Curso:	1º CFS DAM	Fecha límite:	24/05/23 10:00 (40 horas)
Trimestre:	3	Modalidad:	Equipo de 4* alumnos

ESCUELAS PROFESIONALES SAGRADA FAMILIA
SAFA

PROYECTO

APLICACIÓN GESTIÓN DE AEROPUERTO
(EN LENGUAJE JAVA Y MARIADB)

RESULTADOS Y COMPETENCIAS PERSEGUIDOS

En consonancia a los resultados de aprendizaje y competencias que el alumno debe completar cursando el módulo de **Programación**, este trabajo contribuye a conseguir los siguientes:

R4) Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.
R5) Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases.
R6) Escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos.
R7) Desarrolla programas aplicando características avanzadas de los lenguajes orientados a objetos y del entorno de programación.
R8) Utiliza bases de datos orientadas a objetos y NoSQL, analizando sus características y aplicando técnicas para mantener la persistencia de la información.
R9) Gestiona información almacenada en bases de datos relacionales manteniendo la integridad y consistencia de los datos.

El módulo de **Entornos de Desarrollo** por su parte pretende que los resultados de aprendizaje y competencias profesionales a alcanzar sean los siguientes:

R2) Evalúa entornos integrados de desarrollo analizando sus características para editar código fuente y generar ejecutables.
R3) Verifica el funcionamiento de programas diseñando y realizando pruebas.
R4) Optimiza código empleando las herramientas disponibles en el entorno de desarrollo.
R5) Genera diagramas de clases valorando su importancia en el desarrollo de aplicaciones y empleando las herramientas disponibles en el entorno.

El módulo de **Bases de Datos** por su parte pretende que los resultados de aprendizaje y competencias profesionales a alcanzar sean los siguientes:

R6) Diseña modelos relacionales normalizados interpretando diagramas entidad/relación.
R7) Gestiona la información almacenada en bases de datos objeto-relacionales, evaluando y utilizando las posibilidades que proporciona el sistema gestor.

METODOLOGÍA DE REVISIÓN

Dado que desde el módulo de Entornos de Desarrollo se ha formado a los alumnos en el manejo de la aplicación de control de versiones **Git** además de haber creado su propio espacio **GitHub**, la metodología de revisión comenzará por monitorizar el recorrido que la solución proporcionada por el alumno ha tenido desde su primera edición.

Una vez comprobada la ubicación totalmente funcional de la aplicación en el espacio **Github** elegido por el equipo, se revisará si cumple con los cánones de la programación orientada a objetos para, seguidamente, observar que no genera ningún error de compilación ni excepción no controlada.

Hecho esto, se procederá a la ejecución en ordenador de la solución propuesta por el alumno para determinar si cumple con los requerimientos funcionales y no funcionales expuestos en el enunciado y si dicha solución podía presentarse de manera más profesional según el momento de aprendizaje del alumno.

DESCRIPCIÓN DEL PROYECTO

Este es un proyecto que calibrará el grado de familiaridad y destreza que el alumno va teniendo en Java a la hora de aplicar los conceptos y habilidades adquiridos y por adquirir a lo largo de este trimestre en los módulos de **Programación, Bases de Datos y Entornos de Desarrollo.**

Se trata de implementar la parte más básica de una aplicación que contemple los aspectos que concurren en la gestión de un aeropuerto. Para empezar, se presentará una pantalla inicial (bajo formato Swing) que realizará la tarea de autentificar al usuario que pretende utilizar la aplicación a partir de la introducción de usuario y contraseña mediante teclado. La manera de contrastar que el usuario está habilitado para operar con la aplicación pasa por comprobar que dicho usuario y contraseña tecleados se encuentran en una tabla (Usuarios) de la base de datos MariaDB denominada **Aeropuerto**. Dicha tabla se rellena automáticamente al comenzar el programa cogiendo los datos del fichero **UsuariosHoy.xml** (suministrado).

Si el usuario y contraseña se encuentran en dicha tabla, se superpondrá en pantalla un cuadro de diálogo dando la bienvenida a ese usuario e indicando la fecha y hora (cogidas del sistema, claro) en la que se ha conectado. El pulsado del botón **Aceptar** del cuadro de diálogo llevará al usuario a la pantalla principal de la aplicación.

Caso de no estar en la tabla, se mostrará otro cuadro de diálogo informando de que ese usuario y/o contraseña no son válidos, no permitiéndole el acceso a la aplicación y volviendo a presentar la pantalla en la que se teclean usuario y contraseña.

Una vez el usuario habilitado para operar con la aplicación, se observa que la misma implementa unos procesos básicos de gestión de vuelos, tripulación y pasajeros. De ahí que se presente una pantalla central conformada por cinco pestañas que parcelarán la aplicación en sendas áreas: **Aviones, Tripulantes**, **Pasajeros, Vuelos y Asignación**. Cada pestaña mostrará en su área de trabajo un letrero cabecera en la parte superior que estará centrado horizontalmente:

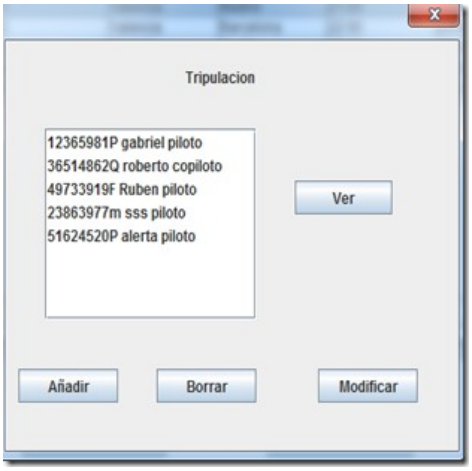
a) la pestaña **Aviones** estará activada por defecto al iniciarse el programa. El área de la ventana pasa a mostrar una interfaz que mostrará una tabla (Aviones, [idAvion, matricula, modelo, numAsientos, estado]), con los datos de las aeronaves que conforman el parque aeronáutico actualmente operativo.

Inicialmente los datos de los aviones estarán cargados en un fichero JSON denominado **Aviones.json** (suministrado) que deberán ser transferidos a la tabla nada más ejecutarse el programa. Una vez en la tabla podrán ser accedidos para su representación en pantalla en forma tabular. El campo estado puede llevar dentro un 1 o un 0 dependiendo de si la aeronave se encuentra operativa o no.

AVIONES OPERATIVOS

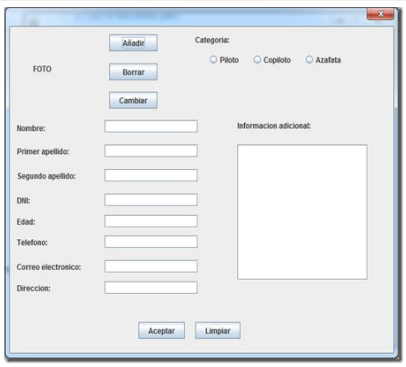
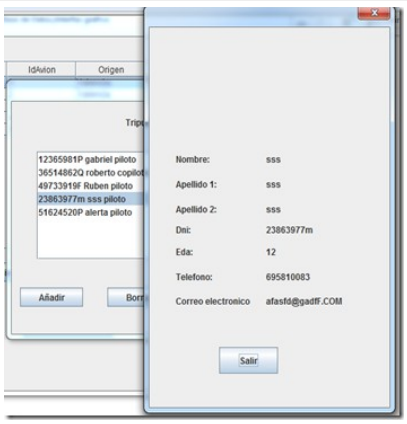
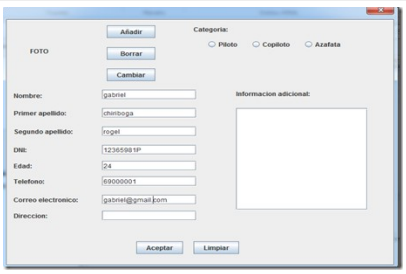
idAvion	Matrícula	Modelo	Asientos
1	EC-CMZ	Airbus A-320	200
2	EC-LBR	Boeing 737	220
3	EC-TPL	Airbus A-340	300
4	EC-KEG	Boeing 727	160

b) la pestaña **Tripulantes** mostrará una interfaz de gestión de datos (insertar, modificar y borrar) que mantendrá una tabla (Miembros, [idTripulacion, nombre, apellido1, apellido2, edad, telefono, ecorreo, direccion, categoria]), con los datos de las personas que pueden formar parte de las tripulaciones de los vuelos.

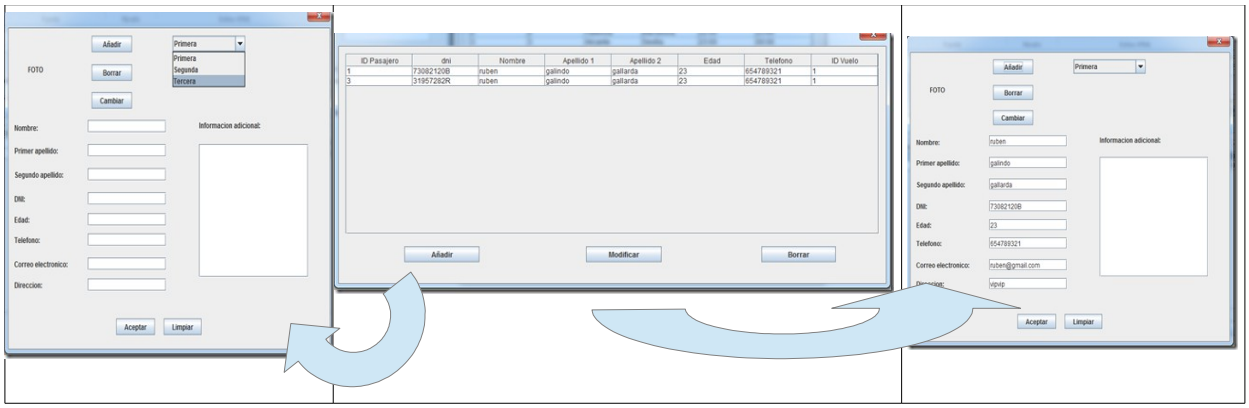


La justificación del botón **Ver** radica en que una vez elegido uno de los miembros de la lista, se deben presentar todos los datos del mismo.

El pulsado del resto de botones que se muestran en la interfaz, llevarán a la aparición de las pantallas que se visualizan justo debajo.

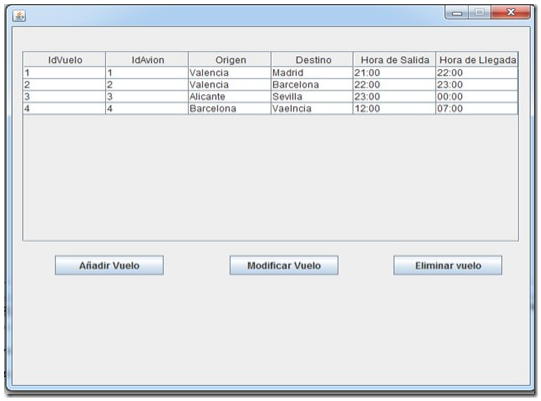
Opción: Añadir tripulante	Opción: Ver tripulante	Opción: Modificar tripulante
		

c) la pestaña **Pasajeros** mostrará una interfaz de gestión de datos (insertar, modificar y borrar) que mantendrá una tabla (Pasajeros, [idPasajero, nombre, apellido1, apellido2, edad, telefono, ecorreo, direccion, foto]), con los datos de las personas que pueden ser clientes de esos vuelos.

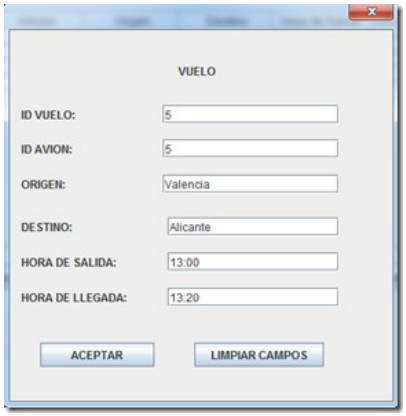
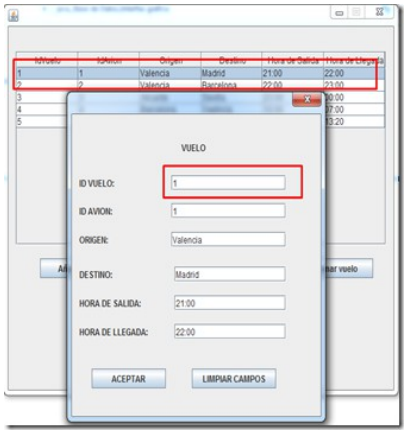
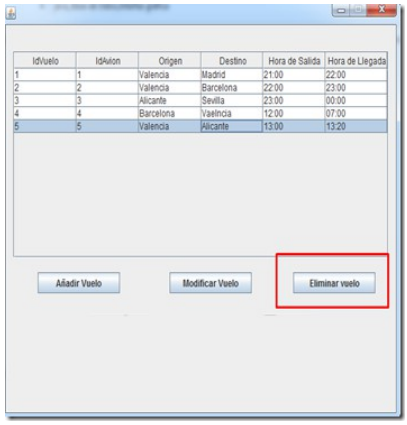


El pulsado de los botones que se muestran en la interfaz, llevarán a la aparición de las pantallas que se visualizan a su lado izquierdo y derecho.

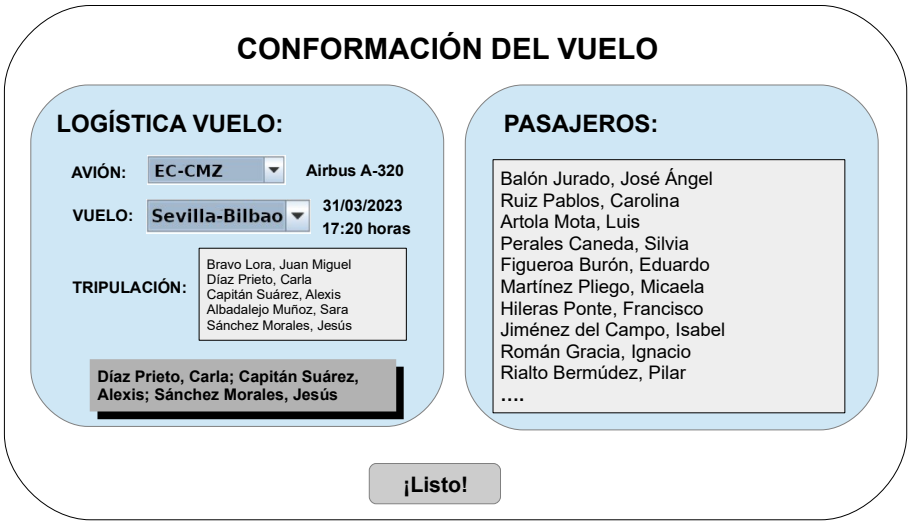
d) la pestaña **Vuelos** mostrará una interfaz de gestión de datos (insertar, modificar y borrar) que mantendrá una tabla (Vuelos, [idVuelo, idAvion, idTrayecto, fecha, horasalida, horallegada]), con los datos de los vuelos que se pueden operar.



Como bien puede observarse en la tabla **Vuelos** existe un campo denominado idTrayecto pero en la interfaz se muestran las columnas Origen y Destino. De ahí se deriva la existencia del fichero binario **Trayectos.dat** (suministrado) que tendrá tres campos: idTrayecto, Origen, Destino. Éste viene ya relleno y sólo habrá que volcarlo en la tabla **Trayectos**.

Opción: Añadir vuelo	Opción: Modificar vuelo	Opción: Eliminar vuelo
		

e) la pestaña **Asignación** permitirá mostrar los pasajeros con plaza en un vuelo así como el avión, vuelo y tripulantes que operarán el mismo.



Nota: *En las capturas se visualiza simplemente un esbozo de las interfaces a realizar tanto en Swing como en JavaFX, buscando el equipo el mejor diseño para presentar estas pantallas.*

Los datos que aparecen en las listas desplegables deben obtenerse de las respectivas tablas de la base de datos. Una vez realizada la elección del avión a través de la matrícula, se mostrará a su derecha el modelo de éste. Se realizará lo propio con el vuelo, donde una vez seleccionado, se acompañará de su fecha y hora de salida.

Para indicar la tripulación, se seleccionarán de la lista de posibles tripulantes hasta un máximo de cinco. A medida que se vayan escogiendo se irá formando debajo una lista de aquellos que se han elegido.

El pulsado del botón **¡Listo!** hará visible el mensaje de diálogo que mostrará un mensaje del estilo a "¡Vuelo correctamente configurado! Puede ocurrir que este avión ya haya sido asignado, lo cual deberá ser informado al usuario y ofrecerle la posibilidad de comenzar de nuevo el proceso.

La información seleccionada en cada desplegable deberá quedar almacenada en forma de registro en una tabla de nombre **Asignados** en la base de datos. El registro se complementará con un último campo que guardará la fecha y hora del momento de la configuración del vuelo.

Toda operación realizada en la aplicación (transferencia de fichero a tabla, nuevo tripulante, modificación pasajero, borrar vuelo, asignación, etc.) quedará reflejada mediante un mensaje en un fichero de texto que se llamará **Logomens.txt**. Ni que decir tiene que el fin de la ejecución del programa está vinculado al cierre de la ventana de la interfaz.

Queda claro entonces que todos los datos quedarán finalmente consignados en sendas tablas de una base de datos soportada por MariaDB denominada **Proy3TEx** (siendo x el nº del equipo).

Además de haber implementado la solución con tecnología Swing, se propone una segunda versión de la misma en JavaFX. Y ya puestos en la senda de adquirir estrellas de profesionalidad, se adoptará una solución basada en el patrón de software denominado Modelo/Vista/Controlador. El tránsito por esa senda llevará al equipo a tener que elegir entre crear la Vista con o sin fichero de interfaz FXML. En el caso de decantarse por hacerlo con tecnología FXML se recomienda la utilización de la herramienta "Scene Builder".

Dado que un trabajo profesional, requiere documentar correctamente la aplicación, se empezará confeccionando el diagrama UML de clases de la misma. Para ello se deben diseñar las **clases** (con sus atributos y visibilidad) y **relaciones** (herencia, asociación, roles, navegabilidad, cardinalidades, todo-parte) del código fuente resultante, todo ello con el software **Visual Paradigm**.

Independientemente del IDE utilizado, deben quedar reflejadas en la documentación las depuraciones realizadas, así como las refactorizaciones (de obligado cumplimiento para cualquier programador que se precie) consistentes en realizar pequeñas transformaciones en el código de un programa, para **mejorar la estructura interna** sin que cambie el comportamiento ni funcionalidad del mismo. Con ello se consigue un código **limpio** minimizando la posibilidad de introducirle errores, de ahí la importancia de documentar el procedimiento de depuración seguido y refactorizaciones realizadas.

Tan importante es lo anterior como llevar a cabo la documentación de código con comentarios que expliquen exhaustivamente su funcionamiento, de forma que cualquier persona que los lea pueda entender la finalidad del código. Debe tratar de explicar todo lo que no resulta evidente. **NO** repetir qué hace el código, sino **POR QUÉ** lo hace. Todos esos comentarios deberán reunirse en una documentación generada con la aplicación **JavaDoc**.

Las mejoras introducidas por el equipo en el diseño y funcionalidad de esta aplicación serán valoradas.

DOCUMENTOS Y ENLACES DE PARTIDA

Los siguientes documentos pueden ayudar a la implementación de algunos de los aspectos requeridos y se encuentran disponibles en el aula virtual:

Puesta a punto del IJ para JavaFX.pdf	**Documentación sobre JavaFX.pdf**

Los siguientes enlaces pueden ayudar a la implementación de algunos de los aspectos requeridos:

SQL Table → JTable: **https://www.youtube.com/watch?v=2d4i6BXQPFA**

SQL Table → JTable: **https://www.youtube.com/watch?v=P0opfx23Czw**

Múltiples pestañas: **https://youtu.be/Gidsbny-wu8**

JavaFX Design: **https://www.youtube.com/watch?v=ChVV36xWCTg**

JavaFX Setting UI Design: **https://www.youtube.com/watch?v=gJYXctDSIl8**

PRESENTACIÓN

Además de las revisiones que los profesores de los tres módulos realizarán sobre las partes del proyecto de su ámbito, la presentación del proyecto se realizará en el aula virtual y vendrá determinada por los documentos:

Parte Swing

En documento PDF y siguiendo este orden (en buzón de cada módulo)**:**

a) Modelo relacional de la Base de Datos de la aplicación
b) Script SQL de creación de las tablas
c) Procedimiento de depuración seguido y refactorizaciones realizadas
d) Mejoras de presentación y funcionalidad realizadas a la aplicación

En formato HTML

a) Resultado de la documentación realizada del código con JavaDoc

Los códigos resultantes de la solución (Swing):

a) Ficheros **.java** generados
b) Fichero **.jar** ejecutable generado a partir de todo lo anterior

Parte JavaFX

En documento PDF y siguiendo este orden (en buzón de cada módulo)**:**

a) Diagrama UML de las clases de la aplicación (diferenciando MVC)
b) Procedimiento de depuración seguido y refactorizaciones realizadas
c) Mejoras de presentación realizadas a la aplicación
d) Mejoras de funcionalidad realizadas a la aplicación

Los códigos resultantes de la solución (JavaFX):

a) Ficheros **.java** generados
b) Fichero **.jar** ejecutable generado a partir de todo lo anterior

EVALUACIÓN

Por tanto, una vez entregado el contrato de equipo con las cláusulas y planificación requeridas, la calificación del caso vendrá determinada según el cuadro siguiente:

Las notas de la solución presentada en Java Swing - (Entrega-Revisión 17 Mayo) - 50%
Solución mediante interfaz gráfica Swing (45%) [100%-P]
Solución persistencia: CRUD contra las tablas, ficheros de texto, binarios, JSON y XML (15%) [100%-P]
Solución con patrón Modelo/Vista/Controlador (9%) [100%-P]
Modelo relacional de la base de datos (5%) [100%-P]
Script SQL de creación de las tablas (4%) [100%-P]
Depuración y refactorizaciones realizadas (9%) [50%-P / 50%-A]
Mejoras en la presentación del GUI y funcionalidad de la aplicación (4%) [100%-P]
Presentación de la solución en JAR ejecutable (2%) [100%-P]
Documentación de la aplicación generada con Javadoc (7%) [50%-P / 50%-A]
Funcionamiento de los todos códigos sin errores de compilación **[Excluyente]**
Modularidad mínima exigida en las soluciones **[Excluyente]**

Las notas de la solución presentada en JavaFX - (Entrega-Revisión 24 Mayo) - 50%
Solución mediante tecnología JavaFX (43%) [100%-P]
Solución persistencia: CRUD contra las tablas, ficheros y XML (15%) [100%-P]
Solución JavaFX con patrón Modelo/Vista/Controlador (10%) [100%-P]
Diagramas UML de la aplicación (15%) [50%-P / 50%-A]
Depuración y refactorizaciones realizadas (10%) [50%-P / 50%-A]
Mejoras en la presentación del GUI (5%) [100%-P]
Presentación de la solución en JAR ejecutable (2%) [100%-P]
Funcionamiento de los todos códigos sin errores de compilación **[Excluyente]**
Modularidad mínima exigida en las soluciones **[Excluyente]**