Diseño de sistemas

Proponga una solución de arquitectura de muy alto nivel (no se necesita código) para la siguiente declaración:

X Company proporciona un producto de software como servicio llamado Parrot, Parrot debería poder proporcionar operaciones como Crear, Leer, Actualizar y Eliminar documentos de la empresa, herramienta de informes, administración de usuarios y conexión a servicios externos como Slack y CRM del cliente. **Tenga en cuenta lo siguiente:** existe una base de datos única para toda la solución, por lo que su diseño debe poder manejar diferentes clientes utilizando la misma base de datos. Proporcione lo siguiente:

- Diseño de backend (¿Cómo organizaría el backend, si va a utilizar servicios, describa cada servicio?)
- Diseño de Frontend (¿Qué marco usará, qué patrón? ¿Redux? ¿Servicios angulares?)
- Protocolo de autorización (cómo manejar el inicio de sesión)
- ¿Qué base de datos? SQL, sin SQL, ¿ambos? Explique este punto.
- Recomendaciones de rendimiento (cómo escalar cuando miles de usuarios utilizan este SaaS)
- Cómo le gustaría a esta solución en la nube (puede usar un proveedor específico como AWS o ser más genérico) Opcional

body-parser: para analizar las solicitudes de peticiones entrantes al servidor express: como framework o marco de trabajo de node JS mysgl: modulo para simplificar la conexión y operar mejor las consultas al gestor BD de Mysal Diseño del Backend multer: para subida de imágenes en este caso de los empleados o usuarios Jsonwebtoken: modulo para brindar acceso a usuarios Módulos a **Node JS** Instalar bcryptjs: modulo para encriptación tanto síncrona como asíncrona de claves y parámetros de Usuario confidenciales Via npm file-upload: para subir documentos o cualquier otro tipo de archivos al servidor **Tecnologías** nodemailer, twilio: para contactar a los empleados de la empresa @slack/web-api: para realizar solicitudes a la API web de Slack **Dependencias de desarrollo:** typescript, dotenv, nodemon, ts-node. Cada servicio a utilizar tendrá su propio controlador el cual contendrá la lógica de programación para mantener el Servicios a patrón MVC dentro de todo lo posible. **Programar** Servicios: de bitácoras de usuarios para manejar con mayor control lo que cada usuario hace, de reporteria y todo lo referente a operaciones CRUD o REST API de los usuarios, guardar, eliminar, consultar, actualizar, cambio de clave o ajustes de cuenta, mensajería por correo entre otras operaciones a realizar en el futuro Únicamente aquí se consumiría la API de slack mediante @slack/web-api paquete contiene un cliente HTTP simple, conveniente y configurable para realizar solicitudes a la API web de Slack. Úselo en su aplicación para Rest-Api o consumo de servicios llamar a cualquiera de los más de 130 métodos y deje que maneje el formateo, la cola, el reintento, la de terceros paginación y más. y módulos pertenecientes a CRM de clientes.

Di/seño de Frontend (¿Qué marco usará, qué patrón? ¿Redux? ¿Servicios angulares?)

<u>R://</u> Primeramente Propongo usar el marco o framework Angular ya que por defecto es un framework MVC y también por el hecho de que incorpora tanto Typescript como Scss entre otras funcionalidades que lo hacen muy robusto y eficiente, para manejar estados si no es posible manejarlos con los decoradores @Input o @output que angular trae por defecto, se usara NGXS ya que es un manejador de estados más completos, también como la servicios angulares con observables para poder acceder a dicho servicio desde cualquier componente inyectándolo al componente que sea requerido reutilizando código y de esa manera haciendo más eficiente y escalable el código.

Como segundo propongo React una librería de solo la vista, de igual manera mantiendo el modelo o patrón MVC usando redux como manejador de estado.

Protocolo de autorización (cómo manejar el inicio de sesión)

<u>R://</u> usando los modulos descritos en el mapa de arriba, primero se hara la encriptacion de la clave de un usuario logueado y se le asignara un token de acceso una vez este logueado o registrado dicho token contara con vencimiento o expiración esto es de parte del backend, en cuanto al frontend si es Angular el que se utilizara se protegerán las rutas con Guards y certificación https, caso contrario si es con React las rutas se protegerán con React.Context();

¿Qué base de datos? SQL, sin SQL, ¿ambos? Explique este punto.

R:// Se utilizará una base de datos relacional o SQL en este caso MySql, primeramente, porque hay más control y orden en una base de datos relacional que en una no relacional, si bien es cierto la cuestión de rendimiento es muy importante pero pienso que el rendimiento de MySql es muy bueno hasta con una gran cantidad de información siempre y cuando las relaciones que apliquemos y la estructura de las tablas, procedimientos almacenados, Jobs , triggers etc. de dicha base de datos este muy bien realizada.

Recomendaciones de rendimiento (cómo escalar cuando miles de usuarios utilizan este SaaS)

<u>R://</u> Una buena escalabilidad se logra desde el interior, bueno en cuando a la aplicación de cómo este estructurado el proyecto en sí o el código de programación, realizando una buena modularizacion de cada aspecto cada funcionalidad cada controlador código limpio y aplicando buenas prácticas de programación, aplicando pruebas de integración y pruebas unitarias para corroborar el correcto funcionamiento, y también que la plataforma que utilicemos para hacer deploy de la aplicación nos ayude con eso y contando con un equipo de logística o servicio, y en cuanto a la empresa una de las mejores formas seria con Business Inteligence minería de datos y conceptos de datawarehouse.

Cómo le gustaría a esta solución en la nube (puede usar un proveedor específico como AWS o ser más genérico) Opcional

La ventaja de AWS es que se puede tener todo en la misma organización AWS y ayuda el hecho de que se escala de manera separada lo que promete una escalabilidad muy alta y un gran rendimiento, también está el hecho de que puedes configurarlo a tu manera seleccionando lo que tus necesidades requieran SO, lenguaje de programación Base de datos.

De igual manera propongo también el uso de Digital Ocean que ofrece algo similar, pero con la diferencia del costo del alojamiento ya que es significativamente menor, de igual manera es posible crear un entorno propio acorde a las necesidades, cabe recalcar que lo he probado, pero no con licencia comercial pero es una Solucion muy valida también.