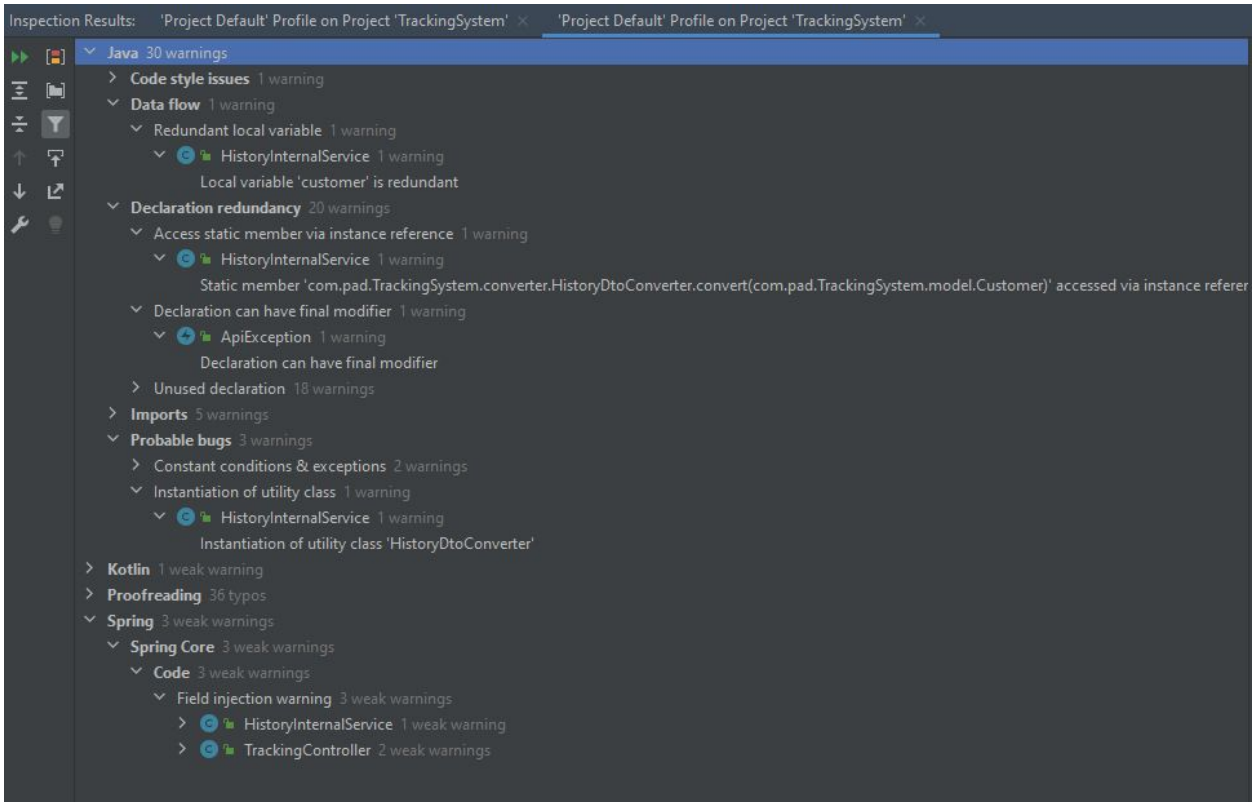


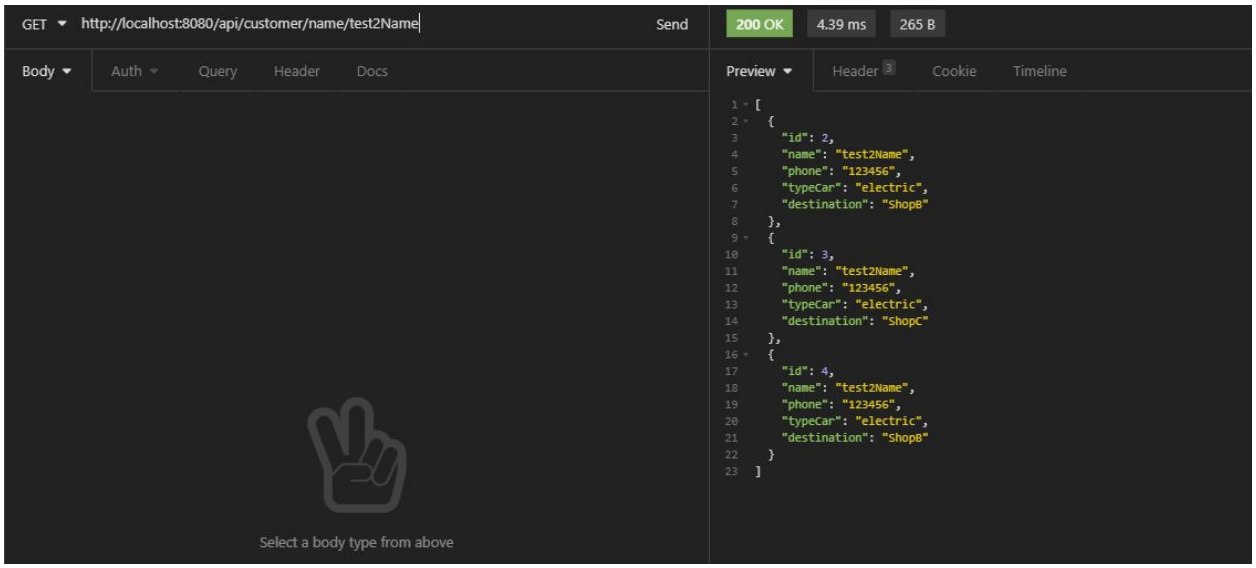

1. Redundant local variable

What I've used is a stream to create a list of HistoryDto collected by the name of the destination.

```
public List<HistoryDto> getDestinationsHistoryByName(String name) {
    List<HistoryDto> customer = historyRepository.findDestinationsByName(name) List<Customer>
            .stream() Stream<Customer>
            .map(HistoryDtoConverter::convert) Stream<HistoryDto>
            .collect(Collectors.toList());
    return customer;
}
```

How I solved:

```
    public List<HistoryDto> getDestinationsHistoryByName(String name) {
//      List<HistoryDto> customer = historyRepository.findDestinationsByName(name)
//              .stream()
//              .map(HistoryDtoConverter::convert)
//              .collect(Collectors.toList());
        return historyRepository.findDestinationsByName(name) List<Customer>
                .stream() Stream<Customer>
                .map(HistoryDtoConverter::convert) Stream<HistoryDto>
                .collect(Collectors.toList());
    }
```




2. Declaration can have final modifier

```
private final HttpStatus responseHttpStatus;
```

3. Access static member via instance reference

```
private HistoryDto convertToHistoryDto(Customer customer) {
    HistoryDtoConverter historyDtoConverter = new HistoryDtoConverter();
    return historyDtoConverter.convert(customer);
}
```

I just access the member via class directly.

```
private HistoryDto convertToHistoryDto(Customer customer) {
    return HistoryDtoConverter.convert(customer);
}
```

4. Unused imports

5. Spring - field injection warning

```
@Autowired
private HistoryRepository historyRepository;
```

We create a constructor.

```
private final HistoryRepository historyRepository;

public HistoryInternalService(HistoryRepository historyRepository) {
    this.historyRepository = historyRepository;
}
```

This is what remains: