

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

D02-Student#4-Analysis Report



Grado en Ingeniería Informática – Ingeniería del Software


DISEÑO Y PRUEBAS II

Curso 2024 – 2025

Fecha	Versión
12/03/2025	V1r0


Grupo: C1.006	
Autores	
Pereira Campos, Macarena	macpercam@alum.us.es

Repositorio GITHUB: <https://github.com/rubpergar/Acme-ANS-D02.git>

	<p style="text-align: center;">DISEÑO Y PRUEBAS II D02-Student#4-Analysis Report</p>
---	--

Índice de contenido

1. Tabla de versiones	2
2. Resumen ejecutivo	2
3. Introducción	2
4. Contenido	2
5. Conclusiones	7
6. Bibliografía	7

	<p style="text-align: center;">DISEÑO Y PRUEBAS II D02-Student#4-Analysis Report</p>
---	--

1. Tabla de versiones

Fecha	Versión	Descripción
13/03/2025	V1r1	Creación y redacción del documento.

2. Resumen ejecutivo

En este informe se presentará el análisis realizado sobre los requisitos del Deliverable 2, en el cual se desarrollaron funcionalidades para la creación de entidades, roles, formularios y la población de datos. Durante el proceso, se encontraron diversos obstáculos debido a la falta de claridad en algunos requisitos o a una mala interpretación de estos. Sin embargo, la mayoría de estos inconvenientes fueron resueltos a través del foro de la asignatura, el cual consulté periódicamente para identificar requisitos implícitos que inicialmente no había notado.

A lo largo del informe, se documentarán los problemas encontrados, las soluciones implementadas y los requisitos implícitos identificados en el documento


3. Introducción

Se presentará un análisis detallado de cada requisito. En dicho análisis, se abordarán las dificultades encontradas, las alternativas consideradas para solucionarlas y la solución finalmente elegida.

Esta documentación es fundamental para garantizar la coherencia y calidad del trabajo, ya que no solo permite cumplir con los requisitos establecidos, sino que también sirve como referencia para futuras revisiones y optimizaciones.

4. Contenido

Un error común en todas las entidades fue la falta de algunas restricciones. Este problema ocurrió porque las implementé antes de llegar a la parte de la teoría que mencionaba la importancia de definir rangos de datos. En ese momento, no sabía que era necesario establecer siempre un mínimo y un máximo.

	<p style="text-align: center;">DISEÑO Y PRUEBAS II D02-Student#4-Analysis Report</p>
---	--

Una vez que me di cuenta de esto, simplemente agregué los límites a todos los atributos que lo requerían.

Empezaremos comentando los requisitos mandatory de el entregable.

- *The **assistance agents** are the people responsible for recording and managing post-flight incidents reported by passengers. The system must store the following data about them: an **employee code** (unique, pattern "[A-Z]{2-3}\d{6}\$", where the first two or three letters correspond to their initials), a list of **spoken languages** (no longer than 255 characters), the **airline** for which they work, the **moment** on which they began to work for that airline (in the past), and optionally, a **brief bio** (up to 255 characters), their **salary**, and a link to a **photo** that should be stored anywhere else.*

El principal problema que encontré con este requisito fue que no sabía que se trataba de un rol y no de una entidad normal. Tras hablar con mis compañeros y darnos cuenta de esto, revisamos el foro y las diapositivas para comprender qué eran los roles, dónde debían ubicarse en la estructura del proyecto y cómo debían implementarse. Finalmente, lo resolví añadiéndolo a *Realms* y extendiendo la entidad a *AbstractRole*.

El siguiente obstáculo surgió al descubrir, tras leer el foro, que era necesario un *custom validator* para garantizar que las letras de *employee code* coincidieran con las iniciales del agente. Sin embargo, la implementación de este validador no presentó mayores dificultades y resultó ser bastante sencilla.

Por último, tuve problemas con el atributo *languages*, que se definía como una lista, a pesar de que en la asignatura no se recomienda este tipo de estructura de datos. Evalué dos alternativas:

Dejarlo como un *String* con un máximo de 255 caracteres, sin validaciones adicionales.

Modificar el validador del *assistance agent* para restringir la entrada a una lista de lenguajes permitidos, evitando que el usuario ingresara valores arbitrarios.

Opté por la primera opción, ya que un sistema de validación estricta podría generar errores innecesarios al escribir los idiomas, por el uso de espacios, mayúsculas o la posible ausencia de un idioma en la lista predefinida

La relación con *airline* no dio ningún tipo de problema, solo fue necesario revisar las diapositivas de teoría que hablan de las relaciones para asegurarme que lo hacía de forma correcta.

- A **claim** is a formal request or complain made by a passenger or customer due to a problem or inconvenience experienced during a flight. They are registered by the **assistance agents**, and the data to store when registering a **claim** is the following: the **registration moment** (in the past), the **passenger email**, a **description** (up to 255 characters), a **type** ("FLIGHT ISSUES", "LUGGAGE ISSUES", "SECURITY INCIDENT", "OTHER ISSUES") and an **indicator** of whether the claim is accepted or not.

Esta entidad no presentó mayor complejidad. La única duda que surgió fue si era necesario establecer una relación entre *Claim* y *AssistanceAgent*.

Tras revisar el foro de la asignatura, encontré a otra persona con la misma duda, y la respuesta confirmaba que sí era necesaria dicha relación. Con esta información, revisé la teoría para implementarla correctamente, la añadí y problema resuelto.

- Claims need to be tracked through **tracking logs**. A tracking log records each step in the procedure followed to resolve or reject a claim, ensuring that all actions and decisions are documented. The system must store the following data about **tracking logs**: the **last update moment**, the **step** undergoing (up to 50 characters), a **resolution percentage**, and an **indicator** on whether the claim was finally accepted or not. When a claim is accepted or rejected, the system must store its **resolution** indicating the reason why was rejected or the compensation to offer (up to 255 characters).

El principal problema de esta entidad fue la gran cantidad de requisitos implícitos que no había tenido en cuenta cuando la implementé por primera vez.

Al principio, mis únicas dudas fueron qué tipo de datos y restricciones aplicar al *resolution percentage*. Las opciones que consideré fueron: un Double con ValidNumber que estuviera entre 0 y 1 y permitiera hasta dos dígitos fraccionarios, o un Double con ValidScore, que, tras revisar la documentación del framework, encontré que ofrecía restricciones entre 0 y 100. Opté por la segunda alternativa, ya que me parecía más adecuada y sencilla.

Otra duda fue si era necesario crear una relación entre esta entidad y Claim. Decidí que sí lo era, y consulté el foro para verificar mi decisión.



Al revisar el foro, me di cuenta de que había implementado gran parte de esta entidad incorrectamente, además de descubrir que era necesario un validador personalizado para varios atributos.

El primer error fue en la implementación del atributo *indicator*, que lo definí como un Boolean para indicar si la Claim había sido aceptada o no. No tuve en cuenta que también podría estar en un estado "pendiente". En el foro se recomendaba crear un enumerado con los valores: "PENDING", "ACCEPTED" y "REJECTED".

El siguiente error fue no considerar la restricción de que si la *claim* aún no estaba aceptada, no podía tener *resolution*, y que si estaba aceptada o rechazada, debía tener obligatoriamente el *resolution percentage* al 100%.


Además, encontré otra restricción implícita en el foro: el porcentaje de resolución debía ser creciente. Esto significaba que el tracking log anterior de una *claim* no podía tener un porcentaje mayor que el actual. Para implementar esta restricción, fue necesario crear un repositorio que accediera a todos los tracking logs de una *claim*, ordenados por la fecha de actualización, y luego comparar el porcentaje más reciente con el actual para asegurarse de que el porcentaje actual fuera mayor.

El último error fue la falta de datos de prueba para verificar el funcionamiento del validador. Para solucionarlo, decidí crear los datos de prueba necesarios y comprobar que todo funcionara correctamente.

- *Produce assorted sample data to test your application informally. The data must include two **assistance agent** accounts with credentials “**agent1/agent1**” and “**agent2/agent2**”. Create an additional agent account with credentials “**manager3/manager3**” that accounts for a new agent with no associated data, except for his or her profile.*

Esta tarea no presentó mayores problemas, excepto el de no entender bien el enunciado al principio. Sin embargo, tras comentarlo con mis compañeros y aclararlo, resultó ser bastante sencilla de realizar.

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

	<p style="text-align: center;">DISEÑO Y PRUEBAS II D02-Student#4-Analysis Report</p>
---	--

Tarea muy sencilla, ya realizada en la entrega anterior. No dio ningún problema.

Ahora comentaremos los requisitos “supplementary”:

- *The system must handle **assistance agent dashboards** with the following **indicators**:*
 - *The ratio of claims that have been resolved successfully.*
 - *The ratio of claims that have been rejected.*
 - *The top three months with the highest number of claims.*
 - *The average, minimum, maximum, and standard deviation of the number of logs their claims have.*
 - *The average, minimum, maximum, and standard deviation of the number of claims they assisted during the last month.*

El único obstáculo de esta tarea fue entender el enunciado, y que se trataba de un Form, tras ello no tuvo ninguna complejidad.

- *Produce a UML domain model regarding the information requirements in your project.*
Gracias a las diapositivas de teoría, esta tarea no presentó mayor complejidad. La única duda que me surgió fue si debía añadir los enumerados y los *Forms*, y decidí que sí era necesario incluirlos.
- *The system is required to store **flight status** or **delays** that assistance agents can consult to help them with some claims. A web service must be used to populate this entity with information about flight statuses/delays. 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.*

Entender este enunciado fue bastante complejo, pero gracias a la ayuda de mi compañera, pude entender cómo abordarlo. Una vez que me lo explicó, la tarea se volvió mucho más sencilla. Opté por usar la API *AVIATIONSTRACK* y entendí que lo único necesario para este requerimiento era la creación de un formulario.

- *Produce an analysis report.*
Ninguna complejidad en este requerimiento.



- *Produce a planning and progress report.*

Ninguna complejidad en este requerimiento.

5. Conclusiones

A lo largo del desarrollo del Deliverable 2, uno de los mayores desafíos fue la presencia de requisitos implícitos que no fueron claramente especificados al inicio. Esto provocó una serie de dificultades durante la implementación de algunas entidades y funcionalidades. Sin embargo, la revisión constante del foro de la asignatura y la colaboración con compañeros fueron fundamentales para identificar y resolver estos problemas, mejorando la comprensión de los requisitos.

Las diapositivas proporcionadas fueron esenciales para resolver muchos de los obstáculos encontrados. Al estudiar la teoría de las relaciones entre entidades, validaciones y restricciones, se logró corregir errores y aplicar las mejores prácticas en el desarrollo del sistema.

En general, aunque surgieron varios obstáculos debido a la falta de claridad en los requisitos y algunas interpretaciones iniciales incorrectas, la constante comunicación con los compañeros y la consulta de recursos como el foro y las diapositivas fueron clave para resolver los problemas y cumplir con los requisitos del **Deliverable 2** de manera eficiente.

6. Bibliografía

- [1]. Profesores del departamento de la asignatura, On your tutorials, enseñanza virtual us, 2025.