ACME AirNav Solutions



D03 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 3 Abril, 2025

**TABLA DE CONTENIDOS**

[**Resumen Ejecutivo** 3](#_Toc199183093)

[**Tabla de Revisiones** 3](#_Toc199183094)

[**Introducción** 4](#_Toc199183095)

[**Conflictos** 4](#_Toc199183096)

[**Conclusión** 10](#_Toc199183097)

[**Bibliografía** 10](#_Toc199183098)

#### **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 | 03/04/2025 | 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.** Validación de solapamiento de reservas de vuelos para un mismo pasajero

**Descripción del conflicto:**

Durante el desarrollo de las operaciones relacionadas con la reserva de vuelos por parte de los pasajeros, surgió la duda sobre si se debía **validar que un pasajero no tenga reservas en vuelos distintos que coincidan en el tiempo**. En otras palabras, ¿debería el sistema **impedir que un pasajero reserve dos vuelos distintos con horarios solapados**?

**Alternativas de solución evaluadas:**

1. **Validar solapamiento de horarios en las reservas del pasajero.**
   * *Ventajas:* Evita inconsistencias aparentes en las reservas. Puede prevenir errores por parte del usuario, como comprar dos vuelos que no podrá tomar.
   * *Desventajas:* Restringe la libertad del usuario. En la práctica, puede haber casos legítimos donde el pasajero desee reservar vuelos superpuestos (p. ej., por motivos logísticos, planificación flexible o reembolsos). Además, **aumenta la complejidad del sistema** al tener que comprobar solapamientos entre escalas (legs) y vuelos.
2. **No validar solapamiento de vuelos.**
   * *Ventajas:* Permite mayor flexibilidad al usuario. Refleja el comportamiento real de los sistemas de reservas, donde es el propio cliente quien decide si le interesa reservar vuelos que coinciden en el tiempo. Simplifica la implementación.
   * *Desventajas:* Puede generar situaciones poco coherentes desde el punto de vista lógico, si un pasajero reserva vuelos incompatibles.

**Solución adoptada:**

Se optó por **la segunda opción,** el principal motivo es que **el sistema no debe imponer restricciones artificiales a las decisiones del usuario**. En la vida real, las plataformas de aerolíneas permiten al cliente reservar vuelos con solapamiento, bajo su responsabilidad.

**Conformidad de la solución adoptada:**

La solución ha sido dada por válida por el profesor a lo largo de las prácticas de laboratorio siempre y cuando se posible justificarse (tal y como se está realizando ahora).

**Conflicto 2.** Almacenamiento del enlace al vídeo de prueba informal (Requisito #14)

**Descripción del conflicto:**

El requisito [D03-SG-33] especifica que se debe proporcionar un enlace a un vídeo en el que se pruebe informalmente el Requisito #14, pero no dice dónde debe incluirse en el repositorio.

Estas dudas se plasman en las siguientes dos preguntas:

* **Pregunta 1**: ¿Dónde exactamente dentro de las instalaciones de la USE debemos almacenar el vídeo?
* **Pregunta 2**: ¿Dónde exactamente debemos incluir el enlace en el repositorio?

**Alternativas de solución evaluadas:**

**Para la Pregunta 1 (ubicación del vídeo):**

1. **Utilizar la opción "adjuntar archivo" en los centros de entrega.**
   * *Ventajas:* Método directo y conocido.
   * *Desventajas:* No es posible obtener un enlace desde esta opción, lo que impide referenciar el vídeo en el repositorio.
2. **Utilizar el directorio personal en OneDrive (facilitado por la USE).**
   * *Ventajas:* Permite generar un enlace compartible. Es un entorno validado por la USE y cumple la normativa.
   * *Desventajas:* Dudas iniciales sobre accesibilidad del enlace por parte del profesorado y vigencia del enlace.

**Para la Pregunta 2 (ubicación del enlace):**

1. **Incluir el enlace en el archivo README.md.**
   * *Ventajas:* Fácilmente localizable en la raíz del repositorio.
   * *Desventajas:* Ninguna identificada.
2. **Crear un archivo específico (video-links.txt) para almacenar enlaces.**
   * *Ventajas:* Posibilidad de centralizar varios enlaces (vídeos, dashboards, etc.).
   * *Desventajas:* Dudas sobre la mejor ubicación del archivo (¿docs/group? ¿docs/studentX?).

**Justificación de la solución adoptada:**

Finalmente, se optó por la **alternativa 2 para pregunta 1** y la **alternativa 2 para pregunta 2**.  
El vídeo fue grabado usando “Recortes” de Windows y almacenado en OneDrive, siguiendo las recomendaciones del profesorado. El enlace se compartió mediante un archivo .txt ubicado en la carpeta personal de mi estudiante dentro de la carpeta “Student 2”.

**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?course_id=_89154_1&nav=discussion_board&nav=discussion_board&restrict_end_datetime=2025-5-26%2017%3A56%3A00&conf_id=_426211_1&conf_id=_426211_1&restrict_start_time=12%3A00%20AM&search_type=forum&forum_id=_253522_1&filterSearch=forum&restrict_start_date=05%2F26%2F2025&restrict_end_date=05%2F26%2F2025&pickname=&pickname=&pageIndex=&isSearch=Y&restrict_start_datetime=2025-5-26%200%3A0%3A00&action=collect_forward&restrict_end_time=05%3A56%20PM&origRequestId=2872600865_1748274989440&pickdate=&pickdate=&>

**Conflicto 3.** Gestión y comportamiento de campos “*readonly*”

**Descripción del conflicto:**

Durante el desarrollo de los requisitos R08 y R09, me surgieron dudas sobre **cómo deben tratarse los campos mostrados como “*readOnly”* en los formularios**. Pues cuando se marca un campo como “*readOnly*” en el formulario, **su valor no se actualiza**, pero tampoco se bloquea ni se ignora automáticamente durante el proceso de binding en el método **bind()**, lo cual hace que me pregunte como debo reaccionar a acciones ilegales.

**Alternativas de solución evaluadas:**

1. **Permitir el binding de atributos mostrados como readOnly.**
   * *Ventajas:* Simplicidad de implementación, no requiere lógica adicional.
   * *Desventajas:* Inseguro, ya que mediante un F12 o Postman sería posible realizar Post-hacking al valor pudiendo modificarlo
2. **Ignorar los atributos readOnly en el método bind().**
   * *Ventajas:* Protege contra cualquier modificación no autorizada o hackeo de valores mostrados como de solo lectura.
   * *Desventajas:* El atributo queda desprotegido en la capa de autorización (authorise), y no se impide su manipulación explícita si no se realiza una comprobación adicional.

**Solución adoptada:**

Se optó por la **alternativa 2**, es decir, **ignorar los atributos readOnly en el método bind()**, de forma que si un atacante intenta modificarlos, el sistema simplemente conserva su valor original. Esta decisión se tomó tras la revisión del profesorado, quien confirmó que esta solución era aceptable en el contexto actual del curso, dado que **no existe una forma sencilla de detectar de forma automática si un atributo readOnly ha cambiado de valor**.

**Conformidad de la solución adoptada:**

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

El profesorado recalcó que, si bien esta solución es válida para el curso, en un **entorno profesional** se debería implementar un **control más estricto en los métodos authorise()** para prevenir cualquier intento de modificación no autorizada. También insistieron en que los **métodos bind() no deben vincular atributos mostrados como de solo lectura**, y que es responsabilidad del equipo de desarrollo asegurar que:

* El precio de la reserva no pueda ser manipulado.
* El purchaseMoment no pueda ser alterado tras su asignación inicial.
* No se pueda asociar una reserva con vuelos ya pasados ni con pasajeros de otro cliente.

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=_468122_1&nav=discussion_board>

**Conflicto 4.** Etiquetado e identificación única de vuelos

**Descripción del conflicto:**

Durante el desarrollo de las funcionalidades relacionadas con reservas de vuelos, surgió un problema de **identificación clara y única de los vuelos** desde el usuario. En concreto, al listar opciones de vuelo o al realizar una selección, **no existía un identificador natural o etiqueta que permitiera distinguir claramente un vuelo concreto**, lo que generaba ambigüedad e inseguridad en el proceso de reserva.

Las entidades incluían identificadores internos (id) y atributos como tag, pero ninguno de ellos era adecuado para ser presentado o no eran únicos (tag), o eran poco significativos desde una interfaz (caso del id, que es un identificador técnico de base de datos).

**Alternativas de solución evaluadas:**

1. **Usar el ID del vuelo.**
   * *Ventajas:* Simple y directo.
   * *Desventajas:* No tiene sentido para el usuario, ya que es un número sin contexto. Además, el id pertenece en realidad a la entidad Leg, no a Flight.
2. **Usar el tag del vuelo.**
   * *Ventajas:* Accesible y fácil de mostrar.
   * *Desventajas:* Según [D02-S02-04], el tag representa características como "más barato" o "más rápido", pero **no identifica de forma única** un vuelo.
3. **Usar un formato personalizado String con atributos del vuelo.**
   * *Ventajas:* Da mucha información útil al usuario, al ser claro y suficiente para identificar un vuelo.
   * *Desventajas:* Requiere implementar una lógica adicional y personalizada para componer esa información.

**Solución adoptada:**

Se optó por implementar un método personalizado en la entidad Flight, que genera dinámicamente una **etiqueta descriptiva del vuelo** a partir de sus tramos (Leg). Dicha etiqueta incluye: hora de salida, ciudad y aeropuerto de origen; hora de llegada, ciudad y aeropuerto de destino. Esta solución sigue el enfoque sugerido por el profesorado, que propuso usar una etiqueta compuesta como “origen del primer tramo – destino del último tramo”. Siendo así una mezcla de la opción que puso el profesor del ejemplo de Iberia, más la recomendación del profesor de laboratorio Manuel de añadirle los aeropuertos para poder hacerlos únicos e intuitivos a primera vista.

**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=_465660_1&nav=discussion_board>

**Conflicto 5.** Edición de reservas con vuelos ya pasados

**Descripción del conflicto:**

Durante el desarrollo del sistema se detectó un problema relacionado con las reservas que permanecen en **modo borrador** (draftMode) durante un largo periodo de tiempo.

Concretamente, si un cliente crea una reserva asociada a un **vuelo futuro**, pero **no la publica** y el tiempo avanza hasta que el vuelo se convierte en **pasado**, entonces dicho vuelo **desaparece del selector** que permite modificar la reserva. Como resultado, al intentar acceder o editar esa reserva se genera un **error inesperado**, ya que el **vuelo ya no puede ser seleccionado**.

**Alternativas de solución evaluadas:**

1. **Dejar el vuelo como nulo si ya no es seleccionable.**
   * *Ventajas:* Permite seguir editando la reserva, aunque el vuelo original ya no esté disponible. El selector aparecería vacío y permitiría elegir otro vuelo futuro.
   * *Desventajas:* Se pierde información sobre la selección anterior.
2. **Lanzar un “Access Not Authorised” y bloquear la edición.**
   * *Ventajas:* Restringe la edición de reservas que ya no pueden completarse con coherencia (vuelo ya pasado).
   * *Desventajas:* Es excesivamente restrictivo, ya que **el usuario no está intentando hacer nada ilegal**, sino simplemente modificar una reserva aún en modo borrador. Además, **no permitiría corregir la situación**, bloqueando por completo la reserva.

**Solución adoptada:**

Se optó por la **primera alternativa**, **poniendo el vuelo a null** cuando ya no está disponible por ser pasado, permitiendo así continuar editando la reserva y elegir un nuevo vuelo futuro en el selector.

Esta solución **no considera ilegal** el hecho de tener un vuelo pasado en una reserva borrador, ya que **la reserva aún no ha sido publicada**. Por tanto, no justifica lanzar un **AccessNotAuthorised**.

**Conformidad de la solución adoptada:**

La decisión está alineada con el principio general de **permitir inconsistencias temporales en objetos en modo borrador**, siempre y cuando puedan ser corregidas antes de su publicación. Esta solución facilita una mejor experiencia de usuario, evitando así bloqueos innecesarios.

**Conflicto 6.** Ausencia de servicios de borrado (delete) para Booking, Passenger y BookingRecord

**Descripción del conflicto:**

Los requisitos funcionales del sistema no especifican en ningún momento la necesidad de que los clientes puedan **eliminar (delete)** reservas (Booking), pasajeros (Passenger) o las asociaciones entre ambos (BookingRecord). En consecuencia, se plantea el conflicto de **si es obligatorio implementar servicios delete para estas entidades** y, en caso negativo, porqué.

**Requisitos relevantes:**

Según los requisitos funcionales proporcionados:

* Los clientes pueden **listar**, **mostrar**, **crear** y **actualizar** reservas mientras estén en modo borrador (draftMode = true).
* Lo mismo ocurre con los pasajeros: pueden ser **listados**, **mostrados**, **creados** y **actualizados** si no han sido publicados.
* En ningún momento se menciona la opción de **eliminar** ninguna de estas entidades.

**Justificación de la decisión:**

1. **Ausencia de requisitos explícitos.**  
   Los requisitos funcionales del cliente **no exigen** la implementación de servicios de borrado. De acuerdo con la metodología, los desarrolladores no deben añadir funcionalidades no solicitadas a menos que exista una ambigüedad o laguna que deba resolverse para la coherencia del sistema. Y no parece ser el caso.
2. **Necesidad operativa cubierta con funcionalidades actuales.**  
   El flujo funcional es completamente operativo sin necesidad de eliminar reservas, pasajeros o sus relaciones. Un cliente puede modificar sus reservas y pasajeros mientras estén en modo borrador, lo cual cubre los principales escenarios de corrección de errores o cambios de última hora.
3. **Conformidad con la interpretación aceptada por el cliente.**  
   El propio profesor ha confirmado que:

“Cualquier interpretación *con sentido* se puede aceptar siempre que no cumpla con los criterios de suspenso.”

No implementar servicios de borrado puede ser considerada una **decisión de diseño coherente**, que no va contra los requisitos, ni introduce contrasentidos.

**Solución adoptada:**

Finalmente he optado por la no implementación de los servicios delete para Booking, Passenger y BookingRecord es una **decisión válida**, **justificada**, y **alineada con los requisitos** y las indicaciones del profesorado.

**Conformidad de la solución adoptada:**

La solución adoptada finalmente está nombrada en 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&nav=discussion_board&conf_id=_426211_1&forum_id=_253522_1&message_id=_470629_1>

**Conflicto 7.** Validación de solapamiento de reservas de vuelos para un mismo pasajero

**Descripción del conflicto:**

Durante el desarrollo de las operaciones relacionadas con la reserva de vuelos por parte de los pasajeros, surgió la duda sobre si se debía **validar que un pasajero no tenga reservas en vuelos distintos que coincidan en el tiempo**. En otras palabras, ¿debería el sistema **impedir que un pasajero reserve dos vuelos distintos con horarios solapados**?

**Alternativas de solución evaluadas:**

1. **Validar solapamiento de horarios en las reservas del pasajero.**
   * *Ventajas:* Evita inconsistencias aparentes en las reservas. Puede prevenir errores por parte del usuario, como comprar dos vuelos que no podrá tomar.
   * *Desventajas:* Restringe la libertad del usuario. En la práctica, puede haber casos legítimos donde el pasajero desee reservar vuelos superpuestos (p. ej., por motivos logísticos, planificación flexible o reembolsos). Además, **aumenta la complejidad del sistema** al tener que comprobar solapamientos entre escalas (legs) y vuelos.
2. **No validar solapamiento de vuelos.**
   * *Ventajas:* Permite mayor flexibilidad al usuario. Refleja el comportamiento real de los sistemas de reservas, donde es el propio cliente quien decide si le interesa reservar vuelos que coinciden en el tiempo. Simplifica la implementación.
   * *Desventajas:* Puede generar situaciones poco coherentes desde el punto de vista lógico, si un pasajero reserva vuelos incompatibles.

**Solución adoptada:**

Se optó por **la segunda opción,** el principal motivo es que **el sistema no debe imponer restricciones artificiales a las decisiones del usuario**. En la vida real, las plataformas de aerolíneas permiten al cliente reservar vuelos con solapamiento, bajo su responsabilidad.

**Conformidad de la solución adoptada:**

La solución ha sido dada por válida por el profesor a lo largo de las prácticas de laboratorio siempre y cuando se posible justificarse (tal y como se está realizando ahora).

#### **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.