MAYO DE 2025

DOCUMENTACIÓN DE LA API DE TRUCK BITES

ANDRÉS CASO IGLESIAS

Contenidos

Introducción

La API de Truck Bites proporciona una interfaz RESTful para gestionar food trucks, menús, pedidos, usuarios, ubicaciones y notificaciones en la plataforma Truck Bites. Este documento detalla cada endpoint, incluyendo su funcionalidad, formatos de solicitud y respuesta, parámetros y ejemplos de uso. La API está alojada en http://localhost:8080/api y utiliza JSON para el intercambio de datos.

Autenticación

Actualmente, la API de Truck Bites no requiere autenticación para la mayoría de los endpoints. Las versiones futuras podrían implementar autenticación basada en JWT para un acceso seguro, especialmente para endpoints administrativos.

Endpoints

La API está organizada en torno a los siguientes recursos: Food Trucks, Menús, Pedidos, Usuarios, Ubicaciones y Notificaciones. Cada endpoint se describe con su método HTTP, ruta, parámetros, formatos de solicitud/respuesta y un ejemplo de uso con curl.

Food Trucks

Obtener Todos los Food Trucks

- **Endpoint**: GET /api/foodtrucks
- **Descripción**: Recupera una lista de todos los food trucks en el sistema.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos food truck.
- **Formato de Respuesta**:

```
[
  {
    "id": Long,
    "nombre": String,
    "tipoCocina": String,
    "ubicacionActual": String
  },
  ...
]
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/foodtrucks
```

Respuesta:

```
[
  {
    "id": 1,
    "nombre": "Taco Fiesta",
    "tipoCocina": "Mexicana",
    "ubicacionActual": "Nueva York, 5th Ave"
  },
  ...
]
```

Obtener Food Truck por ID

- **Endpoint**: GET /api/foodtrucks/{id}
- **Descripción**: Recupera los detalles de un food truck específico por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID del food truck (ej., 1).
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, objeto JSON del food truck; 404 Not Found si el ID no existe.
- **Formato de Respuesta**:

```
{
  "id": Long,
  "nombre": String,
  "tipoCocina": String,
  "ubicacionActual": String
}
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/foodtrucks/1
```

Respuesta:

```
{
  "id": 1,
  "nombre": "Taco Fiesta",
  "tipoCocina": "Mexicana",
  "ubicacionActual": "Nueva York, 5th Ave"
}
```

Crear Food Truck

- **Endpoint**: POST /api/foodtrucks
- **Descripción**: Crea un nuevo food truck.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Objeto JSON con los detalles del food truck.
- **Formato de Solicitud**:

```
{
  "nombre": String,
  "tipoCocina": String,
  "ubicacionActual": String
}
```

- **Respuesta**: 200 OK, objeto JSON del food truck creado; 400 Bad Request si la validación falla.
- **Formato de Respuesta**: Igual que Obtener Food Truck por ID.
- **Ejemplo**:

```
curl -X POST http://localhost:8080/api/foodtrucks \
-H "Content-Type: application/json" \
-d '{"nombre":"New Truck","tipoCocina":"Fusion","ubicacionActual":"Nueva
York, 10th St"}'
```

Respuesta:

```
{
  "id": 7,
  "nombre": "New Truck",
  "tipoCocina": "Fusion",
  "ubicacionActual": "Nueva York, 10th St"
}
```

Actualizar Food Truck

- **Endpoint**: PUT /api/foodtrucks/{id}
- **Descripción**: Actualiza los detalles de un food truck existente.
- **Parámetros**:
  - id (ruta, requerido): El ID del food truck.
- **Cuerpo de la Solicitud**: Objeto JSON con los campos a actualizar (al menos uno requerido).
- **Formato de Solicitud**:

```
{
  "nombre": String (opcional),
  "tipoCocina": String (opcional),
  "ubicacionActual": String (opcional)
}
```

- **Respuesta**: 200 OK, objeto JSON del food truck actualizado; 404 Not Found si el ID no existe; 400 Bad Request si no se proporcionan campos.
- **Formato de Respuesta**: Igual que Obtener Food Truck por ID.

Ejemplo:

```
curl -X PUT http://localhost:8080/api/foodtrucks/1 \
-H "Content-Type: application/json" \
-d '{"nombre":"Taco Fiesta Updated","ubicacionActual":"Nueva York,
Broadway"}'
```

Respuesta:

```
{
  "id": 1,
  "nombre": "Taco Fiesta Updated",
  "tipoCocina": "Mexicana",
  "ubicacionActual": "Nueva York, Broadway"
}
```

Eliminar Food Truck

- **Endpoint**: DELETE /api/foodtrucks/{id}
- **Descripción**: Elimina un food truck por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID del food truck.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, sin contenido; 404 Not Found si el ID no existe.

Ejemplo:

```
curl -X DELETE http://localhost:8080/api/foodtrucks/7
```

Respuesta: (Cuerpo vacío)

Obtener Food Trucks Cercanos

- **Endpoint**: GET /api/foodtrucks/cerca
- **Descripción**: Recupera food trucks cercanos a una ciudad y, opcionalmente, una calle.
- **Parámetros**:
  - ciudad (query, requerido): La ciudad (ej., "Nueva York").
  - calle (query, opcional): La calle (ej., "5th Ave").
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos food truck.
- **Formato de Respuesta**: Igual que Obtener Todos los Food Trucks.

Ejemplo:

```
curl -X GET
"http://localhost:8080/api/foodtrucks/cerca?ciudad=Nueva%20York&calle=5th%2
0Ave"
```

Respuesta:

```
[
  {
    "id": 1,
    "nombre": "Taco Fiesta",
    "tipoCocina": "Mexicana",
    "ubicacionActual": "Nueva York, 5th Ave"
  },
  ...
]
```

Recomendar Food Trucks

- **Endpoint**: GET /api/foodtrucks/recomendar
- **Descripción**: Recomienda food trucks basados en la ciudad y el tipo de cocina.
- **Parámetros**:
  - ciudad (query, requerido): La ciudad (ej., "Nueva York").
  - tipoCocina (query, requerido): El tipo de cocina (ej., "Mexicana").

- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos food truck.
- **Formato de Respuesta**: Igual que Obtener Todos los Food Trucks.

Ejemplo:

```
curl -X GET
"http://localhost:8080/api/foodtrucks/recomendar?ciudad=Nueva%20York&tipoCo
cina=Mexicana"
```

Respuesta:

```
[
  {
    "id": 1,
    "nombre": "Taco Fiesta",
    "tipoCocina": "Mexicana",
    "ubicacionActual": "Nueva York, 5th Ave"
  },
  ...
]
```

Obtener Top Food Trucks por Pedidos

- **Endpoint**: GET /api/foodtrucks/top-by-orders
- **Descripción**: Recupera los food trucks más populares según el número de pedidos completados.
- **Parámetros**:
  - limit (query, opcional): Número máximo de food trucks a devolver (predeterminado: 3).
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos food truck con conteo de pedidos.
- **Formato de Respuesta**:

```
[
  {
    "id": Long,
    "nombre": String,
    "tipoCocina": String,
    "ubicacionActual": String,
    "orderCount": Long
  },
  ...
]
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/foodtrucks/top-by-orders?limit=3
```

Respuesta:

```
[
  {
    "id": 1,
    "nombre": "Taco Fiesta",
    "tipoCocina": "Mexicana",
    "ubicacionActual": "Nueva York, 5th Ave",
    "orderCount": 5
  },
  ...
]
```

Obtener Beneficio Medio por Pedido

- **Endpoint**: GET /api/foodtrucks/{id}/average-profit
- **Descripción**: Recupera el beneficio medio por pedido para un food truck específico.
- **Parámetros**:
  - id (ruta, requerido): El ID del food truck.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, número JSON que representa el beneficio medio; 404 Not Found si el ID no existe.
- **Formato de Respuesta**: Double

Ejemplo:

```
curl -X GET http://localhost:8080/api/foodtrucks/1/average-profit
```

Respuesta:

```
5.30
```

Obtener Menús Más Consumidos

- **Endpoint**: GET /api/foodtrucks/most-consumed-menus
- **Descripción**: Recupera el menú más consumido y el cliente principal para cada food truck basado en pedidos completados.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos con detalles del food truck, menú principal y cliente principal.
- **Formato de Respuesta**:

```
[
  {
    "foodTruckId": Long,
    "foodTruckNombre": String,
    "menuItem": String,
    "itemCount": Long,
    "customerName": String,
    "customerOrderCount": Long
  },
  ...
]
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/foodtrucks/most-consumed-menus
```

Respuesta:

```
[
  {
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta",
    "menuItem": "Tacos al Pastor",
    "itemCount": 4,
    "customerName": "Juan Perez",
    "customerOrderCount": 4
  },
  ...
]
```

Menús

Obtener Todos los Menús

- **Endpoint**: GET /api/menus
- **Descripción**: Recupera una lista de todos los menús en el sistema.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos menú.
- **Formato de Respuesta**:

```
[
  {
    "id": Long,
    "nombre": String,
    "descripcion": String,
    "precio": Double,
    "foodTruckId": Long,
    "foodTruckNombre": String
  },
  ...
]
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/menus
```

Respuesta:

```
[
  {
    "id": 1,
    "nombre": "Tacos al Pastor",
    "descripcion": "Tacos con carne de cerdo y piña",
    "precio": 3.5,
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta"
  },
  ...
]
```

Obtener Menú por ID

- **Endpoint**: GET /api/menus/{id}
- **Descripción**: Recupera los detalles de un menú específico por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID del menú (ej., 1).
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, objeto JSON del menú; 404 Not Found si el ID no existe.
- **Formato de Respuesta**:

```
{
  "id": Long,
  "nombre": String,
  "descripcion": String,
  "precio": Double,
  "foodTruckId": Long,
  "foodTruckNombre": String
}
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/menus/1
```

Respuesta:

```
{
  "id": 1,
  "nombre": "Tacos al Pastor",
  "descripcion": "Tacos con carne de cerdo y piña",
  "precio": 3.5,
  "foodTruckId": 1,
  "foodTruckNombre": "Taco Fiesta"
}
```

Crear Menú

- **Endpoint**: POST /api/menus
- **Descripción**: Crea un nuevo menú para un food truck.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Objeto JSON con los detalles del menú.
- **Formato de Solicitud**:

```
{
  "foodTruckId": Long,
  "nombre": String,
  "descripcion": String (opcional),
  "precio": Double
}
```

- **Respuesta**: 200 OK, objeto JSON del menú creado; 400 Bad Request si la validación falla.
- **Formato de Respuesta**: Igual que Obtener Todos los Menús.
- **Ejemplo**:

```
curl -X POST http://localhost:8080/api/menus \
```

```
-H "Content-Type: application/json" \
-d '{"foodTruckId":1,"nombre":"Burrito","descripcion":"Burrito con
carne","precio":5.0}'
```

Crear Múltiples Menús

- **Endpoint**: POST /api/menus/bulk
- **Descripción**: Crea múltiples menús para un food truck en una sola solicitud.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Arreglo JSON de objetos menú.
- **Formato de Solicitud**:

```
[
  {
    "foodTruckId": Long,
    "nombre": String,
    "descripcion": String (opcional),
    "precio": Double
  },
  ...
]
```

- **Respuesta**: 200 OK, arreglo JSON de objetos menú creados; 400 Bad Request si la validación falla.
- **Formato de Respuesta**: Igual que Obtener Todos los Menús.
- **Ejemplo**:

```
curl -X POST http://localhost:8080/api/menus/bulk \
-H "Content-Type: application/json" \
-d '[{"foodTruckId":1,"nombre":"Torta","descripcion":"Torta
mexicana","precio":4.5},{"foodTruckId":1,"nombre":"Tamales","precio":3.0}]'
```

Respuesta:

```
[
  {
    "id": 19,
    "nombre": "Torta",
    "descripcion": "Torta mexicana",
    "precio": 4.5,
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta"
  },
  {
    "id": 20,
    "nombre": "Tamales",
    "descripcion": null,
    "precio": 3.0,
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta"
  }
]
```

Respuesta:

```
{
  "id": 18,
```

```
  "nombre": "Burrito",
  "descripcion": "Burrito con carne",
  "precio": 5.0,
  "foodTruckId": 1,
  "foodTruckNombre": "Taco Fiesta"
}
```

Actualizar Menú

- **Endpoint**: PUT /api/menus/{id}
- **Descripción**: Actualiza los detalles de un menú existente.
- **Parámetros**:
  - id (ruta, requerido): El ID del menú.
- **Cuerpo de la Solicitud**: Objeto JSON con los campos a actualizar (al menos uno requerido).
- **Formato de Solicitud**:

```
{
  "foodTruckId": Long (opcional),
  "nombre": String (opcional),
  "descripcion": String (opcional),
  "precio": Double (opcional)
}
```

- **Respuesta**: 200 OK, objeto JSON del menú actualizado; 404 Not Found si el ID no existe; 400 Bad Request si no se proporcionan campos.
- **Formato de Respuesta**: Igual que Obtener Todos los Menús.
- **Ejemplo**:

```
curl -X PUT http://localhost:8080/api/menus/1 \
-H "Content-Type: application/json" \
-d '{"nombre":"Tacos al Pastor Premium","precio":4.0}'
```

Respuesta:

```
{
  "id": 1,
  "nombre": "Tacos al Pastor Premium",
  "descripcion": "Tacos con carne de cerdo y piña",
  "precio": 4.0,
  "foodTruckId": 1,
  "foodTruckNombre": "Taco Fiesta"
}
```

Eliminar Menú

- **Endpoint**: DELETE /api/menus/{id}
- **Descripción**: Elimina un menú por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID del menú.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, sin contenido; 404 Not Found si el ID no existe.

Ejemplo:

```
curl -X DELETE http://localhost:8080/api/menus/18
```

Respuesta: (Cuerpo vacío)

Obtener Menús por Food Truck

- **Endpoint**: GET /api/menus/foodtruck/{foodTruckId}
- **Descripción**: Recupera todos los menús de un food truck específico.
- **Parámetros**:
  - foodTruckId (ruta, requerido): El ID del food truck.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos menú; 404 Not Found si el food truck no existe.
- **Formato de Respuesta**: Igual que Obtener Todos los Menús.

Ejemplo:

```
curl -X GET http://localhost:8080/api/menus/foodtruck/1
```

Respuesta:

```
[
  {
    "id": 1,
    "nombre": "Tacos al Pastor",
    "descripcion": "Tacos con carne de cerdo y piña",
    "precio": 3.5,
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta"
  },
  ...
]
```

Pedidos

Obtener Todos los Pedidos

- **Endpoint**: GET /api/pedidos
- **Descripción**: Recupera una lista de todos los pedidos en el sistema.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos pedido.
- **Formato de Respuesta**:

```
[
  {
    "id": Long,
    "estado": String,
    "fechaCreacion": String,
    "fechaProgramada": String (nullable),
    "items": String,
    "montoTotal": Double,
```

```
    "foodTruckId": Long,
    "foodTruckNombre": String,
    "usuarioId": Long
  },
  ...
]
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/pedidos
```

Respuesta:

```
[
  {
    "id": 19,
    "estado": "COMPLETADO",
    "fechaCreacion": "2025-05-14T22:00:00",
    "fechaProgramada": null,
    "items": "Tacos al Pastor",
    "montoTotal": 3.5,
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta",
    "usuarioId": 2
  },
  ...
]
```

Obtener Pedido por ID

- **Endpoint**: GET /api/pedidos/{id}
- **Descripción**: Recupera los detalles de un pedido específico por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID del pedido (ej., 19).
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, objeto JSON del pedido; 404 Not Found si el ID no existe.
- **Formato de Respuesta**:

```
{
  "id": Long,
  "estado": String,
  "fechaCreacion": String,
  "fechaProgramada": String (nullable),
  "items": String,
  "montoTotal": Double,
  "foodTruckId": Long,
  "foodTruckNombre": String,
  "usuarioId": Long
}
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/pedidos/19
```

Respuesta:

```
{
  "id": 19,
  "estado": "COMPLETADO",
  "fechaCreacion": "2025-05-14T22:00:00",
  "fechaProgramada": null,
  "items": "Tacos al Pastor",
  "montoTotal": 3.5,
  "foodTruckId": 1,
  "foodTruckNombre": "Taco Fiesta",
  "usuarioId": 2
}
```

Crear Pedido

- **Endpoint**: POST /api/pedidos
- **Descripción**: Crea un nuevo pedido para un usuario y food truck.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Objeto JSON con los detalles del pedido.
- **Formato de Solicitud**:

```
{
  "usuarioId": Long,
  "foodTruckId": Long,
  "items": String,
  "montoTotal": Double,
  "estado": String,
  "fechaCreacion": String (opcional),
  "fechaProgramada": String (opcional)
}
```

- **Respuesta**: 200 OK, objeto JSON del pedido creado; 400 Bad Request si la validación falla.
- **Formato de Respuesta**: Igual que Obtener Todos los Pedidos.

Ejemplo:

```
curl -X POST http://localhost:8080/api/pedidos \
-H "Content-Type: application/json" \
-d '{"usuarioId":2,"foodTruckId":1,"items":"Tacos al
Pastor,guacamole","montoTotal":13.0,"estado":"PENDIENTE"}'
```

Respuesta:

```
{
  "id": 37,
  "estado": "PENDIENTE",
  "fechaCreacion": "2025-05-14T22:00:00",
  "fechaProgramada": null,
  "items": "Tacos al Pastor,guacamole",
  "montoTotal": 13.0,
  "foodTruckId": 1,
  "foodTruckNombre": "Taco Fiesta",
  "usuarioId": 2
}
```

Actualizar Pedido

- **Endpoint**: PUT /api/pedidos/{id}
- **Descripción**: Actualiza los detalles de un pedido existente.
- **Parámetros**:
  - id (ruta, requerido): El ID del pedido.
- **Cuerpo de la Solicitud**: Objeto JSON con los campos a actualizar (al menos uno requerido).
- **Formato de Solicitud**:

```
{
  "usuarioId": Long (opcional),
  "foodTruckId": Long (opcional),
  "items": String (opcional),
  "montoTotal": Double (opcional),
  "estado": String (opcional),
  "fechaCreacion": String (opcional),
  "fechaProgramada": String (opcional)
}
```

- **Respuesta**: 200 OK, objeto JSON del pedido actualizado; 404 Not Found si el ID no existe; 400 Bad Request si no se proporcionan campos.
- **Formato de Respuesta**: Igual que Obtener Todos los Pedidos.

Ejemplo:

```
curl -X PUT http://localhost:8080/api/pedidos/37 \
-H "Content-Type: application/json" \
-d '{"estado":"COMPLETADO"}'
```

Respuesta:

```
{
  "id": 37,
  "estado": "COMPLETADO",
  "fechaCreacion": "2025-05-14T22:00:00",
  "fechaProgramada": null,
  "items": "Tacos al Pastor,guacamole",
  "montoTotal": 13.0,
  "foodTruckId": 1,
  "foodTruckNombre": "Taco Fiesta",
  "usuarioId": 2
}
```

Eliminar Pedido

- **Endpoint**: DELETE /api/pedidos/{id}
- **Descripción**: Elimina un pedido por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID del pedido.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, sin contenido; 404 Not Found si el ID no existe.
- **Ejemplo**:

```
curl -X DELETE http://localhost:8080/api/pedidos/37
```

Respuesta: (Cuerpo vacío)

Obtener Pedidos Pendientes por Food Truck

- **Endpoint**: GET /api/pedidos/pendientes/{foodTruckId}
- **Descripción**: Recupera todos los pedidos pendientes de un food truck específico.
- **Parámetros**:
  - foodTruckId (ruta, requerido): El ID del food truck.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos pedido; 404 Not Found si el food truck no existe.
- **Formato de Respuesta**: Igual que Obtener Todos los Pedidos.

Ejemplo:

```
curl -X GET http://localhost:8080/api/pedidos/pendientes/1
```

Respuesta:

```
[
  {
    "id": 37,
    "estado": "PENDIENTE",
    "fechaCreacion": "2025-05-14T22:00:00",
    "fechaProgramada": null,
    "items": "Tacos al Pastor,guacamole",
    "montoTotal": 13.0,
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta",
    "usuarioId": 2
  },
  ...
]
```

Obtener Ventas Totales por Food Truck

- **Endpoint**: GET /api/pedidos/ventas/{foodTruckId}
- **Descripción**: Recupera las ventas totales de un food truck en un rango de fechas.
- **Parámetros**:
  - foodTruckId (ruta, requerido): El ID del food truck.
  - startDate (query, requerido): Fecha de inicio (formato ISO, ej., "2025-05-01T00:00:00").
  - endDate (query, requerido): Fecha de fin (formato ISO, ej., "2025-05-31T23:59:59").
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, número JSON que representa las ventas totales; 404 Not Found si el food truck no existe.
- **Formato de Respuesta**: `Double`

Ejemplo:

```
curl -X GET "http://localhost:8080/api/pedidos/ventas/1?startDate=2025-05-01T00:00:00&endDate=2025-05-31T23:59:59"
```

Respuesta:

```
23.50
```

Obtener Pedidos por Usuario y Food Truck

- **Endpoint**: GET /api/pedidos/usuario/{usuarioId}/foodtruck/{foodTruckId}
- **Descripción**: Recupera todos los pedidos de un usuario para un food truck específico.
- **Parámetros**:
  - usuarioId (ruta, requerido): El ID del usuario.
  - foodTruckId (ruta, requerido): El ID del food truck.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos pedido; 404 Not Found si el usuario o food truck no existe.
- **Formato de Respuesta**: Igual que Obtener Todos los Pedidos.

Ejemplo:

```
curl -X GET http://localhost:8080/api/pedidos/usuario/2/foodtruck/1
```

Respuesta:

```
[
  {
    "id": 19,
    "estado": "COMPLETADO",
    "fechaCreacion": "2025-05-14T22:00:00",
    "fechaProgramada": null,
    "items": "Tacos al Pastor",
    "montoTotal": 3.5,
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta",
    "usuarioId": 2
  },
  ...
]
```

Obtener Pedidos por Usuario

- **Endpoint**: GET /api/pedidos/usuario/{usuarioId}
- **Descripción**: Recupera todos los pedidos de un usuario específico.
- **Parámetros**:
  - usuarioId (ruta, requerido): El ID del usuario.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos pedido; 404 Not Found si el usuario no existe.
- **Formato de Respuesta**: Igual que Obtener Todos los Pedidos.

Ejemplo:

```
curl -X GET http://localhost:8080/api/pedidos/usuario/2
```

Respuesta:

```
[
  {
    "id": 19,
    "estado": "COMPLETADO",
    "fechaCreacion": "2025-05-14T22:00:00",
    "fechaProgramada": null,
    "items": "Tacos al Pastor",
    "montoTotal": 3.5,
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta",
    "usuarioId": 2
  },
  ...
]
```

Usuarios

Obtener Todos los Usuarios

- **Endpoint**: GET /api/usuarios
- **Descripción**: Recupera una lista de todos los usuarios en el sistema.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos usuario.
- **Formato de Respuesta**:

```
[
  {
    "id": Long,
    "nombre": String,
    "email": String,
    "password": String,
    "ubicacion": String
  },
  ...
]
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/usuarios
```

Respuesta:

```
[
  {
    "id": 2,
    "nombre": "Juan Perez",
    "email": "juan@example.com",
    "password": "hashed_password",
    "ubicacion": "Nueva York, 5th Ave"
  },
  ...
]
```

Obtener Usuario por ID

- **Endpoint**: GET /api/usuarios/{id}

- **Descripción**: Recupera los detalles de un usuario específico por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID del usuario (ej., 2).
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, objeto JSON del usuario; 404 Not Found si el ID no existe.
- **Formato de Respuesta**:

```
{
  "id": Long,
  "nombre": String,
  "email": String,
  "password": String,
  "ubicacion": String
}
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/usuarios/2
```

Respuesta:

```
{
  "id": 2,
  "nombre": "Juan Perez",
  "email": "juan@example.com",
  "password": "hashed_password",
  "ubicacion": "Nueva York, 5th Ave"
}
```

Crear Usuario

- **Endpoint**: POST /api/usuarios
- **Descripción**: Crea una nueva cuenta de usuario.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Objeto JSON con los detalles del usuario.
- **Formato de Solicitud**:

```
{
  "nombre": String,
  "email": String,
  "password": String,
  "ubicacion": String (opcional)
}
```

- **Respuesta**: 200 OK, objeto JSON del usuario creado; 400 Bad Request si la validación falla (ej., email duplicado).
- **Formato de Respuesta**: Igual que Obtener Usuario por ID.

Ejemplo:

```
curl -X POST http://localhost:8080/api/usuarios \
-H "Content-Type: application/json" \
-d '{"nombre":"Ana
Lopez","email":"ana@example.com","password":"abc123","ubicacion":"Nueva
York, 7th Ave"}'
```

Respuesta:

```
{
  "id": 8,
  "nombre": "Ana Lopez",
  "email": "ana@example.com",
  "password": "abc123",
  "ubicacion": "Nueva York, 7th Ave"
}
```

Actualizar Usuario

- **Endpoint**: PUT /api/usuarios/{id}
- **Descripción**: Actualiza los detalles de un usuario existente.
- **Parámetros**:
  - id (ruta, requerido): El ID del usuario.
- **Cuerpo de la Solicitud**: Objeto JSON con los campos a actualizar (al menos uno requerido).
- **Formato de Solicitud**:

```
{
  "nombre": String (opcional),
  "email": String (opcional),
  "password": String (opcional),
  "ubicacion": String (opcional)
}
```

- **Respuesta**: 200 OK, objeto JSON del usuario actualizado; 404 Not Found si el ID no existe; 400 Bad Request si no se proporcionan campos.
- **Formato de Respuesta**: Igual que Obtener Usuario por ID.

Ejemplo:

```
curl -X PUT http://localhost:8080/api/usuarios/8 \
-H "Content-Type: application/json" \
-d '{"nombre":"Ana Lopez Updated","ubicacion":"Nueva York, 8th Ave"}'
```

Respuesta:

```
{
  "id": 8,
  "nombre": "Ana Lopez Updated",
  "email": "ana@example.com",
  "password": "abc123",
  "ubicacion": "Nueva York, 8th Ave"
}
```

Eliminar Usuario

- **Endpoint**: DELETE /api/usuarios/{id}
- **Descripción**: Elimina un usuario por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID del usuario.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, sin contenido; 404 Not Found si el ID no existe.

Ejemplo:

```
curl -X DELETE http://localhost:8080/api/usuarios/8
```

Respuesta: (Cuerpo vacío)

Obtener Usuario por Email

- **Endpoint**: GET /api/usuarios/email/{email}
- **Descripción**: Recupera los detalles de un usuario por su dirección de correo electrónico.
- **Parámetros**:
  - email (ruta, requerido): El email del usuario (ej., "juan@example.com").
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, objeto JSON del usuario; 404 Not Found si el email no existe.
- **Formato de Respuesta**: Igual que Obtener Usuario por ID.

Ejemplo:

```
curl -X GET http://localhost:8080/api/usuarios/email/juan@example.com
```

Respuesta:

```
{
  "id": 2,
  "nombre": "Juan Perez",
  "email": "juan@example.com",
  "password": "hashed_password",
  "ubicacion": "Nueva York, 5th Ave"
}
```

Ubicaciones

Obtener Todas las Ubicaciones

- **Endpoint**: GET /api/ubicaciones
- **Descripción**: Recupera una lista de todas las ubicaciones en el sistema.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos ubicación.
- **Formato de Respuesta**:

```
[
  {
    "id": Long,
    "foodTruckId": Long,
    "foodTruckNombre": String,
    "ciudad": String,
    "calle": String,
    "fecha": String
  },
  ...
]
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/ubicaciones
```

Respuesta:

```
[
  {
    "id": 1,
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta",
    "ciudad": "Nueva York",
    "calle": "5th Ave",
    "fecha": "2025-05-14T22:00:00"
  },
  ...
]
```

Obtener Ubicación por ID

- **Endpoint**: GET /api/ubicaciones/{id}
- **Descripción**: Recupera los detalles de una ubicación específica por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID de la ubicación (ej., 1).
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, objeto JSON de la ubicación; 404 Not Found si el ID no existe.
- **Formato de Respuesta**:

```
{
  "id": Long,
  "foodTruckId": Long,
  "foodTruckNombre": String,
  "ciudad": String,
  "calle": String,
  "fecha": String
}
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/ubicaciones/1
```

Respuesta:

```
{
  "id": 1,
  "foodTruckId": 1,
  "foodTruckNombre": "Taco Fiesta",
  "ciudad": "Nueva York",
  "calle": "5th Ave",
  "fecha": "2025-05-14T22:00:00"
}
```

Crear Ubicación

- **Endpoint**: POST /api/ubicaciones
- **Descripción**: Crea una nueva ubicación para un food truck.

- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Objeto JSON con los detalles de la ubicación.
- **Formato de Solicitud**:

```
{
  "foodTruckId": Long,
  "ciudad": String,
  "calle": String,
  "fecha": String
}
```

- **Respuesta**: 200 OK, objeto JSON de la ubicación creada; 400 Bad Request si la validación falla.
- **Formato de Respuesta**: Igual que Obtener Ubicación por ID.

Ejemplo:

```
curl -X POST http://localhost:8080/api/ubicaciones \
-H "Content-Type: application/json" \
-d '{"foodTruckId":1,"ciudad":"Nueva York","calle":"10th St","fecha":"2025-
05-14T22:00:00"}'
```

Respuesta:

```
{
  "id": 2,
  "foodTruckId": 1,
  "foodTruckNombre": "Taco Fiesta",
  "ciudad": "Nueva York",
  "calle": "10th St",
  "fecha": "2025-05-14T22:00:00"
}
```

Actualizar Ubicación

- **Endpoint**: PUT /api/ubicaciones/{id}
- **Descripción**: Actualiza los detalles de una ubicación existente.
- **Parámetros**:
  - id (ruta, requerido): El ID de la ubicación.
- **Cuerpo de la Solicitud**: Objeto JSON con los campos a actualizar (al menos uno requerido).
- **Formato de Solicitud**:

```
{
  "foodTruckId": Long (opcional),
  "ciudad": String (opcional),
  "calle": String (opcional),
  "fecha": String (opcional)
}
```

- **Respuesta**: 200 OK, objeto JSON de la ubicación actualizada; 404 Not Found si el ID no existe; 400 Bad Request si no se proporcionan campos.
- **Formato de Respuesta**: Igual que Obtener Ubicación por ID.

Ejemplo:

```
curl -X PUT http://localhost:8080/api/ubicaciones/2 \
-H "Content-Type: application/json" \
-d '{"calle":"Broadway"}'
```

Respuesta:

```
{
  "id": 2,
  "foodTruckId": 1,
  "foodTruckNombre": "Taco Fiesta",
  "ciudad": "Nueva York",
  "calle": "Broadway",
  "fecha": "2025-05-14T22:00:00"
}
```

Eliminar Ubicación

- **Endpoint**: DELETE /api/ubicaciones/{id}
- **Descripción**: Elimina una ubicación por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID de la ubicación.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, sin contenido; 404 Not Found si el ID no existe.

Ejemplo:

```
curl -X DELETE http://localhost:8080/api/ubicaciones/2
```

Respuesta: (Cuerpo vacío)

Obtener Ubicaciones Recientes por Food Truck

- **Endpoint**: GET /api/ubicaciones/foodtruck/{foodTruckId}
- **Descripción**: Recupera las ubicaciones recientes de un food truck específico.
- **Parámetros**:
  - foodTruckId (ruta, requerido): El ID del food truck.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos ubicación; 404 Not Found si el food truck no existe.
- **Formato de Respuesta**: Igual que Obtener Todas las Ubicaciones.

Ejemplo:

```
curl -X GET http://localhost:8080/api/ubicaciones/foodtruck/1
```

Respuesta:

```
[
  {
    "id": 1,
    "foodTruckId": 1,
    "foodTruckNombre": "Taco Fiesta",
    "ciudad": "Nueva York",
    "calle": "5th Ave",
```

```
    "fecha": "2025-05-14T22:00:00"
  },
  ...
]
```

Notificaciones

Obtener Todas las Notificaciones

- **Endpoint**: GET /api/notificaciones
- **Descripción**: Recupera una lista de todas las notificaciones en el sistema.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, arreglo JSON de objetos notificación.
- **Formato de Respuesta**:

```
[
  {
    "id": Long,
    "mensaje": String,
    "fechaEnvio": String,
    "usuarioId": Long
  },
  ...
]
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/notificaciones
```

Respuesta:

```
[
  {
    "id": 1,
    "mensaje": "Tu pedido # лиц ha sido completado.",
    "fechaEnvio": "2025-05-14T22:00:00",
    "usuarioId": 2
  },
  ...
]
```

Obtener Notificación por ID

- **Endpoint**: GET /api/notificaciones/{id}
- **Descripción**: Recupera los detalles de una notificación específica por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID de la notificación (ej., 1).
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, objeto JSON de la notificación; 404 Not Found si el ID no existe.
- **Formato de Respuesta**:

```
{
  "id": Long,
```

```
  "mensaje": String,
  "fechaEnvio": String,
  "usuarioId": Long
}
```

Ejemplo:

```
curl -X GET http://localhost:8080/api/notificaciones/1
```

Respuesta:

```
{
  "id": 1,
  "mensaje": "Tu pedido #19 ha sido completado.",
  "fechaEnvio": "2025-05-14T22:00:00",
  "usuarioId": 2
}
```

Crear Notificación

- **Endpoint**: POST /api/notificaciones
- **Descripción**: Crea una nueva notificación para un usuario.
- **Parámetros**: Ninguno
- **Cuerpo de la Solicitud**: Objeto JSON con los detalles de la notificación.
- **Formato de Solicitud**:

```
{
  "usuarioId": Long,
  "mensaje": String,
  "fechaEnvio": String
}
```

- **Respuesta**: 200 OK, objeto JSON de la notificación creada; 400 Bad Request si la validación falla.
- **Formato de Respuesta**: Igual que Obtener Notificación por ID.

Ejemplo:

```
curl -X POST http://localhost:8080/api/notificaciones \
-H "Content-Type: application/json" \
-d '{"usuarioId":2,"mensaje":"Tu pedido #37 ha sido
completado.","fechaEnvio":"2025-05-14T22:00:00"}'
```

Respuesta:

```
{
  "id": 2,
  "mensaje": "Tu pedido #37 ha sido completado.",
  "fechaEnvio": "2025-05-14T22:00:00",
  "usuarioId": 2
}
```

Actualizar Notificación

- **Endpoint**: PUT /api/notificaciones/{id}
- **Descripción**: Actualiza los detalles de una notificación existente.

- **Parámetros**:
  - id (ruta, requerido): El ID de la notificación.
- **Cuerpo de la Solicitud**: Objeto JSON con los campos a actualizar (al menos uno requerido).
- **Formato de Solicitud**:

```
{
  "usuarioId": Long (opcional),
  "mensaje": String (opcional),
  "fechaEnvio": String (opcional)
}
```

- **Respuesta**: 200 OK, objeto JSON de la notificación actualizada; 404 Not Found si el ID no existe; 400 Bad Request si no se proporcionan campos.
- **Formato de Respuesta**: Igual que Obtener Notificación por ID.

Ejemplo:

```
curl -X PUT http://localhost:8080/api/notificaciones/2 \
-H "Content-Type: application/json" \
-d '{"mensaje":"Tu pedido #37 ha sido completado con éxito."}'
```

Respuesta:

```
{
  "id": 2,
  "mensaje": "Tu pedido #37 ha sido completado con éxito.",
  "fechaEnvio": "2025-05-14T22:00:00",
  "usuarioId": 2
}
```

Eliminar Notificación

- **Endpoint**: DELETE /api/notificaciones/{id}
- **Descripción**: Elimina una notificación por su ID.
- **Parámetros**:
  - id (ruta, requerido): El ID de la notificación.
- **Cuerpo de la Solicitud**: Ninguno
- **Respuesta**: 200 OK, sin contenido; 404 Not Found si el ID no existe.

Ejemplo:

```
curl -X DELETE http://localhost:8080/api/notificaciones/2
```

Respuesta: (Cuerpo vacío)

Obtener Notificaciones por Usuario

- **Endpoint**: GET /api/notificaciones/usuario/{usuarioId}
- **Descripción**: Recupera todas las notificaciones de un usuario específico.
- **Parámetros**:
  - usuarioId (ruta, requerido): El ID del usuario.
- **Cuerpo de la Solicitud**: Ninguno

- **Respuesta**: 200 OK, arreglo JSON de objetos notificación; 404 Not Found si el usuario no existe.
- **Formato de Respuesta**: Igual que Obtener Todas las Notificaciones.

Ejemplo:

```
curl -X GET http://localhost:8080/api/notificaciones/usuario/2
```

Respuesta:

```
[
  {
    "id": 1,
    "mensaje": "Tu pedido #19 ha sido completado.",
    "fechaEnvio": "2025-05-14T22:00:00",
    "usuarioId": 2
  },
  ...
]
```

Manejo de Errores

La API utiliza códigos de estado HTTP estándar para indicar éxito o fallo:

- **200 OK**: La solicitud fue exitosa.
- **400 Bad Request**: Datos de solicitud inválidos (ej., campos requeridos faltantes).
- **404 Not Found**: Recurso no encontrado (ej., ID inválido).
- **500 Internal Server Error**: Error del servidor (ej., problema con la base de datos).

Las respuestas de error incluyen un objeto JSON con un campo message:

```
{
  "message": "Recurso no encontrado"
}
```