INFORME DE ANÁLISIS INDIVIDUAL PARA LA ENTREGA D3

Imagen que contiene Logotipo

El contenido generado por IA puede ser incorrecto.

Grupo: C1.061

Repositorio: <https://github.com/Davidvt04/Acme-ANS-C1.61>

Miembro:

David Valencia Toscano *davvaltos@alum.us.es*

Fecha: 26-05-2025

Índice

[Executive Summary 3](#_Toc190780304)

[Revision Table 5](#_Toc190780305)

[Introduction 6](#_Toc190780306)

[Contents 7](#_Toc190780307)

[Conclusions 10](#_Toc190780308)

[Bibliography 11](#_Toc190780309)

Executive Summary

En este informe se han analizado los los requisitos 8,9,29 y 30.

Requisito 8: “*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.*”.

Tras el análisis se ha llegado a la conclusión de que la mejor alternativa es definir un atributo derivado que identifique en formato String a un vuelo, tal como ha recomendado Rafael Corchuelo Gil.

Requisito 9: “*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.*”

Tras el análisis se ha llegado a la conclusión de que la mejor alternativa es definir el borrado de entidades de forma escalonada, solicitando al cliente que elimine las asociaciones entre la reserva y el pasajero antes de poder eliminar cualquiera de los dos.

Requisito 29: “*Operations by* ***customers*** *on* ***recommendations****:*

* *List recommendations related to final destinations in their bookings.”*

Requisito 30: *“Operations by* ***administrators*** *on* ***recommendations****:*

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

Tras el análisis se ha llegado a la conclusión de que la mejor alternativa es que el administrador sea quien realiza exclusivamente la llamada a la API, poblando la base de datos de recomendaciones que los clientes pueden consultar.

Revision Table

|  |  |  |
| --- | --- | --- |
| **Revision Number** | **Date** | **Description** |
| 1 | 26-05-2025 | Redacción del documento de análisis |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Introduction

En este informe se explican los distintos análisis que se han llevado a cabo sobre varios del listado de requisitos individuales del estudiante #2 (incluyendo obligatorios y suplementarios) para la entrega D03.

Los requisitos que aparecen en este informe son los que han generado alguna duda en cuanto a la correcta interpretación. Por ello, para cada uno se han planteado y valorado posibles soluciones a la problemática generada por la incertidumbre, mencionando ventajas y desventajas. Finalmente, se ha elegido una solución como válida, ya sea el motivo de la elección la validación del análisis por parte de uno de los profesores en el foro de la asignatura, el consejo dado por el profesor de laboratorio o, en caso de que no se haya consultado a un profesor por cualquier motivo, la conclusión a la que se ha llegado tras valorar las ventajas y desventajas de cada una de las alternativas.

Contents

Análisis del requisito 8

El requisito número 8 dicta lo siguiente:

*“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. ”.*

Para el requisito de funcionalidad obligatorio del Student #2 número 8 me ha surgido la duda respecto a cómo representar adecuadamente el vuelo a la hora de seleccionarlo en una reserva.

Tras investigar en el foro las posibles soluciones, encontré el mensaje del estudiante *Cristian Lacatus*, quien tenía la misma duda y planteaba alternativas tales como utilizar el identificador del vuelo, su tag o un atributo derivado a partir del mismo, alternativa finalmente escogida.

El análisis fue verificado y corregido por el profesor *Rafael Corchuelo Gil* en el foro de la asignatura: <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=_465699_1&nav=discussion_board>

Análisis del requisito 8 y 9

El requisito número 8 dicta lo siguiente:

*“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. ”.*

El requisito número 9 dicta lo siguiente:

*“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.”*

En los requisitos 8 y 9 no semenciona explícitamente la necesidad de poder eliminar reservas ni pasajeros, sin embargo estas funcionalidades se asumen implícitas en el flujo de trabajo de un usuario *customer* en la aplicación.

Como ambas entidades se relacionan a partir de una clase intermedia, se plantean para la acción de borrado las siguientes alternativas:

* Alternativa 1: No permitir la eliminación de reservas y pasajeros
  + Ventajas: Nos ceñimos fielmente a los requisitos indicados por el cliente.
  + Desventajas: Limitamos la funcionalidad de la aplicación, dificultando su utilización por el usuario final.
* Alternativa 2: Implementar una eliminación en cascada, de forma que al eliminar un pasajero se eliminan todas las relaciones con sus reservas.
  + Ventajas: Mayor velocidad a la hora de utilizar la aplicación, pues no será el cliente quien debe eliminar al pasajero de la reserva explícitamente.
  + Desventajas: El usuario puede ser propenso a eliminar al pasajero equivocado y con eso provocar un fallo mayor en cascada hacia todas sus reservas.
* Alternativa 3: Implementar una eliminación condicionada.
  + Ventajas: Para eliminar un pasajero, el cliente tiene que eliminarlo expresamente de las reservas donde lo incluya, obteniendo así la autorización expresa del mismo para realizar dicha acción.
  + Desventajas: Requiere mayor intervención del usuario, haciendo el flujo de trabajo más tedioso.

Tras valorar las distintas alternativas de solución, se ha elegido la alternativa 3, de acuerdo a los consejos proporcionados por José González Enríquez. De este modo, se evitan posibles fallos que la eliminación en cascada pudiera producir.

Análisis del requisito 29 y 30

El requisito número 29 dicta lo siguiente:

*“Operations by* ***customers*** *on* ***recommendations****:*

* *List recommendations related to final destinations in their bookings.”*

El requisito número 30 dicta lo siguiente:

“*Operations by* ***administrators*** *on* ***recommendations****:*

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

Ambos requisitos exprimen la funcionalidad de mostrar recomendaciones de lugares turístico en el destino de un usuario, es decir, el destino del vuelo asociado a la reserva del cliente. Las dudas sobre estos requisitos surgen a la hora de determinar de donde y cuando se obtienen los datos de recomendaciones.

Las posibles alternativas que se han planteado son las siguientes:

* Alternativa 1 Se obtienen las recomendaciones de una ciudad con cada llamada a la funcionalidad, es decir, una llamada a la API por acceso a la funcionalidad.
  + Ventajas: Se obtienen las recomendaciones más recientes en comparación a la alternativa 2.
  + Desventajas: Muchas llamadas a la API, pudiendo agotar la quota.
* Alternativa 2: Es el administrador el único que puede llamar a la API, almacenando los resultados en la base de datos de la aplicación.
  + Ventajas: Limitamos el número de datos solicitados, asegurándonos que no superamos la quota impuesta por la API.
  + Desventajas: Los datos de recomendaciones pueden quedarse desactualizados si el administrador no los actualiza regularmente.

Tras valorar las distintas alternativas de solución, se ha elegido la alternativa 2, pues es la que cumple de forma más fiel con los requisitos establecidos.

La selección de la solución se ha basado en el consejo dado por el profesor José González Enríquez durante las clases de Follow-Up.

Conclusions

Tras analizar los requisitos 8,9,29 y 30 se han llegado a las siguientes conclusiones:

* Para el requisito 8 se ha valorado que la mejor alternativa es crear un atributo derivado, tal como recomienda *Rafael Corchuelo Gil* en el foro.
* Para el requisito 8 y 9 se ha valorado que la mejor alternativa es implementar la eliminación de entidades de forma escalada, solicitando al cliente eliminar manualmente cada entidad.
* Para el requisito 29 y 30 se ha valorado que la mejor alternativa es que sea el administrador el único con capacidad para llamar a la API y cargar los datos de recomendaciones.

Bibliography

Intentionally blank