la front-end y a su vez con los que se trabaja en la api-rest back-end.

Cada variable que se almacena en el **store** puede ser modificada únicamente dentro de **mutaciones** que son funciones que permiten la alteración de variables que están en el objeto llamado **state**, a su vez las funciones **actions**, pueden llamar a cada mutation para modificar los valores “globales” del **state.**

Router: el archivo router, este es un archivo javascript que usa librerías de vue router, su función es agregar rutas a un arreglo llamado routes, que contiene el path, dirección dentro del navegador, nombre del componente, y el componente como tal.

App.vue, este módulo contiene un template donde se llama al componente que contiene a otros componentes para funcionalidad, dentro de la parte script se exporta el nombre del módulo, y el mismo componente.

Módulo index.html, contiene un id que llama a App.vue y este a su vez llama a otro componente que enlaza a los demás

Toda esta descripción de la arquitectura del programa viene orientada de la documentación que se encuentra en la página oficial de vuejs [1], ya que entre las tecnologías y librerías que se usarán para esta app, están vuex y vuejs

1. https://vuejs.org/v2/guide/

Diagrama de despliegue

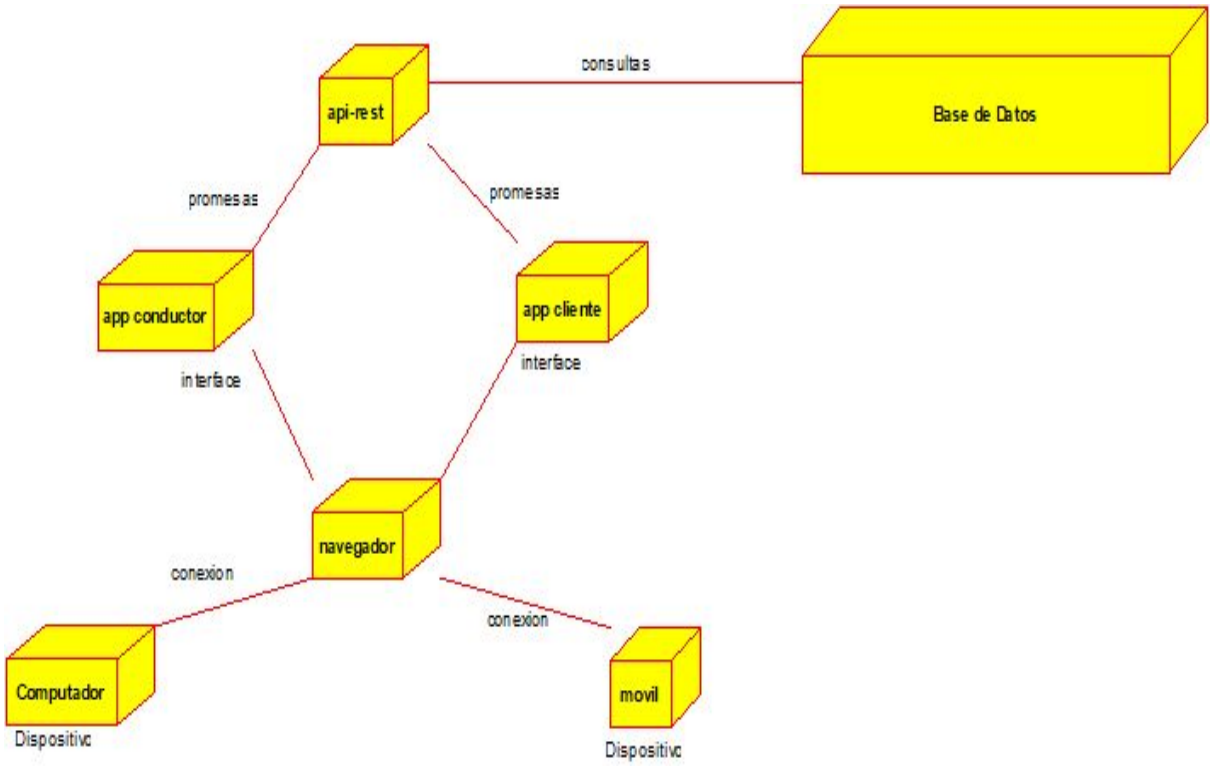
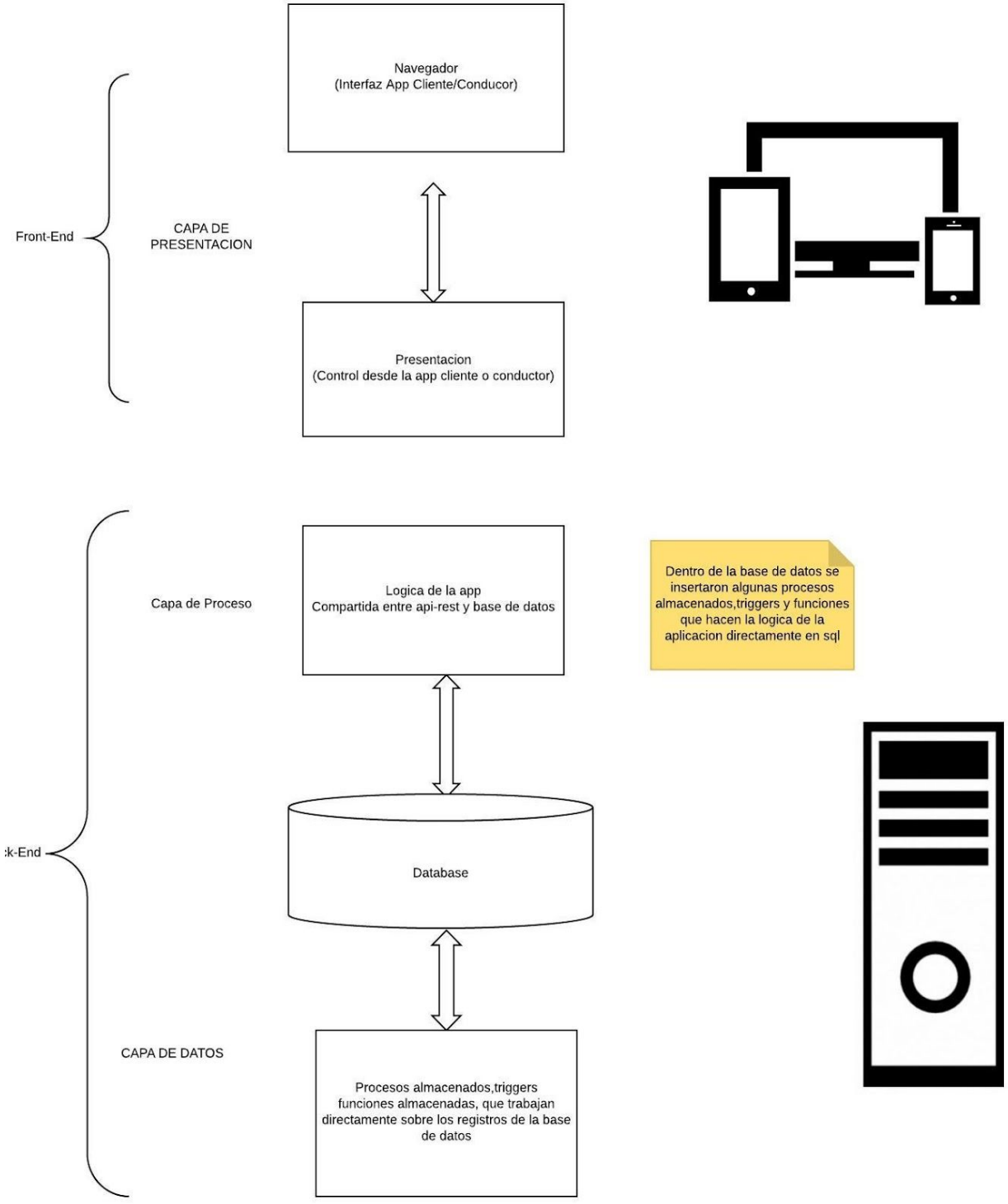
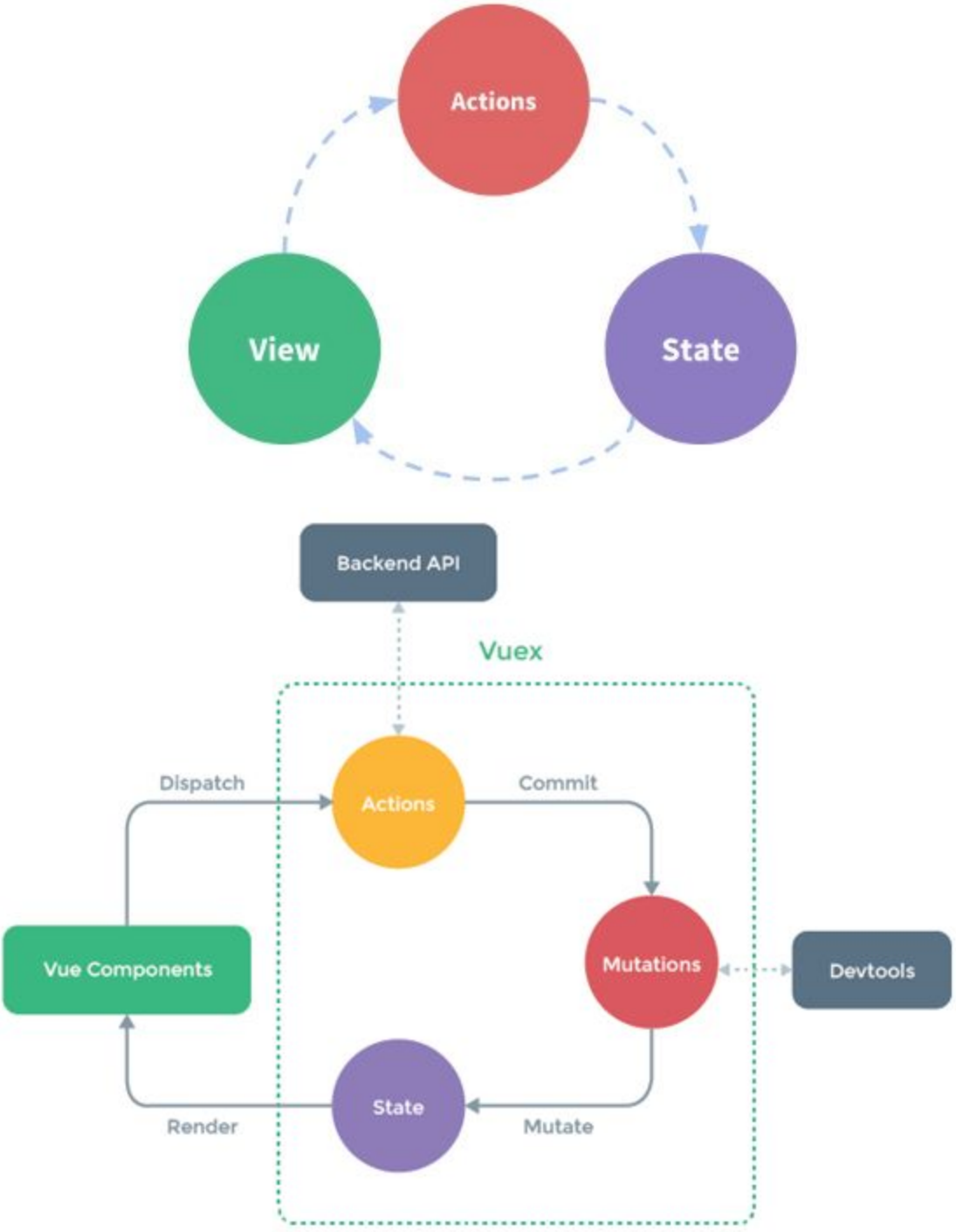


Diagrama de Capas



Dado que la app tanto del cliente como del conductor están sobre la tecnología de vuejs y vuex, se dispone de los diagramas de cada uno para dar un mayor entendimiento de cómo funciona la aplicación, más concretamente su estructura.



6. CONCLUSION

El proyecto satisface las necesidades del cliente de forma eficiente, ofreciendo aplicaciones fáciles de usar, adaptadas a las necesidades de cada usuario, ya que disponen de herramientas para el buen funcionamiento de todos los requerimientos que se necesitan, desde funcionalidades básicas como modificar la información personal, hasta implementaciones más complejas como solicitar servicios de transporte.

7. BIBLIOGRAFIA

Referencias:
NPM https://www.npmjs.com/
VUEJS https://vuejs.org/
NODE JS https://nodejs.org/es/
Postgres https://www.postgresql.org/
Postgis https://postgis.net/
Leaflet https://leafletjs.com/
Open street Maps https://www.openstreetmap.org/#map=5/4.632/-74.299

Links a los repositorios

Api:
https://github.com/carban/taxiapi
Cliente:
https://github.com/carban/taxiclient
Conductor:
https://github.com/carban/taxidriver