DESAFÍO DE CODIFICACIÓN DE BACKEND ACEPTADO

Este desafío técnico nos ayudará a tener una idea de cómo usaría sus habilidades técnicas, conocimientos y experiencia para proporcionar infraestructura de back-end para nuestros usuarios finales.

Realmente valoramos tu tiempo y para nosotros es muy importante, por lo que este desafío ha sido diseñado con eso en mente. No hay una cantidad específica de tiempo que le lleve completar este desafío, ya que el nivel de esfuerzo dependerá de qué tan familiarizado esté con el desarrollo de back-end y si aborda o no la funcionalidad opcional. ¡Siéntase libre de pasar todo el tiempo que desee construyendo su solución preferida y esperamos que se divierta construyéndola!

MISIÓN

Primero un poco de contexto:

Las bicicletas robadas son un problema típico en las grandes ciudades. **La Policía quiere ser más eficiente en la resolución de casos de bicicletas robadas**. Decidieron crear un software que pueda **automatizar sus procesos**: el software que va a desarrollar.

Su misión es crear una infraestructura de back-end que **proporcione API para una aplicación de front-end** que satisfaga todos los requisitos a continuación.

REQUISITOS

1. Los dueños de bicicletas pueden **reportar una bicicleta robada**.

2. Una bicicleta puede tener **múltiples características**: número de placa, color, tipo, nombre completo del dueño, fecha, descripción del robo, dirección donde fue robada la bicicleta, estado del caso.

3. La policía tiene **varios departamentos** que son responsables de las bicicletas robadas.

4. Un departamento puede tener **cierta cantidad de oficiales** de policía que pueden trabajar en casos de bicicletas robadas.

5. La Policía puede **escalar su número de departamentos** y puede **aumentar el número de policías por departamento**.

6. Cada oficial de policía debe poder buscar bicicletas **por diferentes características** en una **base de datos** y ver qué **departamento** es responsable de un caso de bicicleta robada.

7. Los **nuevos casos** de bicicletas robadas deben **asignarse automáticamente a cualquier oficial de policía libre** en cualquier departamento.

8. **Un oficial** de policía solo puede manejar **un caso de bicicleta** robada a la vez.

9. Cuando la policía encuentra una bicicleta, el caso se **marca como resuelto** y el **oficial de policía responsable está disponible** para tomar un nuevo caso de bicicleta robada.

10. El sistema debería poder **asignar de bicicletas robadas no asignadas** automáticamente cuando un oficial de policía esté disponible.

11. Incluya **instrucciones** para construir y **probar** el backend y cualquier **otra información** relevante que deba tener en cuenta en un archivo **README.md**.

12. Cargue su código en **Github**, Bitbucket o Gitlab.

IDEAS

Las ideas enumeradas a continuación son todas opcionales y debe sentirse libre de implementar cualquiera de estas u otras de sus preferencias según lo permita el tiempo.

¡No sea tímido y déjenos sorprender con su aplicación!

1. Cuando se registre una bicicleta robada, **guarde las coordenadas de latitud y longitud** según la información de la dirección.

2. **Implementación del proceso de inicio de sesión/registro y RBAC** (Control de acceso basado en **roles**) para: Propietarios de bicicletas, Oficiales de policía, Departamentos de directores de policía. Asigne el permiso de reglas según sus propios criterios.

3. Los propietarios de bicicletas pueden **verificar el estado** introduciendo la identificación del registro asociado al caso.

4. Envíe una **notificación al propietario** de la bicicleta por correo electrónico cuando cambie el estado de su caso.

Utilice cualquier biblioteca, marco, motor de base de datos o herramienta que se ajuste a sus necesidades técnicas.

BONUS

1. Una **aplicación frontend** simple React o Vue, y/o una página web impulsada por Express, que ofrece una visualización de todos o parte de los datos utilizando la API que ha creado como backend.

2. **Pruebas unitarias** y de integración

3. **Documentación de APIs** con una herramienta visual tipo Slate, Swagger, ApiDoc o cualquier otra de su preferencia,

RECURSOS

Para obtener las coordenadas de latitud y longitud, puede usar la solución que prefiera. Sugerimos, para mayor comodidad, la **API de codificación geográfica de Google**. Tiene límites de nivel gratuitos aceptables y está bien documentado.

Para enviar notificaciones por correo electrónico, también puede utilizar cualquier solución que prefiera. Sugerimos **SendGrid**, ya que también tienen un límite de nivel gratuito aceptable y está bien documentado.

CONSIDERACIONES

Entre otros criterios, su presentación será evaluada en:

1. **Implementación** de los requisitos establecidos.

2. **Arquitectura** de fondo.

3. La **calidad** general del código y su **resistencia al bloqueo**.

4. Su uso de las **convenciones** de codificación de Javascript.

5. Conocimiento y uso de **bibliotecas** y SDK de Javascript/NodeJS.

6. Claridad de las **comunicaciones** en comentarios y otra documentación.

7. **Consultas** de base de datos eficientes

8. No se trata solo de manejar el camino feliz**. Piense en la concurrencia, la seguridad, la escalabilidad**.

9. **Haz pública tu API**. Despliéguelo utilizando el servicio de su elección (es decir, AWS, Heroku, Digital Ocean)

10. Los revisores de desafíos de codificación también deberían poder cargar su proyecto y **ejecutarlo fácilmente en un entorno local**. Si el revisor necesita realizar alguna configuración del proyecto (es decir, agregar sus propias claves API de Google/SendGrid, instalar motores de base de datos locales) y no puede simplemente instalar npm o algo similar, esa es razón suficiente para rechazar su solución.

11. Apreciamos que se pide mucho en este ejercicio. Si necesita más tiempo, no dude en preguntar. Si necesita quitarle prioridad a algo, aplique el mismo criterio que usaría en un proyecto profesional, argumente su decisión.

Ah, y no te preocupes si tu solución tiene personalidad o sentido del humor. ¡Queremos que te diviertas!