Vaga Desenvolvedor(a) em Node.js - Cognum

Avaliação Técnica

🌟 Olá, Candidato(a) à Cognum! 🌟

Primeiramente, gostaríamos de expressar nossa gratidão por você se interessar em fazer parte da nossa equipe e embarcar conosco nessa jornada de inovação e desenvolvimento.

Queremos que saiba que nossa avaliação técnica é projetada para entender suas habilidades, seu ritmo e sua familiaridade com as tecnologias que usamos. No entanto, entendemos que a vida é repleta de imprevistos e compromissos, e nem sempre temos o tempo ou as condições ideais para mostrar todo o nosso potencial.

Dito isso, **os desafios propostos não são obrigatórios**. Se você decidir não completar todos eles, não se preocupe. Não desclassificaremos ninguém com base nisso. Nosso objetivo é identificar seu nível de maturidade com as tecnologias propostas e sua capacidade de entrega no tempo disponível.

Entendemos que cada pessoa tem um conjunto único de habilidades, experiências e circunstâncias, e valorizamos a diversidade em todas as suas formas. Estamos mais interessados em conhecer **você**, sua paixão por tecnologia e sua abordagem para resolver problemas do que simplesmente verificar se todas as caixas de um desafio foram marcadas.

Então, respire fundo, dê o seu melhor e, acima de tudo, divirta-se com os desafios! Estamos ansiosos para conhecer mais sobre você através de seu código e suas soluções.

Um abraço caloroso de toda a equipe Cognum e boa sorte! 💡🚀

Instruções:

1. Configuração Inicial:
    a. Inicie um novo projeto Node.js em sua máquina local. Você pode usar o gerador de aplicativos Express, a CLI do NestJS ou qualquer outra ferramenta/framework de sua preferência.
    b. Estruture seu projeto seguindo as boas práticas de desenvolvimento e padrões de design que você está acostumado.
2. Ao completar cada desafio, faça commits frequentes, demonstrando seu progresso e pensamento por trás de cada etapa.
3. Após finalizar todos os desafios que desejar realizar, crie um repositório no GitHub e faça o push de sua solução.
4. Não esqueça de incluir um arquivo README.md detalhando como configurar, executar e testar sua solução, além de quaisquer outras informações que considere relevantes.

Lembrando mais uma vez, você não precisa completar todos os desafios. Faça o que sentir que é capaz no tempo que tiver disponível. Estamos interessados em entender sua abordagem, suas habilidades e sua paixão pela tecnologia. Desejamos a você um ótimo desafio! 🌟🚀

Desafios

Desafio 1

Objetivo: Implementar uma API RESTful simples usando Express.js.

Instruções:

1. Crie uma API que tenha um endpoint GET /hello.
2. Ao acessar o endpoint, ele deve retornar uma mensagem: {"message": "Hello, Cognum!"}.
3. Garanta que sua API esteja rodando na porta 3000.

Critérios de Avaliação:

- ☐ O código está claro e limpo.
- ☐ O endpoint foi implementado corretamente.
- ☐ A resposta está no formato correto.

Desafio 2:

Objetivo: Integração com banco de dados e operações CRUD.

Instruções:

1. Utilize um banco de dados de sua escolha (MySQL, PostgreSQL, MongoDB).
2. Implemente uma estrutura de CRUD para um recurso chamado Employee com os seguintes campos: id, name, role.
3. O sistema deve suportar as operações básicas: CREATE, READ, UPDATE, DELETE.

Critérios de Avaliação:

- ☐ A escolha e implementação do banco de dados é apropriada.
- ☐ A estrutura do banco está correta.
- ☐ As operações CRUD funcionam sem erros.
- ☐ O código está organizado e limpo.

Desafio 3:

Objetivo: Integração com uma API desconhecida e avaliação de leitura em inglês.

Instruções:

1. Você deve integrar sua aplicação do desafio 2 (CRUD de **Employee**) com a API externa **Random User Generator**. Esta API fornece informações aleatórias sobre usuários, que você usará para povoar sua base de dados.
2. Estude a documentação da API **aqui**. Seu objetivo é buscar detalhes de 10 usuários e inseri-los em sua base de dados como **Employee**. Note que a API pode fornecer mais informações do que você precisa; você deve mapear os dados relevantes para seu sistema.
3. Além das operações CRUD tradicionais, implemente um novo endpoint '**GET /populate'**. Ao chamar este endpoint, sua aplicação deve:
    a. Consumir a API Random User Generator para obter detalhes de 10 usuários.
    b. Transformar esses detalhes no formato do seu recurso **Employee**.
    c. Inserir esses novos employees em sua base de dados.
    d. Retornar uma resposta com os 10 novos employees inseridos.

Critérios de Avaliação:

- ☐ A capacidade de entender e integrar com a API externa.
- ☐ Transformação apropriada dos dados do Random User Generator para o formato Employee.
- ☐ O novo endpoint funciona conforme esperado, populando o banco de dados com novos registros.
- ☐ O código é bem organizado, considerando a nova complexidade introduzida.
- ☐ Entendimento correto da documentação em inglês e aplicação das funcionalidades da API externa.

Desafio 4:

Objetivo: Construir um micro-serviço escalável com balanceamento de carga, cache e monitoramento em tempo real.

Instruções:

1. Micro-serviço de notificação:
    a. Crie um micro-serviço separado em Node.js que seja responsável por enviar notificações (e-mails fictícios) para os **Employees**. Para simular o envio de notificações, você pode simplesmente registrar uma mensagem no console ou em um arquivo de log.
    b. Esse serviço deve expor um endpoint **POST /send-notification** que aceite uma lista de IDs de Employee e uma mensagem. Ao receber uma solicitação, ele deve "enviar" a notificação para os Employees correspondentes.
2. Escalabilidade e Balanceamento de Carga:
    a. Implemente o balanceamento de carga usando algo como o **nginx** ou outra solução de sua escolha. Seu objetivo é ter várias instâncias do micro-serviço de notificação rodando e distribuir a carga entre elas.
3. Cache:
    a. Integre um sistema de cache como **Redis** para armazenar informações frequentemente acessadas dos Employees. Por exemplo, quando uma notificação é enviada, antes de consultar o banco de dados, o sistema deve verificar se as informações do Employee estão disponíveis no cache.
4. Monitoramento em Tempo Real:
    a. Integre uma ferramenta de monitoramento em tempo real, como **Prometheus** ou **Grafana**. Esta ferramenta deve fornecer métricas sobre:
        i. Número de notificações enviadas por minuto.
        ii. Tempo médio de resposta dos endpoints.
        iii. Quantidade de solicitações atendidas por cada instância do micro-serviço.

Entrega:

- ☐ Seu código deve ser acompanhado de um Dockerfile para cada serviço e um docker-compose.yml para inicializar todo o sistema.
- ☐ Inclua um arquivo README.md detalhado com instruções sobre como configurar, executar e testar sua solução, além de explicações sobre as decisões de design tomadas.

Critérios de Avaliação:

- ☐ Correta implementação e separação do micro-serviço de notificação.
- ☐ Eficiência na implementação do balanceamento de carga e sua justificativa.
- ☐ Uso eficaz do sistema de cache para melhorar o desempenho.
- ☐ Configuração e visualização apropriada das ferramentas de monitoramento em tempo real.
- ☐ Qualidade do código, organização e clareza das soluções implementadas.
- ☐ Robustez e capacidade de lidar com falhas e cargas pesadas de solicitações.