Analysis Report

Grupo	C1.019
Repositorio	https://github.com/adolfoborrego/Acme-ANS-D02-25.1.0
Student #1	**ID**: 29584665H **UVUS**: XXB5458 **Nombre**: Borrego González, Adolfo Agustín **Roles**: Project Manager
Student #2	**ID**: 77873179D **UVUS**: SSK0456 **Nombre**: Martínez Díaz, Ignacio **Roles**: Analyst
Student #3	**ID**: 12830191D **UVUS**: PVL1690 **Nombre**: Mir Ceballos, Miguel **Roles**: Operator
Student #4	**ID**: 52077055H **UVUS**: TCP2748 **Nombre**: Sánchez Carmona, Germán **Roles**: Developer
Student #5	**ID**: 54794337B **UVUS**: CFV7375 **Nombre**: Regidor García, Miguel **Roles**: Tester

1. Introducción

1.1. Executive Summary

Este informe de análisis para el desarrollo del proyecto Acme ANS tiene como objetivo analizar los requisitos clave en busca de posibles problemas y documentar las decisiones tomadas para su solución.

1.2. Introducción

El presente **Reporte de Análisis** documenta la evaluación de los requisitos identificados para el desarrollo del proyecto **Acme AirNav Solutions (Acme ANS)**, desarrollado por el grupo **C1.019**. Este informe se centra en analizar aquellos requisitos que son de mayor complejidad y un buen análisis permitiría tener una idea más acertada de cómo llevarlos a cabo.

El objetivo de este análisis es identificar posibles inconsistencias, ambigüedades o mejoras necesarias en los requisitos, proporcionando conclusiones detalladas y decisiones tomadas para su corrección, incluyendo enlaces al foro en caso de que haya sido necesario a la hora de llevar a cabo estas decisiones.

La estructura de los contenidos en este documento se organiza de la siguiente manera: en primer lugar, se presentan los registros de análisis con los requisitos a evaluar, seguidos de las conclusiones derivadas del análisis y las decisiones tomadas para su solución. Finalmente, se presentan las conclusiones finales y la bibliografía.

2. Registros de Análisis

2.1 DELIVERABLE D01: GROUP | REQ-1

"Instantiate and customise the appropriate starter project so that you can work on this project. Make sure that the name of your project folder, maven configuration (pom.xml), and database is "Acme-ANS-D⟨dd⟩", where "⟨dd⟩" denotes the deliverable number using two digits. Make sure that you have followed the instructions in the "On Your Deliverables" document to package and deliver your work. This requirement must be fulfilled in this and every other group or individual deliverable for it to be considered satisfied."

Durante nuestro análisis, hemos identificado una ambigüedad significativa relativa a la instanciación y customización del proyecto, ya que en los requisitos se menciona "the appropriate starter project" sin especificar cuál es. Esto nos lleva a dos posibles alternativas:

- **Alternativa 1**: Usar como starter project el proyecto **Hello World**, que es más básico y puede servir fácilmente como base para cualquier otro desarrollo.
- **Alternativa 2:** Usar como starter project el proyecto **Acme Jobs**, que incluye más funcionalidades, pero no es tan genérico.

Tras el análisis, hemos concluido que la opción más adecuada es utilizar el proyecto Hello World, ya que partir de Acme Jobs implicaría realizar numerosos cambios para ajustarlo a nuestras necesidades. Además, ninguna de sus funcionalidades resulta útil para el desarrollo de ACME-ANS.

Para respaldar nuestro análisis, hemos consultado la siguiente entrada del foro.

2.2 DELIVERABLE D02: STUDENT #2 | REQ-3

*"**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)."*

Durante nuestro análisis, hemos identificado una posible ambigüedad en la definición del atributo *earned points*, ya que el requisito no especifica si estos puntos tienen una relación directa con un valor monetario o si su uso se limita a un sistema de recompensas independiente. Esta falta de precisión nos ha llevado a evaluar dos posibles enfoques para la tipificación del atributo:

- **Alternativa 1:** Definir *earned points* como un atributo de tipo Integer.
    - **Ventajas:**
        - Implementación directa y sencilla
        - No requiere manejo de datos específicos
        - Facilita el trabajo con los datos.
    - **Desventajas:**
        - Para aplicar descuentos a futuro requeriría la implementación de una lógica distinta para su canjeo.

- **Alternativa 2:** Definir *earned points* como un atributo de tipo Money, interpretando que puede representar un descuento aplicable en futuras compras.
    - **Ventajas:**
        - Si los puntos representan descuentos, se facilita su integración en cálculos financieros. No requiere manejo de datos específicos
        - Es más intuitivo para el usuario a la hora de canjearlo
    - **Desventajas:**
        - Requiere una mayor complejidad y existe la posibilidad de que esté alejado de las pretensiones del cliente.
- **Alternativa 3:** Crear una clase específica, por ejemplo, EarnedPoints, que encapsula la lógica de los puntos y su posible conversión a valores monetarios
    - **Ventajas:**
        - Permite encapsular la lógica de los puntos y su posible conversión en un solo lugar.
        - Es extensible y facilita modificaciones futuras si se define una equivalencia con valores monetarios.
        - Evita la confusión entre si los puntos deben tratarse como enteros o dinero, dejando la opción abierta según futuras decisiones del cliente.
    - **Desventajas:**
        - Si finalmente no está alineado con las necesidades del cliente podría considerarse desproporcionado y un malgasto del tiempo de trabajo.

Dado que el requisito no especifica una conversión explícita entre los *earned points* y un valor monetario, y para evitar suposiciones que puedan llevar a errores de implementación o alternativas que puedan ralentizar innecesariamente el ritmo de trabajo, se ha decidido modelar el atributo *earned points* como un Integer. Esta decisión permite una implementación más sencilla y alineada con la información proporcionada en el requisito.

Para respaldar nuestro análisis, hemos consultado la siguiente entrada del foro.

2.3 DELIVERABLE D02: STUDENT #1 |REQ-3, REQ-4

“Relación entre Airlane Manager y Flight”

Durante nuestro análisis, hemos identificado una ambigüedad significativa relativa a la relación entre las entidades Airline Manager y Flight, ya que en los requisitos se menciona que los Airline Managers son responsables de manejar los Flights, pero no se especifica si un vuelo puede ser manejado por un solo Airline Manager o por varios. Esto nos lleva a 2 posibles alternativas:

- **Alternativa 1:** Usar una relación **ManyToMany**, donde varios Flights pueden ser manejados por varios **Airlane Managers**:
    - **Ventajas**:
        - Permite una mejor gestión de la carga de trabajo entre varios Airlane Managers.
    - **Desventajas**:
        - El uso de esta relación provoca la creación de Entidades intermedias virtuales que complican mucho la gestión del código en caso de error.
        - A nivel práctico que varias personas pueden decidir sobre un mismo vuelo podría causar conflictos en las decisiones.
- **Alternativa 2:** Usar una relación **ManyToOne**, donde un Airline Manager maneja varios Flights, y cada Flight es gestionado por un único Airline Managers.
    - **Ventajas:**
        - Mejor mantenimiento de la relación
        - Permite evitar conflictos en la toma de decisiones ya que cada Manager gestiona el vuelo que sea.
    - **Desventajas:**
        - Permite que un Manager gestione varios vuelos a la vez y desconocemos si en la práctica esto es posible.

Tras analizar las diversas opciones y teniendo en cuenta las conclusiones que hemos podido sacar del foro, nos hemos decantado por la **Alternativa 2** ya que a la vez que cumple con su cometido para el modelado, permite evitar usar relaciones que podrían derivar en errores difíciles de localizar.

2.4 DELIVERABLE D02: STUDENT #5,#1,#2,#4 | REQ-6

Ejemplo en el Student #5: "Produce assorted sample data to test your application informally. The data must include two ***technician*** *accounts with credentials "****technician1/ technician1****" and "****technician2/ technician2****". Create an additional technician account with credentials "****technician3/ technician3"*** *that represents a technician with no data, but his or her profile.*

"

Durante nuestro análisis hemos hallado una ambigüedad que nos ha llevado a cierta confusión, esta reside en que el requisito indica que se debe crear un technician(en el caso de ejemplo, technician3) el cual no tenga datos. Esto plantea un problema, ya que podría interpretarse como la necesidad de dejar todos los campos en null, excepto su user account y su propio user. Sin embargo, esto resulta conflictivo teniendo en cuenta que gran parte de los campos en la entidad son Mandatory. Además, al llevarlo a cabo nos hemos dado cuenta de la que la anotación @Mandatory no obliga a que los campos no sean nulos sino que únicamente muestra un warning pero genera los datos igualmente en la BD esto nos ha llevado a considerar varias formas de gestionar este requisito:

- **Alternativa 1:** Generar el sample data con todos los campos nulos, tal como indica explícitamente el requisito
    - **Ventajas**:
        - Genera bien los datos y no nos obliga a solucionarlo complicando el proceso
    - **Desventajas**:
        - Estaríamos obviando el hecho de que un atributo mandatory no debería ser nulo y por tanto creando valores que no son reales en el uso de nuestra aplicación.
- **Alternativa 2:** Crear el sample data dándole valores a los campos que son mandatory y dejando nulos solo los optionals.
    - **Ventajas**:
        - Esto mantendría la obligatoriedad de la adición de datos a estos campos y no generaría ninguna incongruencia con los requisitos.
    - **Desventajas**:
        - Bajo esta interpretación del requisito, realmente no lo estamos cumpliendo correctamente.
        - Aunque nosotros le demos bien los datos ahora la vulnerabilidad sigue existiendo.
- **Alternativa 3:** Crear el sample data dándole valores a los campos anotados con @Mandatory y dejando nulos solo los anotados con @Optional y además añadir la anotación @NotNull en aquellos campos que sabemos que no deberían ser nulos.

- **Ventajas**:
  - No generarían ninguna incongruencia con los requisitos ya que estaríamos añadiendo los datos que son obligatorios
  - Al poner el @NotNull también estamos evitando que en un futuro se pueda producir este error estando la aplicación en producción y nos ahorramos posibles problemas.
- **Desventajas**:
  - Estrictamente no estamos cumpliendo correctamente con el requisito.

Tras analizar las opciones decidimos que lo mejor era llevar a cabo la **Alternativa 1**, ya que era la única que nos aseguraba explícitamente cumplir con el requisito.

Sin embargo, tras hablarlo hoy con el profesor(**Manuel Jesús Jiménez Navarro**), comprendimos mejor el comportamiento de la anotación @Mandatory. Esta no impide que los valores sean nulos, sino que solo muestra una advertencia sin bloquear el flujo de ejecución. De esta forma, permite que todas las clases sean compiladas y el sistema funcione sin problemas y permita probar la funcionalidad. La solución que finalmente optamos por tomar es la **Alternativa 2**, es decir, que usaremos @Mandatory en todos los atributos de la entidad que lo necesiten y @Optional en los que sean opcionales, incluyendo también las asociaciones, de esta forma podremos poblar la Base de datos sin requerir que exista ningún tipo de asociación previa.

3. Conclusión

El análisis ha permitido llegar a un entendimiento mucho mayor de los requerimientos del proyecto ACME ANS, obteniendo datos relevantes para la producción, extraídos tanto del análisis de los requisitos, como de la investigación en el foro para la resolución de nuestras propias dudas.

Este proceso nos ha permitido tomar decisiones fundamentales basadas en la simplicidad, claridad y minimización de errores a largo plazo. Además, la evaluación de alternativas nos ha proporcionado una visión más detallada de las posibles implicaciones de cada elección permitiendo tomar una decisión que, aparte de cumplir a nivel funcional, también nos garantice cierta estabilidad a lo largo del proyecto.

4. Historial de versiones

Número de Revisión	Fecha	Descripción
1.0	02/03/2025	Borrador inicial
1.1	03/03/2025	Añadidos primeros registros e introducción
1.2	10/03/2025	Añadidos nuevos registros y conclusión
1.3	11/03/2025	Añadido registro 2.4 y documento formateado

5. Bibliografía

Intentionally blank.