CASE STUDY – Kotlin Engineer- Bidding

This assignment is designed to give you a glimpse of some of the challenges you will be facing in this role. Please be aware there are no perfect solutions - for us, it's more important to see how you find solutions, process your ideas, structure your thoughts and how you make your decision paths.

Be creative but realistic about what's possible. We are thrilled to get to know a bit more about the way you solve tasks.

Are you up for the challenge?

SUBMISSION DEADLINE: As indicated in the email.

THE CHALLENGE

At trivago we maintain a large amount of hotel data. Imagine you need to write a tool that converts the data from one format to other formats. This should not be a one-off script. Instead, imagine that your team will need to maintain the tool and extend it to other formats.

Task:

Write a command line program that fulfills the following requirements:

1. It reads data from an input CSV file. We provide a sample `hotels.csv` file with the structure that the program may expect for its input.
2. Then, it validates the data. To keep it simple, let's go with the following rules:
- A hotel name may only have up to 100 characters.
- Hotel ratings are given as a number from 0 to 5 stars.
- The URL is syntactically correct.
4. Finally, it writes the information of the file in an output format chosen by the user: either JSON or XML.
5. For each of the previous points, design the tool in such a way that it is easy to add new input file formats, validation rules or new output formats.

Your check-list:

- The programming language should be **Kotlin.**
- Feel free to choose any framework, library or existing tool you like. The only constraint is that your tool needs to run on our local machine (using Docker for this is a nice to have, but not a requirement).

- We'll test your code on an unmodified Debian stable machine or the current Mac OSX operating system, depending on your choice.
- There's no need to create a front-end for the tool, but a nice command line interface would be a plus.
- We care more about **code quality (readability and architecture)** than about performance, although fast execution is always nice.
- Consider what **should happen when the tool fails** and **how easy it is to debug a problem.**
- We understand that the specifications for the tasks are vague. This is intended. In case of doubt, try to **come up with sensible defaults and document your decisions**.
- In any case, a README file is always welcome!

Good luck! 😊