Un dibujo con letras

El contenido generado por IA puede ser incorrecto.

G

GGrupo: C1.036

Sevillano Barea, Alejandro - alesevbar@alum.us.es

Naredo Bernardos, Ignacio- ignnarber@alum.us.es

Robles Borrego, Adrián - adrrobbor@alum.us.es

S abido González, Francisco Manuel - [fransabgon@alum.us.es](mailto:fransabgon@alum.us.es)

Escudero Aldana, David - davescald@alum.us.es

TESTING REPORT

Repository: https://github.com/C1-036/Acme-ANS-D04

March 4th, 2025

Índice

1.Resumen ejecutivo 3

2. Introducción 4

3. Contenido

a. Pruebas funcionales 5-29

b. Análisis de desempeño 29-38

4. Conclusiones 39

5. Bibliografía 39

Tabla de versiones

|  |  |  |
| --- | --- | --- |
| Fecha | Versión | Descripción |
| 13/05/2025 | 1.0 | Hasta pruebas funcionales |
| 20/05/2025 | 1.1 | Documento terminado |

Tabla de revisiones

|  |  |  |
| --- | --- | --- |
| Fecha | Número de revisión | Descripción |
| 25/05/2025 | 1 | Revisado todo el documento |
| 26/05/2025 | 2 | Revisado de nuevo por ser el día de la entrega final |

Resumen ejecutivo

En el presente documento describimos de forma detallada el proceso de testeo al que ha sido sometido nuestro proyecto ACME-ANS. Con este proceso podemos garantizar el correcto cumplimiento y funcionamiento de todos los requisitos funcionales solicitados por nuestro cliente, así como que tengan una calidad aceptable.

Este proceso incluye distintos tipos de pruebas, los cuales incluyen la comprobación exhaustiva y concienzuda de todos los requisitos solicitados, que la aplicación es segura contra los distintos tipos de hacking, asegurándonos así de que se cumple con la ley orgánica 3/2018, y por ultimo la evaluación de rendimiento y tiempo de ejecución.

En resumen, con este documento se busca asegurarnos y dejar recogido en un solo documento que nuestra aplicación cumple con los requisitos funcionales exigidos por el cliente y que tiene unos baremos de rendimiento y ejecución correctos.

Introducción

Un informe de testing se trata de un documento donde se realiza una explicación de cómo se han creado las pruebas de nuestra aplicación, ACME-ANS, en este documento se recogen los análisis propios de los requisitos 8 y 9 del student 2, así como de la entidad intermedia que ha sido necesaria para la implementación de estos dos documentos.

Para la realización de este documento se ha seguido la estructura marcada por el documento anexo que hemos tenido disponible durante todo el curso.

Pese a esta marcada estructura las partes de mayor importancia son los referidos al contenido, los cuales son, el primero en el que se especifica las pruebas realizadas, el porcentaje de código que ha sido cubierto con ellas y la justificación de aquella parte de código que no ha sido cubiertas por las pruebas.

La segunda parte de especial importancia contiene las gráficas y análisis comparando el tiempo de ejecución y rendimiento de la aplicación en equipos distintos y antes y tras la adicción de índices en las entidades, dicha adición se realizó con la intención de mejorar el rendimiento de nuestra aplicación.

Contenido

Pruebas funcionales

Se han realizado pruebas para todos los servicios ofrecidos para las dos entidades (y la intermedia) relativos a el student 2, estas entidades son “Booking” y “Passenger”, la entidad intermedia se trata de “make”.

Estas pruebas están divididas en aquellas que acaban en “.safe” para probar los comportamientos aceptados por la aplicación y en “.hack” para comprobar la respuesta de la aplicación a los intentos de hackeo.

Quedando así el soporte total:  
  
Tabla

El contenido generado por IA puede ser incorrecto.

A continuación, enumeraremos todas las pruebas realizadas y hablaremos de que partes has sido ejecutado totalmente y que partes no, explicando el por qué de aquellas partes que no han sido ejecutadas completamente.

Booking

List

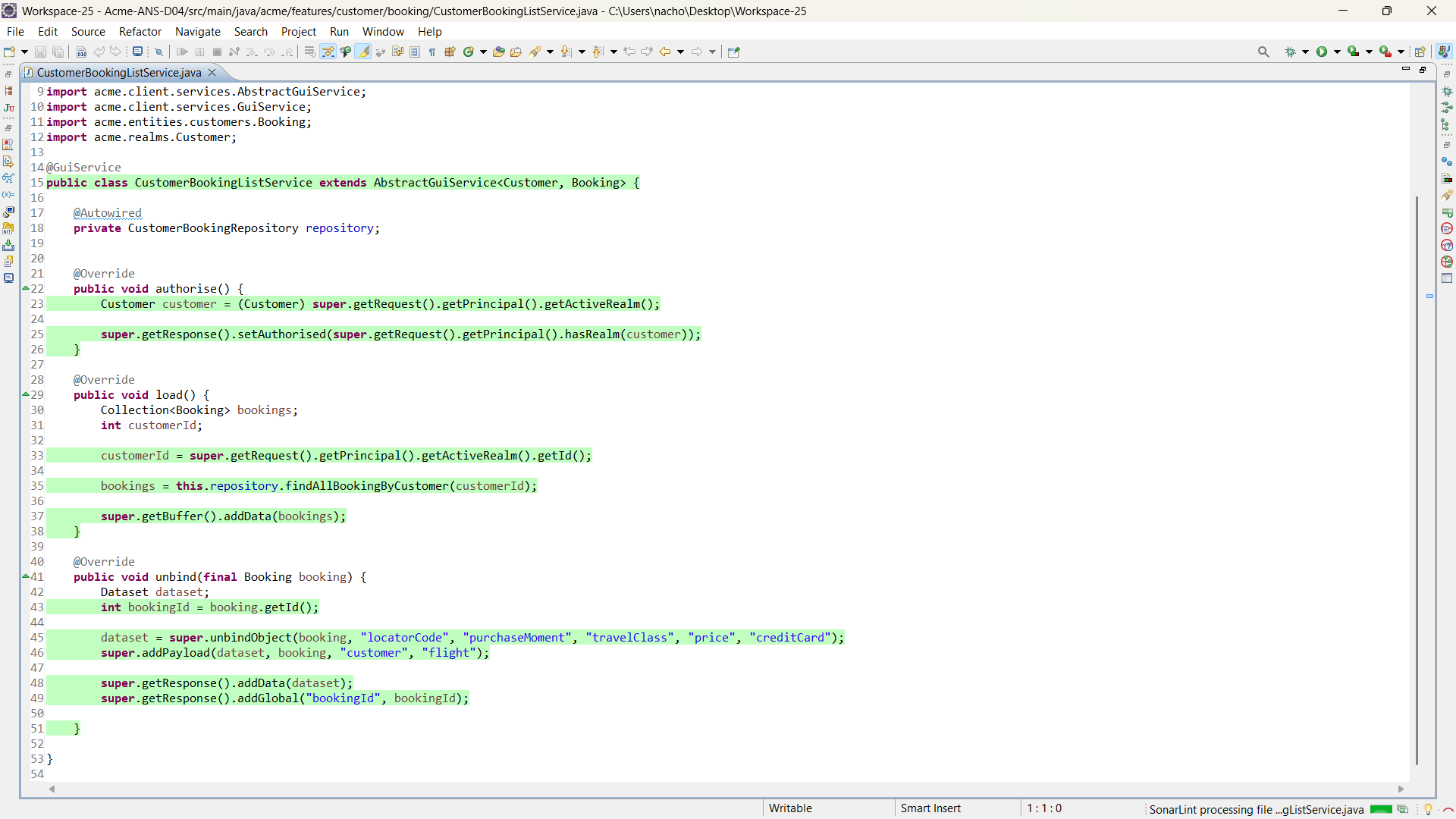
List.safe

-Listado de todos los booking que pertenecen a un customer concreto.

List.hack

-Intento de acceder a un listado de bookings desde un rol distinto al de customer, que es el único que puede acceder.

-Intento de acceder a un listado de bookings sin estar loggeado en ningún rol.



Como podemos observar se ejecutan todas las líneas de código al completo.

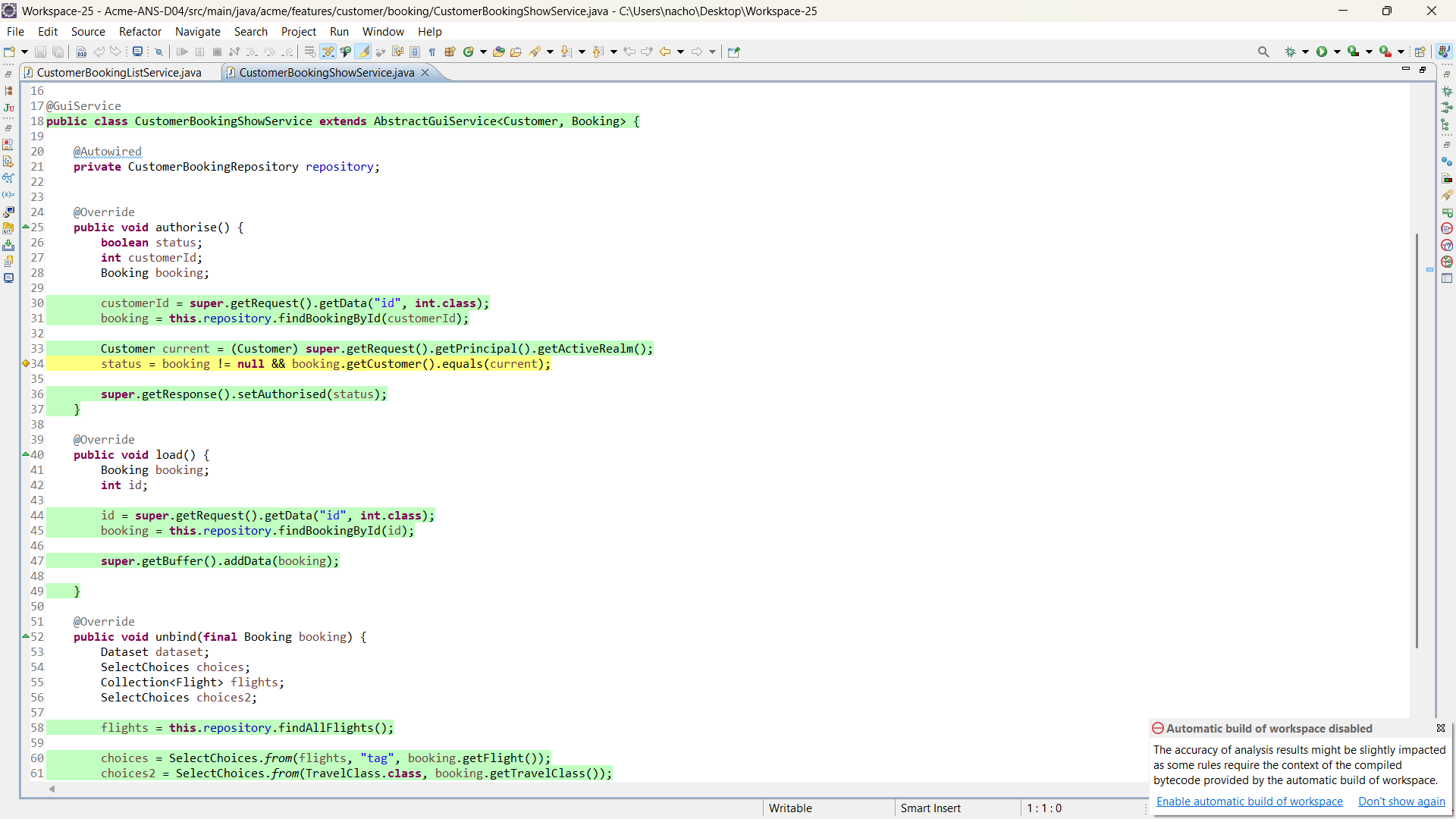
Show

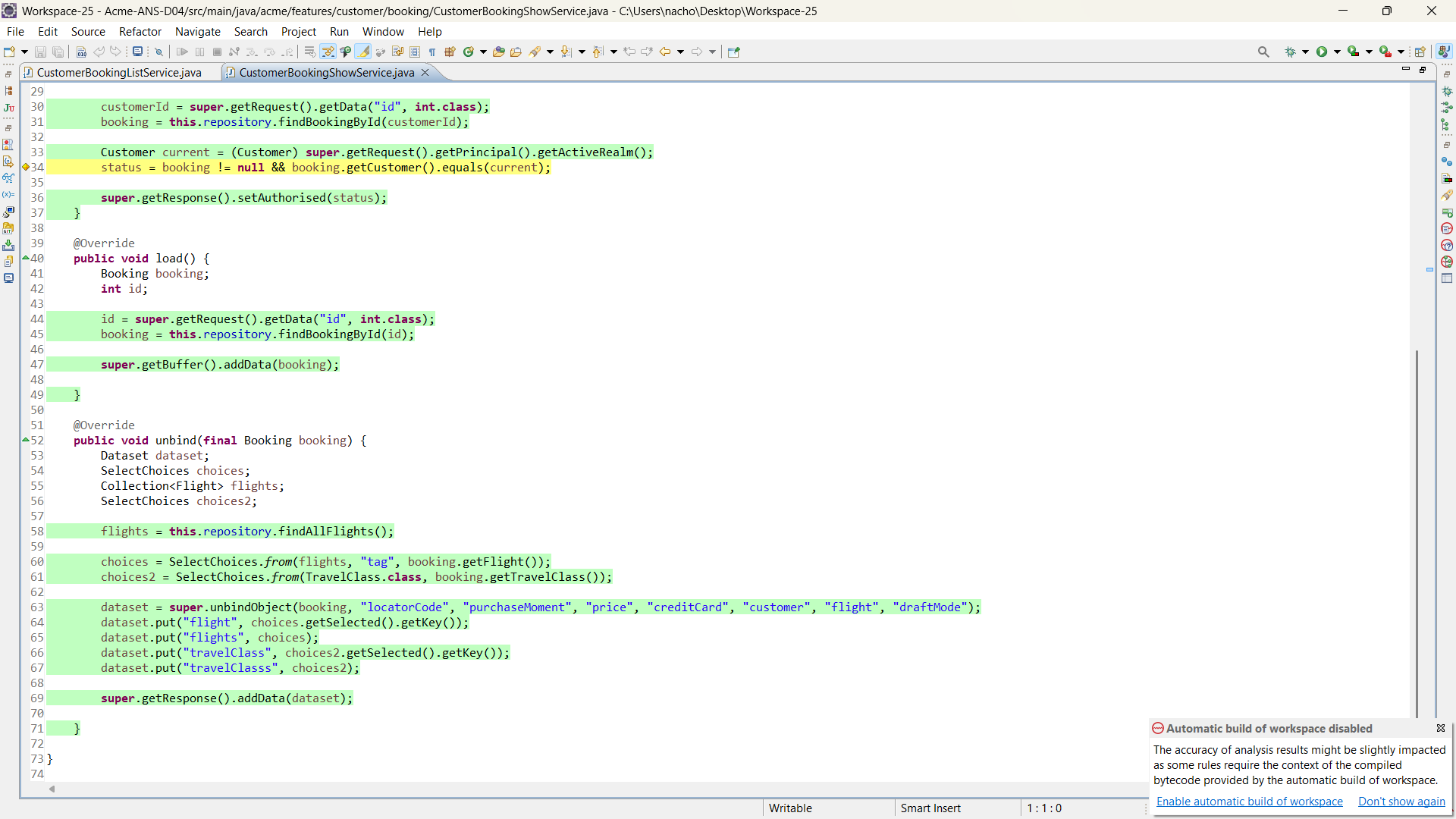
Show.safe

* Enseña los datos de un booking que pertenece al usuario.

Show.hack

* Intentar acceder a los datos de un booking sin estar loggeado.
* Intentar acceder a los datos de un booking estando loggeado en otro rol.
* Intentar acceder a los datos de un booking estando loggeado como un customer al que no pertenece ese booking.





Se ejecutan todas las líneas al completo, menos aquella relativa a las condiciones del authorise. No es refactorizable ya que no podemos eliminar ninguna para asegurar el correcto cumplimiento, no se puede combinar al completo porque si no se cumple la primera condición no se puede probar la segunda.

Create

Create.safe

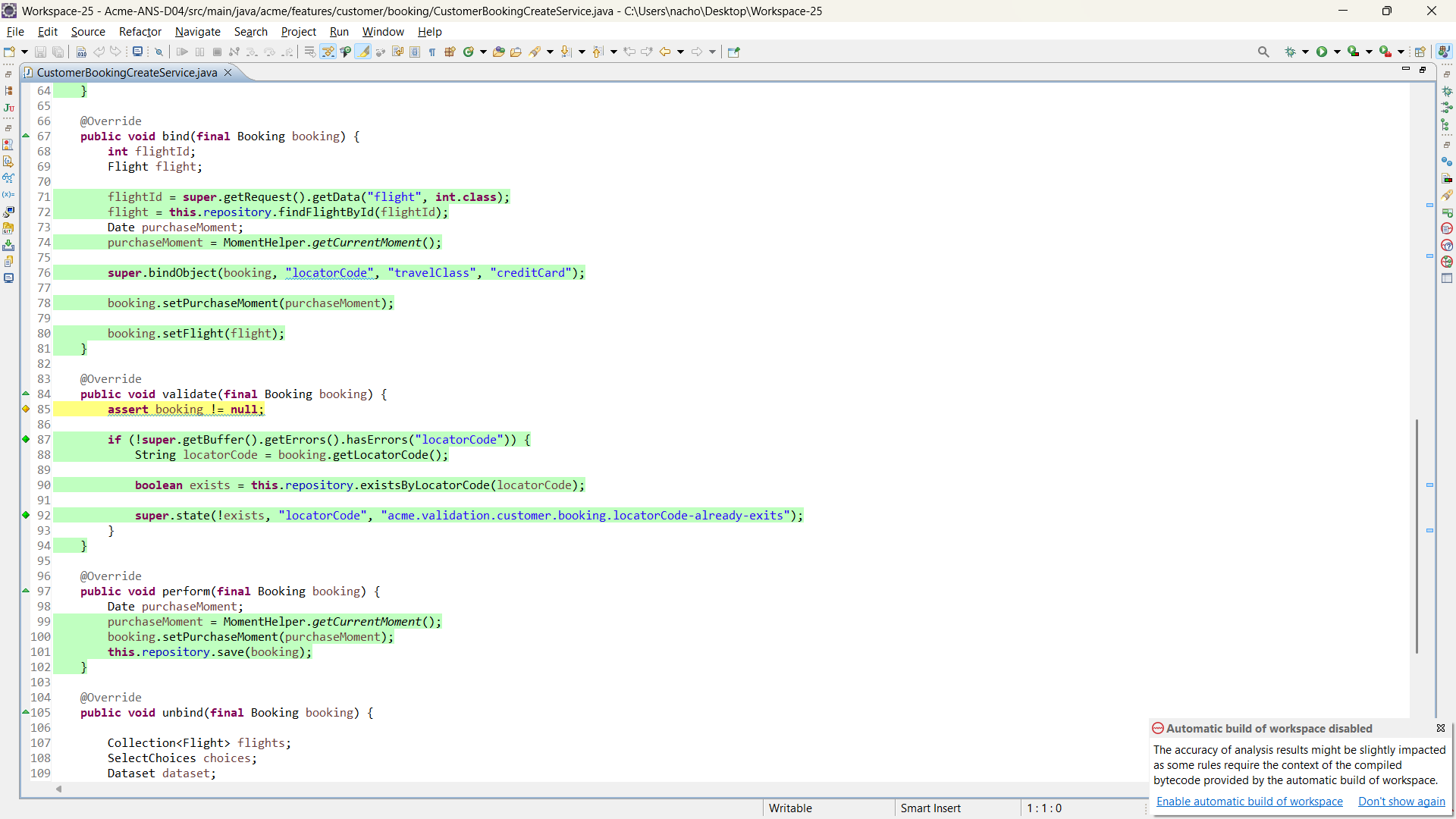
* Crear un booking con los datos vacíos.
* Crear un booking con toda la variedad de datos erróneos de los atributos.
* Crear un booking con datos límites tanto validos cómo inválidos para los atributos.
* Crear un booking con todos los datos correctos de los atributos.

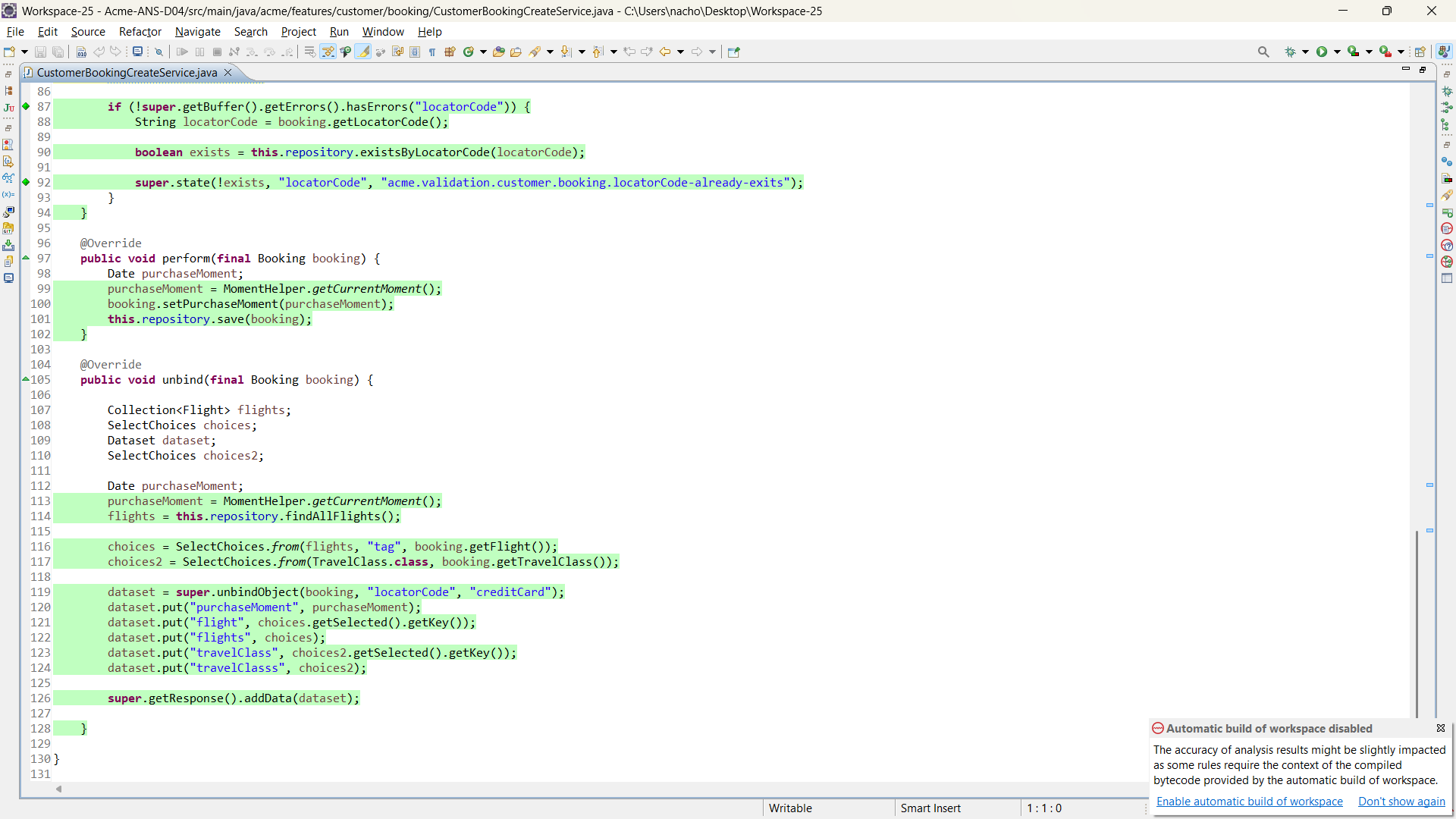
Create.hack

* Intentar crear un booking sin estar loggeado.
* Intentar crear un booking estando loggeado con otro rol.
* Intentar crear un booking dando valores inválidos para el elemento choice “flight”.
* Intentar crear un booking modificando los valores no editables por pantalla mediante el f12, aunque esta posibilidad no devuelve un 500 si no que simplemente es ignorado los valores introducidos por el unbind.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.





Todas las líneas de código han sido ejecutadas a excepción de dos, las cuales explicaremos el motivo a continuación:

-Assert object != null. Es una línea obligatoria que verifica que el objeto no sea nulo.

- La línea:

status **super**.getRequest().getPrincipal().hasRealm(customer) && isFlightAccessible;

Es una línea de condición que no se puede cumplir al completo ya que la condición 2 solo se puede verificar si vemos que el usuario efectivamente tiene el rol customer.

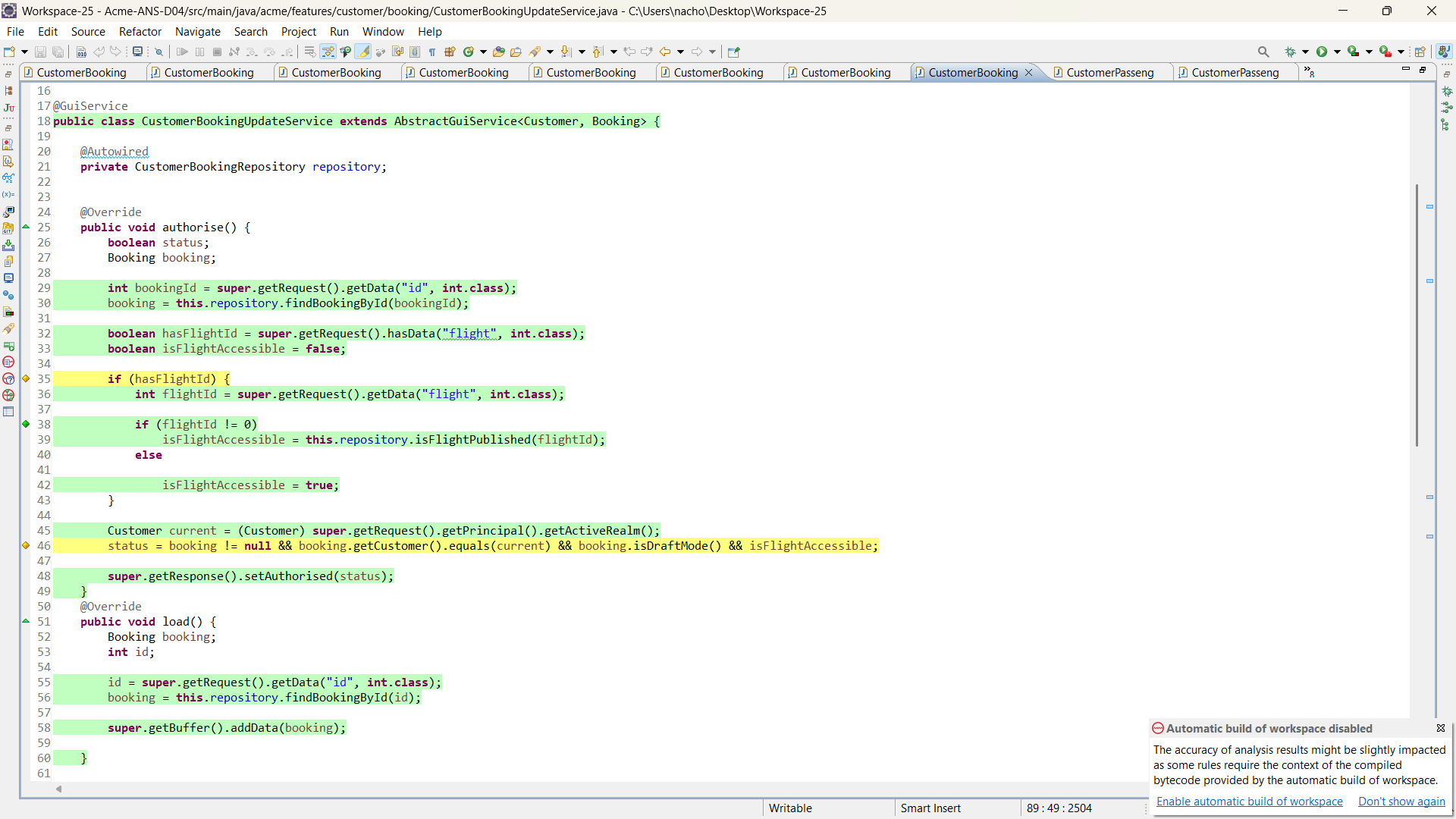
Update

Update.safe

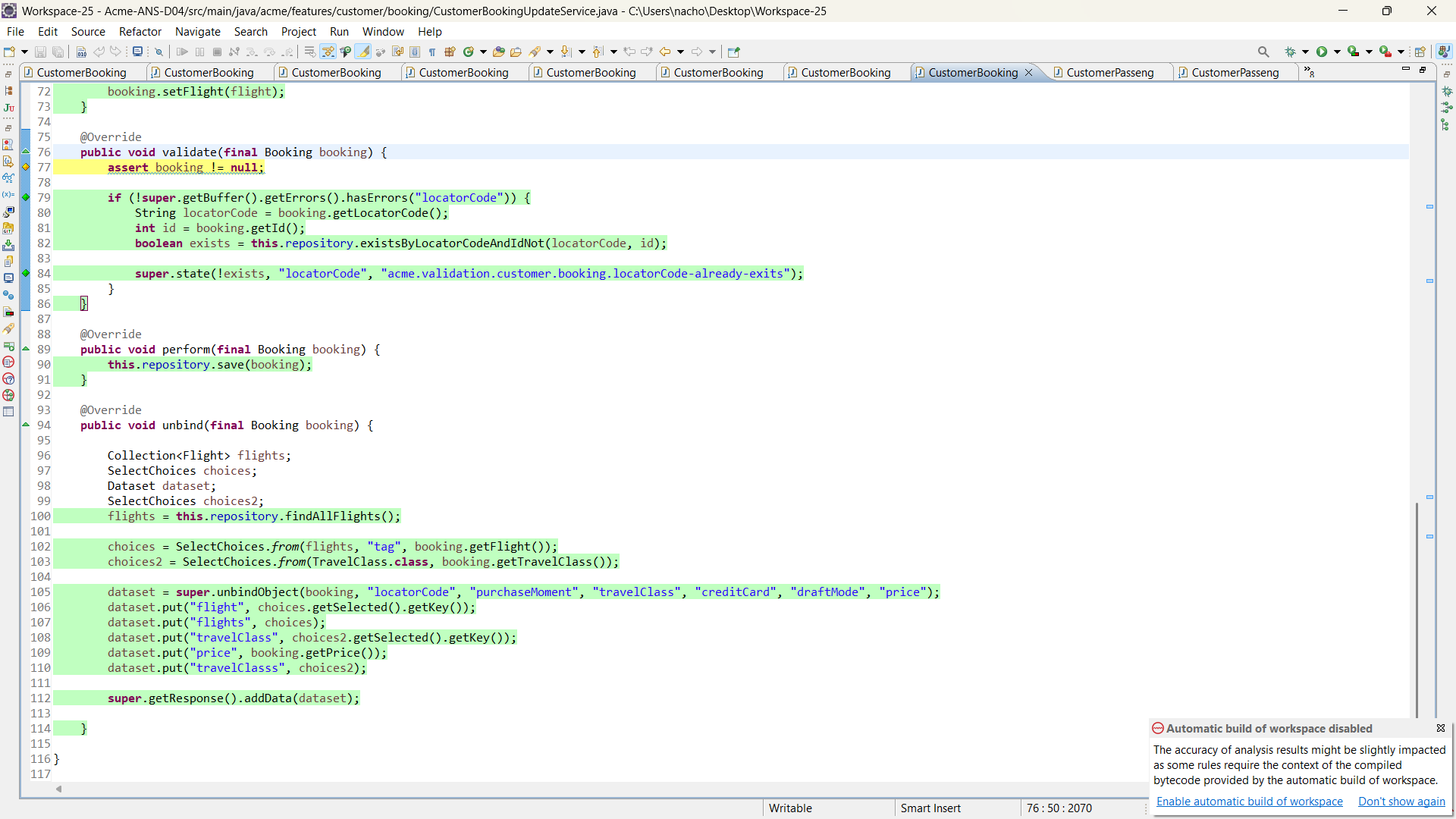
* Actualizar un booking con los datos vacíos.
* Actualizar un booking con toda la variedad de datos erróneos de los atributos.
* Actualizar un booking con datos límites tanto validos cómo inválidos para los atributos.
* Actualizar un booking con todos los datos correctos de los atributos.

Update.hack

* Intentar actualizar un booking sin estar loggeado.
* Intentar actualizar un booking estando loggeado con otro rol.
* Intentar actualizar un booking dando valores inválidos para el elemento choice “flight”.
* Intentar actualizar un booking modificando los valores no editables por pantalla mediante el f12, aunque esta posibilidad no devuelve un 500 sino que simplemente es ignorado los valores introducidos por el unbind.







-Assert object != null. Es una línea obligatoria que verifica que el objeto no sea nulo.

- La línea if(hasFlightId) no se puede cumplir siempre ya que solo esta puesto para que en el caso de que el usuario al intentar actualizar el booking no haya seleccionado ninguno no se produzca un error 500 al calcular el resto de las condiciones del vuelo.

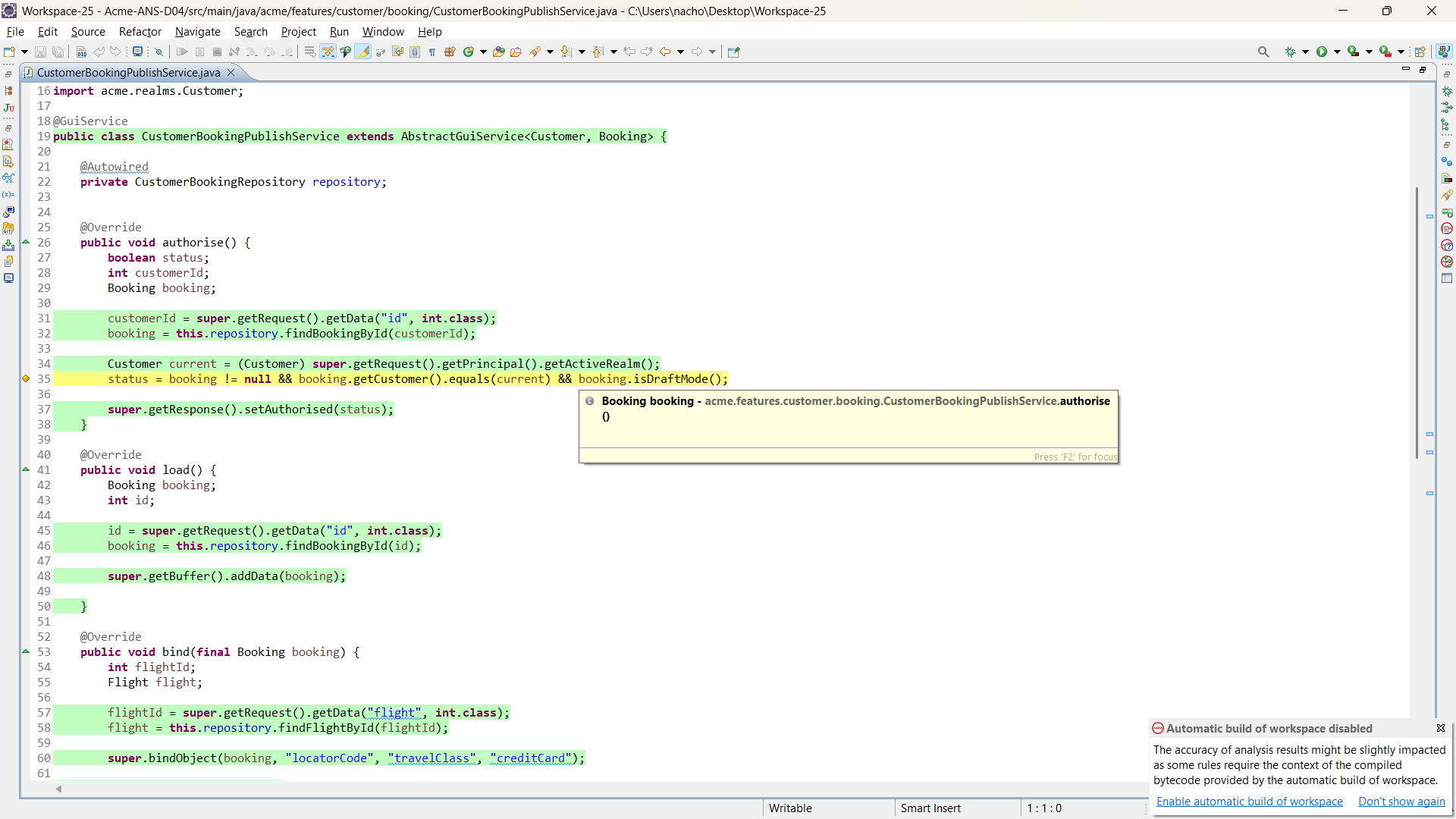
Publish

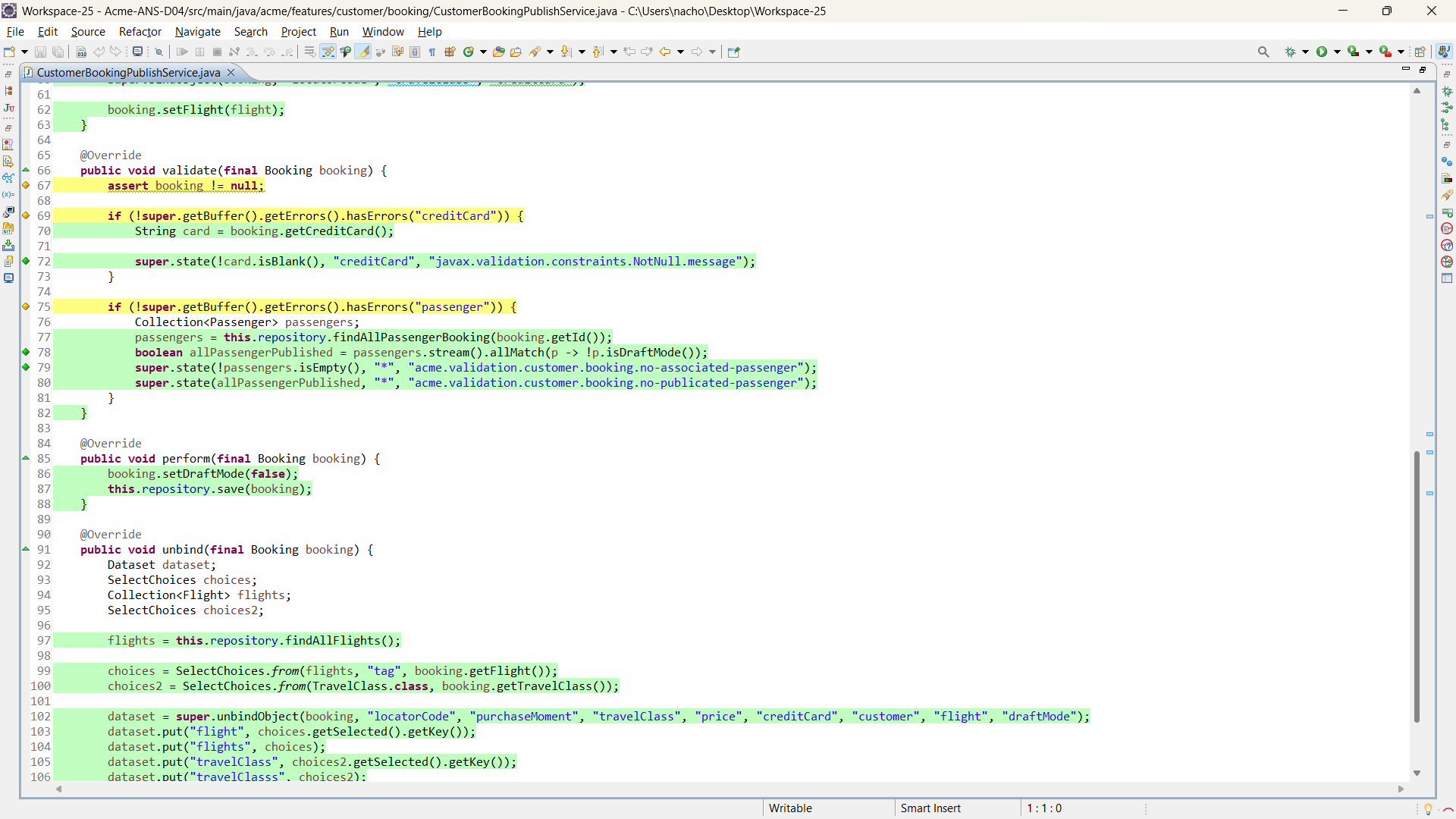
Publish.safe

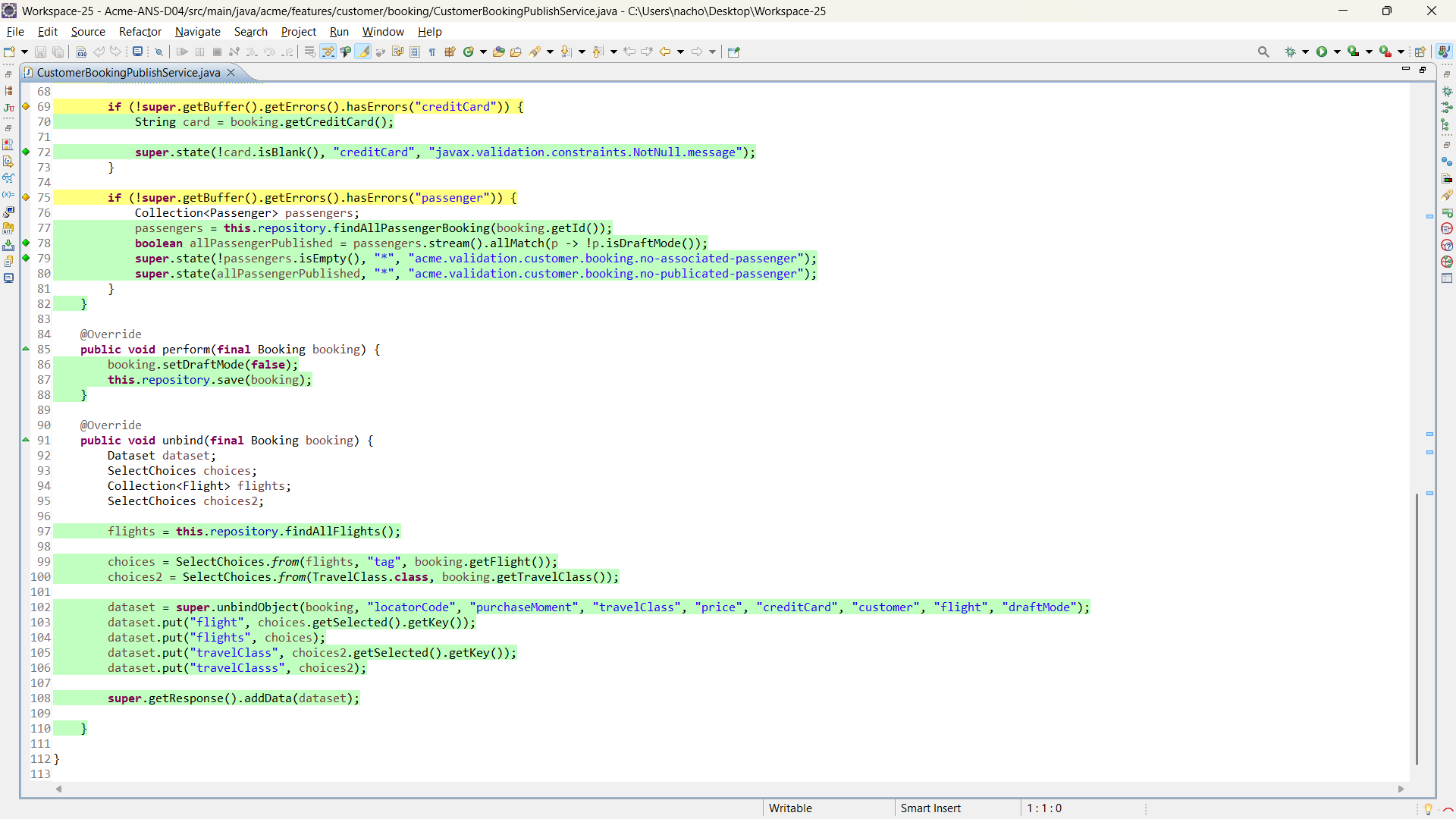
* Publicar un booking que cumpla todas las condiciones necesarias.
* Publicar un booking que no cumpla la condición necesaria de tener asignada tarjeta de crédito.
* Publicar un booking que no cumpla la condición necesaria de tener todos los pasajeros publicados.
* Publicar un booking que no cumpla la condición necesaria de tener al menos un pasajero asociado.
* Probar todos los atributos con valores correctos e incorrectos ya que se comporta como un update.

Publish.hack

* Intentar publicar un booking sin estar loggeado.
* Intentar publicar un booking estando loggeado como otro rol.
* Intentar publicar un booking desde un customer al que no le pertenece ese booking.
* Intentar modificar mediante el f12 los valores de un booking ya publicado.







- Assert object != null. Es una línea obligatoria que verifica que el objeto no sea nulo.

- **if** (!**super**.getBuffer().getErrors().hasErrors("creditCard")) {

Línea usada para validar que tiene el booking una tarjeta de crédito asociada antes de su publicación y que solo saltara cuando no la tenga asociada.

* **if** (!**super**.getBuffer().getErrors().hasErrors("passenger")) {

Línea usada para validar que tiene el booking al menos un passanger asociado antes de su publicación y que este está publicado, esta línea solo saltara cuando no la tenga asociada o no este publicado.

Delete

Delete.safe

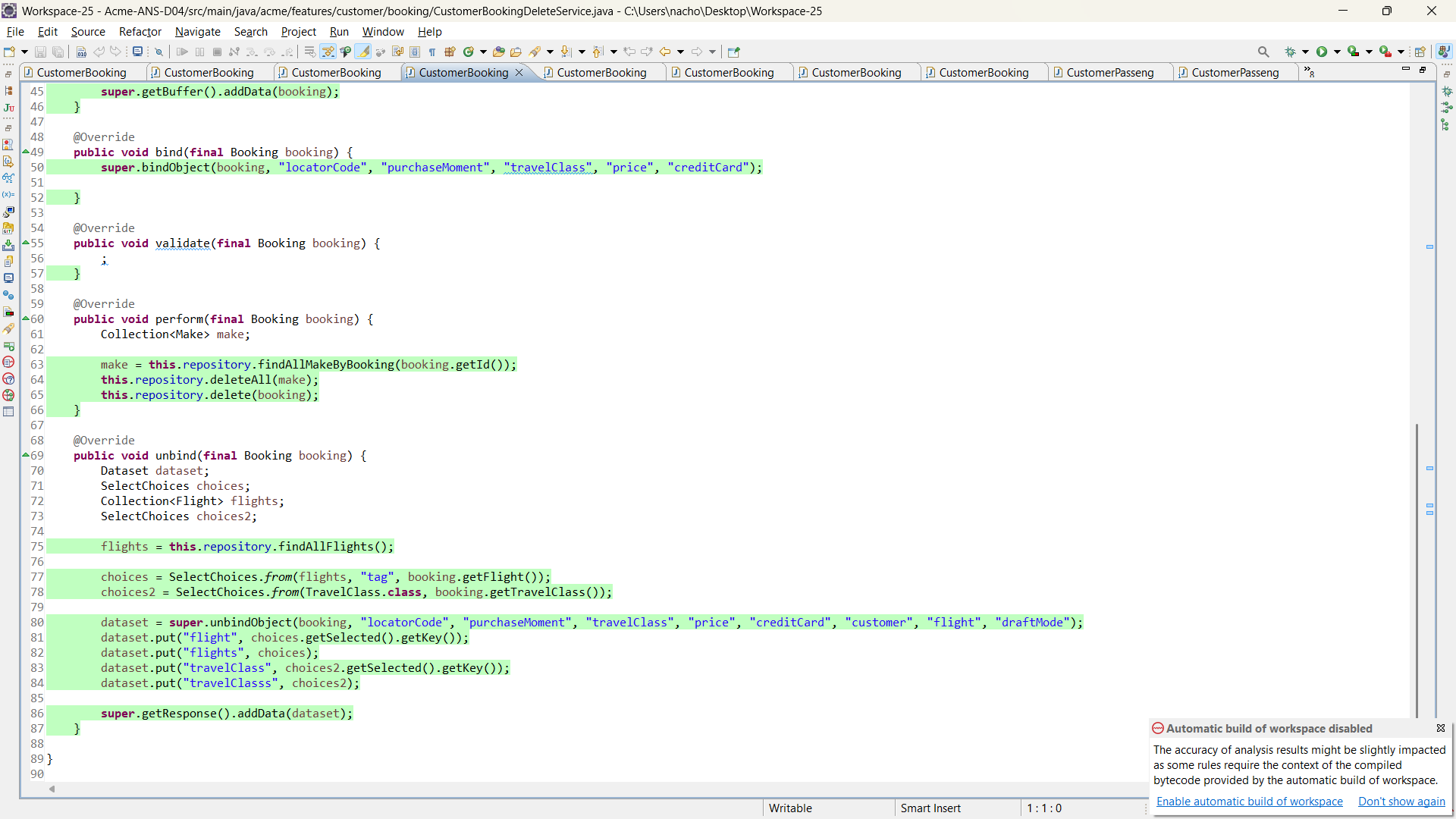
* Eliminar un booking que pertenece al usuario.

Delete.hack

* Intentar eliminar un booking sin estar loggeado.
* Intentar eliminar un booking estando loggeado con otro rol.
* Intentar eliminar un booking estando loggeado como un customer al que no le pertenece.
* Intentar eliminar un booking publicado.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.



* La línea :

status = booking != **null** && booking.getCustomer().equals(current) && booking.isDraftMode();

Es una línea de condición del authorise que no puede cumplirse completa porque en caso de que el booking sea null no se ejecutan las siguientes. La segunda condición es que pertenezca al customer actual, y la tercera que no esté publicado.

Controlador

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Se ejecuta al completo el código ya que se trata de todas las urls que se han usado en todos los servicios analizados anteriormente.

Passenger

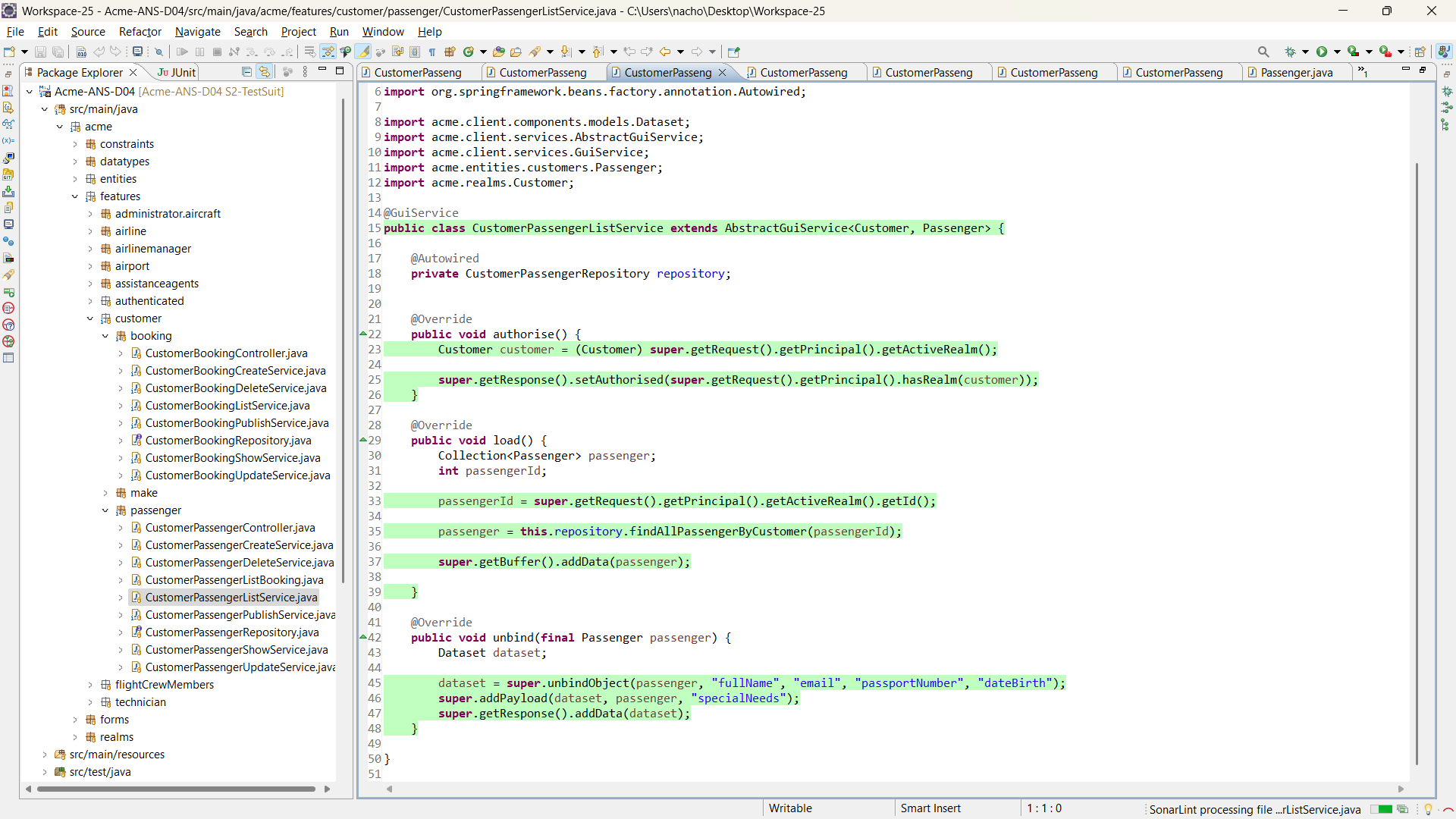
List

List.safe

* Listado de todos los Passenger que pertenecen a un customer concreto.

List.hack

* Intento de acceder a un listado de Passenger sin estar loggeado.
* Intento de acceder a un listado de Passenger loggeado con un rol distinto a customer.



Se ejecutan todas las líneas de código.

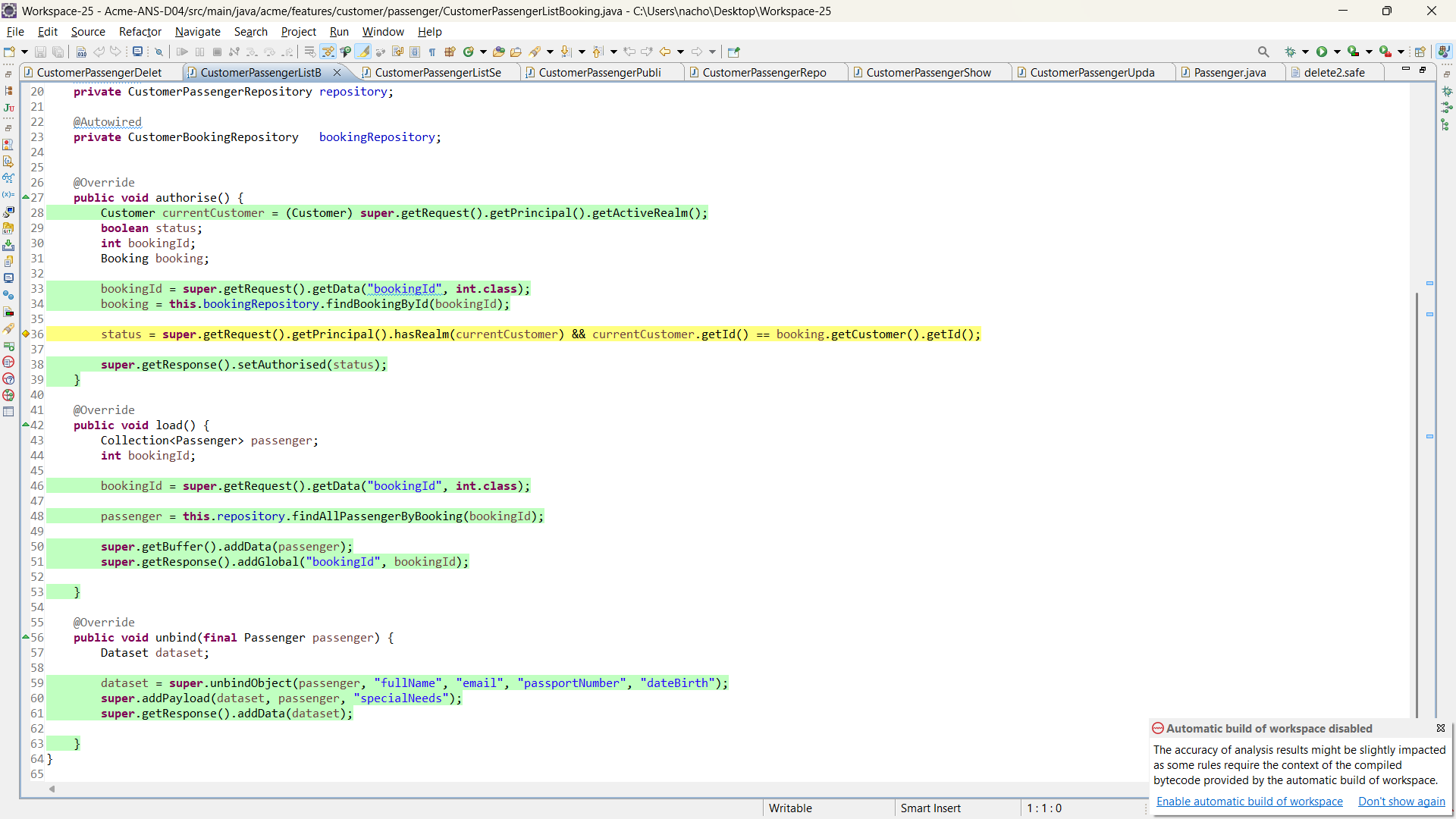
List-Passenger-Booking

List.safe

* Mostrar lista de los pasajeros pertenecientes a un booking.

List.hack

* Intentar mostrar lista de pasajeros pertenecientes a un booking sin estar loggeado.
* Intentar mostrar lista de pasajeros pertenecientes a un booking estando loggeado como un customer al que no le pertenece.
* Intentar mostrar una lista de pasajeros pertenecientes a un booking loggeado con otro rol.



Se ejecutan todas las líneas menos la 36 en la que no se cumple una de las condiciones debido a que una condición y la otra son incompatibles.

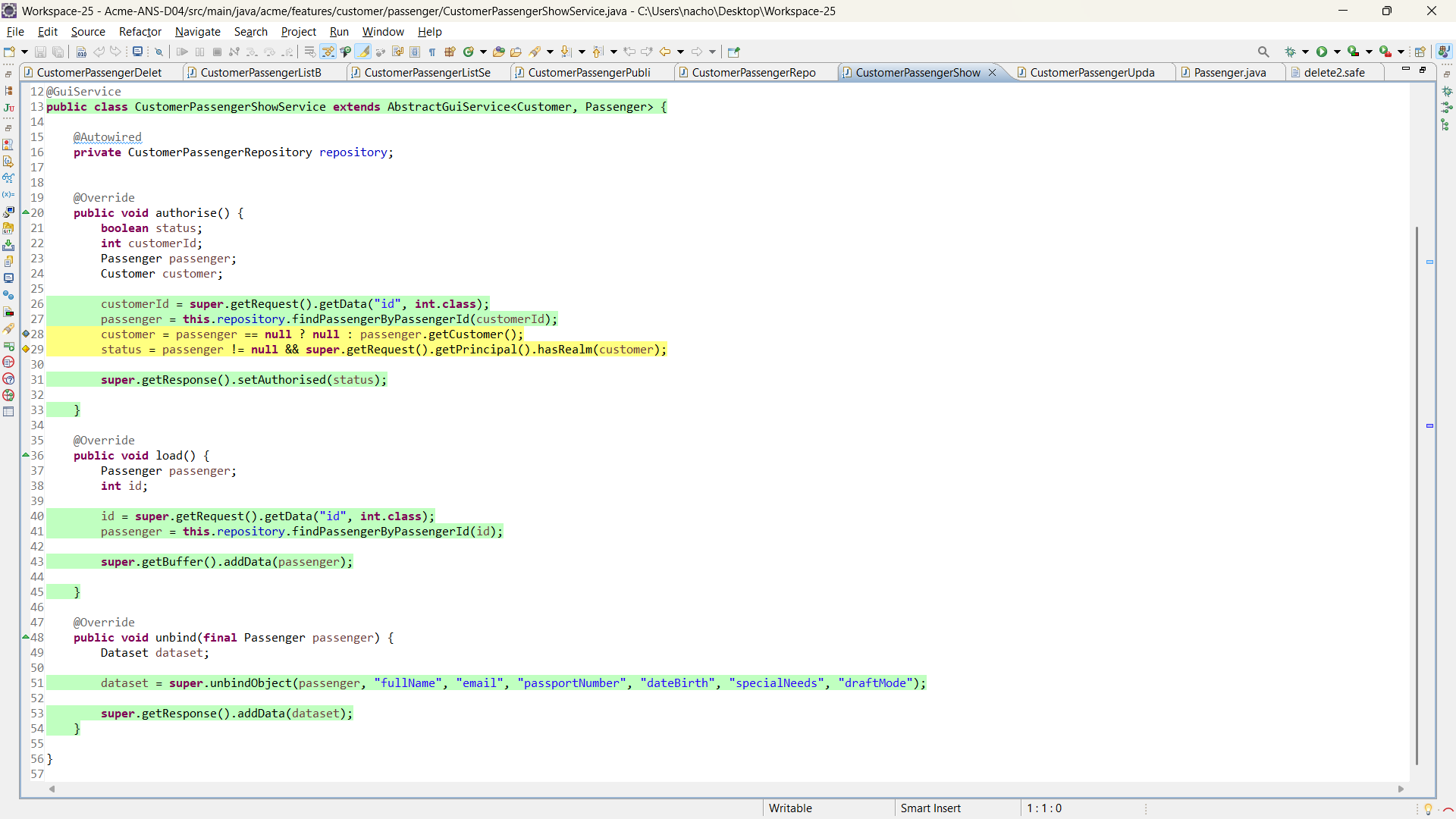
Show

show.safe

* Mostrar información de un pasajero en concreto.

show.hack

* Intentar mostrar información de un pasajero sin estar loggeado.
* Intentar mostrar información de un pasajero loggeado como un customer al que no le pertenece.
* Intentar mostrar información de un pasajero loggeado con un rol distinto a customer.



Se ejecutan todas las líneas menos dos relativas a condiciones del authorise las cuales, como hemos dicho antes, son incompatibles entre sí.

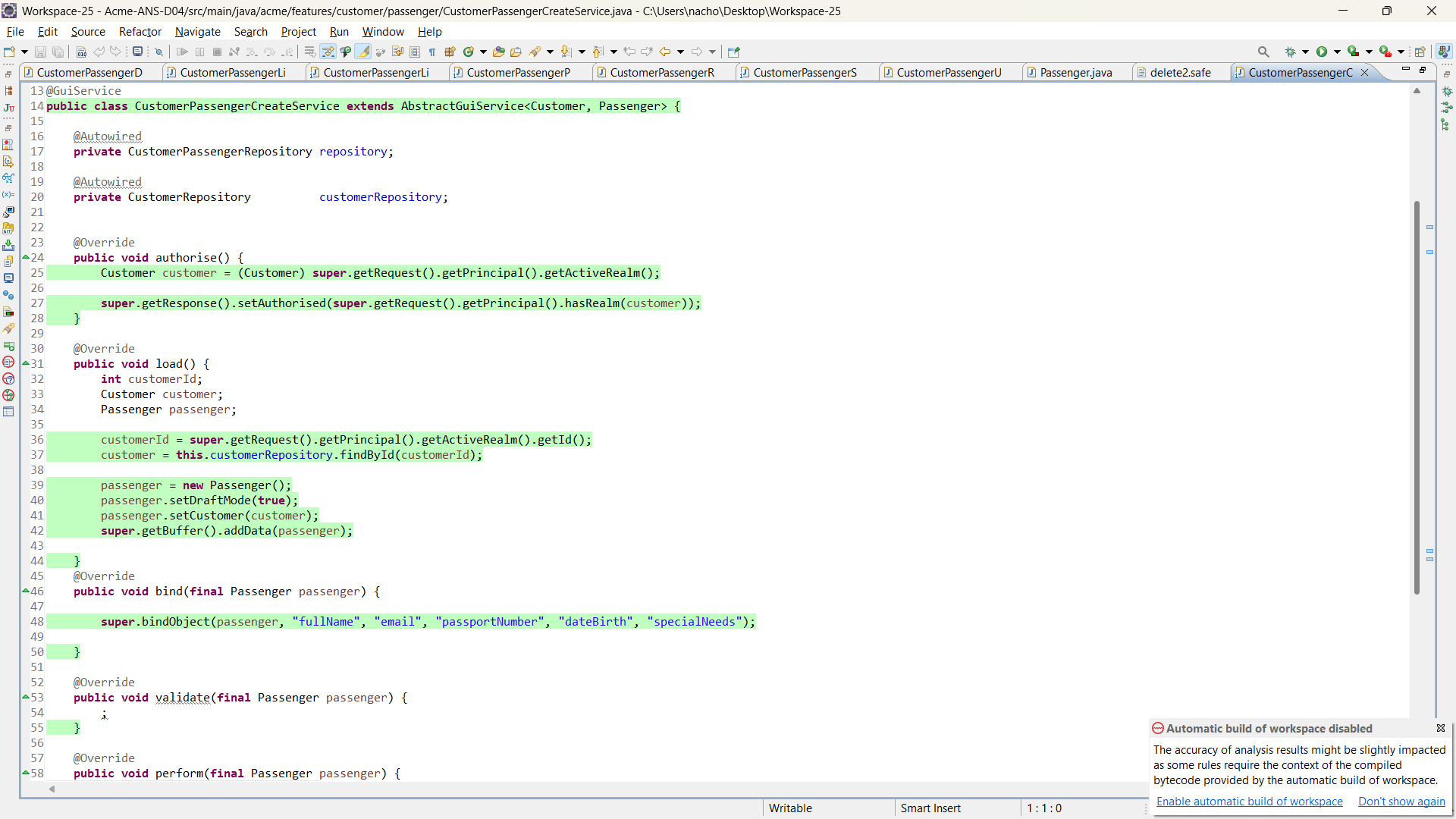
Create

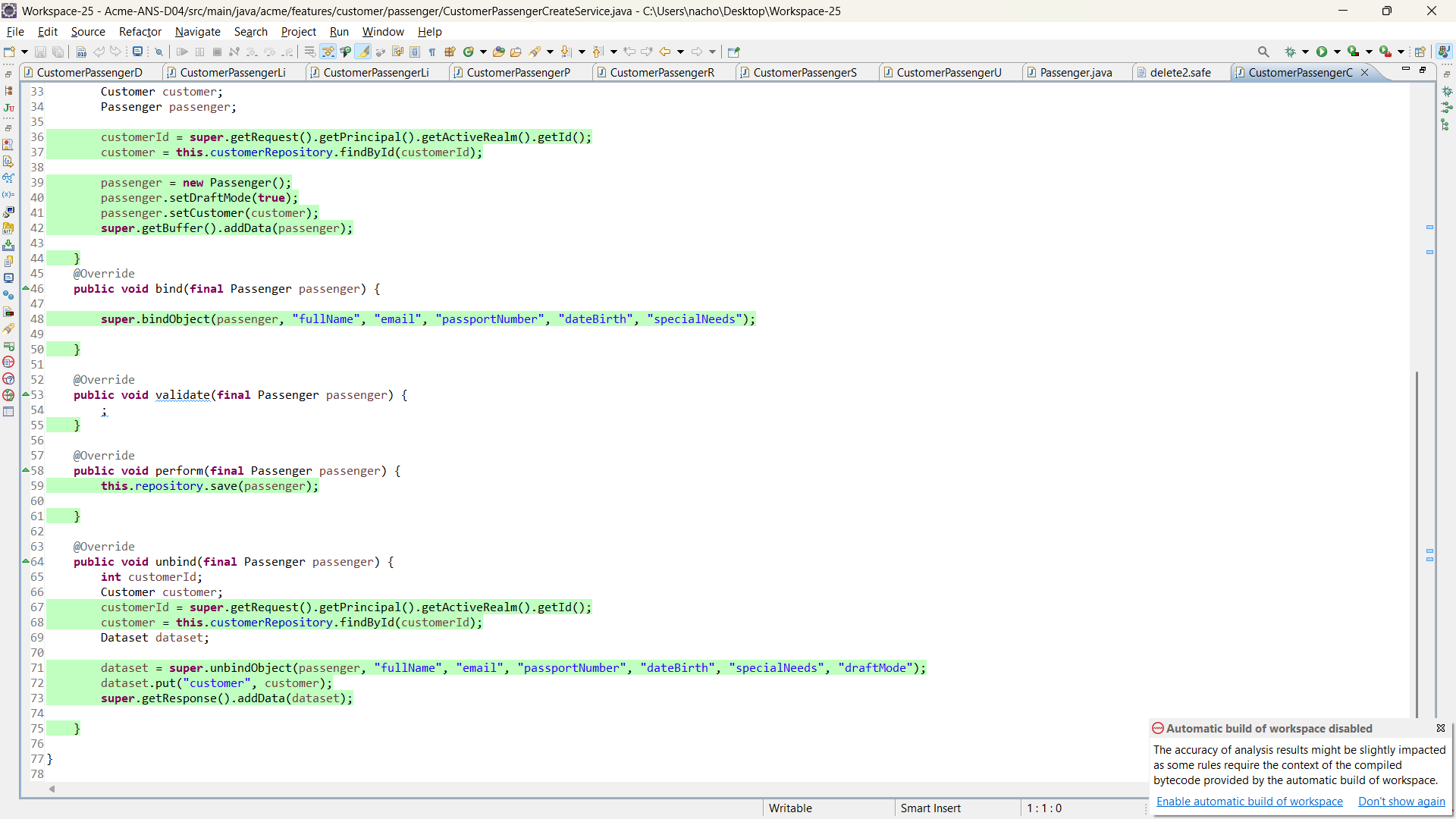
Create.safe

* Crear un Passenger con los datos vacíos.
* Crear un Passenger con toda la variedad de datos erróneos de los atributos.
* Crear un Passenger con datos límites tanto validos cómo inválidos para los atributos.
* Crear un Passenger con todos los datos correctos de los atributos.

Create.hack

* Intentar crear un Passenger sin estar loggeado.
* Intentar crear un Passenger estando loggeado con otro rol.





Se ejecutan todas las líneas de código.

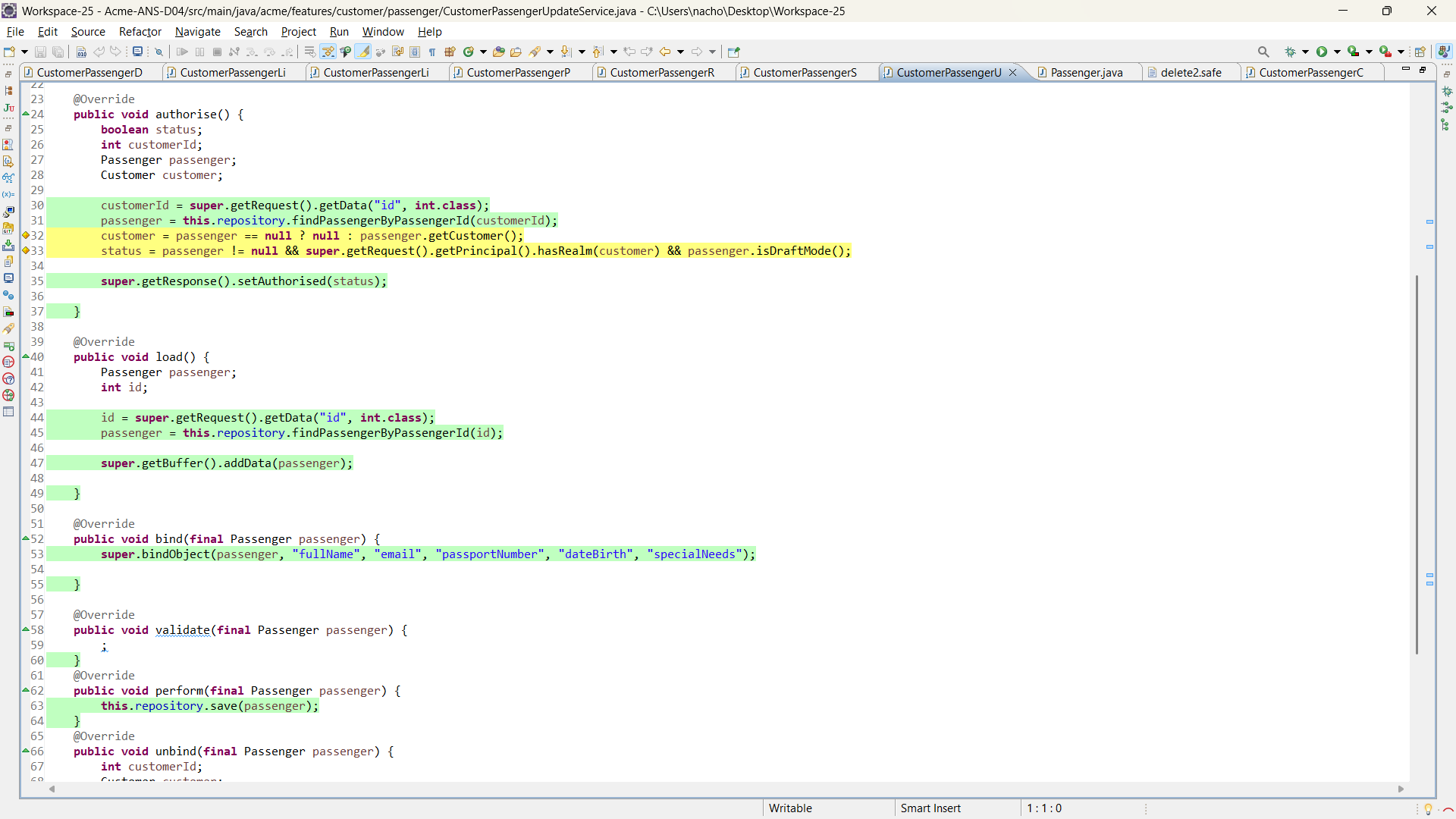
Update

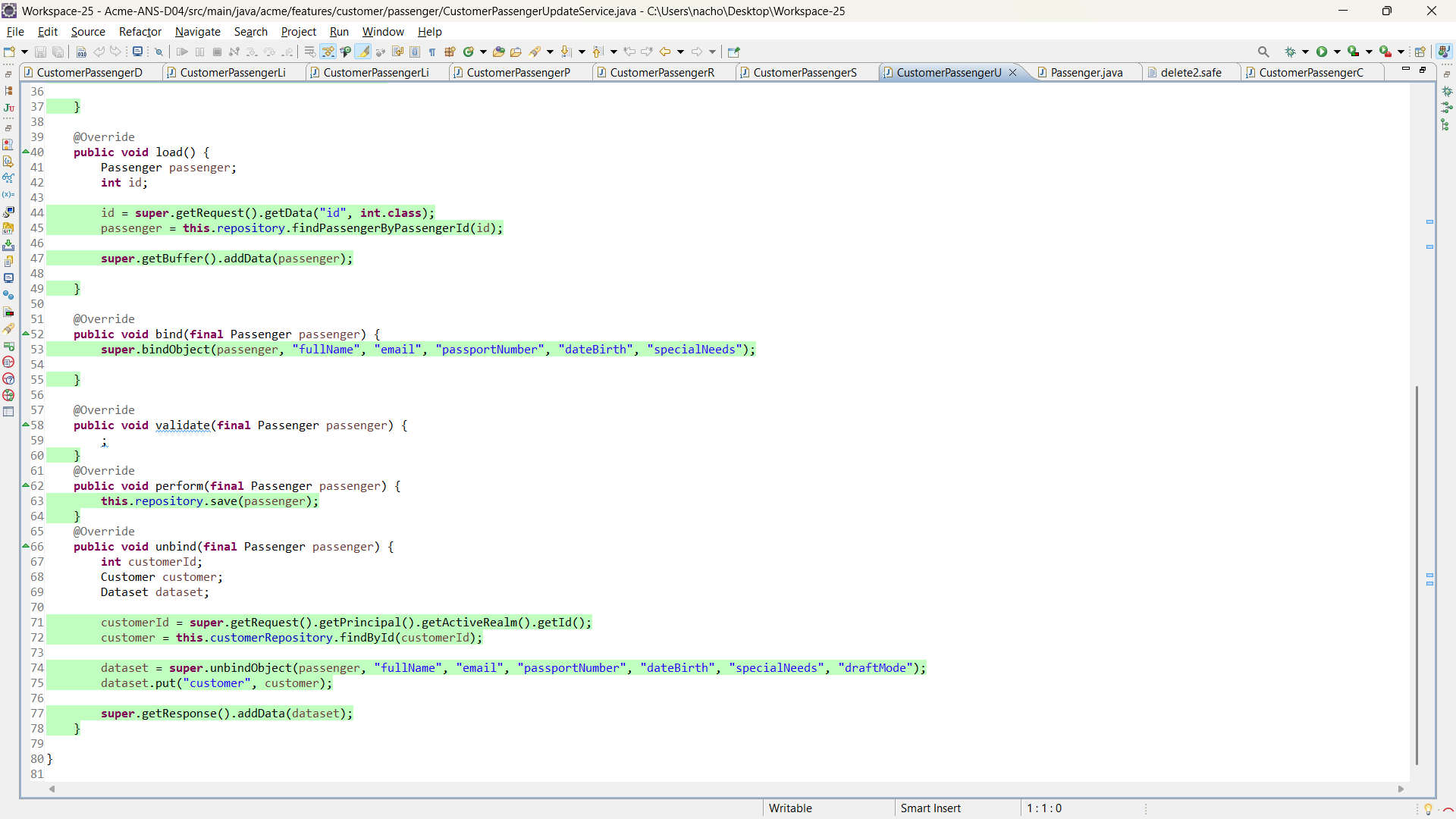
Update.safe

* Actualizar un Passenger con los datos vacíos.
* Actualizar un Passenger con toda la variedad de datos erróneos de los atributos.
* Actualizar un Passenger con datos límites tanto validos cómo inválidos para los atributos.
* Actualizar un Passenger con todos los datos correctos de los atributos.

Update.hack

* Intentar actualizar un Passenger sin estar loggeado.
* Intentar actualizar un Passenger estando loggeado con otro rol.





Se ejecutan todas las líneas menos las relativas al authorise anteriormente comentadas.

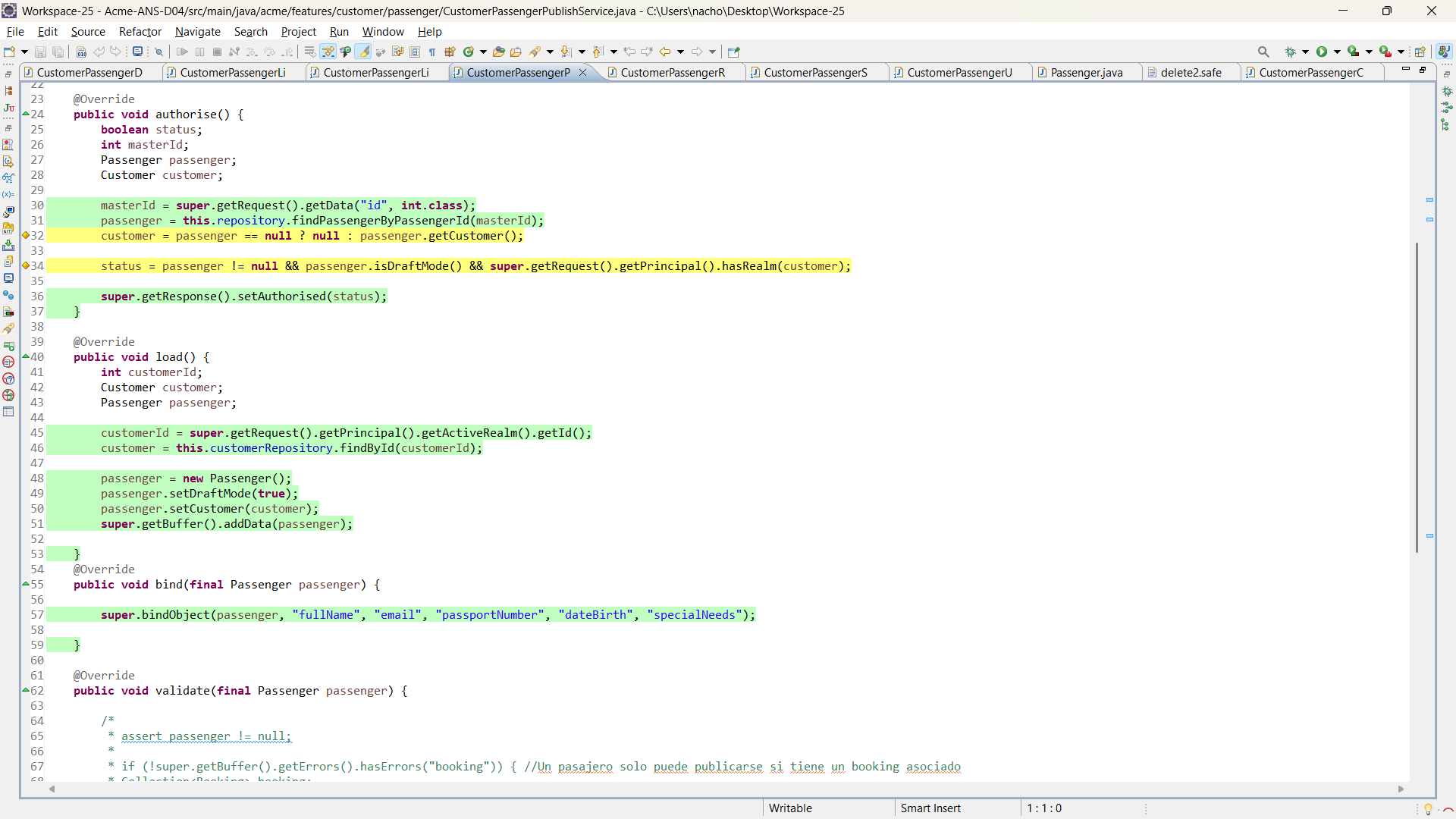
Publish

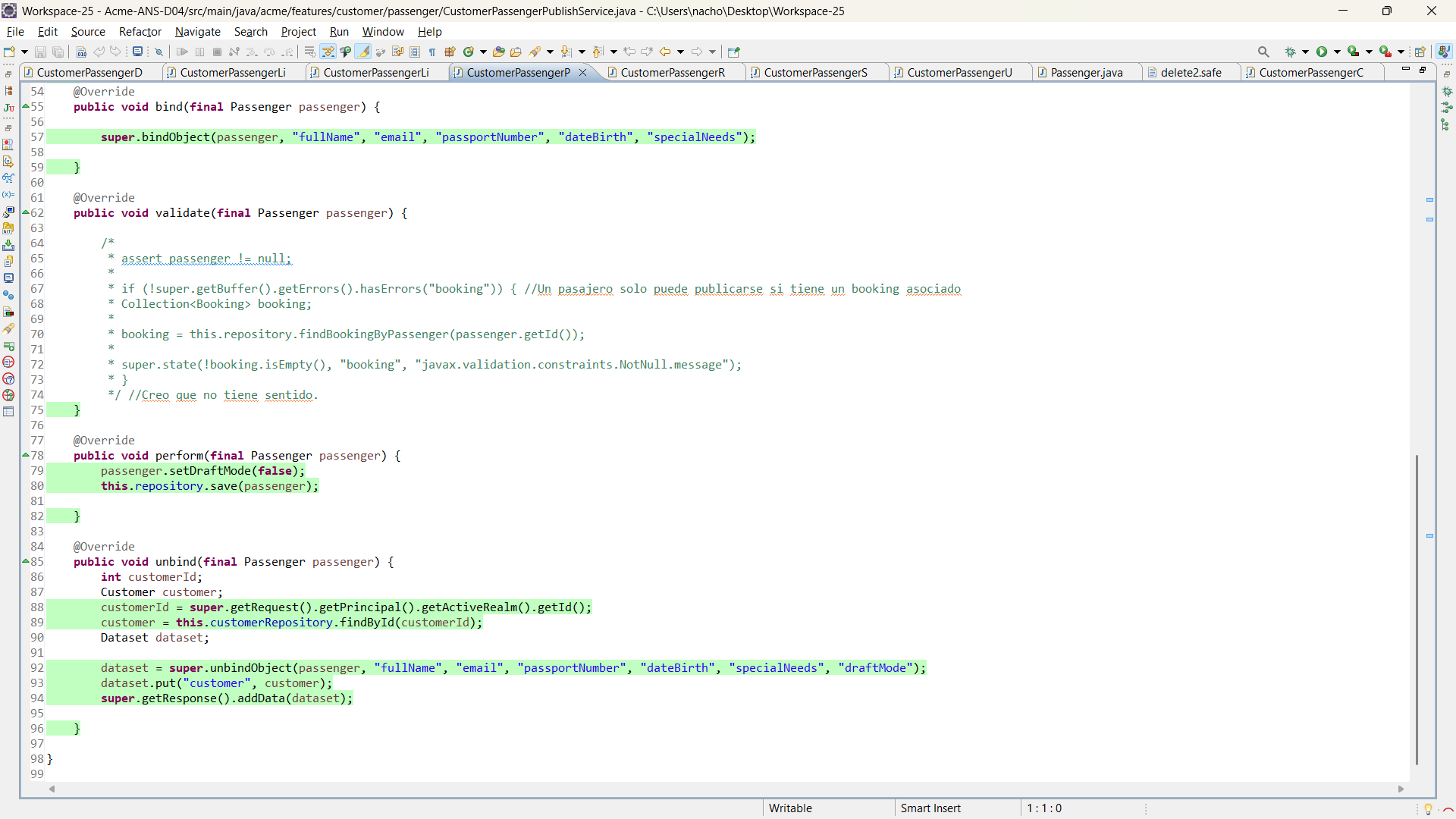
Publish.safe

* Publicar un Passenger que cumpla todas las condiciones necesarias.
* Probar todos los atributos con valores correctos e incorrectos ya que se comporta como un update.

Publish.hack

* Intentar publicar un Passenger sin estar loggeado.
* Intentar publicar un Passenger estando loggeado como otro rol.
* Intentar publicar un Passenger desde un customer al que no le pertenece ese booking.
* Intentar modificar mediante el f12 los valores de un Passenger ya publicado.





Se ejecutan todas las líneas de código menos las del authorise ya comentadas.

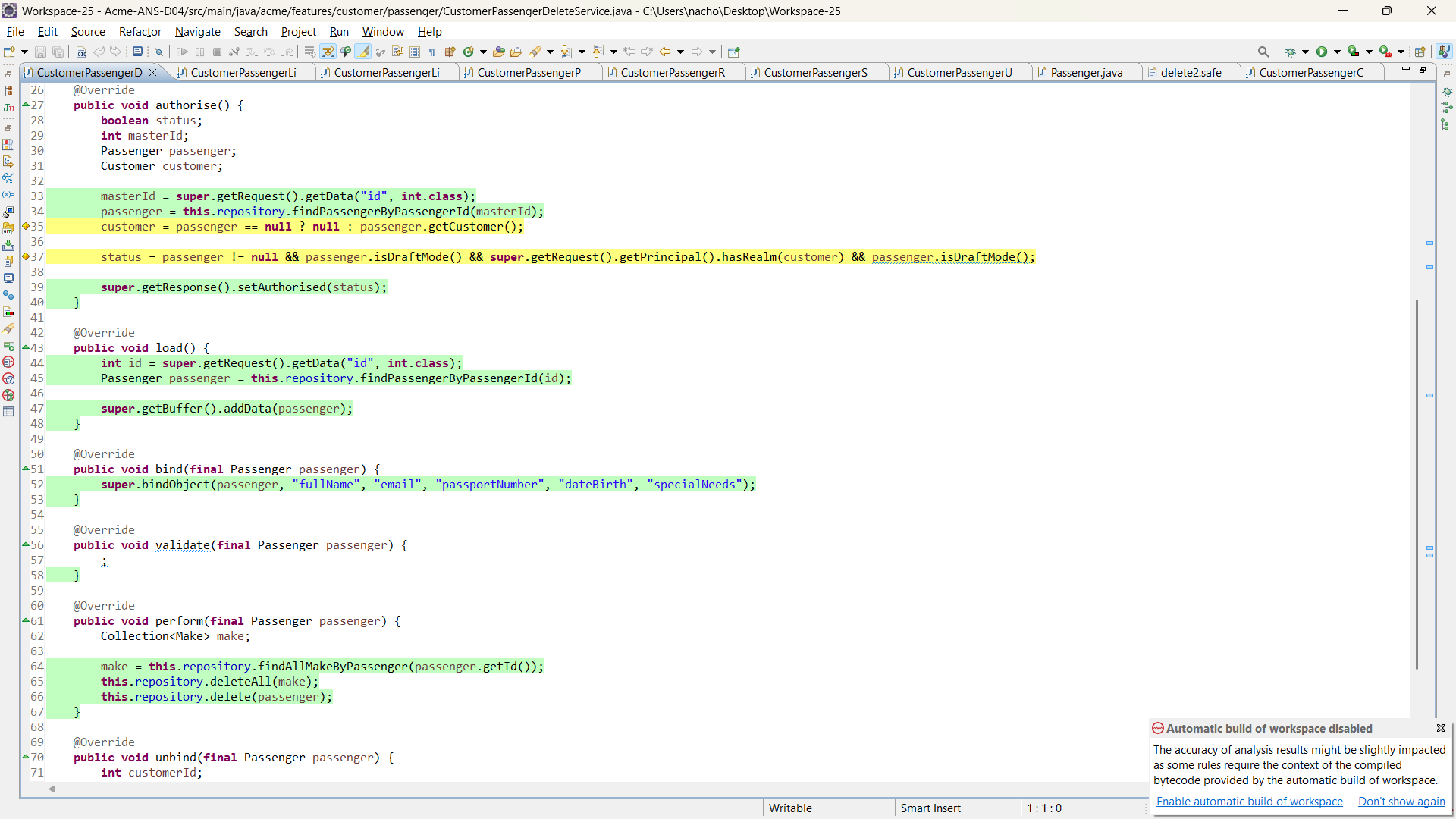
Delete

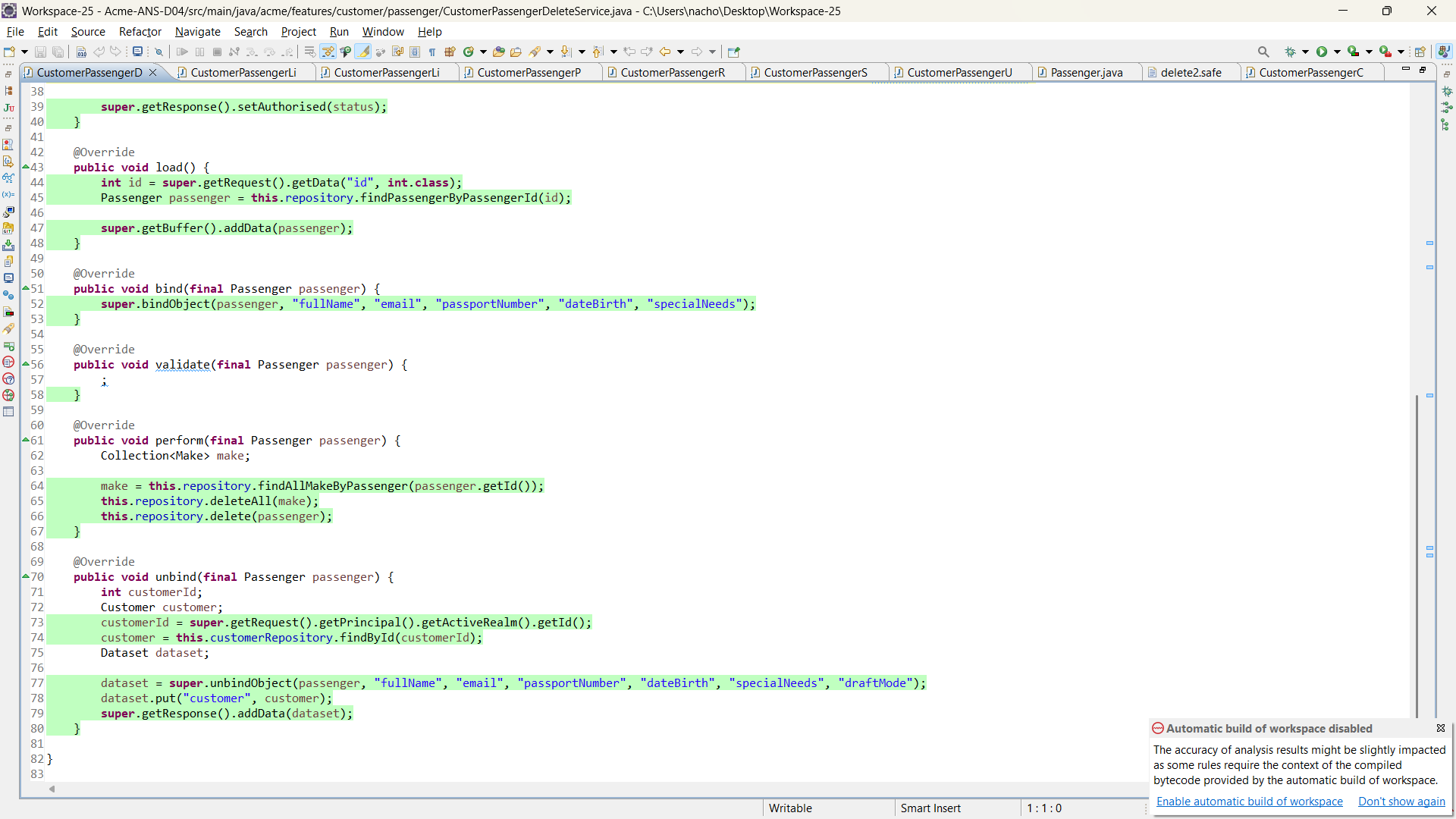
Delete.safe

* Eliminar un Passenger que pertenece al usuario.
* Eliminar un Passenger que pertenece a un booking para probar que se eliminan también las relaciones en la entidad intermedia.

Delete.hack

* Intentar eliminar un Passenger sin estar loggeado.
* Intentar eliminar un Passenger estando loggeado con otro rol.
* Intentar eliminar un Passenger estando loggeado como un customer al que no le pertenece.
* Intentar eliminar un Passenger publicado.





Se ejecutan todas las líneas menos las del authorise ya comentadas.

Controlador

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Se ejecuta al completo el código ya que se trata de todas las urls que se han usado en todos los servicios analizados anteriormente.

Make

La entidad Make se trata de la entidad intermedia usada para relacionar la entidad Booking con la entidad Customer, esta entidad ha sido creada con el objetivo de poder tener relaciones seguras con ambas entidades y se comporta como una entidad normal, es por esto por lo que ha sido sometido a los mismos tipos de tests que si fuese una de las entidades principales.

Create

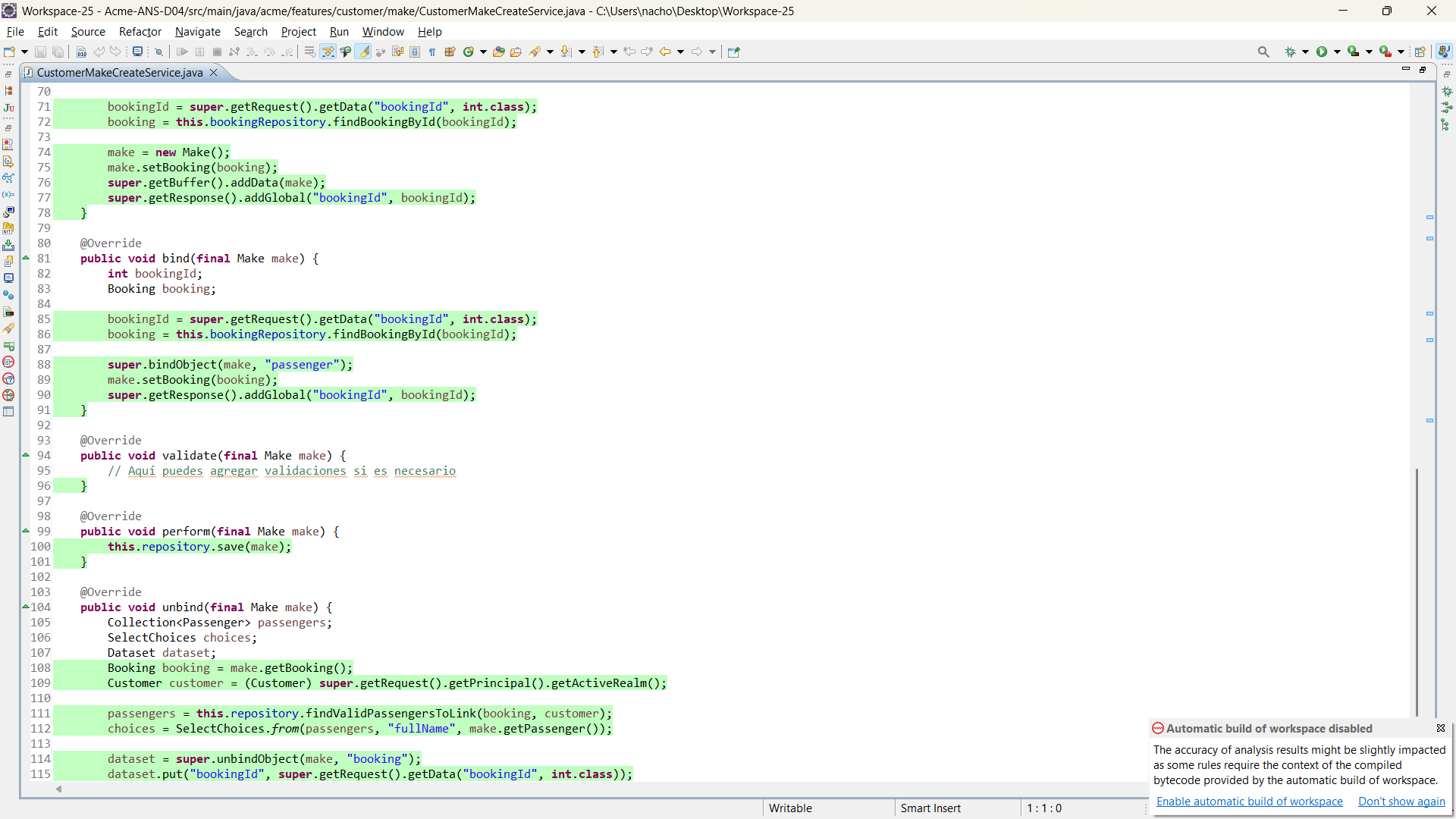
Create.safe

* Linkeo de un Passenger con un Booking.

Create.hack

* Intento de linkear un pasajero que no pertenece al customer.
* Intento de linkear un pasajero que ya ha sido linkeado.
* Intento de linkear un pasajero que no existe.
* Intento de linkear un pasajero sin estar loggeado.
* Intento de linkear un pasajero estando loggeado con otro rol.
* Intento de linkear desde un customer al que no le pertenece el booking.
* Intento de modificar el label del booking usando f12.





Este servicio tiene un problema de ejecución de líneas de código en el authorise, este problema se da debido a que al ser una relación de las dos entidades las posibilidades de hackeo pueden darse desde estas dos entidades, es por eso por lo que se ha de validar que las id coincidan en ambas entidades y eso genera un problema a la hora de conseguir que se den todos los casos de las condiciones a la vez.

A su vez tiene una línea en rojo que realmente es redundante para el bucle al que pertenece pero que es necesaria para que no salte un error de authorise si el usuario no selecciona ningún pasajero y pulsa el botón Link.

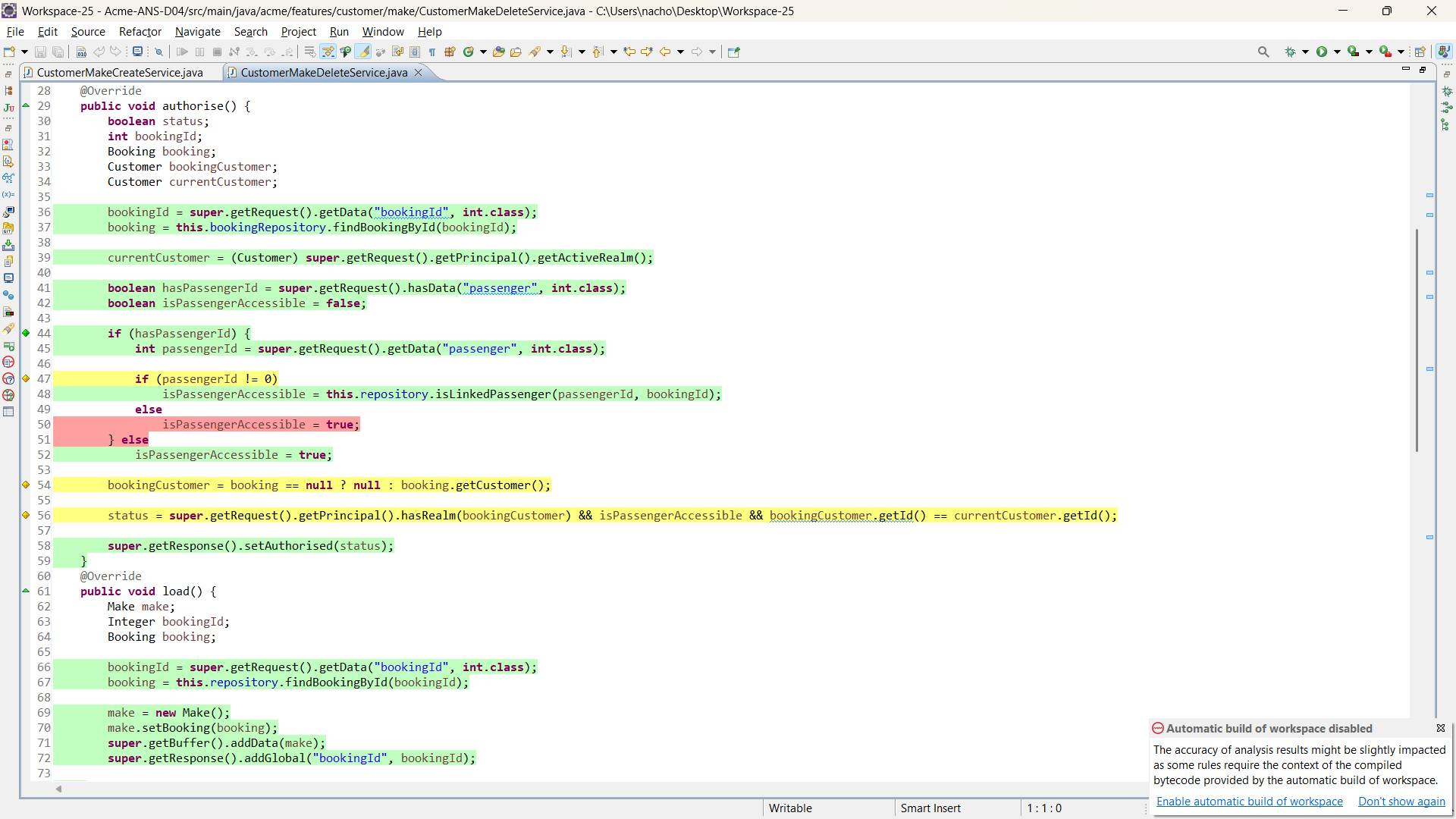
Create

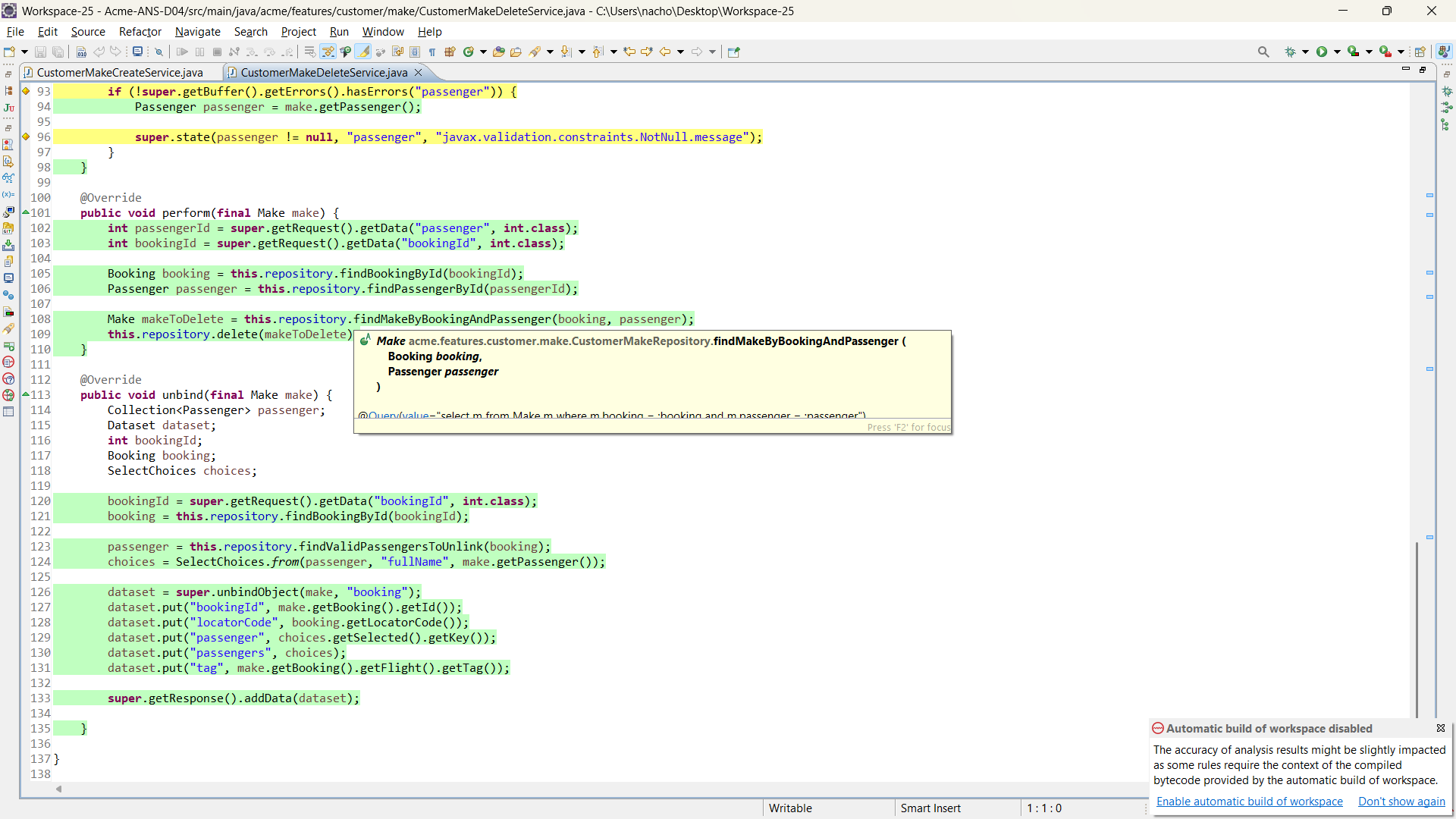
Delete.safe

* Unlinkeo de un Passenger con un Booking.

Delete.hack

* Intento de unlinkear un pasajero que no pertenece al customer.
* Intento de unlinkear un pasajero que ya ha sido linkeado.
* Intento de unlinkear un pasajero que no existe.
* Intento de unlinkear un pasajero sin estar loggeado.
* Intento de unlinkear un pasajero estando loggeado con otro rol.
* Intento de unlinkear desde un customer al que no le pertenece el booking.
* Intento de modificar el label del booking usando f12.





Controlador



Se ejecuta al completo el código ya que se trata de todas las urls que se han usado en todos los servicios analizados anteriormente.

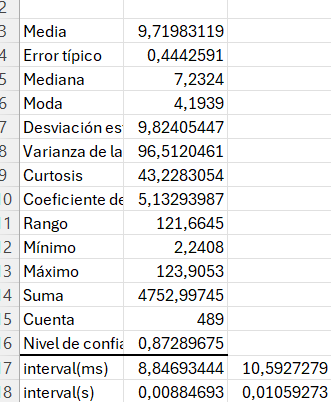
Análisis de desempeño

En este apartado del documento se lleva a cabo la realización de distintas pruebas de rendimiento para evaluar el nivel que tiene nuestro trabajo respecto a este baremo. Para ello se han llevado a cabo distintas pruebas de análisis tanto a las entidades Booking y Passenger como a la entidad intermedia Make.

En la primera de las pruebas se analiza el rendimiento sin los índices necesarios para optimizar las consultas SQL, seguida de esta prueba se realizan los índices necesarios para optimizar estas consultas y se vuelve a realizar el mismo análisis de rendimiento, de este modo se trata de comprobar la efectividad o no de los índices.

Por último, se realiza la prueba con índices en el PC de mi compañero de grupo Alejandro Sevillano Barea, para tener una prueba más en un ordenador con un hardware distinto, ya que esto puede hacer variar los resultados.

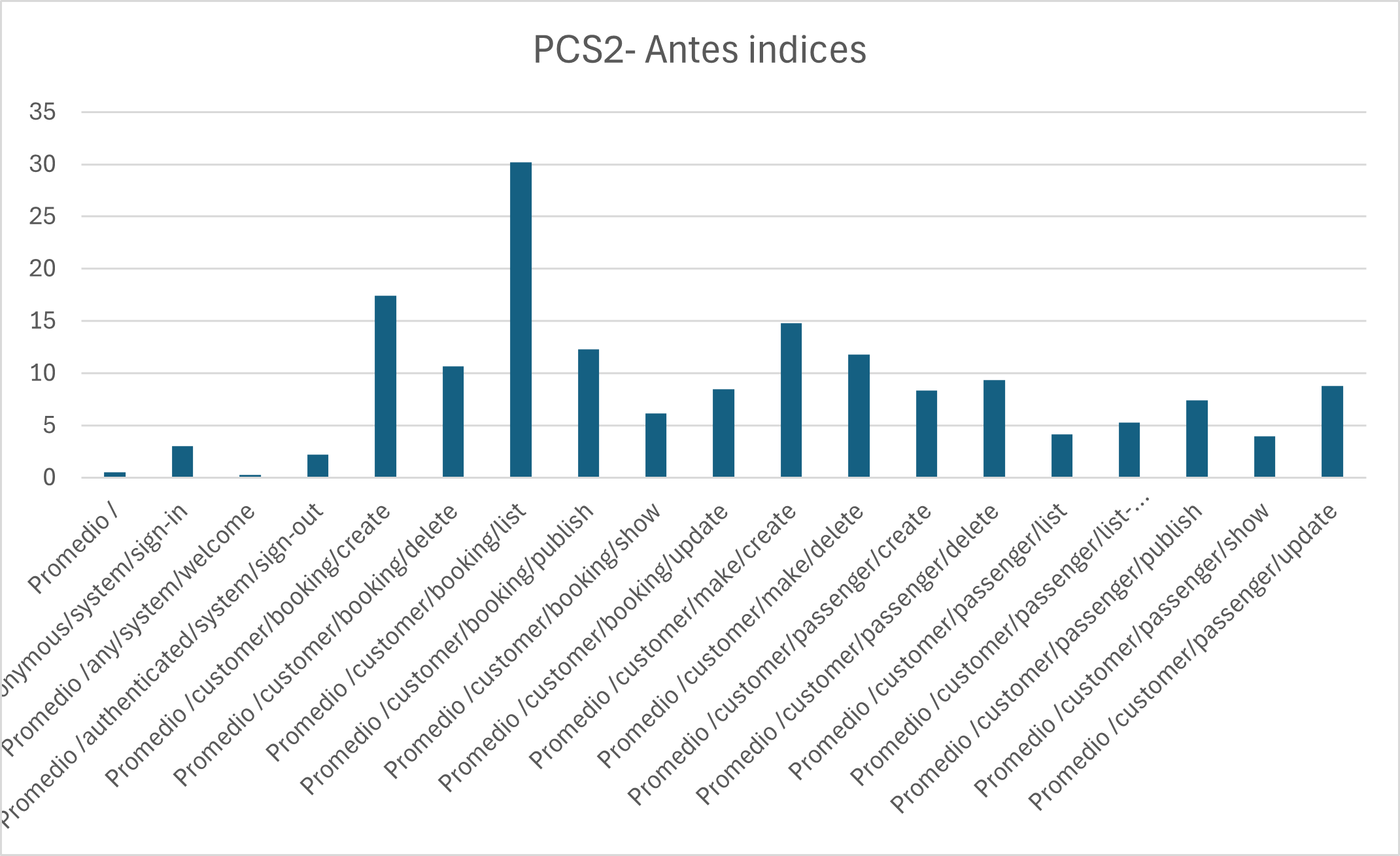
Comparación antes y después de los índices

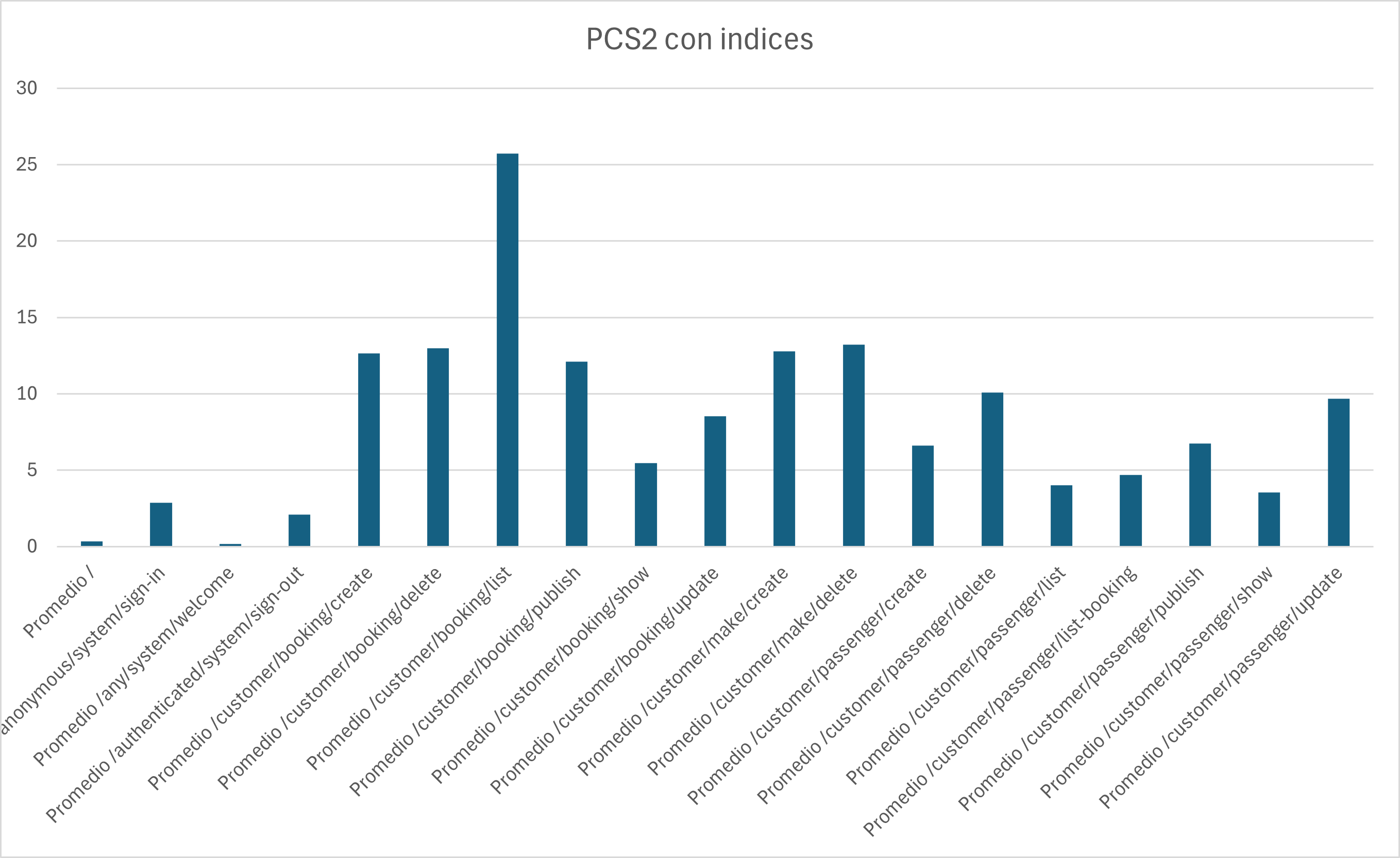




Estos son los resultados obtenidos usando la herramienta de estadística descriptiva proporcionada por Excel. Como podemos comprobar con los datos obtenidos tras la realización de los índices, la varianza, la desviación estándar y la curtosis han bajado sensiblemente tras la realización de los índices, por lo que el rendimiento sí que parece haber mejorado tras su implementación.

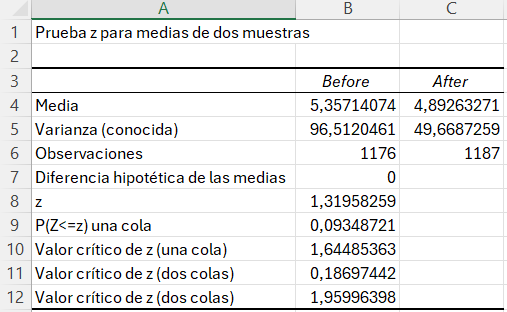
Observemos ahora los histogramas:





En estos histogramas podemos visualizar las variaciones antes vistas de forma numérica por los datos, ya que podemos ver variaciones en las barras, especialmente en las de Booking que son los índices que mayor han contribuido a optimizar nuestro código.

Pasaremos ahora al último análisis estadístico, usando las herramientas de pruebas Z:

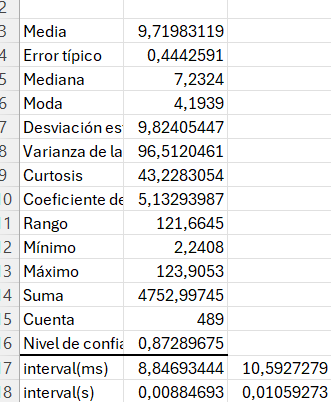


Cálculo realizado para las medias de dos muestras, la columna before corresponde a las pruebas antes de añadir los índices y la after a la realizada con los índices.

Vemos gracias al valor crítico de z (dos colas) que se puede confirmar lo anterior dicho, como el valor esta entre [0,0.5] podemos concluir que efectivamente hemos conseguido optimizar la aplicación con el uso de índices.

Comparativa dos ordenadores

En este apartado voy a realizar una comparativa con los resultados obtenidos tras ejecutar los test de mis servicios en el equipo de mi compañero de grupo Alejandro Sevillano Barea



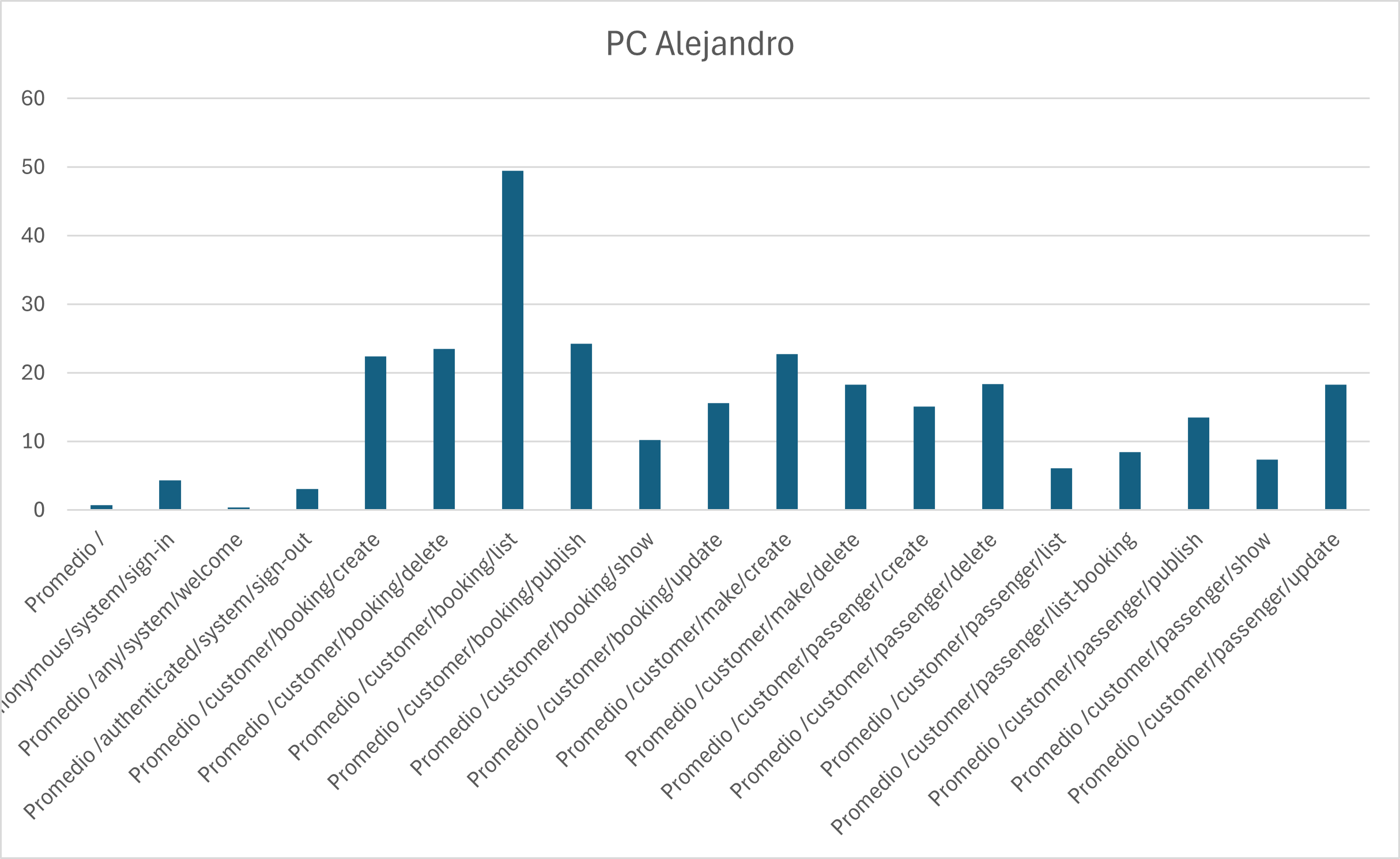


Estos son los resultados obtenidos tras realizar el cálculo utilizando la herramienta de análisis por estadísticas descriptivas. Con estos resultados se puede ver que hay una gran diferencia en todos los valores obtenidos. De este análisis asumimos que mi equipo esta a priori mejor optimizado que el suyo, y que debe de tener mejores prestaciones.

Pasando ahora a los histogramas se podrá observar que se mantienen las hipótesis de estos resultados.

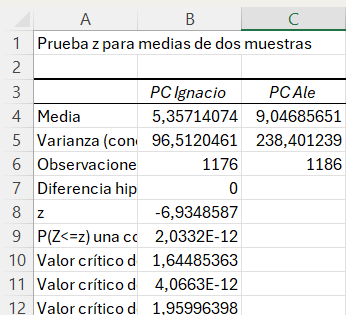
Gráfico, Gráfico de barras

El contenido generado por IA puede ser incorrecto.



Se observa como en algunas consultas los tiempos de ejecución en algunas consultas son prácticamente del doble que en las de mi equipo, lo cual confirma la hipótesis realizada a partir de los datos de análisis descriptivos.

Por último, realizamos un cálculo de prueba Z para los resultados de su pc y los comparamos con los obtenidos en el mío.



Con este cálculo podemos entender si hay realmente una diferencia significativa o no. Nos vamos a fijar en el valor critico de z de (colas) que como podemos ver es muy chico ya que esta elevado a -12, por lo que claramente esta dentro del umbral de (0,alfa), es decir, (0,0.5). Esto nos demuestra que si existe una diferencia significativa entre los dos pc, siendo mi ordenador el que ofrece mejor rendimiento.

Conclusiones

Durante el desarrollo de esta amplio testing hemos podido comprobar que nuestro sistema funciona correctamente y es de la calidad exigida por el cliente a comienzo del curso. Esto ha sido claramente gracias al proceso de testing ya que durante este hemos tenido que corregir diversos errores, algunos de los cuales llevábamos arrastrando gran parte del curso y no hubiésemos detectado sin la realización de estos tests.

Por tanto, con estos tests podemos garantizar que hemos realizado un proyecto sólido y que cumple claramente con todos los requisitos solicitados por el cliente.

En la segunda parte de nuestro documento hemos realizado análisis de rendimiento con herramientas profesionales, lo cual nos ha ayudado a entender la necesidad de hacer un código optimo y de tomar buenas decisiones de diseño.

En conclusión, este documento nos ha permitido adentrarnos en profundidad en lo que es una fase de pruebas profesional y ser consciente de la importancia y utilidad que tiene.

Bibliografía

Intencionalmente en blanco.