**Analysis Report**

**Group**: C1.027  
**Repository**: <https://github.com/DP2-C1-027/AirNav-Logistics.git>

### **Group Members:**

1. **Garcia de Tejada Delgado, Jose**
   * Role: Developer
   * Email: josgardel8@alum.us.es
2. **Peñaloza Friqui, Nora**
   * Role: Developer
   * Email: norpennfri@alum.us.es
3. **Niza Cobo, Manuel Jesús**
   * Role: Manager, Developer
   * Email: mannizcob@alum.us.es
4. **Gomez Claraco, Nicolas**
   * Role: Developer
   * Email: nicgomcla@alum.us.es
5. **Campos Diez, Lucia**
   * Role: Tester, Developer
   * Email: luccamdie@alum.us.es

03/12/2025

**Table of Contents**

1. **Executive Summary**
2. **Revision Table**
3. **Introduction**
4. **Contents**
   1. **Analysis of the requirements of D01**
   2. **Analysis of the requirements of D02**
   3. **Analysis of the requirements of D03**
   4. **Final Analysis**
5. **Conclusions**
6. **Bibliography**

**Executive Summary**

This document provides an overview of the ersión report for the AirNav-Logistics ersión. It summarizes the key updates and adjustments made to the system ersión the ersión of the ersión requirements. While most tasks were straightforward and did not require extensive evaluation, some changes were necessary to improve the system’s accuracy and alignment with real-world scenarios.

Key updates include the implementation of attribute validation to ensure data consistency and accuracy, as well as adjustments to the data model to reflect the relationships between key entities more accurately. These changes enhance the system’s overall integrity and functionality, ensuring it meets the specified requirements.

The report highlights the importance of aligning the system’s structure with the project’s domain needs and maintaining consistency in data validation and entity relationships.

**Revision Table**

|  |  |  |
| --- | --- | --- |
| **Revision Number** | **Date** | **Description** |
| 1.0 | 02/19/2025 | Initial version of the report |
| 2.0 | 03/12/2025 | Update of the report with the analisis of the requirements of D02 |
| 3.0 | 03/29/2025 | Update of the report with the analisis of the requirements of D03 |
| 4.0 | 05/25/2025 | Final version of the report |
|  |  |  |
|  |  |  |
|  |  |  |

**Introduction**

The purpose of this document is to present an analysis of the tasks performed in the AirNav-Logistics project. However, given that the assigned tasks were straightforward and did not require significant analytical work, no substantial analyses were necessary.

**Contents**

* 1. **Analysis of the requirements of D01:**

No specific analysis records are included in this report as the individual tasks completed did not necessitate detailed requirement evaluations. Therefore, no requirement modifications or validations were needed.

* 1. **Analysis of the requirements of D02**

Regarding requirements 3 and 4, it has been necessary to create two validators: one for the Identifier attribute to verify that the attribute's letters indeed correspond to the initials, and another for the lastNibble attribute to verify, in case it is not null, that its length is exactly 4.

With regard to the validation of the attributes, I have based my work on an Excel file that Mr. Rafael Corchuelo Gil published in response to one of the forum posts (<https://ev.us.es/webapps/discussionboard/do/message?conf_id=_426211_1&forum_id=_253522_1&course_id=_89154_1&action=list_messages&nav=discussion_board&message_id=_461165_1>)

Finally, the last analysis concerned the relationship between Booking and Passenger, which I initially thought would be a ManyToOne relationship from Passenger to Booking. However, after further analysis, I determined that it was a ManyToMany relationship. Consequently, an intermediate entity BookingRecord was created, establishing a ManyToOne relationship with each of the original entities, since a passenger can be associated with multiple bookings, and a booking can include multiple passengers. This setup reflects real-world scenarios where passengers can have multiple bookings and a single booking can involve several passengers

* 1. **Analysis of the requirements of D03**

Regarding requirement 8, as indicated, the customer's bookings have been listed, and their details have been displayed. To show the passengers associated with a booking, I have found it convenient to add a button within the form view. When clicked, this button will redirect to a list containing all the passengers for that booking.

Additionally, for the customer’s convenience, selecting a passenger will display their details. If the passenger is not published, the option to update or publish them will be available.

Regarding the creation process, when selecting a flight, a dropdown menu will be shown displaying the available and published flights. This has not been implemented in the update process because I personally believe that if the booking is not going to be made for a specific flight, the best approach would be to delete it (provided it is not published, as with the update process) and create a new one for the desired flight.

Finally, a booking cannot be published until the last credit card nibble attribute has been saved and at least one associated passenger is published. If the passenger is not published, it would still be possible to update them, which would result in an indirect change to the booking — something that would not be possible once the booking is published.

Regarding requirement 9, all passengers have been listed and their details displayed. Additionally, it is possible to create and update passengers. Publishing is also allowed without any restrictions at the time of publishing, except that once published, the passenger can no longer be updated.

Since a Passenger is associated with a Customer through a ManyToOne relationship, a passenger can be created in two ways. The first is through the passenger interface, where the created passenger would not be associated with any booking. The second is within a specific booking, in which case the passenger would automatically become part of that booking.

Additionally, a new view has been created to allow the creation of new associations between passengers and bookings. This corresponds to the intermediate table BookingRecord. This view lists all existing pairs, allows the creation of new pairs (through dropdowns), and enables updates and deletions.

When implementing this, it was ensured that already published bookings do not appear, for the same reason mentioned earlier: if a booking is published and a new passenger is added, it would be considered a modification, which is not allowed. Furthermore, it has been designed so that there cannot be two entries for the same pair.

Regarding requirement 17, since creating a customer requires a code where the letters match the customer's initials, the most straightforward and intuitive approach has been to create a method that correctly generates the initials

Regarding requirement 19, when listing passengers within a booking, only the name and email are displayed. Further details are not shown since this is considered private information, which the administrator does not necessarily need to access.

* 1. **Final Analysis:**

As a recap, an analysis will be conducted on the Booking and Passenger sections of the website.

First of all, it is important to note that, in addition to the Booking and Passenger sections, there is also a BookingRecord at the code level, although the user is unaware of its existence.

Starting with the Booking section, the booking price has been set to correspond to the price of the selected flight multiplied by the number of passengers included in the booking. Additionally, only flights that are published and scheduled for a future date—relative to the moment the user is creating the booking—can be selected.

In this way, it has been decided that, for greater ease for the end user, the current date at the time the booking is created will be automatically set in the purchaseMoment attribute.

When updating a booking, the flight cannot be changed, and therefore neither can the purchaseMoment. If a flight change is desired, it is easier to delete the existing booking and create a new one, rather than modifying the associated entries and passengers

Regarding publishing, a booking can only be published if it has the lastNibble set and has associated passengers who are also published. Furthermore, publishing a booking does not make it visible to other users, as it contains personal information such as passenger email addresses or home addresses, which should not be accessible to just anyone.

Within each booking, the associated passengers can be viewed, and new passengers can be added either by creating them through the form or by simply linking existing passengers. In the case of existing passengers, they can only be added to a booking if they belong to the same user. However, it does not matter whether these passengers are published or not at the time of addition, since ultimately, in order for the booking to be published, all associated passengers must also be published.

Booking and Passenger are separate entities, meaning that deleting one does not directly affect the other—only the intermediate entity, BookingRecord, is impacted. With this in mind, if a non-published booking or passenger is deleted, the corresponding entry is removed along with any BookingRecord relationships it had. However, the deletion does not affect the related bookings or passengers themselves. Similarly, when deleting a booking, all associated BookingRecord entries are also deleted, but the passengers linked to that booking are not removed.

A user can create, delete, and update a passenger. The only special case is that passengers can be created in two ways: the first is through the Passenger window, where the user fills out a form and the passenger is added to the user’s passenger list; the second is within the Booking window, where creating a passenger not only adds them to the user’s passenger list but also immediately associates them with that booking.

Finally, within a passenger’s profile, you can view a list of all bookings in which the passenger is included. If a booking is not published, the user can delete the association (i.e., the corresponding BookingRecord), effectively removing the passenger from that booking.

To conclude, the URLs of the form entries used as a reference during the work are included.:

Thread: [D03-S02-R08] & [D03-S02-R09] Comportamiento de los delete

<https://ev.us.es/webapps/discussionboard/do/message?action=list_messages&course_id=_89154_1&nav=discussion_board&conf_id=_426211_1&forum_id=_253522_1&message_id=_470629_1>

Thread: [D&T] -[D03- S02- R08] - Clarification on Flight Identification for Bookings

<https://ev.us.es/webapps/discussionboard/do/message?action=list_messages&course_id=_89154_1&nav=discussion_board&conf_id=_426211_1&forum_id=_253522_1&message_id=_465660_1>

Thread: [D&T] -[D03- S02- R08/09]

<https://ev.us.es/webapps/discussionboard/do/message?action=list_messages&course_id=_89154_1&nav=discussion_board&conf_id=_426211_1&forum_id=_253522_1&message_id=_465248_1>

Thread: [D&T] Duda sobre el atributo price de booking

<https://ev.us.es/webapps/discussionboard/do/message?action=list_messages&course_id=_89154_1&nav=discussion_board&conf_id=_426211_1&forum_id=_253522_1&message_id=_463588_1>

Thread: [D&T][D02-S02-IR4] & [D02-S02-IR5]: Relación de las entidades

<https://ev.us.es/webapps/discussionboard/do/message?action=list_messages&course_id=_89154_1&nav=discussion_board&conf_id=_426211_1&forum_id=_253522_1&message_id=_461014_1>

**Conclusions**

In conclusion, the analysis of the AirNav-Logistics project tasks revealed that most were straightforward, with validations for the Identifier and lastNibble attributes based on an existing Excel guide. The relationship between Booking and Passenger was adjusted from ManyToOne to ManyToMany, leading to the creation of the BookingRecord intermediate entity to better reflect real-world scenarios. User experience improvements include a button in the booking form view to list associated passengers, an intuitive dropdown for selecting published flights when creating bookings, and a method to automatically generate customer initials. Additionally, a new view was added to manage BookingRecord entries, ensuring published bookings remain unchanged and preventing duplicate entries. These enhancements improve both system security and usability, ensuring a robust and efficient experience.

**Bibliography**

Intentionally blank.