Author: Bartosz ‘BaSz’ Szurgot

job interview - candidate questions

contrary to some ppl's opinion, job interview works both ways: employer checks if your knowledge meets expectations, you check if this job will be fun and creative. below is my personal list of questions i tend to ask, before accepting any job offer. recently i've decided to make a list out of these, since often, when i forgot about some parts, they came out not the way i expected… questions are important not to get disappointed on the first day. if the employer does not want to spare some time on answering this, it means that either you'll be just another “human resource” no one gives a damn about, or is hiding something. both cases you probably prefer to step back.

1. **how will recruiting process look like?**
   I. how many interviews?
   II. when need to go on-site?
   III. how long before getting the feedback?
2. **working conditions**:
   I. where is the office located? *think about getting there and back on a daily basis.*
   II. is there a place to park a car? *even if you don't use car daily, sometimes it might still be needed – where will you park then?*
   III. open space or separate rooms? *i find open spaces noisy, which makes it impossible to focus on work – this in turn kills innovation.*
   IV. what's the dressing policy? *not everyone's finding working “under a tie” acceptable.*
   V. is there obligatory (long) lunch break? *it's annoying when employer tells you when and how long you will be eating lunch. especially when you usually don't…*
   VI. ask recruiter “why do YOU like your job?”. *nice way of observing if your (potential) future coworker really enjoying work, or “just works here”.*
   VII. can get support to do any off-time projects? *say: extra money for a hardware, or a spare computation power at night.*
   VIII. have time for writing/creating publications? *some employers allow to publish interesting materials in magazines. this means +1 for your surname and +1 for company's PR.*
   IX. have time for preparing conference talks? *some employers support presenting skills of they employees during conferences. this means +1 for your surname and +1 for company's PR.*
   X. who will you work with? seniors/juniors? any1 known in the industry? *the best bet would be to get a job where you are the least experience in the team – this will ensure you to learn a lot! :)*
3. **employment**:
   I. how much? :)
      a. salary net/gross?
      b. bonus size and how often payed?

c. tax break?

II. is work time flexible?

III. does work include supports, weekends, night shifts, etc… *your time is grates resource you have – be sure not to waste it.*

IV. is work from home possible? how often?

V. is part time employment possible? *money are not the top priority – in a long run, it's all about the time.*

VI. can hire as an external contractor? *this usually avoids few social taxes*

4. **trainings**:

I. does employer ensure trainings?

II. how often? how long? *employer should provide at least few days training every year, as a "social minimum".*

III. can go to a conference? *sometimes on-site is the only option, so for instance ACCU-organized conferences are out of scope.*

5. **work tasks**:

I. what challenges shall i expect at work? *if the tasks as too simple, you'll be bored to death…*

II. ask what could be your first 3 tasks (can be examples). what tasks you can get after 3-6 months? *this will give you a nice overview on what can you expect on day-to-day basis.*

III. development? *how much actual programming there is daily?*

IV. software architecture? *can you design or at least influence architecture?*

V. CM? *who does configuration management. can it be influenced, if you see field for improvement?*

VI. build management? *the same as for CM.*

6. **project organization**:

I. is the market regulated (eg.: banking, medical devices, etc…)? *regulated markets typically means a lot of paper work, waterfall-like processes and very formal approaches to everything. might be what you want, but be sure to think twice.*

II. how to introduce a change/fix/improvement? how long does it take? *after proving this is work it with PoC or a fair discussion, this should be a no-event.*

III. how much time is dedicated to finding new solutions and redesigning old ones? *many companies do not actually expect any change to come to their project. if so – R U N !*

IV. what methodology do you use: SCRUM, RUP, SDLC+, none? *are you sure you want to take part in project which main task is to produce documentation and/or meeting minutes?*

V. how do you document project? *there are reasonable limits both ways – no documentation usually means no architecture and spaghetti code; tones of documentation usually means "programming in ms-word".*

VI. is project local or multi-site? *multi-site is not bad as such, but often means a lot of politics involved in your daily work.*

VII. who makes project decisions (which company site/location)? *it's all about decision making. be sure your location have something to say.*

VIII. what is the policy is bug is found is open-source library? *it's good to share – otherwise you'll need to maintain your "sneaky patch" forever…*

IX. does company maintain any open-source projects or participate in any? *again - openness for sharing*

7. **technologies**:[1]
   I. C++14?
   II. Linux?
   III. gcc? clang? what versions?
   IV. boost?
   V. other open source libraries?
   VI. which (distributed) version control system?
   VII. docker?
   VIII. virtualization?
   IX. big-scale/cloud solutions?
   X. what technologies company is proud of (either developing or using)? *it tells a lot about who do they target to hire and what environment you can expect.*
8. **workstations**:
   I. is linux installed locally? *corporate environments are often abusing windows everywhere – even when it does not fit your job description.*
   II. working locally or remotely? *working on "very" remote machines may mean lags and latencies – not nice.*
   III. what machine is there? *will it be reasonable fast to perform your work? if you develop locally loads of CPUs and RAM is what you want! :)*
9. **toolchain and environment**:
   I. is the project greenfield, recent or legacy?
   II. what's the code-base size (in KLOC)? *bigger gets exponentially worse in legacy code. small may mean relatively new or redesigned project.*
   III. is continuous integration used? what tools are used to implement it? *CI is standard nowadays. but note, that having build server does NOT equals to having CI.*
   IV. is continuous delivery used? what tools are used to implement it? *CD is a step towards from CI, needed to minimize human factors. having real CD usually means good CI.*
   V. have automated tests? *no automated testing usually means tightly coupled architecture with tons of errors lurking in a code… usually legacy code.*
      a. how many tests are there?
      b. how many KLOC for tests vs. KLOC for production? *by the rule of thumb – 50:50 is usually nice.*

c. what is code coverage? *though the % does not guarantee anything, high % coverage may suggest ppl are really testing what they are doing.*

d. what types of tests are implemented (UT, MT, IT, ST)? which are automated, which manual?

VI. is there project documentation?

a. is it up to date?

b. who's updating it?

c. how this documentation is done? *usually its a good idea to keep it in VCS-friendly format, like LaTeX or asciidoc in the same repo as the code, so that it can be easily updated to track changes in the code.*

last but not least, there is one important observation you should make. think about the questions/tasks you were given during the interview. how hard they were for you? were you able to answer all of them straight away or you had to think hard, maybe failed on some? the harder questions were, the better for you. recruiters usually ask about things and skills that are required for a given job. if you found them hard and innovative, there is a good chance you'll actually learn a lot there and not get bored. in contrast, if the interview was simple (or even trivial) – this is the way your job will probably look like. still interested? being overqualified for a job is boring on a daily basis and a step back in a long term. if your kung-fu is really the best around, who do you plan to learn from?