

## PROJECTE VET

### INDEX

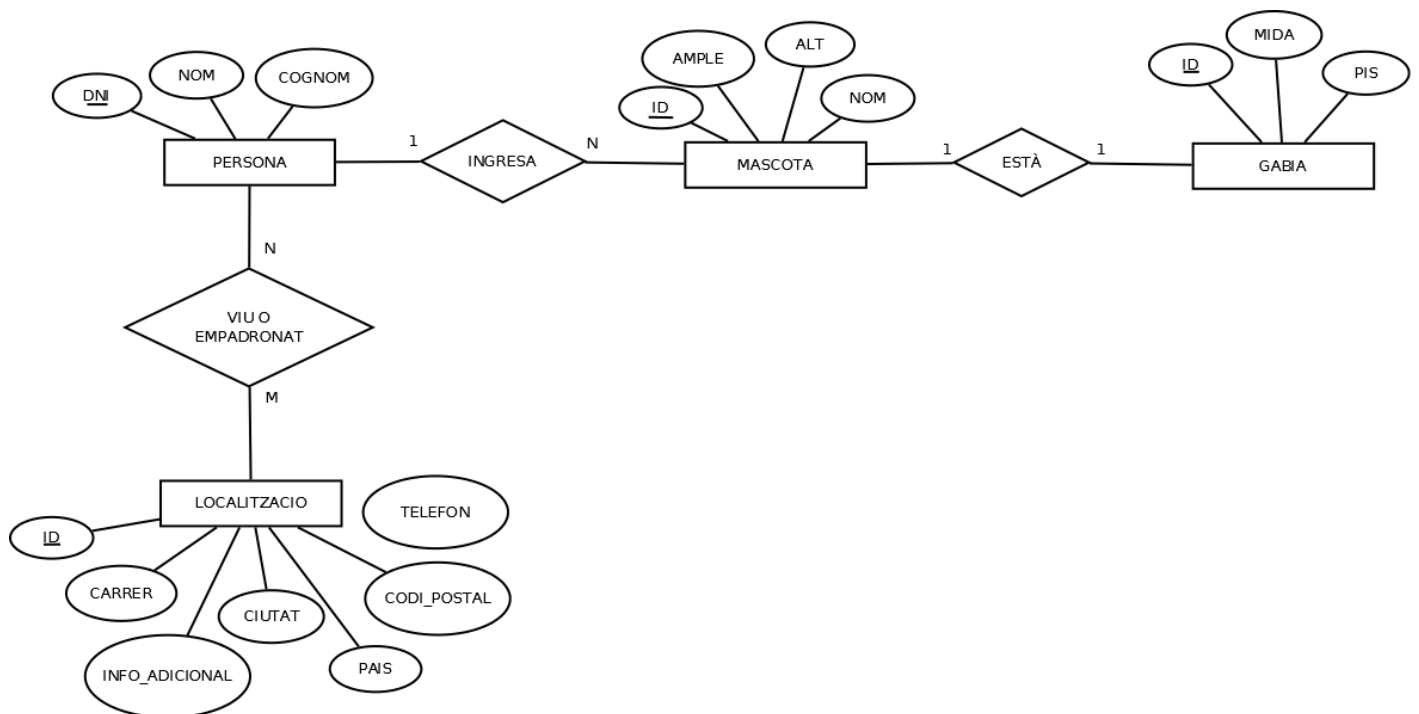
Presentació.....	2
Diagrama.....	2
Hibernate i Entitats.....	3
Taules finals.....	3
Test.....	4
Test Inserció.....	4
Test Junit.....	7

## Presentació

Aquest projecte tracte de mostrar el funcionament d'una aplicació de gestió i administració de veterinària molt simplificada i totes les relacions que tenen les taules entre si.

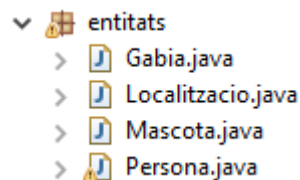
## Diagrama

El diagrama té totes les possibles relacions. 1-1, 1-N i M-N



## Hibernate i Entitats

Entitats creades:



Les relacions creades són les següents:

### Persona

```
@ManyToMany(cascade=CascadeType.ALL)

@JoinTable(name="localitzacions_persones",
joinColumns={@JoinColumn(name="DNI_PERSONA")},
inverseJoinColumns={@JoinColumn(name="ID_LOCALITZACIO")})
@JoinColumn(name="ID_LOCALITZACIO")
private Set<Localitzacio> localitzacio = new LinkedHashSet();
```

### Mascota

```
@OneToOne


@JoinColumn(name="ID_GABIA")
private Gabia gabia;

@ManyToOne
@JoinColumn(name="DNI_PERSONA")
private Persona persona;
```

## Taules finals

S'han creat finalment les següents taules amb les estructures onetomany, onetoone i manytomany:

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
gabia	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	utf8_spanish_ci	16 KB	-
localitzacio	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	utf8_spanish_ci	16 KB	-
localitzacions_persones	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	6	InnoDB	utf8_spanish_ci	32 KB	-
mascota	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8_spanish_ci	48 KB	-
persona	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	5	InnoDB	utf8_spanish_ci	16 KB	-
5 tablas	Número de filas	25	MyISAM	utf8_spanish_ci	128 KB	0 B

 <b>INSTITUT AUSIÀS MARCH</b> Consorci d'Educació de Barcelona Generalitat de Catalunya Ajuntament de Barcelona	<b>UF6 – Introducció a la persistència en BBDD</b> <b>Projecte Final</b>	<b>Mòdul 03:</b> <b>Programació</b> <b>Eduard Vallès</b>
--	---	--

## Test

A continuació veurem captures dels test realitzats, tan captures directament de la base de dades com test mitjançant la utilitat JUNIT:

## Test Inserció

Realitzem les següents insercions directament, i comprovarem a les taules com les relacions estan creades correctament:

```
LocalitzacioDAOImplementacio l = new LocalitzacioDAOImplementacio();
PersonaDAOImplementacio p = new PersonaDAOImplementacio();
MascotaDAOImplementacio m = new MascotaDAOImplementacio();
GabiDAOImplementacio g = new GabiDAOImplementacio();

Localitzacio loc1 = new Localitzacio(8028, "Avinguda Madrid 40", "Pis quart, porta dos", "Barcelona","Espanya", 666666666);
Localitzacio loc2 = new Localitzacio(8029, "Passeig de Sant Juan", "sense número", "Barcelona","Espanya", 45957854);
Localitzacio loc3 = new Localitzacio(3000, "Carrer Major 31", "7e 4t", "Lleida","Espanya", 758654545);
Localitzacio loc4 = new Localitzacio(4000, "Calle Mejia 12", "bla bla 1", "Girona","Espanya", 14253674);
Localitzacio loc5 = new Localitzacio(5525, "Straufen 122", "123 po.", "Berlin","Alemanya", 123789456);

l.save(loc1);
l.save(loc2);
l.save(loc3);
l.save(loc4);
l.save(loc5);

Persona persona1 = new Persona("23346789P", "Edu", "Val");
Persona persona2 = new Persona("12563413Y", "Bil", "Bil");
Persona persona3 = new Persona("97625151S", "Jos", "Jos");
Persona persona4 = new Persona("01232342Q", "Mar", "Mar");
Persona persona5 = new Persona("63434233M", "Pol", "Pol");

// Creem la variable localitzacions amb un SET, la SET localització P1 té dos localitzacions
Set<Localitzacio> localitzacionsP1 = new LinkedHashSet();
localitzacionsP1.add(loc1); // Té casa a Barcelona
localitzacionsP1.add(loc4); // Té segona residència a Girona

// Afegim aquest SET de localitzacions a la persona 1
persona1.setLocalitzacio(localitzacionsP1);

// Fem els següents casos amb els altres llocs
Set<Localitzacio> localitzacionsP2 = new LinkedHashSet();
localitzacionsP2.add(loc2);
persona2.setLocalitzacio(localitzacionsP2);

Set<Localitzacio> localitzacionsP3 = new LinkedHashSet();
localitzacionsP3.add(loc3);
persona3.setLocalitzacio(localitzacionsP3);

// La persona 4 i 5 viuen junts
Set<Localitzacio> localitzacionsP4 = new LinkedHashSet();
localitzacionsP4.add(loc5);
persona4.setLocalitzacio(localitzacionsP4);
persona5.setLocalitzacio(localitzacionsP4);

// Ara guardem les dades de persones i totes les seves localitzacions
p.save(persona1);
p.save(persona2);
p.save(persona3);
p.save(persona4);
p.save(persona5);
```

```
// Instanciem les gabies amb la seva informació
Gabia gabia1 = new Gabia(1, 1);
Gabia gabia2 = new Gabia(2, 1);
Gabia gabia3 = new Gabia(3, 2);
Gabia gabia4 = new Gabia(4, 2);
Gabia gabia5 = new Gabia(1, 2);

// Guardem els canvis a la base de dades
g.save(gabia1);
g.save(gabia2);
g.save(gabia3);
g.save(gabia4);
g.save(gabia5);

// Instanciem les mascotes amb el seu responsable/amo
Mascota mascota1 = new Mascota(1,1,"Felix",gabia1 , persona1);
Mascota mascota2 = new Mascota(2,1,"Rex",gabia2 , persona3);
Mascota mascota3 = new Mascota(3,2,"T-REX",gabia4 , persona2);
Mascota mascota4 = new Mascota(1,1,"Piolin",gabia5 , persona2);
m.save(mascota1);
m.save(mascota2);
m.save(mascota3);
m.save(mascota4);
```

Procedim a veure la base de dades per comprovar les dades creades:

#### GABIA:

ID	MIDA	PIS
1	1	1
2	2	1
3	3	2
4	4	2
5	1	2

#### LOCALITZACIO:

ID	CARRER	CIUTAT	CODI_POSTAL	INFO_ADICIONAL	PAIS	TELEFON
1	Avinguda Madrid 40	Barcelona	8028	Pis quart, porta dos	Espanya	666666666
2	Passeig de Sant Juan	Barcelona	8029	sense número	Espanya	45957854
3	Carrer Major 31	Lleida	3000	7e 4t	Espanya	758654545
4	Calle Mejia 12	Girona	4000	bla bla 1	Espanya	14253674
5	Straufen 122	Berlin	5525	123 po.	Alemanya	123789456

#### PERSONA:


DNI	COGNOM	NOM
01232342Q	Mar	Mar
12563413Y	Bil	Bil
23346789P	Val	Edu
63434233M	Pol	Pol
97625151S	Jos	Jos

**LOCALITZACIONS\_PERSONES (TAULA MN)**

DNI_PERSONA	ID_LOCALITZACIO
23346789P	1
12563413Y	2
97625151S	3
23346789P	4
01232342Q	5
63434233M	5

**MASCOTA**

ID	ALT	AMPLE	NOM	ID_GABIA	DNI_PERSONA
1	1	1	Felix	1	23346789P
2	1	2	Rex	2	97625151S
3	2	3	T-REX	4	12563413Y
4	1	1	Piolin	5	12563413Y

 <b>INSTITUT AUSIÀS MARCH</b> Consorci d'Educació de Barcelona Generalitat de Catalunya Ajuntament de Barcelona	<b>UF6 – Introducció a la persistència en BBDD</b> <b>Projecte Final</b>	<b>Mòdul 03:</b> <b>Programació</b> <b>Eduard Vallès</b>
--	---	--

## Test Junit

Anem a fer proves amb la utilitat JUNIT, tant per comprovar que les dades existeixen a les taules, com per fer proves de consulta, actualitzacions i esborrat.

El test està ordenat per nom del mètode test ascendent


I té unes variables estàtiques de la classe amb mètodes per tancar i obrir noves sessions:

```
@FixMethodOrder(MethodSorters.NAME_ASCENDING)
public class TestVet {

    private static LocalitzacioDAOImplementacio l = new LocalitzacioDAOImplementacio();
    private static PersonaDAOImplementacio p = new PersonaDAOImplementacio();
    private static MascotaDAOImplementacio m = new MascotaDAOImplementacio();
    private static GabiaDAOImplementacio g = new GabiaDAOImplementacio();

    public static void open(){
        l = new LocalitzacioDAOImplementacio();
        p = new PersonaDAOImplementacio();
        m = new MascotaDAOImplementacio();
        g = new GabiaDAOImplementacio();
    }

    public static void close(){
        l.close();
        p.close();
        m.close();
        g.close();
    }
}
```

 <b>INSTITUT AUSIÀS MARCH</b> Consorci d'Educació de Barcelona Generalitat de Catalunya Ajuntament de Barcelona	<b>UF6 – Introducció a la persistència en BBDD</b> <b>Projecte Final</b>	<b>Mòdul 03:</b> <b>Programació</b> <b>Eduard Vallès</b>
--	---	--

Té un mètode amb anotació `@beforeClass` per realitzar els inserts de forma neta abans de res. Així assegurament que les modificacions no afectin al rellençar el test. Es un mock de carrega de dades.

```
/**
 * Mètode que carrega un cop abans del test per carregar les dades inicials
 */
@BeforeClass
public static void insercions() {

    Localitzacio loc1 = new Localitzacio(8028, "Avinguda Madrid 40", "Pis quart, porta dos", "Barcelona", "Espanya", 666666666);
    Localitzacio loc2 = new Localitzacio(8029, "Passeig de Sant Juan", "sense número", "Barcelona", "Espanya", 45957854);
    Localitzacio loc3 = new Localitzacio(3000, "Carrer Major 31", "7e 4t", "Lleida", "Espanya", 758654545);
    Localitzacio loc4 = new Localitzacio(4000, "Calle Mejia 12", "bla bla 1", "Girona", "Espanya", 14253674);
    Localitzacio loc5 = new Localitzacio(5525, "Straufen 122", "123 po.", "Berlin", "Alemanya", 123789456);

    l.save(loc1);
    l.save(loc2);
    l.save(loc3);
    l.save(loc4);
    l.save(loc5);

    Persona persona1 = new Persona("23346789P", "Edu", "Val");
    Persona persona2 = new Persona("12563413Y", "Bil", "Bil");
    Persona persona3 = new Persona("97625151S", "Jos", "Jos");
    Persona persona4 = new Persona("01232342Q", "Mar", "Mar");
    Persona persona5 = new Persona("63434233M", "Mar", "Pol");

    // Creem la variable localitzacions amb un SET, la SET localització P1 té dos localitzacions
    Set<Localitzacio> localitzacionsP1 = new LinkedHashSet();
    localitzacionsP1.add(loc1); // Té casa a Barcelona
    localitzacionsP1.add(loc4); // Té segona residència a Girona

    // Afegim aquest SET de localitzacions a la persona 1
    persona1.setLocalitzacio(localitzacionsP1);

    // Fem els següents casos amb els altres llocs
    Set<Localitzacio> localitzacionsP2 = new LinkedHashSet();
    localitzacionsP2.add(loc2);
    persona2.setLocalitzacio(localitzacionsP2);

    Set<Localitzacio> localitzacionsP3 = new LinkedHashSet();
    localitzacionsP3.add(loc3);
    persona3.setLocalitzacio(localitzacionsP3);

    // La persona 4 i 5 viuen junts
    Set<Localitzacio> localitzacionsP4 = new LinkedHashSet();
    localitzacionsP4.add(loc5);
    persona4.setLocalitzacio(localitzacionsP4);
    persona5.setLocalitzacio(localitzacionsP4);


    // Ara guardem les dades de persones i totes les seves localitzacions
    p.save(persona1);
    p.save(persona2);
    p.save(persona3);
    p.save(persona4);
    p.save(persona5);

    // Instanciem les gabies amb la seva informació
    Gabia gabia1 = new Gabia(1, 1);
    Gabia gabia2 = new Gabia(2, 1);
    Gabia gabia3 = new Gabia(3, 2);
    Gabia gabia4 = new Gabia(4, 2);
    Gabia gabia5 = new Gabia(1, 2);

    // Guardem els canvis a la base de dades
    g.save(gabia1);
    g.save(gabia2);
    g.save(gabia3);
    g.save(gabia4);
    g.save(gabia5);

    // Instanciem les mascotes amb el seu responsable/amo
    Mascota mascota1 = new Mascota(1,1,"Felix",gabia1 , persona1);
    Mascota mascota2 = new Mascota(2,1,"Rex",gabia2 , persona3);
    Mascota mascota3 = new Mascota(3,2,"T-REX",gabia4 , persona2);
    Mascota mascota4 = new Mascota(1,1,"Piolin",gabia5 , persona2);
    m.save(mascota1);
    m.save(mascota2);
    m.save(mascota3);
    m.save(mascota4);
}
```



 <b>INSTITUT AUSIÀS MARCH</b> Consorci d'Educació de Barcelona Generalitat de Catalunya Ajuntament de Barcelona	<b>UF6 – Introducció a la persistència en BBDD</b> <b>Projecte Final</b>	<b>Mòdul 03:</b> <b>Programació</b> <b>Eduard Vallès</b>
--	---	--

Després tenim 4 mètodes amb anotació @test per les proves de les 4 entitats:

**GABIA:**

```
@Test
public void testA_Gabia() {

    List<Gabia> gabies;

    // Busquem totes les gabies
    // Tenim un total de 5
    gabies = g.findAll();
    Assert.assertEquals(5, gabies.size());

    // Provem a cerca les gabies del pis 2
    // Tenim un total de 3
    gabies = g.findByPis(2);
    Assert.assertEquals(3, gabies.size());

    // Provem a actualitzar la gabia amb ID 5
    // Li possem el pis 1
    g.updateGabia(5, 1, 2);

    // Ara provem de nou a cercar les gabies del pis 2
    // Tenim ara 2
    gabies = g.findByPis(2);
    Assert.assertEquals(2, gabies.size());

    // Esborrem la gabia 3, està desocupada i volem fer manteniment
    g.delete(3);
    // Busquem totes les gabies
    // Amb el esborrat, ara tenim 4
    gabies = g.findAll();
    Assert.assertEquals(4, gabies.size());
}
```

**PERSONA:**

```
@Test
public void testB_Persona() {

    List<Persona> persona;
    Persona personaInd;


    // Busquem totes les persones
    // Tenim un total de 5
    persona = p.findAll();
    Assert.assertEquals(5, persona.size());

    // Provem a cerca persona amb nom i cognom
    // Tenim dues persones que es diuen 'Mar'
    // Però els cognoms dels dos són diferents
    persona = p.findByNameSurname("Mar", "Mar");
    Assert.assertEquals(1, persona.size());

    // Però si ara busquem només pel nom, trobarà 2 Mar
    persona = p.findByName("Mar");
    Assert.assertEquals(2, persona.size());

    // Busquem una persona pel seu DNI
    personaInd = p.findByDni("23346789P");
    // Aquesta persona es diu Edu
    Assert.assertEquals("Edu", personaInd.getNom());

    // Abans d'actualitzar, mirem si existeix Eduard Valles
    persona = p.findByNameSurname("Eduard", "Valles");
    // Esperem 0 persones
    Assert.assertEquals(0, persona.size());
    // Actualitzem les dades d'aquesta persona amb Eduard Valles
    p.updateName(personaInd.getDni(), "Eduard", "Valles");
    // Ara comprovem com existeix un usuari amb aquest nom
    persona = p.findByNameSurname("Eduard", "Valles");
    Assert.assertEquals(1, persona.size());
}
```

 <b>INSTITUT AUSIÀS MARCH</b> Consorci d'Educació de Barcelona Generalitat de Catalunya Ajuntament de Barcelona	<b>UF6 – Introducció a la persistència en BBDD</b> <b>Projecte Final</b>	<b>Mòdul 03:</b> <b>Programació</b> <b>Eduard Vallès</b>
--	---	--

```

// Afegim una nova persona
Persona novaPersona = new Persona("12987452A", "Marcel", "Lopez");
p.save(novaPersona);

// Ara comprovem que hi ha 6 persones en total
persona = p.findAll();
Assert.assertEquals(6, persona.size());

// Esborrem aquesta persona
p.delete(novaPersona.getDni());

// Ara comprovem que hi ha 5 persones de nou
persona = p.findAll();
Assert.assertEquals(5, persona.size());

// Ara anem a veure les localitzacions de la persona
// amb DNI 23346789P, n'ha de tenir 2
personaInd = p.findByDni("23346789P");
Set<Localitzacio> localitzacionsEdu = personaInd.getLocalitzacio();
Assert.assertEquals(2, localitzacionsEdu.size());

// Li afegim una nova localització
Localitzacio newloc = new Localitzacio(18752, "Carrer de la pera", "55 3a", "Malgrat de Mar", "Espanya", 945851254);
// Guardem la localització
l.save(newloc);
// Afegim aquesta localització a la persona
localitzacionsEdu.add(newloc);
// Guardem les dades d'Edu
p.save(personaInd);
// I comprovem que té les 3
personaInd = p.findByDni("23346789P");
localitzacionsEdu = personaInd.getLocalitzacio();
Assert.assertEquals(3, localitzacionsEdu.size());

```

## MASCOTA:

```

@Test
public void testC_Mascota() {

    List<Mascota> mascotes;
    Mascota mascotaInd;

    // Busquem totes les mascotes
    // Tenim un total de 4
    mascotes = m.findAll();
    Assert.assertEquals(4, mascotes.size());


    // Busquem la mascota Felix
    mascotes = m.findByNom("Felix");
    // Tenim una mascota amb aquest nom
    Assert.assertEquals(1, mascotes.size());

    // Busquem les mascotes del client (en té 2)
    mascotes = m.mascotesClientDNI("12563413Y");
    // Comprovem que son 2
    Assert.assertEquals(2, mascotes.size());

    // Anem a buscar les mascotes més amples que 2
    mascotes = m.mesAmpleQue(2);
    // Només n'hi ha 1, que es de 3
    Assert.assertEquals(1, mascotes.size());

    // Anem a buscar les mascotes menys ample que 3
    mascotes = m.menysAmpleQue(3);
    // Només n'hi ha 3
    Assert.assertEquals(3, mascotes.size());
}

```

 <b>INSTITUT AUSIÀS MARCH</b> Consorci d'Educació de Barcelona Generalitat de Catalunya Ajuntament de Barcelona	<b>UF6 – Introducció a la persistència en BBDD</b> <b>Projecte Final</b>	<b>Mòdul 03:</b> <b>Programació</b> <b>Eduard Vallès</b>
--	---	--

```
// Anem a buscar les mascotes mes baixes que 4
mascotes = m.mesBaixQue(4);
// Només n'hi ha 4, totes
Assert.assertEquals(4, mascotes.size());

// Anem a fer un update, una mascota ara té una mida diferent
// I pertany a la persona amb dni 01232342Q
Persona persona = p.findByDni("01232342Q");
// La mascota 3 "T-REX" té una nova alçada de 9 metres i ample de 4
m.update(3, "T-REX", 9, 4, persona);

// Tornem a obtenir les dades després del update
close();
open();

// Tornem a obtenir les dades per aquesta mascota
mascotaInd = m.findById(3);
Assert.assertEquals("T-REX", mascotaInd.getNom());
Assert.assertEquals(4, mascotaInd.getAmple());
Assert.assertEquals(9, mascotaInd.getAlt());
Assert.assertEquals("01232342Q", mascotaInd.getPersona().getDni());
```

## LOCALITZACIO

```
@Test
public void testD_Localitzacio() {

    List<Localitzacio> localitzacions;
    Localitzacio loc;

    // Busquem totes les mascotes
    // Tenim un total de 6
    // Comptant la creada al testB
    localitzacions = l.findAll();
    Assert.assertEquals(6, localitzacions.size());

    //Comprovem, de les localitzacions de Barcelona
    // Que un té el telefon 666666666
    localitzacions = l.findByCiutat("Barcelona");
    List<Localitzacio> expected = new ArrayList<Localitzacio>();

    // Agafem els mateixos objectes per afegir al expected
    Localitzacio loccopy1 = new Localitzacio(8028, "Avinguda Madrid 40", "Pis quart, porta dos", "Barcelona","Espanya", 666666666);
    Localitzacio loccopy2 = new Localitzacio(8029, "Passeig de Sant Juan", "sense número", "Barcelona","Espanya", 45957854);
    expected.add(loccopy1);
    expected.add(loccopy2);

    // Comprovem que tenen el mateix telefon, carrer i codi postal
    assertEquals(localitzacions.get(0).getTelefon(), is(expected.get(0).getTelefon()));
    assertEquals(localitzacions.get(0).getCarrer(), is(expected.get(0).getCarrer()));
    assertEquals(localitzacions.get(0).getCodi_postal(), is(expected.get(0).getCodi_postal()));

    assertEquals(localitzacions.get(1).getTelefon(), is(expected.get(1).getTelefon()));
    assertEquals(localitzacions.get(1).getCarrer(), is(expected.get(1).getCarrer()));
    assertEquals(localitzacions.get(1).getCodi_postal(), is(expected.get(1).getCodi_postal()));

    // Cerquem la localitzacio amb id 4
    loc = l.findById(4);
    // Dades que té Calle Mejia 12 Girona 4000 bla bla 1 Espanya 14253674
    Assert.assertEquals(4000, loc.getCodi_postal());
    Assert.assertEquals("Calle Mejia 12", loc.getCarrer());
    Assert.assertEquals("bla bla 1", loc.getInfo_adicional());

    // Anem a canviar les dades de informació adicional
    l.updateInfo(4, "Client VIP");

    // Tornem a obtenir les dades, primer tanquem sessió
    close();
    open();

    loc = l.findById(4);
    // I comprovem que sigui així
    Assert.assertEquals("Client VIP", loc.getInfo_adicional());
}
```

Test JUNIT en verd:

