Back-end developer test

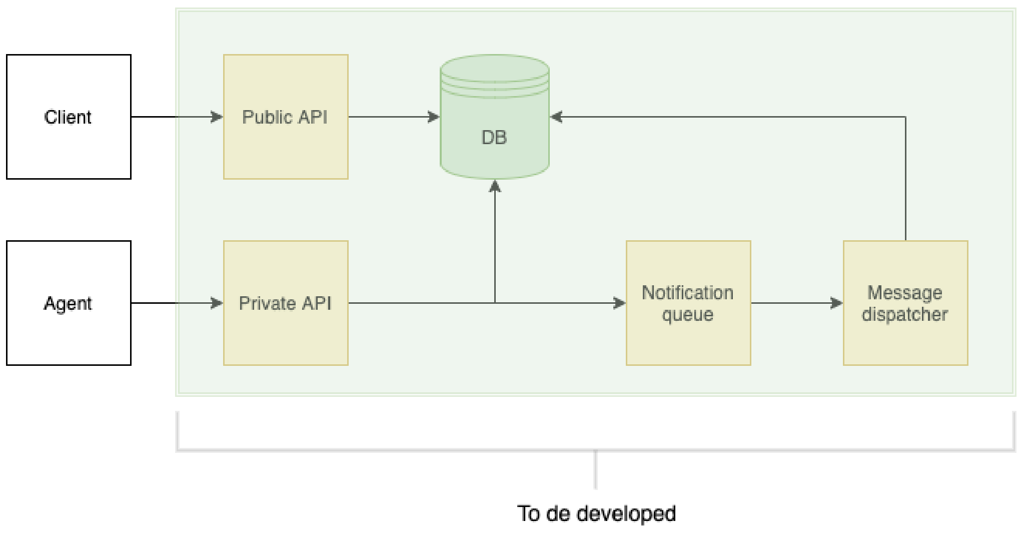
1 Description

Regular communication with customers is key for success. Notifications can help with retention, increasing the lifetime value of customers. However, with large customer base it is harder to handle notification process… Asynchronous processing comes to the rescue.

Your task is to create small solution/service for processing notifications, where you demonstrate your skill in creating web-services, interacting with Message Queues, managing data and applying best coding practices.

2 Details

1. Create public API which is able to provide CRUD operations for client. Clients must be stored in Database.
2. Create private API with authorization which is able to create/view notifications and view clients. There should be two notification channels – sms, email. Notifications must be stored in Database.
3. Once notification is sent status of notification should be updated. Sending process should happen asynchronously using Message Queue. (*no need to implement actual sender, just a stub*)



2.1 Public API details

Public REST API should provide following functionality:

- add client
- view client
- update client
- delete client

Create client payload example:

```
{
      "firstName": "John",
      "lastName": "Doe",
      "email": "john.doe@mail.com",
      "phoneNumber": "+37101234567"
}
```

Where input data validation should be applied:

- firstName: *only latin characters, with size 2-32*
- lastName*: only latin characters, with size 2-32*
- email: *valid email*
- phoneNumber: *valid phone number, compatible with E.164*

2.2 Private API details

Private REST API should provide following functionality:

- add one or multiple notifications
- get client
- get paginated list of clients
- get notification
- get paginated list of notifications, with possibility to filter notifications by client

Create notification payload example:

```
{
      "clientId": <client_id>,
      "channel": "sms",
      "content": "Lorem ipsum dolor sit amet, consectetur adipiscing elit.."
}
```

Where input data validation should be applied:

- clientId: *id of existing client*
- channel*: sms or email*
- content: *in case of sms, content shouldn't be longer then 140 characters*

3 Considerations

- PHP 7+ version should be used.
- Code styling and designing best practices. Think about daily code reviews which you will have to pass (and later about the code you will have to review). Do your best.
- Test coverage. Cover as much functionality as you can with tests.
- You can use any framework you familiar with.
- Use any relational database you familiar with.
- Use any message broker you familiar with.
- Please send your sources and brief description on how to launch/test your app.

4 Extra points

- Use Symfony or Zend framework.
- Use RabbitMQ as message broker
- Use PostgreSQL as database and Doctrine ORM.
- Use Swagger for API documentation.
- Dockerize your application.
- Use Mailhog or Gotify for sending messages