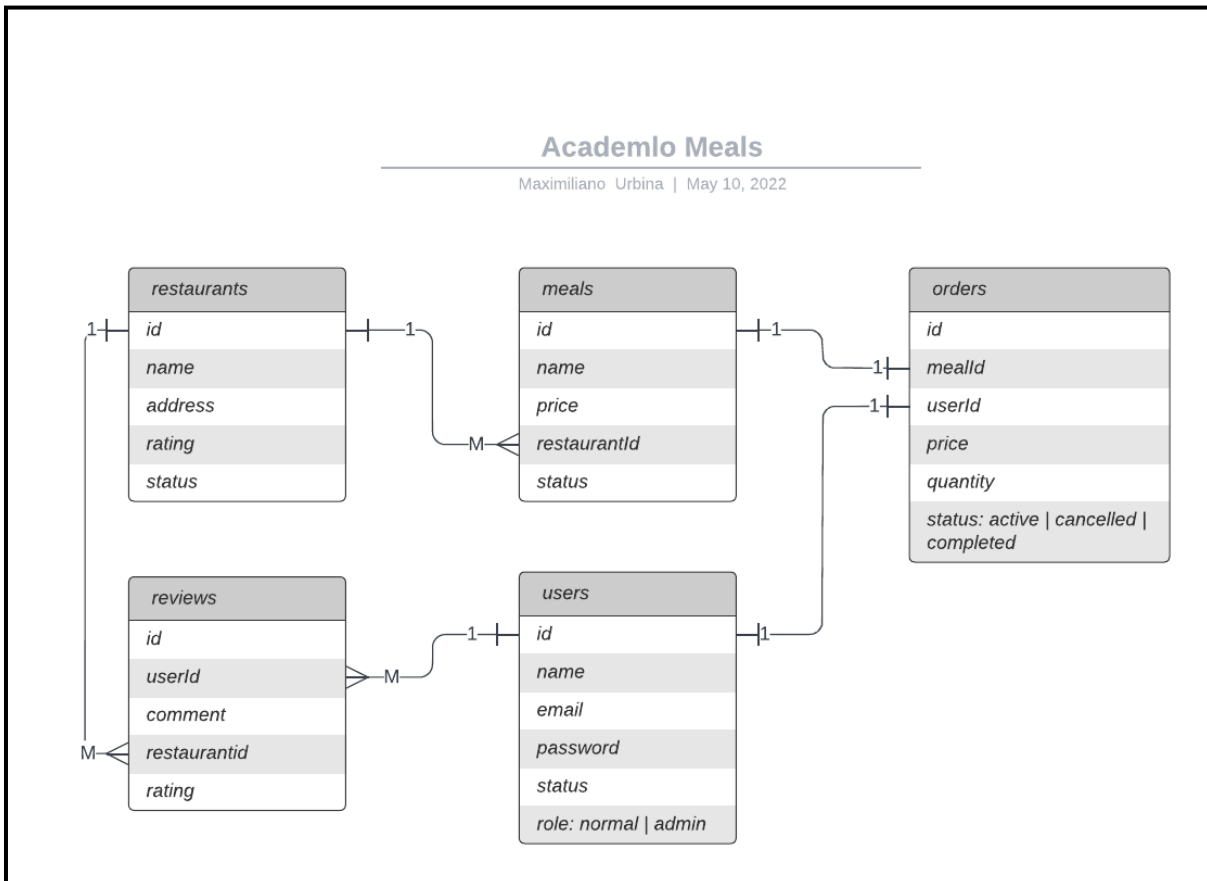


## Requerimientos para proyecto #2: Academlo Meals

Este proyecto no contara con un front-end, la intención es que el alumno sea capaz de experimentar lo más real posible el entorno de un desarrollador de back-end, el resultado final debe ser un servidor de Express que sea capaz de cumplir con los puntos a mencionar.

Se deben considerar los siguientes modelos y sus relaciones:



Se deben considerar los siguientes endpoints:

/api/v1/users		
HTTP Verb	Route	Description
POST	/signup	Crear usuario (enviar username, email, y password por req.body)
POST	/login	Iniciar sesión (enviar email y password por req.body)
PATCH	/:id	Actualizar perfil de usuario (solo name y email)
DELETE	/:id	Deshabilitar cuenta de usuario
GET	/orders	Obtener todas las ordenes hechas por el usuario

<b>GET</b>	<b><i>/orders/:id</i></b>	<i>Obtener detalles de una sola orden dado un ID</i>
------------	---------------------------	--

- Todas las rutas, excepto para crear usuario e iniciar sesión, se deben proteger por un medio de autenticación, es decir, por JWT.
- Se debe usar express-validator para el endpoint de crear usuarios.
- El endpoint ***/orders*** y ***/orders/:id***, debe buscar los productos del usuario en sesión (del token que se envió), extraer el id del token y usarlo para buscar dichas órdenes.
- Los métodos **PATCH** y **DELETE** deben estar protegidos para que únicamente el dueño de la cuenta a modificar pueda realizar dichas acciones.
- Para los endpoints ***/orders***, se debe incluir la siguiente información:
  - El restaurant de donde se pidió la comida

<b><i>/api/v1/restaurants</i></b>		
<b>HTTP Verb</b>	<b>Route</b>	<b>Description</b>
<b>POST</b>	<b><i>/</i></b>	<i>Crear un nuevo restaurant (enviar name, address, rating (INT))</i>
<b>GET</b>	<b><i>/</i></b>	<i>Obtener todos los restaurants con status <b>active</b></i>
<b>GET</b>	<b><i>/:id</i></b>	<i>Obtener restaurant por id</i>
<b>PATCH</b>	<b><i>/:id</i></b>	<i>Actualizar restaurant (name, address)</i>
<b>DELETE</b>	<b><i>/:id</i></b>	<i>Deshabilitar restaurant.</i>
<b>POST</b>	<b><i>/reviews/:id</i></b>	<i>Crear una nueva reseña en el restaurant, siendo :id el id del restaurant (enviar comment, rating (INT) en req.body)</i>
<b>PATCH</b>	<b><i>/reviews/:id</i></b>	<i>Actualizar una reseña hecha en un restaurant, siendo :id el id del restaurant (comment, rating) SOLO EL AUTOR DE LA RESEÑA PUEDE ACTUALIZAR SU PROPIA RESEÑA</i>
<b>DELETE</b>	<b><i>/reviews/:id</i></b>	<i>Actualizar una reseña hecha en un restaurant a status <b>deleted</b>, siendo :id el id del restaurant. SOLO EL AUTOR DE LA RESEÑA PUEDE ACTUALIZAR SU PROPIA RESEÑA</i>

- Todas las rutas, excepto **GET /** y ***/:id***, deben estar protegidas por un método de autenticación. Se debe incluir las reseñas de los restaurants.
- El endpoint para crear restaurants, debe estar protegido con express-validator.
- Los endpoints **PATCH */:id*** y **DELETE */:id*** deben estar protegidos para que únicamente el usuario admin pueda realizar estas acciones.

<b>/api/v1/meals</b>		
<b>HTTP Verb</b>	<b>Route</b>	<b>Description</b>
<b>POST</b>	<b>/:id</b>	<i>Crear una nueva comida en el restaurant, siendo :id el id del restaurant (enviar name, price (INT) en req.body)</i>
<b>GET</b>	<b>/</b>	<i>Obtener todas las comidas con status <b>active</b></i>
<b>GET</b>	<b>/:id</b>	<i>Obtener por id una comida con status <b>active</b></i>
<b>PATCH</b>	<b>/:id</b>	<i>Actualizar comida (name, price)</i>
<b>DELETE</b>	<b>/:id</b>	<i>Deshabilitar comida</i>

- Todas las rutas, excepto **GET /** y **/:id**, deben estar protegidas por un método de autenticación.
- El endpoint para crear comidas, debe estar protegido con express-validator.
- Los métodos **PATCH** y **DELETE** deben estar protegidos para que únicamente el usuario admin pueda realizar estas acciones.
- Para los endpoints **GET**, se debe incluir la información de su restaurant.

<b>/api/v1/orders</b>		
<b>HTTP Verb</b>	<b>Route</b>	<b>Description</b>
<b>POST</b>	<b>/</b>	<i>Crear una nueva order (enviar quantity y mealId por req.body)</i>
<b>GET</b>	<b>/me</b>	<i>Obtener todas las ordenes del usuario</i>
<b>PATCH</b>	<b>/:id</b>	<i>Marcar una orden por id con status <b>completed</b></i>
<b>DELETE</b>	<b>/:id</b>	<i>Marcar una orden por id con status <b>cancelled</b></i>

- Todas las rutas deben estar protegidas por un método de autenticación.
- Para el endpoint **POST /** se debe realizar lo siguiente:
  - Se debe buscar si existe la comida (meal), si no, enviar error.
  - Calcular el precio para el usuario, multiplicar el precio de la comida (meal) encontrada previamente, por la cantidad solicitada por el usuario.
  - Crear una nueva orden, pasando el precio calculado, el mealId de la comida ya encontrada y la cantidad solicitada por el usuario.
- Para el endpoint **PATCH y DELETE**, validar que la orden este con status **active** antes de realizar la operación, enviar error en caso de que no tenga este status.
  - Solo el usuario que hizo la orden solamente puede realizar estas operaciones

- Para el endpoint **/me**, se debe incluir la información de la comida que se ordenó, y del restaurant de donde se pidió la comida.