MEMORIA PROYECTO JAVA:

Rent a car con interfaz gráfica y conexión a base de datos

Desarrollo de aplicaciones web - CIFP Francesc Borja Moll

María Victoria Peñas Miró

José María Hernández Bruno

1. Alcance

El objetivo del presente documento consiste en realizar una propuesta de proyecto para realizar la programación de un sistema para un *Rent A Car.*

El motivo por el cual se ha decidido realizar el proyecto de esta temática es porqué queríamos optar por algún tema relacionado con el sector turístico.

El proyecto se va a abordar en diferentes FASES:

I. Definición de los casos de uso en UML. *12/Mayo*
II. Diseño conceptual y lógico de la base de datos. *12/Mayo*
III. Diseño del diagrama de clases en UML. *12/Mayo*
IV. Definición de los requisitos funcionales del proyecto. *12/Mayo*
V. Definición del roadmap dónde se marcan los objetivos previstos en las diferentes fases: fase de diseño, desarrollo y pruebas. *12/Mayo*
VI. Implantación de la base de datos. *15/Mayo*
VII. Formación interfaces gráficas.
VIII. Desarrollo de las clases definidas en el diagrama de clases realizado en la fase III.
IX. Documentación de los tests realizados tras la realización de la fase VIII.
X. Desarrollo e implementación de las mejoras propuestas en la fase X.
XI. Testeos end-to-end. *1/Junio*
XII. Creación de un vídeo con la presentación del proyecto.
XIII. Redacción de las conclusiones y comentarios acerca de lo que se ha aprendido durante la realización del proyecto, dificultades encontradas, etc para cerrar las memorias.

El código fuente y la documentación de este proyecto está subido al siguiente repositorio de GitHub:
https://github.com/cifpfbmoll/proyecto-1o-daw-equipo-vicky-y-chema

2. Descripción General

Consiste en la recapitulación y descripción de todos los factores que afectan a la aplicación y sus requisitos. De esta manera se tiene una visión global de la aplicación, de las funciones que debe realizar, de los usuarios que existen, así como de las restricciones existentes de la aplicación.

• Perspectiva del producto

El producto a desarrollar es una simulación de un programa para un “Rent A Car” elaborado en el lenguaje de programación “Java” para consolidar lo aprendido en el primer año de “Desarrollo de aplicaciones Web” en el centro formativo “CIFP Francesc de Borja Moll”.

El objetivo es crear una aplicación estable, con acceso a base de datos, escritura en un documento e interfaz gráfica, para ésta tarea hemos elegido Swing.

- # Funciones del producto

La versión estable del producto será capaz de, mediante una interfaz sencilla, permitir la gestión de una empresa para el alquiler de vehículos vacacionales (rent a car), haciendo posible que el cliente pueda acceder a un listado de vehículos y después de rellenar un formulario poder reservar el vehículo deseado.

En la aplicación se podrá realizar diferentes funciones dependiendo del tipo de usuario o rol que esté accediendo. De esta manera, un usuario no registrado en el sistema podrá realizar consultas de los vehículos existentes, por otro lado, un usuario identificado podrá realizar consultas, reservas, y cancelaciones y, por último el administrador del sistema podrá realizar todas las acciones comentadas además de retirar y añadir vehículos, así como realizar modificaciones de precios.

- # Pantallas:

Se van a desarrollar la siguientes pantallas o ventanas:

1. Menú principal.
    a. Registro.
    b. Login.
    c. Buscador de vehículos.
        i. Listar.
        ii. Modificar.
        iii. Eliminar.
2. Formulario registro de vehículos.
3. Formulario de reserva.
    a. Listar.
    b. Eliminar.

- # Mejoras propuestas

1. Generación de bonos de confirmación y cancelación.

Al realizar una reserva se generará un fichero .txt con los datos de confirmación o de cancelación según corresponda.

2. Reservar más de un vehículo en una misma Reserva

Se podrá seleccionar más de un vehículo y añadirlo en la misma reserva.

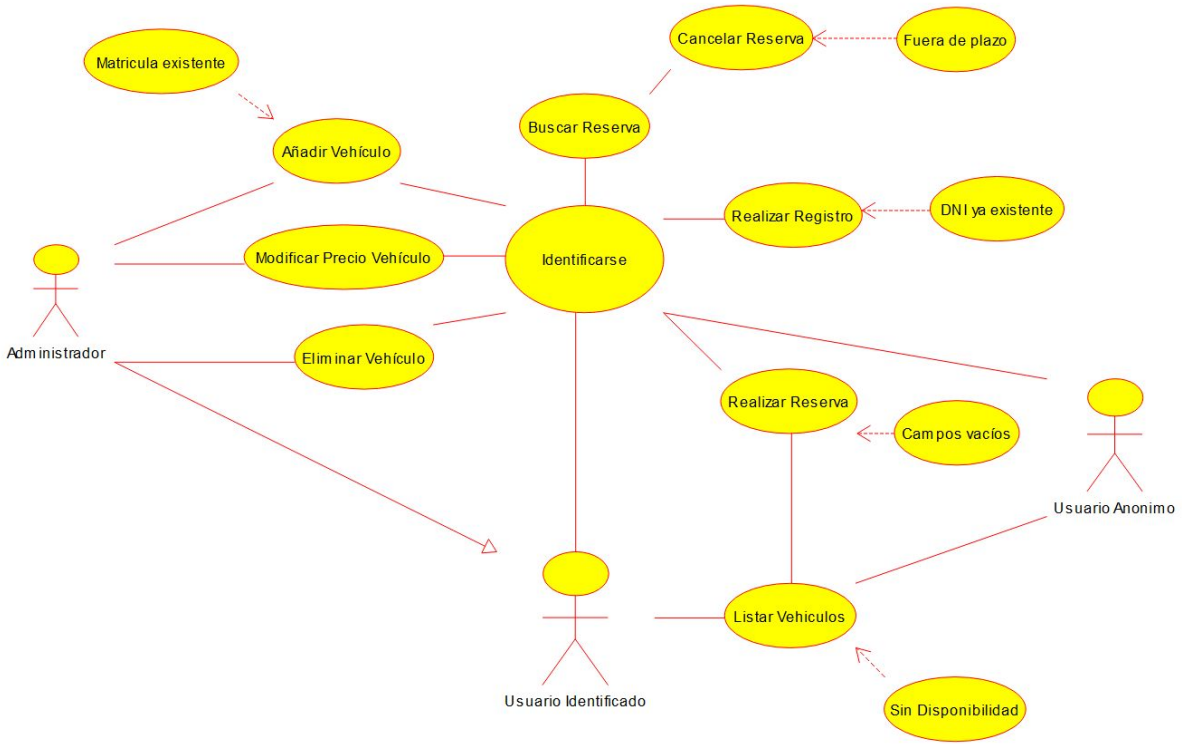
3. Registro logs BBDD

   Cada vez que se elimine o actualice un vehículo quedará almacenado en una tabla de logs en la base de datos.

3. FASES

A continuación, se detalla la evolución de las diferentes fases que componen el presente proyecto:

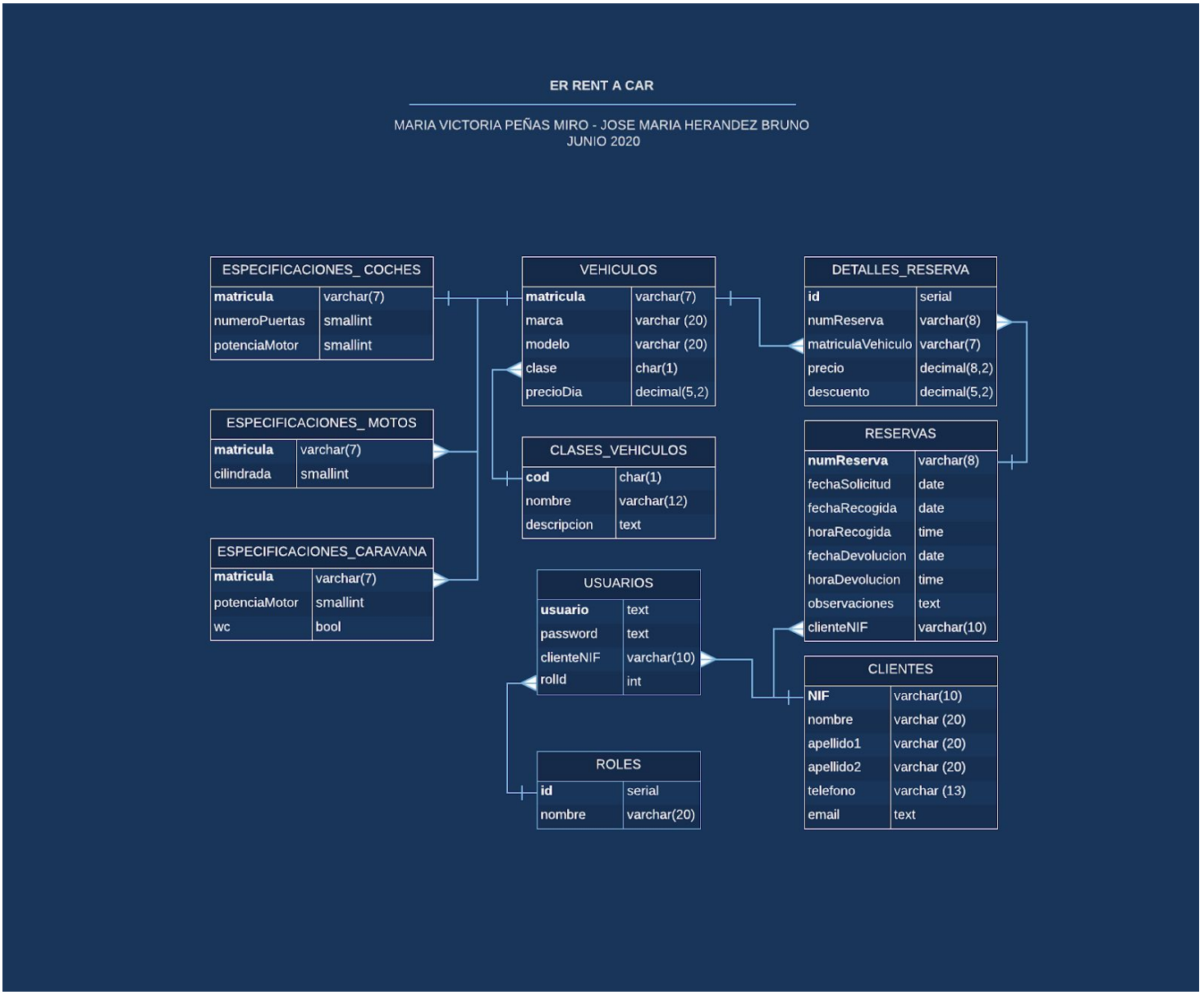
FASE I: Definición de los casos de uso en UML.



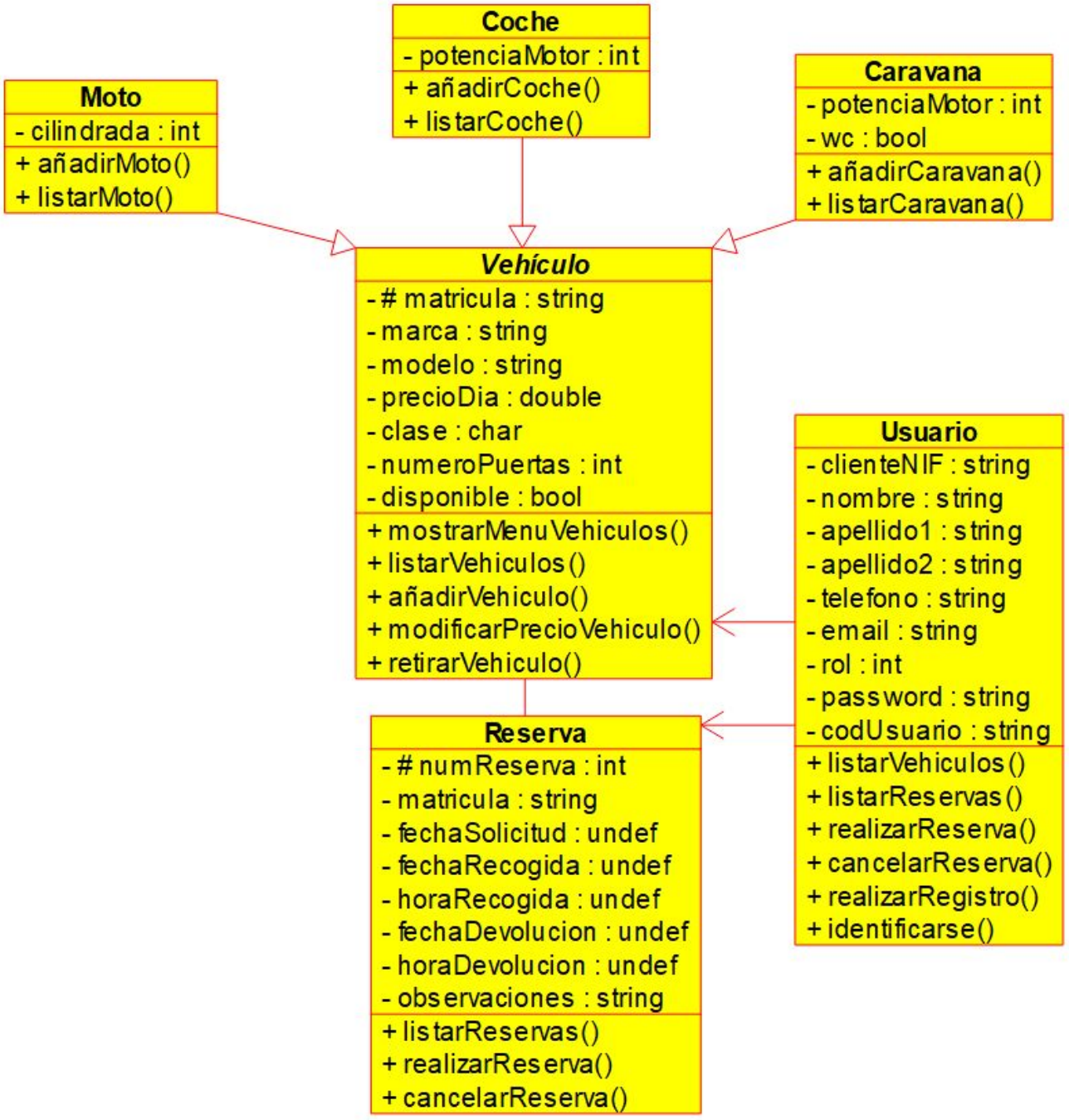
FASE II: Diseño conceptual y lógico de la base de datos.

Hemos optado el gestor de base de datos postgreSQL.

Se realizará la instalación de la base de datos en un servidor en la nube.



FASE III: Diseño del diagrama de clases en UML.



FASE IV: Definición de los requisitos funcionales del proyecto.

Listar Vehículos	
Versión	1.0
Objetivos	Mostrar vehículos.
Descripción	Muestra por pantalla los vehículos, sus características y precio, permitiendo seleccionar el vehículo deseado.
Precondición	El programa debe haber arrancado correctamente.
Secuencia normal	Si el cliente está identificado, en caso de elegir uno disponible se procederá a la reserva. En caso de ser un usuario anónimo, en caso de elegir uno disponible se solicitará el registro como usuario o que se identifique.
Secuencia alternativa	En caso de seleccionar un vehículo no disponible, se lanzará una excepción (RCarException).
Frecuencia	Alta, cada vez que se desee realizar una reserva o informarse de stock actual.

Realizar Reserva	
Versión	1.0
Objetivos	Reservar vehículos.
Descripción	Mediante un formulario que deberá cumplimentar el cliente, permitirá el alquiler de un vehículo.
Precondición	- Haber seleccionado un vehículo mostrado anteriormente en “Listar Vehículos”. - Estar Identificado en el sistema.
Secuencia normal	Una vez cumplimentado los datos, la reserva estará completada.
Secuencia alternativa	En caso de haber dejado un o varios campos vacíos, mostrará un mensaje solicitando que se rellenen dichos campos.
Frecuencia	Media, siempre que un usuario quiera reservar un vehículo.

Identificarse	
Versión	1.0
Objetivos	Identificar un usuario en el sistema.
Descripción	Mediante el número de NIF y una contraseña, el usuario podrá identificarse en el sistema.
Precondición	El programa debe haber arrancado correctamente.
Secuencia normal	Se mostrará una pantalla de Login con dos campos (donde introducirá NIF y contraseña) y dos botones (“Identificarse” y otro de “Registrarse”).
Secuencia alternativa	En caso de usuario inválido o contraseña incorrecta se lanzará una excepción (RCarException).
Frecuencia	Media, siempre que el usuario quiera acceder al sistema para algo más que Listar Vehículos.

Realizar Registro	
Versión	1.0
Objetivos	Registrar un nuevo cliente en el sistema.
Descripción	Se mostrará al usuario un formulario a rellenar para guardar dichos datos.
Precondición	El usuario debe haber solicitado el registro mediante el botón “Registrarse”.
Secuencia normal	El usuario introduce los datos solicitados y al enviar el formulario aparecerá un mensaje de “registrado correctamente”.
Secuencia alternativa	En caso de ya existir el NIF en el sistema, se lanzará una excepción (RCarException).
Frecuencia	Baja, sólo cuando un usuario no existente desee registrarse en el sistema.

Buscar Reserva	
Versión	1.0
Objetivos	Recuperar reservas existentes.
Descripción	A través de un buscador, se podrá realizar una búsqueda de las reservas realizadas.
Precondición	Haberse identificado.
Secuencia normal	El usuario podrá solicitar un listado de reservas existentes.
Secuencia alternativa	Si se intenta recuperar una reserva que no existe, se lanzará una excepción (RCarException).
Frecuencia	Baja, ejecutada por el usuario cuando quiera buscar reservas para informarse o bien cancelar una.

Cancelar Reserva	
Versión	1.0
Objetivos	Cancelar reservas en curso
Descripción	A través de un buscador de reservas, se podrá cancelar una reserva
Precondición	Siempre y cuando no se haya recogido el vehículo.
Secuencia normal	El usuario tendrá un botón capaz de cancelar la reserva una vez haga la búsqueda de reservas.
Secuencia alternativa	Si se intenta cancelar una reserva pasada se lanzará una excepción (RCarException).
Frecuencia	Baja, sólo cuándo el usuario requiera de cancelar una reserva

Añadir Vehículo	
Versión	1.0
Objetivos	Añadir vehículos a la flota de la empresa
Descripción	A través de un formulario se podrá añadir un vehículo a la flota
Precondición	-
Secuencia normal	El administrador será capaz de rellenar un formulario para añadir el vehículo al sistema
Secuencia alternativa	Si se intenta añadir un vehículo con una matrícula ya existente se lanzará una excepción (RCarException)
Frecuencia	Media, cuándo el administrador añada vehículos al sistema

Modificar Vehículo	
Versión	1.0
Objetivos	Modificar el precio de vehículos.
Descripción	El administrador será capaz de modificar el precio del vehículo.
Precondición	El vehículo a modificar debe haber sido añadido anteriormente.
Secuencia normal	El administrador accederá al listado de vehículos dónde podrá modificar el precio del mismo.
Secuencia alternativa	-
Frecuencia	Baja, sólo cuándo el administrador desee cambiar el precio.

Eliminar Vehículo	
Versión	1.0
Objetivos	Retirar el vehículo
Descripción	El administrador será capaz de retirar un vehículo del sistema.
Precondición	El vehículo a retirar debe haber sido añadido anteriormente.
Secuencia normal	El administrador podrá acceder a un listado de vehículos para retirarlo, seleccionará el deseado y podrá modificar y guardar cambios.
Secuencia alternativa	-
Frecuencia	Baja, sólo cuándo el administrador desee retirar un vehículo.

FASE V: ROADMAP.

A continuación se detalla la planificación prevista para la realización de este proyecto:

FASE VI: Implantación de la base de datos.

El script de creación de la base de datos y de las tablas se puede consultar desde el siguiente enlace:

https://github.com/cifpfbmoll/proyecto-1o-daw-equipo-vicky-y-chema/blob/master/rentacar.sql

FASE VII: Formación interfaces gráficas.

Nos hemos basado en la formación que imparte el señor Juan Díaz, creador de contenido audiovisual en Youtube, se puede acceder desde el siguiente enlace.

FASE VIII-IX: Desarrollo y programación. Testeos realizados:

14/Mayo

- *Desarrollo clase Usuario*

Se ha creado la clase Usuario y se ha empezado a crear la ventana con swing dónde se pintará el formulario con la solicitud de los datos.

Para construir el formulario de registro, se necesitarán dos nuevas clases: Formulario y Lamina.

15/Mayo

- *Desarrollo de las clases Formulario y Lamina.*

Se han creado las clases Formulario y Lamina que se utilizarán para diseñar las diferentes ventanas que se utilizarán para pedir los datos al usuario final.

Se ha realizado el diseño del formulario de registro.

- *Conexión a la BBDD.*

Se realiza la primera prueba de conexión a la BBDD desde JAVA, ha funcionado correctamente.

- *Programación ActionEvent formulario de registro.*
  Se ha realizado la configuración del JButton del formulario de registro para insertar clientes en la BBDD.

  Se intenta cargar la query directamente en el PreparedStatement, pero falla. Se construye siguiendo el formato habitual con “?” para setear los valores y ahora funciona correctamente. Ya se insertan nuevos registros.

  PROBLEMA! El JTextField teléfono al convertirse a integer, lanza una excepción cuando enviamos el campo vacío o con valor 0.

Finalmente, resolvemos el problema anterior haciendo la conversión a Integer en el seteo del valor para hacer el insert.

20/Mayo

- *Acordamos utilizar los JFrame que ofrece netBeans para trabajar con swing.* Las clases desarrolladas anteriormente las declaramos como deprecated y empezamos de nuevo.
- *Actualizamos diferentes archivos de interfaz mediante el uso de JFrame de NetBeans así como los Internal Frames que nos permite tener una estética de "sistema operativo" dónde podemos mover las ventanas internas, minimizarlas y cerrarlas sin que afecte a la ventana principal*

21/Mayo

- *Proceso GIT.* Se establece el siguiente proceso de trabajo:
    - Se va a trabajar mediante ramas. Se creará una rama por funcionalidad. El proceso será el siguiente:

      Crear una rama, desarrollar funcionalidad, al finalizar merge a master.

  El flujo será:

  1. De master, crear una rama.
     git branch nombrerama
     git checkout nombrerama

  2. Desarrollar y commitear sobre esa rama.
     git add .
     git commit -m "comentarios"

  3. Subir esa rama al repositorio.
     git push --all

  4. Hacer merge de esa rama a master.
     git checkout master
     git merge nombrerama

  5. Hacer pull de master en local (para tener los últimos cambios descargados).
     git checkout master
     git pull
     git push

  6. Volver al paso 1 para desarrollar otra cosa.

- *Ventana ListarVehiculos.* Se ha creado el internal JFrame dónde se listarán todos los vehiculos: Interfaz_ ListarVehiculos. Los vehiculos se muestran correctamente, así

como el recuento del total.

- *Ventana Home.* Se ha creado la interaz de la página principal del programa con tres opciones: Acceder, registrarse y Listar vehiculos.
- *Configuración de ventana login con 3 opciones (Consultar Vehiculos, Registrarse e Identificarse).*

22/Mayo

- *Interfaz Login.* Se ha creado la pantalla de login y se ha configurado la validación del usuarios por BBDD para que pueda acceder a la aplicación. Se ha vinculado la Interfaz_Administrador.
- Interfaz Cliente. Se ha creado una pantalla para el post-login de un usuario cliente con diferentes opciones, un fondo de pantalla y un botón de cerrar sesión.

- *Ventana Home.* Se ha configurado el Main y personalizado la ventana. A partir de ahora, para acceder a la pantalla de administrador debe hacerse con credenciales a través de esta ventana.

- *Modificación clase Vehiculo.* El atributo numeroPuertas se definió inicialmente en la clase abstracta, se ha eliminado y se pondrá en las subclases Coche y Caravana.

23/Mayo

- *Subclases de Vehiculo.* Se ha decidido que el método registrarVehiculo sea abstracto. Inicialmente se había pensado que en la clase padre se hiciera el insert en la tabla vehículos y posteriormente en las hijas se hiciera el insert en su correspondiente tabla de especificaciones, sin embargo, como todo debe estar en una transacción, se harán ambos inserts desde las clases hijas.

- *Eliminamos atributo "disponible" de la clase Vehiculo.* Este atributo se creó para detectar si un coche está reservado o no, pero finalmente realizaremos este chequeo con las fechas de reserva.

- Ya se insertan los vehículos y especificaciones de vehículos en BBDD correctamente.

- *Eliminar vehiculos.* A la hora de realizar las funciones para eliminar los vehículos de la BBDD, nos hemos dado cuenta que era necesario modificar el esquema de vehículos para una columna donde controlaremos si el vehículo de ha dado de baja o no.

24/Mayo

- *Modificar Precios vehiculos.* Se ha configurado la opción de que un administrador pueda modificar los precios de un vehículo, mediante la clase JOptionPane. Aparentemente todo funciona correctamente.

26/Mayo

- *Creación clase ModeloTabla* para pintar las tablas que se utilizarán en toda la aplicación.

27/Mayo

- *Interfaz_ListarClientes.* Se configurar la interfaz listar clientes con JTable. Aparentemente, la tabla se rellena correctamente con el Resultset, pero no se muestran los resultados en el JFrame.

  Finalmente, hemos conseguido mostrar los resultados modificando el layout del InternalFrame a overlay, sin embargo, los resultados se pintar detrás de los Jbutton del Frame administrador. Se han realizado varias pruebas con diferentes métodos que ofrece JScrollPane (movetoFront(), jugando con los layouts) pero el error no se corrige. Si en vez de utilizar JInternalFrame, se utiliza JFrame funciona correctamente.

  Creemos que es un error de layout en el JScrollPane.

28/Mayo

- *Interfaz_ListarClientes.* Configuración de un botón para buscar un cliente por NIF.

  No se pueden cerrar los recursos de PreparedStatement y Resultset, ya que sino no se muestran los datos en el Frame y salta una excepcion. Comentado con Rafa.
- Viendo que las JFrame tienen el método “dispose” hemos optado por usarlo ya que cierra todo lo que esa instancia ha abierto previamente → enlace

29/Mayo

- *Creación de una tabla dentro de el JFrame interno de Interfaz_Reservas ya que así podremos autorellenar el formulario de Reservas dependiendo del vehículo seleccionado.*

30/Mayo

- *Unificación de clases.* Se han eliminado las clases que solo se utilizaban para crear JFrames: Interfaz_ListarVehiculos e Interfaz_ListarClientes y se han cambiado por métodos.
- *Detectamos un bug en el registro de vehículos.* Al intentar registrar un vehículo dejando campos en blanco salta una excepcion.

  RESUELTO: La verificación de los campos obligatorios para realizar el insert se hacían a través de un método de instancia de vehículo. Siempre daba false el registro porqué esta verificación se hacía antes de setear los valores de vehículo.

Se ha creado un método en Interfaz_RegistroVehiculos donde se verifican si los campos están rellenados o no en el formulario y en función de eso seteamos los valores de la correspondiente instancia.

- *Detectamos un bug al hacer click en "cancelar" del JOptionPane al modificar el precio de un vehículo.*

  RESUELTO: Ahora controlamos el valor del input y solo si no está vacío procedemos a ejecutar el siguiente código.

- *Clase Reserva.* Creamos la clase Reserva, finalmente no creamos todos los atributos planteados en el diagrama de clases y lo dejamos como sigue:

```
private static final AtomicInteger RESERVA = new AtomicInteger(99);
private int numReserva;
private String matricula;
private final Calendar FECHASOLICITUD = Calendar.getInstance();
private Calendar fechaHoraRecogida;
private Calendar fechaHoraDevolucion;
private String observaciones;
```

31/Mayo

- *Mostrar especificaciones de cada vehículo.* Al buscar un vehículo por número de matrícula se imprimen los datos obtenidos de la tabla vehiculos. Para incluir las especificaciones propias de cada vehículo necesitamos identificar qué tipo es, para ello hemos creado una función en plpgsql: tipoVehiculoMatricula.

- *Funcionalidades Reserva*. Se han creado lo métodos para cancelar una reserva y listar reserva. La query para listar las reservas es muy larga y los títulos de las columnas en la JTable se cortan, para solucionarlo hemos creado un método en la clase reserva únicamente para crear un frame a medida para esta JTable.

- *RCExcepcion.* Se crea la clase RCException para controlar excepciones propias de la aplicación.

1/Junio

- *Control de excepciones.* Se desarrollan métodos para verificar que el número de teléfono no contenga caracteres y que los datos del nombre no contenga números.

  Obtengo una excepción a veces por la longitud del número de teléfono introducido, es por el tipo de dato int.

2/Junio

- *Cancelar Reservas - Listar reservas desde entorno cliente.* Con el objetivo de que un cliente solo pueda listar y cancelar sus propias reservas se ha tenido que modifica la consulta a BBDD y el método para que reciba por parámetro el nif del cliente. Además, se ha movido el método de cancelación a la clase Reserva, ya que inicialmente se trataba desde la interfaz de administrador.

- Creación de una clase nueva para en un futuro imprimir el componente que pasemos por parámetro, probablemente lo usemos en reservas.

3/Junio

- *Clase General.* Se ha creado la clase final General para agrupar todo los métodos generales que se desde varias clases de la aplicación.

- *Creación de diagrama de clases desde NetBeans.* Nos hemos instalado el plugin EasyUML en NetBeans para generar diagramas de clases desde el IDE. Hemos generado el diagrama de clases de este proyecto.

4/Junio

- *Conexión clase Reserva con clase Interfaz_Reserva.* Se han creado los métodos en Reserva para insertar datos en las tablas de Reservas mediante una transacción.

  PROBLEMA: Las excepciones que lanza el método registrarReserva no aparecen en el front.

5/Junio

- *Clase Interfaz_Reserva.* Se ha programado un método para que la aplicación vuelque de forma automática los datos personales del cliente que se ha logueado en la reserva.

- Se ha resuelto el problema detectado el 4/Junio, ya capturamos correctamente la excepción y se muestra por pantalla mediante un JOptionPane.

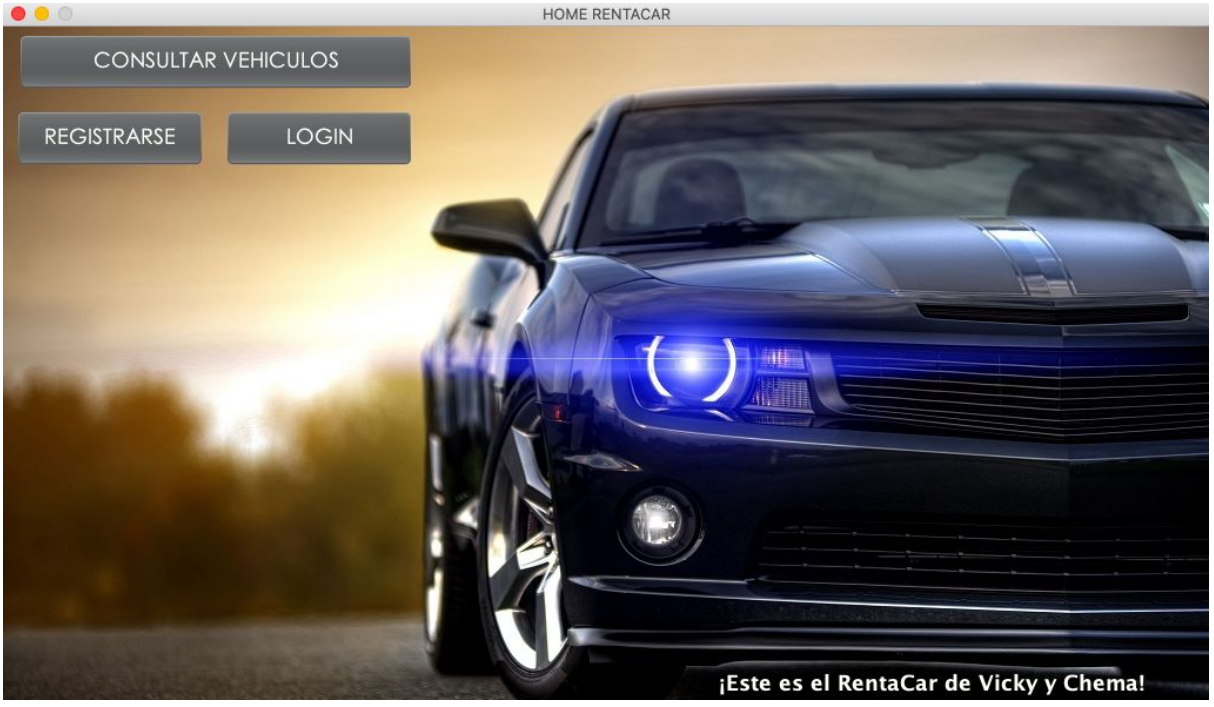
FASE X: Desarrollo e implementación de mejoras.

- La clase Impresora.java ha sido una clase que hemos implementado para imprimir el componente que esté en activo en ese momento y lo utilizamos en creación de nuevas reservas.
- Habíamos creado un correo electrónico y una clase fuera del proyecto para mandar un correo electrónico cada vez que se solicitara una reserva, después de haber configurado todo, se mostrarían mensajes automatizados como el siguiente:

Por falta de tiempo no hemos podido implementarlo con los métodos que hubiéramos deseado, ya que teníamos en mente incluir un documento o albarán mediante las técnicas aprendidas en el ciclo.

FASE XI: Testeos end-to-end.

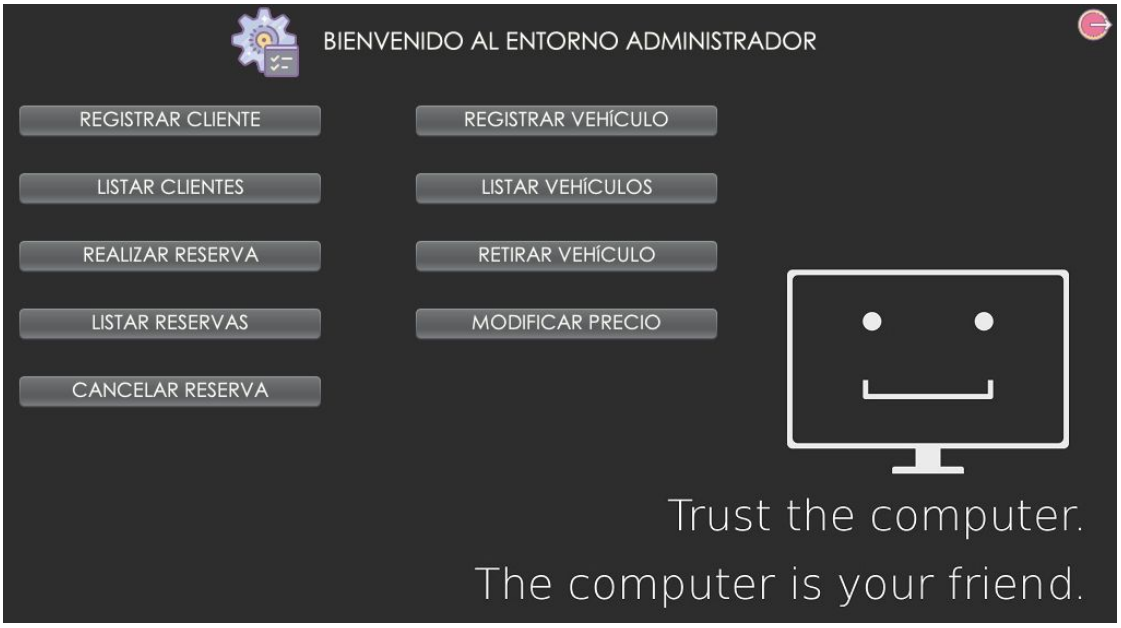
Pantalla de LOGIN: Al iniciar el programa, se despliegue el porlet de inicio para acceder a la intranet.



Desde esta pantalla, es posible realizar una visualización de los vehículos disponibles (CONSULTAR VEHICULO), así como poder buscar un vehiculo por matricula:

También, se pueden crear nuevos usuarios desde REGISTRARSE. Se hace un control mediante los getters i setters de la longitud de la contraseña, que los datos del nombre no contengan valores numéricos mediante el uso de expresiones regulares, que el teléfono no contenga letras y que el email sea válido, es decir, que incluya un '@';

Entorno administrador:
Usuario: admin
Contraseña: admin123

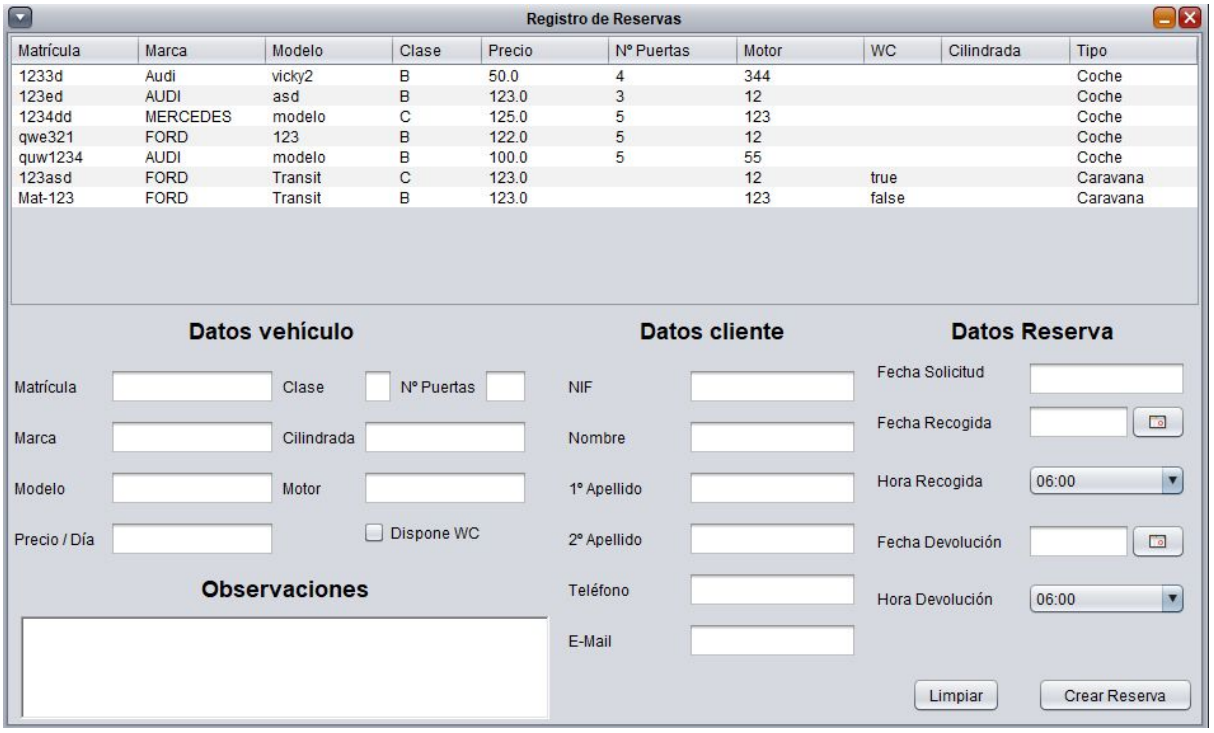


Desde el entorno administrador, se puedes registrar nuevos clientes/usuarios, listar los clientes existentes en el sistema, realizar reservas, listar reservas, cancelar reservas, registrar nuevos vehículos, listar los vehículos disponibles, retirar o dar de baja vehiculos, modificar el precio de los vehículos.

LISTAR CLIENTES: JTable con el listado de clientes creados en la aplicación con la posibilidad de realizar una búsqueda por NIF:

nif	nombre	apellido1	apellido2	telefono	email
444	Chema	Hernandez	Bruno	333	222@mail.com
5645646	asdasdf	asd	asfasf	14	cacca@mail.com
22211223d	pili	pili	carrera	545556	dddd@dddd.com
estoes45d	prueba	de	vicky	234	email@vi.com
pruebofin	porfi	termina	ya	234	mi@turno.com
11111111J	Jose Maria	Hernandez	Bruno	222222	mail@mail.com
123456789J	Test	Testing	Testing	332332332	test@test.com
1111FFF	rosalia	rosalia	de castro	222	222@mail.com

REALIZAR RESERVA: JInternalFrame con el listado de vehículos disponibles, cuándo uno de éstos es seleccionado, se introducen los datos automáticamente en la sección de “Datos vehículo”, en la zona de “Datos Cliente”, al tratarse de un usuario administrador está vacío y nos permitirá introducir los campos de cualquier cliente, por último se asigna una fecha y hora de recogida, fecha y hora para la devolución gracias al “jCalendar” y finalmente crearemos la reserva en “Crear Reserva”.



LISTAR RESERVAS: JTable con el listado de reservas existentes en la aplicación con la posibilidad de realizar una búsqueda por número de reserva:

CANCELAR RESERVA: Desde esta opción aparece el JTable de LISTAR RESERVAS, y mediante un JOptionPane se puede indicar el número de reserva a cancelar. Solo se podrá cancelar si se trata de una reserva NO anterior al día de HOY.

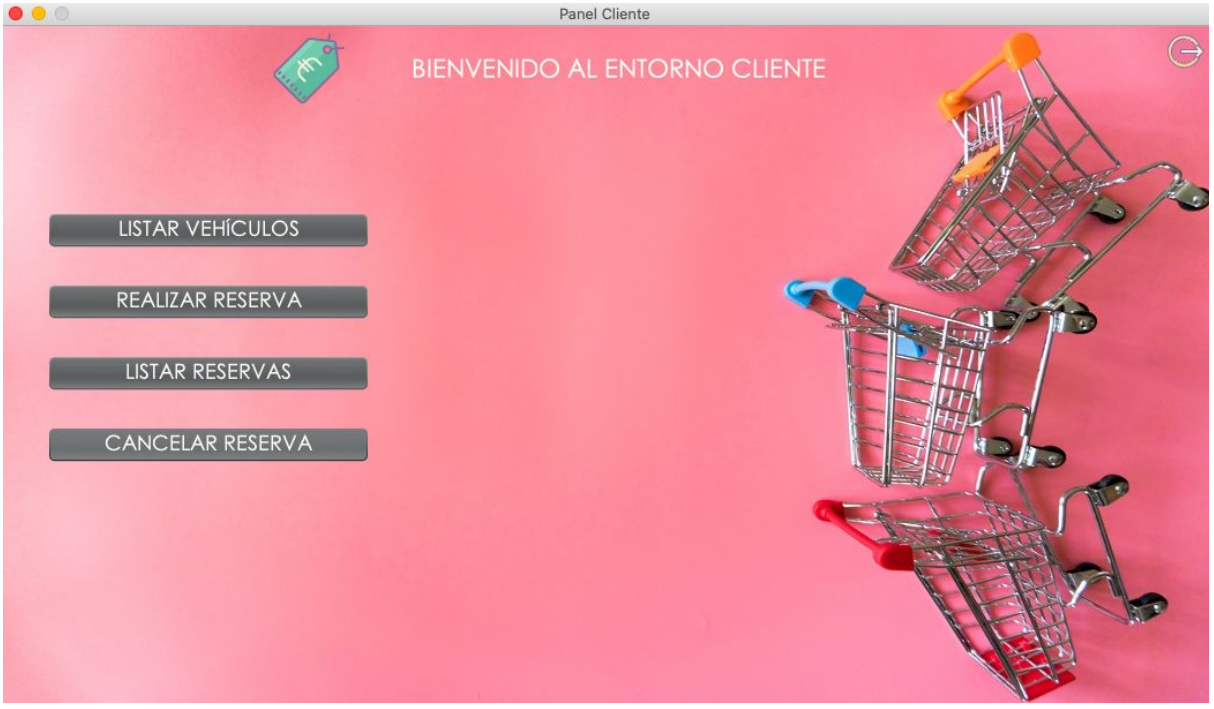
REGISTRO VEHICULOS: Desde este 'formulario' se pueden registrar nuevos vehículos, como dependiendo del tipo de vehículo se tiene que hacer un insert en tabla vehículos + en su tabla específica, se realiza todo mediante una transacción. Además, se realiza una verificación de que todos los campos estén completados y se capturan todas las excepciones, mostrando el mensaje en rojo de la excepción en un JLabel. En el caso de que el vehículo se inserte correctamente en la BBDD, se mostrará un mensaje en verde de vehículo registrado con éxito.

RETIRAR VEHÍCULO: Desde esta opción se pueden retirar vehículos desde esta opción. Los vehículos NO se eliminan de la base de datos, ya que si han sido seleccionados en reservas obtendremos una SQLExcepcion, sino que la columna 'retirado' cambia a true y el vehículo ya no aparecerá en los listados para ser seleccionado en futuras reservas.

MODIFICAR VEHICULO: Desde esta opción podemos modificar el precio de un vehiculo que este activo, es decir retirado = false.

Entorno cliente:

Usuario: rosi
Contraseña: rosalia123



1. **LISTAR VEHICULOS:** Lista el JFrame explicado anteriormente con el total de vehículos disponibles para reservar.

2. **REALIZAR RESERVA:** A diferencia del entorno administrador, cuando se conecta como un usuario cliente los datos personales vuelcan de forma automática en los datos de cliente de la reserva:

3. **LISTAR RESERVAS:** Al igual que desde el administrador, desde esta opción se muestra un JTable con las reservas existentes PERO solo del usuario que ha logueado, con posibilidad de buscar por número de reserva.

4. **CANCELAR RESERVA:** Idem que en el caso anterior, el usuario solo puede cancelar sus propias reservas.

FASE XII: Vídeo de presentación del proyecto.

Hemos subido el video a YouTube como privado. Hemos abierto permisos para que puedas acceder a él con la cuenta del instituto: rgion@cifpfbmoll.eu

Vídeo del proyecto

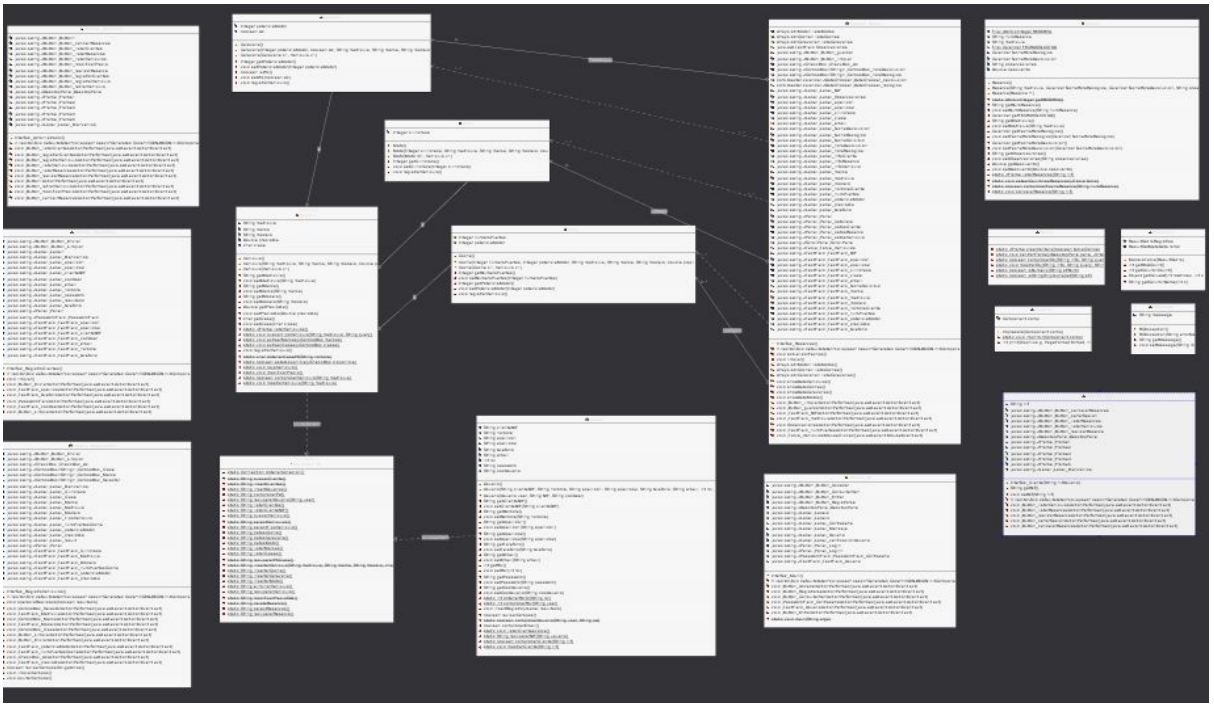
FASE XIII: Conclusiones finales.

A continuación se detallan las horas reales que ha conllevado la realización de este proyecto:

FORMACIÓN: 45 horas

DESARROLLO APLICACIÓN: 115 horas

Se han aplicado cambios en el esquema de base de datos y en el diagrama de clases, quedando de la siguiente manera:



Hemos hecho uso de easyUML, y podemos comprobar que el diagrama ha cambiado drásticamente al original, aquí tenemos una imágen que se puede acceder a ella desde NetBeans y Github.

Uno de los principales retos con los que nos hemos encontrado ha sido trabajar con GIT. A pesar de que en entornos de desarrollo se hayan realizado prácticas individuales y con compañeros, para la realización de este proyecto hemos tenido que establecer un flujo de trabajo con GIT para no pisarnos y estar coordinados. Trabajar juntos y comunicándonos ha sido el factor clave para poder realizar esta práctica.

Finalmente hemos dedicado bastantes más horas de las previstas, hemos tenido que realizar muchísima autoformación por nuestra cuenta que nos ha retrasado y nos ha dejado sin tiempo para implementar todas las mejoras que nos hubiera gustado.