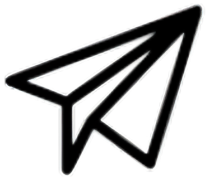
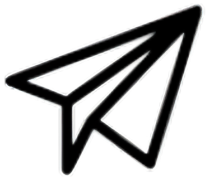
AEROPUERTOS JUAN

Mijael Tamayo
Ametz Segovia
Jon Urrutia





Junit

Creemos que JUnit es importante para garantizar la calidad del software, facilitar el desarrollo, detectar errores tempranamente y proporcionar una base sólida para el mantenimiento y evolución del código.

Pruebas conexión

Sesión

Probamos si la sesión funciona correctamente.

```
@Test
@RepeatedTest(3)
public void prueba_sesion() {
    String resultado="cdca39b6089d200e8907ce09a5603cb3ff736b2e952fd35d466d0892c4447f4b";
    String contraseña = "Zubiri123";
    String contraseña2 = "si123";
    Assertions.assertEquals(Inicio_de_sesion.getHashedPassword(contraseña),resultado);
}
```

Contar filas de vuelos

Probamos si la cantidad de líneas que nos devuelve son correctas.

```
@Test
public void pruebaContFilas() {
    Conexion conexion = new Conexion();
    int numFilas = Inicio_de_sesion.Contfilas(conexion);
    assertEquals( expected: 32, numFilas);
}
```

Windows

Solo de que se ejecute en windows.

```
@Test
@EnabledOnOs(OS.WINDOWS)
public void pruebaWindows() {
    Assertions.assertTrue( condition: true);
}
```

Driver

Probamos si se crea correctamente la conexión con la base de datos.

```
@Test
@BeforeAll
public void pruebadriver() {
    Conexion con = new Conexion();
    Assertions.assertNotNull(con);
}
```

Hash

Probamos si el método de hash devuelve lo que queremos.

```
@Test
public void PruebaHash(){
    String contraseña= "Zubiri123";
    String hash="cdca39b6089d200e8907ce09a5603cb3ff736b2e952fd35d466d0892c4447f4b";
    assertEquals(hash, Inicio_de_sesion.getHashedPassword(contraseña));
}
```

Vuelos

Nos tiene que devolver un Array doble con los datos de la tabla vuelos.

```
@Test
public void Pruebavuelos(){
    Conexion conexion = new Conexion();
    String[] Columnas = {"Avion", "Id_vuelo", "Origen", "Destino", "Salida", "Llegada", "Precio"};
    String[][] vuelos = Inicio_de_sesion.Mostrarvuelos(conexion,Columnas);
    Assertions.assertNotNull(vuelos);
}
```

Pasajes

Nos tiene que devolver un Array doble con los datos de la tabla pasajes.

```
@Test
public void Pruebapasajes(){
    Conexion conexion = new Conexion();
    Cliente cliente = new Cliente( DNI: "05752043L",  nombre: "Teri",  apellido: "Stainer",
    String[] Columnas = {"Avion", "Origen", "Destino", "Salida", "Llegada", "Precio"};
    String[][] pasajes = InicioClientes.Mostrarpasajes(conexion,Columnas,cliente);
    Assertions.assertNotNull(pasajes);
}
```

Cerrar sesión

Tiene que eliminar al usuario y devolvernos la pantalla de inicio.

```
@Test
public void Pruebacerrarsesion(){
    Cliente cliente = new Cliente( DNI: "05752043L",  nombre: "Teri",  apellido: "Stainer",
    cliente = null;
    Assertions.assertNull(cliente);
}
```

Vuelos del piloto

Nos tiene que devolver un Array doble con los datos de la tabla Vuelos que pertenecen al piloto.

```
@Test
public void Pruebavuelospiloto(){
    Conexion conexion = new Conexion();
    Piloto piloto = new Piloto( numemp: 99,  nombre: "Tiffi",  apellido: "Braiden",  trabajo: "piloto",
    String[] Columnas = {"Avion", "Id_vuelo", "Origen", "Destino", "Salida", "Llegada"};
    String[][] vuelos = InicioPiloto.Mostrar_mis_vuelos(conexion,Columnas,piloto);
    Assertions.assertNotNull(vuelos);
}
```

Contar vuelos del piloto

Probamos si nos devuelve la cantidad de vuelos del piloto que esperamos.

```
@Test
public void PruebacontVuelospiloto(){
    Conexion conexion = new Conexion();
    Piloto piloto = new Piloto( numemp: 99,  nombre: "Tiffi",  apellido: "Braiden",
    int cant = InicioPiloto.Contfilas_vuelos_propios(conexion,piloto);
    Assertions.assertEquals( expected: 2, cant);
}
```

Pasajes filtrados

Nos tiene que devolver un Array doble con los datos de la tabla pasajes que pertenecen al vuelo pedido.

```
@Test
public void Pruebamostrarpasajes(){
    Conexion conexion = new Conexion();
    int id_vuelo = 3;
    String[] Columnas_pasajes = {"Id pasaje", "Id vuelo", "Clase", "Asiento", "Precio", "DNI"};
    String[][] pasajes = InicioRecepcion.MostrarPasajes(conexion,id_vuelo,Columnas_pasajes);
    Assertions.assertNotNull(pasajes);
}
```

Contar pasajes

Probamos si nos devuelve la cantidad de pasajes del vuelo que esperamos.

```
@Test
public void Pruebacontpasajes(){
    Conexion conexion = new Conexion();
    int id_vuelo = 3;
    int cant = InicioRecepcion.Contfilas_Pasaje(conexion,id_vuelo);
    Assertions.assertEquals( expected: 5, cant);
}
```

Pruebas CRUD

Crear piloto

Probamos en constructor de piloto.

```
@Test
public void pruebacrearpiloto() {
    Assertions.assertNotNull(new Piloto( numemp: 1,  nombre: "Juan",  apellido: "Aereo",  trabajo: "piloto",
}
```

Crear cliente

Probamos en constructor de cliente.

```
@Test
public void pruebacrearcliente() {
    Assertions.assertNotNull(new Cliente( DNI: "05752043L",  nombre: "Teri",  apellido: "Stainer",
}
```

Leer piloto

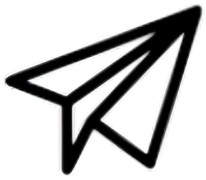
Probamos el get del piloto.

```
@Test
public void pruebaleerpiloto() {
    Piloto p1 = new Piloto( numemp: 1,  nombre: "Juan",  apellido: "Aereo",  trabajo: "piloto",
    Assertions.assertEquals(p1.getNombre(),  actual: "Juan");
}
```

Leer cliente

Probamos el get del cliente.

```
@Test
public void pruebaleercliente() {
    Cliente cliente = new Cliente( DNI: "05752043L",  nombre: "Teri",  apellido: "Stainer",
    Assertions.assertEquals(cliente.getNombre(),  actual: "Teri");
}
```



Cambiar piloto

Probamos el set de piloto.

```
@Test
public void prubacambiarpiloto() {
    Piloto p1 = new Piloto( numemp: 1,  nombre: "Juan",  apellido: "Aereo",  trabajo: "piloto",
    p1.setNombre("Juan1");
    Assertions.assertEquals(p1.getNombre(), actual: "Juan1");
}
```

Cambiar cliente

probamos el set del cliente.

```
@Test
public void pruebacambiarcliente() {
    Cliente cliente = new Cliente( DNI: "05752043L",  nombre: "Teri",  apellido: "Stainer",
    cliente.setNombre("Teri1");
    Assertions.assertEquals(cliente.getNombre(), actual: "Teri1");
}
```

Eliminar piloto

Eliminamos el piloto.

```
@Test
public void Pruebaeliminarpiloto() {
    Piloto p1 = new Piloto( numemp: 1,  nombre: "Juan",  apellido: "Aereo",  trabajo: "piloto",
    p1 = null;
    Assertions.assertNull(p1);
}
```

Eliminar cliente

Eliminamos el cliente.

```
@Test
public void pruebaeliminarcliente() {
    Cliente cliente = new Cliente( DNI: "05752043L",  nombre: "Teri",  apellido: "Stainer",
    cliente = null;
    Assertions.assertNull(cliente);
}
```