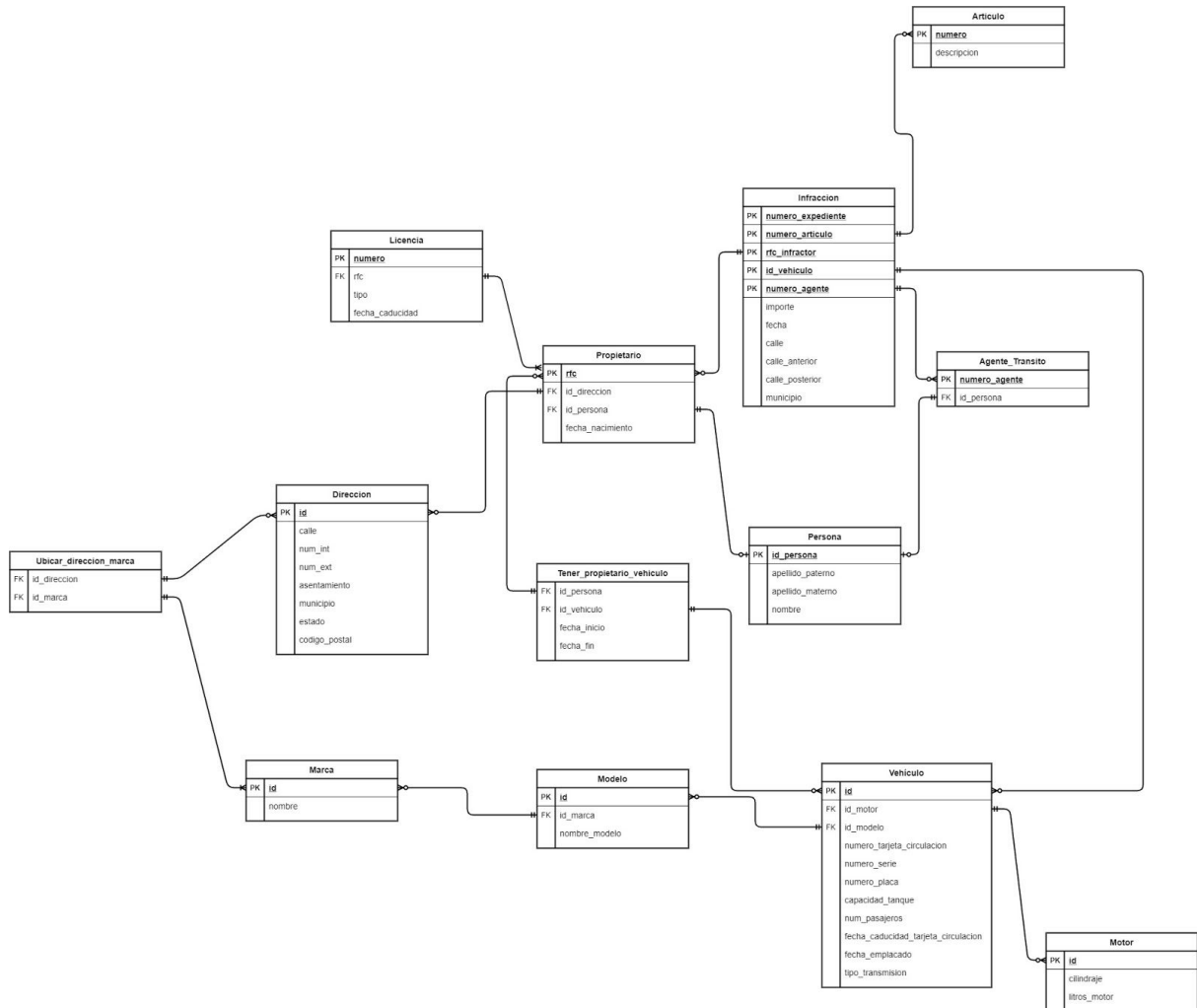


## 2. Traducción al modelo relacional y dependencias funcionales

El modelo se encuentra en la carpeta Modelo Relacional del proyecto. Esta carpeta contiene el diagrama en formato .drawio, es decir el formato del diagramador usado y en formato .jpg para visualizar como imagen



### Dependencias funcionales

**Articulo**(numero,descripcion).

F{numero->descripcion}

**Infraccion**(numero\_expediente, numero\_articulo, rfc\_infractor,id\_vehiculo, numero\_agente, importe,fecha,calle, calle\_anterior, calle\_posterior,municipio).

F{numero\_expediente->importe,fecha,calle, calle\_anterior, calle\_posterior,municipio;  
numero\_expediente->numero\_articulo, rfc\_infractor, numero\_agente,id\_vehiculo;}

**Licencia** (numero, rfc, tipo, fecha\_caducidad).

F{numero->rfc,fecha\_caducidad,tipo}

**Direccion**(id, calle, num\_int, num\_ext, asentamiento, municipio, estado, codigo\_postal)

F{id->calle, num\_int, num\_ext, asentamiento, municipio, estado, codigo\_postal}

**Marca**(id,nombre)

F{id->nombre}

**Modelo**(id, id\_marca,nombre\_modelo)

F{id->id\_marca,nombre\_modelo}

**Ubicar\_direccion\_marca**(id\_direccion, id\_marca)

F{}

**Vehiculo**(id, id\_motor, id\_modelo, numero\_tarjeta\_circulacion, numero\_serie, numero\_placa, capacidad\_tanque, numero\_pasajeros, fecha\_caducidad\_tarjeta\_circulacion, fecha\_emplacado, tipo\_transmision)

F{id->id\_motor,id\_modelo,numero\_tarjeta\_circulacion, numero\_serie, numero\_placa, numero\_pasajeros,fecha\_caducidad\_tarjeta\_circulacion,tipo\_transmision;  
numero\_tarjeta\_circulacion->id,fecha\_caducidad\_tarjeta\_circulacion;  
numero\_serie->id, num\_pasajeros, capacidad\_tanque;  
numero\_placa->id, fecha\_emplacado}

**Motor**(id, cilindraje, litros\_motor)

F{id->cilindraje, litros\_motor}

**Persona**(id\_persona, apellido\_paterno, apellido\_materno, nombre)

F{id\_persona->apellido\_paterno, apellido\_materno, nombre}

**Propietario**(rfc,id\_direccion, id\_persona, fecha\_nacimiento)

F{rfc->id\_direccion, id\_persona, fecha\_nacimiento; id\_persona->rfc}

**Agente\_Transito**(numero agente, idpersona)

F{numero\_agente->id\_persona; id\_persona->numero\_agente }

**Tener\_Propietario\_Vehiculo**(id\_persona, id\_vehiculo, fecha\_inicio, fecha\_fin)

F{id\_persona,id\_vehiculo->fecha\_inicio, fecha\_fin}

### 3. Normalización

Utilizaremos BCNF para la normalización del modelo relacional

**Articulo**(numero,descripcion).

**F**{numero->desripcion}

Observemos que numero es llave de la relación Articulo pues {numero}\*={numero,descripcion} así el lado izquierdo de la única dependencia funcional que tenemos en la tabla es una llave, por lo tanto la relación está en BCNF

**Infraccion**(numero\_expediente, numero\_articulo, rfc\_infractor, id\_vehiculo, numero\_agente, importe, fecha, calle, calle\_anterior, calle\_posterior, municipio).

**F**{numero\_expediente->importe, fecha, calle, calle\_anterior, calle\_posterior, municipio; numero\_expediente->numero\_articulo, rfc\_infractor, numero\_agente, id\_vehiculo}

Fijémonos que {numero\_expediente}\*={numero\_expediente, numero\_articulo, rfc\_infractor, numero\_agente, importe, fecha, calle, calle\_anterior, calle\_posterior, municipio}, por lo tanto numero\_expediente es una llave de Infraccion, y dado que es el único elemento que aparece del lado izquierdo de las dependencias funcionales la relación se encuentra en BCNF

**Licencia** (numero, rfc, tipo, fecha\_caducidad).

**F**{numero->rfc, fecha\_caducidad, tipo}

Fijémonos que {numero}\*={numero rfc, fecha\_caducidad, tipo}, por lo tanto numero es una llave de Licencia, y dado que es el único elemento que aparece del lado izquierdo de la dependencia funcional en la relación, la relación se encuentra en BCNF

**Direccion**(id, calle, num\_int, num\_ext, asentamiento, municipio, estado, codigo\_postal)

**F**{id->calle, num\_int, num\_ext, asentamiento, municipio, estado, codigo\_postal}

(Obs. Es evidente que el id es el único elemento que puede establecer dependencia funcional para el resto de elementos en Dirección, el único caso que no es tan trivial es el código postal con el asentamiento, el municipio y el estado, pero debido a que existen diversos modelos de códigos postales en distintos países y en particular al

menos México y Alemania comparten un esquema de 6 dígitos, además que en la especificación debido a que tenemos una relación entre marca y dirección no podemos asegurar que las direcciones provistas sean solo de México dado que existen muchas marcas que fabrican vehículos en México y los importan, puede ocurrir que una armadora se encuentre en otro país y tenga el mismo código postal, por lo tanto no podemos asegurar ninguna dependencia funcional entre el código postal y el resto de atributos antes mencionados)

Fijémonos que  $\{id\}^* = \{id, calle, num\_int, num\_ext, asentamiento, municipio, estado, codigo\_postal\}$ , por lo tanto id es una llave de Direccion, y dado que es el único elemento que aparece del lado izquierdo de las dependencias funcionales la relación se encuentra en BCNF

**Marca**(id,nombre)

**F**{id->nombre}

Fijémonos que  $\{id\}^* = \{id, nombre\}$ , por lo tanto id es una llave de Marca, y dado que es el único elemento que aparece del lado izquierdo de la dependencia funcional de la relación, esta se encuentra en BCNF

**Modelo**(id, id\_marca,nombre\_modelo)

**F**{id->id\_marca,nombre\_modelo}

Fijémonos que  $\{id\}^* = \{id, id\_marca, nombre\_modelo\}$ , por lo tanto id es una llave de Modelo, y dado que es el único elemento que aparece del lado izquierdo de la dependencia funcional de la relación, esta se encuentra en BCNF

**Ubicar\_direccion\_marca**(id\_direccion, id\_marca)

**F**{}

Dado que no tenemos dependencias funcionales agregamos a F la dependencia id\_direccion, id\_marca->id\_direccion, id\_marca y dado que las dependencias triviales no generan violaciones a la BCNF, esta tabla se encuentra en BCNF

**Vehiculo**(id, id\_motor, id\_modelo, numero\_tarjeta\_circulacion, numero\_serie, numero\_placa, capacidad\_tanque, numero\_pasajeros, fecha\_caducidad\_tarjeta\_circulacion, fecha\_emplacado, tipo\_transmision)

**F**{id->id\_motor,id\_modelo,numero\_tarjeta\_circulacion, numero\_serie, numero\_placa, numero\_pasajeros,fecha\_caducidad\_tarjeta\_circulacion,tipo\_transmision;  
numero\_tarjeta\_circulacion->id,fecha\_caducidad\_tarjeta\_circulacion;  
numero\_serie->id, num\_pasajeros, capacidad\_tanque;  
numero\_placa->id, fecha\_emplacado}

Primero observemos que {id}\*={id\_motor, id\_modelo, numero\_tarjeta\_circulacion, numero\_serie, numero\_placa, capacidad\_tanque, numero\_pasajeros, fecha\_caducidad\_tarjeta\_circulacion, fecha\_emplacado,tipo\_transmision}, de lo cual podemos decir que id es una llave de Vehículo, además observemos que numero\_tarjeta\_circulacion, numero\_serie y numero\_placa generan a id, así podemos decir que estos 3 campos también son llaves de Vehículo y dado que estos 4 campos son los únicos que aparecen del lado izquierdo de las dependencias funcionales y los 4 son llaves de manera individual, la relación se encuentra en BCNF.

**Motor**(id, cilindraje, litros\_motor)

**F**{id->cilindraje, litros\_motor}

Fijémonos que {id}\*={id,cilindraje, litros\_motor}, por lo tanto es una llave de Motor, y dado que es el único elemento que aparece del lado izquierdo de la dependencia funcional de la relación, esta se encuentra en BCNF.

**Persona**(id\_persona, apellido\_paterno, apellido\_materno, nombre)

**F**{id\_persona->apellido\_paterno, apellido\_materno, nombre}

Fijémonos que {id\_persona}\*={id\_persona, apellido\_paterno, apellido\_materno, nombre}, por lo tanto id\_persona es una llave de Persona, y dado que es el único elemento que aparece del lado izquierdo de la dependencia funcional de la relación, esta se encuentra en BCNF

**Propietario**(rfc,id\_direccion, id\_persona, fecha\_nacimiento)

**F**{rfc->id\_direccion, id\_persona, fecha\_nacimiento; id\_persona->rfc}

Primero observemos que {id}\*={rfc,id\_direccion, id\_persona, fecha\_nacimiento}, de lo cual podemos decir que id es una llave de Propietario, además observemos que id\_persona genera a id, así podemos decir que éste campo también es llave de Propietario y dado que estos 2 campos son los únicos que aparecen del lado izquierdo de las dependencias funcionales y los 2 son llaves de manera individual, la relación se encuentra en BCNF.

**Agente\_Transito**(numero agente, idpersona)

**F{numero\_agente->id\_persona; id\_persona->numero\_agente}**

Observemos que, numero agente y id\_persona se generan mutuamente y son los únicos campos de la relación, de tal modo que ambos son llaves y ambos aparecen en el lado izquierdo de la dependencia, por lo tanto las dependencias no violan las condiciones de BCNF y la tabla está normalizada.

**Tener\_Propietario\_Vehiculo(id\_persona, id\_vehiculo, fecha\_inicio, fecha\_fin)**

**F{id\_persona,id\_vehiculo->fecha\_inicio, fecha\_fin}**

Observemos que {id\_persona,id\_vehiculo}\*={id\_persona, id\_vehiculo, fecha\_inicio, fecha\_fin} por lo tanto (id\_persona,id\_vehiculo) es una llave para Tener\_Propietario\_Vehiculo, además veamos que sólo esta tupla aparece del lado izquierdo de la dependencia funcional asociada con la relación, por lo tanto no se violan las restricciones de la BCNF, y así la tabla ya se encuentra normalizada.

Observemos que al normalizar el modelo relacional resultado de la traducción no obtuvimos cambios, por lo tanto el modelo anterior es el que utilizaremos para la creación de la base y podemos asegurar dada la traducción y que el modelo se encuentra normalizado que no tenemos duplicidad de información.

Ahora utilizaremos **PostgreSQL** como el lenguaje para crear, poblar y generar las consultas a nuestra base de datos ya que el lenguaje tiene varias herramientas que nos ayudarán a poblar la base en particular las funciones que vienen implementadas en el lenguaje PLPGSQL nos servirán mucho, además que la conexión a la aplicación web que crearemos usando **php** y **Laravel** será muy sencilla.

#### **4. Scripts SQL (creación y poblamiento)**

Los scripts para la creación de la base se encuentran en el archivo creacion.sql en la carpeta SQL/Creación. Además para generar los datos se utilizará la herramienta **Mockaroo** y se adjuntarán los archivos .csv generados en la misma.

Puntos a notar:

- a. Los datos generados en cada tabla tendrán una captura de pantalla con la configuración que se usó en Mockaroo para crear los datos.
- b. Las tablas ubicar\_direccion\_marca y tener\_propietario\_vehiculo recibieron una columna adicional llamada id para el correcto funcionamiento de la base y la aplicación

## Población de las tablas

Todos los scripts de población de la base se encuentran en la carpeta del proyecto SQL/Poblado con el nombre de la tabla asociada a dicho script, para no tener problemas a la hora de poblar la base de datos ejecutar los scripts en el orden en el que se encuentran descritas las tablas en este documento.

Además la configuración de Mockaroo de todos los campos de las tablas se encuentran como imágenes en la carpeta Mockaroo del proyecto

### Dirección

Debido a que no se encontró una base con nombres de calles de la cdmx se decidió usar la opción de nombres de calle provista por la herramienta con el detalle de que están en inglés, los asentamientos fueron obtenidos de los datos abiertos de la Ciudad de México en el siguiente [link](#), en el estado solo se utilizó la CDMX y el código postal fue creado de manera aleatoria respetando que para todos los cp de la Ciudad de México empiezan y terminan en 0

Field Name	Type	Options
id	Row Number	blank: 0 % fx
calle	Street Name	blank: 0 % fx
num_int	Street Number	blank: 0 % fx
num_ext	Street Number	blank: 0 % fx
asentamiento	Custom List	LA JOYA, MIGUEL HIDALGO, EL MIRADOR, EL RODEO, FRANCISCO VILLA, LA JO random blank: 0 % fx
municipio	Custom List	Álvaro Obregón, Azcapotzalco, Benito Juárez, Coyoacán, Cuajimalpa de Morel random blank: 0 % fx
estado	Custom List	CDMX random blank: 0 % fx
codigo_postal	Character Sequence	##### blank: 0 % fx

### Marca:

Las marcas se obtuvieron del siguiente [link](#), el cual incluye las marcas de autos que se venden en México.

Field Name	Type	Options
id	Row Number	blank: 0 % fx
nombre	Custom List	Acura,Alfa Romeo,ASTON MARTIN,Audi,BAIC,Bentley,BMW,BUICK,Cadillac,Che sequential blank: 0 % fx

Add another field

---

# Rows: 37 Format: SQL Table Name: marca ☐ Include create table

### Ubicar\_direccion\_marca:

Se toma un número aleatorio entre 1 y 1000 para las direcciones mientras que se toman las 37 marcas y se asigna el valor resultado a la tabla .



Field Name	Type	Options
id	Row Number	blank: 0 % $\text{fx}$ ×
id_direccion	Custom List	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20 random blank: 0 % $\text{fx}$ ×
id_propietario	Custom List	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20 sequential blank: 0 % $\text{fx}$ ×
Add another field		

---

# Rows: 40 Format: SQL Table Name: ubicar\_direccion\_marca ☐ include create table

### Modelo:

Se generaron 150 modelos usando el conjunto de datos predeterminado de la herramienta tomando de manera aleatoria la marca a la cual se lo asignamos, generando tuplas que si bien no corresponden a los que existen en la vida real generan un campo de datos lo suficientemente grande como para hacer las pruebas en el sistema.

Field Name	Type	Options
id	Row Number	blank: 0 % $\text{fx}$ ×
cilindraje	Custom List	4,6,8,10,12 random blank: 0 % $\text{fx}$ ×
litros_motor	Number	min: 1.0 max: 3.0 decimals: 1 blank: 0 % $\text{fx}$ ×
Add another field		

---

# Rows: 150 Format: SQL Table Name: motor ☐ include create table

### Motor:

Se generaron 25 combinaciones de motores con distintas configuraciones de número de cilindros y litros de capacidad en el motor, al igual que en el caso anterior puede haber configuraciones que no se encuentren en la realidad pero los datos funcionan para probar el sistema.

Field Name	Type	Options
id	Row Number	blank: 0 % $\text{fx}$ ×
cilindraje	Custom List	4,6,8,10,12 random blank: 0 % $\text{fx}$ ×
litros_motor	Number	min: 1.0 max: 3.0 decimals: 1 blank: 0 % $\text{fx}$ ×
Add another field		

---

# Rows: 150 Format: SQL Table Name: motor ☐ include create table

Vehiculo:

Se generaron 1000 vehículos seleccionando para cada uno, un motor de los 25 que generamos en la tabla y los 150 modelos de autos que generamos. Generamos el número de la tarjeta de circulación siguiendo como parámetro que las que son asignadas a autos particulares comienzan con la letra A seguido de 7 números, el número de serie del vehículo tomamos una estructura bastante similar a la que se us en la realidad pero al generar aleatoriamente todos estos datos no trae toda la información que en los autos realmente, sin embargo la estructura sintáctica es similar (es decir en los espacio que solo pueden ir números o cadenas se respetó dicha nomenclatura al igual que en las posiciones que puede haber letras y números también). Para el número de placas se utilizó la estructura comúnmente vista en las placas de la CDMX que contiene 3 letras al principio y 4 números al final. Para la capacidad del tanque se tomaron los datos más comunes entre los autos particulares y camionetas. Por último en las fechas de emplacado y caducidad de la tarjeta de circulación se generaron aleatoriamente entre el día 13 de septiembre del 2020 y el 1 de enero del 2000 y aleatoriamente escogimos entre estándar y manual para representar el tipo de transmisión en una función de php que estará anexada más adelante.

Field Name	Type	Options
id	Row Number	blank: <input type="checkbox"/> % <input checked="" type="checkbox"/> fx ×
id_motor	Custom List	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25 random: <input type="checkbox"/> % <input checked="" type="checkbox"/> fx ×
id_modelo	Custom List	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30 random: <input type="checkbox"/> % <input checked="" type="checkbox"/> fx ×
numero_tarjeta_circulacion	Digit Sequence	##### blank: <input type="checkbox"/> % <input checked="" type="checkbox"/> fx
numero_serie	Digit Sequence	3XXXXXXXXX blank: <input type="checkbox"/> % <input checked="" type="checkbox"/> fx
numero_placa	Digit Sequence	^##### blank: <input type="checkbox"/> % <input checked="" type="checkbox"/> fx
capacidad_tanque	Number	min: <input type="text" value="30"/> max: <input type="text" value="80"/> decimals: <input type="text"/> blank: <input type="checkbox"/> % <input checked="" type="checkbox"/> fx ×
num_pasajeros	Custom List	2,4,5,7,8 random: <input type="checkbox"/> % <input checked="" type="checkbox"/> fx ×
fecha_caducidad_tarjeta_circulacion	Datetime	01/01/2000 to 09/13/2020 in yyyy/mm/dd blank: <input type="checkbox"/> % <input checked="" type="checkbox"/> fx ×
fecha_emplacado	Datetime	01/01/2000 to 09/13/2020 in yyyy/mm/dd blank: <input type="checkbox"/> % <input checked="" type="checkbox"/> fx ×

### Persona:

Se generaron 1000 personas distintas usando los conjuntos de datos para nombres y apellidos de la herramienta, en particular se insertaron personas con un solo nombre pero en la aplicación no se pondrá dicha restricción.

Field Name	Type	Options
id	Row Number	blank: 0 % fx ×
apellido_paterno	Last Name	blank: 0 % fx ×
apellido_materno	Last Name	blank: 0 % fx ×
nombre	First Name	blank: 0 % fx ×
<button>Add another field</button>		

### Agente\_transito:

Tomamos que las primeras 100 personas que almacenamos en la tabla persona y las hicimos agentes de tránsito, el número de agente lo asignamos del 1 al 100.

Field Name	Type	Options
numero_agente	Row Number	blank: 0 % fx ×
id_persona	Row Number	blank: 0 % fx ×
<button>Add another field</button>		

---

# Rows:  Format:  Table Name:  ☐ include create table

### Propietario:

Se escogió que las 950 últimas personas en la tabla fueran propietarias de vehículos, asignándoles direcciones aleatorias y fechas de nacimiento entre 1940 y 2002 para que dichas personas tuvieran la mayoría de edad y pudieran tener un vehículo a su nombre, por último para la columna rfc a pesar de venir indicada, simplemente se trata de un placeholder ya que el rfc real se calculará con una función de php descrita al final terminar la sección de consultas para el reporte y las funciones y triggers.

Field Name	Type	Options
id	Row Number	blank: 0 % fx ×
rfc	Character Sequence	^^^^#####^  blank: 0 % fx ×
id_direccion	Custom List	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20 random blank: 0 % fx ×
id_persona	Custom List	50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65, sequential blank: 0 % fx ×
fecha_nacimiento	Datetime	1940/01/01 to 9/14/2020 in yyyy/mm/dd blank: 0 % fx ×
Add another field		

---

# Rows: 950 Format: SQL Table Name: propietario ☐ include create table

### Licencia:

Se generó una licencia para cada propietario, tomando como tipo de licencia la de conducción normal, es decir de tipo A, y para el número, dado el tipo de licencia que generamos se utilizó la letra A seguida de 7 dígitos aleatorios, por último se generó la caducidad tomando una fecha aleatoria entre el 14 de septiembre del 2020 y la misma fecha del 2028 para generar vigencias no mayores a 8 años.

Field Name	Type	Options
id	Row Number	blank: 0 % fx ×
numero	Character Sequence	A#####  blank: 0 % fx ×
id_propietario	Custom List	50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65, sequential blank: 0 % fx ×
fecha_caducidad	Datetime	9/14/2020 to 9/14/2028 in yyyy/mm/dd blank: 0 % fx ×
tipo	Custom List	A sequential blank: 0 % fx ×
Add another field		

---

# Rows: 950 Format: SQL Table Name: licencia ☐ include create table

### Tener\_propietario\_vehiculo:

Para esta tabla asignamos a todos los vehículos un propietario de manera aleatoria entre los 950 que tenemos, dejando la fecha de inicio como a el día 14 de septiembre de 2020 para asegurar que no tuviéramos propietarios con un vehículo asignado antes de que tuvieran la mayoría de edad, y dejando la fecha final vacía

para que todos los campos que generamos se mantuvieran como el propietario actual del vehículo.

Field Name	Type	Options
<input type="text" value="id"/>	Row Number	blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×
<input type="text" value="id_vehiculo"/>	Custom List	<input type="text" value="1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20"/> sequential blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×
<input type="text" value="id_propietario"/>	Custom List	<input type="text" value="1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20"/> sequential blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×
<input type="text" value="fecha_inicio"/>	Datetime	<input type="text" value="09/14/2019"/> to <input type="text" value="09/14/2020"/> in <input type="text" value="m/d/yyyy"/> blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×

Add another field

# Rows:  Format:  Table Name:  ☐ include create table

### Artículo:

Se generaron 20 filas siendo estas algunos de los 20 artículos del reglamento de tránsito más comunes de infringir en un lenguaje coloquial.

Field Name	Type	Options
<input type="text" value="numero"/>	Row Number	blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×
<input type="text" value="descripcion"/>	Custom List	<input type="text" value="velocidad superior a 80 km/h, daños a la nacion"/> random blank: <input type="text" value="0"/> % <input type="text" value="fx"/> ×

Add another field

# Rows:  Format:  Table Name:  ☐ include create table

### Infraccion:

Se generaron 1329 multas tomando de manera aleatoria todos los parámetros de la misma, pero asegurando que los agentes, infractores y artículos ya se encontraran en la tabla(es decir tomando solo los id o números que sabíamos existían). tomando al igual que en la dirección las calles con el dataset que provee la herramienta y tomando como municipio alguna de las 16 alcaldías de la CDMX, y tomando como que todas las multas ocurrieron el día 14 de septiembre de 2020 y su homólogo del año 2019 para evitar errores con las condiciones de mayoría de edad de los propietarios.(Es posible dado como construimos nuestros datos aleatorios que se generen multas que tengan fecha anterior a la fecha en que se registró al propietario con el vehículo pero dichos errores serán eliminados para la aplicación con uno de los triggers que explicaremos más adelante).Para el llenado del vehículo en la tabla se utilizó una función de php que escoge aleatoriamente entre los vehículos del propietario, ésta función se mostrará más adelante.(Obs. al principio se generan 2000 datos pero al terminar la ejecución de la asignación de vehículos quedan 1329)

Field Name	Type	Options
numero_expediente	Row Number	blank: 0 % <i>fx</i> ×
numero_articulo	Custom List	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20 random blank: 0 % <i>fx</i> ×
id_infractor	Custom List	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20 random blank: 0 % <i>fx</i> ×
numero_agente	Custom List	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20 random blank: 0 % <i>fx</i> ×
importe	Number	min: 1200 max: 100000 decimals: 2 blank: 0 % <i>fx</i> ×
fecha	Datetime	09/14/2020 to 09/14/2020 in yyyy/mm/dd blank: 0 % <i>fx</i> ×
calle	Street Name	blank: 0 % <i>fx</i> ×
calle_anterior	Street Name	blank: 0 % <i>fx</i> ×
calle_siguiente	Street Name	blank: 0 % <i>fx</i> ×
municipio	Custom List	Álvaro Obregón, Azcapotzalco, Benito Juárez, Co random blank: 0 % <i>fx</i> ×
Add another field		

Anexo: Funciones creadas para la población de la BD a continuación. Para correrlas simplemente hace falta descomentar dichas funciones y sus botones en la vista de multas, al hacer click en el botón se correrá la función y automáticamente se poblará la base de datos.

### Función php para calcular el RFC

```

public function rfc(Request $request){
    $propietarios= Propietario::with(['idPersona'])->get();
    foreach($propietarios as $propietario){
        $apellidoPaterno=strtoupper(substr($propietario->idPersona->apellido_paterno,0,2));
        $apellidoMaterno=strtoupper(substr($propietario->idPersona->apellido_materno,0,1));
        $nombre=strtoupper(substr($propietario->idPersona->nombre,0,1));
        $fecha=explode('-', $propietario->fecha_nacimiento);
        $homoclave_letra1= strtoupper(chr(rand(97,122)));
        $homoclave_letra2= strtoupper(chr(rand(97,122)));
        $homoclave_numero=rand(0,9);
        $rfc=$apellidoPaterno.$apellidoMaterno.$nombre.substr($fecha[0],2,2).$fecha[1].
        .explode(' ', $fecha[2])[0].$homoclave_letra1.$homoclave_letra2.$homoclave_numero;
        Propietario::updateOrCreate(
            ['id'=>$propietario->id],
            ['rfc'=>$rfc]
        );
    }
}

```

### Función php para asignar tipo de transmisión.

```

public function tipoTransmision(Request $request){
    $vehiculos=Vehiculo::all();
    $transmission=["automatico", "estandar"];
    foreach($vehiculos as $vehiculo){
        $aux=$transmission[rand(0,1)];
        Vehiculo::updateOrCreate(
            ['id'=>$vehiculo->id],
            ['tipo_transmision'=>$aux]
        );
    }
    return;
}

```

### Función php para asignar a la infracción el vehículo correspondiente adecuado

```

public function asignarVehiculo(Request $request){
    $infracciones=Infraccion::with(['idInfractor'])->get();
    foreach($infracciones as $infraccion){
        $tieneVehiculos=TenerPropietarioVehiculo::where('id_propietario',$infraccion->id_infractor)->first();
        //tamaño buscado del aleatorio
        if(!is_null($tieneVehiculos)){
            Infraccion::updateOrCreate(
                ['id_infractor'=>$tieneVehiculos->id_propietario],
                ['id_vehiculo'=>$tieneVehiculos->id_vehiculo]
            );
        }else{
            Infraccion::where('numero_expediente',$infraccion->numero_expediente)->delete();
        }
    }
}

```

## 5. Consultas:

Las 15 consultas que se crearon como reporte se encuentran en la carpeta SQL/Consultas y cada una tiene asociada un comentario en donde se describen los datos que resultan de la misma.

## 6. Funciones y triggers:

Estas se encuentran en la carpeta SQL del proyecto y ambas se encuentran documentadas, con su funcionamiento. Para su correcto funcionamiento no hace falta más que tener construida la base y ejecutar tanto las funciones como los triggers para que estos se almacenen en la base.

## 7. Aplicación:

La aplicación está construida en el lenguaje PHP, con el uso del framework **Laravel** además de ciertas herramientas utilizadas como **Eloquent** para el manejo de los modelos, **infyom** como creador de los modelos y **Bootstrap 4** como plantilla de estilos.

Para correr la aplicación hace falta tener instalado en la computadora las herramientas **Composer** y **Node.js**, además de tener instalado también **PHP** para poder mostrar la aplicación, la instalación de dichas herramientas se encuentra en varios tutoriales en línea además de en las páginas oficiales de las herramientas y dado que dependiendo de la distribución del sistema operativo de cada computadora se recomienda visitar dichas páginas para obtener la instalación correcta de cada una de ellas. A continuación se dejará un enlace a cada una de las páginas. una vez estas se tengan instaladas descargar e instalar **Laravel**.

Link PHP: <https://www.php.net/downloads.php>

Link Composer: <https://getcomposer.org/download/>

Link Node.js: <https://nodejs.org/es/download/>



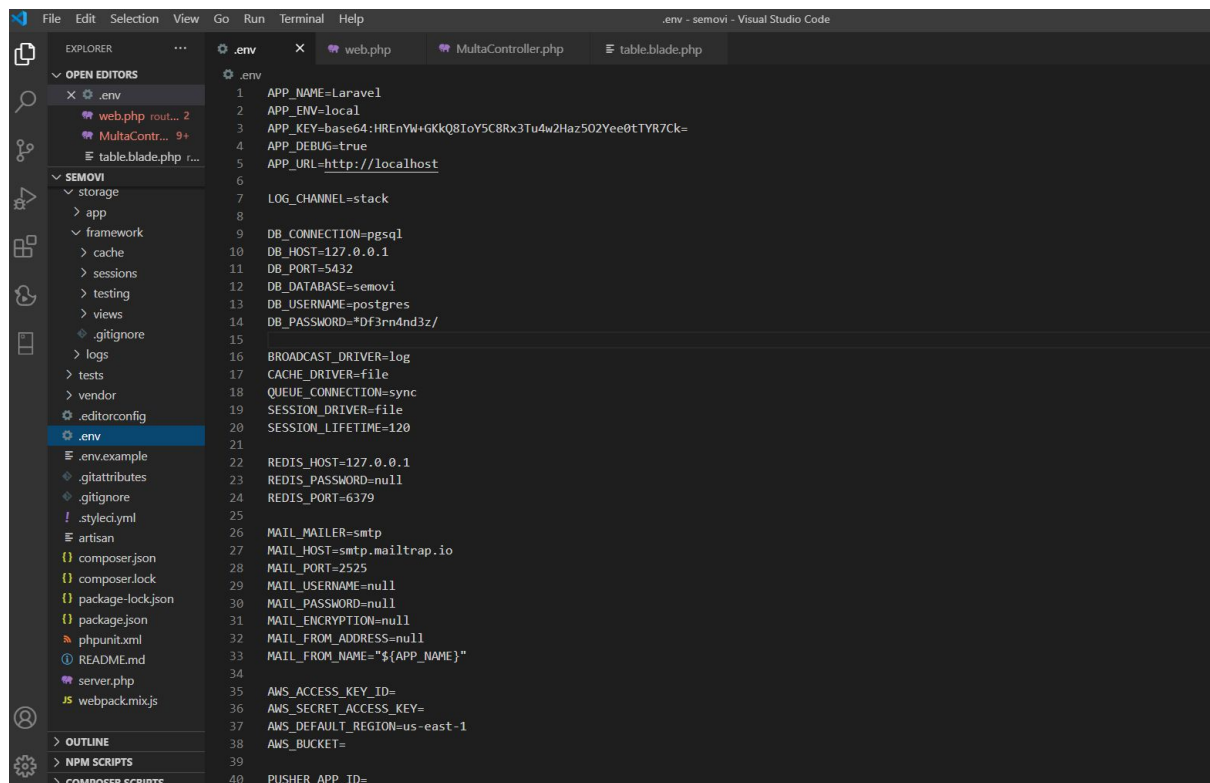
Link instalación Laravel: <https://laravel.com/docs/4.2/installation>

Una vez instaladas todas estas herramientas pararse en la carpeta aplicación y correr los siguientes comandos:

1. composer install
2. composer update
3. npm install
4. npm dev run

Con estos comandos se instalarán todas las dependencias del proyecto y para compilar y montar el proyecto hace falta definir algunos parámetros para el correcto funcionamiento del proyecto.

En el archivo .env modificar los datos que se muestran a continuación con los datos de la base de datos de postgres creada con las especificaciones del proyecto anteriormente vistas.(El puerto de postgres, el nombre de la base y el usuario y contraseña de acceso a la misma)



```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:HREnYW+GKkQ8IoY5C8Rx3Tu4w2Haz502Yee0tTYR7Ck=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=pgsql
10 DB_HOST=127.0.0.1
11 DB_PORT=5432
12 DB_DATABASE=semovi
13 DB_USERNAME=postgres
14 DB_PASSWORD=#Df3rn4nd3z/
15
16 BROADCAST_DRIVER=log
17 CACHE_DRIVER=file
18 QUEUE_CONNECTION=sync
19 SESSION_DRIVER=file
20 SESSION_LIFETIME=120
21
22 REDIS_HOST=127.0.0.1
23 REDIS_PASSWORD=null
24 REDIS_PORT=6379
25
26 MAIL_MAILER=smtp
27 MAIL_HOST=smtp.mailtrap.io
28 MAIL_PORT=2525
29 MAIL_USERNAME=null
30 MAIL_PASSWORD=null
31 MAIL_ENCRYPTION=null
32 MAIL_FROM_ADDRESS=null
33 MAIL_FROM_NAME="${APP_NAME}"
34
35 AWS_ACCESS_KEY_ID=
36 AWS_SECRET_ACCESS_KEY=
37 AWS_DEFAULT_REGION=us-east-1
38 AWS_BUCKET=
39
40 PUSHER_APP_ID=
```

Posteriormente fijarse en el archivo database.php guardado en la carpeta config y modificar la configuración por default a 'pgsql' y completar los datos de acceso a la base que se muestran a continuación en la sección 'pgsql'(Es decir el esquema donde se almacena la base).



```

54 'unix_socket' => env('DB_SOCKET', ''),
55 'charset' => 'utf8mb4',
56 'collation' => 'utf8mb4_unicode_ci',
57 'prefix' => '',
58 'prefix_indexes' => true,
59 'strict' => true,
60 'engine' => null,
61 'options' => extension_loaded('pdo_mysql') ? array_filter([
62     PDO::MYSQL_ATTR_SSL_CA => env('MYSQL_ATTR_SSL_CA'),
63 ]) : [],
64 ],
65
66 'pgsql' => [
67     'driver' => 'pgsql',
68     'url' => env('DATABASE_URL'),
69     'host' => env('DB_HOST', '127.0.0.1'),
70     'port' => env('DB_PORT', '5432'),
71     'database' => env('DB_DATABASE', 'forge'),
72     'username' => env('DB_USERNAME', 'forge'),
73     'password' => env('DB_PASSWORD', ''),
74     'charset' => 'utf8',
75     'prefix' => '',
76     'prefix_indexes' => true,
77     'schema' => 'public',
78     'sslmode' => 'prefer',
79 ],
80
81 'sqlsrv' => [
82     'driver' => 'sqlsrv',
83     'url' => env('DATABASE_URL'),
84     'host' => env('DB_HOST', 'localhost'),
85     'port' => env('DB_PORT', '1433'),
86     'database' => env('DB_DATABASE', 'forge'),
87     'username' => env('DB_USERNAME', 'forge'),
88     'password' => env('DB_PASSWORD', ''),
89     'charset' => 'utf8',
90     'prefix' => '',
91     'prefix_indexes' => true,
92 ],
93

```

Posteriormente hace falta configurar el sistema de login por lo tanto corremos el comando

5. php artisan migrate:install

y

6. php artisan migrate

Por último para montar la página web en el servidor local correr:

7. php artisan serve [--port (el puerto donde se ejecutará el proyecto)]

Después en la terminal aparecerá el link que se debe de copiar y pegar en el navegador para ver el sistema en línea

Una vez se encuentre adentro del sistema se debe crear un perfil para acceder, esto se hace dando click en la barra de navegación en el tag “Register” y por último crear un usuario y una contraseña, hacer el login con los datos ingresados y en ese punto todas las funcionalidades de la aplicación se encontrarán en la barra de navegación del sitio.