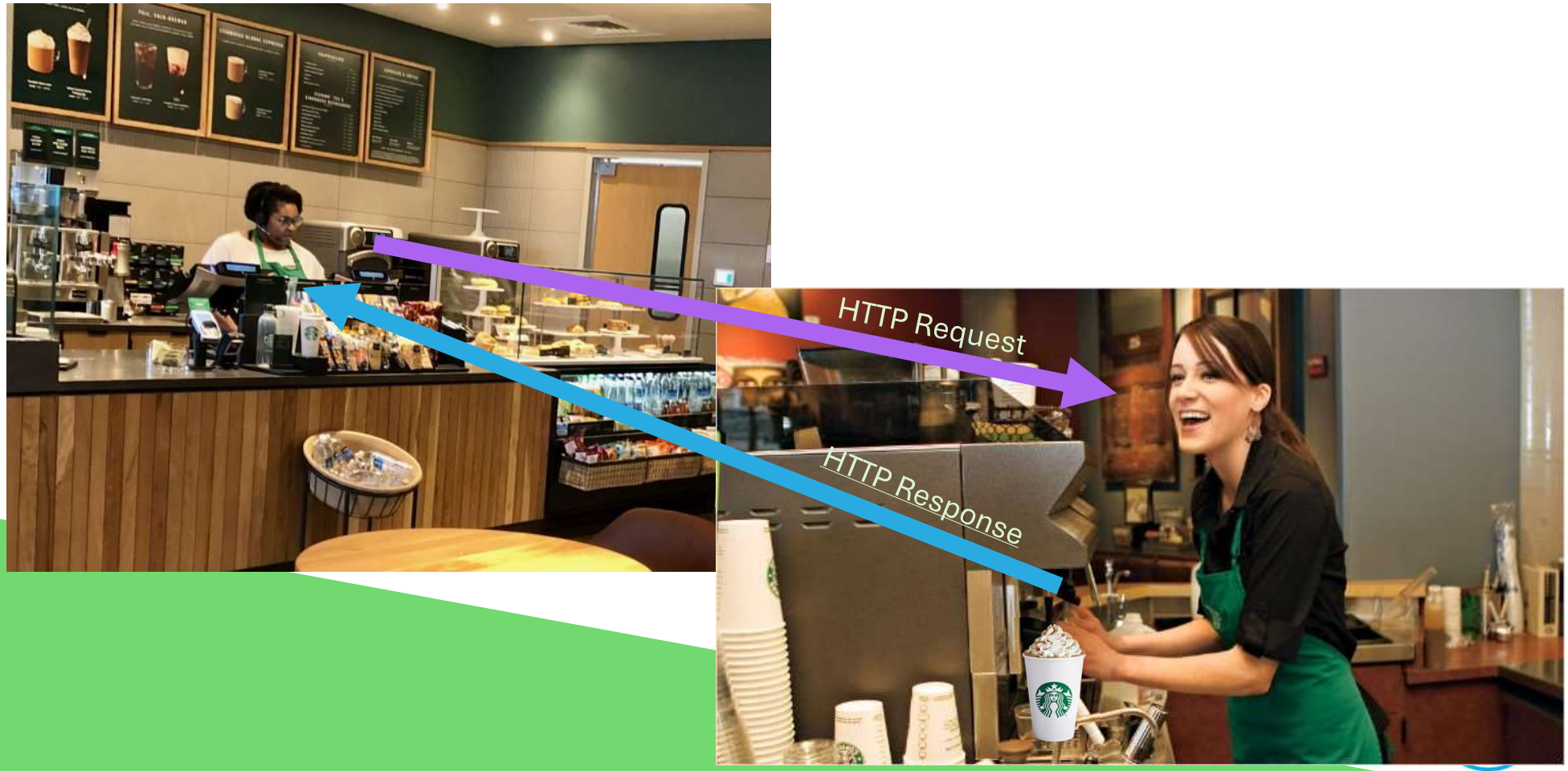
MODULE 2

# Review – Web Services

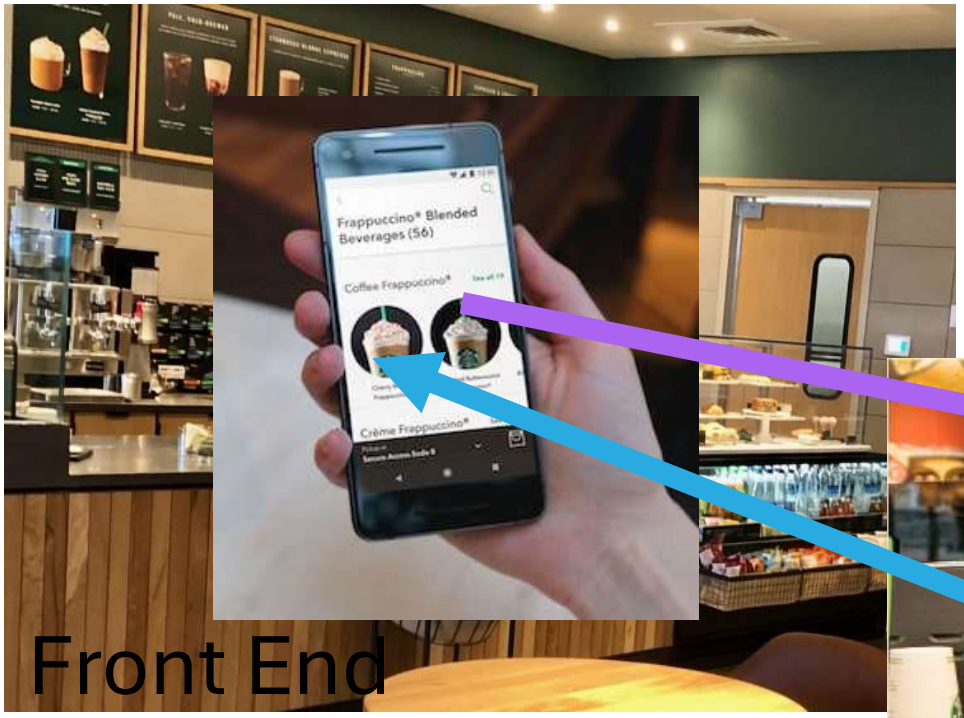
TECH ELEVATOR
A Stride Company

Let’s get some coffee!

# Coffee process



# Coffee process



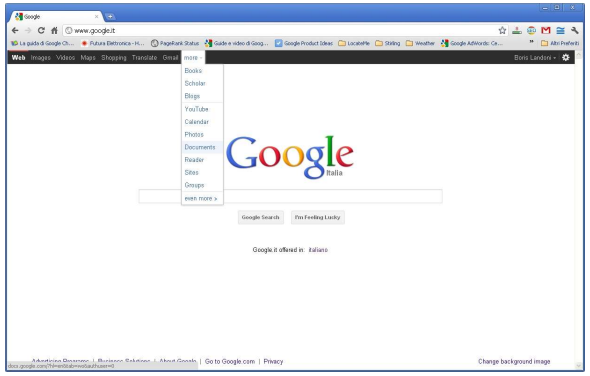

## Back End

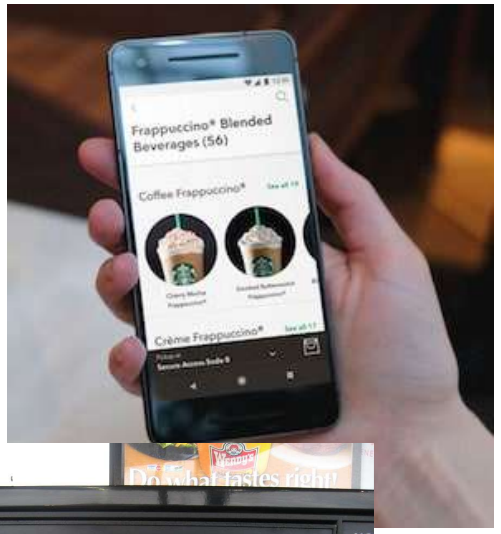

```
{
    "size":"venti"
    "description":"mocha"
    "whippedcream":true
}
```

ELEVATE YOURSELF

# Front Ends – You've used

- Web Browser
- Mobile App
- ATM Machine
- Sheetz MTO Kiosk
- Gas Pump



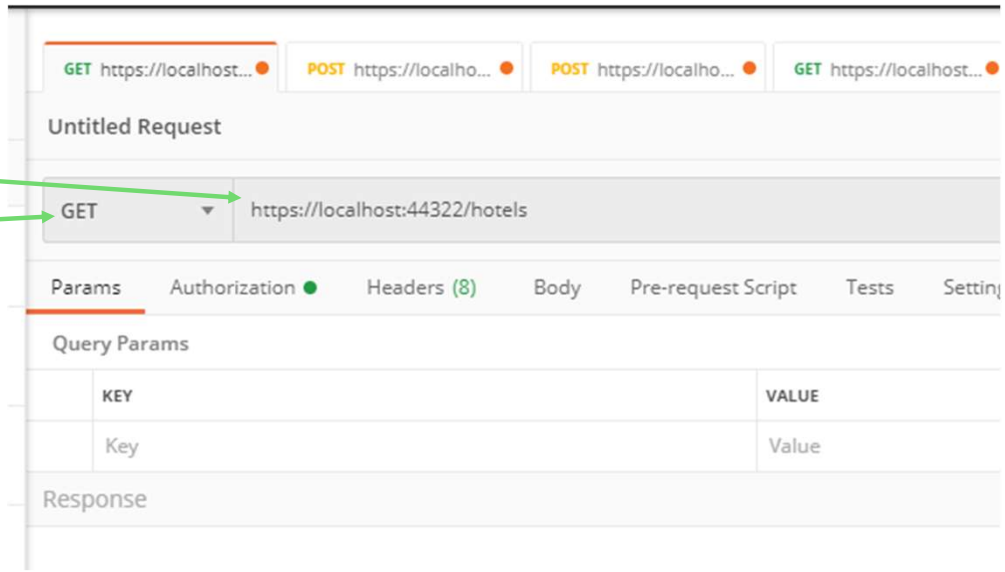


# Request Needs

- Reference to Back End
  - URL (end point)
- Method (or verb)
  - HttpGet – retrieve information
  - HttpPost – add information
  - HttpPut – update information
  - HttpDelete – delete information
- How to talk to the back end
  - Client

# Client Needs – Postman

- Reference to Back End
  - URL (end point)
- Method (or verb)
  - HttpGet – retrieve information
  - HttpPost – add information
  - HttpPut – update information
  - HttpDelete – delete information
- How to talk to the back end
  - Client




# Consumer of API– Code

- How to talk to the back end
  - Client
- Reference to Back End
  - URL (end point)
- Method (or verb)
  - HttpGet – retrieve information

- RestClient
  - get() *sets up a GET request*
    .uri(String) *sets the endpoint*
    .retrieve() *sends the GET request*
    .body(Class) *extracts the Body*

Hotel reservation server

- https://te-pgh-api.azurewebsites.net/api/hotels

# RestClient Methods

```
restClient.get()
        .uri("/reviews")
        .retrieve()
        .body(Review[].class);
restClient.post()
         .uri("/reservations")
         .contentType(MediaType.APPLICATION_JSON)
         .body(newReservation)
         .retrieve()
         .body(Reservation.class);
```

```
restClient.put()
        .uri("/reservations/" + updatedReservation.getId())
        .contentType(MediaType.APPLICATION_JSON)
        .body(updatedReservation)
        .retrieve()
        .toBodilessEntity();
```

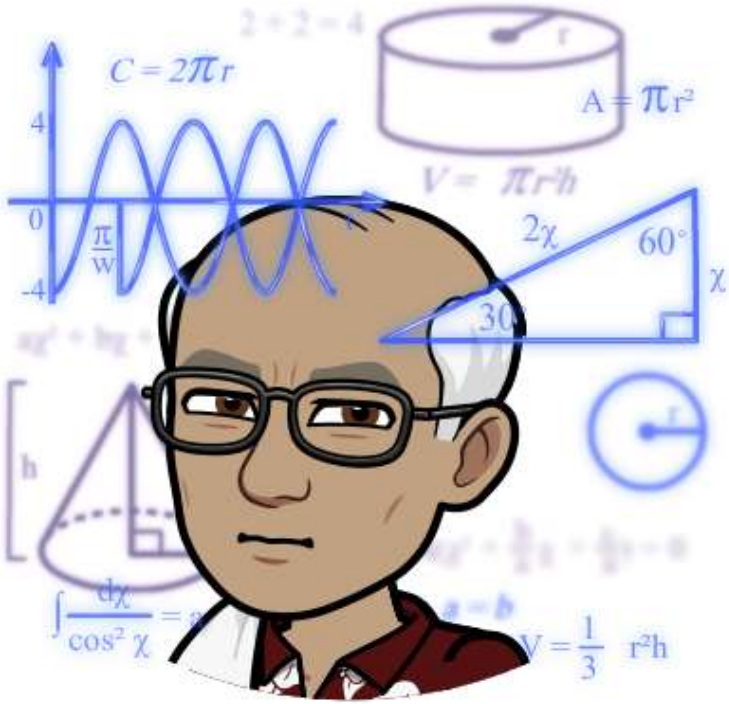
```
restClient.delete()
        .uri("/reservations/" + id)
        .retrieve()
         .toBodilessEntity();
```

# Back End Needs – Process the Request

Instantiate Controller

Execute Action

Return Data



# Back End Needs

- Instantiate Controller
  - `https://localhost:44322/`**`hotels`**
- Execute Action
  - Determined by the method and the route

  **HttpGet** -- `https://localhost:44322/hotels`
- Return Data
  - Send back JSON

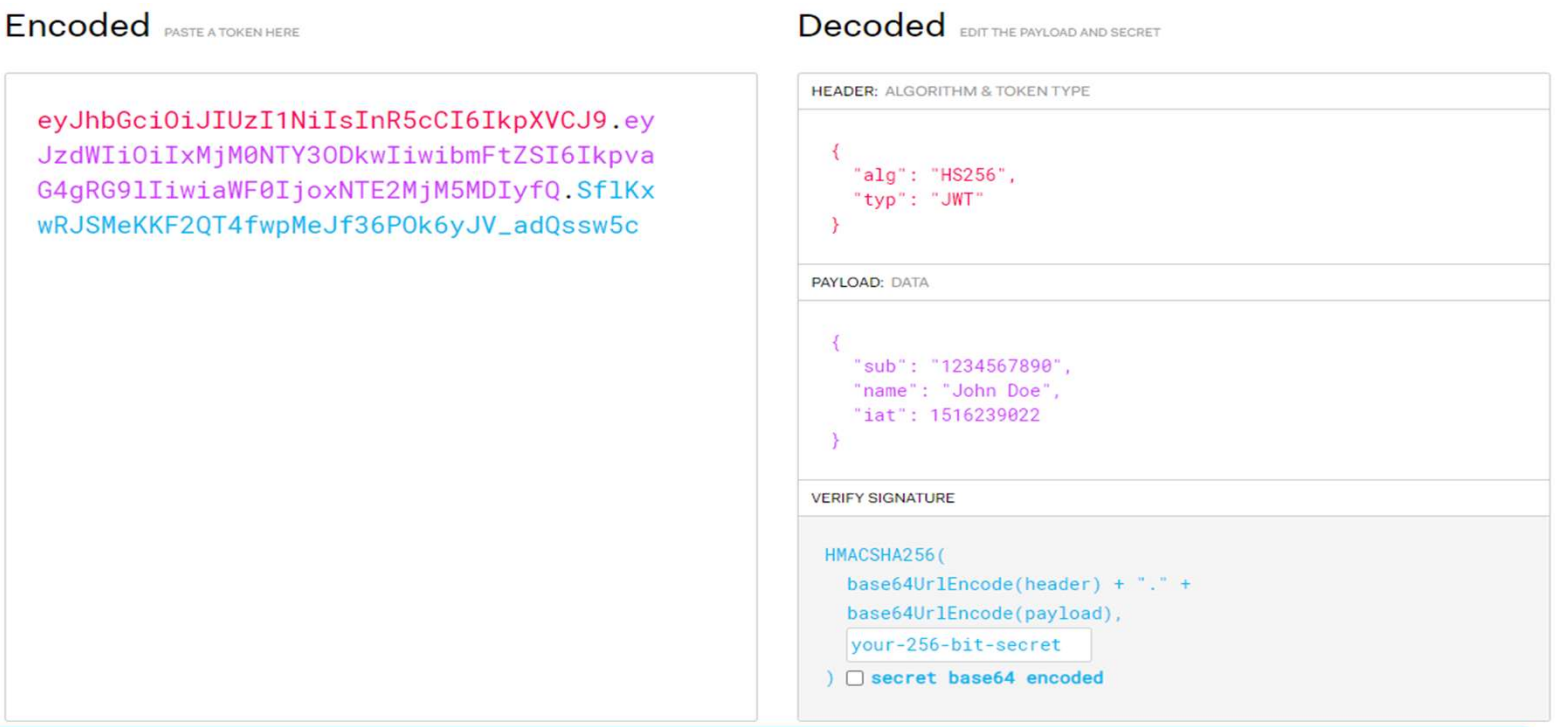
# Back End Needs

- Instantiate Controller
  - `https://localhost:44322/`**`hotels`**
- Execute Action
  - Determined by the method and the route
  - **HttpGet** -- `https://localhost:44322/hotels`
- Return Data
  - Send back JSON

```
@RequestMapping(path = "/hotels", method = RequestMethod.GET)
  public List<Hotel> list() {
    return hotelDao.list();
  }
```

# JW-what?

- JWT – JSON Web Token
  - A way of remembering a user has logged in.



# JWT Client Side



Need to get the token from the server (login)



Store the token for future requests



Add the token to the HttpRequest Headers

```
headers.setBearerAuth(authToken);
```



Send using restTemplate.exchange

Or other methods that allow the adding of the entity object

# JWT Server Side

- @PreAuthorize annotation tells Spring to:
  - Look for JWT
  - If it is there, decode it.
  - Apply the PreAuthorize rule
  - If that passes, continue executing the code

    - @PreAuthorize(“isAuthenticated()”)
    - @PreAuthorize(“hasRole(‘admin’)”)
    - @PreAuthorize(“hasAnyRole(‘admin’,’student’)”)
    - @PreAuthorize(“isAnonymous()”)
    - @PreAuthorize(“permitAll”)

# LET’S CODE!