Prueba Técnica - Programador Back-end

El presente documento define un ejercicio práctico y las condiciones de evaluación para los postulantes a ***programador Back-end*** de **Osana Salud**.

Introducción

Ésta guía contiene una serie de requerimientos de un Caso Práctico, que busca evaluar las capacidades técnicas del candidato con respecto a las principales funciones y responsabilidades que se requieren dentro del área de Desarrollo de Tecnología de ***Osana Salud***.

Modo de evaluación

El desafío está pensado para resolverse en un máximo de **3 (tres) horas** por lo que recomendamos emplear como máximo ese tiempo y enviar todo lo que pueda para su evaluación.

Una vez realizada la entrega, coordinaremos un llamado para analizar algunos aspectos sobre la metodología de trabajo, entre otros.

Haremos especial hincapié en lo siguiente:

- Creatividad para resolver los requerimientos
- Metodología de trabajo
- Calidad del código (estructura y buenas prácticas)
- Conocimiento de "librerías" y el ecosistema.

Notas

- Antes de comenzar a programar
    - Realizar un `fork` de este repositorio.
    - Crear un `branch` en su `fork` (a partir de la rama `develop` de este repositorio) utilizando su nombre de usuario.
- Al finalizar
    - Crear un `pull-request` a la rama `master` de este repositorio, desde el `branch` con su nombre de usuario.

Ejercicio práctico

Contexto y objetivos

Osana Salud tiene dos fuentes de datos de usuarios que debe centralizar en un único servicio para uso interno. Optamos por el desarrollo de una API ReST la cual debe abstraer, en esta etapa, las dos fuentes de datos siguientes:

- Un par de archivos CSV donde uno contiene la información de la cuenta de usuario, y el otro los datos del perfil. Ambos están relacionados mediante el ID del usuario (ver los archivos `data/users.csv` y

`data/profiles.csv` en la raíz del proyecto).

- El API ReST pública de (GitHub)[https://docs.github.com/en/free-pro-team@latest/rest]

El proyecto está planteado en PHP7 haciendo uso de su ecosistema mediante el gestor de paquetes (Composer)[https://getcomposer.org/] y se encuentra en un estado avanzado de desarrollo.

El modelo de datos con el que vamos a trabajar, se corresponde con el siguiente en formato JSON:

```
{
    "id": "",
    "login": "",
    "type": "",
    "profile": {
        "name": "",
        "company": "",
        "location": ""
    }
}
```

Cabe aclarar que este modelado ya se encuentra implementado, y solo se debe asegurar que `type` se corresponda con el servicio del cual proviene (`local` o `github`).

Además, el proyecto se encuentra configurado para trabajar con el archivo `.env` en la raíz del mismo, donde se definen algunos parámetros del API. Para comprender mejor estas configuraciones, revisar el archivo `env.example`.

Requerimientos generales

Información y version del API

```
GET / HTTP/1.1
Content-Type: application/json
```

Respuesta de ejemplo

```
HTTP/1.1 200 OK

{
    "name": "Osana Salud",
    "version": "1.0"
}
```

> NOTA: `name` debe cohincidir con su nombre de usuario en GitHub.

Búsqueda de usuarios

Consulta de ejemplo

```
GET /users HTTP/1.1
Content-Type: application/json

{
    "q": "....",
    "limit": 10
}
```

Campo	Tipo	Requerido	Descripción
`q`	String	No	Patrón de búsqueda para el campo `login` del modelo
`limit`	Number	No	Límite de resultados. En caso de no indicarse, se asume un 20 por defecto

> NOTA: El valor asignado a `q` corresponde al prefijo de `login` para la búsqueda. Es decir, si se envía un `q = osa` debe devolver todos los registros de ambas fuentes cuyo campo `login` inicien con `osa`. En caso de que no se indique `q`, se optará por los registros en orden de aparición de cada fuente.

> NOTA2: Como la búsqueda debe realizarse en ambas fuentes, se optará por intentar obtener 50% de cada una. Es decir, si se solicitaran 10 resultados como límite, se deberá intentar obtener 5 registros de cada fuente. En caso de que no fuese posible, se intentará llegar al límite dándole mayor prioridad a la fuente local.

Respuesta de ejemplo

```
HTTP/1.1 200 OK

[
    {
        "id": "1",
        "login": "mojombo",
        "type": "github",
        "profile": {
            "name": "Tom Preston-Werner",
            "company": "@chatterbugapp, @redwoodjs, @preston-werner-
ventures ",
            "location": "San Francisco"
        }
    },
    {
        "id": "CSV1",
        "login": "quam",
        "type": "local",
        "profile": {
            "name": "Tom Preston-Werner",
            "company": "Dolor Limited",
```

```
            "location": "Castelverete in Val Fortore"
        }
    }
]
```

Mostrar un usuario

Consulta de ejemplo

```
GET /users/{service}/{login} HTTP/1.1
Content-Type: application/json
```

Campo	Tipo	Requerido	Descripción
`service`	String (ruta)	Si	Nombre del orígen de datos. Puede ser `local` o `github`.
`login`	String (ruta)	Si	El nombre de usuario de búsqueda

Respuesta de ejemplo

```
HTTP/1.1 200 OK

[
    {
        "id": "1",
        "login": "mojombo",
        "type": "github",
        "profile": {
            "name": "Tom Preston-Werner",
            "company": "@chatterbugapp, @redwoodjs, @preston-werner-
ventures ",
            "location": "San Francisco"
        }
    }
]
```

Guardar un nuevo usuario

Consulta de ejemplo

```
POST /users HTTP/1.1
Content-Type: application/json

{
    {
```

```
        "login": "hugo",
        "profile": {
            "name": "Hugo Martín",
            "company": "Osana Salud",
            "location": "Argentina"
        }
    }
}
```

Campo	Tipo	Requerido	Descripción
`login`	String	Si	Nombre de inicio de sesión del usuario
`profile.name`	String	Si	Nombre completo del usuario
`profile.company`	String	Si	Nombre de la compañia donde trabaja el usuario
`profile.location`	String	Si	Ciudad o país de residencia del usuario

Respuesta de ejemplo

```
HTTP/1.1 201 Created

[
    {
        "id": "CSV101",
        "login": "hugo",
        "type": "local",
        "profile": {
            "name": "Hugo Martín",
            "company": "Osana Salud",
            "location": "Argentina"
        }
    }
]
```

> NOTA: Solo es posible guardar los datos en la fuente local, por lo que deberá respetarse el orden ascendente en los IDs y mantener ambas tablas actualizadas.

Extras

La siguiente lista de puntos extra, no son necesarios pero serán muy valorados si se incluyen:

- Manejo de errores/excepciones con respuestas adecuadas
- Implementación de pruebas unitarias y funcionales