BACK END ASSESSMENT

V5.0 – 08-11-2022

BACK END WEB DEVELOPMENT ASSESSMENT

Congratulations, you have made it to the next phase of DTT's application procedure!

This phase will consist of a back-end web development assessment made in **PHP**. During your time at DTT, you will work for strong international brands, including Greenpeace, Randstad, ING, Philips, Rabobank, FOX Sports, and KPMG. Through this assessment, you can demonstrate that you are capable of contributing to DTT and her clients. DTT promises her clients a detail orientated approach, a high quality standard for deliveries, and a work ethic that goes that extra mile: we look for applicants that understand these values.

This assessment gives us a better understanding of how you work, and provides you with insight into what you can expect when working at DTT.

THE ASSESSMENT

We challenge you to develop a (REST) API that stores and retrieves information about catering facilities. Such an API can be consumed by different clients, such as a mobile application, a web page or a CMS.

For this assessment, it is important that you understand the concept of CRUD. Your project should be **straight-forward** and you should keep things **simple**. The code should be secure, properly structured, well commented and object-oriented (OOP).

Before you start:

I. Make sure to read the whole assignment before you start.
II. Make sure to read the included README.md to set up and test your local development environment before starting development.
III. Log the hours you spend on the project in the provided hour log spreadsheet. Provide a detailed description of your activities including the time spent to assist us in the grading process.
IV. The assessment can be completed within 20 hours. Try to adjust the scope based on the available time. Time taken for bonus points will not be included in the final time evaluation.

Additional requirements:

I. The included skeleton PHP project should be used as a starting point.
II. MySQL/MariaDB should be used as the database for persistent storage.
III. The request body data can be formatted as *form-data*, *x-www-form-urlencoded* or *raw JSON format*. The returned response data has to be in *raw JSON format*.
IV. The API should follow the REST architectural style:
    a. Correct structure of the URLs of the API endpoints
    b. Correct choice of used HTTP method based on the type of API call
    c. Correct choice of returned HTTP status code
V. Hour log spreadsheet included
VI. Database dump, including a few sample records for each table
VII. Export of API documentation created using the Postman platform that includes all the available API calls.

User stories

DTT uses user stories as a baseline for all our (technical/design) documentation. Make sure your assessment adheres to the user stories and conditions of satisfaction as stated below.

For the minimal viable product (MVP), the following user stories are to be implemented:

User story 1: As a developer, I want to create a database for persistent storage of the data.

- *The **data types** should be chosen appropriately.*
- ***Primary and foreign keys** should be used correctly.*
- ***Junction tables** should be used correctly.*
- ***Database Schema requirements:***
    i. *A **Facility** has a name and creation date.*
    ii. *A **Tag** has a name.*
    iii. ***Tag** names MUST be unique and re-usable among different **Facilities**.*
    iv. *A **Location** has a city, address, zip code, country code and phone number.*
    v. *A **Facility** can have one **Tag**, multiple **Tags**, or none at all*
    vi. *A **Facility** must always belong to one **Location**.*

User story 2: As a user, I want to manipulate **facilities** within my API.

- *I want to be able to **Create** a facility and its tags.*
- *I want to be able to **Read one** facility, its location and its tags.*
- *I want to be able to **Read multiple** facilities, their location and their tags.*
- *I want to be able to **Update** a facility and its tags.*
- *I want to be able to **Delete** a facility and its tags.*

User story 3: As a user, I want to be able to search for facilities with a **search query.**

- *I want to be able to search facilities by the **facility name**, **tag name**, or **location city**, or **any combination of those** in a **single** API call.*
- *I want the returned results to also include **partial matches**. For example, searching for location city 'ams' should return all facilities with city 'Amsterdam'.*

Bonus features to make an impression (all optional)

I. Usage of **version control (Git)**
II. Implement **pagination** for the results (extra bonus for cursor pagination).
III. Implement **sanitation of client data**
IV. Implement **employees of the facilities** to further display your understanding of database table relationships.
V. Add an **authentication mechanism** to secure the API endpoints.
VI. Include configuration that would allow the API to run as a **Docker container(s)**.
VII. Write **unit tests** using PHPUnit framework.
VIII. Have a fun idea? We'd love to see it! 🙂

These bonus features are optional and do not count towards your 'total time'. We do not negatively judge an assessment that does not include bonus features, but they will be reviewed.

Limitations

I. The API should be written in plain PHP **without** relying on 3rd party frameworks (Laravel, Symfony, etc.).
II. It is not allowed to make use of API generators.
III. When consulting an online source while working on the project, refer to these sources in your hour logs. We are fine with a tutorial being used as long as the assessment deviates/adds enough for us to grade your ability to create something of your own.

Delivery

When you are satisfied with your API, **zip** your project (including your hour log, API documentation and database dump) and send the zip as attachment to apply@d-tt.nl. Make sure to **not include third party code installed by Composer** in the */vendor* folder.

If you've used a GIT repository, please follow the steps below:

I. Send the link to the Git repository to *apply@d-tt.nl* and make sure to explain that you're done with the project. Also, please invite *apply@d-tt.nl* as a contributor with Admin privileges

II. Make sure that the repository includes the full project source code.
III. Make sure that the repository includes a copy of any other deliverables:
    A. Hour log spreadsheet.
    B. Database dump including sample records
    C. Your API documentation (export of Postman collection)
    D. Etc.

Our review

We review your assessment on different criteria. Functional requirements are graded on their completeness and the chosen method of implementation. The quality and structure of code are graded on (among other factors) consistency, clarity, optimization, extensibility, re-usability, and single-responsibility. The hour log is graded on its completeness, language, and level of detail.

Among other factors, the following points are essential to us:

I. **Standards:** Adhere to PSR standards – particularly naming conventions – and implement them consistently.
II. **Clarity**: You should strive to make your code easy to understand without relying on comments as a crutch. Use comments to explain more complex parts of the code. Do not use comments to state the obvious.
III. **Single responsibility**: Is the code maintainable, isolated, generalized, reusable. Always avoid duplicate code and avoid putting different types of operations in the same file or functions.

Questions?

If you have any questions regarding the requirements of the assignment, or if you are stuck on a problem for too long, please don't hesitate to contact us at *apply@d-tt.nl*.

More DTT

Feel free to have a look at all our apps at: https://www.d-tt.nl/en/apps.