Screening interview for backend

Step 1

Post a very specific job offers. Don’t recruit backend engineers just for the sake of it. Recruits people for specific subjects (not the same needs for all team). We shouldn’t recruit a pool of people, that we will “re-route” to team in need. We should list what we exactly need and recruit what we exactly need.

Step 2

Let’s meet with the candidate. Cultural fit and motivation should the most important characteristics at this step. We should have an informal talk about what he wants, what we’re expecting from him, and what he’s capable of.

Check if already made something he deems interesting (doesn’t has to be something complex/advanced)

Let him speak about what he has done in a previous job (or at school or even for a personal project). Something he think was technically interesting, and/or something he’s proud of.

As a bonus: Do he have a github with interesting projects?

Check if he has skill-sets compatible with the job offer

Ask him on how comfortable he is with some concepts he would need to understand for the job. It’s specific to where he will go, but could be something like: multi-threading, message passing, sql, some architecture design, ...

We should have a list of concepts and questions to start a technical conversation with the candidate. We could regroup a list of questions about concept we’re using in the company, and what we should expect as answers to considered the candidate knows a concept well enough.

The goal here is not to trick the candidate, but let him confidently speak about what he knows. It’s important here for the interviewer to have the mindset of someone who will

recruit a future teammate. It’s not a showdown.

Also check for red flags

- Must be capable of saying he doesn’t know something
- Must not lie, or greatly overestimate what he knows
- Must not avoid a question (answering something else or changing subject many time in a row, on purpose)

Check if he’s curious

It means many things at once. Is the candidate someone who like to learn about things? Be it about the company, about tech or about people.

Interview curiosity

- When he hasn’t the answer to something, is he asking for the solution?
- If he doesn’t know a concept, does he ask for what is it? Or for resource on how to learn this concept? (He could also say something like: “I should document myself more about this subject, seems interesting”).

General curiosity

- Is he following new tech? (technology watch)
- Does he have any technical hobbies, not linked to the company (like demo scene, reverse engineering, making a game, moderator on a tech forum, electronic, teaching, ...)
- Does he like testing new things?
- Does he like reading technical book, or articles?

Company curiosity

- Does he ask any questions about the company?
- Does he ask why we’re working for our company? Or how interesting it is?

Company technical curiosity

- Does he ask how we made feature X?
- Does he ask how we're doing thing?
- Does he ask about our technical stack?
- Does he ask about throughput and volume?

Step 3

Junior and senior couldn't be treated the same. We expect a junior just to know how to code, but we expect a senior how to make architecture, and structure a code, on top of that.

Junior

For a junior, let him remake a small existing part of the company. The goal here is to see if he's able to do it, and do it cleanly (clean code, not horrible performance). If he's able to do that, it means he would have succeeded in the company.

Give an assignments to do at home, without any time limit. Let the candidate choose how much time he need (and check how well he's good at estimating that).

Also check for any bonus (things not asked on purpose):

- Does he commented his code?
- Does he documented his project?
- Does he made unit tests?
- Does he made benchmarking?
- Does he make any non required things to improve the project? (Doesn't need to be implemented, could be written ideas).

Senior

For a senior we could ask for an architecture design of something existing in the company. We should ask him to write a paper. The goal here is to see if he's able to design correctly something. The assignment is followed by an in-person technical discussion. The assignments and the talk after weights equally.

It’s possible to have a bad design, but if we point the flaw and the candidate fix that during the conversation, it’s fine.

Time limit here is not necessary. More time doesn’t mean a better design. You either know how to design or not.

Also check for any bonus:

- Does he propose something not asked to improve the whole solution?
- Does he speak about monitoring, alerting? (Not asked on purpose)
- Does he made any images, schema, graphic?
- Does he checks the efficiency of his design, using O(n) complexity notation, math, or simple back-of-the-envelope calculation?