# Requirements – Student #2

Please, fill in the following form, make sure that you have ticked the requirements that you consider fulfilled, save this document, **and attach it in its original format (.docx)** to every deliverable. Regarding your ID, please keep only four random digits and mask the others using an asterisk. **Please, note that this document must be edited with the desktop version of Word since the web version does not properly support forms.** Attaching this document entails that you are the authors of the work delivered, you have not cheated in any way, and you have read and understood the information delivered regarding the subject, with a special emphasis on the methodological guidelines and how your work is going to be graded. Make sure that your project works well with the latest version of the development framework.

|  |
| --- |
| **Group:** C2.040 |
| **Repository:** https://github.com/DP2-2025-C1-040/Acme-ANS |
| **Student #2**  **ID Number:** \*\*\*\*4667  **UVUS:**  josporhue  **Name:**  Portela Huerta, José María  **Roles:**  Developer |
| **Date:** Seville October 15, 2025 |
|  |

# MANDATORY Deliverable D01: introduction

## Information requirements

Intentionally blank.

## Functional requirements

1. Modify the anonymous menu so that it shows an option that takes the browser to the home page of your favourite web site. The title must read as follows: “〈id-number〉: 〈surname〉, 〈name〉”, where “〈id-number〉” denotes your DNI, NIE, or passport number, “〈surname〉” denotes your surname/s, and “〈name〉” denotes your name/s.

X

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Provide a link to your planning dashboard in GitHub to review the tasks, their current status, and your schedule.

X

# MANDATORY Deliverable D02: data models

## Information requirements

1. **Customers** are the people who purchase flights. The system must store the following data about them: an **identifier** (unique, pattern "^[A-Z]{2-3}\d{6}$", where the first two or three letters correspond to their initials), a **phone number** (pattern "^\+?\d{6,15}$"), a **physical address** (up to 255 characters), plus a **city** and a **country** (both up to 50 characters). Optionally, customers may have some **earned points** (up to 500k points).

X

1. A **booking** is a reservation made by a **customer** to purchase a **flight**, guaranteeing some seats on a specific itinerary and associating some **passengers**' details with the trip. The system must manage the following information for each **booking**: a **locator code** (unique, pattern "^[A-Z0-9]{6,8}$"), a **purchase moment** (in the past), a **travel** **class** ("ECONOMY", BUSINESS"), and a **price**. Optionally, the system should record the **last nibble** of the credit card used for payment.

X

1. A **passenger** is an individual who takes a flight and he or she must be registered in the corresponding booking. The system must store the following data about passengers: a **full name** (shorter than 256 characters), an **email**, a **passport number** (pattern “^[A-Z0-9]{6,9}$”), a **date of birth**, and, optionally, his or her **special needs** (shorter than 51 characters).

X

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

1. Produce assorted sample data to test your application informally. The data must include two **customer** accounts with credentials “**customer1**/**customer1**” and “**customer2**/**customer2**”. Create an additional customer account with credentials “**customer3/ customer3”** that represents a customer with only profile data.

X

## Managerial requirements

1. Provide a link to your planning dashboard in GitHub to review the tasks, their current status, and your schedule.

X

# MANDATORY Deliverable D03: implementing features

## Information requirements

Intentionally blank.

## Functional requirements

1. Operations by **customers** on **bookings**:

* List their bookings.
* Show the details of their bookings and the associated passengers, if any.
* Create or update their bookings. Bookings can be updated as long as they have not been published. A booking can be published only when the last credit card nibble has been stored.

X Segunda convocatoria: este requisito no ha sido corregido, sigue teniendo el mismo problema, se muestran flights que ya han empezado. Un flight se considera que ha empezado si alguna leg ya ha tenido lugar. Muestro a continuación un ejemplo. Éstos son los flights que se muestran en el desplegable a la hora de realizar un booking:

A screenshot of a computer

AI-generated content may be incorrect.

Se muestran todos los flights publicados, pero no se tiene en cuenta si alguno ha comenzado. Un vuelo ha comenzado si su primera leg ya está en el pasado. Podemos comprobar que los vuelos con id del 88 al 92 están publicados en la siguiente captura:

A screenshot of a computer

AI-generated content may be incorrect.

A continuación, vamos a mostrar las legs en la base de datos únicamente de estos vuelos (aquellos con id del 88 al 92):

A screenshot of a computer

AI-generated content may be incorrect.

Se puede comprobar que hay flights publicados en la BD con legs que aún no están publicadas, algo que ya reporté que no era correcto porque estos datos son inconsistentes (aunque sea responsabilidad del estudiante 1). No obstante, podemos comprobar que tanto el flight con id 88 tiene legs en el pasado, lo que significa que ya ha empezado y por tanto no se puede reservar. El flight con id 89 tiene una leg (id 213) que también ha comenzado. Lo mismo para el resto de flights.

Se muestran todos los vuelos publicados independientemente de si sus legs están en el pasado o en el futuro. Además, hay inconsistencia en los datos, hay 14 flights publicados de los cuales solo 4 o 5 tienen legs. Un flight no puede estar publicado si no tiene legs. Esto no es responsabilidad del estudiante #2 en este caso, simplemente lo indico para que en el futuro, se tenga en cuenta que sólo flights publicados con legs publicadas pueden mostrarse en la lista deplegable. Salvo unas pocas legs, el resto están en el pasado y por tanto los flights no deberían de mostrarse en la lista desplegable.

Para realizar bien este requisito se ha modificado el leg.csv a valores correctos según la corrección, además de dos cambios importantes, en el repositorio y clases afectadas que incluían los métodos cambiados:

1. Se ha añadido findAllPasportsForBooking, con el fin de que calcule los pasaportes que incluye una reserva concreta, para que, con el método findRepeatedPassportsForFlight se vea adecuadamente cuáles son los pasaportes repetidos. Con ello se consigue la validación de que en un vuelo no se puede repetir el número de pasaporte:  
     
   SELECT br.passenger.passportNumber FROM BookingRecord br WHERE br.booking.id = :id
2. Se ha añadido findRepeatedPassportsForFlight con el fin de que calcule los pasaportes existentes en reservas publicadas, para el mismo vuelo que la que se pretende publicar. Con ello, se consigue evitar número de pasaporte duplicado para dicho vuelo, una vez se publica, consiguiéndose que solo 1 pasajero del mundo real pueda tener un asiento del vuelo, con la siguiente query:  
     
   SELECT br.passenger.passportNumber FROM BookingRecord br WHERE br.booking.flight.id = :id AND br.booking.draftMode = FALSE AND br.passenger.passportNumber IN :passports
3. Se ha añadido la validación que usa los métodos anteriores, mostrando los pasaportes repetidos, para que el usuario pueda quitarlos de la lista y realizar la reserva adecuadamente, avisando así quienes ya tienen, para ese vuelo, una reserva publicada. Se ha decidido realizar solo en la publicación, para que se pueda hacer ediciones en todo momento de la reserva, hasta que se tenga que confirmar la información.
4. Se ha cambiado findOneFlightPublishedById a que ahora se quede solo con los vuelos que solo incluyan todas sus legs a futuro, cambiando a una Query más avanzada, más concretamente:  
     
   SELECT f FROM Flight f WHERE f.id = :flightId AND f.draftMode = FALSE AND not exists(SELECT l FROM Leg l WHERE l.flight.id = f.id and ((l.draftMode = true) or (:now > l.scheduledDeparture)))  
     
   Ahora solo coje una única vez el vuelo con la id que está buscando, y tiene que estar publicado, con todas sus legs publicadas y fecha de salida posterior al momento actual, calculado en cada uno de los servicios. Este método es necesario editarlo para poder conseguir una correcta prevención de hacking por asignación malintencionada de un vuelo erróneo, según la corrección del Second Call.
5. Se ha cambiado findAllFlights a que ahora recoja los vuelos que contengan todas sus legs en el futuro, cambiando a una Query más avanzada, más concretamente:  
     
   SELECT f FROM Flight f WHERE f.draftMode = FALSE AND not exists(SELECT l FROM Leg l WHERE l.flight.id = f.id and ((l.draftMode = true) or (:now > l.scheduledDeparture)))  
     
   Ahora solo coje una única vez cada vuelo que estén publicados, con todas sus legs publicadas y fecha de salida posterior al momento actual, calculado en cada uno de los servicios. Con este cambio nos aseguramos que solo se recojan los vuelos correctos y sean los únicos que se muestren al usuario final, evitando vuelos ya empezados.
6. Operations by **customers** on **passengers**:

* List the passengers in their bookings.
* Show the details of their passengers.
* Create a passenger and record the information related to that passenger.
* Update a passenger as long as it has not been published.

X

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Provide a link to your planning dashboard in GitHub to review the tasks, their current status, and your schedule.

X

# MANDATORY Deliverable D04: formal testing

## Information requirements

1. Create appropriate indices for your entities, if required.

X

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

1. Produce a test suite for Requirements #8 and #9.

X

## Managerial requirements

1. Provide a link to your planning dashboard in GitHub to review the tasks, their current status, and your schedule.

X

1. Produce a testing report.

X

# SUPPLEMENTARY I Deliverable D01: introduction

## Information requirements

Intentionally blank.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

Intentionally blank.

# SUPPLEMENTARY I Deliverable D02: data models

## Information requirements

1. The system must handle **customers** **dashboards** with the following **indicators**:

* The last five destinations.
* The money spent in bookings during the last year.
* Their number of bookings grouped by travel class.
* Count, average, minimum, maximum, and standard deviation of the cost of their bookings in the last five years.
* Count, average, minimum, maximum, and standard deviation of the number of passengers in their bookings.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Produce a UML domain model regarding the information requirements.

# SUPPLEMENTARY I Deliverable D03: implementing features

## Information requirements

Intentionally blank.

## Functional requirements

1. Operations by **anonymous principals** on user **accounts**:

* Sign up to the system and become a customer.

1. Operations by **customers** on user **accounts**:

* Update their profiles.

1. Operations by **administrators** on **bookings**:

* List the bookings in the system that are published.
* Show the details of the bookings (including the passengers).

1. Operations by **customer** on **dashboards**:

* Show their customer dashboards.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Provide a link to a video in which you informally test requirement #8 and #9. Videos should not exceed 10 minutes in length and must be stored at the USE's facilities.

# SUPPLEMENTARY I Deliverable D04: formal testing

## Information requirements

Intentionally blank.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Perform five mutations in your code and report on the results.

## Managerial requirements

1. Produce a lint report.

# SUPPLEMENTARY II Deliverable D01: introduction

## Information requirements

Intentionally blank.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Produce an analysis report.

X

1. Produce a planning and progress report.

X

# SUPPLEMENTARY II Deliverable D02: data models

## Information requirements

1. The system must include a board to recommend something in the city and/or country of a given airport. **Recommendations** can be about experiences, activities, restaurants, accommodation or any other thing that a person may find interesting at the destination. A web service must be used to populate this entity with information about recommendations. Thus, the exact data to store depends on the chosen service, and it is the students' responsibility to define them accordingly. It is also the students’ responsibility to find the appropriate service; no implicit or explicit liabilities shall be covered by the University of Seville or their individual affiliates if the students contract pay-per-use services!  The students are strongly advised to ensure that the service they choose is free of charge.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Produce an analysis report.

1. Produce a planning and progress report.

# SUPPLEMENTARY II Deliverable D03: implementing features

## Information requirements

Intentionally blank.

## Functional requirements

1. Operations by **customers** on **recommendations**:

* List recommendations related to final destinations in their bookings.

1. Operations by **administrators** on **recommendations**:

* Populate the database with recommendations somehow related to locations (e.g., related to cities or countries).

## Non-functional requirements

Intentionally blank.

## Testing requirements

Intentionally blank.

## Managerial requirements

1. Produce an analysis report.

1. Produce a planning and progress report.

# SUPPLEMENTARY II Deliverable D04: formal testing

## Information requirements

Intentionally blank.

## Functional requirements

Intentionally blank.

## Non-functional requirements

Intentionally blank.

## Testing requirements

1. Produce as a complete test suite as possible for Requirement #29 ensuring that the API is properly mocked.

## Managerial requirements

1. Produce an analysis report.

1. Produce a planning and progress report.