Prueba candidatos Backend (PHP-Laravel)

La prueba consiste en desarrollar una API Rest con 4 endpoints.

Los endpoints son:

- Generar access
  token POST /auth
  Solicitud:

```
{
  "username": "tester",
  "password": "PASSWORD"
}
```

  Respuesta:200
  OK

```
{
  "meta": { "success":
    true,"errors": []
  },
  "data": {
    "token": "TOOOOOKEN",
    "minutes_to_expire": 1440
  }
}
```

  401 Unauthorized

```
{
  "meta": { "success":
    false,"errors": [
      "Password incorrect for: tester"
    ]
  }
}
```

- Crear
  candidato
  POST /lead
  Solicitud:

```
{
  "name": "Mi candidato",
  "source": "Fotocasa",
  "owner": 2
}
```

Respuesta:201
OK


```
{
  "meta": { "success":
    true,"errors": []
  },
  "data": { "id":
    "1",
    "name": "Mi candidato",
    "source": "Fotocasa",
    "owner": 2,
    "created_at": "2020-09-01 16:16:16",
    "created_by": 1
  }
}
```


401 Unauthorized


```
{
  "meta": { "success":
    false,"errors": [
      "Token expired"
    ]
  }
}
```


- Obtener candidatoGET
  /lead/{id}
  Para obtener un candidato con un id en concreto

  Respuesta:200
  OK


```
{
  "meta": { "success":
    true,"errors": []
  },
  "data": { "id":
    "1",
    "name": "Mi candidato",
    "source": "Fotocasa",
    "owner": 2,
    "created_at": "2020-09-01 16:16:16",
    "created_by": 1
  }
}
```


  401 Unauthorized


```
{
```

```
  "meta": { "success":
    false,"errors": [

      "Token expired"
    ]
  }
}
```

404 Not found

```
{
  "meta": { "success":
    false,"errors": [
      "No lead found"
    ]
  }
}
```

- Obtener todos los candidatosGET
  /leads
  Devuelve todos los candidatos asignados al agente o si es usuario managerdevuelve todas los candidatos.

  Respuesta:200
  OK

```
{
  "meta": { "success":
    true,"errors": []
  },
  "data": [
    {
      "id": "1",
      "name": "Mi candidato",
      "source": "Fotocasa",
      "owner": 2,
      "created_at": "2020-09-01 16:16:16",
      "created_by": 1
    },
    {
      "id": "2",
      "name": "Mi candidato 2",
      "source": "Habitaclia",
      "owner": 2,
      "created_at": "2020-09-01 16:16:16",
      "created_by": 1
    }
  ]
}
```

401 Unauthorized

```
{
  "meta": { "success":
    false,

    "errors": [ "Token
      expired"
    ]
  }
}
```

A tener en cuenta:
Hay 2 tipos de usuarios: agent y manager.
Los usuarios con rol **manager** tienen permiso para crear y obtener todos los candidatos. Los usuarios con rol **agent** solo pueden obtener los candidatos que tienen asignados (owner). No pueden crear candidatos.
Los token caducan cada 24 horas, el token se tiene que enviar en los headers en las 3 llamadas de crear y obtener.

Modelo de datos:
Usuario:

```
{
  "id": "1",
  "username": "tester",
  "password": "PASSWORD",
  "last_login": "2020-09-01 16:16:16",
  "is_active": true,
  "role": "manager" //manager o agent
}
```

Candidato:

```
{
  "id": "1",
  "name": "Mi candidato",
  "source": "Fotocasa",
  "owner": 2, //Id usuario responsable
  "created_at": "2020-09-01 16:16:16",
  "created_by": 1 //Id usuario que ha creado el candidato
}
```

Requisitos:

Utilizar Laravel 9 o 10, BBDD MySQL y Redis para caché, JWT para el token,
Crear test unitarios
Crear factories de todos los modelos
Seeder para crear usuario con los 2 roles
Buenas prácticas de programación en base al estándar de la comunidad de Laravel!

Extra
Utilizar caché para obtener los candidatos
Implementar patrón repository
Form Request
Manejo de excepción
Eloquent Api Resource
Cobertura 100% de unit testing
Utilizar Sonarqube para el analisis del código estatico

A entregar:
Código con el .env.exemple listo para copiar a .env