Implementation

The class MessageController was created as a @RestController that mappes requests of the path /api/message via
@RequestMapping. The class holds the two private variables apiMessageDefault and currentApiMessage.

```
@RestController
@RequestMapping("/api/message")
public class MessageController {

    private String apiMessageDefault = "Status Ok";
    private String currentApiMessage;
```

feature/api-message

The `message` feature returns either the current service message or the default service message, if the `current` message hasn't been set.

```
@GetMapping("")
    public ResponseEntity<String> Message() {
        if(currentApiMessage == null) {
            if (apiMessageDefault.isBlank()) {
                return new ResponseEntity<>("No default message or message set",
HttpStatus.INTERNAL_SERVER_ERROR);
                }
            return new ResponseEntity<>(apiMessageDefault, HttpStatus.OK);
            }
        return new ResponseEntity<>(currentApiMessage, HttpStatus.OK);
    }
```

feature/api-message-set

The `set` feature is supposed to **set** the current message of the server to a provided message.

```
 @GetMapping("/set")
    public ResponseEntity<String> SetMessage(@RequestParam(name="m",
required=true) String msg){
        currentApiMessage = msg;
        return new ResponseEntity<>(currentApiMessage, HttpStatus.OK);
    }
```

feature/api-message-reset

The `reset` feature is supposed to **reset** the current message to a default message of the server.

```java
@GetMapping("/reset")
    public ResponseEntity<String> MessageReset() {
        if(apiMessageDefault.isBlank()) {
            return new ResponseEntity<>("Default message is not set.", 
HttpStatus.INTERNAL_SERVER_ERROR);
        }
        currentApiMessage = apiMessageDefault;
        return new ResponseEntity<>(currentApiMessage, HttpStatus.OK);
    }
```