Los métodos utilizados son:

POST <http://localhost:8080/demo/api/consulta>

GET <http://localhost:8080/demo/api/personas>

GET <http://localhost:8080/demo/api/personas/id>

PUT <http://localhost:8080/demo/api/personas/id>

**//@PostMapping("/consulta"):**

Esta anotación indica que este método manejará solicitudes HTTP POST en la ruta /api/consulta.

//En otras palabras, este método será invocado cuando se realice una solicitud POST a esa ruta específica.

**private final AtomicLong idGenerator = new AtomicLong(1);**

La línea de código **private final AtomicLong idGenerator = new AtomicLong(1);** inicializa una variable llamada **idGenerator** como un objeto de tipo **AtomicLong** con un valor inicial de 1. Aquí hay una explicación de los elementos clave:

AtomicLong: Es una clase en Java que proporciona operaciones atómicas para variables de tipo long. Las operaciones atómicas son aquellas que se ejecutan como una sola unidad ininterrumpida, sin posibilidad de que otras operaciones se intercalen. En este caso, se utiliza para garantizar que la generación de IDs sea segura en entornos de concurrencia.

private final: Establece que esta variable es privada y final, lo que significa que no se puede modificar después de su inicialización. La finalidad de declararla final es garantizar que el objeto AtomicLong no se pueda cambiar después de su creación.

= new AtomicLong(1): Crea una nueva instancia de AtomicLong con un valor inicial de 1. Este será el valor inicial del generador de IDs.

En el contexto del código que proporcionaste, este idGenerator se utiliza para asignar automáticamente IDs incrementales a medida que se agregan nuevas personas a la lista. Cuando se necesita un nuevo ID, se llama a idGenerator.getAndIncrement(), que obtiene el valor actual del AtomicLong y luego incrementa su valor en uno de manera atómica, asegurando que ningún otro hilo pueda interferir y obtener el mismo ID simultáneamente. Esto es útil en entornos multihilo o concurrentes para evitar problemas de concurrencia al generar IDs únicos.

**@GetMapping("/personas/{id}")**

he agregado un nuevo endpoint **@GetMapping("/personas/{id}")** que permite obtener una persona por su ID. La ID se obtiene de la URL mediante la anotación **@PathVariable**. Se utiliza **Optional** para manejar el caso en que no se encuentre la persona con la ID proporcionada. Si se encuentra la persona, se devuelve como parte de un **ResponseEntity**. Si no se encuentra, se devuelve un **HttpStatus.NOT\_FOUND**.