

# PRÁCTICA ENTORNOS DE DESARROLLO



## **EFA EL CAMPICO**

### **Entornos de desarrollo**

#### **PRÁCTICA FINAL**

Autor:

**Jose Antonio Ortuño Ortuño**

Profesor:

**Pedro Serna Sánchez**

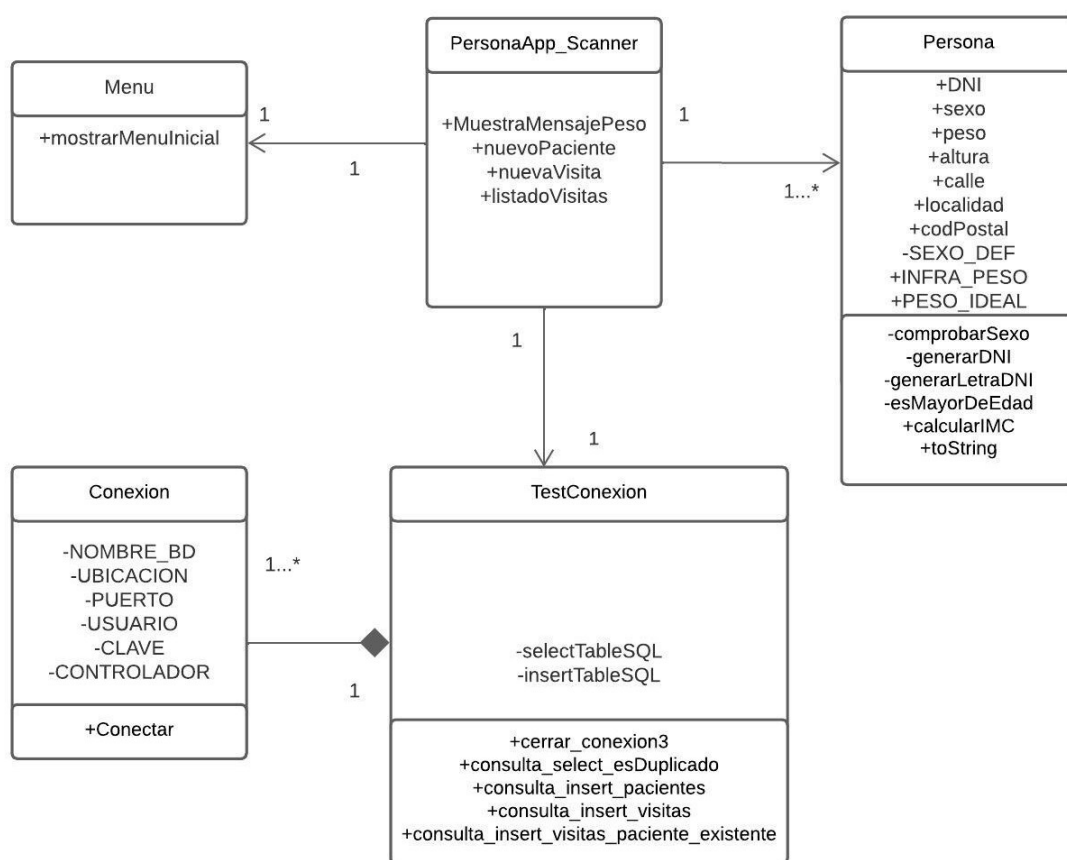
Orihuela, 21 Mayo de 2023

## ÍNDICE

1. Diagramas UML y de actividades.....	pag. 3
2. Base de datos.....	pag. 6
3. Fase de testing: Junit.....	pag. 8
4. Posibles mejoras y ampliaciones.....	pag.11

## 1.- Diagramas UML y de actividades

A continuación, se muestra el diagrama de clases asociado a la práctica final de la asignatura. Indicar que, tanto para este diagrama como para el de actividades se ha usado la herramienta gratuita Lucid Chart.



En dicha figura se pueden ver cada una de las 5 clases asociadas a dicho proyecto (Menu , PersonaApp\_Scanner , Persona , Conexion y TestConexion) donde se muestran los atributos de cada una de las mismas así como sus métodos asociados. Los atributos de cada clase se han listado, aunque se tratasen de constantes, ya que son propiamente atributos de las mismas.

Las relaciones que se han considerado entre ellas serían las siguientes:

- Menu – PersonaApp\_Scanner, se considera una relación de asociación entre ellas de 1 a 1. Una instancia de PersonaApp\_Scanner solo puede tener un

objeto de tipo Menu . No se considera otra relación de agregación o composición ya que no tienen más relación conceptual entre ambas salvo que una tiene una referencia a la otra.

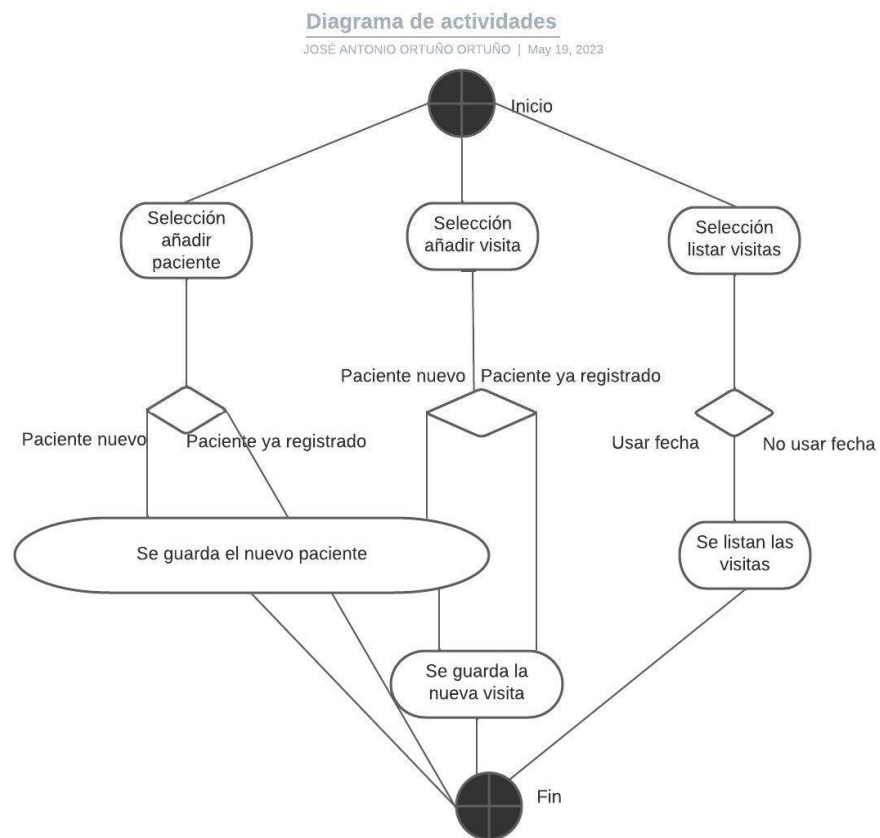
- PersonaApp\_Scanner – Persona, se considera una relación de asociación 1 a 1..\* dado que un objeto tipo PersonaApp-Scanner puede generar varios objetos tipo Persona.
- PersonaApp\_Scanner – TestConexion , se considera una relación de asociación 1 a 1 dado que un objeto Persona\_App Scanner genera un objeto tipo TestConexion.
- TestConexion – Conexion, se considera una relación de asociación 1 a 1..\* ya que un objeto TestConexion puede generar varios objetos de tipo Conexión según el método del primero al que se llame (cada uno de los cuales estaría asociado a cada una de las queries que puede realizar la aplicación en BBDD). Se entiende que la relación entre ambas clases sería de composición ya que TestConexion no tendría sentido si no existiese la clase Conexion que va a probar explícitamente , lo cual se ha indicado en el diagrama con el rombo relleno en negro.

**i Ojo !.** Faltaría por añadir la clase Config.java con su método leerConfig() , que se ha añadido a posteriori , para leer la configuración de la conexión a BBDD desde fichero y que también tendría una relación de asociación con PersonaApp\_Scanner de cardinalidad 1.

#### Notas acerca de las versiones anteriores

El diagrama de clases UML previo se corresponde con la versión actual 3 de la aplicación en la que se ha añadido nueva funcionalidad y guardado en BBDD. Para la versión anterior simplemente habría que eliminar la clase Conexión y sustituir TestConexion por TratamientoFicheros , todo seguiría siendo idem. Y para la primera versión la rama inferior se eliminaría esta última clase también.

Lo siguiente sería indicar el diagrama de actividades que se adjunta seguidamente:



En dicho diagrama se puede observar que hay principalmente 3 casos de prueba dependiendo de la selección del usuario:

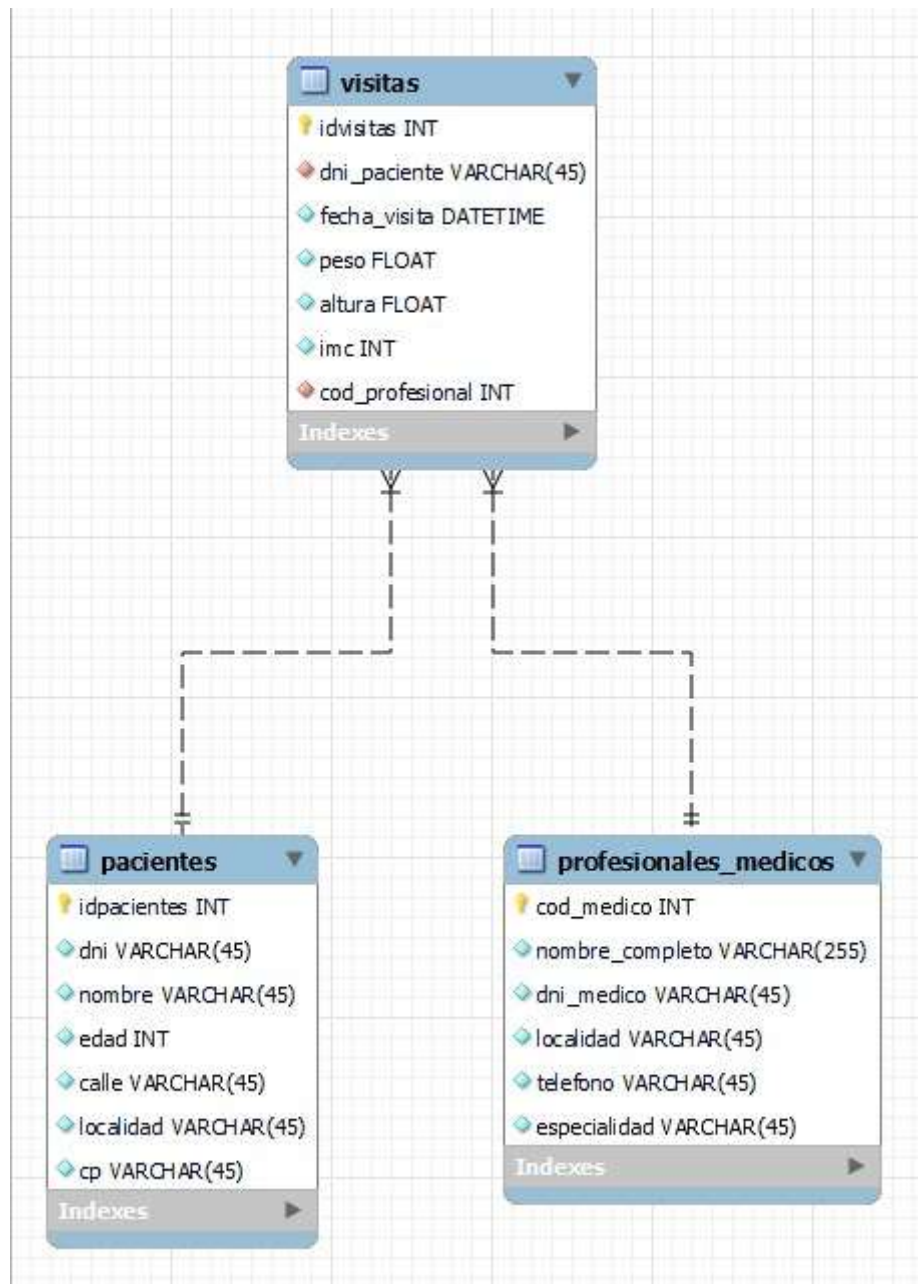
1. Añadir paciente, que se añadirá en caso de que no se encuentren coincidencias del mismo en BBDD. En caso contrario no se añade el mismo y se informa al usuario.
2. Añadir visita, para lo cual se discrimina si se trata de un paciente nuevo o uno ya existente. Si ya está registrado se añade la visita directamente, si es nuevo se añadirá el mismo y posteriormente la visita asociada.
3. Listar visitas, para lo cual se puede considerar o no una fecha informada por el usuario.

## Notas acerca de las versiones anteriores

El diagrama de actividades previo se corresponde con la versión actual 3 de la aplicación actual, para la versión anterior se sustituiría el estado “Se guarda X en BBDD” por “Se guarda X en fichero” y se eliminaría la rama de listas visitas. Por otro lado, para la versión inicial se cambiaría el guardado en fichero por “Se genera X” donde X sería el paciente o la visita asociada.

## 2.- Base de datos

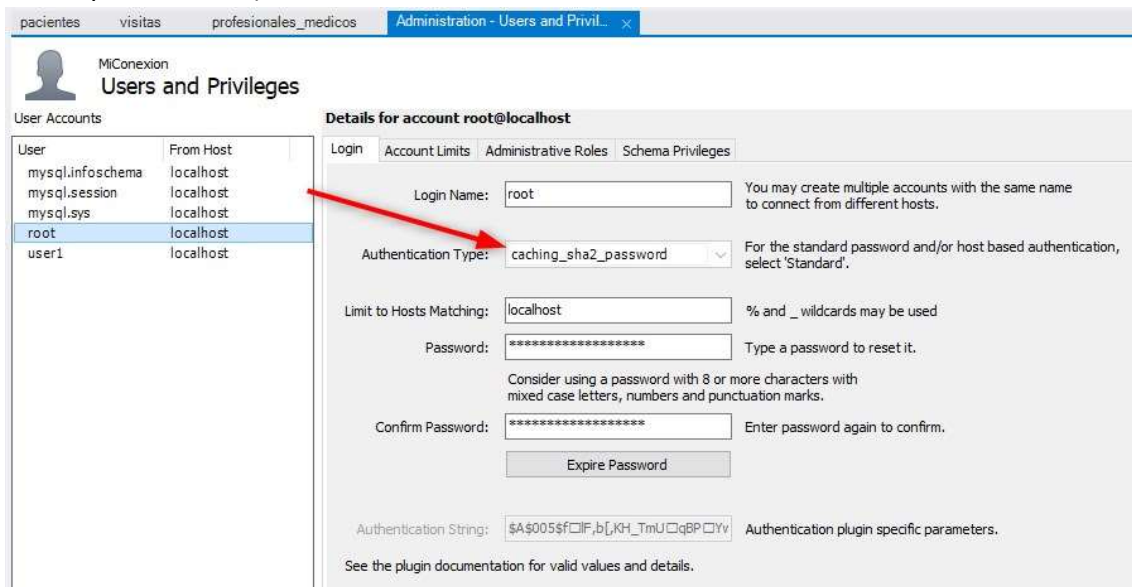
Con respecto al modelo de BBDD que se ha generado en el Workbench sería algo tal que así:



, donde se puede comprobar que la tabla Pacientes tiene una relación 1:N con Visitas , ya que un paciente puede realizar varias visitas y cada una de ellas , a su vez , tiene asociado un médico. A su vez la tabla Profesionales\_medicos también tiene una relación 1:N con visitas ya que un mismo médico puede ser asignado a varias visitas realizadas por un mismo paciente.

A parte de este diagrama en el repo de dicha práctica se ha adjuntado también un .sql con la estructura y los datos de las tablas asociadas.

Y por último indicar que se ha generado un nuevo usuario de BBDD (con método de autenticación Standard) para conectar con la BBDD que se ha usado ya que con el usuario root daba problemas de autenticación (el cliente no soporta la autenticación usada por el server).



The screenshot shows the MySQL 'Users and Privileges' window. On the left, a table lists existing users. A red arrow points from the 'root' user in this table to the 'Details for account root@localhost' panel on the right. The 'Authentication Type' is set to 'caching\_sha2\_password'.

User	From Host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost
user1	localhost

**Details for account root@localhost**

Login Name: root

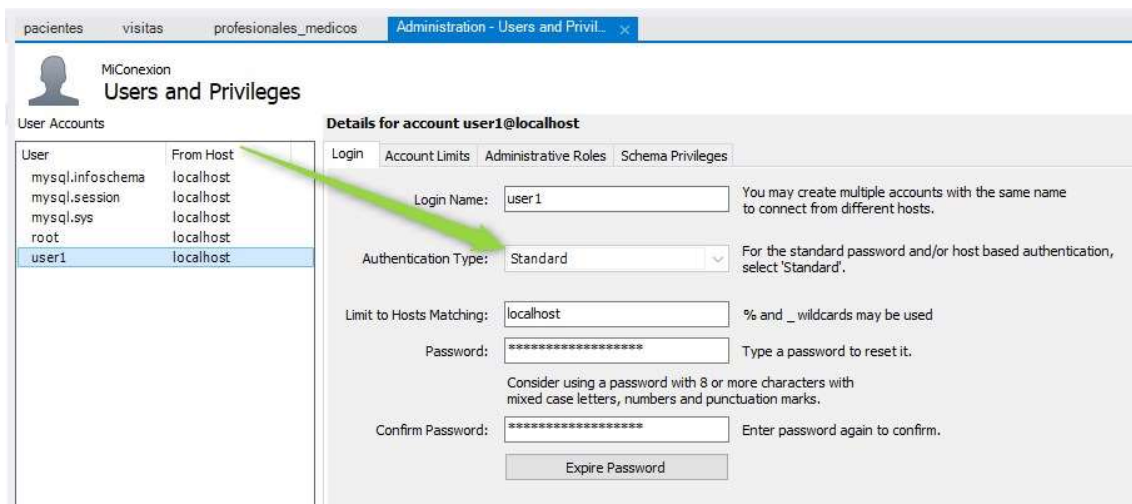
Authentication Type: caching\_sha2\_password

Limit to Hosts Matching: localhost

Password: [masked]

Confirm Password: [masked]

Authentication String: \$A\$005\$f0IF,b[.KH\_TmU0qBP0Yv



The screenshot shows the MySQL 'Users and Privileges' window. On the left, a table lists existing users. A green arrow points from the 'user1' user in this table to the 'Details for account user1@localhost' panel on the right. The 'Authentication Type' is set to 'Standard'.

User	From Host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost
user1	localhost

**Details for account user1@localhost**

Login Name: user1

Authentication Type: Standard

Limit to Hosts Matching: localhost

Password: [masked]

Confirm Password: [masked]

Pasamos a la parte del código. Con respecto a las pruebas de validación de JUnit se han generado las clases de tests asociadas a las clases a validar.

### 3.- Fase de testing: JUnit

Según nos indica el enunciado hay que realizar el testing con JUnit de la fase 1 y 2.1 de la aplicación actual. La fase 1 consiste en añadir una clase Profesionales\_medicos con los atributos indicados, dicha clase se ha incluido, pero no se está usando en la propia aplicación ya que no se pide explícitamente que se puedan dar de alta médicos con la misma.

Lo que sí que ha hecho son inserciones en la tabla asociada de BBDD de 5 registros correspondientes a 5 médicos disponibles que luego se podrán asociar a cada una de las visitas nuevas dadas de alta con la aplicación. Esa asociación ahora es obligatoria.

Tras generar la clase Profesionales\_medicos las únicas pruebas que se pueden realizar de la misma son aquellas asociadas a los métodos getters y setters de dicha clase. Se ha generado una clase Profesionales\_medicosTest en la que se añaden los métodos de prueba para cada uno de los getters y setters asociados.

Se comprueba que funcionan correctamente los mismos y que se cubre el 100% de cobertura de dicha clase, como se puede ver en la captura adjunta.

The screenshot shows an IDE with several tabs open: Persona.java, Subscripcion.java, SubscripcionTest.java, PersonaApp\_Scanner.java, Profesionales\_medicos.java, and Profesionales\_medicosTest.java. The main editor displays the code for Profesionales\_medicosTest.java, which includes three test methods: setNombre(), setDni\_medico(), and setLocalidad(). Each method creates a new Profesionales\_medicos object, sets a specific attribute, and then asserts that the get method returns the expected value.

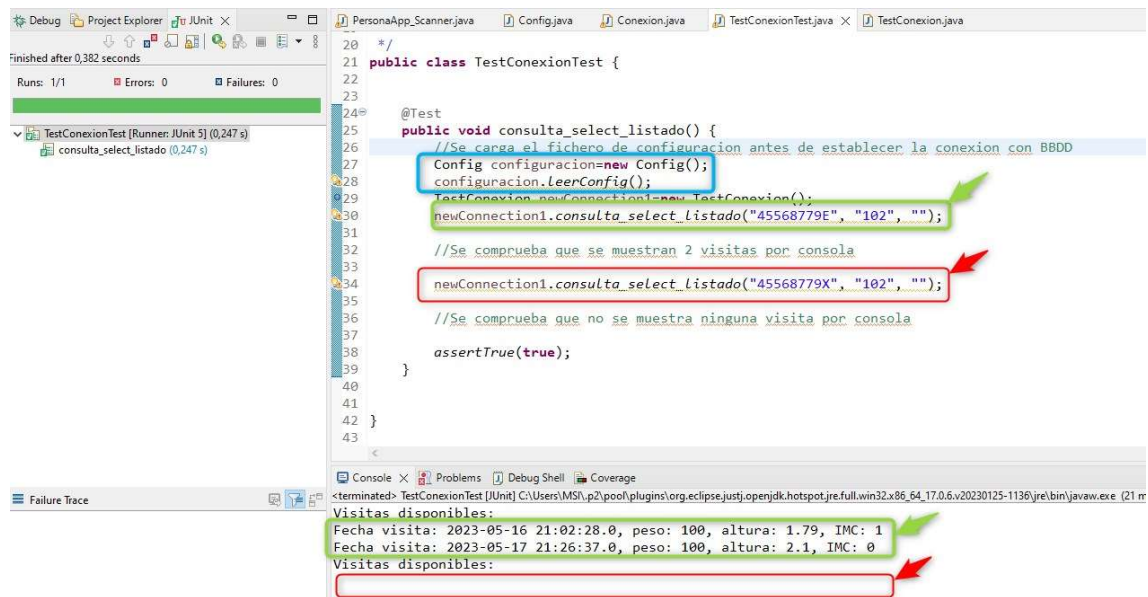
Below the code editor, the 'Coverage' tab is active, showing a table of coverage data for the project. A green arrow points to the 'Profesionales\_medicosTest.java' row, which shows 100% coverage.

Element	Coverage	Covered Instruction...	Missed Instructions	Total Instructions
Practica_final_ED	10,4 %	211	1.816	2.027
src	10,4 %	211	1.816	2.027
PracticaED	10,4 %	211	1.816	2.027
TestConexion.java	0,0 %	0	436	436
TratamientoFicheros.java	0,0 %	0	420	420
PersonaApp_Scanner.java	0,0 %	0	399	399
Persona.java	0,0 %	0	326	326
FileScanner.java	0,0 %	0	68	68
Menu.java	0,0 %	0	51	51
EscribeFichero.java	0,0 %	0	49	49
Conexion.java	0,0 %	0	30	30
LeerFichero.java	0,0 %	0	30	30
PersonaApp_ScannerTest.java	0,0 %	0	7	7
Profesionales_medicos.java	100,0 %	53	0	53
Profesionales_medicosTest.java	100,0 %	158	0	158



Se procedería idem con la funcionalidad 2.1 listado de visitas por paciente/médico sin usar fecha.

Se genera una nueva clase de test con un método de prueba para validar aquel método de listado de TestConexion asociado al listado de visitas. Se ejecuta con unos valores que deben devolver visitas y otros que no devolverían nada por consola y se valida el mismo. Ojo , hay que tener en cuenta que antes de instanciar un objeto TestConexion hay que llamar a los métodos que cargan la configuración de la conexión a BBDD , en caso contrario fallará.



```
20 */
21 public class TestConexionTest {
22
23
24 @Test
25 public void consulta_select_listado() {
26     //Se carga el fichero de configuracion antes de establecer la conexion con BBDD
27     Config configuracion=new Config();
28     configuracion.LeerConfig();
29     TestConexion newConexion1=new TestConexion();
30     newConexion1.consulta_select_Listado("45568779E", "102", "");
31
32     //Se comprueba que se muestran 2 visitas por consola
33
34     newConexion1.consulta_select_Listado("45568779X", "102", "");
35
36     //Se comprueba que no se muestra ninguna visita por consola
37
38     assertTrue(true);
39 }
40
41 }
42
43 }
```

Failure Trace

Console

<terminated> TestConexionTest [JUnit] C:\Users\MSI\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86\_64.17.0.6.v20230125-1136\jre\bin\javaw.exe (21 m)

Visitas disponibles:

Fecha visita: 2023-05-16 21:02:28.0, peso: 100, altura: 1.79, IMC: 1

Fecha visita: 2023-05-17 21:26:37.0, peso: 100, altura: 2.1, IMC: 0

Visitas disponibles:

The screenshot shows an IDE with a Java test class `TestConexionTest` and its coverage report.

**Test Class Code:**

```

20 */
21 public class TestConexionTest {
22
23
24 @Test
25 public void consulta_select_listado() {
26     //Se carga el fichero de configuracion antes de establecer la conexion con BBDD
27     Config configuracion=new Config();
28     configuracion.LeerConfig();
29     TestConexion newConexion1=new TestConexion();
30     newConexion1.consulta_select_Listado("45568779E", "102", "");
31
32     //Se comprueba que se muestran 2 visitas por consola
33
34     newConexion1.consulta_select_Listado("45568779X", "102", "");
35
36     //Se comprueba que no se muestra ninguna visita por consola
37
38     assertTrue(true);
39 }
40
41
42 }
43

```

**Coverage Report:**

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Practica_final_ED	16,3 %	262	1.346	1.608
src	16,3 %	262	1.346	1.608
PracticaED	16,3 %	262	1.346	1.608
PersonaApp_Scanner.java	0,0 %	0	404	404
TestConexion.java	22,5 %	98	338	436
Persona.java	0,0 %	0	326	326
Profesionales_medicosTest.java	0,0 %	0	158	158
Profesionales_medicos.java	0,0 %	0	53	53
Menu.java	0,0 %	0	51	51
Conexion.java	79,3 %	46	12	58
Config.java	96,0 %	95	4	99
TestConexionTest.java	100,0 %	23	0	23

Con respecto a GIT indicar que todo el código , fichero .rar con los .sqls y guión de la practica se ha subido a la rama ED\_final\_develop con su carpeta asociada ([https://github.com/GithubOrtu1/RepoOrtuED/tree/ED\\_final\\_develop](https://github.com/GithubOrtu1/RepoOrtuED/tree/ED_final_develop)) y en la rama ED\_final\_testing se ha subido lo mismo incluyendo las dos clases de prueba de Junit ([https://github.com/GithubOrtu1/RepoOrtuED/tree/ED\\_final\\_testing](https://github.com/GithubOrtu1/RepoOrtuED/tree/ED_final_testing)) .

Se entiende que , una vez calificada (revisada) dicha práctica , ya se subiría a la rama master para ser más realista con el procedimiento habitual (una vez realizada y revisada la PR).

#### **4.- Posibles mejoras y ampliaciones**

Como posibles mejoras/ampliaciones de esta aplicación se podrían plantear las siguientes:

1. Añadir interfaz gráfica. Sería sencilla de implementar y serviría para limitar los posibles inputs inválidos introducir por el usuario.
2. Añadir profesionales médicos. No se ha indicado como requisito, se decía explícitamente que ya había 5 profesionales médicos almacenados en el sistema por lo cual se han introducido los mismos directamente por BBDD.
3. Uso de objetos tipo Pacientes , Visitas y Profesionales\_Medicos incluso considerando la posibilidad de usar colecciones (arrayLists) de los mismos. Esto tendría más sentido cuando hubiese lógica asociada al procesamiento de los mismos, que ahora mismo hay. La aplicación se limita a la captura/mostrado d datos de ahí que no se considerase necesario actualmente, pero en un futuro sería conveniente.