



Programa de gestión de
una Base Aérea Militar

AirBase

Parte 3

Ficheros y Log

Constantino Lapaz Giménez

Juan Mauricio González Arenas

Rafael Antonio Vázquez Flores

DAM Sector

Índice

Punto de partida	1
Trello	2
Estructura del Proyecto	3
Gestión Piloto	3
Modificaciones	5
Para el resto de las clases	6
Gestión Ruta	6
Gestión Incidencias	8
Fichero .log	9
Resumen	11
Nota final	11

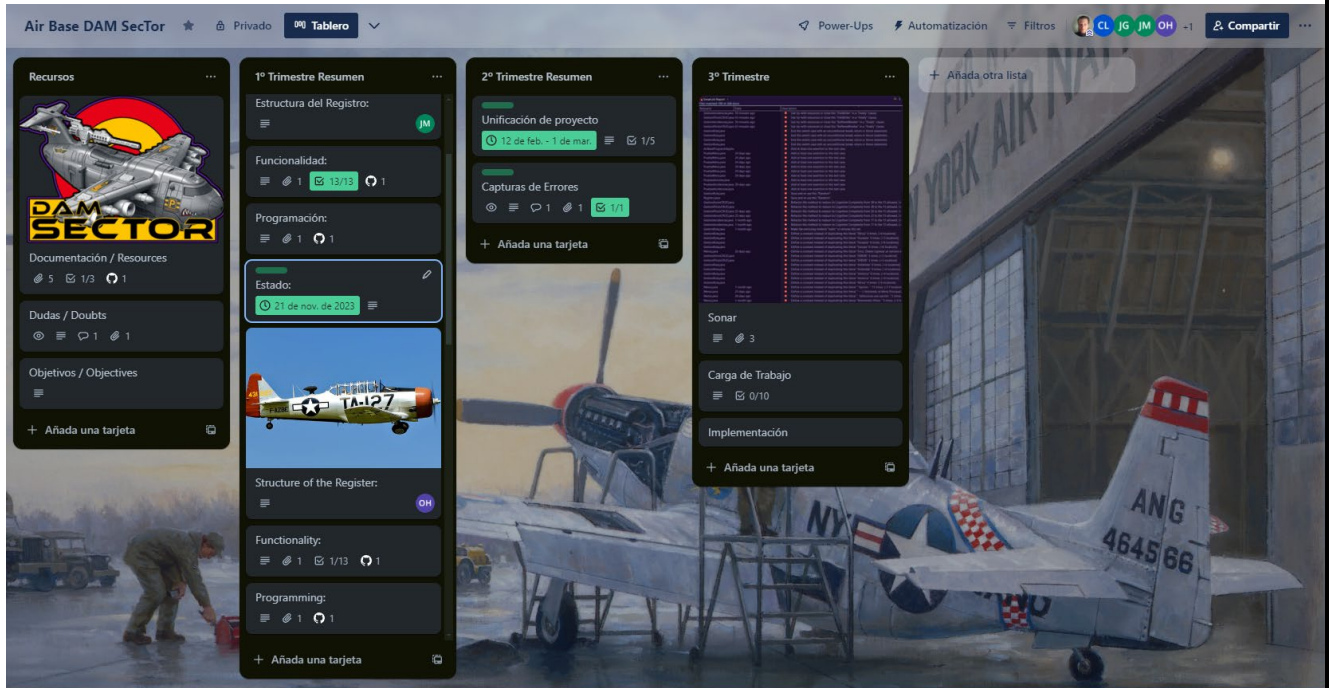
Punto de partida

La tarea consiste en añadir ficheros y uso de cadenas de texto al proyecto:

- Uso de ficheros para leer datos: Los datos deben estar estructurados en formato csv, esto es, cada campo separado de otro campo por ";" y en cada línea un registro. Se valorará positivamente el uso de distintas funciones de cadenas de textos (split, indexOf, contains, substring, sprit, toLowerCase, toUpperCase, etc.)
- Uso de ficheros para escribir un log de toda vuestra aplicación: Cada opción de menú, cada método, cada excepción controlada, deben escribir un registro en un archivo .log. Para ello el programa cuando se ejecuta debe crear una carpeta logs dentro de vuestro proyecto, borrando la carpeta en caso de que exista (carpeta y archivos). Una vez creada la carpeta se creará un archivo para los logs con extensión .log en el que se irán incluyendo los registros de log. Se valorará positivamente la creación de varios archivos por día, por tipo de registro de log, cuando se supere un tamaño, etc...
- Se debe crear un archivo de datos para cada uno de los módulos realizados en la aplicación de forma que toda la información de la aplicación esté en los archivos y no directamente en el código.

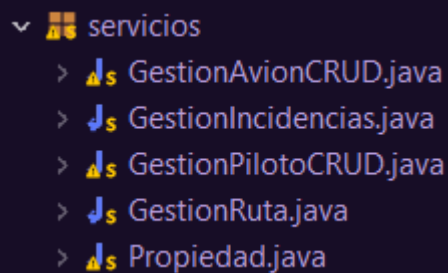
Trello

Hemos actualizado Trello para esta nueva entrega realizando así las modificaciones pertinentes para este trimestre.



Estructura del Proyecto

Generamos diferentes ficheros en cada uno de los métodos concretos, en este caso son:



```
servicios
├── GestionAvionCRUD.java
├── GestionIncidencias.java
├── GestionPilotoCRUD.java
├── GestionRuta.java
└── Propiedad.java
```

Gestión Piloto

```
public ArrayList <Piloto> pilotos;
private Random random = new Random();
public String rutaArchivo = "ficheros/Pilotos.txt";

public void inicializar(Registro r) {
    pilotos = new ArrayList <Piloto>();
    /*NO ES NECESARIO INICIALIZARLO, LO CREAMOS DESDE 0
    pilotos.add(new Piloto("P001", "Faly", 200, true, true));
    pilotos.add(new Piloto("P002", "Constantino", 200, true, false));
    pilotos.add(new Piloto("P003", "Omar", 200, true, true));
    pilotos.add(new Piloto("P004", "Mauricio", 200, true, false));
    */
    cargarDesdeArchivoCSV();
}
```

Creamos la variable **rutaArchivo** y la declaramos como publica ya que serán varias clases la que accederán a ella. Cada clase modificara la ruta para su conveniencia, pero todas serán el mismo Directorio "ficheros"

```
public void archivoPiloto() {
    File fichero = new File(rutaArchivo);
    try {
        if(fichero.createNewFile()) {
            System.out.println("Archivo creado correctamente");
        }else {
            System.out.println("");
        }
    }catch (IOException e) {
        escribirError("ERROR en crear archivo");
    }
}
```

Generamos el archivo mediante esta función, capturamos una excepción para luego (más tarde lo detallare) integrar ese error en un archivo log.

```
public ArrayList<Piloto> leerPilotos(String rutaArchivo) {
    ArrayList<Piloto> pilotos = new ArrayList<>();
    try {
        BufferedReader reader = new BufferedReader(new FileReader(rutaArchivo));
        String linea;
        // Leer encabezados (ignorar primera línea)
        reader.readLine();
        // Leer datos de los pilotos
        while ((linea = reader.readLine()) != null) {
            String[] campos = linea.split(",");
            String codigo = campos[0];
            String nombre = campos[1];
            int horasVuelo = Integer.parseInt(campos[2]);
            boolean operativo = Boolean.parseBoolean(campos[3]);
            Piloto piloto = new Piloto(codigo, nombre, horasVuelo, operativo, false);
            pilotos.add(piloto);
            for(int i =0;i< campos.length;i++) {
                System.out.print(campos[i]+",");
            }
            System.out.println();
        }
        reader.close();
        System.out.println("\nLos datos de los pilotos se han leído desde el archivo correctamente.");
    } catch (IOException e) {
        escribirError("Error al leer los datos de los pilotos desde el archivo :"+ e.getMessage());
    }
    return pilotos;
}
```

Escribimos en el ArrayList con el formato del fichero indicando que se salte la primera línea que usaremos como cabecera.

Después indicamos que cada columna está separada por “,” y a que pertenece cada **campo**

Para terminar, mostramos lo escrito e indicamos que lo hemos leído correctamente, cualquier error lo capturamos con el catch y lo volcamos como mensaje en el archivo log.

```
private void actualizarArchivoCSV() {

    try (FileWriter writer = new FileWriter(rutaArchivo)) {
        // Escribir los encabezados en el archivo si es necesario
        System.out.println("Escrito Correctamente...");
        writer.append("Codigo,Nombre,HorasVuelo,Operativo\n");
        // Escribir cada piloto en el archivo
        for (Piloto piloto : pilotos) {
            writer.append(piloto.getCodigo() + "," +
                piloto.getNombre() + "," +
                piloto.getHorasVuelo() + "," +
                (piloto.getOperativo() ? true : false) + "\n");
        }
    } catch (IOException e) {
        escribirError("Error al actualizar el archivo txt: " + e.getMessage());
    }
}
```

Para Actualizar el archivo, solo tenemos que escribir en los nuevos campos que queremos añadir directamente, para eso generamos en un **for** los GET que tenga nuestra Clase

Nuevamente volvemos a capturar la excepción para el archivo log.

Modificaciones

```
public void verRutas(ArrayList<Ruta> rutas) {

    /*
    for(int i = 0; i < rutas.size(); i++) {
        System.out.println("");
        System.out.println("    " + rutas.get(i).getPaisOrigen() + " - " + rutas.get(i).getPaisDestino() + " ~ " + rutas.get(i).getH());
    }
    */

    leerRutas(rutaArchivo);

}
```

Como podemos ver algunas funciones antiguas, ya no son necesarias por lo que modificamos y actualizarlos para que ahora se haga todo desde el fichero y no desde el **arrayList**

```
public void mostrarRuta() {
    int h = altaRuta(rutas);
    System.out.println("    La duracion de la ruta es: "+ h+" horas\n");

    // escribimos el return en el archivo
    actualizarArchivoCSV();
}
```

Ahora debemos llamar en el resto de funciones a nuestro método **actualizarArchivoCSV()** de esta manera siempre que generemos un dato clave para nuestro fichero, estará agregado correctamente.

Para el resto de las clases

Hemos usado el mismo método y funcionalidad cambiando las partes claves que involucren dichas clases.

Gestión Ruta

```
private void cargarDesdeArchivoCSV() {
    //Esta funcion lo que hace es leer el archivo nada mas se ejecuta el programa.
    //Mientras leemos el archivo se van guardando las ruta en un arrayList hasta el final del archivo
    try {
        BufferedReader reader = new BufferedReader(new FileReader(rutaArchivo));
        String linea;
        //Para saltar la primera linea
        reader.readLine();
        // Leer cada linea del archivo CSV
        while((linea = reader.readLine()) != null) {
            // Dividir la linea en campos usando la coma como delimitador
            String[] campos = linea.split(",");
            // Crear una nueva ruta
            String paisOrigen = campos[0];
            String paisDestino = campos[1];
            int h = Integer.parseInt(campos[2]);
            Ruta ruta = new Ruta(paisOrigen, paisDestino, h);
            rutas.add(ruta);
        }
        reader.close();
    } catch (IOException e) {
        escribirError("No se pudo cargar desde el archivo: " + e.getMessage());
    }
}
```

```
private void actualizarArchivoCSV() {
    try (FileWriter writer = new FileWriter(rutaArchivo)) {
        System.out.println("Escrito Correctamente...");
        writer.append("Pais Origen: Pais Destino : Horas de la ruta\n");

        for (Ruta ruta : rutas) {
            writer.append(ruta.getPaisDestino() + "," +
                ruta.getPaisOrigen() + "," +
                ruta.getH() + "\n");
        }
    } catch (IOException e) {
        escribirError(e.getMessage());
        System.out.println("Error al actualizar el archivo txt: " + e.getMessage());
    }
}
```



```
public ArrayList <Ruta> leerRutas(String rutaArchivo) {
    ArrayList <Ruta> rutas = new ArrayList<>();
    try {
        BufferedReader reader = new BufferedReader(new FileReader(rutaArchivo));
        String linea;

        // Leer encabezados (ignorar primera línea)
        reader.readLine();
        while ((linea = reader.readLine()) != null) {
            String[] campos = linea.split(",");
            String paisOrigen = campos[0];
            String paisDestino = campos[1];
            int horasVuelo = Integer.parseInt(campos[2]);

            Ruta ruta = new Ruta(paisOrigen, paisDestino, horasVuelo);
            rutas.add(ruta);
            // par que salga las rutas que hemos creado
            for(int i =0;i< campos.length;i++) {
                System.out.print(" " + campos[i]);
            }
            System.out.println();
        }

        reader.close();
        System.out.println("\n Las rutas se han leído desde el archivo correctamente.");

    } catch (IOException e) {
        escribirError(" Error al leer las rutas del archivo: " + e.getMessage());
    }
    return rutas;
}
```

Gestión Incidencias

```

public void archivoIncidencia() {
    File fichero = new File(rutaArchivo);
    try {
        if(fichero.createNewFile()) {
            System.out.println("Archivo creado correctamente (Incidencias.txt)");
        }else {
            System.out.println("Error al crear archivo. Ya existe (Incidencias.txt)");
        }
    }catch (IOException e) {
        escribirError(e.getMessage());
        e.printStackTrace();
    }
}

```

```

private void cargarDesdeArchivoCSV() {
    //Esta funcion lo que hace es leer el archivo nada mas se ejecuta el programa.
    //Mientras leemos el archivo se van guardando los aviones en un arrayList hasta el final del archivo
    try {
        BufferedReader reader = new BufferedReader(new FileReader(rutaArchivo));
        String linea;
        //Para saltar la primera linea
        reader.readLine();
        // Leer cada linea del archivo CSV
        while((linea = reader.readLine()) != null) {
            String[] campos = linea.split(",");

            String codigoincidencia = campos[0];
            String codigo = campos[1];
            String descripcion = campos[2];
            String operativo = campos[3];
            Incidencias incidencia2 = new Incidencias(codigoincidencia, codigo, descripcion, operativo);
            incidencia.add(incidencia2);
        }
        reader.close();
    } catch (IOException e) {
        escribirError(e.getMessage());
        System.out.println("No se pudo cargar desde el archivo: " + e.getMessage());
    }
}

```

```

public ArrayList<Incidencias> leerIncidencias(String rutaArchivo) {
    ArrayList<Incidencias> incidencia2 = new ArrayList<>();
    try {
        BufferedReader reader = new BufferedReader(new FileReader(rutaArchivo));
        String linea;

        reader.readLine();

        while ((linea = reader.readLine()) != null) {
            String[] campos = linea.split(",");
            String codigoincidencia = campos[0];
            String codigo = campos[1];
            String descripcion = campos[2];
            String operativo = campos[3];
            Incidencias incidencias = new Incidencias(codigoincidencia, codigo, descripcion, operativo);
            incidencia.add(incidencias);
            for(int i =0;i< campos.length;i++) {
                System.out.print(campos[i]+" ,");
            }
            System.out.println();
        }
        reader.close();
        System.out.println("\nLos datos de los pilotos se han leído desde el archivo correctamente.");
    } catch (IOException e) {
        escribirError(e.getMessage());
        System.out.println("Error al leer los datos de los pilotos desde el archivo : " + e.getMessage());
    }
    return incidencia2;
}

```

```
private void actualizarArchivoCSV() {

    try (FileWriter writer = new FileWriter(rutaArchivo)) {

        System.out.println("Escrito Correctamente...");
        writer.append("Numero incidencia,Código,Descripción,Operativo\n");

        for (Incidencias incidencia2 : incidencia) {
            writer.append(incidencia2.getNumeroIncidencia() + "," +
                incidencia2.getCodigoRef() + "," +
                incidencia2.getDescripccion() + "," +
                incidencia2.getOperativo()+"\n");
        }

    } catch (IOException e) {
        escribirError(e.getMessage());
        System.out.println("Error al actualizar el archivo txt: " + e.getMessage());
    }
}
```

Fichero .log

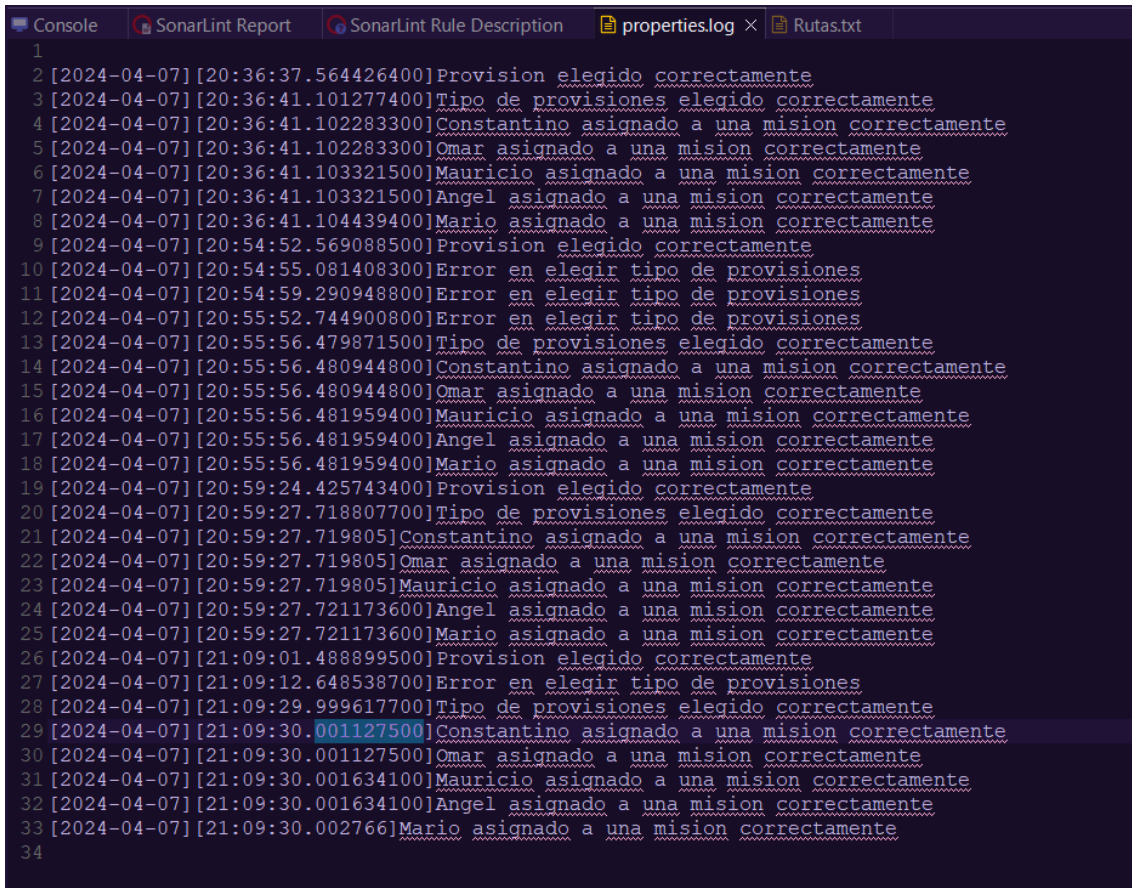
Son fichero en los que podemos ver características o propiedades de nuestro programa, nosotros hemos generado un .log donde vemos los errores que saltan en nuestro programa.

```
public void escribirError(String error) {
    File f = new File("logs/properties.log");
    FileWriter fw;
    try {
        if(f.exists()) {

            fw = new FileWriter(f, true);
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write "["+LocalDate.now()+" "+LocalTime.now()+" "+error+"\n" );
            bw.flush();
            bw.close();
            fw.close();
        } else if(f.length()>=1000) {
            for (int i = 0; i<10;i++) {
                String arch="logs/properties";

                fw = new FileWriter(arch+i+".log", true);
                BufferedWriter bw = new BufferedWriter(fw);
                bw.write "["+LocalDate.now()+" "+LocalTime.now()+" "+error+"\n" );
                bw.flush();
                bw.close();
                fw.close();
            }
        }

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```



```

1
2 [2024-04-07] [20:36:37.564426400] Provision elegido correctamente
3 [2024-04-07] [20:36:41.101277400] Tipo de provisiones elegido correctamente
4 [2024-04-07] [20:36:41.102283300] Constantino asignado a una mision correctamente
5 [2024-04-07] [20:36:41.102283300] Omar asignado a una mision correctamente
6 [2024-04-07] [20:36:41.103321500] Mauricio asignado a una mision correctamente
7 [2024-04-07] [20:36:41.103321500] Angel asignado a una mision correctamente
8 [2024-04-07] [20:36:41.104439400] Mario asignado a una mision correctamente
9 [2024-04-07] [20:54:52.569088500] Provision elegido correctamente
10 [2024-04-07] [20:54:55.081408300] Error en elegir tipo de provisiones
11 [2024-04-07] [20:54:59.290948800] Error en elegir tipo de provisiones
12 [2024-04-07] [20:55:52.744900800] Error en elegir tipo de provisiones
13 [2024-04-07] [20:55:56.479871500] Tipo de provisiones elegido correctamente
14 [2024-04-07] [20:55:56.480944800] Constantino asignado a una mision correctamente
15 [2024-04-07] [20:55:56.480944800] Omar asignado a una mision correctamente
16 [2024-04-07] [20:55:56.481959400] Mauricio asignado a una mision correctamente
17 [2024-04-07] [20:55:56.481959400] Angel asignado a una mision correctamente
18 [2024-04-07] [20:55:56.481959400] Mario asignado a una mision correctamente
19 [2024-04-07] [20:59:24.425743400] Provision elegido correctamente
20 [2024-04-07] [20:59:27.718807700] Tipo de provisiones elegido correctamente
21 [2024-04-07] [20:59:27.719805] Constantino asignado a una mision correctamente
22 [2024-04-07] [20:59:27.719805] Omar asignado a una mision correctamente
23 [2024-04-07] [20:59:27.719805] Mauricio asignado a una mision correctamente
24 [2024-04-07] [20:59:27.721173600] Angel asignado a una mision correctamente
25 [2024-04-07] [20:59:27.721173600] Mario asignado a una mision correctamente
26 [2024-04-07] [21:09:01.488899500] Provision elegido correctamente
27 [2024-04-07] [21:09:12.648538700] Error en elegir tipo de provisiones
28 [2024-04-07] [21:09:29.999617700] Tipo de provisiones elegido correctamente
29 [2024-04-07] [21:09:30.001127500] Constantino asignado a una mision correctamente
30 [2024-04-07] [21:09:30.001127500] Omar asignado a una mision correctamente
31 [2024-04-07] [21:09:30.001634100] Mauricio asignado a una mision correctamente
32 [2024-04-07] [21:09:30.001634100] Angel asignado a una mision correctamente
33 [2024-04-07] [21:09:30.002766] Mario asignado a una mision correctamente
34

```

Como se puede ver muy bien en el ejemplo, cada error añade su línea de código al fichero especificándonos donde se cometió el error, la fecha de cuando sucedió y a que hora agregando minutos segundos y milésimas de segundo.

Resumen

Gracias a las Buenas Practicas no nos fue muy complicado añadir y modificar esta parte.

Nota final

Si bien los **`arrayList`** nos permiten gestionar los datos de nuestro programa, estos no han de ser inicializado si queremos trabajar con ellos, sin posibilidad de guardarlos de ninguna manera. Solventamos con los ficheros este gran problema, pudiendo acceder a ellos una vez el programa cierre su ejecución