# **CASE STUDY – PHP Engineer (Advertiser Connectivity Engineering)**

This case study puts you in the position of a PHP backend developer at trivago.

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 – Build a hotel data converter

Your code needs to fulfill the following requirements:

1. Your solution should be implemented in PHP 7.X. It should make use of the Symfony 5 framework.
2. In the resources section, you can find an initial project. It contains:
   1. a basic composer.json file
   2. input data in <project\_root>/var/in: a JSON file and a XML file which you should use as data sources
3. Your code should be able to convert these formats into CSV via a CLI command called „trivago:convert“ and should receive an input file as its only argument. The command should generate a file called hotels\_<input\_format>.csv in the folder „<project\_root>/var/out“.
4. Validate the data. To keep it simple, please follow these rules:
   1. A hotel name may not contain non-ASCII characters.
   2. The hotel URL must be valid (please come up with a good definition of "valid").
   3. Hotel ratings are given as a number from 0 to 5 stars (never negative numbers).
5. Provide unit and/or integration tests for your code which should be runnable via „composer test“.

**Bonus tasks**

If you find additional time here are some extra options you might consider. These are optional to your submission. Make the tool as easily extensible to new input formats as possible.  
We care more about code quality (readability, software architecture) than code performance - although fast execution is a plus.

Generate a log file to let potential users know how they could improve data quality. Add options to sort/group/filter the data before writing it to the output file.

**How we run your code**

Your code will run on an x86-64 machine with the most recent release or the recent Mac OSX.

The code will run on a local (not docker) environment. Here we will trigger the CLI command described above and compare the outcome with what we expect.