Kandydat: Jan Kowalski
Technologia: JavaScript
Subtechnologie: React, Node.js, TypeScript
Poziom: Junior
Język rozmowy: Polski
Data: 01/01/2024

Ogólne Wrażenia

Podczas spotkania wykazałeś się dobrą znajomością JavaScriptu i TypeScriptu oraz podstawową znajomością Reacta i Node.js (poziom Junior). Nie znasz bardziej zaawansowanych mechanizmów Reacta i Node.js. Masz braki z zakresu testowania kodu.

Podczas rozwiązywania problemów powinieneś bardziej skupić się na prostocie tworzonego rozwiązania, przeanalizować możliwe rozwiązania i jaśniej tłumaczyć swoje pomysły. Twoje pomysły były proste ale uzasadnione a zaangażowanie i dotychczasowe doświadczenia są obiecujące. Według nas kwalifikujesz się na pozycję Junior.

Doświadczenie i projekty

Podczas aplikowania na poziom juniorski ważne są prywatne projekty. Bardzo dobrze, że je posiadasz. Jeszcze lepiej, że projekty są ambitne i wykorzystujesz w nich wiele sensownych technologii.

Uwagi:

- Czasem używasz zmiennych w konwencji camelCase a czasem snake_case. Dobrze byłoby to ustandaryzować. To samo tyczy się organizacji kodu i nazewnictwa plików.
- Twoje metody często odpowiadają za kilka/kilkanaście rzeczy przez co są bardzo długie i nieczytelne.
- Podczas rozmowy wspomniałeś, że angażujesz się w projekty open-source'owe. Dobrze byłoby o tym wspomnieć w CV.

Umiejętności techniczne

Na większość pytań odpowiedziałeś wyczerpująco i poprawnie. Zdarzyły się jednak takie, gdzie twoja odpowiedź była zbyt ogólna lub jej nie udzieliłeś.

1. **React i ogólny frontend**
    - Znasz podstawowe metody renderowania strony - w wolnej chwili możesz także poczytać o ISR oraz SSG i poznać różnicę między nimi
    - Nie znasz pojęcia memoizacji w React
    - Masz podstawową wiedzę o hookach w React jednak nie znasz pojęcia customowych hooków
    - Nie do końca rozumiesz różnice pomiędzy DOM a Virtual DOM
    - Znasz Reduxa i Context API

2. **Craftsmanship**
    - Starasz się robić komponenty reużywalne, znasz niektóre dobre praktyki w kodowaniu, wspomniałeś zasady DRY, KISS, zachęcamy jedynie do szerszego zrozumienia zasad SOLID
    - Wiesz co robić i jak szybko działać, gdy zablokuje cię task
3. **Backend**
    - Znasz podstawy NodeJS
    - Znasz ORMy i wiesz jakie są ich plusy
    - Wiesz jak działa OAUTH i do czego służą tokeny JWT
    - Wiesz jakie są różnice między REST, a GQL
    - Nie znasz założeń Node.js (event loop)
    - Nie do końca zdajesz sobie sprawę ze wszystkich funkcjonalności metody next()
4. **Typescript**
    - Widać, że znasz i rozumiesz Typescript
    - Nie potrafiłeś wskazać antypatternów przy korzystaniu z TSa
5. **Javascript**
    - Świetnie poradziłeś sobie z pytaniami dotyczącymi JSa (deep clone obiektów, konstrukcje wprowadzone w ES6, wskazanie niepoprawnie działających fragmentów kodu)
6. **Testowanie**
    - Wymieniłeś kilka narzędzi pozwalających testować kod ale nie powiedziałeś jakiego typu testy możesz tworzyć przy ich pomocy

Rozwiązywanie problemów

Podczas rozmowy otrzymałeś takie zadanie:

Tablicę arr = [1, 1, 2, 4, 9, 1, 3, 5, 3, 7] przekształć w taki sposób aby zawierała tylko unikalne wartości

Postanowiłeś rozwiązać zadanie dodając tablicę pomocniczą a cały algorytm oprzeć o dwie pętle (pętla w pętli). Twoje rozwiązanie zadziałało ale nie jest to najlepszy sposób. Złożoność obliczeniowa napisanego przez ciebie algorytmu to $O(n^2)$. Dużo lepiej wykorzystać pomocniczy obiekt zamiast tablicy (obiekty przechowują dane w formie klucz => wartość). W ten sposób mógłbyś usunąć wewnętrzną pętlę a algorytm miałby złożoność O(n). Jeszcze lepszym podejściem jest wykorzystanie metody **Set()**.

Proponowane rozwiązanie:

```
const uniqueArray = (arr) => Array.from(new Set(arr));
```

Wiedza o złożonościach obliczeniowych wykracza poza poziom Juniorski jednak dobrze jest mieć to na uwadze i optymalizować algorytmy. Dodatkowo, użycie pomocniczego obiektu w tym przypadku powinno być dla ciebie naturalne.

Komunikacja

1. **Forma odpowiedzi**
    - Zwykle wyczerpująca i rozbudowana jeśli dobrze znasz temat.
2. **Współpraca z rekruterem**
    - W kilku sytuacjach miałem wrażenie, że gdybyś dopytał mnie o szczegóły zadanego pytania to odpowiedziałbyś lepiej.

- Każdą niejasność dyskutuj z rekruterem, dopytuj. Brak pytań do rekrutera działa na twoją niekorzyść.

3. **Dialog**
    - Podczas pisania algorytmu zabrakło z Twojej strony komunikowania nad czym aktualnie się zastanawiasz lub co reprezentuje pisany przez ciebie kod.
    - Twoje zdolności komunikacyjne w sferze miękkiej są na wysokim poziomie, co sprawiło, że dobrze się z Tobą rozmawiało.
    - Twoja otwartość i umiejętność słuchania podkreślają Twoją zdolność do efektywnej współpracy w zespole.

Dodatkowe uwagi

Skup się na pogłębianiu wiedzy w React i Node.js. Przeanalizuj dobre praktyki programistyczne. Warto także nauczyć się korzystać z narzędzi/bibliotek do testowania jak Jest czy Cypress, które są często wymagane w ofertach pracy z frontendem. W wolnej chwili spójrz na materiały dostępne tutaj: https://github.com/getify/You-Dont-Know-JS. Inne uwagi/pojęcia na które powinieneś zwrócić uwagę to:

- Dobór odpowiedniej architektury aplikacji, jak zorganizować projekt, granice odpowiedzialności modułów, wady/zalety różnych podejść
- Złożoności obliczeniowe
- Metody wbudowane w Javascript
- Pisanie kodu zgodnego z SRP (wniosek po analizie twoich projektów)