servicepad

# Backend Developer Technical Test

## Logic test

Delivery time: **4 Days** after sending the test

1. Code a program (in python) that displays the numbers from 1 to 100 on the screen, substituting the multiples of 3 for the word "fizz", the multiples of 5 for "buzz" and the multiples of both, that is, the multiples of 3 and 5, by the word "fizz buzz".

   Plus: Implement a test to the task performed (Preferably with UnitTest or pytest).

2. In mathematics, the Fibonacci sequence or serie is the following infinite sequence of natural numbers: 0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597 ...
   where f (0) = 0, f (1) = 1 and f (n) = f (n-1) + f (n-2).
   Code a program (in python) that solves for any number in the Fibonacci series.

   Plus: Implement a test to the task performed (Preferably with UnitTest or Pytest).

3. Given an input text, Code a program (in python) that displays the number of repetitions of each word.
   Sample text: "Hi how are things? How are you? Are you a developer? I am also a developer"

   Plus: Implement a test to the task performed (Preferably with UnitTest or Pytest).

servicepad

# Backend knowledge application test

Delivery time: **4 Days** after sending the test

As a backend developer, you should be able to create REST APIs, implementing their respective actions, as well as be able to perform a user authentication method and the ability to document it. This test requires that you use the Python programming language to perform it.

Instructions:

1. Create a Rest API, which allows authorized users to manage their own Publications (filtering by current user): Create, Read, Update and Delete
2. Create the unit tests to check the correct functioning of the methods offered by the API
3. Document the API using a Swagger-like tool.
4. Publish the code to a public repository. For review and evaluation.
5. Please make installation and running instructions in the README.md.

It should include:

- User authentication mechanism (login, logout).
- User management CRUD (fields: email, password, fullname, photo).
- Persistence of data in a Relational database (Postgres is preferred).
- Publication management CRUD (fields: title, description, priority, status, time since it has been published (seconds, minutes, hours, days, etc.), user, created_at, updated_at) and view details of a publication.
- Units tests must be created with **Pytest**
- The API must be built with **Flask**
- Document relevant methods used
- Use of schemas (ex.: marshmallow) is ideal, but not mandatory
- Must create a README file with instructions to execute your API
- The API should be testable with Swagger
- The source code must have the requirements.txt file