Programa de gestión de una Base Aérea Militar

Air Base

DAM Sector

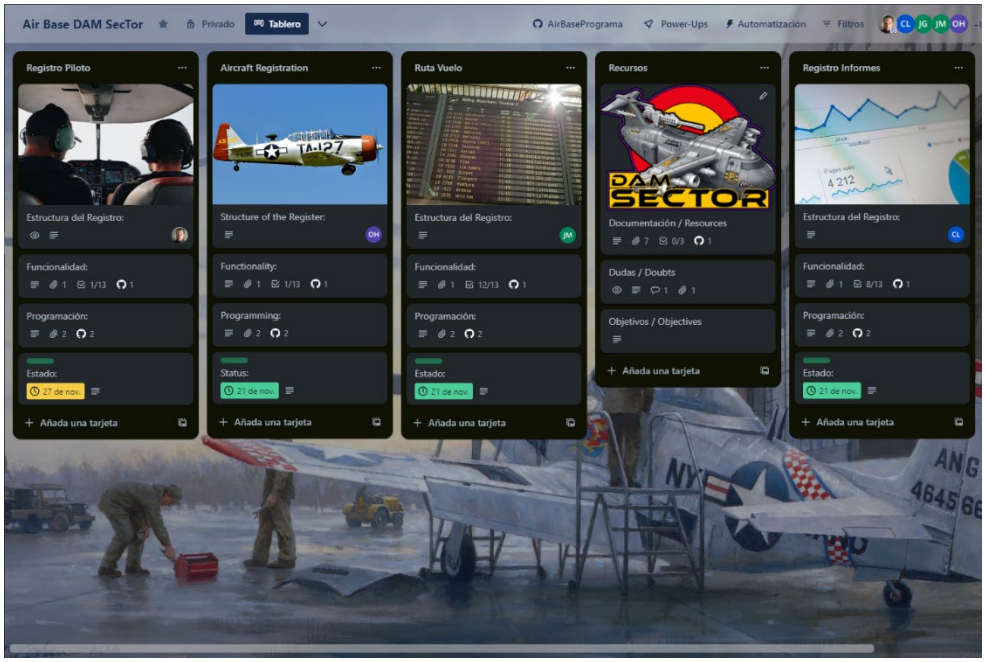
Rafael Antonio Vázquez Flores (FaLy)

Índice

Estructura del trabajo

Explicaremos a continuación la estructura de nuestro Trello y GitHub, como también la mención de como de donde situamos nuestro WorKspace. Esta parte es común en todos los trabajos ya que todos usamos la misma estructura de todo.

Trello



La estructura de trello esta asignada de la siguiente forma, cada integrante del grupo tiene su lista y en ella varias tarjetas, a modo de plantilla.

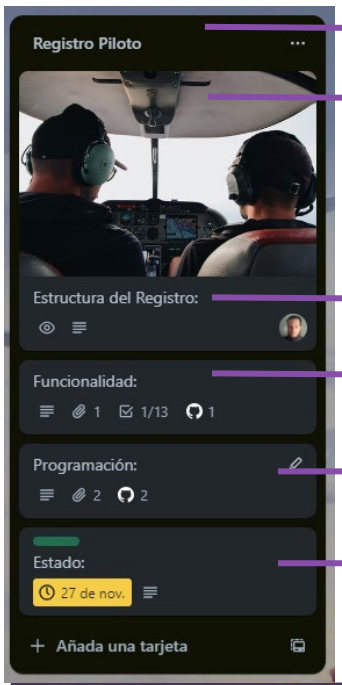


Imagen identificativa

Título de la Lista

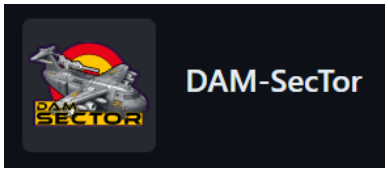
- **Tarjeta** del Registro guardara información sobre la funcionalidad del código a desarrollar.
- **Funcionalidad** hace referencia directa al código, tiene una checklist en función de las características que debe tener el código y enlace con GitHub al recurso.
- **Programación** contiene el Código en bruto de nuestra sección junto con un GitHub al recurso y una actualización del último Commit
- **Estado**, dudas sobre nuestra sección y fechas límites para ir actualizando.

26-9-2023

En la tarjeta de programación de cada una de las listas también hay una pequeña imagen de perfil de la persona que está trabajando en ese código.

Para finalizar en la lista "Recursos" podemos encontrar la documentación de proyecto (diagramas) plantilla del trabajo, estructura general de todo el código, una tarjeta donde cada compañero ira poniendo las dudas que le valla surgiendo, por si otro compañero puede solucionarlas, y una última tarjeta donde se irán definiendo los objetivos de este.

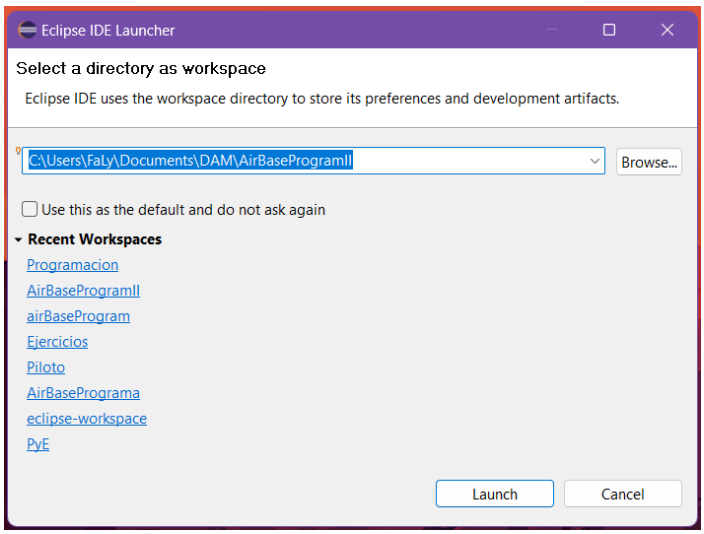
Cada integrante del grupo podrá ver también en esta tarjeta la información relacionada con la persona que hizo el ultimo commit del proyecto.



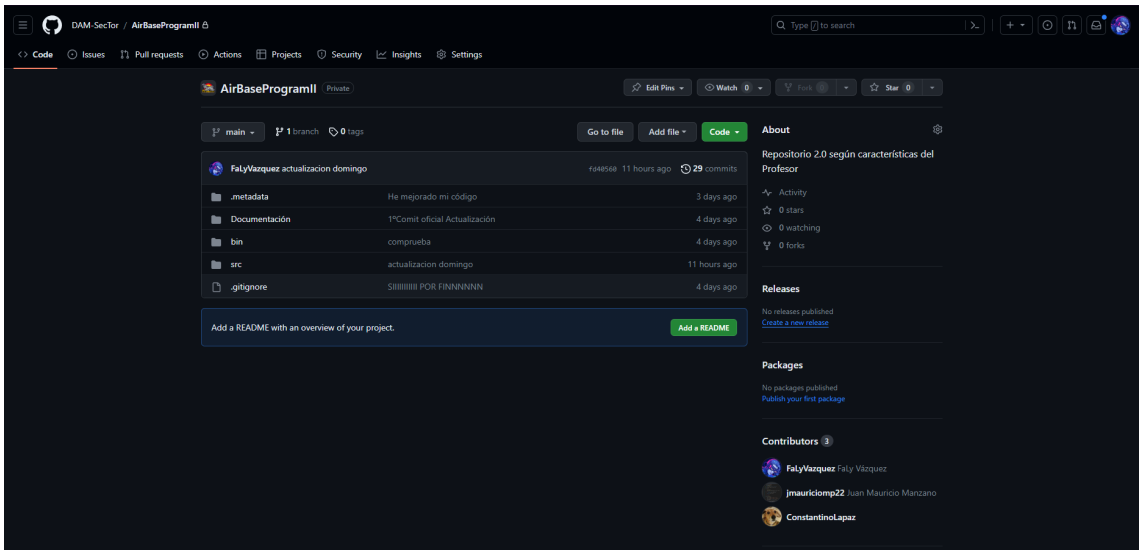
GitHUB

La estructura de GitHub es bastante sencilla hemos creado una organización con unas estructuras de carpetas.

Esta estructura está basada al igual que en trello en secciones de archivo cada uno trabajara en su directorio y usaremos como WorkSpace en eclipse nuestra carpeta asignada así el proyecto de eclipse nuestro directorio.



De esta forma siempre que trabajemos con nuestro proyecto y actualicemos los compañeros podrán ver las modificaciones de nuestra clase.

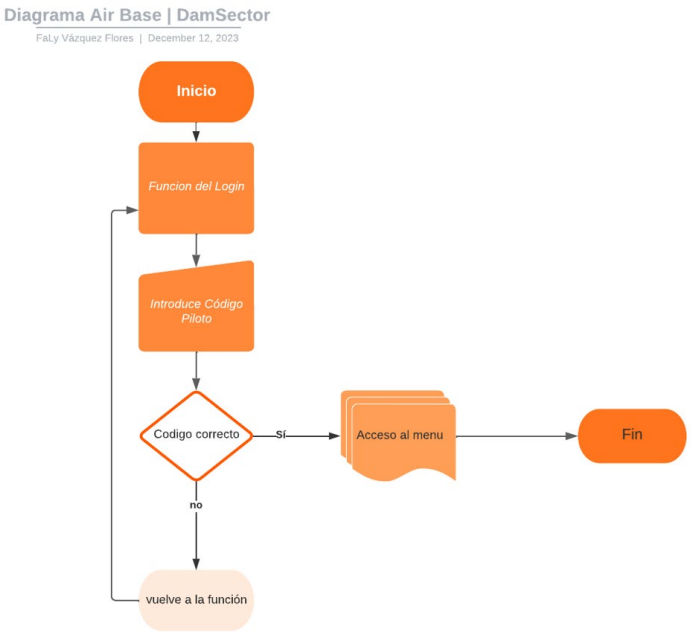


Esta es la estructura de GitHub y la estructura de nuestras package.

En la imagen se puede ver toda la información relativa a nuestra organización.

Toda esta estructura nos permite trabajar de manera conjunta, visualizando las modificaciones de todos los compañeros y aparte organizar el trabajo mediante fechas de entrega y demás. Resulta verdaderamente útil sincronizar trello con GitHub.

Diagrama del Programa parte Login



Clase Registro Piloto Modulo Faly

Mi código, corresponde a la creación de un login para la base aérea, para ello el piloto ingresara su código de piloto y acceder al menú, siempre que este ingrese un código incorrecto o no introduzca nada, no podrá acceder volviendo al menú para que ingrese un nuevo código, puede intentarlo las veces que se quiera asta que ingrese bien el código.

Comenzaremos explicando la clase en la que estuve trabajando.

En las primeras líneas tenemos el ***package piloto*** y la importación de las utilidades de ***Scanner*** y ***Random*** para poder utilziarlo en nuestro main, después creamos la ***public class RegistroPiloto*** que es como llame a mi clase, en la siguiente línea cree un ***String Error***[1]con un mensaje para hacer un llamado cuando sea necesario.

Función Main

Después creamos nuestra función main puede observar en las primeras líneas de código esta comentado una seudo-estructura de la forma que tendrá y como tratare las matrices con las que voy a trabajar, todas estas líneas están comentadas.

```
package piloto;
import java.util.Random;
import java.util.Scanner;

public class RegistroPiloto {
    static String Error = "  -= ERROR =-  ";
    public static void main (String []arg) {
        /*
          Pilotos
              |--Codigo--|----- Nombre ------|Horas|Misiones|Incidencias|
                         Faly          ,...,...,  680 ,   50   ,    16
                         Constantino,...,...,  1500,   246  ,    50
                         Mauricio    ,...,...,  860 ,   120  ,    20
                         Omar        ,...,...,  750 ,   96   ,    15

          Aviones
              |--Codigo--|--Tipo--|Nombre|Horas|Misiones|Incidencias
                -          Cargero , B55  ,5564 , 561    ,    50
                -          Caza    , V486 ,3504 , 300    ,    15
                -          Cargero , B8   ,64   , 5      ,    0
                -          Caza    , Nt4  ,120  , 60     ,    2

          Misiones
              |--Codigo--|----Tipo----|Salida-|Destino|Horas|----Carga-----|
                -          Suministros ,Madrid ,  Irac , ... ,Avituallamiento
                -          Combate     ,NewYork, China , ... ,Municion


            Bases
              |--Continente--|----Nombre----
            - América       , Base Aérea Militar de El Salvador
            - América       , Base Aérea de Fort Hood
            - América       , Base Aérea de Andrews
            - América       , Base Aérea de Los Ángeles
            - América       , Base Aérea de Ramstein
            - América       , Base Aérea de Thule
            - Asia          , Base Aérea de Kadena
            - Asia          , Base Aérea de Misawa
            - Asia          , Base Aérea de Yokota
            - Europa        , Base Aérea de Aviano
            - Europa        , Base Aérea de Ramstein
            - Europa        , Base Aérea de Torrejón
            - Oceanía       , Base Aérea de Hickam
            - Oceanía       , Base Aérea de Andersen
            - Antártida     , Base Aérea McMurdo
            - África        , Base Aérea de Atar
            - África        , Base Aérea de Niamey
            - África        , Base Aérea de Dakar

          */
```

[1] Este error lo utilizo para evitar un llamado a NULL que explicare más adelante.

Después tengo una sección donde tengo las verdaderas matrices a utilizar.

```
//--------- Matrices ---------
String [][] piloto = {
        {"0","FaLy Vázquez Flores","680","50","16"},
        {"0","Contantino Lapaz Gimenez","1500","246","50"},
        {"0","Juan Mauricio Manzano Pachón","860","120","20"},
        {"0","Omar Hamza","750","96","15"}
};
String [][] avion = {
        {"0","Caza","Focke-Wulf Fw 190","957","80","15"},
        {"0","Caza"," F-22 Raptor.","580","50","8"},
        {"0","Transporte"," CASA C-295.","5060","237","36"},
        {"0","Transporte","C-130 Hercules","50","16","2"},
        {"0","Transporte","An-70","1568","140","30"},
        {"0","caza","Panavia Tornado","798","296","78"},
        {"0","Caza","Mitsubishi F-2","850","359","6"},
};
String [][] mision = {
        {"0","Combate"},
        {"0","Transporte"},
        {"0","Combate"},
        {"0","Combate"},
        {"0","Combate"},
        {"0","Transporte"},
        {"0","Transporte"},
        {"0","Combate"},
        {"0","Transporte"},
        {"0","Transporte"},
        {"0","Transporte"},
};
String [][] incidencia = {
        {"0","FaLy Vázquez Flores","680","50","16"},
        {"0","FaLy Vázquez Flores","680","50","16"},
};
String [][] base = {
        {"América","El Salvador"},
        {"América","Fort Hood"},
        {"América","Andrews"},
        {"América","Los Ángeles"},
        {"América","Gotham"},
        {"América","Thule"},
        {"Asia","Kadena"},
        {"Asia","Misawa"},
        {"Asia","Yokota"},
        {"Europa","Aviano"},
        {"Europa","Ramstein"},
        {"Europa","Torrejón"},
        {"Oceanía","Hickam"},
        {"Oceanía","Andersen"},
        {"Antártida","McMurdo"},
        {"África","Atar"},
        {"África","Niamey"},
        {"África","Dakar"},
};
```

26-9-2023

Después le sigue una sección que llamo bambalinas, en esta sección coloco código con el que visualizo algunas cosas que necesito comprobar y algunas líneas que considero que actúan por detrás, como viene siendo las inicializaciones de algunas variables.

```
// --------- Bambalinas --------

Scanner datos = new Scanner (System.in);
String contOrigen = "";
String contDestino = "";


// este llamado es necesario porque sin el no se guarda el codigo generado
// en generarCodigo en las Matrices
registrarCodigo(piloto,incidencia,mision,avion);
System.out.println("------------ Comprobaciones ------------");
System.out.println("");
System.out.println("===== Pilotos ================");
pintarPiloto(piloto);




System.out.println("===== Mision ================");
pintarMision(mision);


System.out.println("===== Bases Aereas ================");

System.out.println(origen(base,contOrigen) + " - " + destino(base,contDestino));
```

la siguiente sección es la que realmente se visualiza en el

```
System.out.println("");
System.out.println("=================================================================");
System.out.println("                                                                 \r\n"
        + "                                                                 \r\n"
        + "                                                                 ");
System.out.println("=================================================================");
System.out.println("");
System.out.println("");
System.out.println("\r\n"
        + "                INICIO DEL PROGRAMA                              \r\n"
        + "                                                                 ");
System.out.println("");


System.out.println("===== Menu ================");
System.out.println("");

loginOK(datos,piloto);
menu();
seleccionarOpcion(datos);


System.out.println("");
System.out.println("\r\n"
        + "                FIN DEL PROGRAMA                                 \r\n"
        + "                                                                 ");
System.out.println("");
```

La parte realmente funcional de mi codigo

programa en sí, lo que ve el usuario del programa

```
------------ Comprobaciones ------------

===== Pilotos ================
======================
391I FaLy Vázquez Flores 680 50 16
102P Contantino Lapaz Gimenez 1500 246 50
393J Juan Mauricio Manzano Pachón 860 120 20
185Z Omar Hamza 750 96 15
======================
===== Mision ================
393J Combate
102P Transporte
102P Combate
185Z Combate
393J Combate
185Z Transporte
102P Transporte
102P Combate
102P Transporte
185Z Transporte
185Z Transporte
===== Bases Aereas ================
Ramstein - Gotham
```

visualización de las comprobaciones

```
=================================================================
  DAM SECTOR SYSTEM II
=================================================================

INICIO DEL PROGRAMA

===== Menu ================

 - Introduzca su codigo de piloto:
sdada
El código no es correcto vuelva a intentarlo
 - Introduzca su codigo de piloto:

 Introduzca algun Dato por favor
 - Introduzca su codigo de piloto:
185Z
  -= Welcom Omar Hamza =-
Bienvenido Piloto
[1]. Misiones
[2]. Horas de Vuelo acumuladas
[3]. Aviones Pilotados
[4]. Incidencias
[5]. Proxima Salida
[6]. Cerrar Conexión
Opción
2

FIN DEL PROGRAMA
```

Visualización real del programa

```
    datos.close();

    }
```

Para cerrar el main colocamos el corchete de la final de la función y cerramos el Scanner.

Funciones

Después del main vienen todas las funciones con las que trabajo y pasare a detallar a continuación.

Funciones de Control

Colocada en esta sección las funciones determinadas de control ya que están creadas para ver la funcionalidad de los códigos y visualizar las matrices que creo.

```
//------------------------------ Funciones de Control -------------

    // pintamos las diferentes Matrices
    public static void pintarPiloto(String[][] piloto) {
        System.out.println("======================");
        for (int i = 0; i < piloto.length; i++) {
            for (int j = 0; j < piloto[i].length; j++) {
                System.out.print(piloto[i][j] + " ");
            }
            System.out.println("");
        }
        System.out.println("======================");
    }

    public static void pintarMision(String[][] mision) {
        for (int i = 0; i < mision.length; i++) {
            for (int j = 0; j < mision[i].length; j++) {
                System.out.print(mision[i][j] + " ");
            }
            System.out.println("");
        }
    }
```

Este código define un método llamado ***pintarPiloto***. El método toma un array de cadenas llamado piloto como entrada. El método está declarado como ***public static void***. Esto significa que el método es accesible desde cualquier parte del programa, no devuelve ningún valor y no requiere la creación de un objeto para llamar al método.

- Impresión del borde:

La primera línea, ***System.out.println("======================");*** imprime una línea horizontal de borde usando el método ***System.out.println***. Este borde ayuda a separar visualmente el patrón impreso del resto del programa.

- Bucles anidados:

Los bucles ***for*** anidados entre sí se utilizan para iterar a través del array piloto e imprimir los valores de los elementos del array. El bucle exterior, ***for (int i = 0; i < piloto.length; i++)***, itera a través de las filas del array. El bucle interior, ***for (int j = 0; j < piloto[i]. length; j++)***, itera a través de las columnas de cada fila.

- Impresión de los elementos del array:

Dentro del bucle interior, el valor del elemento ***piloto[i][j]*** se imprime usando el método ***System.out.print***. Esto asegura que cada elemento del array se imprima en una posición separada dentro del patrón impreso.

- Separación de línea vacía:

Después de imprimir los elementos de cada fila, la instrucción ***System.out.println("");*** imprime una línea vacía. Esto agrega una separación visual entre las filas del patrón impreso.

En resumen, este código define un método que toma un array de cadenas como entrada e imprime los elementos del array en un formato de patrón. Utiliza bucles anidados para iterar a través de las filas y columnas del array e imprime cada elemento en la posición adecuada.

Funciones del Programa

A continuación, describiré el código usado paso a paso en el programa.

Generar Codigo

La primera función en la que trabaje era una que me pudiese generar un código aleatorio dentro de unos parámetros y que me diera una letra al azar.

```java
public static String generarCodigo() {
    Random aleatorio = new Random();
    int valor = aleatorio.nextInt(1001);
    char letra;
    letra = (char) (aleatorio.nextInt(26) + 'A');
    String Codigo =""+valor+letra;
    return Codigo;
}
```

El código funciona de la siguiente manera:

1. La primera línea, ***Random aleatorio = new Random();***, crea un objeto de la clase Random. Esta clase se utiliza para generar números aleatorios.
2. La segunda línea, ***int valor = aleatorio.nextInt(1001);***, genera un número aleatorio entre 0 y 1000.
3. La tercera línea, ***char letra;*** declara una variable de tipo char llamada letra.
4. La cuarta línea, ***letra = (char) (aleatorio.nextInt(26) + 'A');***, genera una letra mayúscula aleatoria. El método ***nextInt()*** genera un número aleatorio entre 0 y 25, y luego se le suma el código ASCII de la letra 'A' (65).
5. La quinta línea, ***String Codigo =""+valor+letra;*** crea una cadena con el número y la letra aleatorios.
6. La sexta línea, ***return Codigo;*** devuelve la cadena.

Generar códigos de manera aleatoria

Para mi aplicación y adelantándome a posibles funcionalidades futuras decidí crear tres funciones que son iguales, pero tiene dos pequeñas variables una para pilotos y otra para los

```java
public static String pilotoAleatorio(String [][] piloto) {
    Random aleatorio = new Random();
    String pilotoRandom = piloto[aleatorio.nextInt(piloto.length)][0];
    return pilotoRandom;
}
public static String origen(String [][] base) {
    Random aleatorio = new Random();
    String contOrigen = base[aleatorio.nextInt(base.length)][1];
    return contOrigen;
}
public static String destino(String [][] base) {
    Random aleatorio = new Random();
    String contDestino = base[aleatorio.nextInt(base.length)][1];
    return contDestino;
}
```

continentes de origen y destino

Como puede comprobar los códigos son idénticos a pesar de que arrojan datos distintos por aplicarlos en otras matrices.

Tomamos como entrada la matriz piloto y mediante un random seleccionamos aleatoriamente un registro de la columna de nombres la guardamos en una variable y retornamos dicha variable cuando la función sea llamada.

Para origen y destino es exactamente igual.

Registramos Código

El código generado aleatoriamente lo registramos en las matrices correspondientes en las columnas correspondientes, una función sencilla pero efectiva que nos permite registrar los códigos generados automáticamente.

Es en esta función que para el registro de misión llamamos a la función de generar códigos de manera aleatoria, en concreto a la función ***pilotoAleatorio(piloto)*** para que en la matriz de misiones genere de manera aleatoria que pilotos han realizado las misiones.

```
public static void registrarCodigo(String [][] piloto,String [][] avion,String [][] mision, String [][] incidencia) {
    for (int i = 0; i < piloto.length; i++ ) {
        piloto [i][0] = generarCodigo();
    }
    for (int i = 0; i < avion.length; i++ ) {
        avion [i][0] = generarCodigo();
    }
    for (int i = 0; i < incidencia.length; i++ ) {
        incidencia [i][0] = generarCodigo();
    }

    for (int i = 0; i < mision.length; i++ ) {
        mision [i][0] = pilotoAleatorio(piloto);
    }
}
```

Buscar código

Esta función nos permite mediante un Boolean saber si el código introducido por el usuario se encuentra en la Matriz, devolviendo si es correcto permitiendo al usuario acceder al menú.

```
public static boolean buscarCodigo (String login, String [][] piloto) {
    boolean correcto = false;
    for (int i = 0; i < piloto.length; i++) {
        if (piloto[i][0].equals(login)) {
            correcto = true;
        }
    }
    return correcto;
}
```

Función login

La principal y más importante de mi sección comprueba los datos de introducidos por el usuario (en este supuesto el piloto) son los correctos o no haciendo que, si el usuario no introduce ningún dato, es incorrecto o introduce caracteres aleatorios no le

26-9-2023

permita entrar pidiéndole nuevamente el código, hasta que no introduzca el dato correcto no le permite el acceso.

```
public static String loginOK(Scanner datos, String[][] piloto) {
      boolean correcto = false;
      while (!correcto) {
        System.out.println(" - Introduzca su codigo de piloto: ");
        String login = datos.nextLine();
        if (login.equals("")){
              System.out.println(" Introduzca algun Dato por favor ");
              correcto = false;
        }else if (!login.equals("") && buscarCodigo(login, piloto)){
            for (int i = 0; i < piloto.length;i++) {
                if (piloto[i][0].equals(login)) {
                    System.out.println("  -= Welcom " + piloto[i][1] + " =- ");
                    correcto = true;
                }
            }
        }else {
            System.out.println("El código no es correcto vuelva a intentarlo");
        }
      }
      return Error;
      }
```

Si explicamos por líneas el código

1. ***boolean correcto = false;*** declara una variable de tipo boolean llamada correcto. Esta variable se utiliza para controlar el bucle while.
2. ***while (!correcto){*** inicia un bucle while. El bucle continuará iterando hasta que la variable correcta se establezca en true.
3. S***ystem.out.println(" - Introduzca su codigo de piloto: ");*** imprime un mensaje en la consola que solicita al usuario que introduzca su código de piloto.
4. **String login = datos.nextLine();**, lee la entrada del usuario y la asigna a la variable login.
5. ***if (login.equals("")){***, comprueba si la variable login está vacía. Si es así, el código imprime un mensaje de error y establece la variable ***correcto*** en false para continuar el bucle while.
6. ***else if (!login.equals("") && buscarCodigo(login, piloto)){***, comprueba si la variable login no está vacía y si existe un piloto con ese código. Si es así, el código itera a través del array piloto buscando el piloto con el código login. Si encuentra el piloto, imprime un mensaje de bienvenida y establece la variable ***correcto*** en true para salir del bucle while.
7. ***else {***, comprueba si la variable login no está vacía y si no existe un piloto con ese código. Si es así, el código imprime un mensaje de error y establece la variable ***correcto*** en false para continuar el bucle while.
8. ***return Error;*** devuelve la cadena Error. Esta cadena se utiliza para indicar que el inicio de sesión ha fallado.

El menú y la opción del menú

Las funciones del menú, y la opción de dicho menú las he puesto en funciones diferentes, con tal de añadir más funciones al trabajo

no hay gran cosa que contar de estas dos funciones una, genera una función con muchas sysos mostrando el menú, y la otra función recoge la opción elegida por el usuario para mostrar el menú correspondiente.

```
public static void menu() {
    System.out.println("Bienvenido Piloto ");
    System.out.println("[1]. Misiones ");
    System.out.println("[2]. Horas de Vuelo acumuladas ");
    System.out.println("[3]. Aviones Pilotados");
    System.out.println("[4]. Incidencias");
    System.out.println("[5]. Proxima Salida");
    System.out.println("[6]. Cerrar Conexión");
}
```

```
public static int seleccionarOpcion(Scanner datos) {
    System.out.println("Opción");
    int opcion = datos.nextInt();
    return opcion;
}
```

Miscelanias

Intentando integrar algunos códigos de mis compañeros me topé con ciertos problemas, que tuve que solventar, partiendo de la base del código de mi compañero mauricio el cual generaba una ruta de vuelo. Decidí implementarlo en mi código modificando que cada destino fuera aleatorio, tras mucho batallar lo solucione de la siguiente manera.

```
public static String origen(String [][] base) {
    Random aleatorio = new Random();
    String contOrigen = base[aleatorio.nextInt(base.length)][1];
    return contOrigen;
}
public static String destino(String [][] base) {
    Random aleatorio = new Random();
    String contDestino = base[aleatorio.nextInt(base.length)][1];
    return contDestino;
}
```

Generando destino y origen aleatorio

Generamos un destino y origen aleatorios que usaremos en la siguiente función donde introduciremos como parámetros la matriz de base, y dos variables definidas en el main que son:

```
String contOrigen = origen(base);
String contDestino = destino(base);
```

definida la función hacemos dos bucles for uno para destino y otro para origen, una vez recorrida la matriz de base condicionamos que si en la columna I1 es igual a la variable definida anteriormente (la cual llamaba a la función que generaba nombres de bases orígenes y destino) guarde en dichas variables el Continente al que pertenece el nombre de la base generada con anterioridad.

```
public static String ruta(String [][] base,String contOrigen,String contDestino) {
    for (int i = 0; i < base.length; i++ ) {
        if (base[i][1].equals(contOrigen)) {
            contOrigen = base [i][0];
        }
    }
    for (int i = 0; i < base.length; i++ ) {
        if (base[i][1].equals(contDestino)) {
            contDestino = base [i][0];
        }
    }
```

Una vez resuelto esto creamos la variable ***int ruta = 0;*** y después un Switch introduciendo el parámetro de ***contOrigen*** cada CASE de este switch contendrá a cada continente, y dentro de cada uno de los CASE a su vez tendrán otro Switch con todos los demás continentes, pero en esta ocasión el parámetro es ***contDestino*** haciendo, todo esto se realiza para generar un numero de horas aleatorias que durara el trayecto.

```
int ruta = 0;
switch(contOrigen) {
case "Europa":
    //Europa--->Africa = [6,10]
    //Europa--->Asia = [4,12]
    //Europa--->Oceania = [10,14]
    //Europa--->America = [8,13]
    //Europa--->Antartida = [5,9]
    switch(contDestino) {
    case "África":
        ruta = generarRandom(6,10);
        break;
    case "Asia":
        ruta = generarRandom(4,12);
        break;
    case "Oceanía":
        ruta = generarRandom(10,14);
        break;
    case "América":
        ruta = generarRandom(8,13);
        break;
    case "Antártida":
        ruta = generarRandom(5,9);
        break;
    default:
        ruta = 4;
    }
```

```
case "Antártida":
    //Antartida--->Africa = [8,12]
    //Antartida--->Asia = [4,8]
    //Antartida--->Oceania = [6,12]
    //Antartida--->America = [2,12]
    //Antartida--->Europa = [5,9]
    switch(contDestino) {
    case "África":
        ruta = generarRandom(8,12);
        break;
    case "Asia":
        ruta = generarRandom(4,8);
        break;
    case "Oceanía":
        ruta = generarRandom(6,12);
        break;
    case "América":
        ruta = generarRandom(2,12);
        break;
    case "Europa":
        ruta = generarRandom(5,9);
        break;
    default:
        ruta = 4;
    }
}
```

```
case  "África":
    //Africa--->Europa = [6,10]
    //Africa--->Asia = [3,11]
    //Africa--->Oceania = [7,12]
    //Africa--->America = [6,13]
    //Africa--->Antartida = [8,12]
    switch(contDestino) {
    case "Europa":
        ruta = generarRandom(6,10);
        break;
    case "Asia":
        ruta = generarRandom(3,11);
        break;
    case "Oceanía":
        ruta = generarRandom(7,12);
        break;
    case "América":
        ruta = generarRandom(6,13);
        break;
    case "Antártida":
        ruta = generarRandom(8,12);
        break;
    default:
        ruta = 4;
    }
case "Asia":
    //Asia--->Africa = [3,11]
    //Asia--->Europa = [4,12]
    //Asia--->Oceania = [2,8]
    //Asia--->America = [3,9]
    //Asia--->Antartida = [4,8]
    switch(contDestino) {
    case "África":
        ruta = generarRandom(3,11);
        break;
    case "Europa":
        ruta = generarRandom(4,12);
        break;
    case "Oceania":
        ruta = generarRandom(2,8);
        break;
    case "América":
        ruta = generarRandom(3,9);
        break;
    case "Antártida":
        ruta = generarRandom(4,8);
        break;
    default:
        ruta = 4;
    }
```

```
case "Oceanía":
    //Oceania--->Africa = [7,12]
    //Oceania--->Asia = [2,8]
    //Oceania--->Europa = [10,14]
    //Oceania--->America = [4,10]
    //Oceania--->Antartida = [6,12]
    switch(contDestino) {
    case "África":
        ruta = generarRandom(7,12);
        break;
    case "Asia":
        ruta = generarRandom(2,8);
        break;
    case "Europa":
        ruta = generarRandom(10,14);
        break;
    case "América":
        ruta = generarRandom(4,10);
        break;
    case "Antártida":
        ruta = generarRandom(6,12);
        break;
    default:
        ruta = 4;
    }
case"America","América":
    //America--->Africa = [6,13]
    //America--->Asia = [3,9]
    //America--->Oceania = [4,10]
    //America--->Europa = [8,13]
    //America--->Antartida = [2,12]
    switch(contDestino) {
    case "África":
        ruta = generarRandom(6,13);
        break;
    case "Asia":
        ruta = generarRandom(3,9);
        break;
    case "Oceania","Oceanía":
        ruta = generarRandom(4,10);
        break;
    case "Europa":
        ruta = generarRandom(8,13);
        break;
    case "Antártida":
        ruta = generarRandom(2,12);
        break;
    default:
        ruta =4;
    }
```

una vez resolvemos toda esta lista, retornamos para finalizar la función el ***contOrigen*** el ***contDestino*** mas las horas de vuelo que durara la Misión.

```
    return (contOrigen + " - " + contDestino + " la mision durara " + ruta + " horas de vuelo");
}
```

Generando la siguiente impresión en consola

```
===== Bases Aereas ================
Ramstein - Atar
Europa - África la mision durara 12 horas de vuelo
```

26-9-2023

La otra función que permite este calculo es generarRandom que crea una numero aleatorio con un mínimo y un máximo.

```
public static int generarRandom(int min, int max) {
    Random aleatorio = new Random();
    int horas = aleatorio.nextInt(max-min+1)+min;
    return horas;
}
```

Dando por finalizada esta parte del trabajo.

Conclusión

A pesar de tener mi código listo desde un momento inicial bastante temprano, el querer avanzar e implementar modificaciones relevantes al código ha hecho que me enfrente a diversos problemas, haciendo que no pueda implementar más funcionalidad al programa.

Meta

Para el siguiente trimestre, me gustaría entregar una aplicación altamente funcional con las clases del resto de compañeros completamente integradas en una sola clase.