Moroccan PHPers

1st Challenge - February 2022

Estimated needed time:

We expect you would need about 5 to 10 hours to complete this task. Including the bonus task.

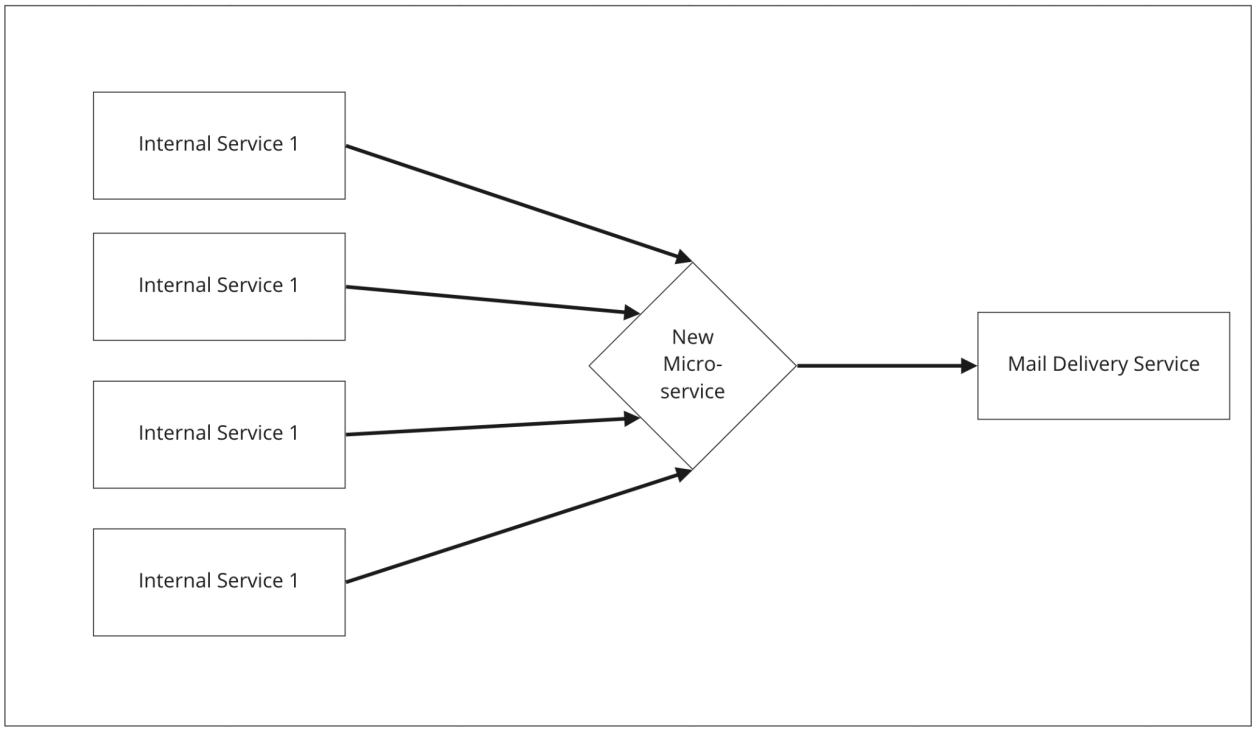
Introduction :

In our company, we send thousands of emails on a daily basis from dozens of internal services. In the current situation, we have a **mail delivery service*** integrated in all our internal services from where we send the emails.

The problem with this architecture is that it became very hard to keep track of the sending process of emails across all the internal services. Moreover, almost all the implementations (like the handling of failed sendings, tracking events) are duplicated in those services. So when the **mail delivery service** introduces a change in their api, we have to do the change multiple times, which is not just annoying but it costs our engineers a lot of time and effort.

So to provide a better solution, we thought about centralizing the sending of all the emails in a single microservice. Only this microservice will be directly connected with the **mail delivery service** and all the other internal services can send the emails only through it.

The illustration below explains the architecture we want to achieve.



The Mail Delivery Service:

The **mail delivery service** is a paid service (3rd party library) that is responsible for delivering an email to its recipient. Like postmark and mailgun.

It provides an endpoint which is *POST* ***/api/emails*** to synchronously submit emails for delivery. The delivery process can take some time and when it's done, the **mail delivery service** can report about the status of the delivery via a webhook.

The ***/api/emails*** is designed to accept a payload as json and looks like this:

```
{
        - request_id: an id for your request.
        - sender: the email address of the sender.
        - recipient: the email address of the receiver.
        - message: email body as HTML format.
}
```

and it responds with

```
{
        "STATUS": "ACCEPTED" | "ERROR"
}
```

And the right HTTP status code. (200 or 500).

In the Webhook call, the **mail delivery service** contains this information in the body.

```
{
        "ID": "The id we sent in the first request",
        "STATUS": "DELIVERED" | "REJECTED" | "FAILED"
}
```

Your task:

Your task is to implement the new microservice as illustrated above. The microservice must provide an API with 2 endpoints.

- **/send** : it accepts requests to send the emails and forward them to the mail delivery service for delivery. Every request must provide this payload as json:

```
{
        - sender: the email address of the sender.
        - recipient: the email address of the receiver.
        - message: email body as HTML format.
}
```

This endpoint must respond with a status and a generated request id to the consumer, that he can use for tracking the status of the request later.
The response should look like this:

```
{
        "status": "accepted",
        "request_id":"059298eb-31ca-48f9-a022-6702058d67e8"
}
```

or in case the request is invalid.

```
{
        "status": "denied",
        "reason": "missing recipients field"
}
```

- **/status** : this endpoint returns the status of a request by providing the request id generated in the first endpoint. The response has to look like this.

```
{
        "request_id": "059298eb-31ca-48f9-a022-6702058d67e8",
        "status": "sent" | "failed" | "processing"
}
```

From our experience with the mail service delivery, it is very usual that their server goes down and therefore some emails never get submitted for delivery. So make sure the new microservice handles this situation.

Bonus task:

Some emails (e.g security alerts and One Time Passwords) must be sent as fast as possible. So it would be a bonus if your solution would support prioritizing the sending of emails.

How to submit your solution?

You can submit your solution as a **zip file* or link to your github/gitlab repository** per email to the email address **anass.rakibi@gmail.com**

The subject of your email must be specified as follows.
"**Your Name - solution for 1st challenge February 2022**"

*** Please exclude any auto-generated files/folders from the zip file (ex: vendor)**

*** If you decide on a github/gitlab repository, please make sure you grant me read access.**

README File:

In order to simplify reviewing your solution, I strongly recommend you to include a "README.md" file at the top level of your project's folder. This file must contain the following information:

- **Name**: your name.
- **Email**: your email.
- **Description**: a short description of your solution.
- [optional] **Assumptions**: list of assumptions if you had to take any.
- [optional] **Architecture** diagram: a diagram of your design (example)
- [optional] **Project setup**: How to run your project.
- [optional] **Improvements**: what would you have added if you had more time.

You can rely on this example here
https://gitlab.com/moroccan-phpers/1st-challenge-ferbraury-2022/-/blob/master/README.md

Evaluation:

When evaluating your solution, we will consider these points:

- **Code Quality**: Clean, Flexible, Maintained, Reusability ...etc
- **Testing**: Is your implementation backed up with automated tests (unit/integration tests)
- **Efficiency**: does your solution really solve the problem?
- **Performance**: Is your solution's execution time fast? Does it consume less resources?
- **Scalability**: Does your solution fit a large use case?