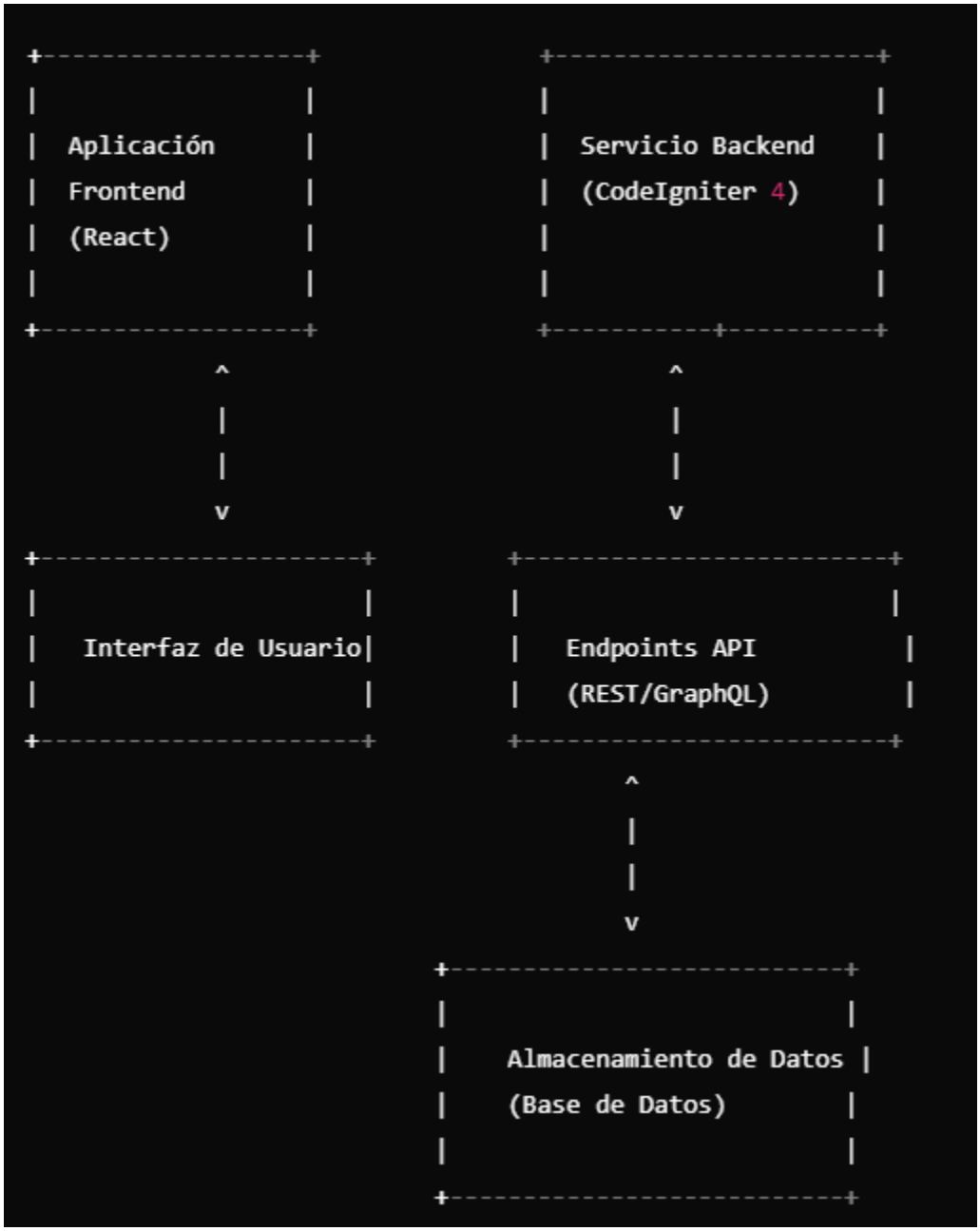
sistema de reservas utilizando React para el frontend y CodeIgniter 4 para el backend



1. **Aplicación Frontend (React)**:
    - **Tecnología**: React
    - **Responsabilidades**:
        - Proporcionar una interfaz de usuario intuitiva para realizar, modificar y gestionar reservas.
        - Mostrar el estado de las reservas, las fechas disponibles y formularios de reserva.
        - Comunicarse con el servicio backend a través de llamadas API.
    - **Componentes**:
        - **Formulario de Reserva**: Permite a los usuarios realizar nuevas reservas.
        - **Lista de Reservas**: Muestra las reservas actuales y permite su modificación.
        - **Panel de Usuario**: Muestra la información específica del usuario y el historial de reservas.
2. **Servicio Backend (CodeIgniter 4)**:
    - **Tecnología**: CodeIgniter 4
    - **Responsabilidades**:
        - Manejar las solicitudes API del frontend.
        - Procesar la lógica de negocio relacionada con las reservas (e.g., verificación de disponibilidad, confirmaciones de reserva).
        - Interactuar con el almacenamiento de datos para almacenar y recuperar información de reservas.
    - **Componentes**:
        - **Endpoints API**: Manejar operaciones CRUD para reservas (Crear, Leer, Actualizar, Eliminar).

Traer todos los datos: `http://localhost/linktic-prueba/reservation/public/`

Crear reserva: `http://localhost/linktic-prueba/reservation/public/booking/`

Actualizar reserva: `http://localhost/linktic-prueba/reservation/public/modify-reservation/1`

Eliminar reserva: `http://localhost/linktic-prueba/reservation/public/cancel-reservation/1`

Ver la reserva: `http://localhost/linktic-prueba/reservation/public/view-reservation/1`

3. **Almacenamiento de Datos**:
    - **Tecnología**: Base de Datos MySQL
    -

El diagrama de bases de datos basado en la estructura y datos proporcionados para el esquema `test-linktic`. Este diagrama incluye tres tablas: `services`, y `type`. A continuación, se describe cómo se relacionan las tablas:

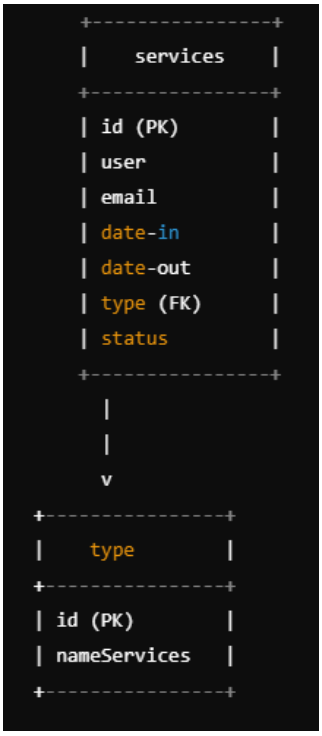
Diagrama de Base de Datos

1. **Tabla: `services`**
    - `id` (INT, PK)
    - `user` (VARCHAR)
    - `email` (VARCHAR)
    - `date-in` (VARCHAR)
    - `date-out` (VARCHAR)
    - `type` (INT, FK -> `type.id`)
    - `status` (ENUM('Y', 'N'))
2. **Tabla: `type`**
    - `id` (INT, PK)
    - `nameServices` (VARCHAR)

Relaciones y Descripción

- **`services`** tiene una columna `type` que actúa como una clave foránea referenciando el `id` en la tabla **`type`**. Esto indica que cada servicio en `services` está asociado con un tipo específico definido en la tabla `type`.

Diagrama de Entidad-Relación (ERD)



Diagrama

- **Tabla `services`**: Registra información sobre servicios, incluyendo el tipo de servicio (`type`), que es una clave foránea referenciando la tabla `type`.
- **Tabla `type`**: Define los diferentes tipos de servicios disponibles.

    - **Responsabilidades**:
        - Almacenar detalles de reservas, información de usuarios y otros datos relevantes.
        - Asegurar la integridad y consistencia de los datos.
    - **Componentes**:
        - **Tabla/colección de Reservas**: Almacena detalles sobre cada reserva.

Instalacion.

Frontend, clone el repositorio desde (branch: **Dev**) y arranque con npm start para que inicie la visualización.

Backend, clone el repositorio en un ambiente apache(branch **services**), configurar la db en el archivo principal database.php o .env, correr archivo db.sql

Flujo de Interacción

1. **Interacción del Usuario**:
    - El usuario interactúa con la aplicación frontend para hacer o gestionar una reserva.
    - La aplicación frontend envía una solicitud al servicio backend a través de los endpoints API.
2. **Procesamiento del Backend**:
    - El servicio backend procesa la solicitud, aplicando la lógica de negocio (e.g., comprobación de disponibilidad).
    - El servicio backend realiza operaciones CRUD en el almacenamiento de datos.
3. **Almacenamiento de Datos**:
    - El almacenamiento de datos mantiene los detalles de las reservas, la información de los usuarios y otros datos relacionados.
    - El servicio backend recupera o actualiza datos según sea necesario.

4. **Actualización del Frontend**:
    - Después de procesar la solicitud, el servicio backend envía una respuesta de vuelta al frontend.
    - La aplicación frontend actualiza la interfaz de usuario según la respuesta (e.g., confirmación de reserva).

- **Escalabilidad**: Asegurar que la arquitectura pueda manejar una carga aumentada utilizando tecnologías escalables (e.g., balanceadores de carga, particionamiento de bases de datos).
- **Manejo de Errores**: Implementar mecanismos robustos para el manejo de errores y registro de eventos para gestionar y depurar problemas.

Este diagrama y descripción ofrecen una visión clara de cómo interactúan los componentes y cómo fluye la información a través del sistema. Puedes ajustar estos elementos según los requisitos específicos de tu proyecto y las tecnologías utilizadas.