15/04/2024

Examen Junior nuevos OPA S.A.S

Luis Camilo Valencia

RAZONAMIENTO LÓGICO:

1.a) IHG:

Ya que son las letras del abecedario desde la d hasta la r, agrupadas por 3 unidades y de atrás hacia adelante.

2.b) OFN:

Cada letra del grupo de 3 letras va convirtiéndose en su siguiente letra en orden alfabético en el siguiente grupo de 3 letras, por ende, pude deducir la respuesta.

3.c) 139:

Podemos calcular la siguiente cifra así:

```
cifra = cifra_anterior + (5 + i)
```

Donde i comienza valiendo 0 y partimos del número 104, ósea, el siguiente de 104 es:

```
cifra = 104 + (5 + 0)
```

lo que da 109, para la siguiente cifra es igual pero i va sumando de a 1 cada vez, lo que da 115 y así sucesivamente hasta llegar a 139 que era la cifra que venía al orden.

4.c) 511:

Aquí la siguiente cifra se puede calcular así:

```
cifra = (cifra_anterior * 2) + 1
```

Lo que me permitió hallar el 511 que era la cifra que seguía porque 511 es:

```
cifra = (255 * 2) + 1
```

ARBOL GENEALÓGICO

B) 1 y 3:

Leonor es nieta de Toño y bisnieta de Andrés, Toño es tío de Celso e hijo de Andrés.

PRUEBA TÉCNICA

Para la prueba técnica la realicé de dos maneras, la primera es un programa usando el código Ruby el cual está en la carpeta "challenges/challenge_OPA" (https://github.com/lcamilov7/challenges/tree/master/ChallengeOpa), aquí hay que correr el archivo "inicio.rb", que dará inicio al programa y la interacción con este será por consola, se asegura de obtener números mayores a 0, que el número mayor sea más grande que el menor y que no sean decimales. Luego imprime en consola los resultados obtenidos como números naturales perfectos dentro del rango recibido por entrada. Esta solución fue más que todo para realizar la lógica del ejercicio.

La segunda solución del problema fue una aplicación web que se encuentra en la carpeta llamada "naturales-perfectos" (https://github.com/lcamilov7/naturales-perfectos) desarrollada con las tecnologías: Ruby on Rails como framework web, Bootstrap como framework de CSS para el frontend, postgreSQL como base de datos y GitHub como controlador de versiones (github.com/lcamilov7). Se siguió el patrón de arquitectura MVC. El código está comentado para su fácil entendimiento y escalabilidad.

Para correr la aplicación correctamente necesitamos:

1. Una terminal para ejecutar comandos
2. Un navegador para visualizar el archivo HTML que obtenemos al realizar peticiones recibidas desde el servidor que levantaremos en la terminal.
3. Tener instalado Ruby on Rails, a continuación, como instalarlo en 3 sistemas operativos diferentes:
    - Mac: https://www.youtube.com/watch?v=OzhawTQSjGs
    - Windows: https://www.youtube.com/watch?v=Byjqlgatpr4
    - Linux: https://www.youtube.com/watch?v=CepoXqIlDeM
4. En la terminal creamos la base de datos con rails 'db:create'
5. Corremos las migraciones con rails 'db:migrate'
6. Instalamos gemas y dependencias necesarias para el funcionamiento con 'bundle install'
7. Levantamos el servidor en el puerto localhost:3000 con 'rails s'
8. Abrimos en el navegador el localhost:3000

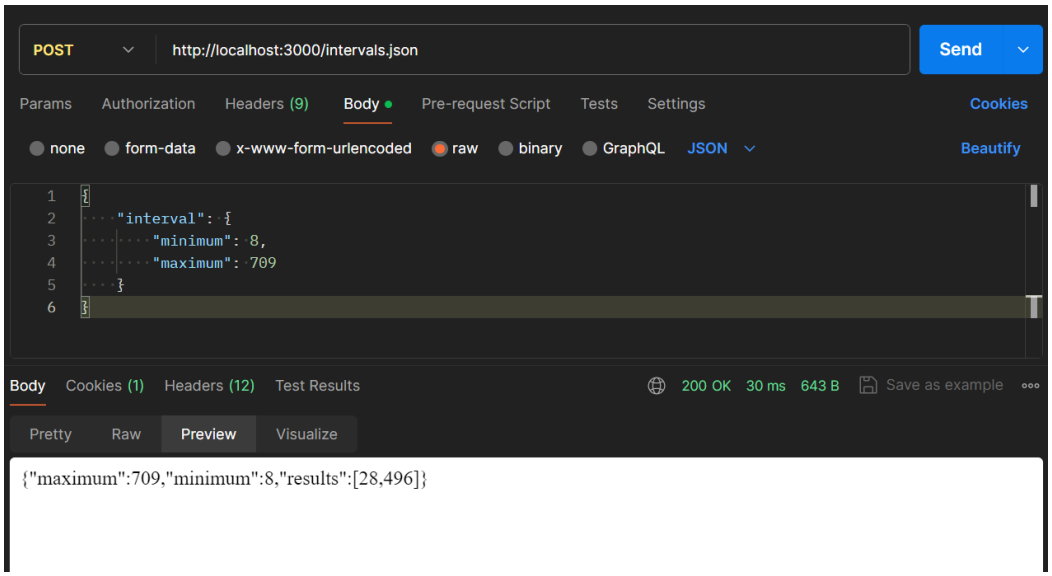
La aplicación comienza con una interfaz amigable e intuitiva que consiste en un form que pide los datos mínimo y máximo del rango, luego, valida que los datos ingresados sean válidos, que el mínimo sea mayor a 0, que el máximo sea mayor al mínimo y que no pueda quedar ningún dato vacío, una vez ingresados datos válidos, redirecciona y

muestra una lista de los números naturales perfectos del rango dado y da la opción de probar nuevamente con otros números.

Si se desea, la aplicación puede usarse como un backend que devuelve un json que contiene los datos minimum(integer), maximum(integer) y los naturales perfectos entre ambos números, results(array), así puede ser consumida por otro frontend distinto si se desea, añadiéndole ese plus de interoperabilidad que ofrece trabajar con json.

Para hacer una petición que devuelva el json anteriormente mencionado podemos usar herramientas como Postman:

1. Creamos una nueva petición de tipo POST
2. Ingresamos la URL http://localhost:3000/intervals.json
3. En el body de la petición establecemos el tipo JSON y en el campo raw pasamos los atributos minimum y maximum en formato json asi:



Nota importante: Fue un desafío interesante en el que pude practicar tanto mi lógica como programador como mi capacidad para construir una aplicación que resuelva los requerimientos solicitados en un periodo corto de tiempo de dos días, decidí realizarla en el medio que más confianza y práctica tengo que es el desarrollo web con el framework Ruby on Rails, aunque también tengo práctica y actualmente profundizo en Python, pero estoy dispuesto a aprender nuevos lenguajes y frameworks de desarrollo de aplicaciones utilizando las mejores prácticas y patrones de diseño, que puedan ser desafiantes y me hagan crecer como profesional que aborda variedad de tecnologías.