|  |
| --- |
| José Coronil Álvarez (joscoralv@alum.us.es)  Miguel García Vizcaíno (miggarviz@alum.us.es)  Javier Ignacio Milá de la Roca Dos Santos (javmildos@alum.us.es)  Emilio Manuel Vázquez Cruz (emivazcru@alum.us.es)  Group C1.028  5-16-2025 |

|  |
| --- |
| Acme AirNav Solutions, Inc. |
| **Testing Report** |
| https://github.com/Emilio-115/DP2-Acme-ANS |



Contents

[Executive Summary 2](#_Toc199169078)

[Revision Table 2](#_Toc199169079)

[Introduction 3](#_Toc199169080)

[Content 3](#_Toc199169081)

[Tests Cases 3](#_Toc199169082)

[Flight 3](#_Toc199169083)

[Legs 5](#_Toc199169084)

[Test methodology 6](#_Toc199169085)

[Test Coverage 7](#_Toc199169086)

[Flight 7](#_Toc199169087)

[Legs 8](#_Toc199169088)

[Performance Analysis 8](#_Toc199169089)

[Before Indices 9](#_Toc199169090)

[After indices 10](#_Toc199169091)

[Comparison 11](#_Toc199169092)

[Conclusions 11](#_Toc199169093)

[Bibliography 11](#_Toc199169094)

# Executive Summary

This document is the test report for student 1, focusing on airline manager features testing, an extensive suite of tests was produced, catching some bugs in the process, and covering most of the code base.

After a performance analysis comparing database indexing and not, we concluded that we could not see any improvement in the performance when using indexing probably due to small database size.

# Revision Table

|  |  |  |
| --- | --- | --- |
| Revision number | Date | Description |
| 1 | 26/05/2025 | Initial version |

# Introduction

This document is the testing report for Student 1 it will cover what tests cases were developed for the project specifically for airline manager features, how good is the test coverage and what are the execution branches that were not achieved by testing and an analysis of performance comparing using and not indexing in the database

# Content

Following the guidelines provided, safe and hack test cases were recorded for airline manager features including everything related to flights and legs management aiming for a high coverage, there is a performance test and a comparison between using or not indices.

## Tests Cases

The following is a list of the test cases implemented, grouped by feature:

### Flight

* **List-show.safe**:  
  These test checks that an airline manager can list only their flights, can show a single flight in detail it was tested for published and draft flights. Because of the simple nature of list where we don’t expose anything user related and the list of flights shown is based on the current logged in user there are no hack cases for list. No bugs were found.
* **show.hack**:  
  These tests were made to ensure safety on the show feature, making sure a airline manager can only see information about their own flights, there are tests like trying to access to others manager fight or non-existing flights. No bugs were found.
* **create.safe**, **update.safe**, **delete.safe**, **publish.safe**:  
  These tests ensure you can properly use all the airline manager legal features, this includes all validation related to flights and their respective legs, like publishing without legs or with not published legs, leaving all fields empty to ensure the backend did not fail, also tested all these features with big set of permutations ensuring expected behavior on every one of them. Some bugs were found like the system allowing flights with no legs or the system allowing the deletion of flights with published legs.
* **create.hack**:  
  This tests that, only airline managers can create flights, and that changing the id of the request to an existing flight does not modify it. This last test case was found during testing and solved effectively.
* **update.hack**, **delete.hack**, **publish.hack**:  
  This are test cases for updating , deleting and publishing flights under non legal situation, they are grouped because of their similarities. All of them check that an airline manager can’t perform these actions for others manager flights, non-existing flights, and their own flights already published. No bugs where found.

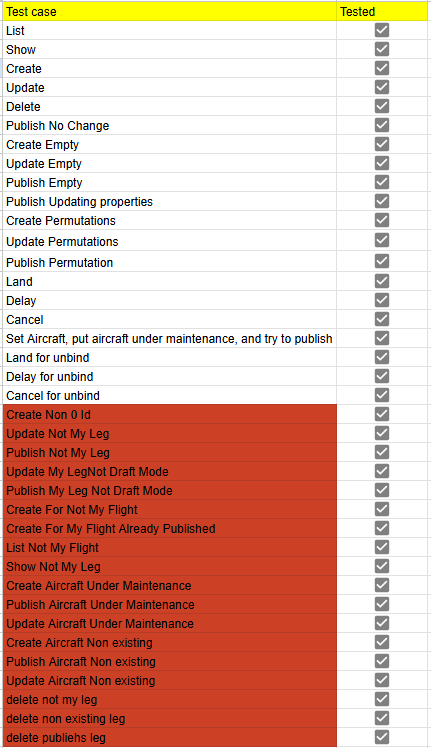
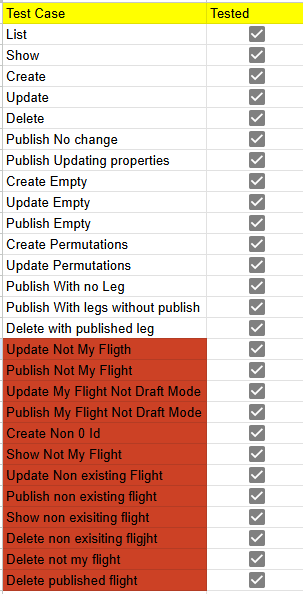
### Legs

* **list-show.safe**   
  These tests check that an airline manager can list legs of their flights, can show a single leg in detail it was tested for published and draft legs and published and draft flights. No bugs were found.
* **List-show.hack**:  
  These tests were made to ensure safety on the show and list features, making sure a airline manager can only see information about their own legs, there are tests like trying to access to others manager legs, others managers flights list of legs or non-existing legs. No bugs were found.
* **create.safe**, **update.safe**, **delete.safe**, **publish.safe**:  
  These tests ensure you can properly use all the airline manager legal features, this includes all validation related to legs, creating a leg in the past or leaving all fields empty to ensure the backend did not fail, also tested all these features with big set of permutations ensuring expected behavior on every one of them. Some changes were made to satisfy all requirements, the hardest one was about aircraft where an aircraft could be set to under maintenance while creating a leg with that same aircraft, the solution was setting it to null in frontend and making sure the airline manager solved it by changing the aircraft.
* **Create.hack, update.hack**, **delete.hack**, **publish.hack**:  
  This tests ensured the airline manager features related with legs were safe by checking different cases that should fail like creating legs for others manger flights, creating for publish flights or updating, deleting and publishing for others managers legs, non-existing legs and published legs. We also tested navigation hacking setting non-existing airports, non-existing aircrafts and aircrafts under maintenance. No bugs were found.
* **Extra.safe:**

These are extra tests made for improving test coverage, with specific cases like showing a past flight with a currently non available flight

## Test methodology

The test methodology followed was the one provided, but for making the process easier it was produced a test case checklist, where red tests were hacking tests, so we could have a record of what tests where done, what tests were not and what to rerecord when it was necessary, this helped not leaving behind any test case.



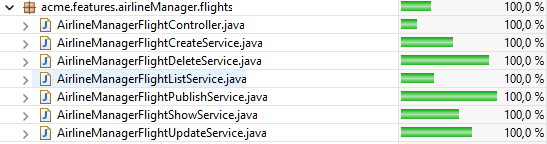
## Test Coverage

The following is a summary of the coverage achieved for all relevant files

### Flight

* **Validator** **(88,2%):**All the missing coverage comes from safety null checks that are never actually executed due to being an impossible branch of execution, all domain logic is 100% covered.
* **Services of features (100%)**

100% of coverage was successfully achieved in the all the services of the flight features





### Legs

* **Validator (92,7%):**  
  All the missing coverage comes from safety null checks that are never actually executed due to being an impossible branch of execution, all domain logic is 100% covered.
* **Controller**, **Delete Service**, **Update Service**, **List Service**, **Show Service**, **Publish Service (100%):**  
  100% of coverage was successfully achieved in these services of the leg features
* **Create Service (98,6%):**  
  The remaining coverage comes from the branch where it can’t find the flight in the database during the load, this is not possible because of it being checked before in the authorize.
* **Land Service, Delay Service, Cancel Service(99,2%)**

These services are not part of the requirements but it was the solution I got to the problem with the state of the leg, these have null checks that never fail so that’s the remaining coverage





## Performance Analysis

The performance analysis was produced following the instructions, for both indexed and not versions of the database, the objective of this section is to analyze both and compare them. This data was extracted from the execution of flight and legs tests.

### Before Indices

Before the indices, the testing resulted in a 95% confidence interval from 13.45ms to 16.59ms and a mean of 15.02ms. The request with the highest response time was /airline-manager/leg/create with 284.64ms.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Response Time* | |  |  |  |
|  |  |  |  |  |
| Media | 15.0240769 |  | Interval(ms) | 13.456585 |
| Error típico | 0.79867517 |  | Interval(s) | 0.01345658 |
| Mediana | 7.8009 |  |  |  |
| Moda | 2.1431 |  |  |  |
| Desviación estándar | 23.9202879 |  |  |  |
| Varianza de la muestra | 572.180175 |  |  |  |
| Curtosis | 39.8080217 |  |  |  |
| Coeficiente de asimetría | 5.0152059 |  |  |  |
| Rango | 283.3089 |  |  |  |
| Mínimo | 1.3349 |  |  |  |
| Máximo | 284.6438 |  |  |  |
| Suma | 13476.597 |  |  |  |
| Cuenta | 897 |  |  |  |
| Nivel de confianza(95.0%) | 1.56749197 |  |  |  |

### After indices

After the indices, the interval was from 13.90ms to 16.90ms, with a mean of 15.40ms. The request with the highest response time was /airline-manager/leg/list with 344.29ms

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Response Time* | |  |  |  |  |
|  |  |  | Interval(ms) | 13.9007298 | 16.90095717 |
| Media | 15.4008435 |  | Interval(s) | 0.01390073 | 0.016900957 |
| Error típico | 0.7643443 |  |  |  |  |
| Mediana | 7.9733 |  |  |  |  |
| Moda | 13.0304 |  |  |  |  |
| Desviación estándar | 22.8920799 |  |  |  |  |
| Varianza de la muestra | 524.047324 |  |  |  |  |
| Curtosis | 51.7650319 |  |  |  |  |
| Coeficiente de asimetría | 5.08670198 |  |  |  |  |
| Rango | 342.9816 |  |  |  |  |
| Mínimo | 1.3108 |  |  |  |  |
| Máximo | 344.2924 |  |  |  |  |
| Suma | 13814.5566 |  |  |  |  |
| Cuenta | 897 |  |  |  |  |
| Nivel de confianza(95.0%) | 1.50011369 |  |  |  |  |

### Comparison

The two-sample z-test for means resulted in a P(Z<=z) two-tail value of 0.733, which is over the alpha value for the 95% confidence level (0.05). With this in mind there is not enough evidence to ensure there is a significative change. Even with this we can see how the mean is higher after applying the indices.

|  |  |  |
| --- | --- | --- |
|  | *Before* | *After* |
| Media | 15.0240769 | 15.4008435 |
| Varianza (conocida) | 572.180175 | 524.04765 |
| Observaciones | 897 | 897 |
| Diferencia hipotética de las medias | 0 |  |
| z | -0.34081457 |  |
| P(Z<=z) una cola | 0.36662159 |  |
| Valor crítico de z (una cola) | 1.64485363 |  |
| P(Z<=z) dos colas | 0.73324318 |  |
| Valor crítico de z (dos colas) | 1.95996398 |  |

# Conclusions

We deeply tested all airline manager features to ensure maximum quality, having 100% tests passing, a really high percentage of test coverage and uncovered branches are not coverable.

After the performance analysis we can see a minor rise in execution time when using indices due to the cost of the indices being bigger than the advantage, they bring us because of the small database size.

# Bibliography

Intentionally blank.