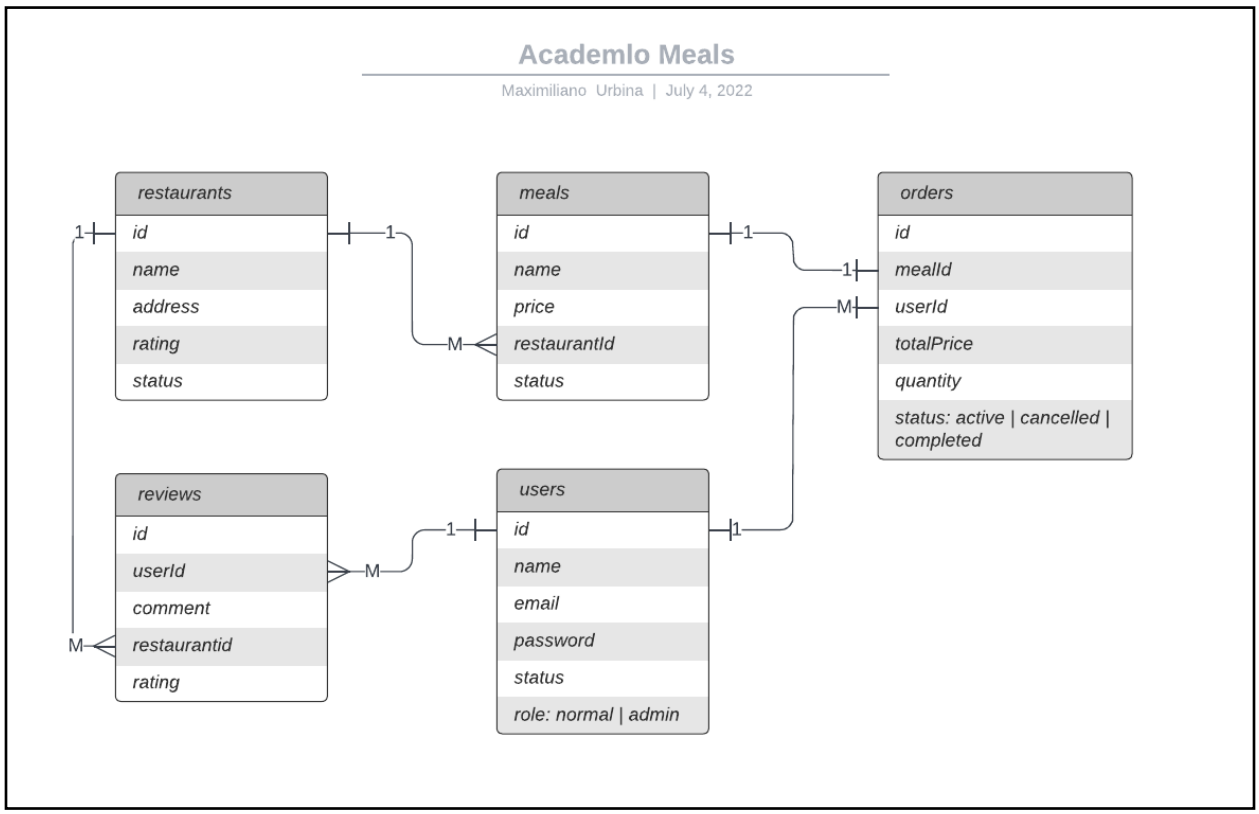
## Requerimientos para proyecto #1: Academlo Meals

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:




En el modelo **orders,** en la columna totalPrice, se refiere al precio total de la orden, se calcula multiplicando el precio unitario de la comida por la cantidad que el usuario solicito.

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* |
| **GET** | ***/orders/:id*** | *Obtener detalles de una sola orden dado un ID* |

- Todas las rutas, excepto para crear usuario e iniciar sesión, se deben proteger por un medio de autentificación, es decir, por JWT.
- Se debe usar express-validator para el endpoint de crear usuarios.
- El endpoint ***/orders*** *y* ***/orders/:id***, debe buscar las órdenes 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

| **/api/v1/restaurants** | | |
|---|---|---|
| **HTTP Verb** | **Route** | **Description** |
| **POST** | */* | *Crear un nuevo restaurant (enviar name, address, rating (INT))* |
| **GET** | */* | *Obtener todos los restaurants con status* ***active*** |
| **GET** | ***/:id*** | *Obtener restaurant por id* |
| **PATCH** | ***/:id*** | *Actualizar restaurant (name, address) UNICAMENTE EL ADMIN PUEDE REALIZAR ESTA ACCIÓN* |
| **DELETE** | ***/:id*** | *Deshabilitar restaurant. UNICAMENTE EL ADMIN PUEDE REALIZAR ESTA ACCIÓN* |
| **POST** | ***/reviews/:restaurantId*** | *Crear una nueva reseña en el restaurant, siendo :* ***restaurantId*** *el id del restaurant (enviar comment, rating (INT) en req.body)* |
| **PATCH** | ***/reviews/:id*** | *Actualizar una reseña hecha en un restaurant, siendo :id el id de la reseña (comment, rating) SOLO EL AUTOR DE LA RESEÑA PUEDE ACTUALIZAR SU PROPIA RESEÑA* |
| **DELETE** | ***/reviews/:id*** | *Actualizar una reseña hecha en un restaurant a* |

| | | |
|---|---|---|
| | | *status **deleted**, siendo :id el id de la reseña. SOLO EL AUTOR DE LA RESEÑA PUEDE ACTUALIZAR SU PROPIA RESEÑA* |

- Todas las rutas, excepto **GET /** y **/:id,** deben estar protegidas por un método de autentificación. 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.

| **/api/v1/meals** | | |
|---|---|---|
| **HTTP Verb** | **Route** | **Description** |
| **POST** | ***/:id*** | *Crear una nueva comida en el restaurant, siendo :id el id del restaurant (enviar name, price (INT) en req.body)* |
| **GET** | */* | *Obtener todas las comidas con status **active*** |
| **GET** | ***/:id*** | *Obtener por id una comida con status **active*** |
| **PATCH** | ***/:id*** | *Actualizar comida (name, price) UNICAMENTE EL ADMIN PUEDE REALIZAR ESTA ACCIÓN* |
| **DELETE** | ***/:id*** | *Deshabilitar comida UNICAMENTE EL ADMIN PUEDE REALIZAR ESTA ACCIÓN* |

- Todas las rutas, excepto **GET /** y **/:id,** deben estar protegidas por un método de autentificación.
- 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.

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

- Todas las rutas deben estar protegidas por un método de autentificación.
- 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.