ACME AirNav Solutions



D02 ANALYSIS REPORT – Student #2

Grupo: C1.050

Miembros: Cristina Fernández Chica ([criferchi@alum.us.es](mailto:criferchi@alum.us.es)), Ángel Amo Sánchez ([angamosan@alum.us.es](mailto:angamosan@alum.us.es)), Candela Jazmín Gutiérrez González ([cangutgon@alum.us.es](mailto:cangutgon@alum.us.es)), Marta Aguilar Morcillo ([maragumor@alum.us.es](mailto:maragumor@alum.us.es)) y Luis Emmanuel Chávez Malavé ([luichamal@alum.us.es](mailto:luichamal@alum.us.es))

Repositorio: <https://github.com/Cristinafernandezchica/Acme-ANS>

Planning dashboard: <https://github.com/users/Cristinafernandezchica/projects/1/views/1>

Sevilla 13 marzo, 2025

**TABLA DE CONTENIDOS**

[**Resumen Ejecutivo** 3](#_Toc199173324)

[**Tabla de Revisiones** 3](#_Toc199173325)

[**Introducción** 4](#_Toc199173326)

[**Conflictos** 4](#_Toc199173327)

[**Conflicto 1.** Atributo “*passportNumber*” no único 4](#_Toc199173328)

[**Conflicto 2.** Propiedad “*price*” con valor fijo o @Transient 5](#_Toc199173329)

[**Conflicto 3.** El atributo “*identifier*” no cambia al modificar el nombre del usuario 6](#_Toc199173330)

[**Conclusión** 7](#_Toc199173331)

[**Bibliografía** 7](#_Toc199173332)

#### **Resumen Ejecutivo**

Este documento identifica requisitos específicos que necesitan revisión debido a inconsistencias, ambigüedades o carencias en su formulación. Para cada requisito analizado, se han documentado las conclusiones obtenidas, las decisiones tomadas para su ajuste y un enlace a la validación realizada por el docente.

Está estructurado para proporcionar una visión clara y organizada de los aspectos analizados, garantizando la trazabilidad de las decisiones tomadas y promoviendo una mejor comprensión de los ajustes realizados en los requisitos. Todo con el fin de asegurar la claridad y la viabilidad de los requisitos, facilitando un desarrollo eficiente y alineado con los objetivos del proyecto.

#### **Tabla de Revisiones**

|  |  |  |  |
| --- | --- | --- | --- |
| **Número de revisión** | **Fecha** | **Descripción de revisión** | **Autor** |
| 1.0 | 13/03/2025 | Se ha revisado que todos los apartados han sido cubiertos y con toda la información necesaria. | Ángel Amo Sánchez |

#### **Introducción**

Este informe documenta y analiza los requisitos ambiguos o inconsistentes dentro del marco del proyecto. La evaluación detallada de estos requisitos permite detectar posibles mejoras y asegurar su correcta definición, minimizando riesgos en etapas posteriores del desarrollo.

Para ello, se han seleccionado únicamente aquellos requisitos que requieren análisis y ajustes, omitiendo aquellos que no presentan inconvenientes. En cada registro de análisis se incluye la copia exacta del requisito afectado, las conclusiones obtenidas tras su evaluación, las decisiones tomadas para su corrección y un enlace a la validación realizada por el docente.

#### **Conflictos**

##### **Conflicto 1.** Atributo “*passportNumber*” no único

**Descripción del conflicto:**

A la hora de definir el atributo surgió la duda sobre si considerarlo como único o no, pues, es cierto que como tal un numero de pasaporte es un identificador único para una persona. Sin embargo, nos encontramos ante ciertos matices.

**Alternativas de solución evaluadas**:

1. Interpretar el “*passportNumber*” como único a nivel de entidad.

* **Ventajas**: Tener un atributo único con el que poder diferenciar un pasajero de otro a nivel de entidad
* **Desventajas**: No sería posible realizar reservas diferentes en clientes diferentes para una misma persona al ya estar registrada en el sistema. Es poco realista.

1. Interpretar el “*passportNumber*” como único a nivel de clientes.

* **Ventajas**: Podría registrarse la misma persona en diferentes clientes, asegurando la facilidad y flexibilidad de poder elegir libremente tus pasajeros.
* **Desventajas**: La necesidad de tener que comprobar si este pasajero es único, pero entre clientes, pudiendo causar confusión e inconsistencias.

**Justificación de la solución adoptada**

Finalmente optamos por la segunda alternativa, ya que extrapolando esta situación a la vida real es cierto que varias personas pueden ser clientes del servicio y a la misma vez poder realizar reservas de la misma persona en cuentas diferentes (como restricción si está implementada que sea único, pero a nivel de cliente, es decir, no es posible repetir los pasaportes en un mismo cliente, pero si entre clientes diferentes).

**Conformidad de la solución adoptada**

La solución ha sido aprobada por el profesor a lo largo de las prácticas de laboratorio.

##### **Conflicto 2.** Propiedad “*price*” con valor fijo o @Transient

**Descripción del conflicto:**

Al definir el precio de una reserva este simplemente se fija como un atributo más en *Booking*. Sin embargo, el cliente no puede elegir el valor de la reserva, por lo tanto, esto hacia que fuera necesario que la propiedad obtuviera el valor de alguna manera.

**Alternativas de solución evaluadas**:

**1. Crear un atributo número de pasajeros para que este se multiplique por el precio del vuelo y calcular el precio de la reserva.**

**2. Cambiar el atributo “*Price*” para que este sea el precio del vuelo por persona** (el mismo precio que el que indica el vuelo).

**3. Calcular el precio multiplicando el precio del vuelo por el número de pasajeros asociado a esa reserva**.

**Justificación de la solución adoptada**

Finalmente opté por la tercera opción ya que extrapolando esta situación a la vida real es la más realista. Pues, en una reserva no tendría sentido tener como precio un valor estático al poder añadir tantas personas como quisiésemos sin coste adicional alguno. Y la misma vez, teniendo un atributo para el precio, crear otro diferente más no tendría sentido (además de que no se ha pedido que esa información se guarde, pudiendo incumplir el pliego de requisitos).

**Conformidad de la solución adoptada**

La solución adoptada finalmente está respaldada por el foro de EV.

Enlace a la discusión en Enseñanza Virtual:

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

**Conflicto 3.** El atributo “*identifier*” no cambia al modificar el nombre del usuario

**Descripción del conflicto:**

Durante el análisis y definición del atributo “*identifier”* (el cual se basa originalmente en el nombre y apellido del usuario). Legué al punto de tener que contemplar que una vez creado, se actualiza el identificador si el usuario cambia su nombre o no.

El conflicto surge de la expectativa de que el identificador debería reflejar siempre el nombre actual del usuario, frente a la decisión técnica de mantenerlo inmutable después de la creación.

**Alternativas de solución evaluadas:**

1. **Actualizar el “*identifier*” automáticamente al cambiar el nombre o apellidos.**

* **Ventajas**: El identificador siempre se corresponde con los datos actuales del usuario, haciéndolo más intuitivo y comprensible para los administradores y ellos mismos.
* **Desventajas**: Requiere la modificación del Framework y la actualización de usuarios. Además, podría causar colisiones o inconsistencias, especialmente porque el “*identifier”* es usado como clave única y de referencia de un Customer en otras entidades.

1. **Mantener el “*identifier*” inmutable tras la creación del Customer.**

* **Ventajas**: Se garantiza que el identificador único permanece constante, lo cual es adecuado para registros persistentes y facilita la trazabilidad.
* **Desventajas**: El identificador puede no reflejar los nuevos datos personales del usuario si estos se actualizan, lo cual puede generar cierta confusión.

**Justificación de la solución adoptada:**

Se optó por la segunda alternativa debido a las limitaciones del framework para la modificación del identificador en el formulario de cambio de datos de usuario. Además, la inmutabilidad de un identificador garantiza consistencia y estabilidad en las referencias que se hagan sobre clientes.

Para garantizar que el identifier sigue siendo válido y único, se implementó una validación personalizada en la clase CustomerValidator, tanto a nivel de entidad (mediante anotaciones como @ValidCustomer) como a nivel de servicio, verificando que el identificador cumple el patrón y no colisiona con otros ya existentes (a menos que pertenezcan al mismo Customer).

**Conformidad de la solución adoptada**

La solución ha sido aprobada por el profesor a lo largo de las prácticas de laboratorio.

#### **Conclusión**

En conclusión, este informe ha permitido identificar y corregir requisitos ambiguos e inconsistentes dentro del proyecto, asegurando su correcta definición y minimizando riesgos futuros. Además, se ha documentado la gestión de conflictos, destacando la importancia de la toma de decisiones efectiva y oportuna para mantener el progreso del proyecto. Este enfoque proactivo y detallado garantiza una base sólida para el desarrollo exitoso del proyecto.

En definitiva, este documento proporciona un marco organizativo que no solo facilita el desarrollo del proyecto, sino que también promueve un ambiente de trabajo cooperativo y profesional, alineado con los estándares de calidad y eficiencia requeridos.

#### **Bibliografía**

Intencionalmente en blanco.