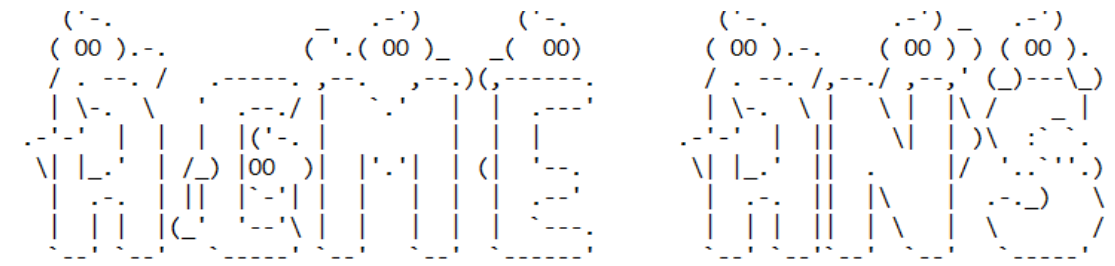
REPORTE DE TESTING Y MUTACIONES DEL CÓDIGO

Acme-ANS-D04



Repositorio: https://github.com/FranciscoFernandezN/Acme-ANS

Creado por el grupo C1.022, del G1

Participantes	
Nombres	Correos
Gutiérrez Arazo, Beatriz	beagutara@alum.us.es

25 de Mayo de 2025

Índice

Resumen ejecutivo

Este es el proyecto del grupo C1.022 sobre Acme AirNav Solutions, S.A. (Acme ANS, S.A. abreviado), la cual es una compañía ficticia especializada en ayudar a aeropuertos a organizar y coordinar sus operaciones a partir de soluciones desarrolladas en software. La logística de los vuelos (la programación de los vuelos, la organización de reservas y de tripulación, etc.) se gestionan mediante el desarrollo de un WIS.

Con esto, para el correcto funcionamiento de la aplicación, se deberá escribir un reporte de testing donde se describan los valores de cobertura del código junto con los valores correspondientes al rendimiento de la aplicación, ambos, previos y posteriores a realizar mutaciones en el código.

Tabla de revisiones

Número	Fecha	Descripción
v1.0.0	25/05/2025	Versión inicial terminada del reporte

Introducción

En este reporte quedarán expuestos los resultados de los tests realizados durante el entregable D04 sobre las funcionalidades desarrolladas por la estudiante 2, Beatriz Gutiérrez Arazo, sobre los requisitos 8, 9 y 29.

Se reportarán 2 tipos de resultados, los relacionados con el testeo funcional y los relacionados con el testeo del rendimiento. A su vez también se reportarán las 5 mutaciones realizadas sobre el código y sus distintos efectos.

Este reporte está organizado de la siguiente forma:

1. **Resumen ejecutivo**: Introducción breve sobre el reporte.
2. **Tabla de revisiones**: Historial de revisiones del documento.
3. **Introducción**: Contextualización de la planificación y progreso además de su importancia.
4. **Functional testing**: Resultados obtenidos de la fase de testing y posibles bugs encontrados.
5. **Performance testing**: Resultados obtenidos de analizar el rendimiento de los test en 2 ordenadores.
6. **Mutaciones en los bookings y passengers**: Resultados de introducir bugs intencionales en el código.
7. **Conclusiones**: Resumen de los hallazgos y la importancia de este reporte.
8. **Bibliografía**: Fuentes consultadas durante la investigación.

Functional testing

- Resultados sobre el testeo en los bookings:

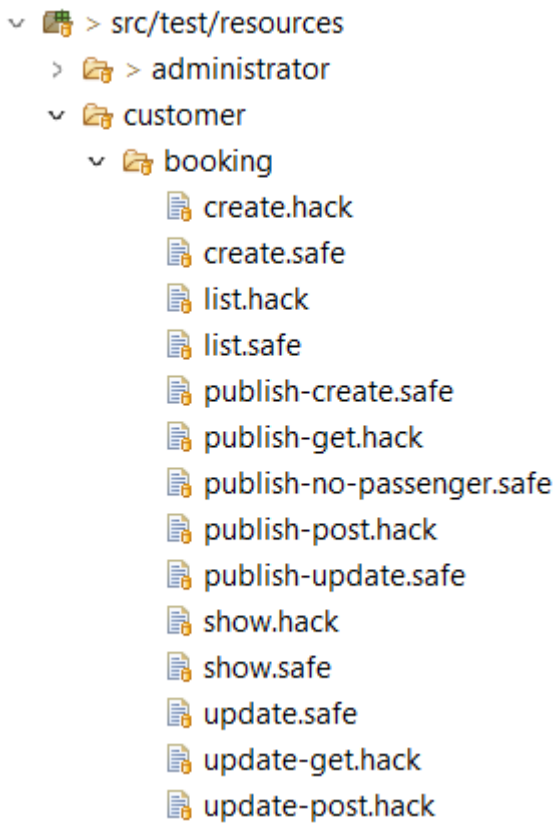


Figura 1: Archivos de testing para los bookings

No se encontraron demasiados bugs durante el testing, aunque sí bastante “código espagueti” el cual era inútil en muchos casos, y la creación de los tests fue muy útil para diferenciar qué podía ser borrado y que no.

acme.features.customer.booking	99,9 %	2.435	2	2.437
CustomerBookingCreateService.java	99,8 %	607	1	608
CustomerBookingPublishService.java	99,9 %	804	1	805
CustomerBookingController.java	100,0 %	30	0	30
CustomerBookingListService.java	100,0 %	101	0	101
CustomerBookingShowService.java	100,0 %	271	0	271
CustomerBookingUpdateService.java	100,0 %	622	0	622

Figura 2: Cobertura de los tests en los bookings

Se ha logrado alcanzar una cobertura casi total de la feature con un 99.9%, que cubre todas las instrucciones a ejecutar a excepción de algunas condicionales.

```java
@Override
public void authorise() {

    boolean status = super.getRequest().getPrincipal().hasRealmOfType(Customer.class);

    if (status && super.getRequest().hasData("flight")) {
        int flightId = super.getRequest().getData("flight", int.class);
        Flight flight = this.repository.findFlightById(flightId);
        status = flightId == 0 || flight != null && !flight.getIsDraftMode();
    }

    if (status && super.getRequest().hasData("passenger")) {
        int passengerId = super.getRequest().getData("passenger", int.class);
        Passenger passenger = this.repository.findPassengerById(passengerId);
        status = passengerId == 0 || passenger != null && this.repository.findPassengersByCustomerId(super.getRequest().getPri
    }

    super.getResponse().setAuthorised(status);

}
```

Figura 3: Condición sin cubrir al completo en CustomerBookingCreateService

Esta es una que se repite varias veces en el código, y se debe a que, tal como está planteado el framework ahora mismo, nadie que no sea customer puede acceder a esta función, por lo que la comprobación de si es customer sobraría. Sin embargo, con tal de evitar posibles errores en caso de que esto pueda cambiar en el futuro, he decidido dejar dicha comprobación.

```java
@Override
public void unbind(final Booking booking) {
    Dataset dataset;
    int customerId = super.getRequest().getPrincipal().getRealmOfType(Customer.class).getId();

    String locatorCode = "";

    if (super.getRequest().hasData("locatorCode"))
        locatorCode = super.getRequest().getData("locatorCode", String.class);
    else {
        int firstUppercaseIndex = 'A';
        boolean uniqueLocatorCode = false;

        while (!uniqueLocatorCode) {
            locatorCode = "";
            for (int i = 0; i < RandomHelper.nextInt(6, 8); i++) {
                Integer letterIndex = RandomHelper.nextInt(26);
                char letter = (char) (firstUppercaseIndex + letterIndex);
                int num = RandomHelper.nextInt(10);
                String chosen = RandomHelper.nextInt(2) == 1 ? String.valueOf(letter) : String.valueOf(num);
                locatorCode += chosen;
            }
            uniqueLocatorCode = this.repository.findBookingByLocatorCode(locatorCode) == null;
        }
    }
```

Figura 4: Condición que rara vez se cumple en CustomerBookingCreateService

En este caso, la condición que no se cumple se debe a que hay una probabilidad muy baja de que ocurra. Para llegar a esta línea, es necesario que el localizador que se genera automáticamente coincida con uno del sistema, lo cual es casi imposible debido a la gran cantidad de combinaciones posibles y que solo hay tres localizadores en el sample.

- Resultados sobre el testeo sobre los passengers:

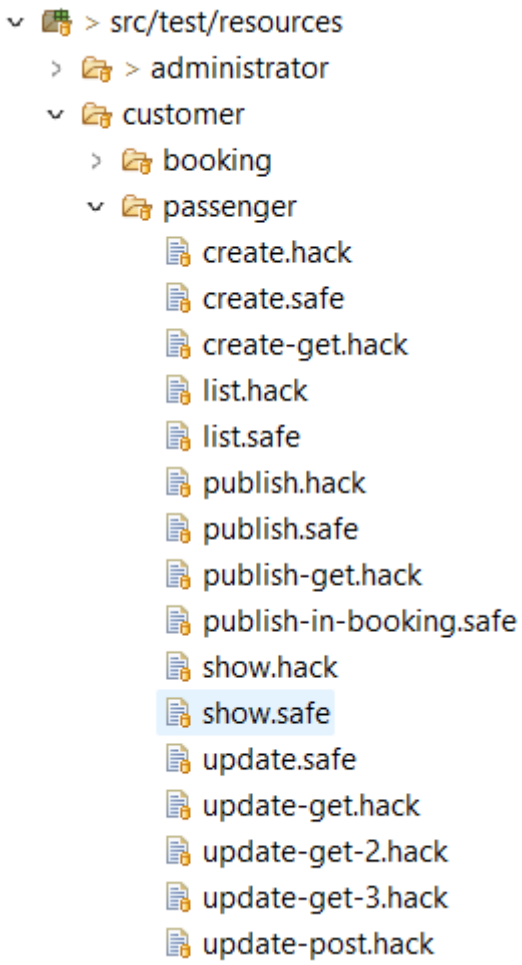


Figura 5: Archivos de testing para los passengers

De nuevo, no fueron encontrados bugs importantes, sino mucho código spaghetti que fue necesario refactorizar o borrar.

Al igual que antes, la cobertura alcanzada ha sido casi del 100%, y no al completo debido a condiciones no cumplidas.

acme.features.customer.passenger	99,9 %	1.845	2	1.847
CustomerPassengerCreateService.java	99,8 %	449	1	450
CustomerPassengerPublishService.java	99,8 %	594	1	595
CustomerPassengerController.java	100,0 %	30	0	30
CustomerPassengerListService.java	100,0 %	180	0	180
CustomerPassengerShowService.java	100,0 %	173	0	173
CustomerPassengerUpdateService.java	100,0 %	419	0	419

Figura 6: Cobertura de los tests en los passengers

```java
@Override
public void authorise() {
    boolean status;
    int masterId;
    int bookingId;
    Booking booking;

    status = super.getRequest().getPrincipal().hasRealmOfType(Customer.class);
    boolean hasMasterId = super.getRequest().hasData(CustomerPassengerController.MASTER_ID);
    boolean hasBooking = super.getRequest().hasData("booking");
    masterId = hasMasterId ? super.getRequest().getData(CustomerPassengerController.MASTER_ID, int.class) : 0;

    if (status && hasMasterId) {
        booking = this.repository.findBookingById(masterId);
        status = booking != null && super.getRequest().getPrincipal().hasRealm(booking.getCustomer()) && booking.getIsDraftMode();
    }

    if (status && hasBooking) {
        bookingId = super.getRequest().getData("booking", int.class);
        booking = this.repository.findBookingById(bookingId);
        status = bookingId == 0 || booking != null && super.getRequest().getPrincipal().hasRealm(booking.getCustomer()) && booking.getIsDraftMode();
        if (status && hasMasterId)
            status = bookingId == masterId;
    }

    super.getResponse().setAuthorised(status);

}
```

Figura 7: Condiciones sin cubrir al completo en CustomerPassengerCreateService

Esta vez son dos condiciones las que no terminan de cubrirse. La primera es igual que las anteriores, mientras que la segunda es una que comprueba que, cuando se tiene un “masterId” en la página (en este caso, estamos creando un pasajero que pertenece a una reserva), la reserva a la que el pasajero está asociado debe ser la misma. La desigualdad de dichos valores no se ha probado porque hay un readOnly en la propiedad, y al estar en un selectChoices no se le puede hacer post hacking. Sin embargo, en caso de que hubiera alguna forma de que sí fuera posible, sería un problema de seguridad ya que la propiedad se encuentra en el bind (para cuando este create no tenga un masterId asociado). Por lo tanto se realiza dicha comprobación, aunque en principios no debiera de incumplirse, con tal de evitar futuros problemas de seguridad.

- Resultados sobre el testeo sobre los recommendations:

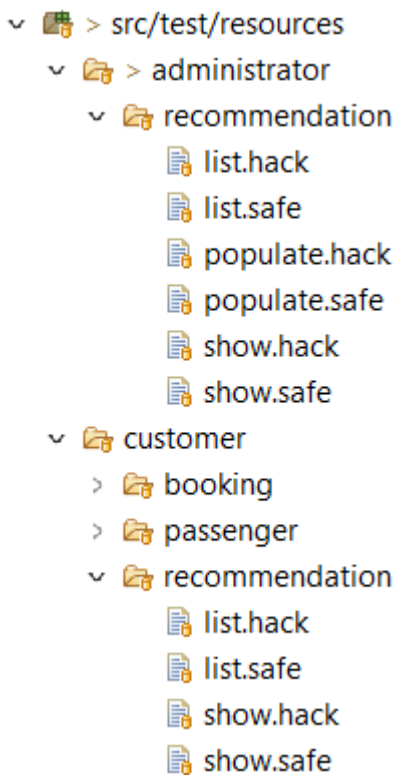


Figura 8: Archivos de testing para las recomendaciones

Durante este apartado no se encontraron bugs, pero sí fue necesario implementar un mockeo de la api para usarlo en las pruebas en vez de llamadas de verdad (cuyos resultados pueden variar).

La cobertura alcanzada por la parte de customer ha sido del 100%. Sin embargo, en la del administrador solo se tiene un 81,1%, lo cual se debe al mockup implementado.

acme.features.customer.recommendation	100,0 %	325	0	325
CustomerRecommendationController.java	100,0 %	20	0	20
CustomerRecommendationListRelatedService.java	100,0 %	83	0	83
CustomerRecommendationListService.java	100,0 %	118	0	118
CustomerRecommendationShowService.java	100,0 %	104	0	104
acme.features.administrator.recommendation	81,1 %	317	74	391
AdministratorRecommendationController.java	67,5 %	154	74	228
AdministratorRecommendationListService.java	100,0 %	59	0	59
AdministratorRecommendationShowService.java	100,0 %	104	0	104

Figura 9: Cobertura de los tests en los recommendations

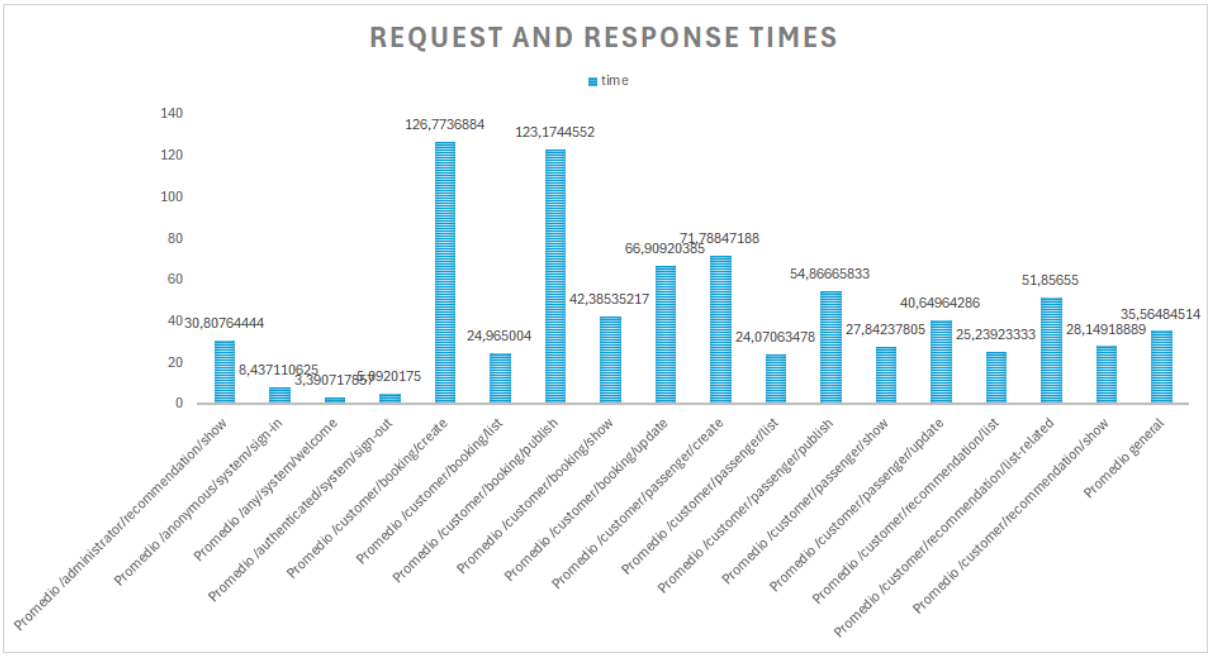
```
protected ModelAndView doPopulate() {

    List<String> cities = this.repository.findAllCities();

    List<List<Recommendation>> recommendations;

    List<String> recommendationsNames = this.repository.findAllRecommendationsNames();

    if (SpringHelper.isRunningOn("testing"))
        recommendations = cities.stream().map(this::findRecommendationOfCityMocked).toList();
    else
        recommendations = cities.stream().map(this::findRecommendationOfCity).filter(c -> c != null).toList();

    for (List<Recommendation> lis : recommendations)
        for (Recommendation rec : lis) {
            String name = rec.getName();
            if (recommendationsNames.contains(name))
                this.repository.delete(this.repository.findRecommendationByName(name));
            this.repository.save(rec);
        }
    ModelAndView result = new ModelAndView();
    result.setViewName("fragments/welcome");
    result.addObject("_globalSuccessMessage", "acme.default.global.message.success");
    return result;
}

protected List<Recommendation> findRecommendationOfCity(final String city) {
    try {
        String formattedCity = city.toLowerCase().replace(" ", "+");
        String apiKey = "AIzaSyC2AU9Q3g-xuKWQiphz4x_meZBn2eKmmTs";
        String url = "https://maps.googleapis.com/maps/api/place/textsearch/json?query=" + formattedCity + "+point+of+interest&language=en&key=" + apiKey;
        RestTemplate api = new RestTemplate();
        ResponseEntity<ResultsPOJO> response = api.getForEntity(url, ResultsPOJO.class);
        List<Recommendation> results = new ArrayList<>();
        Recommendation recommendation;
        for (RecommendationPOJO recPOJO : response.getBody().getRecommendations().subList(0, 5)) {
            recommendation = Recommendation.of(recPOJO, city);
            if (recommendation != null)
                results.add(recommendation);
        }
        return results;
    } catch (final Throwable oops) {
        return null;
    }

}
```

Figura 10: Código sin cubrir en AdministratorRecommendationController

Debido a que no es conveniente hacer llamadas a la APIs durante los tests (los resultados cambian, se pueden agotar las llamadas, etc.), se ha mockeado la aplicación. Esto conlleva que la parte no mockeada no pueda ejecutarse en los tests.

Performance testing

- Resultados en ordenador 1 con índices:



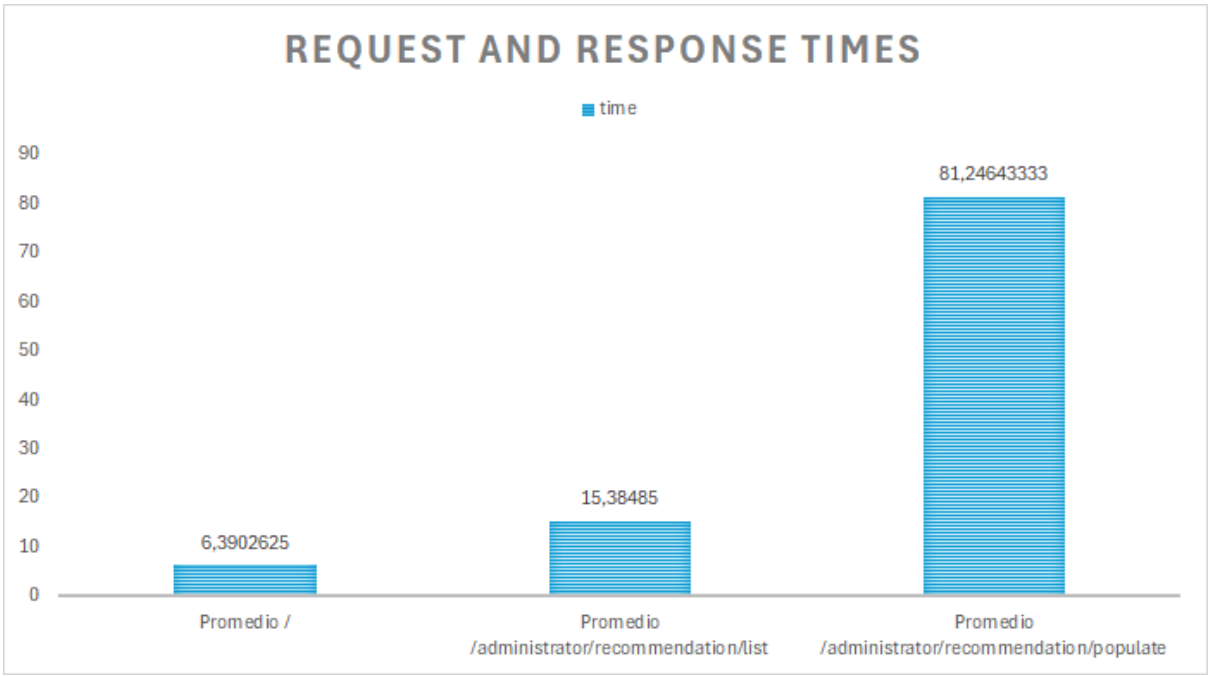


Figura 1 y 2: Tiempos para Ordenador 1 con índices

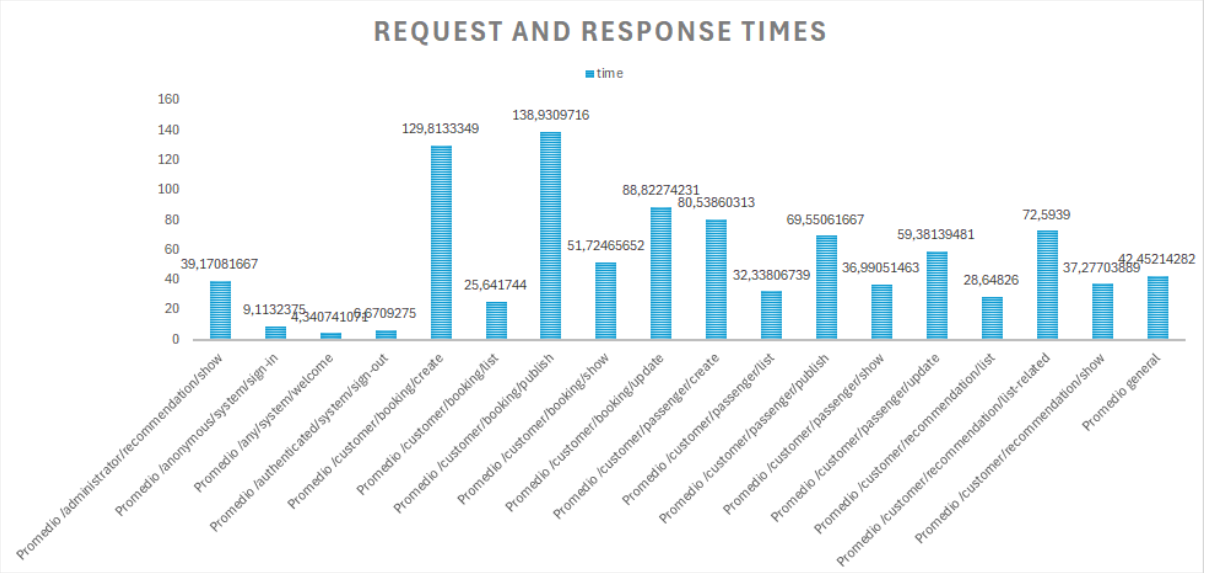
(Debido a problemas que presentaba el excel para crear una sola gráfica con tantas barras, algunos datos han tenido que aparecer en otra distinta).

En esta tabla podemos observar los tiempos promedios de las distintas peticiones realizadas tanto en los bookings como en los passengers con los índices calculados por el ordenador 1.

Es notable que las peticiones que más han tardado son las relacionadas con la creación y publicación de datos, ya que para testear todos los límites de los datos en el formulario han sido necesarias una enorme cantidad de peticiones, además de las numerosas comprobaciones de get hacking y post hacking.

Por otra parte, los lists no han requerido tantas peticiones (la gran mayoría de ellas han sido innecesarias, pues ocurrían al tratar de acceder a los bookings de estas como parte de otro test). Por último, también aparecen otras peticiones no tan relacionadas con el testeo de esta feature, que son las de iniciar sesión, cerrar sesión, y la página de bienvenida.

- Resultados en ordenador 1 sin índices:



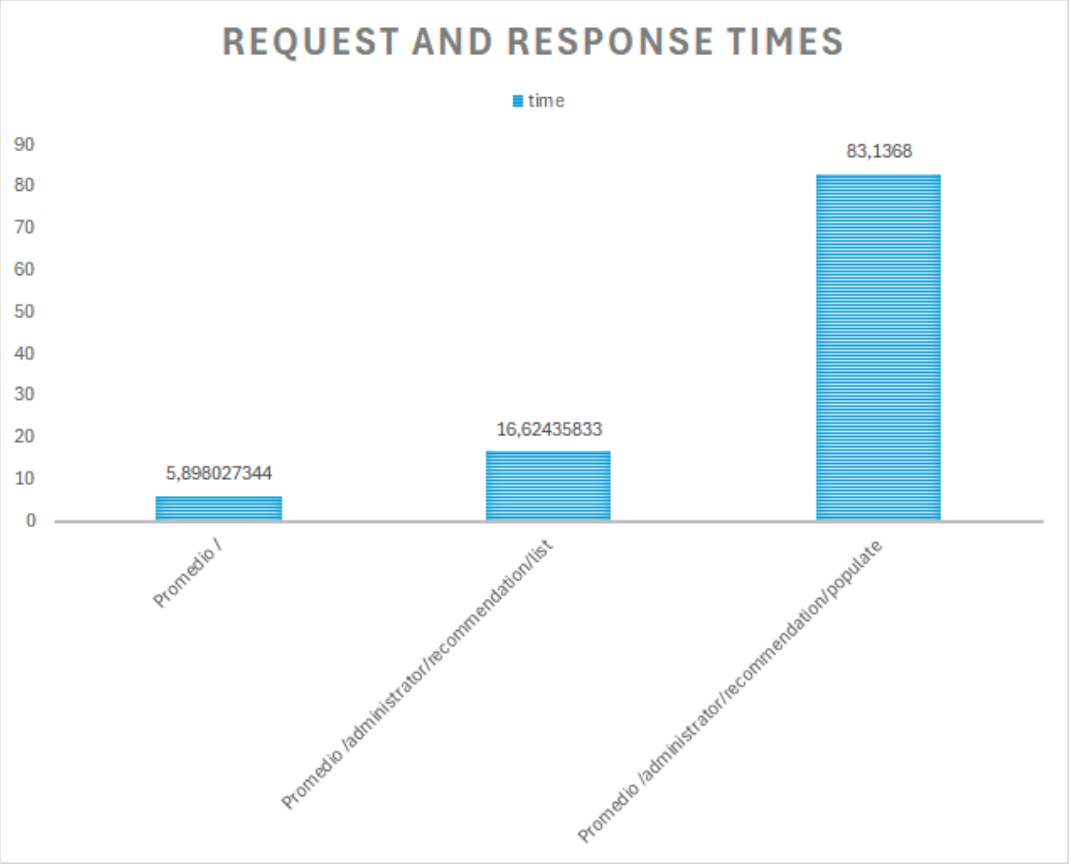


Figura 3 y 4: Tiempos para Ordenador 1 sin índices

Como se puede observar, los tiempos han aumentado al quitar los índices, lo cual significa que estos fueron correctamente colocados para optimizar el tiempo de respuesta de las queries. No solo eso, también podemos ver que el tiempo de esas queries afecta mucho al publish, que si bien con los índices sus peticiones habían consumido menos tiempo que las del create, esta vez ha sido más.

Con índices	
Media	35,5648451
Error típico	1,38645951
Mediana	14,6059
Moda	2,2828
Desviación estándar	46,4205196
Varianza de la muestra	2154,86464
Curtosis	5,56044536
Coeficiente de asimetría	2,23175343
Rango	274,3011
Mínimo	1,8614
Máximo	276,1625
Suma	39868,1914
Cuenta	1121
Nivel de confianza(95,0%)	2,72035049

Sin índices	
Media	42,4521428
Error típico	1,52789378
Mediana	19,0079
Moda	3,502
Desviación estándar	51,1559281
Varianza de la muestra	2616,92898
Curtosis	3,47060957
Coeficiente de asimetría	1,81514569
Rango	295,7
Mínimo	2,267
Máximo	297,967
Suma	47588,8521
Cuenta	1121
Nivel de confianza(95,0%)	2,99785645

Prueba z para medias de dos muestras

	Con índices	*Sin índices*
Media	35,5648451	42,4521428
Varianza (conocida)	2154,86464	2616,92898
Observaciones	1121	1121
Diferencia hipotética de las medias	0	
z	-3,33818833	
P(Z<=z) una cola	0,00042163	
Valor crítico de z (una cola)	1,64485363	
Valor crítico de z (dos colas)	0,00084327	
Valor crítico de z (dos colas)	1,95996398	

Figura 5: Valores estadísticos entre Ordenador 1 con y sin índices

El valor crítico de z (dos colas) es bastante menor que el intervalo de confianza, lo cual significa que se pueden comparar las medias de los tiempos. Fácilmente se puede ver que con índices ha tardado menos, lo cual nos confirma lo que habíamos dicho anteriormente: el uso de índices ha optimizado los tiempos de respuestas.

- Resultados del ordenador 2:

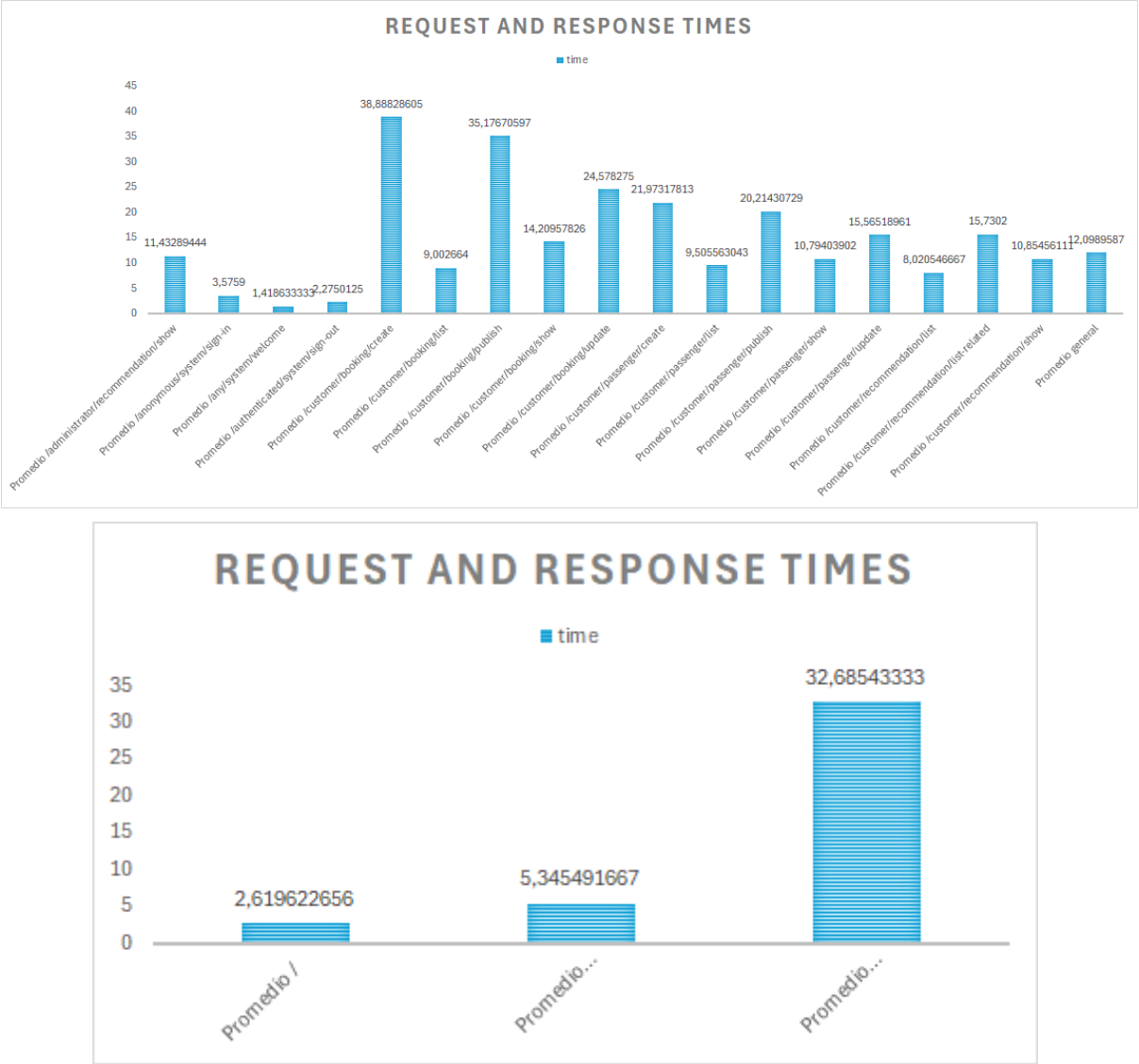


Figura 6: Tiempos para Ordenador 2 con índices

En esta tabla podemos observar los tiempos promedios de las distintas peticiones realizadas tanto en los bookings como en los passengers con los índices calculados por el ordenador 2.

Estos resultados fueron obtenidos con un ordenador de más potencia, y si bien las proporciones para las peticiones se mantienen igual, los tiempos son mucho más cortos que en los otros.

Ordenador 1	
Media	35,56484514
Error típico	1,38645951
Mediana	14,6059
Moda	2,2828
Desviación estándar	46,42051959
Varianza de la muestra	2154,864639
Curtosis	5,560445362
Coeficiente de asimetría	2,231753432
Rango	274,3011
Mínimo	1,8614
Máximo	276,1625
Suma	39868,1914
Cuenta	1121
Nivel de confianza(95,0%)	2,720350486

Ordenador 2	
Media	12,0989587
Error típico	0,41498909
Mediana	5,7109
Moda	1,3386
Desviación estándar	13,89439
Varianza de la muestra	193,054075
Curtosis	4,47620462
Coeficiente de asimetría	1,91332468
Rango	96,7388
Mínimo	0,6416
Máximo	97,3804
Suma	13562,9327
Cuenta	1121
Nivel de confianza(95,0%)	0,81424359

Prueba z para medias de dos muestras

	Ordenador 1	*Ordenador 2*
Media	35,56484514	12,0989587
Varianza (conocida)	2154,86464	2616,92898
Observaciones	1121	1121
Diferencia hipotética de las medias	0	
z	11,37362604	
P(Z<=z) una cola	0	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0	
Valor crítico de z (dos colas)	1,959963985	

Figura 3: Valores estadísticos entre Ordenador 1 y Ordenador 2 con índices

Al ser el valor de z (dos colas) 0, se pueden hacer comparativas entre las medias. En este caso, está claro que el ordenador 2 es mucho más apto que el ordenador 1 para ejecutar esta aplicación.

Mutaciones en bookings y passengers

- Mutación 1, modificar los límites del lastNibble de Booking:

```java
@Optional
@ValidString(min = 4, max = 4, pattern = "\\d{4}")
@Automapped
private String              lastNibble;
```

Figura 1: Estado inicial del código

```java
@Optional
@ValidString(min = 4, max = 5, pattern = "\\d{4,5}")
@Automapped
private String              lastNibble;
```

Figura 2: Mutación 1


```
FAILED POST /customer/booking/create (request-id="df5345a8-7d05-4476-acac-547024d06659",
input="id=0&version=0&locatorCode=1SM1811&purchaseMoment=2025%2F01%2F01+00%3A00&flight=0&travelClass=0&lastNibble=AAAAA&passenger=0"): Expected 'payload' to be '{flight=0, flight$error=Choose a flight. May
not be null., flightChoices=[{"key":"60","label":"London - Barcelona, 2024-03-12 06:00:00.0 - 2024-12-10 09:00:00.0, <<EUR 1000.00>>, LONG FLIGHT","selected":false,"sealed":true},{"key":"64","label":"Los
Angeles - Barcelona, 2030-01-01 20:00:00.0 - 2030-01-02 14:00:00.0, <<EUR 500000.00>>, Lorem ipsum dolor sit ame
","selected":false,"sealed":true},{"key":"67","label":"New York - London, 2031-12-31 02:00:00.0 - 2031-12-31 11:23:00.0, <<EUR 24183.71>>, لوريم إيبسوم
","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], id=0, isDraftMode=true, lastNibble=AAAAA, lastNibble$error=Length must be between 4 and 4., locatorCode=1SM1811,
passenger=0, passengerChoices=[{"key":"104","label":"MFURBWKGN - Rei Fujisaki (天寫)","selected":false,"sealed":true},{"key":"105","label":"353252 - Pasajero 4","selected":false,"sealed":true},
{"key":"108","label":"7777777 - Pasajero baneado 1","selected":false,"sealed":true},{"key":"102","label":"R8GJ4NDI3 - ʋosa
ɱartínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz.","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], purchaseMoment=2025/01/01 00:00, service=158, travelClass=0,
travelClass$error=Choose a travel class. May not be null., travelClasses=[{"key":"0","label":"----","selected":true,"sealed":true},{"key":"ECONOMY","label":"ECONOMY","selected":false,"sealed":true},
{"key":"BUSINESS","label":"BUSINESS","selected":false,"sealed":true}], updatedBooking=false, version=0}', but got '{flight=0, flight$error=Choose a flight. May not be null., flightChoices=
[{"key":"60","label":"London - Barcelona, 2024-03-12 06:00:00.0 - 2024-12-10 09:00:00.0, <<EUR 1000.00>>, LONG FLIGHT","selected":false,"sealed":true},{"key":"64","label":"Los Angeles - Barcelona, 2030-01-
01 20:00:00.0 - 2030-01-02 14:00:00.0, <<EUR 500000.00>>, Lorem ipsum dolor sit ame
","selected":false,"sealed":true},{"key":"67","label":"New York - London, 2031-12-31 02:00:00.0 - 2031-12-31 11:23:00.0, <<EUR 24183.71>>, لوريم إيبسوم
","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], id=0, isDraftMode=true, lastNibble=AAAAA, lastNibble$error=Must match pattern "\d{4,5}"., locatorCode=1SM1811,
passenger=0, passengerChoices=[{"key":"104","label":"MFURBWKGN - Rei Fujisaki (天寫)","selected":false,"sealed":true},{"key":"105","label":"353252 - Pasajero 4","selected":false,"sealed":true},
{"key":"108","label":"7777777 - Pasajero baneado 1","selected":false,"sealed":true},{"key":"102","label":"R8GJ4NDI3 - ʋosa
ɱartínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz.","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], purchaseMoment=2025/01/01 00:00, service=158, travelClass=0,
travelClass$error=Choose a travel class. May not be null., travelClasses=[{"key":"0","label":"----","selected":true,"sealed":true},{"key":"ECONOMY","label":"ECONOMY","selected":false,"sealed":true},
{"key":"BUSINESS","label":"BUSINESS","selected":false,"sealed":true}], updatedBooking=false, version=0}'.
FAILED POST /customer/booking/create (request-id="98ede5a9-4810-472c-8bf9-aa498c6de4d7",
input="id=0&version=0&locatorCode=1SM1811&purchaseMoment=2025%2F01%2F01+00%3A00&flight=0&travelClass=0&lastNibble=AAAA&passenger=0"): Expected 'payload' to be '{flight=0, flight$error=Choose a flight. May
not be null., flightChoices=[{"key":"60","label":"London - Barcelona, 2024-03-12 06:00:00.0 - 2024-12-10 09:00:00.0, <<EUR 1000.00>>, LONG FLIGHT","selected":false,"sealed":true},{"key":"64","label":"Los
Angeles - Barcelona, 2030-01-01 20:00:00.0 - 2030-01-02 14:00:00.0, <<EUR 500000.00>>, Lorem ipsum dolor sit ame
","selected":false,"sealed":true},{"key":"67","label":"New York - London, 2031-12-31 02:00:00.0 - 2031-12-31 11:23:00.0, <<EUR 24183.71>>, لوريم إيبسوم
","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], id=0, isDraftMode=true, lastNibble=AAAA, lastNibble$error=Must match pattern "\d{4}"., locatorCode=1SM1811,
passenger=0, passengerChoices=[{"key":"104","label":"MFURBWKGN - Rei Fujisaki (天寫)","selected":false,"sealed":true},{"key":"105","label":"353252 - Pasajero 4","selected":false,"sealed":true},
{"key":"108","label":"7777777 - Pasajero baneado 1","selected":false,"sealed":true},{"key":"102","label":"R8GJ4NDI3 - ʋosa
ɱartínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz.","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], purchaseMoment=2025/01/01 00:00, service=158, travelClass=0,
travelClass$error=Choose a travel class. May not be null., travelClasses=[{"key":"0","label":"----","selected":true,"sealed":true},{"key":"ECONOMY","label":"ECONOMY","selected":false,"sealed":true},
{"key":"BUSINESS","label":"BUSINESS","selected":false,"sealed":true}], updatedBooking=false, version=0}', but got '{flight=0, flight$error=Choose a flight. May not be null., flightChoices=
[{"key":"60","label":"London - Barcelona, 2024-03-12 06:00:00.0 - 2024-12-10 09:00:00.0, <<EUR 1000.00>>, LONG FLIGHT","selected":false,"sealed":true},{"key":"64","label":"Los Angeles - Barcelona, 2030-01-
01 20:00:00.0 - 2030-01-02 14:00:00.0, <<EUR 500000.00>>, Lorem ipsum dolor sit ame
","selected":false,"sealed":true},{"key":"67","label":"New York - London, 2031-12-31 02:00:00.0 - 2031-12-31 11:23:00.0, <<EUR 24183.71>>, لوريم إيبسوم
","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], id=0, isDraftMode=true, lastNibble=AAAA, lastNibble$error=Must match pattern "\d{4,5}"., locatorCode=1SM1811,
passenger=0, passengerChoices=[{"key":"104","label":"MFURBWKGN - Rei Fujisaki (天寫)","selected":false,"sealed":true},{"key":"105","label":"353252 - Pasajero 4","selected":false,"sealed":true},
{"key":"108","label":"7777777 - Pasajero baneado 1","selected":false,"sealed":true},{"key":"102","label":"R8GJ4NDI3 - ʋosa
ɱartínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz.","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], purchaseMoment=2025/01/01 00:00, service=158, travelClass=0,
```


Figura 3: Fallo detectado

Como se puede ver, si se tratara de introducir un número con menos o más de 4 dígitos, los tests de booking create.safe, publish-create.safe, publish-update.safe y update.safe darán fallos al no estar cumpliéndose ese límite.

- Mutación 2, quitar un argumento en un authorise de CustomerBookingUpdateService:

```java
@Override
public void authorise() {
    boolean status;
    int bookingId;
    Booking booking;

    status = super.getRequest().getPrincipal().hasRealmOfType(Customer.class);

    if (status && super.getRequest().hasData("id")) {
        bookingId = super.getRequest().getData("id", int.class);
        booking = this.repository.findBookingById(bookingId);
        status = booking != null && super.getRequest().getPrincipal().hasRealm(booking.getCustomer()) && booking.getIsDraftMode();
    } else
        status = false;
```

Figura 4: Estado inicial del código

```java
@Override
public void authorise() {
    boolean status;
    int bookingId;
    Booking booking;

    status = super.getRequest().getPrincipal().hasRealmOfType(Customer.class);

    if (status && super.getRequest().hasData("id")) {
        bookingId = super.getRequest().getData("id", int.class);
        booking = this.repository.findBookingById(bookingId);
        status = booking != null && super.getRequest().getPrincipal().hasRealm(booking.getCustomer())
    } else
        status = false;
```

Figura 5: Mutación 2

```
Resetting application (clear schema, populate sample, reset clock, reset randomiser).
Replaying .\src\test\resources\customer\booking\update-get.hack...
FAILED GET /customer/booking/update?id=116 (request-id="b9ffa968-a3ca-42cd-8a9e-eaffe1f8ac02", input=""): Expected 'status' to be '500', but got '200'. Expected 'payload' to be '{service=159}', but got
'{city=Barcelona, flightChoices=[{"key":"60","label":"London - Barcelona, 2024-03-12 06:00:00.0 - 2024-12-10 09:00:00.0, <<EUR 1000.00>>, LONG FLIGHT","selected":true,"sealed":true},
{"key":"64","label":"Los Angeles - Barcelona, 2030-01-01 20:00:00.0 - 2030-01-02 14:00:00.0, <<EUR 500000.00>>, Lorem ipsum dolor sit ame
","selected":false,"sealed":true},{"key":"67","label":"New York - London, 2031-12-31 02:00:00.0 - 2031-12-31 11:23:00.0, <<EUR 24183.71>>, لوريم إيبسوم
","selected":false,"sealed":true}], id=116, isDraftMode=false, lastNibble=1234, locatorCode=R8GJ4NDI, passenger=-1, passengerChoices=[{"key":"102","label":"R8GJ4NDI3 - ʋosa
ɱartínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz.","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], purchaseMoment=1900/01/01 00:00, service=159, travelClass=ECONOMY,
travelClasses=[{"key":"0","label":"----","selected":false,"sealed":true},{"key":"ECONOMY","label":"ECONOMY","selected":true,"sealed":true},
{"key":"BUSINESS","label":"BUSINESS","selected":false,"sealed":true}], updatedBooking=true, version=0}'.
```

Figura 6: Fallo detectado

A partir de este cambio sí se permitiría editar una reserva que haya sido publicada, pero gracias a uno de los tests se puede ver este fallo (devuelve un 200 con la reserva correspondiente cuando sabe que debería devolver un 500 con un not authorised).

- Mutación 3, eliminar una validación en CustomerBookingPublishService:

```java
@Override
public void validate(final Booking booking) {
    String lastNibble;
    int bookingId = booking.getId();

    int flightId = super.getRequest().getData("flight", int.class);
    Flight flight = this.repository.findFlightById(flightId);
    super.state(flight != null, "flight", "customer.booking.publish.flight-must-be-chosen");

    Booking bookingOfLocatorCode = this.repository.findBookingByLocatorCode(super.getRequest().getData("locatorCode", String.class));
    super.state(bookingOfLocatorCode == null || bookingOfLocatorCode.getId() == bookingId, "locatorCode", "customer.booking.publish.locator-not-unique");

    int passengerId = super.getRequest().getData("passenger", int.class);
    Passenger passenger = this.repository.findPassengerById(passengerId);

    if (passenger != null) {
        Collection<Passenger> passengersOfCustomer = this.repository.findPassengersByCustomerId(super.getRequest().getPrincipal().getRealmOfType(Customer.class).getId());
        boolean yours = passengersOfCustomer.contains(passenger);
        super.state(yours, "passenger", "customer.booking.publish.passenger-not-yours");
    }

    lastNibble = super.getRequest().getData("lastNibble", String.class);

    super.state(booking.getTravelClass() != null, "travelClass", "customer.booking.publish.travel-class-does-not-exist");

    super.state(!lastNibble.isBlank(), "lastNibble", "customer.booking.publish.need-last-nibble");

    Collection<Passenger> passengers = this.repository.findPassengersByBookingId(booking.getId());
    if (passenger != null)
        passengers.add(passenger);
    boolean thereArePassenger = !passengers.isEmpty();
    super.state(thereArePassenger, "passenger", "customer.booking.publish.need-passenger");

    if (thereArePassenger)
        super.state(passengers.stream().allMatch(p -> !p.getIsDraftMode()), "passenger", "customer.booking.publish.need-passengers-published");

}
```

Figura 7: Estado inicial del código

```java
@Override
public void validate(final Booking booking) {
    String lastNibble;

    int flightId = super.getRequest().getData("flight", int.class);
    Flight flight = this.repository.findFlightById(flightId);
    super.state(flight != null, "flight", "customer.booking.publish.flight-must-be-chosen");

    int passengerId = super.getRequest().getData("passenger", int.class);
    Passenger passenger = this.repository.findPassengerById(passengerId);

    if (passenger != null) {
        Collection<Passenger> passengersOfCustomer = this.repository.findPassengersByCustomerId(super.getRequest().getPrincipal().getRealmOfType(Customer.class).getId());
        boolean yours = passengersOfCustomer.contains(passenger);
        super.state(yours, "passenger", "customer.booking.publish.passenger-not-yours");
    }

    lastNibble = super.getRequest().getData("lastNibble", String.class);

    super.state(booking.getTravelClass() != null, "travelClass", "customer.booking.publish.travel-class-does-not-exist");

    super.state(!lastNibble.isBlank(), "lastNibble", "customer.booking.publish.need-last-nibble");

    Collection<Passenger> passengers = this.repository.findPassengersByBookingId(booking.getId());
    if (passenger != null)
        passengers.add(passenger);
    boolean thereArePassenger = !passengers.isEmpty();
    super.state(thereArePassenger, "passenger", "customer.booking.publish.need-passenger");

    if (thereArePassenger)
        super.state(passengers.stream().allMatch(p -> !p.getIsDraftMode()), "passenger", "customer.booking.publish.need-passengers-published");

}
```

Figura 8: Mutación 3

```
Resetting application (clear schema, populate sample, reset clock, reset randomiser).
Replaying .\src\test\resources\customer\booking\publish-create.safe...
FAILED POST /customer/booking/publish (request-id="b97f1832-d2bc-4f4b-9f49-08fbfc470b4d",
input="id=0&version=0&locatorCode=R8GJ4NDI&purchaseMoment=2025%2F01%2F01+00%3A00&flight=0&travelClass=0&lastNibble=&passenger=0"): Expected 'payload' to be '{flight=0, flight$error=Choose a flight. May not
be null., flightChoices=[{"key":"60","label":"London - Barcelona, 2024-03-12 06:00:00.0 - 2024-12-10 09:00:00.0, <<EUR 1000.00>>, LONG FLIGHT","selected":false,"sealed":true},{"key":"64","label":"Los
Angeles - Barcelona, 2030-01-01 20:00:00.0 - 2030-01-02 14:00:00.0, <<EUR 500000.00>>, Lorem ipsum dolor sit ame
","selected":false,"sealed":true},{"key":"67","label":"New York - London, 2031-12-31 02:00:00.0 - 2031-12-31 11:23:00.0, <<EUR 24183.71>>, لوريم إيبسوم
","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], id=0, isDraftMode=true, lastNibble=, lastNibble$error=You can't publish this booking because it does not have
the last nibble., locatorCode=R8GJ4NDI, locatorCode$error=This locator already exists., passenger=0, passenger$error=The booking needs at least 1 passenger., passengerChoices=
[{"key":"104","label":"MFURBWKGN - Rei Fujisaki (天寫)","selected":false,"sealed":true},{"key":"105","label":"353252 - Pasajero 4","selected":false,"sealed":true},{"key":"108","label":"7777777 - Pasajero
baneado 1","selected":false,"sealed":true},{"key":"102","label":"R8GJ4NDI3 - ɗosa
ɱartínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz.","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], purchaseMoment=2025/01/01 00:00, service=158, travelClass=0,
travelClass$error=Choose a travel class. May not be null., travelClasses=[{"key":"0","label":"----","selected":true,"sealed":true},{"key":"ECONOMY","label":"ECONOMY","selected":false,"sealed":true},
{"key":"BUSINESS","label":"BUSINESS","selected":false,"sealed":true}], updatedBooking=false, version=0}', but got '{flight=0, flight$error=Choose a flight. May not be null., flightChoices=
[{"key":"60","label":"London - Barcelona, 2024-03-12 06:00:00.0 - 2024-12-10 09:00:00.0, <<EUR 1000.00>>, LONG FLIGHT","selected":false,"sealed":true},{"key":"64","label":"Los Angeles - Barcelona, 2030-01-
01 20:00:00.0 - 2030-01-02 14:00:00.0, <<EUR 500000.00>>, Lorem ipsum dolor sit ame
","selected":false,"sealed":true},{"key":"67","label":"New York - London, 2031-12-31 02:00:00.0 - 2031-12-31 11:23:00.0, <<EUR 24183.71>>, لوريم إيبسوم
","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], id=0, isDraftMode=true, lastNibble=, lastNibble$error=You can't publish this booking because it does not have
the last nibble., locatorCode=R8GJ4NDI, passenger=0, passenger$error=The booking needs at least 1 passenger., passengerChoices=[{"key":"104","label":"MFURBWKGN - Rei Fujisaki (天
寫)","selected":false,"sealed":true},{"key":"105","label":"353252 - Pasajero 4","selected":false,"sealed":true},{"key":"108","label":"7777777 - Pasajero baneado 1","selected":false,"sealed":true},
{"key":"102","label":"R8GJ4NDI3 - ɗosa
ɱartínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz.","selected":false,"sealed":true},{"key":"0","label":"----","selected":true,"sealed":true}], purchaseMoment=2025/01/01 00:00, service=158, travelClass=0,
travelClass$error=Choose a travel class. May not be null., travelClasses=[{"key":"0","label":"----","selected":true,"sealed":true},{"key":"ECONOMY","label":"ECONOMY","selected":false,"sealed":true},
{"key":"BUSINESS","label":"BUSINESS","selected":false,"sealed":true}], updatedBooking=false, version=0}'.
```

Figura 9: Fallo detectado

Si borramos la validación que impide que una reserva tenga el mismo localizador que otra, en los tests se detectará como que no se ha producido un error en ese campo cuando sí debiera de haberse producido.

- Mutación 4, cambiar un símbolo lógico en el authorise de CustomerPassengerCreateService:

```java
@Override
public void authorise() {
    boolean status;
    int masterId;
    int bookingId;
    Booking booking;

    status = super.getRequest().getPrincipal().hasRealmOfType(Customer.class);
    boolean hasMasterId = super.getRequest().hasData(CustomerPassengerController.MASTER_ID);
    boolean hasBooking = super.getRequest().hasData("booking");
    masterId = hasMasterId ? super.getRequest().getData(CustomerPassengerController.MASTER_ID, int.class) : 0;

    if (status && hasMasterId) {
        booking = this.repository.findBookingById(masterId);
        status = booking != null && super.getRequest().getPrincipal().hasRealm(booking.getCustomer()) && booking.getIsDraftMode();
    }

    if (status && hasBooking) {
        bookingId = super.getRequest().getData("booking", int.class);
        booking = this.repository.findBookingById(bookingId);
        status = bookingId == 0 || booking != null && super.getRequest().getPrincipal().hasRealm(booking.getCustomer()) && booking.getIsDraftMode();
        if (status && hasMasterId)
            status = bookingId == masterId;
    }

    super.getResponse().setAuthorised(status);

}
```

Figura 10: Estado inicial del código

```java
@Override
public void authorise() {
    boolean status;
    int masterId;
    int bookingId;
    Booking booking;

    status = super.getRequest().getPrincipal().hasRealmOfType(Customer.class);
    boolean hasMasterId = super.getRequest().hasData(CustomerPassengerController.MASTER_ID);
    boolean hasBooking = super.getRequest().hasData("booking");
    masterId = hasMasterId ? super.getRequest().getData(CustomerPassengerController.MASTER_ID, int.class) : 0;

    if (status && hasMasterId) {
        booking = this.repository.findBookingById(masterId);
        status = booking != null || super.getRequest().getPrincipal().hasRealm(booking.getCustomer()) && booking.getIsDraftMode();
    }

    if (status && hasBooking) {
        bookingId = super.getRequest().getData("booking", int.class);
        booking = this.repository.findBookingById(bookingId);
        status = bookingId == 0 || booking != null && super.getRequest().getPrincipal().hasRealm(booking.getCustomer()) && booking.getIsDraftMode();
        if (status && hasMasterId)
            status = bookingId == masterId;
    }

    super.getResponse().setAuthorised(status);

}
```

Figura 11: Mutación 4

Figura 12: Fallo detectado

Al cambiar el *and* por un *or* se están permitiendo más casos de los que se debieran, lo que explica la cantidad de peticiones en las que se esperaba un 500 pero se obtuvo un 200. Además, dicha condición está diseñada así para que no se intente hacer booking.getDraftMode() de un booking inexistente, lo cual da error como se puede ver.

- Mutación 5, en el CustomerPassengerListService, sustituir la llamada para mostrar solo los pasajeros de una reserva por mostrar todos los pasajeros de ese customer:

```java
@Override
public void load() {
    Collection<Passenger> passengers;
    int customerId;
    Integer bookingId;

    customerId = super.getRequest().getPrincipal().getActiveRealm().getId();

    if (super.getRequest().hasData(CustomerPassengerListService.MASTER_ID)) {
        bookingId = super.getRequest().getData(CustomerPassengerListService.MASTER_ID, int.class);
        passengers = this.repository.findPassengersByBookingId(bookingId);
    } else
        passengers = this.repository.findPassengersByCustomerId(customerId);

    super.getBuffer().addData(passengers);
}
```

Figura 13: Estado inicial del código

```java
@Override
public void load() {
    Collection<Passenger> passengers;
    int customerId;
    Integer bookingId;

    customerId = super.getRequest().getPrincipal().getActiveRealm().getId();

    if (super.getRequest().hasData(CustomerPassengerListService.MASTER_ID)) {
        bookingId = super.getRequest().getData(CustomerPassengerListService.MASTER_ID, int.class);
        passengers = this.repository.findPassengersByCustomerId(customerId);
    } else
        passengers = this.repository.findPassengersByCustomerId(customerId);

    super.getBuffer().addData(passengers);
}
```

Figura 14: Mutación 5

```
Resetting application (clear schema, populate sample, reset clock, reset randomiser).
Replaying .\src\test\resources\customer\passenger\list.safe...
FAILED GET /customer/passenger/list?bookingId=120 (request-id="8bc006ad-161e-4421-a011-0adf9738733a", input=""): Expected 'payload' to be '{booking=120, service=159, showCreate=true}', but got
'{booking=120, dateOfBirth[0]=2024/12/31 00:00, dateOfBirth[1]=2000/01/01 00:00, dateOfBirth[2]=2000/01/01 00:00, dateOfBirth[3]=1980/10/01 00:00, email[0]=uwu@gmail.com, email[1]=psg4@hotmail.com, email
[2]=psg7@hotmail.com, email[3]=eltioraperas@gmail.com, fullName[0]=Rei Fujisaki (天寫), fullName[1]=Pasajero 4, fullName[2]=Pasajero baneado 1, fullName[3]=Մosa
₥artínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz., id[0]=104, id[1]=105, id[2]=108, id[3]=102, isDraftMode[0]=x, isDraftMode[1]=x, isDraftMode[2]=x, isDraftMode[3]=✓, passportNumber[0]=MFURBWKGN,
passportNumber[1]=353252, passportNumber[2]=7777777, passportNumber[3]=R8GJ4NDI3, service=159, showCreate=true, specialNeeds[0]=Peanut allergy, specialNeeds[1]=N/A, specialNeeds[2]=ª, specialNeeds[3]=Մ₥天
寫.ÁAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA, version[0]=0, version[1]=0, version[2]=0, version[3]=0}'.
FAILED GET /customer/passenger/list?bookingId=116 (request-id="4d20a7ba-6e7d-4df0-b920-1deb61050cc5", input=""): Expected 'payload' to be '{booking=116, dateOfBirth[0]=2024/12/31 00:00, dateOfBirth
[1]=2000/01/01 00:00, dateOfBirth[2]=2000/01/01 00:00, email[0]=uwu@gmail.com, email[1]=psg4@hotmail.com, email[2]=psg7@hotmail.com, fullName[0]=Rei Fujisaki (天寫), fullName[1]=Pasajero 4, fullName
[2]=Pasajero baneado 1, id[0]=104, id[1]=105, id[2]=108, isDraftMode[0]=x, isDraftMode[1]=x, isDraftMode[2]=x, passportNumber[0]=MFURBWKGN, passportNumber[1]=353252, passportNumber[2]=7777777, service=158,
showCreate=false, specialNeeds[0]=Peanut allergy, specialNeeds[1]=N/A, specialNeeds[2]=ª, version[0]=0, version[1]=0, version[2]=0}', but got '{booking=116, dateOfBirth[0]=2024/12/31 00:00, dateOfBirth
[1]=2000/01/01 00:00, dateOfBirth[2]=2000/01/01 00:00, dateOfBirth[3]=1980/10/01 00:00, email[0]=uwu@gmail.com, email[1]=psg4@hotmail.com, email[2]=psg7@hotmail.com, email[3]=eltioraperas@gmail.com,
fullName[0]=Rei Fujisaki (天寫), fullName[1]=Pasajero 4, fullName[2]=Pasajero baneado 1, fullName[3]=Մosa
₥artínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz., id[0]=104, id[1]=105, id[2]=108, id[3]=102, isDraftMode[0]=x, isDraftMode[1]=x, isDraftMode[2]=x, isDraftMode[3]=✓, passportNumber[0]=MFURBWKGN,
passportNumber[1]=353252, passportNumber[2]=7777777, passportNumber[3]=R8GJ4NDI3, service=158, showCreate=false, specialNeeds[0]=Peanut allergy, specialNeeds[1]=N/A, specialNeeds[2]=ª, specialNeeds[3]=Մ₥天
寫.ÁAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA, version[0]=0, version[1]=0, version[2]=0, version[3]=0}'.
FAILED GET /customer/passenger/list?bookingId=119 (request-id="08d827b1-67b5-45fd-8348-909a038ee876", input=""): Expected 'payload' to be '{booking=119, dateOfBirth[0]=1980/10/01 00:00, dateOfBirth
[1]=2024/12/31 00:00, email[0]=eltioraperas@gmail.com, email[1]=uwu@gmail.com, fullName[0]=Մosa
₥artínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz., fullName[1]=Rei Fujisaki (天寫), id[0]=102, id[1]=104, isDraftMode[0]=✓, isDraftMode[1]=x, passportNumber[0]=R8GJ4NDI3, passportNumber[1]=MFURBWKGN,
service=160, showCreate=true, specialNeeds[0]=Մ₥天寫.ÁAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA, specialNeeds[1]=Peanut allergy, version[0]=0, version[1]=0}', but got '{booking=119, dateOfBirth
[0]=2024/12/31 00:00, dateOfBirth[1]=2000/01/01 00:00, dateOfBirth[2]=2000/01/01 00:00, dateOfBirth[3]=1980/10/01 00:00, email[0]=uwu@gmail.com, email[1]=psg4@hotmail.com, email[2]=psg7@hotmail.com, email
[3]=eltioraperas@gmail.com, fullName[0]=Rei Fujisaki (天寫), fullName[1]=Pasajero 4, fullName[2]=Pasajero baneado 1, fullName[3]=Մosa
₥artínezzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz., id[0]=104, id[1]=105, id[2]=108, id[3]=102, isDraftMode[0]=x, isDraftMode[1]=x, isDraftMode[2]=x, isDraftMode[3]=✓, passportNumber[0]=MFURBWKGN,
passportNumber[1]=353252, passportNumber[2]=7777777, passportNumber[3]=R8GJ4NDI3, service=160, showCreate=true, specialNeeds[0]=Peanut allergy, specialNeeds[1]=N/A, specialNeeds[2]=ª, specialNeeds[3]=Մ₥天
寫.ÁAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA, version[0]=0, version[1]=0, version[2]=0, version[3]=0}'.
FAILED GET /customer/passenger/list?bookingId=117 (request-id="3d4cdafd-d65d-4869-9640-be2973754287", input=""): Expected 'payload' to be '{booking=117, dateOfBirth[0]=1900/01/01 00:00, dateOfBirth
[1]=2000/01/01 00:00, email[0]=estonoesunacuentafalsa@hotmail.com, email[1]=psg5@hotmail.com, fullName[0]=ɺ, fullName[1]=Pasajero 5, id[0]=103, id[1]=106, isDraftMode[0]=⍰, isDraftMode[1]=x, passportNumber
[0]=098765, passportNumber[1]=MFURBAVRW, service=159, showCreate=true, specialNeeds[0]=N/A, specialNeeds[1]=N/A, version[0]=0, version[1]=0}', but got '{booking=117, dateOfBirth[0]=2000/01/01 00:00,
dateOfBirth[1]=2000/01/01 00:00, dateOfBirth[2]=2000/01/01 00:00, dateOfBirth[3]=1900/01/01 00:00, email[0]=psg5@hotmail.com, email[1]=psg6@hotmail.com, email[2]=psg8@hotmail.com, email
[3]=estonoesunacuentafalsa@hotmail.com, fullName[0]=Pasajero 5, fullName[1]=Pasajero 6, fullName[2]=Pasajero baneado 2, fullName[3]=ɺ, id[0]=106, id[1]=107, id[2]=109, id[3]=103, isDraftMode[0]=x,
isDraftMode[1]=x, isDraftMode[2]=x, isDraftMode[3]=⍰, passportNumber[0]=MFURBAVRW, passportNumber[1]=3A3G5Z, passportNumber[2]=88888888, passportNumber[3]=098765, service=159, showCreate=true, specialNeeds
[0]=N/A, specialNeeds[1]=N/A, specialNeeds[2]=Tendencias homicidas, specialNeeds[3]=N/A, version[0]=0, version[1]=0, version[2]=0, version[3]=0}'.
```

Figura 15: Fallo detectado

Como se puede ver, al tratar de listar solo los pasajeros de la reserva, se obtienen más datos de los esperados.

Conclusiones

Para concluir, no se han llegado a encontrar fallos significativos en el código durante los tests. Sin embargo, sí se han podido realizar varias optimizaciones de código, tanto mejorar el que era necesario en la aplicación como borrar líneas que sobraban. Con el testeo de rendimiento, también he podido observar que mi ordenador no es el más óptimo para estas operaciones.

Bibliografía

Intencionalmente en blanco