Generalitat de Catalunya
Departament d'Ensenyament
INS Provençana

FITXA DEL MÒDUL PROJECTE (DEPARTAMENT D'INFORMÀTICA)

CICLE I GRUP-CLASSE

DAM 2

TÍTOL

Pet Alert

ALUMNES DEL GRUP

Enrique Córdoba Cabo
Luis Gómez Pérez
Tamara Camacho

DESCRIPCIÓ
(Descriure en un màxim de 15 línies els objectius del projecte)

Our project will be a system based in sending and checking alerts on lost pets.
The users will be able to create alerts by presenting a description, an image and specifying the pet location.
They will also check published alerts.

It consists in a mobile client application and a desktop server.

The users will must to identify themselves in order to publish the alerts, unauthenticated users are allowed to check published alerts.

The client applications are connected to a server where the databases are avalaible to answer the queries.

MATERIALS REQUERITS I ESPECIFICACIONS TÈCNIQUES DEL PROJECTE

MySQL
NetBeans
Android Studio
GlassFish Server
Open Street Maps API
JPA API

REQUERIMENTS FUNCIONALS (esborrany) amb la seva prioritat.
(Cal detallar les funcionalitats i especificar els usuaris del producte resultant de cada funcionalitat)

RF1. Registration
- Register a user in the database
- Priority - high

RF2. Login
- The users do have to log to publish alerts
- Priority - high

RF3. Alerts control
- RF3.1. Check an alert
  - The user will be able to check alerts without login in the application
  - Priority - medium
- RF3.2. Create an alert
  - The user will be able to create a new alert. Login required to this action

Priority - high

RF3.3. Search alerts
The user will be able to find the alerts according to specified filtered parameters
Priority - medium

RF3.4. Modify an alert
The user will be able to modify his data alerts
Priority - low

RF3.5. End an alert
The user will be able to mark his alert as resolved
Priority – medium

RF3.6. Inadequate alert report
The user may report the alert if he deems it necessary

RF4.Notifications
RF4.1. Alert notifications – mobile
The user will receive notifications about nearby alerts
Prioriry - medium

RF4.2. Alert notifications – e-mail
The user will receibe an e-mail about nearby alerts

RF5. Messenger service
RF5.1. Send a message
The user will be able to send a message to another user
Priority - medium

RF5.2. Check messages
The user will be able to check received messages
Priority – medium

RF5.3. Delete messages
The user will be able to delete received messages
Priority - low

REQUERIMENTS NO FUNCIONALS

The application will run in an Android based system with an API between 23 and 28 maximum.

- Login/Register

A MySQL database will be used in an Ubuntu Server which will keep the user data, alerts, messages, etc.

- Alert management

The alerts will be kept in a MySQL database.

The alerts will be found according to specified filtered parameters as distance or kind of animal.

The GPS of the mobile device will be used to locate where the alert is created.

Alert data collection from a web service.

Inadequate alert report button, which will be notified to the system administrator.

- Layouts

    Application form layout which allows to add new alerts.

    Application form layout which allows to modify already existing alerts.

    Setting and main window layouts.

- Notifications

    A distance range will be set to receive the alerts.

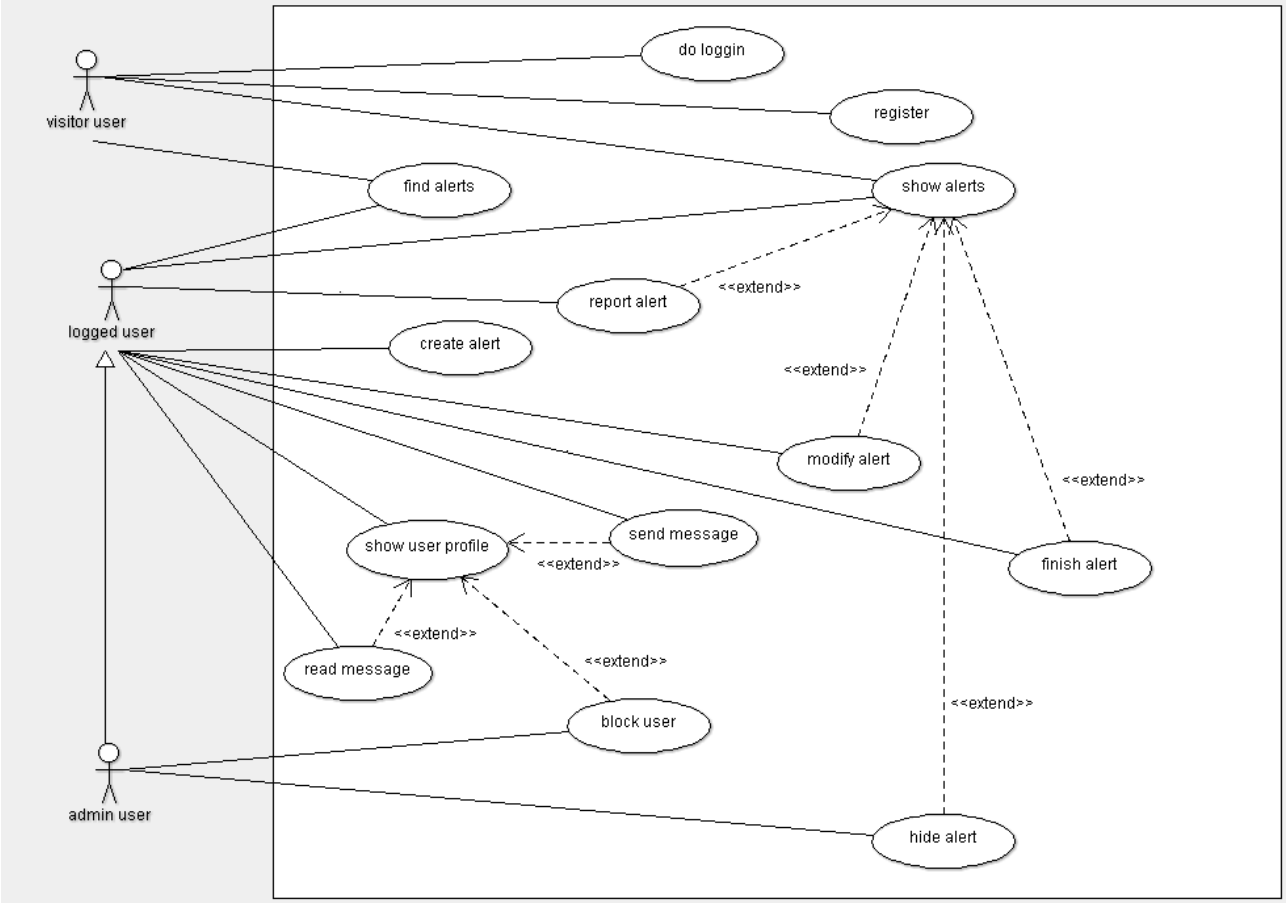
    Notification type selection: the user wants to receive mobile notification alerts or via e-mail.

    Notifications will be enabled or disabled.

- Messenger service

    Internal messenger service system of the application with alert through notification.

Use Case diagram and their textual description



Name:	Do loggin
Actor:	Visitor user
Description:	The user is identified in the system
Normal flow:	1.The actor input his authentification data 2.System recognize the input data
Alternative flow:	1.1.The actor hasn't acount. Can press a button to acces to register form 2.1. Input data is incorrect. The actor try to input another time

Name:	Register
Actor:	Visitor user
Description:	The actor create a new acount in the system
Normal flow:	1. Actor input name and password 2.System save the new user data
Alternative flow:	2.1.Username exist in data, return to step 1 of normal flow

Name:	Show alert
Actor:	Visitor user, logged user, admin user
Description:	System show a layout with an alert
Normal flow:	1. Actor open the alert 2.System show a layout with an alert
Alternative flow:	

Name:	Find alert
Actor:	Visitor user, logged user, admin user
Description	System shows alerts with the parameters indicated by the actor
Normal flow:	1. Actor inpur parameters to find 2.System display a list of alerts
Alternative flow:	2.1. System don't found alerts

Name:	Create alert
Actor:	Logged user, admin user
Description:	Actor create a new alert
Normal flow:	1. Actor input alert data in a form 2.Alert is saved in the system
Alternative flow:	2.1. Data has incorrect values, system ask another time for them

Name:	Modify alert
Actor:	Logged user, admin user
Description:	Actor modify data of alert created for them
Normal flow:	1. Actor input new alert data in a formulario 2.System ask to actor to confirm modification 3.New data saved in the system
Alternative flow:	2.1. Actor doesn't confirm the modify, system return to alert layout 3.1.Data has incorrect values, system ask another time for them

Name:	Finish alert

Actor:	Logged user, admin user
Description:	The alert is marked as finished
Normal flow:	1. Actor mark the alert as finised 2.The change is saved in the system
Alternative flow:	

Name:	Report alert
Actor:	Logged user, admin user
Description:	Actor report an alert with inappropiated content
Normal flow:	1. Actor mark the alert as inappropiated 2.The change is saved in the system
Alternative flow:	

Name:	Show user profile
Actor:	Logged user, admin user
Description:	System show layout with user data
Normal flow:	1. Actor open the profile 2.System show a layout with user data
Alternative flow:	

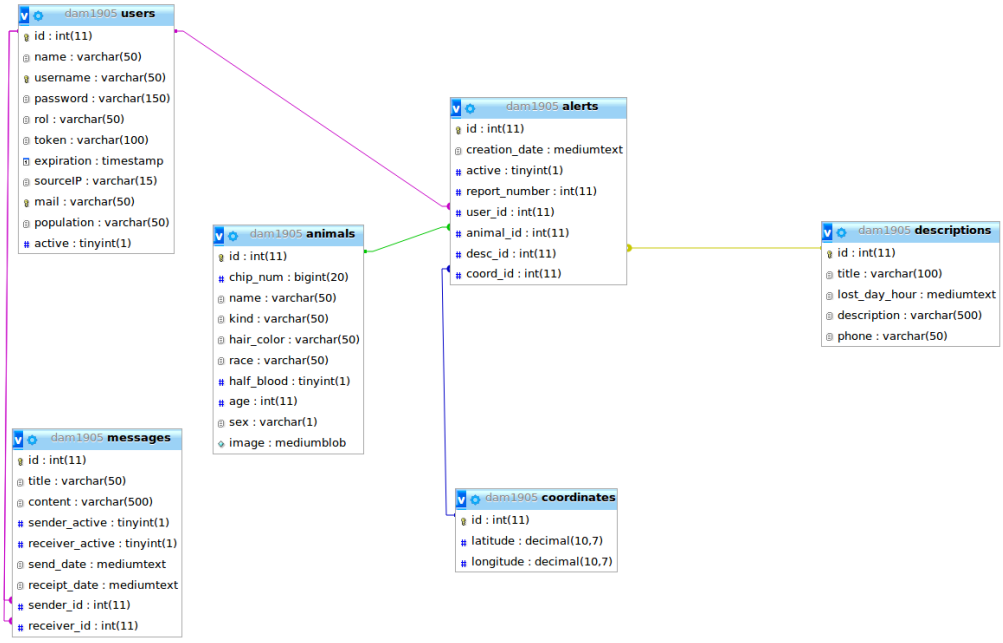
Name:	Send message
Actor:	Logged user, admin user
Description:	Actor send message to another user
Normal flow:	1. Actor select user to send message 2. Actor write the message 3. Actor send the message
Alternative flow:	3.1.The message has't content, return to sep 2 of normal flow

Name:	Read message
Actor:	Logged user, admin user
Description:	Actor read a message
Normal flow:	1. Actor select message 2.System display message content
Alternative flow:	1.1. Actor hasn't messages to select

Name:	Hide alert
Actor:	Admin user
Description:	Actor hide the alert
Normal flow:	1.User mark hide alert option 2.System ask to actor if want to hide 3.Change is saved in system
Alternative flow:	2.1. Actor doesn't confirm the operation, system return to alert layout

Name:	Block user
Actor:	Admin user
Description:	Actor block the user
Normal flow:	1.Actor mark block user option 2.System ask to actor if want to block 3.Change is saved in system
Alternative flow:	2.1.Actor doesn't confirm the operation, system return to user profile layout

Entity-Relation diagram



Relational model

Users {id, username, password, rol, name, mail, population, active}

Messages {id, title, content, sender_active, receiber_active, send_date, receipt_date, send, receipt, receiber_id(FK users), sender_id(FK users)}

Alerts {id, creation_date, active, report_number, user_id(FK users), coordinate_id(FK coordinates), animal_id(FK animals), description_id(FK descriptions)}

Coordinates {id, longitude, latitude}

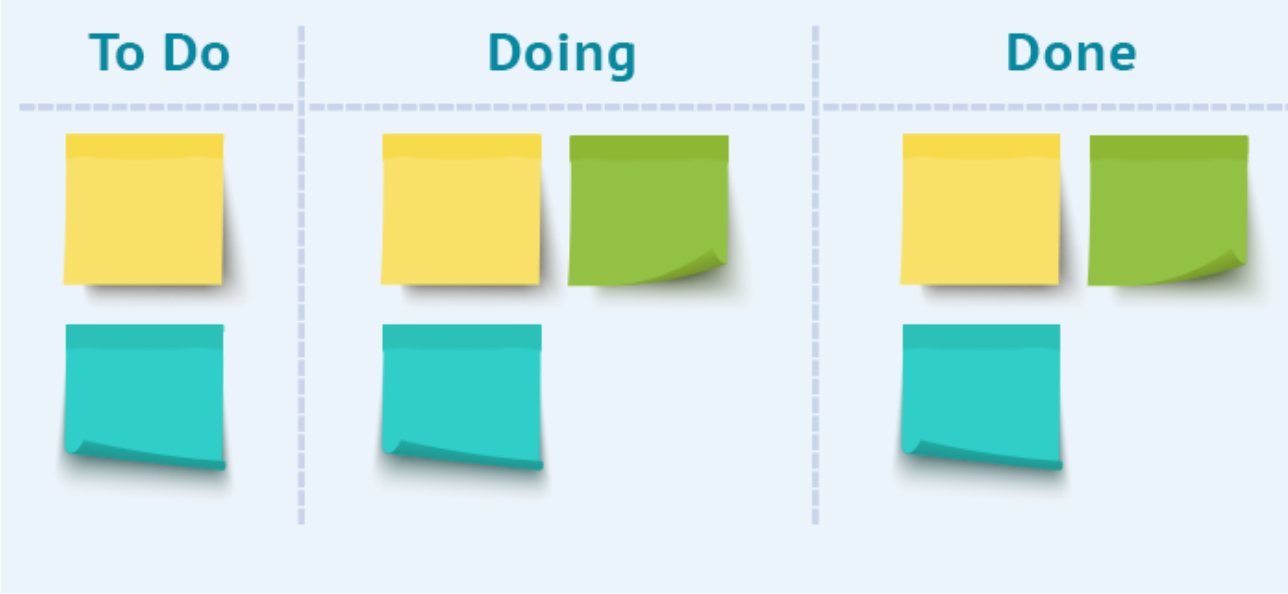
Animals {id, name, kind, race, age, image, hair_color}

Descriptions {id, title, lost_day, lost_hour, description, phone}

Metodogy used

The agile Scrumban methodology will be used.

Simplicity and better adaptation for a new project with little development time, saving time from sprint planning, it is not necessary to have a deliverable from time to time.



Project planning

***Two units of time per day**

Week 1

WEEK 1

TASK	RESOURCE	02/04/2019	03/04/2019
-Server			
database	Luis		
java model	Luis / Tam		
java DAO + BDConnect	Luis / Tam		
service resful	Luis / Tam		
logger, error control			
loggin server			
-App Client			
android main layout	Kike		
android user form layout			
android alert from layout			
android message from layout			
android model			
android OSM API			
android main controller			
android loggin			
bug testing			

Week 2

WEEK 2

TASK	RESOURCE	06/04/2019	07/04/2019	08/04/2019	09/04/2019	10/04/2019
-Server						
database						
java model						
java DAO + BDConnect						
service resful						
logger, error control	Luis					
loggin	Luis / Tam					
-App Client						
android main layout	Kike					
android user form layout						
android alert from layout						
android message from layout						
android model						
android OSM API						
android main controller						
android loggin						
bug testing						

Week 3

WEEK 3

TASK	RESOURCE	15/04/2019	16/04/2019	17/04/2019	18/04/2019	19/04/2019
-Server						
database						
java model						
java DAO + BDConnect						
service resful						
logger, error control						
login server						
-App Client						
android main layout						
android user form layout	Tamara					
android alert from layout	Kike					
android message from layout	Luis					
android model	Kike/ Tam/ Luis					
android OSM API						
android main controller						
android loggin						
bug testing						

Week 4

WEEK 4

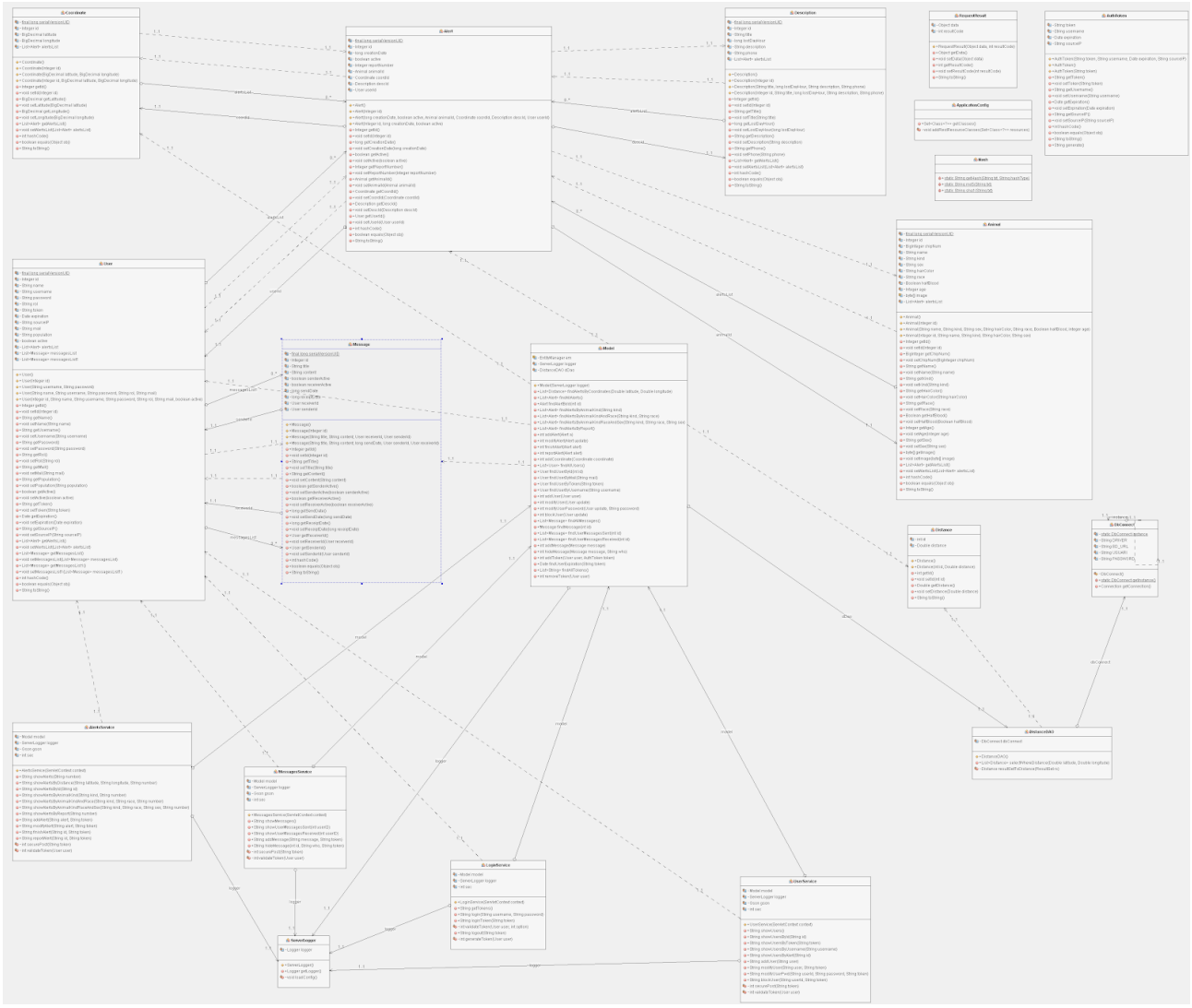
TASK	RESOURCE	22/04/2019	23/04/2019	24/04/2019	25/04/2019	26/04/2019
-Server						
database						
java model						
java DAO + BDConnect						
service resful						
logger, error control						
loggin server						
-App Client						
android main layout						
android user form layout						
android alert from layout						
android message from layout						
android model						
android OSM API	Kike					
android main controller	Kike					
android loggin	Luis / Tam					
bug testing						

Week 5

WEEK 5

TASK	RESOURCE	29/04/2019	30/04/2019	01/05/2019
-Server				
database				
java model				
java DAO + BDConnect				
service resful				
logger, error control				
login server				
-App Client				
android main layout				
android user form layout				
android alert from layout				
android message from layout				
android model				
android OSM API				
android main controller				
android loggin				
bug testing & documentation	Kike / Luis / Tam			

Class diagram java program with web service



Class diagram android program

Generalitat de Catalunya
Departament d'Ensenyament
INS Provençana

Test case

Test Case #: 1.0	**Test Case Name:** Loggin user
System: PetAlert	**Subsystem:** Login
Designed by: PetAlert team	**Design Date:**
Executed by:	**Execution Date:**
Short Description: Test Pet Alert loggin	

Pre-conditions:
User press loggin button
System show loggin layout

Step	Action	Expected System Response	Pass/Fail	Comment
1	User enter his authentification data	System check if input data has correct values		
2	User press loggin button	System show start layout with register complete		
3	Check post-condition 1			
4	Return to step 1 User input incorrect password	System say that password is incorrect		
5	Return to step 1 User input no existing user	System say tha user does't exist		
6	Return to setep 1 User input invaid data	System say that input dara are invalid		

Post-conditions:
1. User account enter in the system

Generalitat de Catalunya
Departament d'Ensenyament
INS Provençana

Test Case #: 2.0	**Test Case Name:** Resgister user
System: PetAlert	**Subsystem:** Register
Designed by: PetAlert team	**Design Date:**
Executed by:	**Execution Date:**
Short Description: Test the Pet Alert register user	

Pre-conditions:
User press register button.
System show register form.

Step	Action	Expected System Response	Pass/ Fail	Comment
1	The user enters data in the form	System will check the entered values are correct		
2	User press register button	System will show a completed registration message		
3	Check post-condition 1			
4	Repeat steps 1, 2 using data already existing in the database	System will show that the user already exist		
5	Check post-condition 2			
6	Repeat step 1, user enter invalid data	System will check the entered values are correct		
7	Check post-condition 3			

Post-conditions:
1.- The new user is saved in the database.
2.- The user already exist and does not allow registration
3.- The values is incorrect and does not allow registration

Test Case #: 3.0	**Test Case Name:** Show alert test
System: PetAlert	**Subsystem:** PetAlert show alert
Designed by: PetAlert team	**Design Date:**
Executed by:	**Execution Date:**
Short Description: Test Pet Alert show alert functionality	

Pre-conditions:
1.User press in an alert

Step	Action	Expected System Response	Pass/Fail	Comment
1	System load the alert			
2	The alert is correctly loaded	System show alert layout		
3	Return to step 1			
4	System can't load the alert	System show message with error		
5	Return to step 1			
6	System loads the alert but doesn't download the image	System show alert layout without image		

Post-conditions:

Test Case #: 4.0	**Test Case Name:** Find alert

System: PetAlert **Subsystem:**

Designed by: PetAlert team **Design Date:**

Executed by: **Execution Date:**

Short Description: Test the Pet Alert find alert

Pre-conditions:
User press filter alert button.
System show filter form.

Step	Action	Expected System Response	Pass/Fail	Comment
1	User enter values in filter form			
2	User press filter button	System will show filtered alerts		
3	Repeat step 1, user enter values that does not correspond with any alert			
4	User press filter button	System will not find matches and show no alerts found message		
5	User press filter button without values	System will show all alerts		

Post-conditions:

Test Case #: 5.0 **Test Case Name:** Create alert test

System: PetAlert **Subsystem:** Pet alert create alert

Designed by: PetAlert team **Design Date:**

Executed by: **Execution Date:**

Short Description: Test of Pet Alert create aler funcionallity

Pre-conditions:
User is logged
System show create alert form

Step	Action	Expected System Response	Pass/Fail	Comment
1	User input correct values in form	System check if values are valid		
2	User press create button	System save data		
3	Return to step 1 User input invalid values	System says that are incorrect values		
4	Return to step 1 User doesn't enter required data	System say that required data is missing		

Post-conditions:
Alert data is saved in database

Test Case #: 6.0 **Test Case Name:** Modify alert test

System: PetAlert **Subsystem:** Pet alert modify alert

Designed by: PetAlert team **Design Date:**

Executed by: **Execution Date:**

Short Description: Test of Pet Alert modify aler funcionallity

Pre-conditions:
User is logged
System show modify alert form

Step	Action	Expected System Response	Pass/Fail	Comment
1	User press modify alert button	System will show modify alert form		
2	User change alert values	System will check values		
3	Check post-contidion 1			
4	Return to step 2 User change alert with incorrect or insufficient values	The system will highlight the incorrect data		
5	Check post-contidion 2			

Post-conditions:
1.- Alert data is saved in database
2.- Alert data will not be modified

Test Case #: 7.0 **Test Case Name:** Finish alert

System: PetAlert **Subsystem:**

Designed by: PetAlert team **Design Date:**

Executed by: **Execution Date:**

Short Description: Test the PetAlert finish alert

Pre-conditions:
The alert has to exist.
User press finish alert alert.

Step	Action	Expected System Response	Pass/Fail	Comment
1	User press finish alert botton	Alert marked as finished		
2	Check post-condition 1			

Post-conditions:
1.-The alert will no longer be in progress

Test Case #: 8.0 **Test Case Name:** Pet Alert report alert test

System: Pet Alert **Subsystem:** Pet Alert report alert

Designed by: Pet Alert team **Design Date:**

Executed by: **Execution Date:**

Short Description: Test of Pet Alert report alert funcionallity

Pre-conditions:
User is logged
System show an alert

Step	Action	Expected System Response	Pass/Fail	Comment
1	User press report button	System register the report		

Post-conditions:
Admin user can see the alert when searching for reported alerts

Test Case #: 9.0 **Test Case Name:** Show user profile

System: PetAlert **Subsystem:**

Designed by: PetAlert team **Design Date:**

Executed by: **Execution Date:**

Short Description: Test the Pet Alert show user profile

Pre-conditions: The user must exist.

Step	Action	Expected System Response	Pass/Fail	Comment
1	User press another user profile	System will show user info form.		

Post-conditions:

Test Case #: 10.0	**Test Case Name:** Pet Alert send message test
System: Pet Alert	**Subsystem:** Pet Alert messages
Designed by: Pet Alert team	**Design Date:**
Executed by:	**Execution Date:**
Short Description: Test of Pet Alert send message funcionallity	

Pre-conditions:
User is logged
System show an user profile

Step	Action	Expected System Response	Pass/Fail	Comment
1	User press send write button	System show form to write message		
2	User write message content			
3	User press send message button	System send the message		
4	Check post-condition 1			
5	Return to step 2 User no write content			
6	User press send message button	System say that can't send a message without content		

Post-conditions:
1.- Message is sended

Test Case #: 11.0 **Test Case Name:** Pet Alert read message test

System: Pet Alert **Subsystem:** Pet read messages

Designed by: Pet Alert team **Design Date:**

Executed by: **Execution Date:**

Short Description: Test of Pet Alert read message funcionallity

Pre-conditions:
User is logged

The user must have messages

Step	Action	Expected System Response	Pass/Fail	Comment
1	User press mailbox button	System will show the messages of user		
2	User select one message	System will show message form with content		
3	Check post-condition 1			
4	Repeat step 1 User has the empty mailbox	System will show no messages to show		

Post-conditions: 1.- Message marked as read

Test Case #: 13.0	**Test Case Name:** Block user
System: PetAlert	**Subsystem:**
Designed by: PetAlert team	**Design Date:**
Executed by:	**Execution Date:**
Short Description: Test the PetAlert block user	

Pre-conditions: Find user. The user must exist.

Step	Action	Expected System Response	Pass/Fail	Comment
1	Admin press block user button			
2	Check post-condition 1			

Post-conditions:
1. - The user will be blocked and will not be able to login.

INSTALLATION GUIDE

Server

Save file PetAlert.war in a folder and open the terminal and execute next command to connect with server:

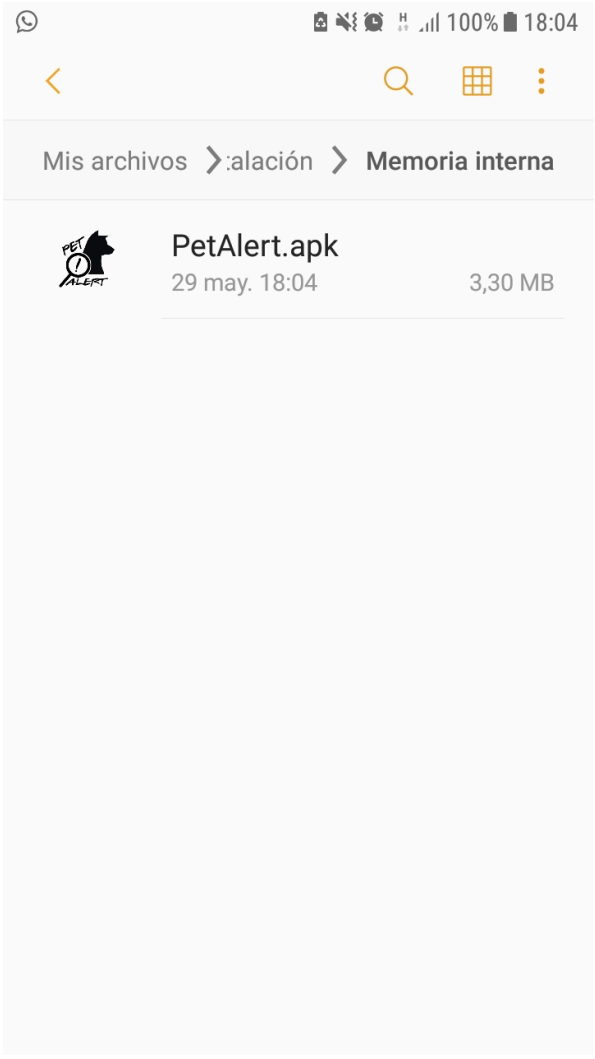
ssh dam1905@apps.proven.cat

scp /home/android/Escriptori/PetAlert/dist/PetAlert.war dam1905@apps.proven.cat:/home/alumnes/dam1905/webapps

Now server is running.

Mobile application

Save apk file in device and execute them.



Now you can use the app