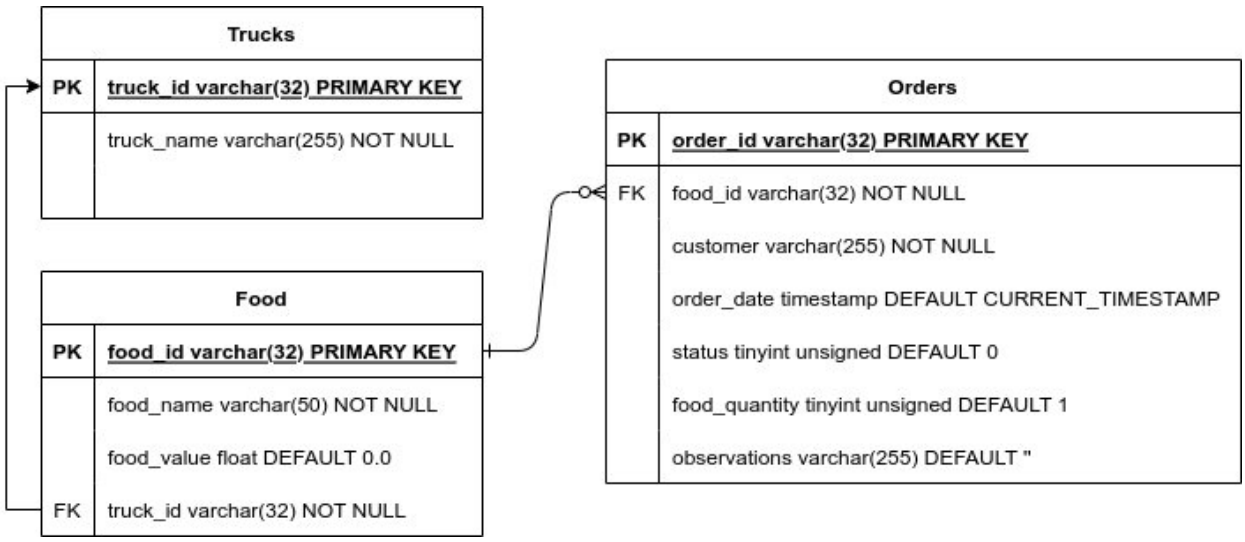
Manual del desarrollo de The Order.

Juan Esteban García Cardona - 1128439829

Estructura de la base de datos:

Para la estructura de la base de datos me decido por un motor de bases de datos relacional con MariaDB en su versión *Ver 15.1 Distrib 10.6.12-MariaDB*, para Debian Linux (mi sistema operativo de uso diario; donde su esquema estructural será la mostrada en el siguiente diagrama entidad – relación.



En la tabla de comidas "*foods*" se almacerá la información relacionada con el plato de comida; cada uno de los platos de comida viene asociado con un restaurante o "*truck*" que cuenta a su vez con un nombre modificable. Mientras que en la entidad ordenes se guardará toda la información relacionada con la toma del pedido, allí se guardará el id del plato de comida (que será un string de 32 posiciones generado por una función dentro del motor de base de datos); el nombre del consumidor, la fecha exacta en que la orden fue tomada, el estado de dicha orden – cuyo valor inicial será 0 que será el correspondiente a orden tomada -, la cantidad de esa comida, las observaciones sobre ese plato de comida y como última opción *where_eat* donde si el pedido es para llevar.
En cuanto al código para crear la base de datos sería el siguiente.

```
CREATE DATABASE IF NOT EXISTS TheOrder;
USE TheOrder;

DELIMITER //
CREATE FUNCTION generate_id()
RETURNS VARCHAR(32)
BEGIN
  DECLARE id VARCHAR(32);
  SELECT REPLACE(UUID(), '-', '') INTO id;
  RETURN id;
END; //
```

```
DELIMITER ;
CREATE TABLE IF NOT EXISTS `trucks` (
  `truck_id` VARCHAR(32) PRIMARY KEY,
  `truck_name` VARCHAR(255) NOT NULL
);

CREATE TABLE IF NOT EXISTS `foods` (
  `food_id` VARCHAR(32) PRIMARY KEY,
  `food_name` VARCHAR(255) NOT NULL,
  `food_value` FLOAT DEFAULT 0.0,
  `truck_id` VARCHAR(32) NOT NULL,
  CONSTRAINT `foods_trucks_truck_id_foreign` FOREIGN KEY (`truck_id`)
REFERENCES `trucks` (`truck_id`)
);

CREATE TABLE IF NOT EXISTS `orders` (
  `order_id` VARCHAR(32) PRIMARY KEY,
  `food_id` VARCHAR(32) NOT NULL,
  `customer` VARCHAR(255) NOT NULL,
  `order_date` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `status` TINYINT UNSIGNED DEFAULT 0,
  `where_eat` BOOLEAN DEFAULT FALSE,
  `food_quantity` TINYINT UNSIGNED DEFAULT 1,
  `observations` VARCHAR(255) DEFAULT '',
  CONSTRAINT `foods_orders_food_id_foreign` FOREIGN KEY (`food_id`)
REFERENCES `foods` (`food_id`)
);
```

Y para insertar una lista de platos de comida:

```
-- To create new trucks
INSERT INTO TheOrder.`trucks` (`truck_id`, `truck_name`)
SELECT generate_id(), 'La hamburgueseria' UNION ALL
SELECT generate_id(), 'Super Papas';

-- To create new foods
INSERT INTO TheOrder.`foods` (`food_id`, `food_name`, `food_value`,
`truck_id`)
SELECT generate_id(), 'Hamburguesa', 5, truck_id FROM `trucks` WHERE
`truck_name` = 'La hamburgueseria' UNION ALL
SELECT generate_id(), 'Salchipapas', 3, truck_id FROM `trucks` WHERE
`truck_name` = 'Super Papas' UNION ALL
SELECT generate_id(), 'Salchipapas Grande', 4.5, truck_id FROM `trucks`
WHERE `truck_name` = 'Super Papas' UNION ALL
SELECT generate_id(), 'Gaseosa', 1, truck_id FROM `trucks` WHERE
`truck_name` = 'La hamburgueseria' UNION ALL
SELECT generate_id(), 'Tacos', 10, truck_id FROM `trucks` WHERE
`truck_name` = 'La hamburgueseria';
```

La función *generate_id*, previamente creada asegurará que cada uno de los platos de comida queden con un ID de tipo *string* de 32 caracteres.

Backend.

El backend está programado en Python 3.9 con el framework de Flask y está compuesto por un archivo de inicialización llamado "*start.py*" donde se cargan las librerías necesarias y se estructura la base del algoritmo para el framework. Las librerías necesarias para la ejecución del código del backend se encuentran en el archivo de "*requirements.txt*" para su fácil instalación. También está presente en el proyecto una carpeta para los archivos de datos y otra para los módulos llamada "*src*".

Dentro de la carpeta source se encuentran los folders de los helpers, que son funciones que pueden servir de ayuda y finalmente una carpeta con los controladores de cada módulo, aquí estarán los archivos de control de cada módulo. Cada uno de los módulos está desarrollado por medio de programación funcional así como el contenido de los helpers.

El primer módulo encontrado en la carpeta "*controllers*" es trucks, que cuenta con las siguientes rutas:

- Obtener trucks: Que lista los restaurantes de la base de datos en /trucks.
- Crear truck: Que se realiza por medio de una pertición POST sobre la url base de trucks.
- Actualizar truck: Por medio de PUT y enviando como parámetro el id del truck a modificar.
- Borrar truck: Que en este caso borra el registro de la base de datos, por medio de una petición DELETE y enviando como parámetro el id del truck a eliminar.

El siguiente controlador cuenta con las mismas rutas esta vez sobre el endpoint de /foods; aquí se agregó una nueva ruta para poder obtener los platos de comida por restaurante

- /foods/<truck_id>: Enviando como parámetro el id del truck del que se desean los platos.

La respuesta del backend se entrega en un JSON con los campos de código de la respuesta, que si es exitosa será un 000, un mensaje y un campo de data que contiene la información solicitada. Cada error está numerado y tiene asignado un código diferente.

Para el caso de las ordenes, delimité a una cada una por cuestiones de tiempo; una orden solo puede tener un plato de comida, para crear una orden bastará con hacer una petición de tipo POST consumiendo el endopoint /orders con los siguientes campos:

- food_id, que es obligatorio.
- customer; que también es obligatorio.
- food_quantity, que es opcional, si no se envía se pondrá un 1 por defecto.
- observations: Para registrar las observaciones sobre el plato de comida.
- where_eat: Para controlar donde se comerá el plato.

Para listar las ordenes, hacer un get sobre /orders o sobre /orders/<truck_id> para filtrar por restaurante. Al igual que las rutas anteriormente mencionadas para actualizar una orden consume con el método PUT y para borrarla con el método DELETE enviando como parámetro el id de la orden.

Frontend.

El frontend de la aplicación decidí montarla en el framework VueJS 3 ya que es uno que me gusta por su facilidad, velocidad y versatilidad.

Los archivos del frontend están distribuidos en el directorio *src* que es la carpeta principal del código donde se encuentran 4 recursos principales, los assets, los componentes, las vistas y el router. En el directorio de los assets, se encuentran las fuentes, imagenes y algunas librerías como bootstrap para dar estilos más rápidamente, también se encuentran las configuraciones del frontend.
Configuraciones del frontend:

- Backend: No es más que un objeto de javascript con un atributo llamado url que denota IP del servidor donde está alojado el backend.
- Lenguages: Es un objeto con todas las palabras de la interfaz gráfica en diferentes idiomas. Cada idioma será un objeto que se inserta en el principal.

Components es una carpeta que contiene los diferentes elementos que tiene la interfaz, por ejemplo el topbar o el backendConnection que es un componente invisible con funciones para conectarse con el backend.
El enrutador de VueJS que lo alojé en su propia carpeta, tiene un solo objeto llamado *router,* contiene 7 posibles rutas, *home, about*, *trucks*, *foods* y *orders* (que serían las vistas administrativas donde se pueden modificar los datos por medio de una tabla con diferentes funcionalidades); otras 2 rutas serán destinadas para los elementos del sistema como lo son la pantalla cuya ruta será /screen y otra para la tablet con URL /tablet. Por último, las vistas que son los archivos que construyen cada una de las páginas mencionadas se alojan en la carpeta views.

Todas vistas están conformadas por un componente que implementa la visualización de la tabla, que a su vez tiene todo el código necesario para modificarse según el caso, pero internamente siempre funciona igual:

- Dentro de una vista de un módulo (por ejemplo FoodsView) va un componente llamado ModuleView que contiene un componente tabla.
- La tabla se secciona en 3 partes, el header que muestra las etiquetas de la columna; el contenido, que es un componente diferente que controla como se muestran los datos y la tabla puede tener una tabla interna (que por cuestiones de tiempo no me da para implementarla completamente)
- También implementé la función de importe (esta no debería funcionar porque requiere que el backend tenga una ruta para recibir archivos y procesarlos) y exporte de datos, así como la de filtrado y ordenación sobre los campos de la tabla.
- Al modificar un campo en la tabla este se podrá guardar.
- Al eliminar un registro se solicitará una razón de la eliminación que en este caso no se usará.
- Cada registro de la tabla se puede duplicar con el botón respectivo, esto copiará el registro sobre la primera fila de la tabla.

Y como caracteristicas del desarrollo del frontend destacaría:

- Todo el sistema está programado bajo la premisa de componentes y reactividad, por lo tanto un cambio se va a propagar y va a poder navegar entre los diferentes módulos.
- También dejé una función para determinar la permisología de un usuario que siempre va a estar respondiendo que el permiso existe.
- Intenté mantener un estilo agradable a la vista durante el proceso con ayuda de bootstrap.
- Se programó posibilidad del cambio de lenguaje de la interfaz pero solo es modificable por código por medio del lenguages de la carpeta assets.

Modo de uso

El consumidor con ayuda de la tablet visitará la url /screen del frontend que contiene una vista de todos los platos de comida disponibles.
Al dar click en alguno de los platos se desplegará una mensaje para digitar la cantidad de unidades, otro para digitar su nombre y uno final para digitar 0 si el pedido no es para llevar y 1 si es para llevar.
Una vez finalizada esta acción se creará una orden.
El restaurante podrá consultar las ordenes en /orders. Alli podrá modificar la orden o solo su estado siendo la opción 0 recibido, 1 en preparación y 2 listo.
En el televisor se consultará la URL /screen, que mostrará las ordenes sobre una tabla con los campos requieridos de cada orden.