LARAVEL ASSESSMENT

In this assessment, you will be building an application that scans and shows fraudulent customers using the customers API. This assessment is entirely fictional. Feel free to add your own touch and put your creativity to work – but make sure you meet the requirements!

SUBMISSION GUIDELINES

- Share your GitHub/GitLab repo link at least **48 hours** before the interview.
- Include setup instructions in a README.md (how to run the code locally).
- Mention what you've implemented and elaborate on the following questions:
    - What did you implement?
    - Why did you implement or skip certain things?
    - What trade-offs did you make (e.g., time, complexity)?
    - If you had more time, what would you improve or add?

CONTEXT

This assignment is designed not just to evaluate your technical skills, but also to understand your **decision-making process,** how you approach challenges, and how you **prioritize features under constraints.** During the interview, we'll use your solution as a starting point to discuss your choices, trade-offs, and any hurdles you encountered.

It's not about perfection! We're more interested in your thinking and how you communicate your approach.

What is fraudulent activity?

- Two customers with the same IP-address or same IBAN are both fraudulent.
- A customer with a phone number from outside the Netherlands is fraudulent.
- A customer that is younger than 18 years old is fraudulent.

1. THE AVAILABLE API

You will be integrating your application with an API:

- The customer API: this API allows you to retrieve a list of all customers.
    - Use our dedicated Docker Image **vzdeveloper/customers-api** to launch the API locally (https://hub.docker.com/r/vzdeveloper/customers-api)
    - **Note:** the customer API can be unstable, be mindful of error handling in your app!

2. REQUIREMENTS

- Time: minimum 1 hour, maximum 16 hours.
- Setup your application with the latest Laravel version.
- Create a Blade page with a trigger (e.g. button) that can start the fraud scan process:
    - The application retrieves all customers from the customer API and checks them for fraudulent activity (this is explained below).
    - The application displays all customers and highlights the fraudulent ones.
- Create a model named “Scan” and “Customer”.
- Every time a scan is completed, insert a new “Scan” with the following data:
    - Scan date/time.
    - Create and attach all customer models (relationship).
    - Per customer, keep track of whether they were marked as fraudulent or not.
- Create a Blade page that displays all scans that have been executed, including all recorded data for that scan.
- Error handling with basic error states (e.g., when the API fails to load)

3. EXTRA TASKS (BONUS)

You can do these extra tasks, in any order, if you have time left.

- Create API endpoints to serve the data.
- Use Tailwind CSS to style your application.
- Display the reason why a customer was reported after scanning, for example: “Customer has non-NL phone nr.” or “Customer has same IP as John Doe (127.0.0.1)”.
- Cache the last scan results and restore them when the page is refreshed.
- Test your application using Pest.