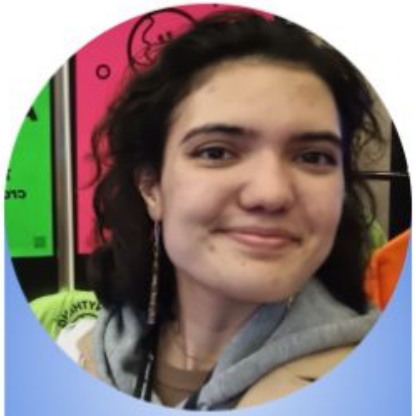
NotMyFitnessPal

Aaron, Marcy, Nasir, Sarina, Suraaj

The Team

Marcy Calderon

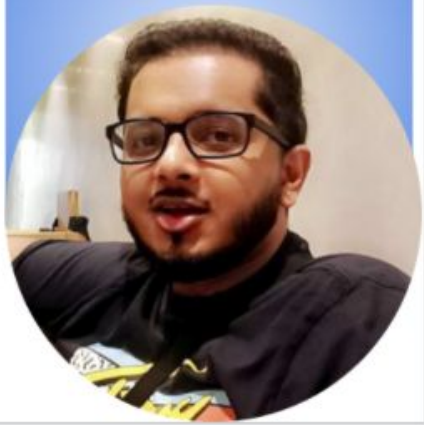


Sarina Salamon

Aaron Nazareth

Mohammed Nasir

Suraaj Leal



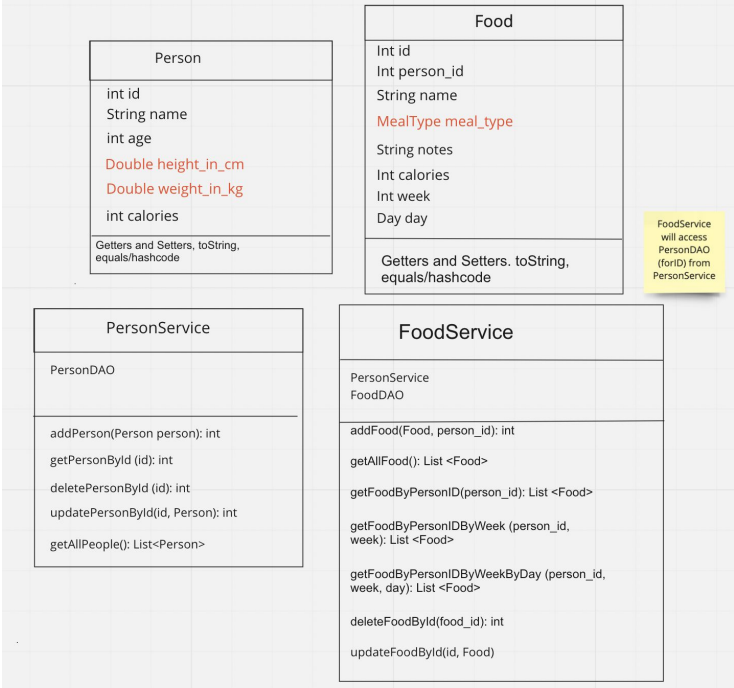
Introducing **Not**MyFitnessPal!

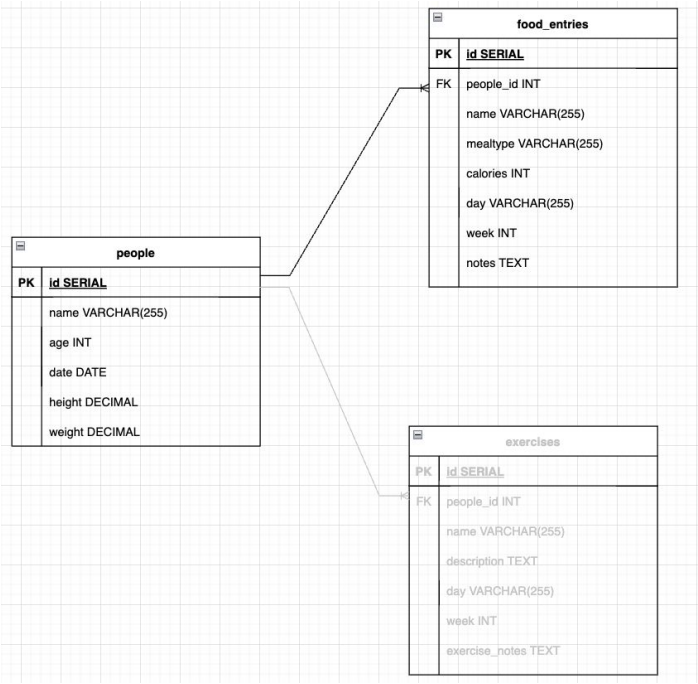
- A food tracking API built using **Java, SQL, Spring Boot**
  - Get stored food from a database
  - Update stored food items
  - Delete food items

- Able to select food by...
  - Person_id
  - Week and day
  - Meal type

Planning - ERD, Class Diagram

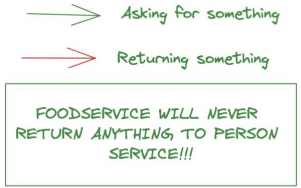
- Friday - POJOs, services, methods, ERD

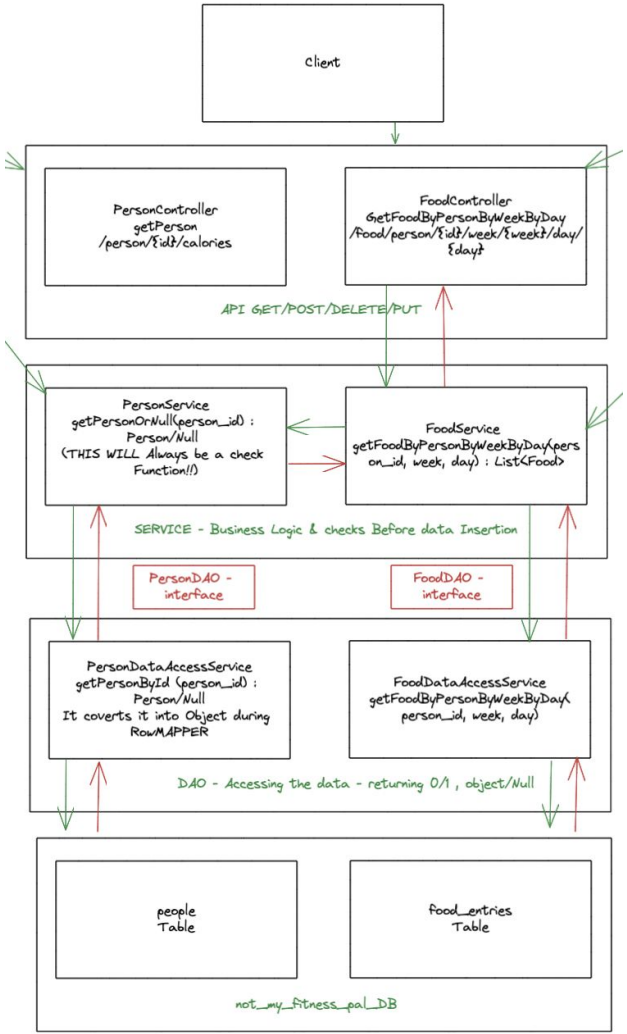


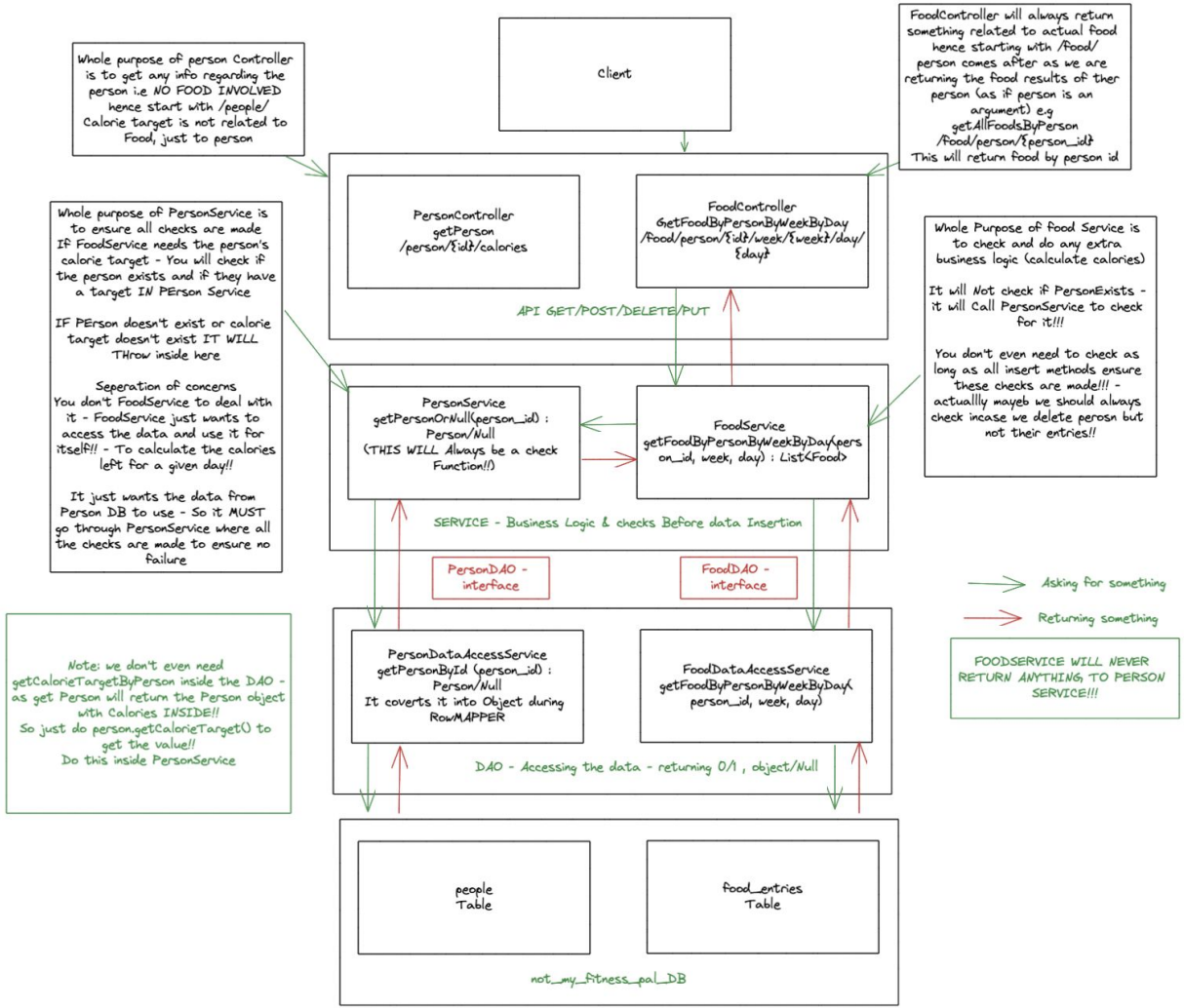


Planning - Application Structure

- Sunday
- Food and Person
- Controller, Service, DAOs
- Connection via service classes






Client
Whole purpose of person Controller is to get any info regarding the person i.e NO FOOD INVOLVED hence start with /people/ Calorie target is not related to Food, just to person
FoodController will always return something related to actual food hence starting with /food/ person comes after as we are returning the food results of ther person (as if person is an argument) e.g getAllFoodsByPerson /food/person/{person_id} This will return food by person id
PersonController getPerson /person/{id}/calories
FoodController GetFoodByPersonByWeekByDay /food/person/{id}/week/{week}/day/{day}
API GET/POST/DELETE/PUT
Whole purpose of PersonService is to ensure all checks are made If FoodService needs the person's calorie target - You will check if the person exists and if they have a target IN PEr son Service
IF PEr son doesn't exist or calorie target doesn't exist IT WILL THrow inside here
Seperation of concerns You don't FoodService to deal with it - FoodService just wants to access the data and use it for itself!! - To calculate the calories left for a given day!!
It just wants the data from Person DB to use - So it MUST go through PersonService where all the checks are made to ensure no failure
Whole Purpose of food Service is to check and do any extra business logic (calculate calories)
It will Not check if PersonExists - it will Call PersonService to check for it!!!
You don't even need to check as long as all insert methods ensure these checks are made!!! - actuallly mayeb we should always check incase we delete perosn but not their entries!!
PersonService getPersonOrNull(person_id) : Person/Null (THIS WILL Always be a check Function!!)
FoodService getFoodByPersonByWeekByDay(person_id, week, day) : List<Food>
SERVICE - Business Logic & checks Before data Insertion
PersonDAO - interface
FoodDAO - interface
Asking for something
Returning something
Note: we don't even need getCalorieTargetByPerson inside the DAO - as get Person will return the Person object with Calories INSIDE!! So just do person.getCalorieTarget() to get the value!! Do this inside PersonService
PersonDataAccessService getPersonById (person_id) : Person/Null It coverts it into Object during RowMAPPER
FoodDataAccessService getFoodByPersonByWeekByDay(person_id, week, day)
FOODSERVICE WILL NEVER RETURN ANYTHING TO PERSON SERVICE!!!
DAO - Accessing the data - returning 0/1 , object/Null
people Table
food_entries Table
not_my_fitness_pal_DB

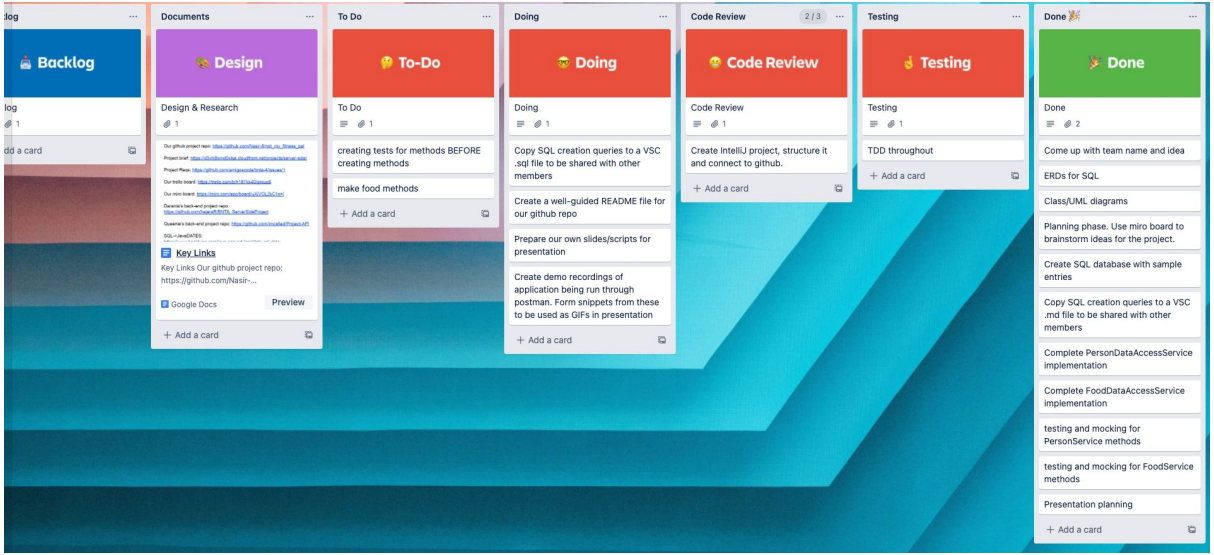
Planning - Week

- Mob Programming
  - Everyone clued up
- One method - reference
- Split up on Tues/Wed
- Key factor of success - collab

Day	Task
Friday	Choose project, Pojo, services
Saturday	Self-Learning
Sunday	**Mob Prog** - Setup Project, Git, TDD
Monday	**Mob Prog** - Complete One method, testing to API
Tuesday	Split up and finish all Methods/Testing ~~MVP~~
Wednesday	Attempt Stretch Goals, Start Presentation prep **(MVP/Testing)**
Thursday	Presentation Dry Run
Friday	Presentation day

Collaboration Tools - Trello

- Switch up the visual tool styles - detail/brainstorming vs **broken down targets.**
- **Cater for different minds**.
- Each day resembled an **agile sprint** (stand-up, implement, review, retrospect) - though **not firmly set** this way.

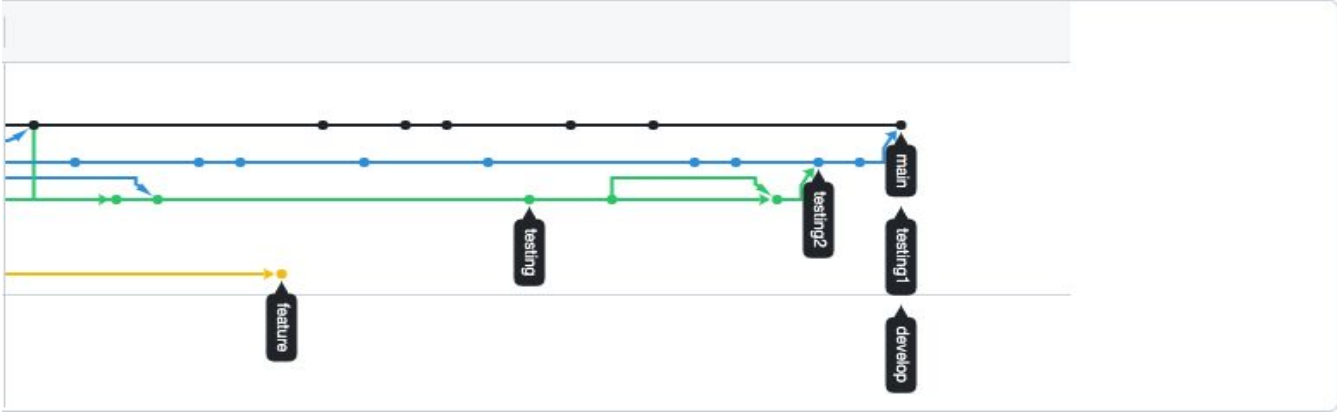


Collaboration Tools - Git

- Additional **branches** to **distinguish** the type of task we were doing and **avoid breaking main** code.
- **Staggered approach** to working through merges - **resolving any conflicts** was smoother.
- Led to much **more efficient merges**, coping as the code base grew.
- **Low** number of **merge conflicts** experienced during our project was a testament to this.

Top-right photo: A rough merge process we drew up - to reference back to when splitting off for pair programming.

Bottom-right photo: Branch structure as per github (towards end of project).



Demo: Post/Get

Home Workspaces API Network Reports Explore Search Postman Invite Upgrade

Overview | GET localhost:8080/food/c | POST localhost:8080/food | No Environment

localhost:8080/food Save

POST localhost:8080/food Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1  {
2      "id": 29,
3      "person_id": 2,
4      "name": "toast",
5      "mealType": "BREAKFAST",
6      "notes": "Wholemeal bread with butter",
7      "calories": 99,
8      "week": 1,
9      "day": "TUESDAY"
10 }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 511 ms Size: 4.02 KB Save Response

Pretty Raw Preview Visualize JSON

```
267         "notes": "bao buns and dumplings",
268         "calories": 300,
269         "week": 1,
270         "day": "TUESDAY"
271     },
272     {
273         "id": 28,
274         "person_id": 2,
275         "name": "spaghetti bolognese",
276         "mealType": "DINNER",
277         "notes": "homemade",
278         "calories": 200,
279         "week": 1,
280         "day": "TUESDAY"
281     }
282 ]
```

Find and Replace Console Capture requests and cookies Bootcamp Runner Trash

Demo: Put/Delete

Home Workspaces API Network Reports Explore Search Postman Invite Upgrade

Overview GET localhost:8080/food/c GET localhost:8080/food/2! No Environment

localhost:8080/food/29 Save

GET localhost:8080/food/29 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1  {
2      "id": 29,
3      "person_id": 2,
4      "name": "toast",
5      "mealType": "BREAKFAST",
6      "notes": "Wholemeal bread with butter",
7      "calories": 99,
8      "week": 1,
9      "day": "TUESDAY"
10 }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 124 ms Size: 302 B Save Response

Pretty Raw Preview Visualize JSON

```
1  {
2      "id": 29,
3      "person_id": 2,
4      "name": "toast",
5      "mealType": "BREAKFAST",
6      "notes": "Wholemeal bread with butter",
7      "calories": 99,
8      "week": 1,
9      "day": "TUESDAY"
10 }
```

Find and Replace Console Capture requests and cookies Bootcamp Runner Trash

Many more

Overview | POST localhost:8080/f... | DEL localhost:8080/fo... | GET localhost:8080/fo...

localhost:8080/food/person/mealtype/SNACK

GET localhost:8080/food/person/mealtype/SNACK

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
12  {
13      "id": 16,
14      "person_id": 4,
15      "name": "fridge raiders",
16      "mealType": "SNACK",
17      "notes": "chicken fridge raiders",
18      "calories": 100,
19      "week": 1,
20      "day": "TUESDAY"
21  },
22  {
23      "id": 22,
24      "person_id": 5,
25      "name": "ice cream",
26      "mealType": "SNACK",
27      "notes": "vanilla flavour",
28      "calories": 200,
29      "week": 1,
30      "day": "TUESDAY"
31  }
```


By Meal Type

Overview | POST localhost:8080/f... | DEL localhost:8080/fo... | GET localhost:8080/fo...

localhost:8080/food/person/5/week/1

GET localhost:8080/food/person/5/week/1

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
1  [
2      {
3          "id": 17,
4          "person_id": 5,
5          "name": "cereal and milk",
6          "mealType": "BREAKFAST",
7          "notes": "frosties",
8          "calories": 250,
9          "week": 1,
10         "day": "MONDAY"
11     },
12     {
13         "id": 18,
14         "person_id": 5,
15         "name": "cheese toastie",
16         "mealType": "LUNCH",
17         "notes": "cheddar",
18         "calories": 500,
19         "week": 1,
20         "day": "MONDAY"
```


By Week

Overview | POST localhost:8080/f... | DEL localhost:8080/fo... | GET localhost:8080/fo...

localhost:8080/food/person/5/week/1/day/TUESDAY

GET localhost:8080/food/person/5/week/1/day/TUESDAY

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE

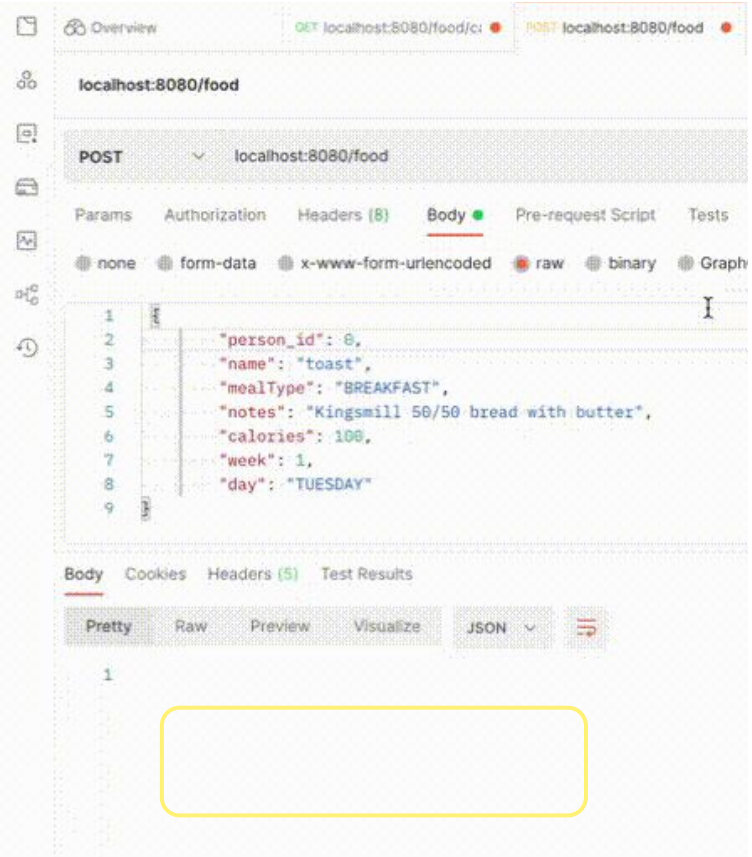
Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
1  [
2      {
3          "id": 20,
4          "person_id": 5,
5          "name": "avocado on toast",
6          "mealType": "BREAKFAST",
7          "notes": "sliced avocado on brown bread",
8          "calories": 200,
9          "week": 1,
10         "day": "TUESDAY"
11     },
12     {
13         "id": 21,
14         "person_id": 5,
15         "name": "chilli con carne",
16         "mealType": "DINNER",
17         "notes": "minced beef and rice",
18         "calories": 700,
19         "week": 1,
20         "day": "TUESDAY"
```


By Week and Day

Client Errors



400 Bad Request

```
@ResponseStatus(value = HttpStatus.BAD_REQUEST)
public class InvalidRequestException extends RuntimeException{
    public InvalidRequestException(String message) {
        super(message);
    }
}
```

404 Not Found

```
@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class FoodNotFoundException extends RuntimeException{
    public FoodNotFoundException(String message) {
        super(message);
    }
}
```

Custom Exception Class

Overview | POST localhost:8080/f... | DEL localhost:8080/fo...

localhost:8080/food/500

DELETE localhost:8080/food/500

Params | Authorization | Headers (6) | Body | Pre-request Script | Test

Query Params

KEY
Key

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | Visualize | JSON

```
1 {
2   "timestamp": "2022-02-24T13:30:08.334+00:00",
3   "status": 404,
4   "error": "Not Found",
5   "message": "Food with id 500 doesn't exist",
6   "path": "/food/500"
7 }
```

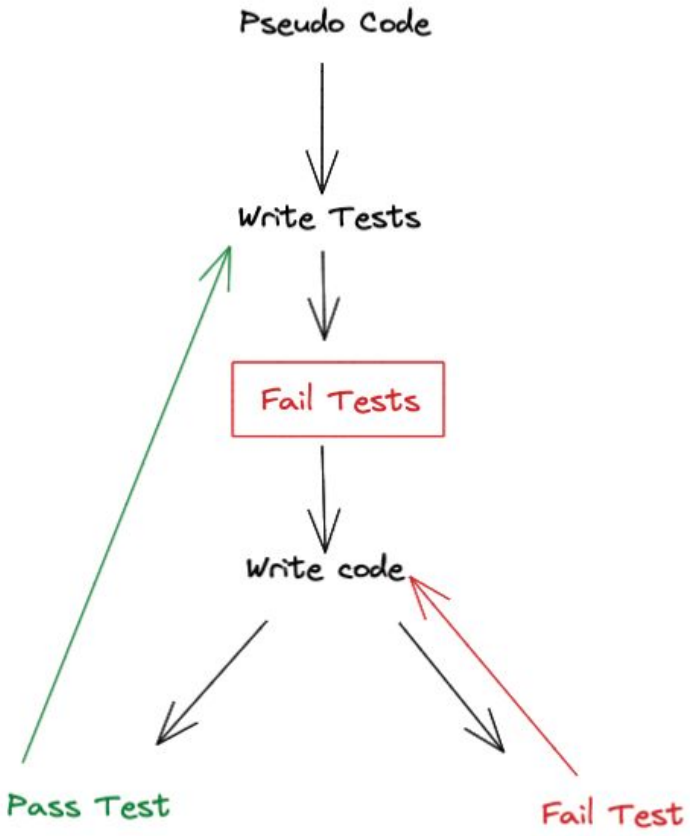
```
public int deleteFood(Integer foodId){
        Food foodInDb = getFoodOrThrowNull(foodId);

        int rowsAffected = foodDao.deleteFoodById(foodId);
        if (rowsAffected!=1){
            throw new IllegalStateException("Food could not be deleted");
        }
        return rowsAffected;
    }

private Food getFoodOrThrowNull(Integer id){
        if (id == null || id <= 0){
            throw new InvalidRequestException("Food id is invalid");
        }
        // This is the scenario where argument capture would help - makes sure id persists throughout
        //id = 25;  // Ignore - we were testing this for scenario mentioned on the line above
        Food food = foodDao.getFoodById(id); //mocking this line

        if(food == null){
            throw new FoodNotFoundException("Food with id " + id + " doesn't exist");
        }
        return food;
    }
```

Our Approach to Testing

- Lacking confidence in testing

- Using TDD - a great opportunity to explore **industry standard development**

- Approaching TDD
  - Mob pseudo code
  - Write tests
  - Write code



TDD in addFoodEntry method

Input

```
///TODO:3) Create Psuedo code for add food
  public int addFoodEntry(Food food) {
      // Check all fields are valid (enums don't have to be checked here):
      //    person_id - use person service, if null then throw exception - using PersonDao. (Mock)
      //    name - can't be null
      //    calories - can't be null or < 0 - 0 is accepted
      //    week - can't be null or <= 0
      // Add food entry to sql db - using FoodDao.  (Mock)
      // foodDao.addFood(someRandom)
      // If result != 1, then throw exception to say it failed

      return 1;
  }
```

Output

```java
@Test
    void addFoodEntry() {
            //Given
        Food food = new Food(1, 1, "toast", MealType.BREAKFAST,
                                "random", 50, 1, Day.MONDAY);

        Person personInDb = new Person(1, "marcy", 23, 157.0, 47.0, 2000);

        given(personDao.getPersonById(food.getPerson_id())).willReturn(personInDb);

        given(foodDao.addFood(food)).willReturn(1);
            //When
        Integer actual = underTest.addFoodEntry(food);
             //Then
        Integer expected = 1;

        assertThat(actual).isEqualTo(expected);
    }
```

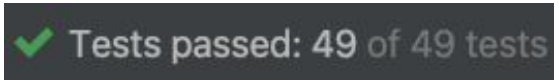
Results of TDD

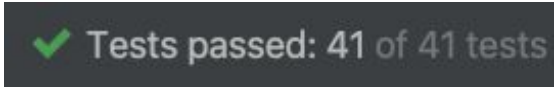
- Sunday/Monday - **TDD focused**

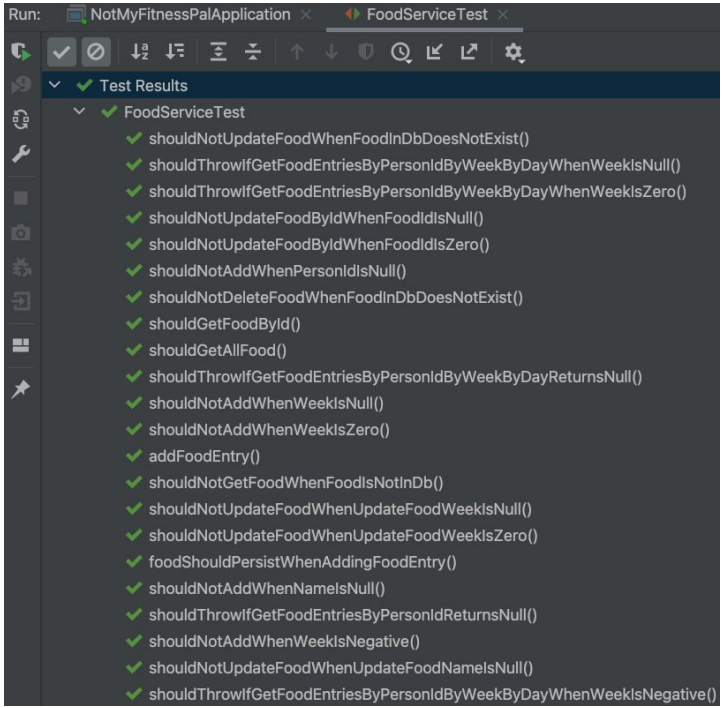
- Tuesday/Wednesday - More ambitious with methods, less TDD
  - Testing took longer, less conscious business logic

- **90 Tests** written and passed - a **reliable** API!

- **Pedagogically valuable,** whole team more confident in testing

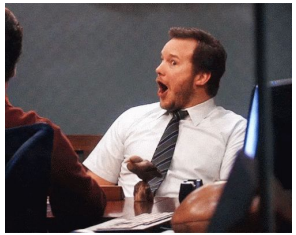






Argument Captors - a fun discovery!

- Originally had conflict over the need for an **ArgumentCaptor** when comparing two tests.
- Discovered the difference between them was down to **primitives** remaining consistent **vs objects** being traced as rows returned in SQL - thus not the specific object.
- **Cleaner code!**



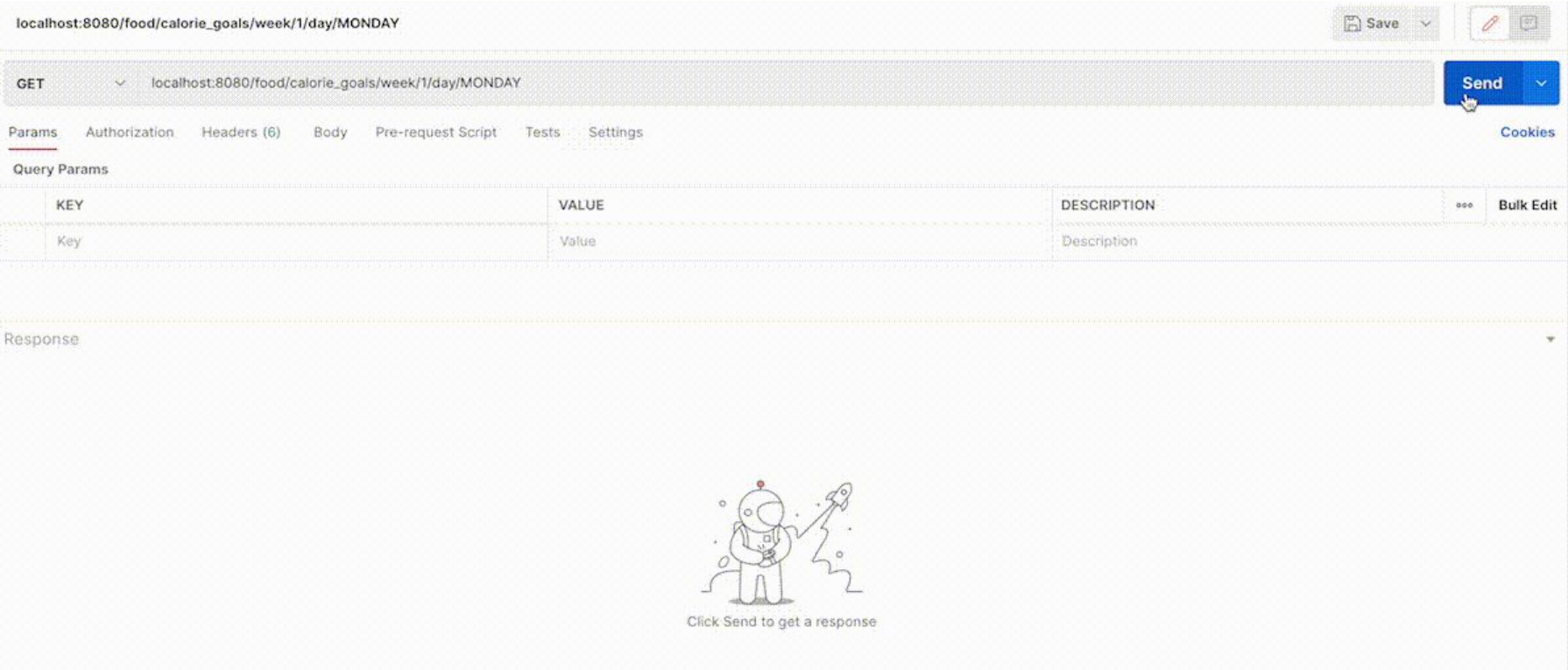
BEFORE

```java
@Test
    void shouldNotGetPersonByIdWhenPersonIdDoesNotExist() {
        //Given
        Integer id = 100; //person with id 100 does not exist within the db
        given(personDao.getPersonById(id)).willReturn(null);

        //When
        assertThatThrownBy(() -> underTest.getPersonById(id))
                .isInstanceOf(PersonNotFoundException.class)
                .hasMessageContaining("Person with id " + id + " doesn't exist");

        //Then
        ArgumentCaptor<Integer> integerArgumentCaptor = ArgumentCaptor.forClass(Integer.class);
        verify(personDao).getPersonById(integerArgumentCaptor.capture());
        Integer actual = integerArgumentCaptor.getValue();
        assertThat(actual).isEqualTo(id);
    }
```

AFTER

```java
@Test
    void shouldNotGetPersonByIdWhenPersonIdDoesNotExist() {
        //Given
        Integer id = 100; //person with id 100 does not exist within the db
        given(personDao.getPersonById(id)).willReturn(null);

        //When
        //Then
        assertThatThrownBy(() -> underTest.getPersonById(id))
                .isInstanceOf(PersonNotFoundException.class)
                .hasMessageContaining("Person with id " + id + " doesn't exist");

    }
```

Extension - daily calorie goals



SQL implementation

```java
public class PersonDailyCalorieGoal {

    private Integer id;
    private String name;
    private Integer calorie_target;
    private Integer week;
    private Day day;
    private Integer total_calories_on_week_on_day;
    // The two below are set in service (business logic)
    private Integer calorie_difference;
    private String calorie_goal_result;
```

Both set in service class

Calculates the sum of the total calories for a day per person

JOIN required as the food_entries table does not have access to the daily calorie target property in the people table

```java
    @Override
    public List<PersonDailyCalorieGoal> getDailyCalorieGoalsByWeekByDay(Integer week,
Day day) {
        String sql = """
                SELECT food_entries.person_id, people.name, people.calorie_target,
                SUM(food_entries.calories) AS total_calorie_intake, food_entries.day
                FROM food_entries
                INNER JOIN people
                ON food_entries.person_id = people.id
                WHERE food_entries.day = ? AND food_entries.week = ?
                GROUP BY (person_id, people.name, people.calorie_target, day)
                """;
```

Business logic

Work out the calorie difference then SET that value for each person

```
for (PersonDailyCalorieGoal calorieGoal : calorieGoalsList) {
        if(calorieGoal.getCalorie_target() == null){
            calorieGoal.setCalorie_goal_result(calorieGoal.getName() + " did not set a calorietarget." );
        } else{
            Integer calorie_difference = calorieGoal.getCalorie_target() - calorieGoal.getTotal_calories_on_week_on_day();
            calorieGoal.setCalorie_difference(calorie_difference);
            if (calorie_difference > 0){
                calorieGoal.setCalorie_goal_result(calorieGoal.getName() + " is " + Math.abs(calorie_difference) + " calories
                below their target." );
            } else if (calorie_difference < 0){
                calorieGoal.setCalorie_goal_result(calorieGoal.getName() + " is " + Math.abs(calorie_difference) + " calories
                above their target." );
            } else {
                calorieGoal.setCalorie_goal_result(calorieGoal.getName() +  " has met their daily calorie target of " +
                calorieGoal.getCalorie_target() + ".");
            }
        }
    }
    return calorieGoalsList;
```

SET the result property as a message indicating how much they are above/below their target

Ideas for the future

- Expand to **include exercise data** - include people's daily workouts and how many calories they are burning.

- Work out people's **daily net calories** then compare to daily calorie target.

Reflections

- Greater understanding of project work, thorough planning and TDD
- Came across interesting challenges
  - How to properly do TDD
  - ArgumentCaptor
  - Calorie Counting method

Please check out our repo if you're interested! Any improvements are welcomed!

https://github.com/Nasir-6/not_my_fitness_pal