**1. Exponer la Lógica de C++ como un Servicio**

Para que tu backend en Node.js pueda interactuar con el sistema en C++, considera exponer la lógica y los datos de C++ como un servicio. Esto se puede hacer de varias maneras:

Opción A: API RESTful o gRPC

* **Crear un Servicio Web en C++**: Utiliza frameworks como **Restbed** (para REST) o **gRPC** (para RPC) en C++ para exponer tus datos y lógica de negocio a través de una API web. Este servicio actuará como un intermediario entre tu backend de Node.js y tu código de C++.
* **Comunicación**: Tu servidor Node.js puede hacer llamadas HTTP o RPC a este servicio para realizar operaciones como obtener puntos de usuario, actualizarlos, o ejecutar el sorteo del jackpot.

Opción B: Módulo Nativo de Node.js

* **Node.js Addons con N-API**: Escribe un addon para Node.js que permita invocar directamente funciones C++ desde JavaScript. Esto es más complejo y requiere un conocimiento profundo de ambos lados, pero puede ofrecer un rendimiento superior y una integración más estrecha.
* **node-ffi-napi**: Como alternativa menos invasiva y más fácil de usar que escribir un addon, puedes usar **node-ffi-napi** para cargar dinámicamente bibliotecas C++ en Node.js y llamar a sus funciones.

**2. Frontend en React**

El frontend en React interactuará únicamente con tu backend en Node.js, sin necesidad de conocer los detalles de la implementación en C++. React enviará solicitudes (p.ej., para ejecutar el sorteo del jackpot) al backend de Node.js, que a su vez comunicará con el servicio o módulo C++ según sea necesario.

* **React App**: Implementa componentes React para mostrar la interfaz del usuario, como el saldo de puntos, la opción para participar en el jackpot, y mostrar los resultados.
* **Axios/Fetch API**: Utiliza estas bibliotecas para hacer solicitudes HTTP desde React a tu servidor Node.js para obtener datos o ejecutar acciones.

**3. Backend en Node.js**

Tu servidor Node.js actúa como un puente entre el frontend en React y tu sistema subyacente en C++. Aquí gestionarás las solicitudes del cliente, procesarás la lógica de negocio necesaria y comunicarás con el servicio C++ para obtener datos o ejecutar operaciones.

* **Express.js**: Un framework popular para construir la API en Node.js, manejando rutas, solicitudes y respuestas.
* **Comunicación con C++**: Según la opción elegida (API RESTful/gRPC o módulo nativo), implementa la lógica necesaria para interactuar con tu código C++ desde Node.js.

**4. Consideraciones Importantes**

* **Seguridad**: Asegúrate de que todas las comunicaciones entre React, Node.js y C++ sean seguras, especialmente si manejas datos sensibles de los usuarios.
* **Rendimiento**: Monitorea el rendimiento de las llamadas entre Node.js y C++, ya que esto podría ser un cuello de botella, especialmente si optas por la comunicación a través de una API web.
* **Mantenibilidad**: Mantener un sistema que cruza lenguajes y entornos puede ser desafiante. Documenta bien las interfaces y considera las implicaciones de cambios futuros en cualquier parte del sistema.

Este enfoque te permite aprovechar las fortalezas de cada tecnología: C++ para el manejo eficiente de datos y operaciones críticas, Node.js como un servidor flexible y fácil de usar para lógica de negocio y comunicación con el cliente, y React para una interfaz de usuario dinámica y moderna.