

Proyecto NotThatEasyTaxy  
Aplicación móvil para solicitar servicio de taxi

Profesor  
Andres Mauricio Castillo Robles

Integrantes  
María Paula Mosquera Rengifo - 1733431  
Santiago Zuluaga Hernández - 1728003

Universidad del Valle - Meléndez  
Facultad de Ingeniería  
Escuela de ingeniería de sistemas y computación  
Tecnología en sistemas de información  
Bases de datos I - Grupo 01  
Santiago de Cali - Valle del Cauca  
2018

## **CONTENIDO**

- 1. Introducción**
  - 1.1. Propósito del sistema**
  - 1.2. Descripción general del problema**
- 2. Sistema propuesto**
  - 2.1. Requerimientos funcionales**
  - 2.2. Modelos del Sistema**
    - 2.2.1. Identificación de actores**
    - 2.2.2. Modelo de casos de uso**
    - 2.2.3. Descripción de casos de uso**
    - 2.2.4. Modelo entidad-relación**
    - 2.2.5. Modelo relacional**
    - 2.2.6. Interfaz de usuario**
  - 2.3. Requerimientos no funcionales**
  - 2.4. Diccionario de datos**
- 3. Manual técnico**
  - 3.1. Base de datos**
    - 3.1.1. Índices en la base de datos**
    - 3.1.2. Integridad referencial**
    - 3.1.3. Permisos**
    - 3.1.4. Procedimientos almacenados**
    - 3.1.5. Vistas**
  - 3.2. Arquitectura de software**
    - 3.2.1. Cliente**
    - 3.2.2. Servidor**
- 4. Manual usuario**
  - 4.1. Introducción al manual de usuario**
    - 4.1.1. Pantalla inicial**
    - 4.1.2. Registro como socio**
    - 4.1.3. Registro como usuario**
    - 4.1.4. Inicio de sesión como socio**
    - 4.1.5. Inicio de sesión como usuario**
    - 4.1.6. Mapa de rutas**
    - 4.1.7. Perfil del cliente y/o conductor**
    - 4.1.8. Viajes realizados**
    - 4.1.9. Mis vehículos**
- 5. Glosario**

# 1. INTRODUCCIÓN

## 1.1. Propósito del sistema

*NotThatEasyTaxy* es un emprendimiento que quiere permitir a los usuarios de taxi de cualquier lugar del mundo, pedir un servicio desde una aplicación móvil.

## 1.2. Descripción general del problema

En la página principal de la plataforma, los taxistas deben registrarse. Este registro comprende la información personal del conductor y los datos del taxi que maneja (marca, modelo, año, baul, soat). Es posible que varios conductores manejen el mismo taxi, cada vez que el taxista esté disponible, debe reportarlo en la aplicación y cada vez que esté ocupado también, para que no sea incluido en las búsquedas. Cuando el taxista se pone en modo disponible, reporta sus coordenadas GPS a la aplicación, y para simplificar esta versión de la aplicación, vamos a suponer que el taxi no se vuelve a mover hasta que sea elegido para un servicio, o salga del modo disponible.

Los usuarios de taxi, también deben registrarse en la aplicación. La información que se le pide es su nombre, número de celular (identificador), dirección de residencia, coordenadas GPS y número de tarjeta de crédito para el pago. El usuario puede registrar otras direcciones de residencia con su coordenada GPS: Son los orígenes y destinos favoritos del usuario.

Cuando un usuario necesita un servicio, se le pide que de las coordenadas de origen y destino. Estas las puede seleccionar de la lista de favoritos, o las puede seleccionar picando en un mapa. Una vez el usuario solicite su taxi, la aplicación elige al taxi más cercano al usuario para prestar el servicio, y le reporta al usuario la placa, el nombre del conductor, el número de estrellas y la posición actual del taxi.

Vamos a suponer que el servicio siempre se presta. El costo de la carrera es directamente proporcional al número de kilómetros que separan el origen y el destino del servicio en línea recta (Es una versión super simplificada de la aplicación). Y que al final el usuario califica el servicio con un número entero de estrellas (0 pésimo, 5 excelente).

Cada vez que el usuario quiera, puede ver cuantos kilometros de servicios ha usado. Cada vez que el taxista quiera, puede consultar cuantos kilometros de servicios le adeuda la compañía. Los contadores se pueden poner en 0 con un botón “pagar” para el cliente y un botón “cobrar” para el conductor.

Los datos de los usuarios, conductores y vehículos se pueden crear, editar y eliminar cuantas veces se necesite.

## **2. SISTEMA PROPUESTO**

### **2.1. Requerimientos funcionales**

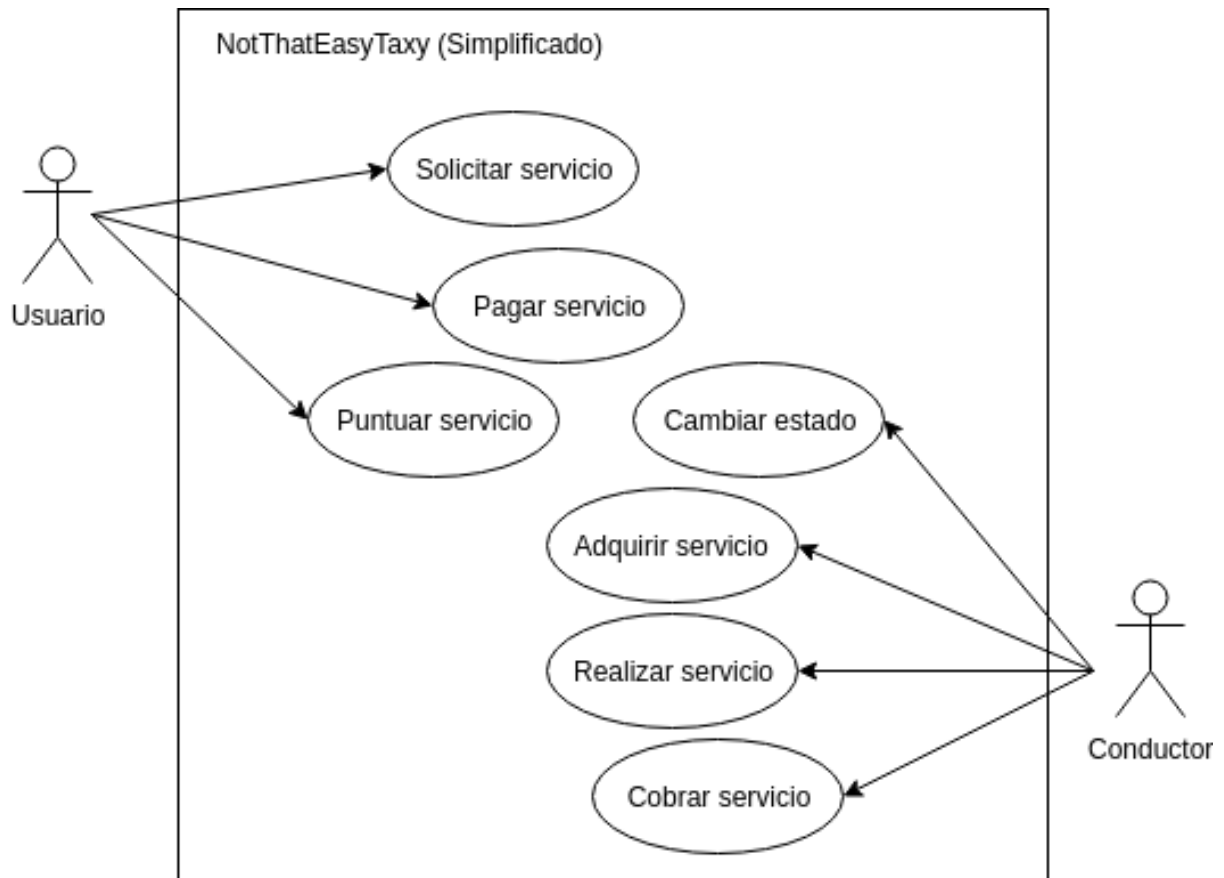
- Permitir el registro de nuevos conductores, para que cada vez que un conductor quiera iniciar sesión, no sea necesario reescribir nuevamente todos los datos, sino que estos ya se encuentren almacenados.
- Permitir el registro de nuevos usuarios, a través de su número de celular, para que cada vez que un usuario quiera iniciar sesión en un nuevo dispositivo, no sea necesario ingresar nuevamente todos los datos, sino que con el número de celular sea posible esta acción.
- Permitir ver los orígenes y destinos favoritos de cada usuario que lo desee.
- Permitir buscar ubicaciones en el mapa para ingresar el punto de partida y el punto de destino que requiera el usuario.
- Permitir que una vez el conductor llegue donde el usuario, esté disponible la opción de habilitar viaje para así poder calcular los kilómetros que conlleve dicha carrera.
- Validar el ingreso al sistema de cada usuario con su número celular y contraseña.
- Validar el ingreso al sistema de cada conductor con su número celular y contraseña.
- Permitir que una vez el usuario solicite un servicio de taxi, la información del conductor le aparezca en la aplicación.
- Permitir ingresar información acerca de la salida de productos, actualizando el sistema.
- Permitir ver cuantos kilómetros en total ha recorrido un conductor en particular.
- Permitir ver cuantos kilómetros en total ha recorrido un usuario.
- Permitir al final del viaje, calificar el viaje ya sea como pésimo (0 estrellas), o excelente (5 estrellas).
- Permitir que se genere una factura en la cual se debe incluir los kilómetros recorridos y el precio de la carrera.

## 2.2. Modelos del sistema

### 2.2.1. Identificación de actores

- **Usuario:** Los usuarios son quienes solicitarán un servicio de taxi a través de la aplicación. Se le buscará el conductor más cercano.
- **Conductor:** Los conductores son quienes adquieren una carrera a realizar para un usuario que así lo requiera.

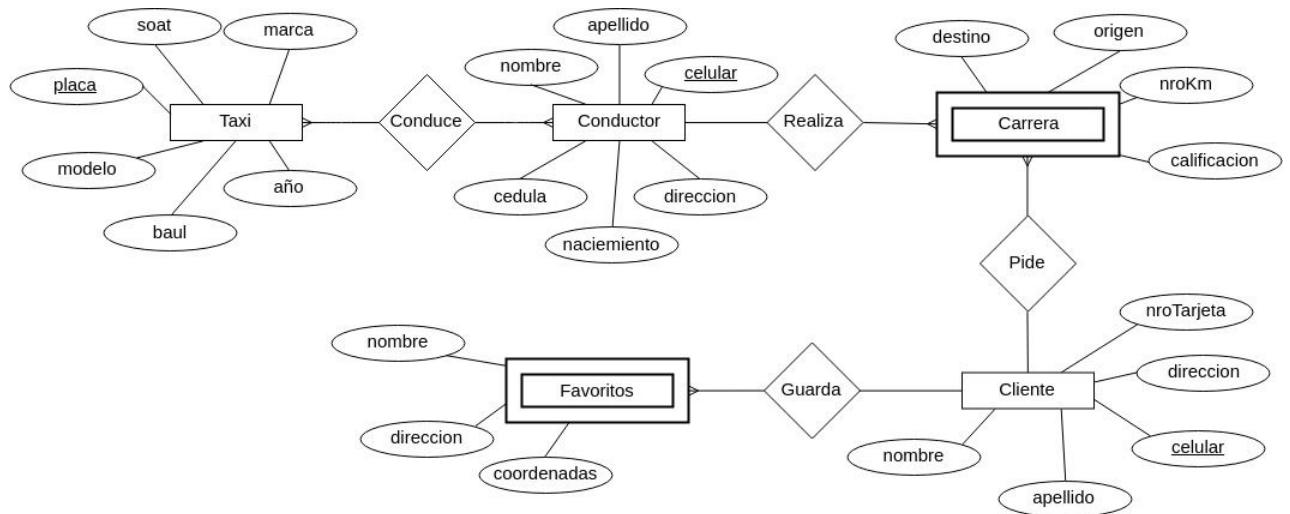
### 2.2.2. Modelo de casos de uso



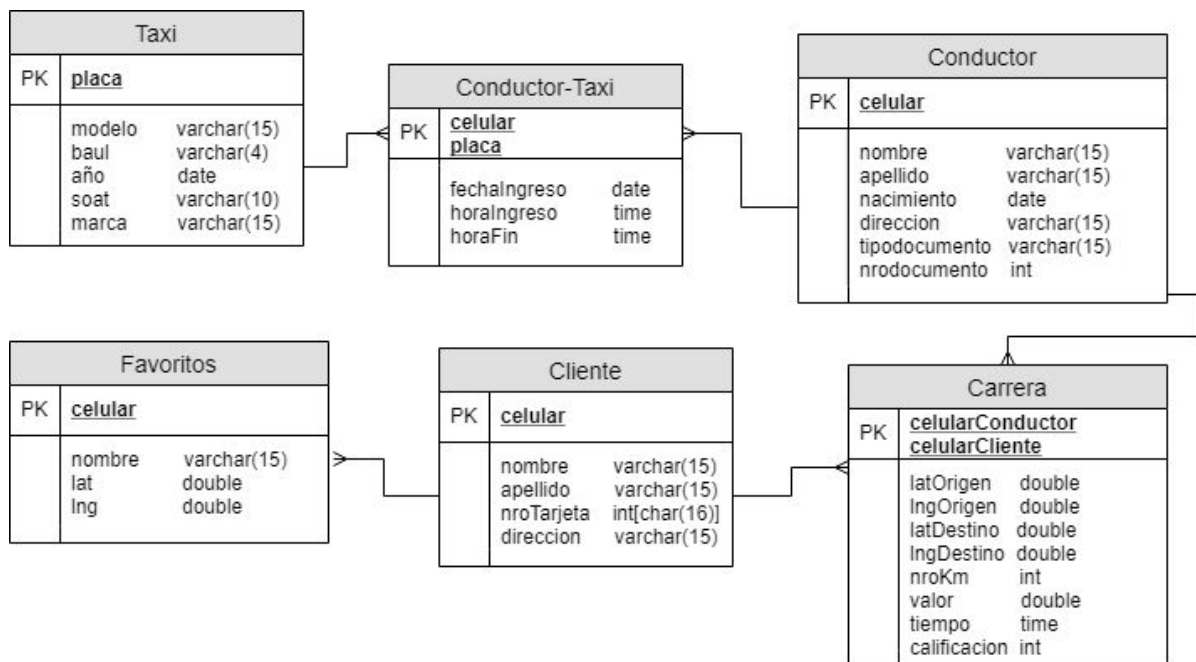
### 2.2.3. Descripción de casos de uso

Actores	Caso de uso	Descripción
Usuario	Solicitar servicio	<p>El caso de uso se inicia una vez el usuario haya iniciado la aplicación, lo que despliega es: iniciar sesión - ingresar punto inicial - ingresar punto final - solicitar servicio</p> <p>Se informa a la aplicación la solicitud del servicio y se buscará el taxista más cercano disponible para realizar dicho recorrido.</p>
Usuario	Pagar servicio	<p>El caso de uso se inicia una vez la carrera haya finalizado, el usuario se verá en la obligación de pagar el servicio, ya sea en efectivo o con tarjeta.</p>
Usuario	Puntuar servicio	<p>El caso de uso se inicia una vez el caso (Pagar servicio/Cobrar servicio) haya finalizado, en este caso de uso se calificará con una puntuación al conductor, ya sea 0 o 5, donde 0 es pésimo y 5 es excelente.</p>
Conductor	Cambiar estado	<p>El caso de uso se inicia desde que la aplicación se inicie, el conductor podrá encontrarse en 4 estados distintos:</p> <ul style="list-style-type: none"><li>-Disponible: no se encuentra realizando carreras, pero se encuentra disponible a realizar nuevos servicios.</li><li>-En recorrido: se encuentra realizando un servicio.</li><li>-Desconectado: no se encuentra realizando viajes por el momento.</li><li>-En camino: ya aceptó un servicio de taxi, sin embargo no ha iniciado el viaje solicitado.</li></ul>
Conductor	Adquirir servicio	<p>El caso de uso se inicia una vez el estado del conductor haya cambiado o esté en estado Disponible, para así poder adquirir nuevos servicios.</p>
Conductor	Realizar servicio	<p>El caso de uso se inicia una vez el estado del conductor cambie de Disponible a En recorrido, a partir de ahí se empiezan a calcular los kilómetros recorridos, de igual forma se inicia la carrera del usuario.</p>
Conductor	Cobrar servicio	<p>El caso de uso se inicia una vez la carrera haya finalizado, se haya calculado el costo de la carrera, el conductor tendrá que cobrarle la carrera al usuario que la haya tomado ya sea en dinero efectivo o tarjeta que tenga registrada.</p>

## 2.2.4. Modelo Entidad-Relación

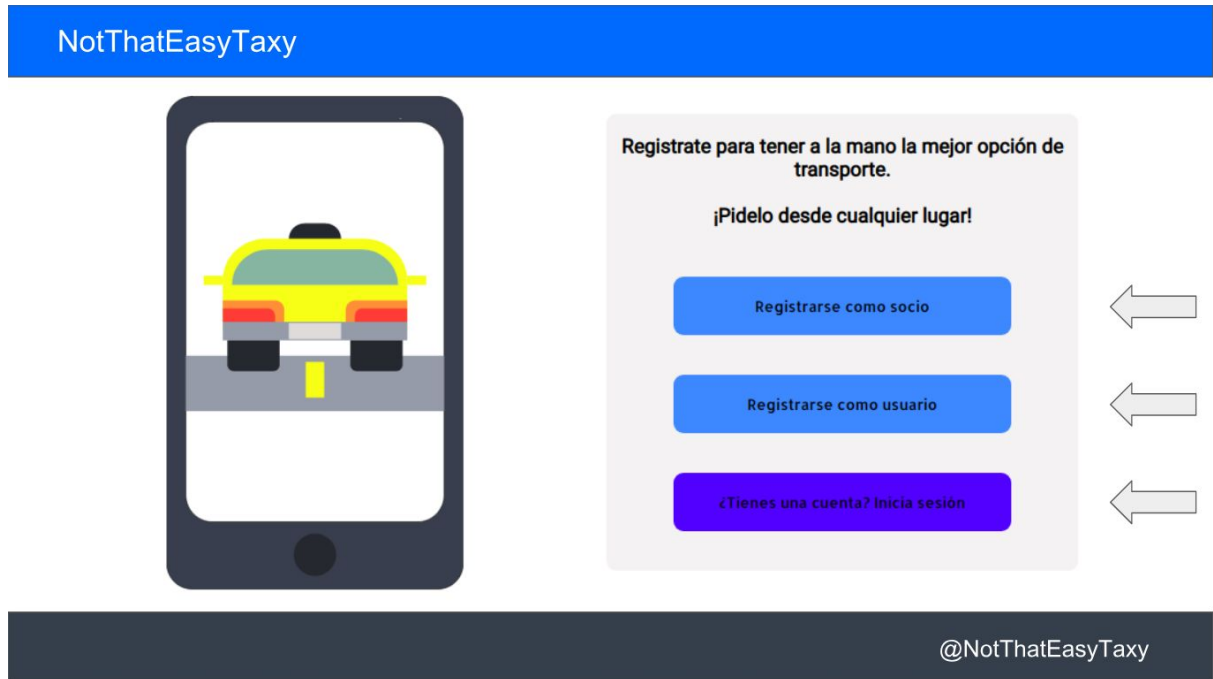


## 2.2.5 Modelo Relacional

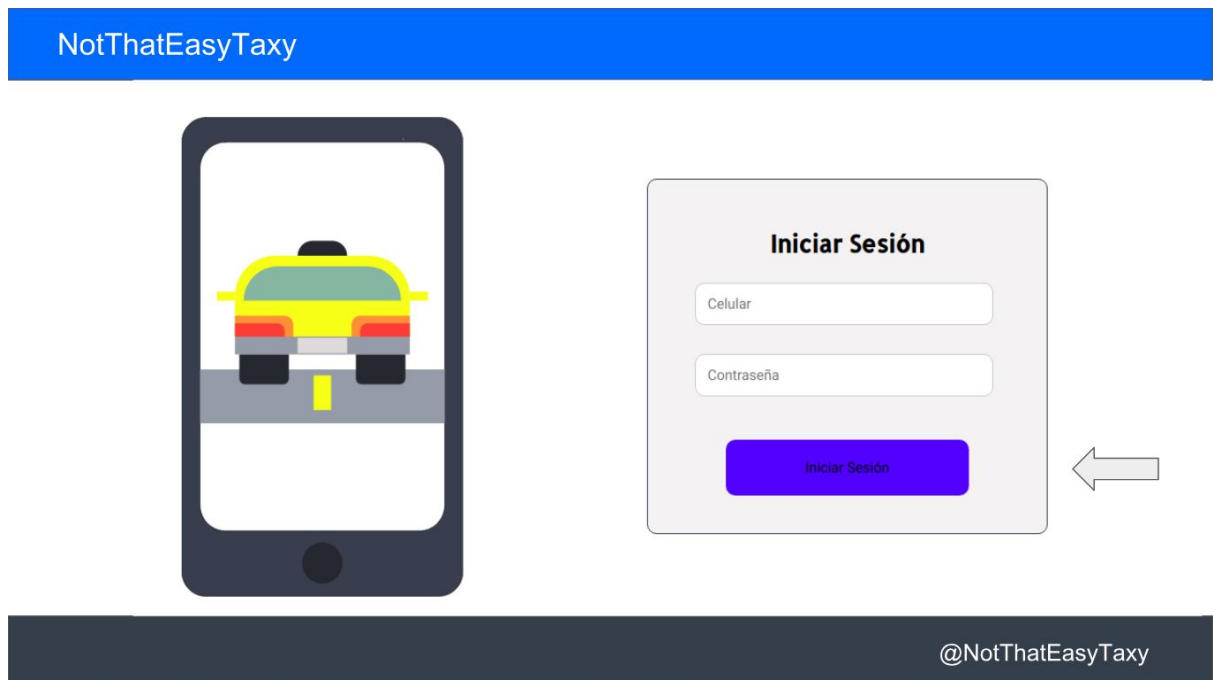


## 2.2.6. Interfaz de usuario

- Ventana inicial - opciones registrarse o iniciar sesión



- Ventana de inicio de sesión





- Ventana de registro conductor

NotThatEasyTaxi

**Regístrate para continuar**

**Seguir**

@NotThatEasyTaxi

- Ventana de registro usuario

NotThatEasyTaxi

**Regístrate para continuar**

**Seguir**

@NotThatEasyTaxi

- Ventana principal del usuario


NotThatEasyTaxy

Usuario

Perfil

Mis viajes

Pedir un viaje

Marzo 3, 2019  
Valor: COP 10000  
Conductor: Pepito Pérez  
Origen: ...  
Destino: ...  
Calificación: 

@NotThatEasyTaxy

- Ventana principal del conductor

NotThatEasyTaxy


Conductor

Perfil

Mis vehículos

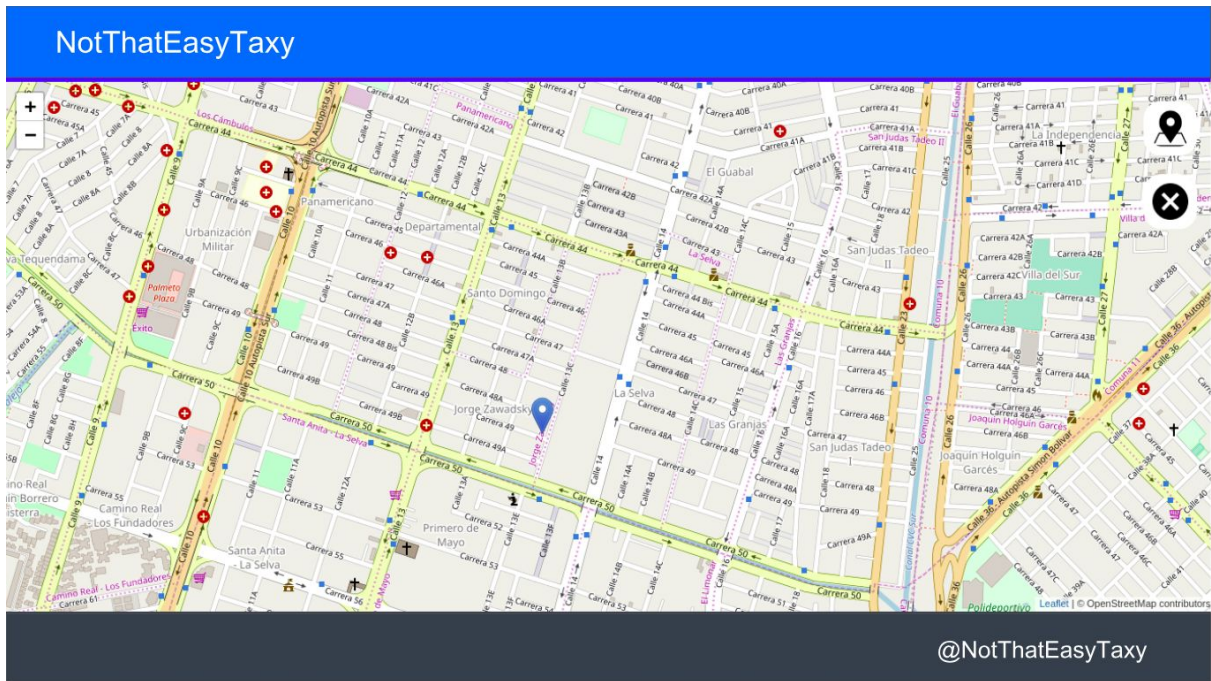
Mis carreras

Mapa

Marzo 3, 2019  
Valor: COP 10000  
Usuario: Pepita Pérez  
Origen: ...  
Destino: ...  
Calificación: 

@NotThatEasyTaxy

- Ventana del mapa



## 2.3. Requerimientos no funcionales

- La aplicación se puede ejecutar en cualquier navegador.
- El tiempo de respuesta de cada búsqueda no debe sobrepasar los 3 segundos.
- La aplicación tendrá mantenimiento dado por el grupo desarrollador.
- La interfaz de usuario debe estar construida bajo un diseño plano, es decir, un diseño que sea fácil de entender y de manejar por el usuario.

## 2.4. Diccionario de datos: Data elements

- Información taxi:

<b>Nombre relación:</b> Taxi <b>Descripción:</b> Tabla que contendrá la información acerca de todos los vehículos de la aplicación.			
Campo	Tipo	Formato	Descripción
placa	varchar(10)	AAA-123	<b>Clave única de cada vehículo.</b>
nroSoat	int	No aplica	Número de seguro que corresponde al vehículo.
marca	varchar(20)	No aplica	Marca del vehículo.
modelo	varchar(10)	No aplica	Modelo del vehículo.
año	date	YYYY-MM-DD	Año del vehículo.
baúl	boolean	True-False	Si el vehículo posee o no baúl.
<b>Campos claves:</b> placa			

- Información de la relación entre el conductor y el taxi:

<b>Nombre relación:</b> Taxi <b>Descripción:</b> Tabla que contendrá la información acerca de todos los vehículos de la aplicación.			
Campo	Tipo	Formato	Descripción
placa	varchar(10)	AAA-123	<b>Clave única de cada vehículo y llave foránea.</b>
celularT	int	No aplica	<b>Clave única del taxista y llave foránea.</b>
<b>Campos claves:</b> placa, celularT			

- Información conductor:

<b>Nombre relación:</b> Conductor <b>Descripción:</b> Tabla que contendrá la información de los conductores de los taxis de la aplicación.			
Campo	Tipo	Formato	Descripción
nombre	varchar(10)	No aplica	Nombre del taxista.
apellido	varchar(10)	No aplica	Apellido del taxista.
celularT	int	No aplica	<b>Teléfono del taxista y Clave única de cada uno.</b>
tipold	varchar(10)	No aplica	Tipo del documento de identificación del taxista.
nrold	int	No aplica	Número del documento de identificación del taxista.
nacimiento	date	YYYY-MM-DD	Fecha de nacimiento del taxista.
dirección	varchar(20)	No aplica	Dirección de residencia del taxista.
<b>Campos claves:</b> celular			

- Información carrera:

<b>Nombre relación:</b> Carrera <b>Descripción:</b> Tabla que contiene la carrera y/o servicio, en la cual están involucrados 1 conductor y 1 cliente.			
Campo	Tipo	Formato	Descripción
celularT	int	No aplica	<b>Celular del taxista que hace la carrera y llave foránea.</b>
celularC	int	No aplica	<b>Celular del cliente que pidió la carrera y llave foránea.</b>
destinolat	float	No aplica	Latitud del punto de destino.
destinlng	float	No aplica	Longitud del punto de destino.
origenlat	float	No aplica	Latitud del punto de origen.
origenlng	float	No aplica	Longitud del punto de origen.
nroKm	float	No aplica	Número de kilómetros del origen al destino.
calificación	int	No aplica	La calificación que se da al servicio.
<b>Campos claves:</b> celularT, celularC			

- Información cliente:

<b>Nombre relación:</b> Cliente <b>Descripción:</b> Tabla que contendrá la información personal de un cliente.			
Campo	Tipo	Formato	Descripción
nombre	varchar(10)	No aplica	Nombre del cliente.
apellido	varchar(10)	No aplica	Apellido del cliente.
celularC	int	No aplica	<b>Celular del cliente y llave primaria.</b>
dirección	varhcar(20)	No aplica	Dirección de residencia del cliente.
nroTarjeta	int	No aplica	Número de la tarjeta de crédito del cliente (si tiene), sino, no aplica.
<b>Campos claves:</b> celularC			

- Información favoritos:

<b>Nombre relación:</b> Favoritos <b>Descripción:</b> Tabla que contiene el nombre e información de los lugares favoritos o que más habitan los clientes.			
Campo	Tipo	Formato	Descripción
celularC	int	No aplica	<b>Celular del cliente y llave foránea.</b>
nombre	varchar(10)	No aplica	Nombre del lugar.
dirección	varchar(10)	No aplica	Dirección del lugar.
lat	float	No aplica	Latitud del lugar.
lng	float	No aplica	Longitud del lugar.
<b>Campos claves:</b> celularC			

## 3. MANUAL TÉCNICO

### 3.1. BASE DE DATOS

#### 3.1.1. Índices en la base de datos

- Cliente: se creó un índice único para celular y contraseña, para consultas, serán los más utilizados, ya que se requiere iniciar sesión ingresando ambos valores.
- Conductor: se crearon índices únicos para celular y contraseña, para que al iniciar sesión, las consultas sean más rápidas.
- Taxi: se creó un índice único para placa, ya que es la primary key, y lo más congruente sería hacer consultas por este atributo si en algún momento lo necesitamos. Se creó también un índice por año, por si se desean consultar a los carros, en orden ascendiente o descendiente de este.
- Carrera: se decidió crear un índice único para celular de conductor y cliente, ya que a la hora de buscar una carrera en particular, serían necesarios estos campos. También se creó un índice para valor, por si se desean consultar los valores de la carrera de mayor a menor, o de menor a mayor.
- Favoritos: se creó un índice único a favoritos con el celular, para consultar los lugares favoritos de algún usuario.
- Conductor\_taxi: se creó un índice único para celular y placa.

#### 3.1.2. Integridad referencial

Cada vez que se actualice un dato de una tabla específica, también se actualizará en las tablas a las que esta haga referencia.

UPDATE del celular CONDUCTOR: Actualiza el atributo de las tablas CONDUCTOR, CONDUCTOR\_TAXI y CARRERA.

UPDATE del celular en CLIENTE: Actualiza el atributo de las tablas CLIENTE, CARRERA y FAVORITOS.

UPDATE de la placa de TAXI: Actualiza el atributo de las tablas TAXI y CONDUCTOR\_TAXI.

#### 3.1.3. Permisos

Creamos distintos roles para la manipulación de la bases de datos:

1. **Admin:** tiene roles de super usuario, puede crear, editar, eliminar, seleccionar.
2. **Eliminación:** tiene permisos para eliminar (DELETE), este será usado a la hora de que un usuario se quiera dar de baja en la cuenta, entonces se eliminará por medio de este rol.
3. **Inserción:** tiene permisos para insertar datos en la BD (INSERT), será utilizado a la hora de crear un nuevo conductor o cliente, igualmente a la hora de insertar una carrera, un taxi, o demás elementos que contenga la BD.

4. **Modificación:** tiene permisos de UPDATE, para actualizar los datos que un usuario requiera, o al momento de actualizar los lugares favoritos de una persona.
5. **Selección:** (SELECT), tiene permiso de selección de datos, nos funcionará al momento de consultar si un celular coincide con una contraseña, para iniciar sesión e ir al mapa.

### 3.1.4. Procedimientos almacenados

Función	Trigger	Descripción
<pre>CREATE OR REPLACE FUNCTION f_valor_carrera() RETURNS TRIGGER AS \$\$ BEGIN IF (TG_OP = 'INSERT') THEN     UPDATE carrera     SET VALOR = nroKm*1000; END IF; RETURN NULL; END; \$\$ LANGUAGE plpgsql;</pre>	<pre>CREATE TRIGGER insertar_valor_carrera AFTER INSERT ON carrera FOR EACH ROW EXECUTE PROCEDURE f_valor_carrera();</pre>	<p>Función para actualizar el valor de la carrera, de acuerdo a los kilómetros y al tiempo de recorrido. Nosotros decidimos que su costo será el número de kilómetros * 1000.</p>
<pre>CREATE OR REPLACE FUNCTION f_fecha_carrera() RETURNS TRIGGER AS \$\$ BEGIN IF (TG_OP = 'INSERT') THEN     UPDATE carrera     SET fecha_carrera = CURRENT_DATE; END IF; RETURN NULL; END; \$\$ LANGUAGE plpgsql;</pre>	<pre>CREATE TRIGGER insertar_fecha_carrera AFTER INSERT ON carrera FOR EACH ROW EXECUTE PROCEDURE f_fecha_carrera();</pre>	<p>Procedimiento almacenado que me actualiza la fecha de la carrera con la fecha del mismo día que se realiza la misma.</p>
<pre>CREATE OR REPLACE FUNCTION f_fecha_ingreso() RETURNS TRIGGER AS \$\$</pre>	<pre>CREATE TRIGGER insertar_fecha_ingreso AFTER INSERT</pre>	<p>Este hace algo similar a la anterior, pero lo que hace es actualizar la fecha de</p>



<pre> BEGIN IF (TG_OP = 'INSERT') THEN     UPDATE conductor_taxi     SET fechaingreso = CURRENT_DATE; END IF; RETURN NULL; END; \$\$ LANGUAGE plpgsql; </pre>	<pre> ON conductor_taxi FOR EACH ROW EXECUTE PROCEDURE f_fecha_ingreso(); </pre>	<p>ingreso del taxista a la aplicación.</p>
<pre> CREATE OR REPLACE FUNCTION f_hora_ingreso() RETURNS TRIGGER AS \$\$ BEGIN IF (TG_OP = 'INSERT') THEN     UPDATE conductor_taxi     SET horaingreso = LOCALTIME; END IF; RETURN NULL; END; \$\$ LANGUAGE plpgsql; </pre>	<pre> CREATE TRIGGER insertar_hora_ingreso AFTER INSERT ON conductor_taxi FOR EACH ROW EXECUTE PROCEDURE f_hora_ingreso(); </pre>	<p>Esta función me actualiza la hora de ingreso del taxista a la aplicación.</p>

### 3.1.5. Vistas

Vista	Descripción
<pre> CREATE VIEW viajesCliente AS (SELECT carrera.celularcliente as cliente, conductor.nombre as conductor, carrera.nrokm as km, carrera.valor, carrera.calificacion FROM conductor INNER JOIN carrera ON celularConductor=celular); </pre>	<p>Se creó esta vista, con la intención de que sea más fácil al mostrar la información de los viajes al cliente.</p>

## 3.2. Arquitectura de software

Como ya sabemos, Cliente/Servidor es una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente.

Con esta arquitectura buscamos que el cliente, en este caso, ya sea el usuario o el conductor, le hagan peticiones al servidor como inicio de sesión, solicitud de carreras, etc, y que el servidor envíe uno o varios mensajes con la respuesta requerida.

### 3.2.1. Cliente

El cliente normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos.

Aquí es donde nosotros diseñamos la interfaz gráfica de la aplicación: los formularios de registro y de inicio de sesión, igualmente el mapa y donde cada cliente pueda ver los viajes que tiene, este lado se desarrolló con:

- Css: el estilo del diseño de la página.
- Html: la maquetación de los módulos del sistema.
- JavaScript: las funcionalidades que tiene el sistema, desde aquí se hacen peticiones al servidor, como verificar si los datos que se ingresan al formulario, se encuentran en la base de datos y permitir el inicio de sesión. También el registro de nuevas rutas de un cliente.

### 3.2.2. Servidor

Es el encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él.

Aquí nosotros tenemos la conexión con la base de datos, las consultas de datos, se valida si los usuarios están loggeados para permitir el acceso a la aplicación y así mismo se da lo que el cliente requiere, ya sea la solicitud de una carrera, la puntuación a un conductor, entre otros...este lado fue desarrollado en:

- Posgresql: para la base de datos, la creación de los trigger, la creación de usuarios.
- Nodejs: para la creación de las respuestas al cliente, se verifica que el usuario esté loggeado y así mismo le devuelve las peticiones al cliente.

### 3.2.3. Modelo vista controlador

Utilizamos el modelo vista controlador, donde el modelo es la base de datos donde se aloja la información de los usuarios y las carreras que se registren. La vista es la interfaz gráfica de la aplicación que se le mostrará al usuario final y el controlador son las funciones de la API que permiten que el usuario se comuniquen con la base de datos en un alto nivel del lenguaje.

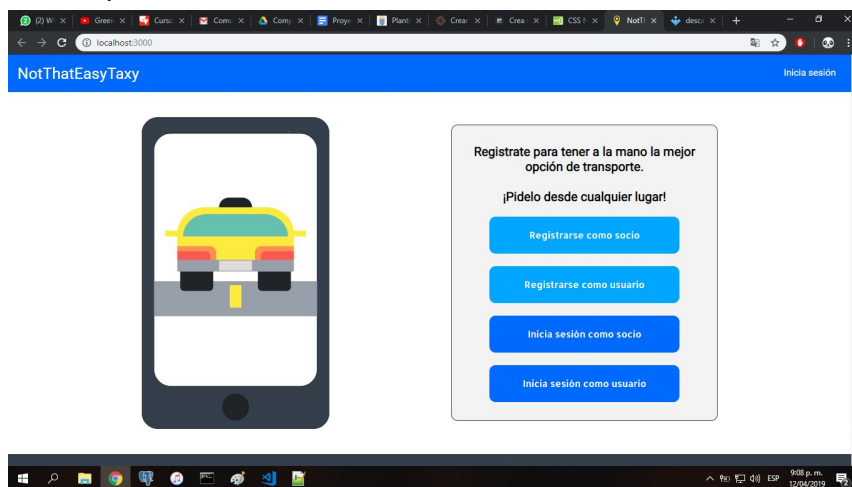
## 4. MANUAL DE USUARIO

### 4.1. Introducción al manual de usuario

El presente documento pretende mostrar al usuario el funcionamiento de la aplicación NotThatEasyTaxy para servicios de taxi.

#### 4.1.1. Pantalla inicial

La pantalla de inicio de la aplicación da la bienvenida al usuario al sistema y ofrece las distintas alternativas para inicio de sesión o creación de usuario nuevo.



#### 4.1.2. Registro como socio

La pantalla de registro como socio contiene un formulario para que la persona que quiera trabajar como taxista, por medio de la aplicación, digite sus datos y pueda empezar a trabajar en ello.

Igualmente, allí mismo se solicitan los datos del taxi que el taxista va a utilizar, como la placa del taxi, el soat, modelo, año, entre otros.

A screenshot of a web browser displaying a form titled "Registro de taxi". The form is centered on a light gray background. It contains six input fields stacked vertically: "Placa", "Modelo", "Basil", "Año", "Soat", and "Marca". Below these fields is a blue button labeled "Continuar". The browser's address bar shows "localhost:3000/Form/Driver". The Windows taskbar is visible at the bottom.

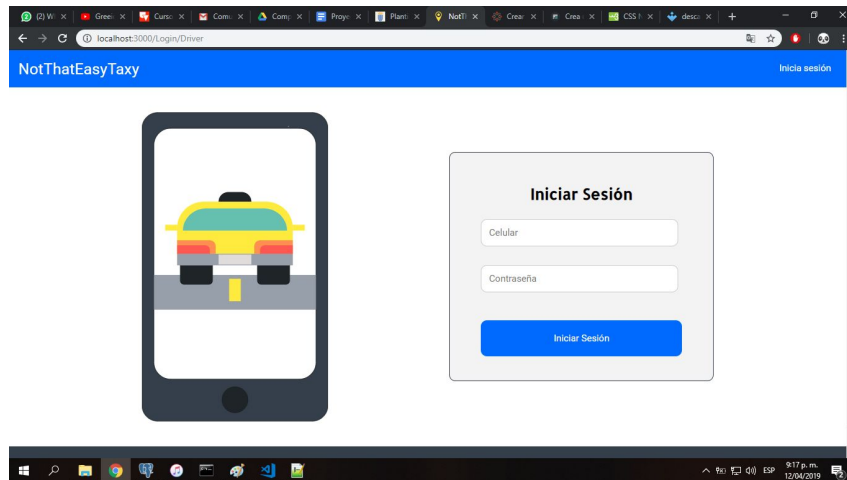
### 4.1.3. Registro como usuario

En esta pantalla se le solicita la información personal al cliente que va a usar la aplicación, igualmente se le solicita una tarjeta, para que realice sus pagos.

A screenshot of a web browser displaying a form titled "Regístrate para continuar". The form is centered on a light gray background. It contains six input fields: "Nombre" and "Apellido" (side-by-side), "Celular", "Dirección", "Número de tarjeta", "Contraseña", and "Repetir contraseña". Below these fields is a blue button labeled "Continuar". The browser's address bar shows "localhost:3000/Form/User". The Windows taskbar is visible at the bottom.

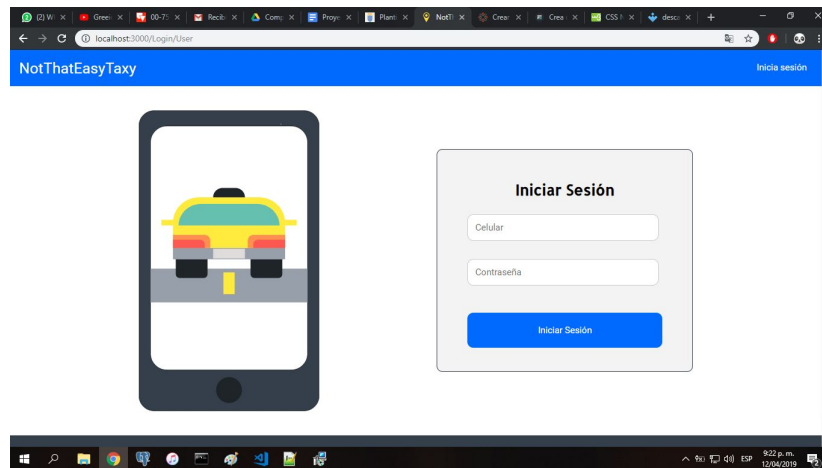
### 4.1.4. Inicio de sesión como socio

Aquí se le solicita al socio-conductor, que escriba su celular y la contraseña con que se registró en la aplicación, para iniciar sesión y ver las demás funcionalidades de la aplicación.



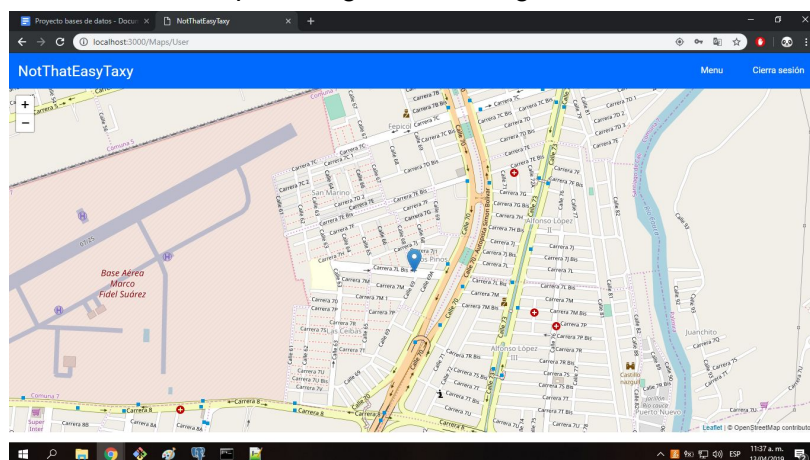
#### 4.1.5. Inicio de sesión como usuario

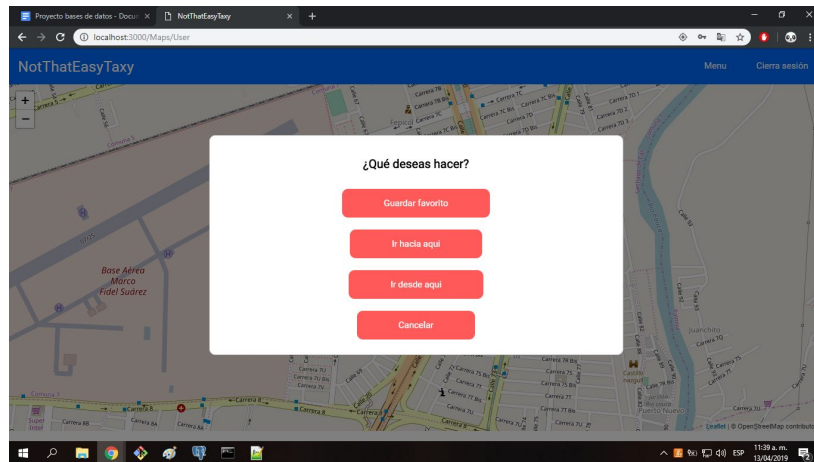
Aquí se le solicita al usuario-cliente, que escriba su celular y la contraseña con que se registró en la aplicación, para iniciar sesión y empezar a solicitar nuevos viajes.



#### 4.1.6. Mapa

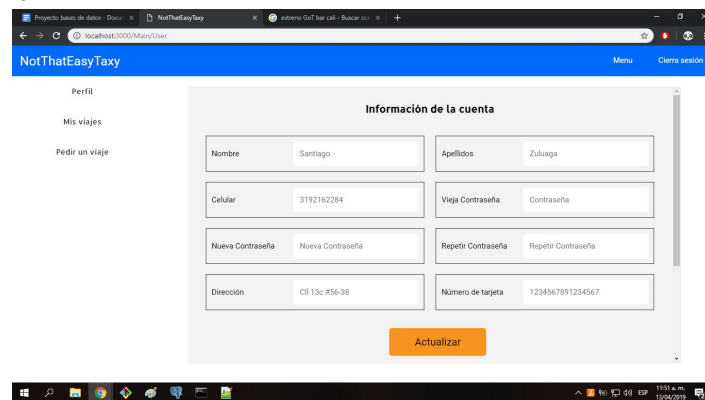
Aquí le aparecerá la ubicación actual, ya sea al cliente o al conductor, donde al cliente le permitirá elegir su origen y su destino a la hora de hacer una ruta, esto lo hace con el click derecho del ratón. Esto también permite guardar un lugar favorito del cliente.





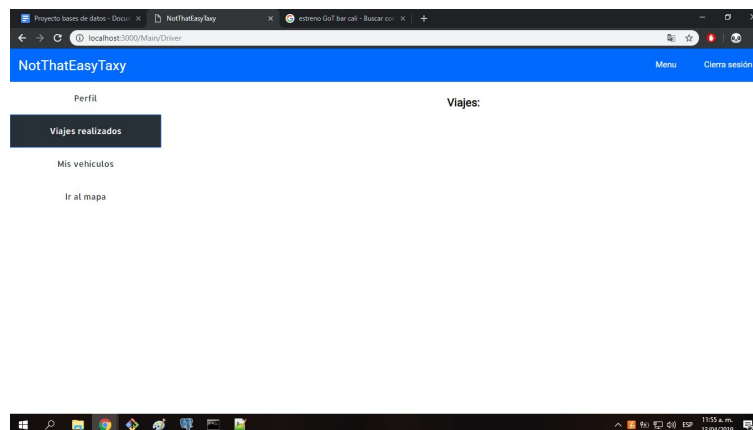
#### 4.1.7. Perfil del cliente y/o conductor

En esta sección aparece la información personal, ya sea del conductor o del cliente, depende de cual haya iniciado sesión, aquí se podrá actualizar esta información.



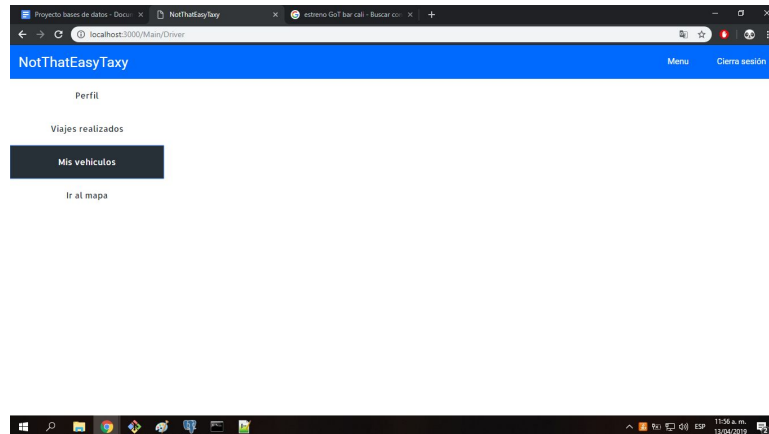
#### 4.1.8. Viajes realizados

En esta pantalla aparecen todos los viajes que ha realizado ya sea el cliente o el conductor, guarda información de precio, kilómetros y calificación.



### 4.1.9. Mis vehículos

Aquí estarán almacenados los vehículos que el conductor tiene registrados en la base de datos.



## 4.2. Participantes

Nombre participante	Rol
María Paula Mosquera Rengifo	Diseñadora de bases de datos, triggers y funciones de consultas para la aplicación.
Santiago Zuluaga Hernández	Desarrollador full-stack.

## 5. Glosario

A continuación se indican los términos utilizados, en el presente documento.

Término	Descripción
NTET	Siglas de NotThatEasyTaxi.
GPS	Coordenadas de ubicación espacial de una persona.