

**Student#2 Testing Report**

C1.046

<https://github.com/DP2-C1-046/Acme-ANS-D01>

|  |  |
| --- | --- |
| Apellidos, Nombre | Email |
| Caro Medina, Rafael David | rafcarmed@alum.us.es |

**Tabla de contenidos**

Contenido

[**Resumen ejecutivo** 3](#_heading=h.30j0zll)

[**Tabla de revisiones** 4](#_heading=h.1fob9te)

[**Introducción** 5](#_heading=h.3znysh7)

[**Contenidos** 6](#_heading=h.2et92p0)

[**Conclusiones** 7](#_heading=h.tyjcwt)

[**Bibliografía** 8](#_heading=h.3dy6vkm)

# **Resumen ejecutivo**

El presente documento de testing tiene como objetivo recopilar y documentar la creación de una Test Suite para la aplicación Acme-ANS con el fin de realizar las pruebas formales (Formal Testing) y de Rendimiento (Performance Testing) necesarias para garantizar una calidad razonable de la aplicación a entregar al cliente.

# **Tabla de revisiones**

|  |  |  |
| --- | --- | --- |
| **Versión** | **Descripción** | **Fecha** |
| v 1.0 | Versión original. | 23/05/2025 |

# 

# **Introducción**

A continuación se procede a detallar los Casos de Prueba y Test Suite generados en la aplicación para el Student#2 con el fin de poder ejecutar, y repetir, las pruebas formales y de rendimiento de ACME-ANS para la cuarta y última entrega D04 de la asignatura.

# **Contenidos**

1. **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).

Para este requisito en primer lugar intenté crear la clase Customer como una Entidad, al no haber leído previamente el requisito 6) que solicita posteriormente crear datos de ejemplo y cuentas de usuario para los Customers, así que luego la modifiqué para ser un Rol, extendiendo la clase de AbstarctRole en lugar de AbstractEntity.

Lo siguiente que voy a comentar es aplicable para todas las entidades y atributos que he implementado para D02:

Al crear el Rol **Customer,** y siguientes entidades, he seguido la metodología indicada en clase de Teoría de crear tres niveles de restricciones para cada atributo: Opcionalidad, Validación y Persistencia.

He detectado que, para el primer atributo, **Identifier**, el patrón presenta una errata pues la cardinalidad debe ser {2,3} y no {2-3}, con coma y no con guion. Debería haber creado un Custom Validator para validar que las dos o tres primeras letras del identificador corresponden con las iniciales del Customer, pero no me ha dado tiempo y lo intentaré para el siguiente Entregable D03. El resto de los atributos no me han presentado mayor problema, y sí que he consultado el foro en varias ocasiones y el libro de Excel que se subió para establecer los tipos y rangos especificados en dicho libro para cada atributo, incluido este rol Customer.

Consulta en foro sobre entidad y atributos de Customer:

<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=_461841_1&nav=discussion_board>

Consulta en foro sobre tipos y rangos de todas las entidades de la aplicación:

<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=_460989_1&nav=discussion_board>

1. A **booking** is a reservation made by a **customer** to purchase a **flight**, guaranteeing some seats on a specific itinerary and associating some **passengers**' details with the trip. The system must manage the following information for each **booking**: a **locator code** (unique, pattern "^[A-Z0-9]{6,8}$"), a **purchase moment** (in the past), a **travel** **class** ("ECONOMY", BUSINESS"), and a **price**. Optionally, the system should record the **last nibble** of the credit card used for payment.

Para implementar este requisito cree la entidad Booking con los atributos especificados, siguiendo la metodología de tres niveles de restricciones para cada atributo, y la clase enumerada TravelClass para el atributo de tipo enumerado TravelClass, el atributo Price de tipo Money con los rangos por defecto que establece la configuración de la aplicación, y sí que el atributo lastCardNibble lo interpreté al principio como el último dígito de la tarjeta de crédito, pero luego consultando y leyendo en el foto descubrí que más bien se trata del bloque de los cuatro últimos dígitos de la tarjeta.

Refenrencia a la pregunta en el foro :

<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=_462198_1&nav=discussion_board>

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

Y a continuación los atributos de navegación de la entidad, que la relacionan con el Customer/Cliente que ha realizado la compra, y con el Flight/Vuelo para el que se ha realizado la reserva.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

De consultar el foro, descubrí que la entidad Booking/Reserva tiene una relación muchos a muchos con la entidad Passenger/Pasajero, debido a que una Reserva puede involucrar a más de un Pasajero, y viceversa, un Pasajero puede a lo largo del tiempo tener más de una Reserva de vuelos, y por tanto se hacía necesaria una entidad intermedia que relacionara a ambas, con relaciones OneToMany, y que simulara esa relación Muchos a Muchos. Por tanto creé esa entidad BookingRecord que se recomienda en el foro para que relacionara de forma intermedia a Booking con varios Passenger, y a Passenger con varias Booking.

Texto, Tabla

El contenido generado por IA puede ser incorrecto.

La referencia a la pregunta en el foto es esta :

<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=_461394_1&nav=discussion_board>

**5**) A **passenger** is an individual who takes a flight and he or she must be registered in the corresponding booking. The system must store the following data about passengers: a **full name** (shorter than 256 characters), an **email**, a **passport number** (pattern “^[A-Z0-9]{6,9}$”), a **date of birth**, and, optionally, his or her **special needs** (shorter than 51 characters).

Y finalmente para implementar la entidad Passenger/Pasajero no he tenido mucho problema porque sus atributos no presentan mucho problema, son la mayoría de tipo String con las restricciones en 3 niveles y los rangos especificados en el libro de Excel del foro, el pasaporte con el patrón indicado, y la fecha de nacimiento de tipo Date con su validación de tipo @ValidMoment, y el atributo de Necesidades Especiales opcional y de tipo String.

Personalmente esta entidad no me ha supuesto ningún problema de interpretación, inexactitud o incongruencia, más allá de haber creado la entidad intermedia BookingRecord comentada en el foro para relacionar a los Pasajeros con Reservas a través de BookingRecords.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

# **Conclusiones**

Para esta segunda entrega D02 ha sido necesario leer y consultar varias veces el foro para solventar varias dudas sobre Requisitos de Información, entidades y roles, así como para establecer los tipos y rangos de atributos de esas entidades y roles, o crear la entidad intermedia BookingRecord para implementar la relación Muchos a Muchos entre Booking y Passenger, es decir, ha sido necesario consultar a los profesores sobre dudas producidas debido a inexactitudes, incongruencias o posibles interpretaciones derivadas de la lectura de los requisitos individuales para el Student#2.

# **Bibliografía**

Intencionalmente en blanco.