



## MY PET CARE

### SPRINT – 3



Enllaços als  
recursos:

[Project record track](#)  
[Taiga](#)

[Repositoris:](#)

[Android](#)

[Servidor](#)

[Lliberies](#)

Cognom, Nom	Rol	Contacte (@est.fib.upc.edu)	Drive (@gmail.com)	Taiga i GitHub
Campos, Xavier	Developer	xavier.campos.diaz	xavicampos99	xavier-campos
Catalan, Oriol	Developer	oriol.catalan.fernandez	catalan.oriol	oricat8
Clemente, Daniel	Developer	daniel.clemente.marin	daniclemente44	danicm47
Del Rey, Santiago	Developer	santiago.del.rey	santi.delrey	santidrj
Hernando, Enric	Master	enric.hernando	enrichernandopuerta	enrichp
Pinto, Albert	Developer	albert.pinto	apintogil	AlbertPG
Simó, Marc	Developer	marc.simo	sgmarcsg	marcIII
Trius, Álvaro	Developer	alvaro.trius	alvarotrius94	AlvarTB

**Assignatura:** Projecte d'Enginyeria del Software

**Professor:** Silverio Martínez

**Curs:** 2019-2020 QP

**Facultat d'Informàtica de Barcelona**

**Universitat Politècnica de Catalunya**

## ÍNDIX DE CONTINGUTS

1. Introducció .....	1
1.1. Resum executiu del sprint .....	1
1.2. Breu descripció del treball individual (Actualitzat) .....	2
1.2.1. Albert Pinto i Gil .....	2
1.2.2. Álvaro Trius Béjar .....	2
1.2.3. Daniel Clemente Marín .....	2
1.2.4. Enric Hernando Puerta .....	3
1.2.5. Marc Simó Guzmán .....	3
1.2.6. Oriol Catalán Fernández.....	4
1.2.7. Santiago Del Rey Juárez .....	4
1.2.8. Xavier Campos Díaz.....	4
1.3. Avaluació dels companys.....	5
1.4. Sprint master report (Nou).....	6
2. Requisits (Actualitzat).....	8
2.1. Product Backlog.....	8
2.1.1. Gestió de mascotes i usuaris .....	8
2.1.2. Salut de la mascota (Actualitzat) .....	9
2.1.3. Assistència a l'exercici (Actualitzat) .....	9
2.1.4. Establiments especialitzats .....	10
2.1.5. Interacció en comunitats temàtiques (Actualitzat) .....	10
2.1.6. Interacció amb xarxes socials (Actualitzat) .....	11
2.1.7. Interacció amb el calendari.....	12
2.1.8. Gestió de medalles i recompenses (Actualitzat).....	12
2.1.9. Aspectes tecnològics .....	12
2.2. Requisits no funcionals.....	14

---

2.2.1. Rapidesa .....	14
2.2.2. Disponibilitat.....	15
2.2.3. Confidencialitat .....	15
2.2.4. Abstracció i Reutilització (Actualitzat) .....	16
2.3. Tractament d'aspectes transversals (Actualitzat).....	17
2.4. Serveis de tercers (Actualitzat) .....	19
2.4.1. Firebase .....	19
2.4.2. Google Calendar.....	19
2.4.3. OAuth2 .....	20
2.4.4. Google Maps (Nou).....	20
3. Cerimònies àgils .....	22
3.1. Sprint Planning (Nou) .....	22
3.2. Sprint review (Nou).....	23
3.2.1. Què s'ha pogut entregar? .....	23
3.2.2. Què no s'ha pogut entregar? .....	23
3.2.3. Per quin motiu? .....	24
3.3. Sprint retrospective.....	24
3.3.1. Reunió General .....	24
3.3.2. Reunió Front .....	25
3.3.3. Reunió Back.....	25
3.3.4. Què s'ha fet bé i es mantindrà? .....	25
3.3.5. Que canviarem per millorar? (nova divisió) .....	26
3.4. Gràfiques (Actualitzat) .....	27
3.4.1. Sprint burndown.....	27
3.4.2. Release burndown .....	27
3.4.3. Velocitat de l'equip.....	28
3.4.4. Dedicació mensual.....	28

---

4. Metodologia .....	29
4.1. Metodologia àgil.....	29
4.2. Definició de finalització .....	30
4.3. Gestió del projecte (actualitzat).....	30
4.3.1. Taiga (nova divisió).....	30
4.3.2. Sistema de control de versions (nova divisió) .....	33
4.3.3. Project record track (nova divisió) .....	38
4.4. Convencions .....	39
4.5. Gitflow (actualitzat) .....	41
4.6. Testeig (actualitzat) .....	43
4.7. GitHub Actions (actualitzat).....	44
4.8. Dependabot (actualitzat) .....	45
4.9. GitGuardian (actualitzat) .....	46
4.10. Qualitat del codi (actualitzat).....	47
4.11. Entorns integrats de desenvolupament .....	48
4.11.1. Android Studio .....	48
4.11.2. IntelliJ IDEA.....	49
4.12. Comunicació (actualitzat) .....	49
4.13. Gestió de bugs (Actualitzat) .....	51
4.14. JitPack (Nou) .....	52
5. Descripció tècnica .....	53
5.1. Concepció general de l'arquitectura .....	53
5.2. Diagrames arquitectònics (Actualitzat) .....	54
5.3. Patrons aplicats .....	57
5.3.1. Factoria abstracta .....	57
5.3.2. Factoria simple.....	58
5.3.3. Adaptador.....	58

5.3.4. Model, vista, controlador.....	59
5.3.5. DAO.....	59
5.3.6. Service Locator .....	60
5.3.7. Singletó .....	61
5.3.8. Expert .....	62
5.3.9. Controlador de transacció .....	62
5.3.10. Patró Plantilla.....	63
5.3.11. Patró estratègia.....	64
5.3.12. Patró observador .....	64
5.4. Model conceptual de les dades (Actualitzat) .....	66
5.5. Altres aspectes tecnològics.....	68
5.5.1. Servidors .....	68
5.5.2. Base de dades .....	68
5.5.3. Nombre de llenguatges.....	69
5.5.4. APIs .....	69
5.5.5. Frameworks .....	69
5.5.7. Desplegament.....	70

## ÍNDIX DE FIGURES

Figura 1. Sprint burndown de la tercera iteració .....	27
Figura 2. Release burndown .....	27
Figura 3. Representació de la velocitat de desenvolupament de l'equip .....	28
Figura 4. Hores de dedicació mensual del membres de l'equip .....	28
Figura 5. Detall de les èpiques del projecte .....	31
Figura 6. Exemple d'una història d'usuari del Taiga .....	31

Figura 7. Product backlog en el Taiga .....	32
Figura 8. Taskboard del sprint 3 .....	32
Figura 9. Informació d'una tasca.....	33
Figura 10. Organització de GitHub del projecte .....	34
Figura 11. Pàgina principal del repositori de l'aplicació .....	34
Figura 12. Pull request del servei web.....	35
Figura 13. Exemple d'un canvi requerit en un pull request .....	36
Figura 14. Exemple d'un pull request bloquejat .....	36
Figura 15. Pull request tracker en el servei web .....	37
Figura 16. Releases del repositori de l'aplicació .....	37
Figura 17. Fragment del project record track .....	38
Figura 18. Tipus de branques del Gitflow .....	42
Figura 19. Evolució de la cobertura d'una de les branques .....	43
Figura 20. GitHub actions en un commit .....	44
Figura 21. Pull request iniciat pel Dependabot.....	45
Figura 22. Correu electrònic de GitGuardian.....	46
Figura 23. Issues detectades pel Codacy en un commit .....	48
Figura 24. Missatges del GitHub en un dels canals de slack .....	50
Figura 25. Issues tancades en l'aplicació .....	51
Figura 26. Read me de la llibreria amb la informació del JitPack.....	52
Figura 27. Arquitectura Enterprise Service Bus .....	54
Figura 28. Representació de la capa física (Original) .....	55
Figura 29. Representació de la capa física (Nova versió) .....	55
Figura 30. Diagrama de components (Versió sprint 2) .....	56
Figura 31. Diagrama de components (Nova versió).....	56
Figura 32. Exemple del disseny UML d'una factoria abstracta.....	57
Figura 33. Exemple del disseny UML d'una factoria simple .....	58
Figura 34. Exemple del disseny UML d'un adaptador.....	58
Figura 35. Exemple del disseny UML del MVC.....	59
Figura 36. Exemple del disseny UML del DAO .....	60
Figura 37. Exemple del disseny UML del Service Locator .....	61
Figura 38. Representació d'un singletó en UML .....	61
Figura 39. Exemple d'aplicació del patró expert.....	62
Figura 40. Exemple del disseny UML del controlador de transacció.....	62

Figura 41. Patró Plantilla .....	63
Figura 42. Patró estratègia.....	64
Figura 43. Patró observador .....	65
Figura 44. Model de dades final (Original) .....	66
Figura 45. Diagrama de classes final (Versió sprint 2) .....	66
Figura 46. Diagrama de classes final (Nova versió).....	67
Figura 47. Diagrama de classes actual (Original) .....	67
Figura 48. Diagrama de classes actual (Versió sprint 2) .....	67
Figura 49. Diagrama de classes actual (Nova versió) .....	68

## ÍNDEX DE TAULES

Taula 1. Valoracions entre companys .....	5
Taula 2. Històries d'usuari del tema: Gestió de mascotes i usuaris .....	8
Taula 3. Històries d'usuari del tema: Salut de la mascota .....	9
Taula 4. Històries d'usuari del tema: Assistència a l'exercici.....	9
Taula 5. Històries d'usuari del tema: Establiments especialitzats .....	10
Taula 6. Històries d'usuari del tema: Interacció en comunitats temàtiques .....	10
Taula 7. Històries d'usuari del tema: Interacció amb xarxes socials .....	11
Taula 8. Històries d'usuari del tema: Interacció amb el calendari .....	12
Taula 9. Històries d'usuari del tema: Gestió de medalles i recompenses .....	12
Taula 10. Històries d'usuari del tema: Aspectes tecnològics .....	12
Taula 11. Històries d'usuari del tema: Requisits no funcionals.....	13
Taula 12. Aspectes Transversals.....	18

## 1. Introducció

En aquest apartat farem una introducció a la tercera iteració del desenvolupament del projecte My Pet Care.

### 1.1. Resum executiu del sprint

En aquest tercer *sprint* del projecte, hem establert uns objectius els quals han estat negociats amb el *product owner*. Aquests consisteixen en la implementació de els grups i foros de les comunitats temàtiques, la geolocalització amb i sense mapa, a més de compartir el contingut de l'aplicació i alguns CRUD més d'informació de la mascota.

Per tal d'assolir aquests objectius, l'equip de desenvolupament ha decidit mantenir els mateixos equips que en la iteració anterior, tot i que s'ha decidit implementar alguns canvis en la manera sobre com es tracten les històries d'usuari en cadascun. Per una banda, en l'equip de *front*, hem decidit que en parelles es realitzaran de forma sincronitzada diverses històries d'usuari amb algun parentesc. D'aquesta manera, en el cas que es detecti alguna necessitat conjunta en la interfície, per exemple, els dos membres es posen d'acord i implementen conjuntament aquesta part. Per l'altre banda, en l'equip de *back*, en lloc de tenir dos equips per implementar la llibreria i el servidor, cada desenvolupador ha d'implementar la història d'usuari des del servidor fins la llibreria. D'aquesta manera no s'endarrereix la implementació i es pot començar a realitzar la comunicació amb el *front*.

A la segona setmana del sprint, hem hagut d'afegir els 3 punts de la història d'usuari del CRUD de les medalles, degut a que la càrrega de treball en back era força menor que la que hi havia a front.

Malgrat tot, al final hem aconseguit finalitzar quasi totes les tasques i les restants seran traslladades al principi de la pròxima iteració.



## **1.2. Breu descripció del treball individual (Actualitzat)**

### **1.2.1. Albert Pinto i Gil**

En aquesta iteració, per una banda m'he dedicat a avançar amb les històries d'usuari de la xarxa social que ens van quedar de la iteració anterior, junt amb el Xavier Campos. En particular, han estat les històries de donar like a un post, reportar posts, afegir imatges als grups i les notificacions de nous missatges. Per l'altre banda, he realitzat la implementació de l'exercici de les mascotes i els passejos, incloent la geolocalització a temps real amb el GPS i el mapa. Aquestes tasques tenien una càrrega de treball elevada i, per aquest motiu, vaig comptar amb l'ajuda del Daniel Clemente.

### **1.2.2. Álvaro Trius Béjar**

En aquesta iteración m'he encarregat d'implementar un sistema al servidor i a les llibreries que guardi les dates i les hores en format utc però que es mostrin a l'usuari en format local. A més, he col·laborat en tasques de documentació i de revisió de codi. Per últim, he treballat amb l'Oriol Catalan en dissenyar i implementar el CRUD de les medalles. Aquest últim treball no era un objectiu d'aquesta iteració però, veient que la càrrega de treball ens ha tocat ha estat reduïda, vam decidir avançar treball per a la següent iteració.

### **1.2.3. Daniel Clemente Marín**

En aquesta iteració, en el qual he tornat a formar part de l'equip de front, he treballat en dos aspectes ben diferents del front-end de la nostra aplicació. Durant la primera setmana vaig treballar de manera conjunta amb el Enric Hernando per tal de finalitzar les històries que ens havien quedat pendents de la integració del Calendari de Google. Durant la resta de la iteració, he implementat amb el meu company Albert la part dels passeigs i la geolocalització de l'aplicació. També vaig ajudar la primera setmana a terminar la integració de Google Calendar.

#### 1.2.4. Enric Hernando Puerta

En aquesta iteració, en el qual he tornat a formar part de l'equip de front, he treballat en dos aspectes ben diferents del front-end de la nostra aplicació. Durant la primera setmana vaig treballar de manera conjunta amb el Daniel Clemente per tal de finalitzar les històries que ens havien quedat pendents de la integració del Calendari de Google, i vaig realitzar el CRUD de rentats. Durant la resta de la iteració he treballat amb el Xavier Campos per tal de finalitzar una altra col·lecció (CRUD) de la mascota, en aquest cas el Perfil Mèdic, el qual està format per dues parts, la part de les Vacunes i la part de les Malalties. Per tal de poder realitzar aquesta nova col·lecció, jo em vaig encarregar de dissenyar tota la interfície i algunes transaccions de la part de Malalties, mentre que en Xavier va ser el responsable de totes les transaccions de Vacunes, i algunes de Malalties, així i com la comunicació amb el servidor, mitjançant la llibreria de les transaccions mencionades anteriorment. A més, al llarg de tota la iteració, he estat donant suport als meus companys quan sorgien dubtes sobre com realitzar certes tasques, i com a *sprint master*, he distribuït i organitzat la feina entre els meus companys per tal de complir amb el termini d'entrega i m'he encarregat de garantir el compliment de les cerimònies àgils pròpies d'un *sprint*.

#### 1.2.5. Marc Simó Guzmán

En aquesta iteració m'he centrat, tant en la llibreria com en el servidor, en refactoritzar la implementació de Mascota i les seves subcoleccions, d'afegir noves subcoleccions i d'implementar noves funcionalitats d'aquestes. Concretament, per una banda he refactoritzat les subcoleccions de Mascota que ja existien (Medication, Weight, Washes...) i la pròpia implementació de Mascota per tal de que sigui més escalable, i per tant que l'addició de noves subcoleccions sigui més senzilla, per l'altra banda, he afegit a Mascota les subcoleccions Exercicis i Passeigs, Rentats, Perfil mèdic i Visites al veterinari, i els seus corresponents CRUDS; i finalment, he implementat l'eliminació de registres més antics d'exercicis periòdicament en back. A més, al llarg de tota la iteració, he estat donant suport als meus companys quan era necessari.

### **1.2.6. Oriol Catalán Fernández**

En aquesta iteració, on he estat al equip de back end, vaig començar a treballar en la part de compartir la aplicació a través de les xarxes socials. Aquesta feina, quan encara estava en el començament, em van comentar des de front que seria molt més facil d'implementar allà, i per tant, vaig deixar de fer aquestes tasques. En el moment que em van comunicar això, em vaig posar a treballar, junt amb el Álvaro Trius, en el tema de les medalles. Vam proposar una idea de implementació al equip, vam acabar de polir la idea entre tots i ens vam posar a desenvolupar-la. Com que el Álvaro tenia feina amb el tema de la conversió de hores, jo vaig començar en la part de WebService de les medalles, mentre que ell es va encarregar d'implementar la part de les llibreries.

### **1.2.7. Santiago Del Rey Juárez**

En aquesta iteració he estat l'encarregat d'implementar la part del back-end de totes les històries que tenen a veure amb la xarxa social pròpia de l'aplicació, és a dir, l'enviament de notificacions cap als usuaris, la possibilitat de poder crear un post amb una imatge, el poder guardar i canviar la icona d'un grup o el poder donar a like a un post. A més a més, al llarg de la primera setmana vaig estar treballant en la creació un conjunt d'adaptadors per poder accedir de manera més fàcil cap a la base de dades.

### **1.2.8. Xavier Campos Díaz**

En aquest tercer sprint, en el qual he tornat a formar part de l'equip de front, he treballat en dos aspectes ben diferents del front-end de la nostra aplicació. Durant la primera setmana vaig treballar de manera conjunta amb l'Albert Pinto per tal de finalitzar les històries que ens havien quedat pendents de la Xarxa Social.

En canvi, durant la resta de la iteració he treballat amb l'Enric Hernando per tal de finalitzar una altre col·lecció (CRUD) de la mascota, en aquest cas el Perfil Mèdic, el qual està format per dues part, la part de les Vacunes i la part de les Malalties. Per tal de poder realitzar aquesta nova col·lecció, l'Enric es va encarregar de dissenyar tota la interfície i algunes transaccions de la part de Malalties, mentre que jo vaig ser el

responsable de totes les transaccions de Vacunes, i algunes de Malalties, així i com la comunicació amb el servidor, mitjançant la llibreria de les transaccions mencionades anteriorment.

### 1.3. Avaluació dels companys

Les valoracions s'han realitzat mitjançant una enquesta realitzada amb Google Forms. Per tal de no condicionar aquestes, els membres no han tingut accés a les valoracions ja realitzades i s'han realitzat de manera anònima.

Tots els membres han hagut de valorar als seus companys en una escala de l'1 al 10, on el primer significa que ha tingut una actitud molt negativa envers el treball i l'altre que ha sigut productiu i ha tingut una bona actitud durant el desenvolupament de la iteració. Per tal de realitzar la mitjana de les valoracions de forma correcta, cada membre s'ha hagut de votar a si mateix amb una valoració de 0. Per tant, aquest valor no indica que no s'hagi treballat en la iteració sinó que no es pot valorar a si mateix.

Un cop tancat el període de valoracions, concretament el dia abans de l'entrega d'aquest document, s'han obtingut les mitjanes de totes les notes i s'han fet públiques a tot l'equip.

	<b>Sprint 1</b>	<b>Sprint 2</b>	<b>Sprint 3</b>	<b>Sprint 4</b>	<b>Mitjana</b>
<b>Albert Pinto</b>	9.29	9.14	8.43		8.95
<b>Álvaro Trius</b>	6.71	6.00	6.14		6.28
<b>Daniel Clemente</b>	7.71	5.57	6.86		6.71
<b>Enric Hernando</b>	8.57	8.43	8.14		8.38
<b>Marc Simó</b>	9.00	9.00	9.00		9.00
<b>Oriol Catalán</b>	7.14	6.71	6.57		6.80
<b>Santiago Del Rey</b>	9.57	9.00	8.57		9.05
<b>Xavier Campos</b>	8.71	8.43	8.57		8.57
<b>Mitjana</b>	8.34	7.79	7.79		7.97

**Taula 1. Valoracions entre companys**

#### 1.4. Sprint master report (Nou)

En aquest tercer *sprint* hem continuat amb el desenvolupament del nostre projecte. Tal i com es va acordar amb el *product owner*, en aquest *sprint* hem acabat d'implementar les funcionalitats que no va donar temps d'acabar en l'*sprint* anterior, les quals són donar i treure like a un post, reportar un post i bloqueig d'usuaris, rebre notificacions de grup, compartir fotos als Fòrums i la comunicació amb el calendari de Google Calendar de l'usuari, i hem implementat les noves funcionalitats de realitzar passeig, consultar ruta de passeig, eliminar ruta de passeig, eliminar els registre més antics, compartir l'aplicació i els CRUD d'exercici realitzat, el de rentats, el de perfil mèdic, el de visita veterinari i el de medalles.

El primer dia, durant l'*sprint planning*, vam decidir continuar amb la distribució del treball seguida en l'anterior iteració, on quatre membres del equip es dediquen a implementar la part del front-end i els altres quatre el *back-end*, ja que és una distribució que ens va funcionar durant el primer *sprint* i ens permet treballar en paral·lel aprofitant més el temps.

Aquest mateix dia i seguint els passos del primer *sprint* i segon *sprint*, es va repartir de manera equitativa la documentació a realitzar per aquest *sprint*, i per tal de tenir-la acabada per la data d'entrega, però sense que suposés massa càrrega de treball de cop, es van dividir els punts a fer en dos conjunts. Així doncs, el primer conjunt de documentació es va realitzar al llarg de la primera i segona setmana i el segon durant la tercera setmana, ja que requeria que l'*sprint* estiguera més avançat; de manera que la documentació quedés completa abans d'acabar la iteració, això si, sense comptar les cerimònies de l'*sprint* que estrictament s'han de realitzar l'últim dia de la iteració, com són l'*sprint review* i *retrospective*.

Durant aquestes tres setmanes, i per seguir traient-li avantatge a la situació de confinament en la que ens trobem, hem seguit amb la realització de reunions periòdiques mitjançant videotrucades, les quals s'han realitzat en dilluns, dimecres i divendres, excepte casos excepcionals, i en les quals hem simulat un *stand up*, per posar-nos al dia de l'estat de la feina de cada un dels membres. Acte seguit, si no s'havia de discutir cap punt del projecte, ens hem estat dividint en els dos equips prèviament mencionats (*front-end* i *back-end*) per treballar en les nostres respectives parts.

En general s'ha realitzat un bon treball al llarg d'aquest *sprint*, en aquesta tercera iteració hem tingut en compte els problemes sorgits durant les iteracions anteriors, i hem millorat la distribució de càrrega al llarg de *l'sprint*, de manera que és vagin iniciant i completant les tasques al llarg de *l'sprint* i no iniciar moltes històries a l'inici i acabar-les totes al final i amb presa degut a problemes de compatibilitat. A més, no hem tingut el problema de la segona iteració de que algunes funcionalitats s'haguin tingut que adaptar posteriorment a la seva finalització per falta de compatibilitat entre front i *back*.

Però, tot i que el volum de càrrega de treball ha estat més distribuït al llarg de *l'sprint*, han sorgit diversos inconvenients al llarg de *l'sprint*, sent el més important la necessitat d'incloure, a la segona setmana, més històries d'usuari a les plantejades originalment durant *l'sprint planning*, ja que que la càrrega de treball en *back* era força menor que la que hi havia a front. Concretament, hem hagut d'afegir el CRUD de medalles, una històries de 3 punts.

Tot i no haver pogut completar totes les històries d'usuari, hem completat un 90% dels punts de les històries de *l'sprint*, quedant per fer 2 històries d'usuari amb un total de 5 punts (un 10% del total de punts), la qual cosa tenint en compte que hem hagut d'afegir una tasca que representa un 6% del total de punts de *l'sprint* en la segona setmana de *l'sprint*, i que en veritat hem complert l'objectiu fixat, acabar-la en *back*, donen a lloc al fet que em sembli un molt bon resultat. Per aquests motius, no considero que la no finalització del 100% de *l'sprint* sigui un motiu d'alarma, ja que tècnicament hem realitzat totes les tasques assignades menys una, i assolit un 78% del total de punts del projecte, superant el 75% ideal de punts assolits al final del tercer *sprint*.






Per últim, m'agradaria afegir que l'últim dia vam realitzar un *sprint* final per intentar realitzar totes les tasques assignades en comptes de deixar alguna fora, la qual cosa va resultar "*catastròfica*" a la demo de la presentació, ja que no vam aconseguir fer el merge de la branca de fòrums a temps i va donar errors a la demo. Prenem aquesta situació com a exemple, perquè en futures entregues no ens torni a passar. En definitiva, val més deixar alguna funcionalitat fora, que presentar un producte no testejat.

## 2. Requisits (Actualitzat)

En aquest apartat es mostren els canvis del *product backlog*, es dona una breu explicació dels requeriments no funcionals i de les aplicacions de tercers que es troben incloses al projecte.

### 2.1. Product Backlog

A continuació es mostren les modificacions de les històries d'usuari agrupades per tema. Els canvis sobre el *product backlog* queden indicats amb els colors següents:

-  Històries d'usuari assignades al primer sprint.
-  Històries d'usuari assignades al segon sprint.
-  Històries d'usuari assignades al tercer sprint.
-  Històries d'usuari assignades al segon i al tercer sprint.
-  Històries d'usuari assignades a tots els sprints.

#### 2.1.1. Gestió de mascotes i usuaris

Gestió del compte de l'usuari	Gestió de les dades de l'usuari	Gestió de les mascotes de l'usuari	Gestió de la informació general de la mascota
Alta usuari	Modificar Nom d'usuari	Alta mascota	Modificar Nom Mascota
Baixa usuari	Modificar Correu Electrònic	Baixa Mascota	Modificar Sexe Mascota
Modificar idioma del sistema	Modificar Contrasenya	Consulta Mascotes	Modificar Raça Mascota
	CRUD Imatge de Perfil		Modificar Data de Naixement
			CRUD imatge de perfil per a la mascota

Taula 2. Històries d'usuari del tema: Gestió de mascotes i usuaris

**2.1.2. Salut de la mascota (Actualitzat)**

Informació bàsica	Alimentació:	Higiene:	Veterinari:
Afegir pes.	CRUD apat	Afegir rentat.	Consultar historial mèdic.
CRUD Dades salut mascota	Calcular kcal aconsellades.	Consultar pròxim rentat.	CRUD visita veterinari
CRUD perfil mèdic mascota	Consultar kcal Consumides.	Consultar historial de rentats.	CRUD medicació

**Taula 3. Històries d'usuari del tema: Salut de la mascota****2.1.3. Assistència a l'exercici (Actualitzat)**

Gestió de les rutes de passeig:	Control de l'exercici realitzat:
Realitzar passeig.	Gestionar l'exercici realitzat.
Consultar ruta de passeig.	Compartir exercici realitzat.
Eliminar ruta de passeig.	Eliminar els registres més antics periòdicament.
Compartir ruta de passeig.	

**Taula 4. Històries d'usuari del tema: Assistència a l'exercici**

La història d'eliminar els registres més antics periòdicament està implementada, però hem decidit com equip deixar-la fora perquè no veiem que tingui molta utilitat en el nostre projecte.



**2.1.4. Establiments especialitzats**

<b>Localització d'establiments especialitzats</b>	<b>Valoració d'establiments especialitzats</b>
Veure establiments propers	CRUD valoració
Filtrar la visualització dels establiments	
Mostrar la ruta més ràpida	

**Taula 5. Històries d'usuari del tema: Establiments especialitzats****2.1.5. Interacció en comunitats temàtiques (Actualitzat)**

<b>Gestió de continguts:</b>	<b>Gestió de grups:</b>
Pujar, veure, comentar i eliminar contingut/fòrum (CRUD).	Crear/Esborrar/Editar grup (CRUD).
Donar like/dislike al contingut/fòrum.	Subscriure's al grup.
Denunciar contingut/fòrum.	Desubscriure's al grup.
	Rebre notificacions del grup

**Taula 6. Històries d'usuari del tema: Interacció en comunitats temàtiques**

### 2.1.6. Interacció amb xarxes socials (Actualitzat)

En l'Sprint planning del tercer Sprint vam decidir eliminar la Èpica Instagram de les nostres xarxes socials degut a la falta de relació amb la nostra aplicació. A més, hem afegit la Èpica Google ja que és necessari iniciar sessió i tancar sessió amb Google per utilitzar algunes funcionalitats de l'aplicació com és interactuar amb el calendari (Google Calendar) i mapes (Google Maps).

Facebook:	Twitter:	Google:	WhatsApp:
Iniciar sessió.	Iniciar sessió.	Iniciar sessió.	Compartir fotos de la galeria.
Tancar sessió.	Tancar sessió.	Tancar sessió.	Compartir aplicació.
Compartir fotos de la galeria.	Compartir fotos de la galeria.		Compartir publicacions de fòrums.
Compartir aplicació.	Compartir aplicació.		Compartir informació de la teva mascota.
Compartir publicacions de fòrums.	Compartir publicacions de fòrums.		
Compartir informació de la teva mascota.	Compartir informació de la teva mascota.		

Taula 7. Històries d'usuari del tema: Interacció amb xarxes socials

**2.1.7. Interacció amb el calendari**

Avisos personals	Avisos automatitzats	Notificacions a l'usuari
CRUD avís personal	CRUD avís generat	CRUD notificació periòdica

Taula 8. Històries d'usuari del tema: Interacció amb el calendari

**2.1.8. Gestió de medalles i recompenses (Actualitzat)**

Control del progrés de les medalles:	Control medalles disponibles:
Consultar requisits i progrés d'una medalla.	CRUD medalla.
Actualitzar progrés de les medalles.	

Taula 9. Històries d'usuari del tema: Gestió de medalles i recompenses

**2.1.9. Aspectes tecnològics**

Aquest tema el vam afegir al principi del primer *sprint* ja que vam considerar que era necessari considerar les històries d'usuari que conté.

Aspectes relacionats amb les tecnologies utilitzades
Navegació entre diferents funcionalitats
CRUD Firebase
Creació API Gestió d'usuaris
Menú de settings
Menú principal
Preparació accés als serveis

Taula 10. Històries d'usuari del tema: Aspectes tecnològics

### 2.1.10. Requisits Funcionals (Nou)

Aquest tema el vam afegir durant el primer sprint ja que tot i que el cost assignat a cada història d'usuari és de 0, és necessari tenir en compte tots els requisits no funcionals, ja que s'han de seguir durant tot el procés de creació de l'aplicació. Per aquesta raó aquestes històries és mantindran en tots els sprints.

Requisits no funcionals
Rapidesa
Disponibilitat
Confidencialitat
Abstracció i Reutilització

**Taula 11. Històries d'usuari del tema: Requisits no funcionals**

## 2.2. Requisits no funcionals

En aquest tercer sprint del nostre projecte ens hem centrat en mantenir els requisits no funcionals que ja vam proposar en els sprints anteriors, la rapidesa, la confidencialitat i la tolerància a fallades de la nostra aplicació, a més de l'abstracció i la reutilització del codi. En el sistema, més concretament en el backlog d'aquest sprint, hem creat les històries d'usuari necessàries per validar el seu compliment. Aquestes històries són:

### 2.2.1. Rapidesa

**Com a** usuari del sistema

**Vull poder** accedir als recursos proporcionats

**Per tal de** navegar entre les diferents funcionalitats el més ràpid possible

#### **Criteris d'acceptació**

- El temps de resposta ha de ser inferior a 5s.

Per tal d'assolir aquest objectiu, hem començat a paral·lelitzar, en l'aplicació, les tasques més costoses que són independents de la interfície gràfica. Com per exemple, l'actualització de dades al servidor és independent de l'actualització en local i, per tant, es pot realitzar en paral·lel amb el *thread* de la interfície gràfica. Malgrat tot, altres tasques com l'obtenció grups de la xarxa social interna han de bloquejar la interfície gràfica perquè es necessiten les dades per poder mostrar-los. En aquests casos, estem executant en paral·lel tot allò que és possible. En la pròxima iteració, realitzarem un petit estudi sobre el temps d'execució de les principals tasques de l'aplicació, i l'eficiència en la seva execució.

### 2.2.2. Disponibilitat

**Com a** usuari del sistema

**Vull poder** accedir als recursos proporcionats sempre i quan disposi d'internet

**Per tal de** poder accedir a l'aplicació des de qualsevol lloc

#### **Criteris d'acceptació**

- L'aplicació ha de poder funcionar encara que es disposi d'una baixa connexió.

Aquesta segona història l'hem pogut validar per l'estat actual de l'aplicació, tot i que, cal dir que sense connexió l'aplicació no funciona. La implementació actual ens permet accedir a certa informació, ja que aquesta es guarda en local, però no ens permetria interactuar amb altres usuaris ni amb el servidor.

### 2.2.3. Confidencialitat

**Com a** usuari del sistema

**Vull poder** guardar les meves dades de manera privada

**Per tal de** poder mantenir segura la meva informació sensible

#### **Criteris d'acceptació**

- Un usuari no pot accedir a informació privada d'altres usuaris.
- Un usuari no pot modificar informació d'altres usuaris.

Aquesta tercera història l'hem pogut validar per l'estat actual de l'aplicació. Això ho hem pogut dur a terme posant un requisit de privacitat a l'inici de sessió, ja que només tindrà accés a les dades de la base de dades de l'usuari logejat.

### 2.2.4. Abstracció i Reutilització (Actualitzat)

**Com a** desenvolupador del sistema

**Vull poder** reutilitzar parts del codi i disminuir la complexitat

**Per tal de** poder escriure nou codi i modificar el codi ja implementat sense tanta dificultat

#### **Criteris d'acceptació**

- Un desenvolupador ha d'intentar reutilitzar i fer codi reutilitzable sempre que sigui possible.

Aquest últim requisit no funcional l'hem pogut dur a terme aplicant molts dels patrons explicats posteriorment en la documentació, concretament els patrons que ens han permès garantir l'abstracció i reutilitzar codi han estat:

- Patró factoria abstracta: per crear famílies d'objectes relacionats sense especificar les seves classes concretes.
- Patró factoria simple: consisteix en utilitzar una classe constructora abstracta amb uns quants mètodes definits i altres abstractes, garanteix reutilització per part de les subclasses.
- Patró Adaptador: s'aplica quan es vol utilitzar una classe però la seva interfície no concorda amb la que necessitem, o quan es vol reutilitzar una classe.
- Patró model, vista, controlador: aquest patró divideix l'aplicació en tres parts interconnectades: el model de dades, la interfície d'usuari i la lògica de control; garantint abstracció entre elles, ja que una no coneix el funcionament intern de l'altra.
- Patró DAO: consisteix separar del tot la lògica de negoci de la lògica per accedir a les dades, l'apliquem al servidor per garantir abstracció.
- Patró Service Locator: serveix per a encapsular els serveis en una capa abstracta, l'hem utilitzat en l'accés a la nostra llibreria d'Android.
- Patró Singletó: restringeix la creació d'objectes, el seu rol és permetre l'existència de només una única instància de l'objecte i proporcionar una referència global a aquest, garantint la reutilització.

- Patró controlador de transaccions: fa d'intermediari entre la interfície i l'algoritme que l'implementa, reduint la complexitat i reutilitzant codi.
- Patró plantilla: és un patró de disseny de comportament que defineix l'esquelet d'un algoritme en un mètode que, en ser heretat per les subclasses, difereix en alguns dels seus passos, garanteix reutilització de codi.
- Patró estratègia: aquest és una patró de disseny de comportament que ofereix diversos algoritmes al client i li dona la possibilitat de canviar entre ells segons les seves necessitats en una sola comanda, garanteix reutilització de codi.

#### **Nova versió**

- Patró constructor: Aquest patró consisteix en definir un Builder per a la construcció del component Entry View, permetent així reutilitzar codi i amagar complexitat.

### **2.3. Tractament d'aspectes transversals (Actualitzat)**

En aquest tercer sprint del nostre projecte ens hem centrat en la geolocalització, tant amb mapa com sense, en la gamificació i en compartir contingut en les xarxes socials. També s'han solucionat alguns problemes amb el calendari de Google i s'ha començat la part de la gamificació amb medalles i trofeus.

A més, s'han mantingut i corregit alguns petits errors de els aspectes transversals tractats en sprints anteriors.





Pel que respecta al tema del multiidioma, hem mantingut el que ja teníem fet de l'sprints anteriors, i l'únic que s'ha fet ha estat solucionar algunes traduccions i afegir-ne les necessàries per als nous component.

De cara a futurs sprint ens agradaria poder tenir més stakeholders reals, ja que malgrat que hem tingut alguns, la situació actual dificulta molt obtenir més feedback. També voldríem completar la implementació de la gamificació amb trofeus i medalles, així i com la web app, de la qual encara hem de definir l'abast i la seva viabilitat.



**Nova versió**

A continuació es mostren les modificacions dels aspectes transversals. Els canvis queden indicats amb els colors següents:

-  Aspecte transversal assignat al primer sprint.
-  Aspecte transversal assignat al segon sprint.
-  Aspecte transversal assignat al segon i al tercer sprint.
-  Aspecte transversal assignat al tercer sprint.



Criteri	Meitat de la nota	Max nota
Geolocalització	Sense mapa	Amb mapa
Xarxes Socials	Login	Compartir contingut
Xat	Instantani	Històric
Stakeholders reals	En funció del nombre, qualitat i freqüència	En funció del nombre, qualitat i freqüència
Refutació	Notificacions	Bloquejos de comptes
Web App	Una funcionalitat	Més funcionalitats
Multiidioma	Arquitectura Preparada	Implementació correcta
Gamificació	Valoracions	Trofeus o similars
Calendari	Intern	Sincronitzat amb el de Google
Gràfiques	Mostrar-les	Interactives

**Taula 12. Aspectes Transversals**

## 2.4. Serveis de tercers (Actualitzat)

A continuació es dona una breu explicació dels serveis externs utilitzats en el projecte.

### 2.4.1. Firebase

Firebase és una plataforma que proporciona diverses eines per desenvolupar aplicacions. Aquesta plataforma proporciona una gran quantitat d'eines de les quals nosaltres hem utilitzat concretament dues, Cloud Firestore com a contenidor online per la nostra base de dades i Firebase Authentication per la gestió de l'autenticació dels usuaris. A continuació s'explica en més detall per a que serveixen aquestes eines.

Cloud Firestore és el servei que hem utilitzat per allotjar la nostra base de dades, aquest proporciona una base de dades NoSQL, flexible, escalable i en el núvol per tal d'emmagatzemar i sincronitzar dades per la programació tant des del client com des del servidor. Nosaltres hem utilitzat únicament l'emmagatzematge i sincronització des del servidor.

Firebase Authentication és el servei que hem utilitzat per gestionar l'autenticació d'usuaris, aquest proporciona serveis de backend, SDK fàcils d'utilitzar i biblioteques de IU ja elaborades per autenticar els usuaris en la teva app. Per aquest motiu i per la seva fàcil integració amb Cloud Firestore, l'hem utilitzat per gestionar l'autenticació d'usuaris.

### 2.4.2. Google Calendar

Google Calendar, és un servei gratuït d'agenda i calendari en línia desenvolupat per Google, que et permet sincronitzar-lo amb gmail i compartir-lo amb altres persones si volem, i et permet afegir esdeveniments i invitacions i fer cerques d'esdeveniments que poden interessar-te a la web.

Algunes característiques de Google Calendar són:

- Ús compartit del calendari: podem crear diversos calendaris per a, per exemple, compartir-los amb un grup d'alumnes o d'amics, amb la família, amb companys de treball, etc. Més informació.

- Invitacions: Crea invitacions a esdeveniments, enviar-les als teus amics i porta el compte de les seves respostes i comentaris. Tot això en un únic lloc. Els teus amics rebran la teva invitació i introduiran les seves respostes, encara que ells no utilitzen Google Calendar. Es poden transferir calendaris a altres usuaris de Google calendar.
- Subscripcions: Et pots subscriure a informació de calendaris públics
- Cercar: Cercar a calendaris públics per descobrir esdeveniments que et puguin interessar i afegir-los a teu propi calendari.
- Accés mòbil: Rep notificacions i recordatoris d'esdeveniments al teu telèfon mòbil.
- Publicació d'esdeveniments: Comparteix els esdeveniments que creguis oportú amb qui vulguis. Al fer públic el teu calendari, tots els teus esdeveniments apareixeran en els resultats de cerca públics de Google Calendar i de Google. A més, altres usuaris podran veure aquesta informació o afegir el calendari a la seva llista de calendaris.

#### **2.4.3. OAuth2**

OAuth 2 és una estructura (framework) d'autorització que li permet a les aplicacions obtenir accés limitat a comptes d'usuari en un servei HTTP, com Google, Facebook, GitHub i DigitalOcean. Delega la autenticació de l'usuari al servei que allotja el compte del mateix i autoritza a les aplicacions de tercers l'accés a l'esmentat compte d'usuari. OAuth 2 ofereix fluxos d'autorització per aplicacions web i d'escriptori; i dispositius mòbils.

Aquest servei és necessari per facilitar l'accés al Google Calendar des de la nostra aplicació utilitzant un compte de Google, ja que les API de Google utilitzen el protocol OAuth 2.0 per autenticació i autorització.

#### **2.4.4. Google Maps (Nou)**

Google Maps, és un servei gratuït de mapes, establiments i rutes en línia desenvolupat per Google, que et permet utilitzar el GPS per saber la teva posició actual, i et permet mostrar mapes actualitzats i poder manipular-los.

Aquest servei és necessari per facilitar l'accés als mapes i al establiments. A la nostra aplicació l'hem utilitzat per mostrar les rutes de passeig de les mascotes.

### 3. Cerimònies àgils

En aquest apartat es detallen els tres esdeveniments més significatius d'un *sprint*: el *sprint planning*, el *sprint review* i el *sprint retrospective*.

#### 3.1. Sprint Planning (Nou)

Aquest nou Sprint va començar el divendres 24 d'Abril i està previst que acabarà el divendres 15 de Maig. Aquest és el resum de la primera reunió:

- 1- Es va decidir que la primera setmana seria dedicada a acabar el que va faltar de la iteració anterior. Concretament: la qüestió de donar like, dislike i denunciar continguts i la gestió de les notificacions.
- 2- Es va revisar les històries d'usuari restants i es va decidir que aquesta iteració seria dedicada, sobretot, a la implementació dels mapes (tant per a la ruta de passeig com per a la cerca d'establiments) com a la possibilitat de compartir l'aplicació. Respecte a aquesta última història d'usuari, es va decidir eliminar la possibilitat de compartir la aplicació per Instagram, degut a que es va veure que aquesta opció ja no era compatible amb la present visió de la nostra aplicació.
- 3- Es va fer la distribució del treball. Es va fer immediatament aparent que les històries d'usuari d'aquesta iteració eren més Front-heavy, per tant, a Back es va decidir que aquesta iteració seria utilitzada, sobretot, per a refactoritzar tot el codi per tal de millorar la seva eficiència. També es va decidir avançar en el CRUD de les medalles, història d'usuari que s'havia deixat per a la següent iteració.
- 4- Es van fer canvis en la forma de revisar el codi. Així com fins ara havia estat suficient que el codi fos revisat per dues persones determinades, es va decidir que, d'una banda, les persones a revisar-ho serien seleccionades a l'atzar i, per l'altra, que n'hi hagués la possibilitat d'escollir una tercera persona.
- 5- Vam acordar que havia hagut una millora en la comunicació entre Front i Back, però que encara era millorable. Es van discutir alguns dels problemes i es va acordar fer un esforç per tal de millorar en aquest aspecte.

- 6- Per últim, de cara a la distribució de la documentació i dels equips, estàvem prou satisfets i vam decidir seguir treballant tal i com ho havíem fet fins ara.

### **3.2. Sprint review (Nou)**

El dia 15/5/2020 va ser el dia de la finalització de l'Sprint. Ens vam reunir per tal de veure què hem aconseguit entregar. Aquest document inclou el resultat de la nostra release.

#### **3.2.1. Què s'ha pogut entregar?**

Documentació: complerta. Hem pogut entregar tots els documents requerits.

- Donar like/dislike al contingut/fòrum.
- Rebre notificacions de grup.
- Compartir fotos de la galeria al fòrum.
- CRUD Exercici realitzat
- Realitzar passeig
- Consultar ruta passeig
- Eliminar ruta passeig
- CRUD rentats
- CRUD perfil mèdic mascota
- CRUD visita veterinari
- Compartir aplicació.

#### **3.2.2. Què no s'ha pogut entregar?**

- CRUD medalla
- Reportar post i bloqueig d'usuari a la xarxa social.
- Eliminar els registres més antics periòdicament.

### 3.2.3. Per quin motiu?

Les 2 primeres històries d'usuari no van poder ser entregades per qüestió de temps, la primera d'elles ja tenien en compte que només finalitzaríem la part de back, cosa que perquè l'hem fet, en canvi la segona, no hem aconseguit finalitzar-la per una qüestió de temps. La història d'eliminar registres més antics periòdicament és un cas a part, ja que un cop finalitzada la seva implementació vam decidir com a equip descartar-la perquè veiem que no tenia molt sentit eliminar documentació que potser és necessària en un futur, com per exemple la informació de les vacunes.

## 3.3. Sprint retrospective

Al dia 15 de Maig, l'equip es va reunir per realitzar l'Sprint Retrospective. Es va decidir realitzar-ho en dues parts: la primera amb tot l'equip, i la segona separant Front i Back per a fer front als seus problemes concrets. Aquest document recull les conclusions d'aquesta reunió:

### 3.3.1. Reunió General

En primer lloc va ser crític discutir per què la demo havia fallat aquell dia. Ens vam adonar que el que ens havia fallat era que havíem realitzat el "release" massa tard. Això va donar com a resultat que quan va arribar l'hora de la demo, aquesta va fallar perquè no havíem tingut temps suficient per a testejar-la.

A partir d'aquí vam raonar quins van ser els motius pels quals havíem arribat a aquesta situació. Possiblement havíem, un altre cop, caigut en l'error de deixar-ho tot per al final, però també es va suggerir que, senzillament, els objectius per a aquesta iteració havien estat massa costosos i, per tant, no hi havia hagut temps d'acabar-ho tot.

En qualsevol cas, es va prendre nota dels errors per tal d'aprendre per a la pròxima iteració.

Després vam comentar quines coses havien anat bé. D'una banda, ens vam posar d'acord en el fet que havíem millorat en qüestions de la distribució del temps. Per l'altra, en aquesta iteració havia millorat considerablement la comunicació entre Front i Back: mitjançant uns documents correctament redactats amb totes les

especificacions requerides i enviats per Google Drive, ja no va haver-hi gaire confusió respecte a què calia implementar.

### 3.3.2. Reunió Front

En la reunió del sprint retrospective de Front vam arribar a la conclusió de què la metodologia de treball en equips de 2 havia estat molt productiva, ja que ens va permetre no encallar-nos tan fàcilment quan sorgia algun problema. A més, vam parlar sobre l'evolució individual de cada membre i vam tractar qüestions individuals, com per exemple el tema de la dedicació. En comparació amb l'anterior iteració, tots els membres han posat un esforç superior per a poder realitzar l'entrega a temps. En la pròxima iteració, hem decidit seguir una estratègia similar per assolir els nostres objectius.

### 3.3.3. Reunió Back

La reunió a Back va ser curta. D'una banda vam estar d'acord en el fet que la distribució del temps havia estat molt millor i això ens havia permès acabar tots els objectius de Back, a més de fer una mica més. Per l'altra, també vam estar d'acord a dir que la comunicació amb Front havia anat molt millor. A part d'això, res més va sortir que fos rellevant.

### 3.3.4. Què s'ha fet bé i es mantindrà?

- **Mètode de realització de documentació:** al principi del *sprint* vam dividir la documentació en tasques a realitzar setmana a setmana. Cada setmana, durant les nostres reunions, dedicàvem 20 minuts a comprovar que tota la documentació d'aquesta setmana estigués llesta i decidíem què s'havia de fer per a la següent setmana i qui s'encarregaria de fer què. Aquest mètode ens ha permès organitzar-nos de tal manera que ningú s'ha vist compromès entre dedicar temps a programació i a documentació i, a més, ens ha permès entregar-la a temps.
- **Reunions fora d'horari de classe:** com que el nostre grup té reunions amb el *product owner* dimecres i divendres, ens vam adonar que l'espai de temps entre aquests dos dies era massa petit per a fer cap avanç important i, a la vegada, massa gran per a poder respondre adequadament a problemes que



poguessin sorgir en el desenvolupament del software. És per això que vam decidir que faríem reunions també els dilluns. Aquesta petita mesura ens ha dotat d'una millor agilitat i capacitat de planificació per tal de fer front als diversos esdeveniments que porta el desenvolupament d'una aplicació.

- **Comunicació per Discord:** la situació actual del COVID-19 ens ha forçat a tots a estar confinats en casa, per tant, vam perdre l'avantatge de la comunicació cara a cara. Afortunadament, vam decidir que ens comunicàriem per Discord, una plataforma de comunicació per a jugadors gratuïta i àgil que ens ofereix la capacitat de fer xat (tant oral com escrit) i *streaming* del nostre ordinador. Així, si algú tenia cap problema i necessitava ajuda, podíem respondre ràpidament.

### 3.3.5. Que canviarem per millorar? (nova divisió)

Hem decidit que en el quart i l'últim sprint farem una millor organització temporal de les tasques per intentar no tenir problemes al final del sprint per intentar anar massa ràpid. A més a més, tot i que la comunicació front-back ha millorat, intentarem millorar-la encara més, per així anar més a la par a l'hora de treballar i intentar resoldre els dubtes d'un equip amb l'altre el més ràpid possible.

### 3.4. Gràfiques (Actualitzat)

#### 3.4.1. Sprint burndown

En la gràfica de la figura 1, podem observar l'evolució del punts d'històries d'usuari pendents en cada dia del *sprint*. Aquesta evolució s'ha mantengut constant pràcticament tota l'estona; malgrat tot, cap al final va començar a disminuir considerablement. De fet, en l'última setmana es va passar de 25 punts per tancar a només 2.

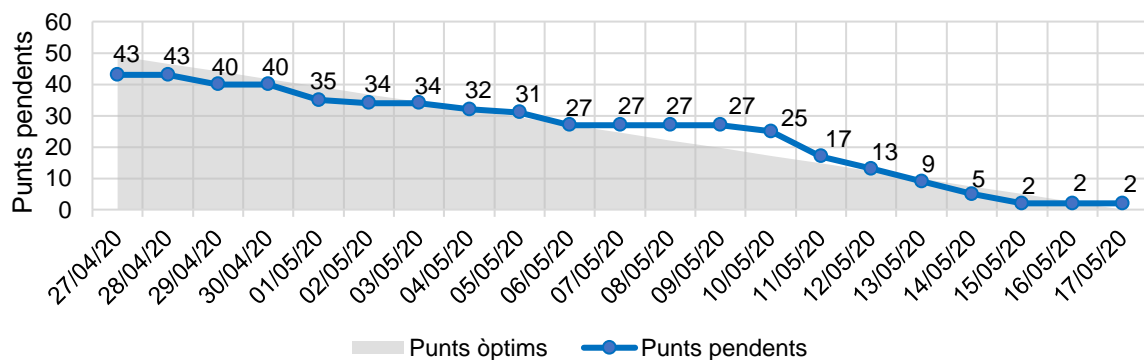


Figura 1. Sprint burndown de la tercera iteració

#### 3.4.2. Release burndown

En la figura 2, podem observar l'evolució general dels punts de les històries d'usuari pendents per implementar. Malgrat les dificultats que hem tingut a l'inici de la iteració, al final hem assolit menys d'una quarta part del projecte.

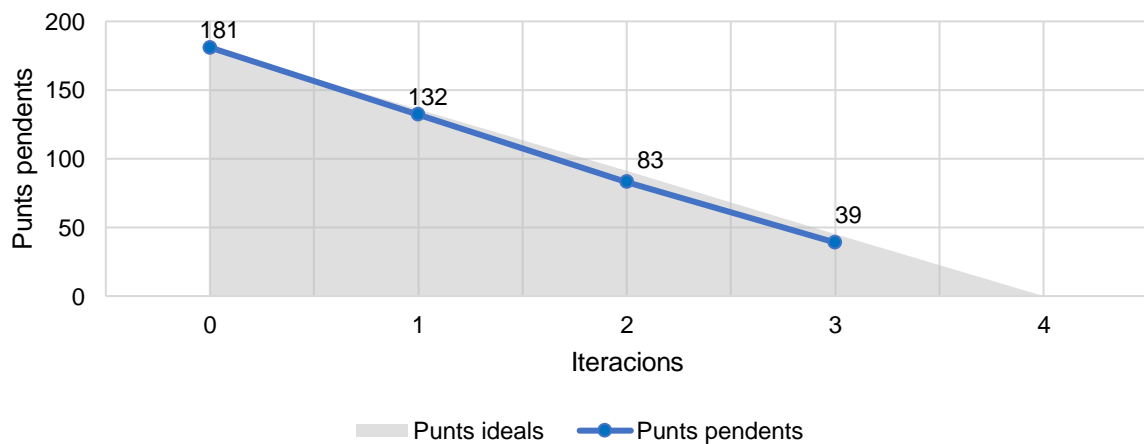


Figura 2. Release burndown

### 3.4.3. Velocitat de l'equip

En la figura 3, es poden observar les velocitats de l'equip durant el transcurs del projecte, fins el moment de la publicació d'aquest document. En aquesta es comparen els punts tancats i els planificats.

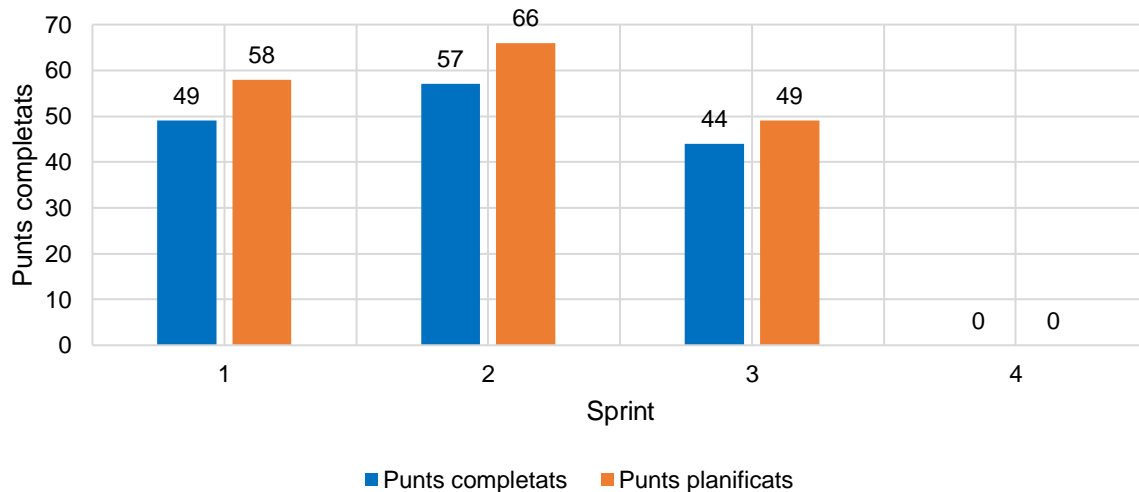


Figura 3. Representació de la velocitat de desenvolupament de l'equip

### 3.4.4. Dedicació mensual

En la figura 4, es troben representades les hores invertides per a cada desenvolupador en els darrers mesos. Podem observar que la quantitat d'hores dedicades durant el mes de març és molt superior a les del mes anterior. Aquest fet és degut a l'inici de la fase de desenvolupament.

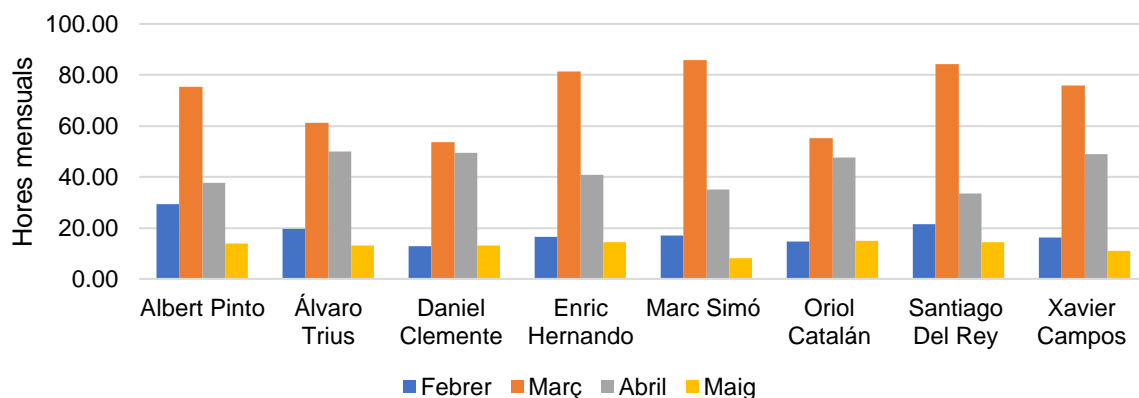


Figura 4. Hores de dedicació mensual del membres de l'equip


## 4. Metodologia

En aquest apartat s'explica el conjunt d'eines i convencions que s'utilitzaran per realitzar el projecte.

### 4.1. Metodologia àgil

Per a realitzar el projecte, hem decidit utilitzar SCRUM, una de les principals metodologies de desenvolupament de software àgils utilitzades actualment. En aquesta, es divideix el desenvolupament del projecte en iteracions o *sprints*, al final de les quals es disposarà d'una versió entregable del sistema.

En aquesta metodologia àgil, es defineixen tres rols destacats. Primerament, el product owner, el qual vetllarà per al compliment de les funcionalitats acordades i, en el nostre cas, serà sempre la mateixa persona. En segon lloc, tenim el *scrum master*, qui verificarà que s'estigui aplicant la metodologia de treball correctament i redactarà la documentació necessària per la iteració actual. Aquest rol serà rotatiu entre els diferents membres de l'equip. Finalment, els desenvolupadors són els encarregats d'implementar les funcionalitats acordades amb el client.

Per tal de coordinar les nostres activitats, durant les iteracions realitzarem les diferents reunions  que estableix aquesta metodologia. Per una banda, al principi de cada iteració, realitzarem una reunió per decidir que realitzarem en aquesta i durant el seu transcurs cada dia que ens trobem en persona indicarem quines tasques hem fet durant la setmana, què farem avui i quins problemes hem tingut. Per l'altra banda, al final de cadascuna de les iteracions realitzarem dues reunions, una amb el client per mostrar les funcionalitats implementades i l'altra per debatre sobre quins aspectes del desenvolupament podríem millorar. La duració de cadascuna de les iteracions serà de tres setmanes.



## 4.2. Definició de finalització

Per tal de decidir quan una història d'usuari està acabada hem establert un conjunt de criteris definits a continuació:

1. Està definida i conté els seus criteris d'acceptació en el Taiga.
2. Està implementada amb els seus tests necessaris per a complir els seus criteris d'acceptació.
3. Els mètodes de les classes estan documentats.
4. Passa la revisió automàtica i és verificada per com a mínim un altre membre.
5. S'ha afegit el codi a la branca de desenvolupament.
6. Els diagrames de classes estan actualitzats.

## 4.3. Gestió del projecte (actualitzat)

Per tal de realitzar el nostre projecte, hem decidit utilitzar un conjunt d'eines per la gestió d'aquest, tan a nivell de tasques com del codi

### 4.3.1. Taiga (nova divisió)

Primerament, per tal de controlar les històries d'usuari que hem dissenyat, ens hem decantat per fer servir Taiga, una eina gratuïta específica per a gestionar projectes àgils. Aquesta ens permet definir les èpiques i les seves històries, puntuar-les segons la seva complexitat, organitzar el product backlog, planificar els *sprints* i observar l'evolució de l'estat de les històries.

#### Nou

Per tal d'utilitzar correctament Taiga, hem definit en primer lloc els temes, les èpiques i les històries d'usuari. Atès que aquesta plataforma només permet fer distinció entre èpiques i històries, hem representat els temes com èpiques amb uns identificadors i colors distintius. Cadascun d'aquest disposa d'un identificador numèric i, acte seguit, es troben les seves respectives èpiques, l'identificador de les quals és en el format X.Y, essent X el tema i Y el número de l'èpica. A la figura 5 podem observar l'organització dels temes, les èpiques i les històries d'usuari.

▲ 0	#4 TEMA 2: Salut de la mascota	EPIC			In progress	
▲ 0	#15 EPICA 2.1: Informació bàsica	EPIC			In progress	
▲ 0	#26 Afegir pes			Sprint #1		Done
▲ 0	#151 CRUD Dades salut mascota			Sprint #2		Done
▲ 0	#152 CRUD perfil mèdic mascota			Sprint #3		In progress
▲ 0	#21 EPICA 2.2: Alimentació	EPIC			Done	
▲ 0	#27 EPICA 2.3: Higiene	EPIC			Done	
▲ 0	#32 EPICA 2.4: Veterinari	EPIC			Done	
▲ 0	#6 TEMA 3: Assistència d'exercici	EPIC			In progress	
▲ 0	#62 EPICA 3.1. Gestió de les rutes de passeig	EPIC			In progress	

Figura 5. Detall de les èpiques del projecte

Cadascuna de les històries d'usuari conté una descripció amb els criteris d'acceptació, les persones que han estat assignades en aquesta tant de front com de back, un camp per indicar dependències amb altres històries i els punts assignats. En la figura 6 en podem veure un exemple.

**#98 Realitzar passeig**  
PES\_MY-PET-CARE USER STORY

This user story belongs to **#62 EPICA 3.1. Gestió de les rutes de passeig** EPIC

[TASKBOARD](#)

[Link to epic](#) [Add tag](#)

Created by albertpg 23 Feb 2020 18:08

**Com a** usuari autenticat al sistema  
**vull poder** realitzar un passeig amb les meves mascotes  
**per tal de** poder controlar l'exercici físic que fa.

**Criteris d'acceptació**

- Ha de poder iniciar un passeig amb les seves mascotes.
- Ha de poder finalitzar un passeig iniciat.
- Ha de registrar en l'aplicació que s'ha realitzat el passeig com a exercici.
- No ha de guardar un passeig que no hagi finalitzat.
- No ha de poder passejar una mascota que no sigui seva.

**Custom Fields**

**Dependències** #105  
Identificadors de les històries que són dependències

**CLOSED** IN PROGRESS

**POINTS**

3 Story points

3 total points

**ASSIGNED**

albertpg

marcll

Daniel Clemente Marin

[+ Add assigned](#)

**WATCHERS**

[+ Add watchers](#) [Watch](#)

**VOTES** 0

Figura 6. Exemple d'una història d'usuari del Taiga

El conjunt de les històries d'usuari forma el product backlog. En el Taiga, podem visualitzar en una mateixa pantalla totes les històries definides, organitzar-les en els diferents sprints i observar l'evolució dels punts realitzats al llarg del projecte. En la figura podem veure com es mostra la informació.

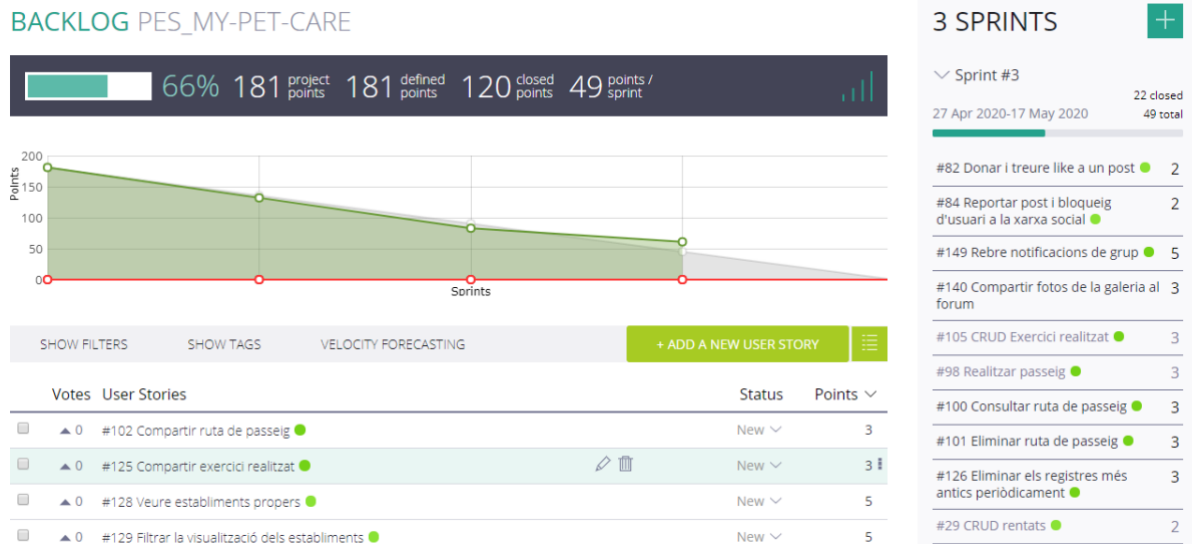


Figura 7. Product backlog en el Taiga

Un cop hem definit el sprint, hem dividit cadascuna de les històries assignades en tasques. En el sprint taskboard hem anat actualitzant l'estat de les tasques a mesura que s'anaven implementant, incloent els seus respectius tests. En la figura 8 podem veure un exemple del taskboard del sprint 3.

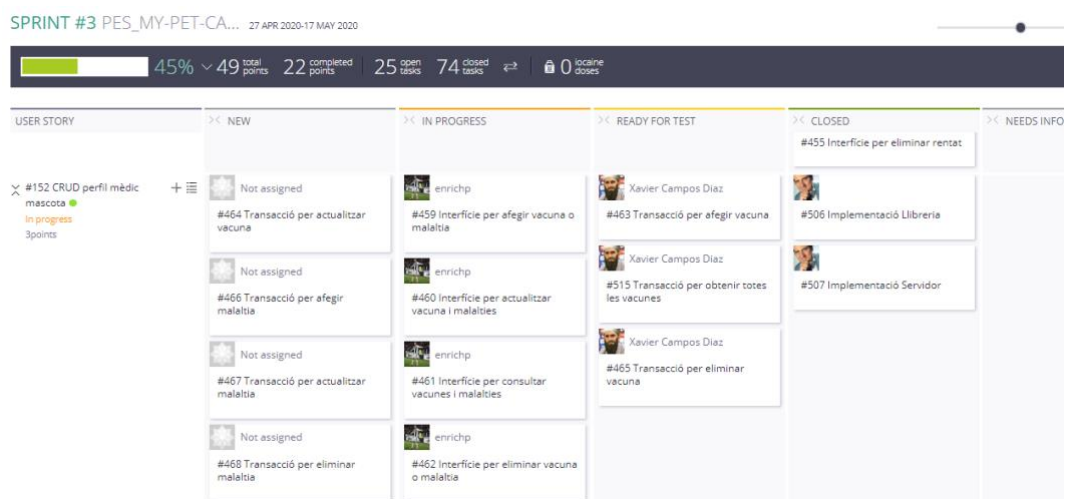


Figura 8. Taskboard del sprint 3

En una tasca d'una història d'usuari, podem veure el seu nom, la persona assignada per implementar-la i el temps emprat per realitzar-la. En la figura 9, podem veure un exemple de com es mostra aquesta informació.

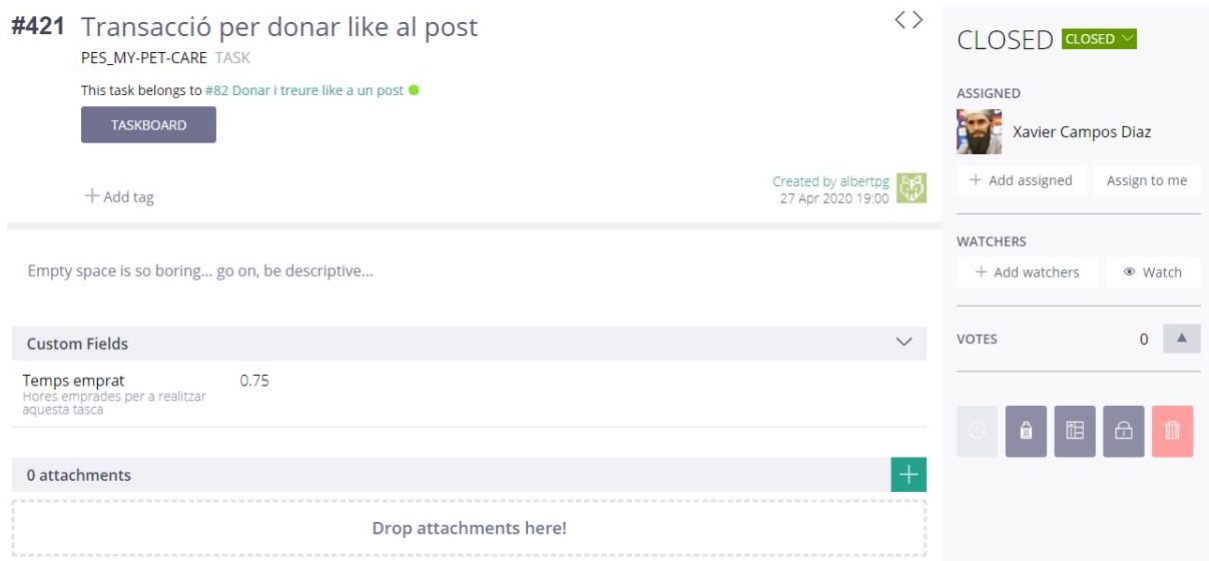


Figura 9. Informació d'una tasca

#### 4.3.2. Sistema de control de versions (nova divisió)

En segon lloc, per tal de gestionar el codi del projecte hem optat per fer servir GitHub, un sistema de control de versions (VCS). En aquest, utilitzem una metodologia de treball anomenada Gitflow, en la qual s'organitza el codi en branques independents entre elles. A més, aquesta eina és compatible amb les altres eines que utilitzarem en el projecte.

#### Nou

En el projecte, hem utilitzat tres repositoris (aplicació, llibreria i servei web) agrupats en una organització de GitHub. En cadascun d'aquests, podem observar les diferents branques que es troben obertes, les seves issues, els pull requests, les GitHub actions, els projectes de Github i les releases. En les figures 10 i 11, observem l'estructura de l'organització i la vista principal del repositori de l'aplicació.



**My Pet Care**  
Software project from FIB (Facultat d'Informàtica de Barcelona)

Repositories 3 Packages People 8 Teams 2 Projects 1 Settings

Find a repository... Type: All Language: All Customize pins New

**PES\_My-Pet-Care-Webservice**  
The My Pet Care application web service  
Java CC0-1.0 0 stars 1 issue 1 pull request Updated 15 minutes ago

**PES\_My-Pet-Care-Libraries**  
A set of Java libraries to access the services of the My Pet Care web service  
Java CC0-1.0 0 stars 3 issues 1 pull request Updated 12 hours ago

**PES\_My-Pet-Care**  
An android app for pet care  
Java CC0-1.0 1 star 3 issues 0 pull requests Updated 15 hours ago

Top languages  
Java

People 8 >

Invite your teammates...  
Invite

Figura 10. Organització de GitHub del projecte

Grupo13-PES-Mascotas / PES\_My-Pet-Care Watch 2 Star 0 Fork 1

Code Issues 3 Pull requests 0 Actions Projects 2 Wiki Security 0 Insights Settings

An android app for pet care Edit

Manage topics

751 commits 7 branches 0 packages 3 releases 6 contributors CC0-1.0

Branch: develop New pull request Create new file Upload files Find file Clone or download

xavier-campos Merge pull request #92 from Grupo13-PES-Mascotas/feature/feature\_vete... Latest commit 718d774 16 hours ago

.github	Update emulator api	19 days ago
.idea	Update My Pet Care libraries and delete unused modules	16 days ago
app	Fix Codacy issues	17 hours ago
docs	Delete PES_My-Pet-Care_Inception-1.pdf	16 days ago
gradle/wrapper	Add the feature settings menu branch	2 months ago
.gitignore	Update .gitignore	2 months ago
LICENSE	Initial commit	3 months ago

Figura 11. Pàgina principal del repositori de l'aplicació

Per tal de poder fer un merge d'una branca qualsevol cap a develop o master, es necessari realitzar un pull request. En aquest, es necessari que dos o tres membres, depenent de si va a develop o a master, que no l'hagin obert realitzin una revisió manual dels fitxers modificats. L'assignació dels membres en el cas de que sigui cap a develop es realitza de forma aleatòria, mentre que cap a master com que els grups són de 4 persones, sempre es requereix que la resta de membres el revisin. Cada revisor pot posar comentaris a diferents parts del codi i, un cop tots els fitxers han estat revisats, decidir si introdueix algun comentari general, aprovar les modificacions o requerir que es realitzi un conjunt de canvis. A més, existeixen un conjunt de comprovacions automàtiques: tests (unitaris i instrumentals), Codacy i GitGuardian. El merge del pull request es manté bloquejat fins que el mínim de revisors indicat hagin aprovat tots els canvis, passin tots els tests i no hi hagi cap issue al Codacy. En les figures 12, 13 i 14, podem observar un pull request del servei web.

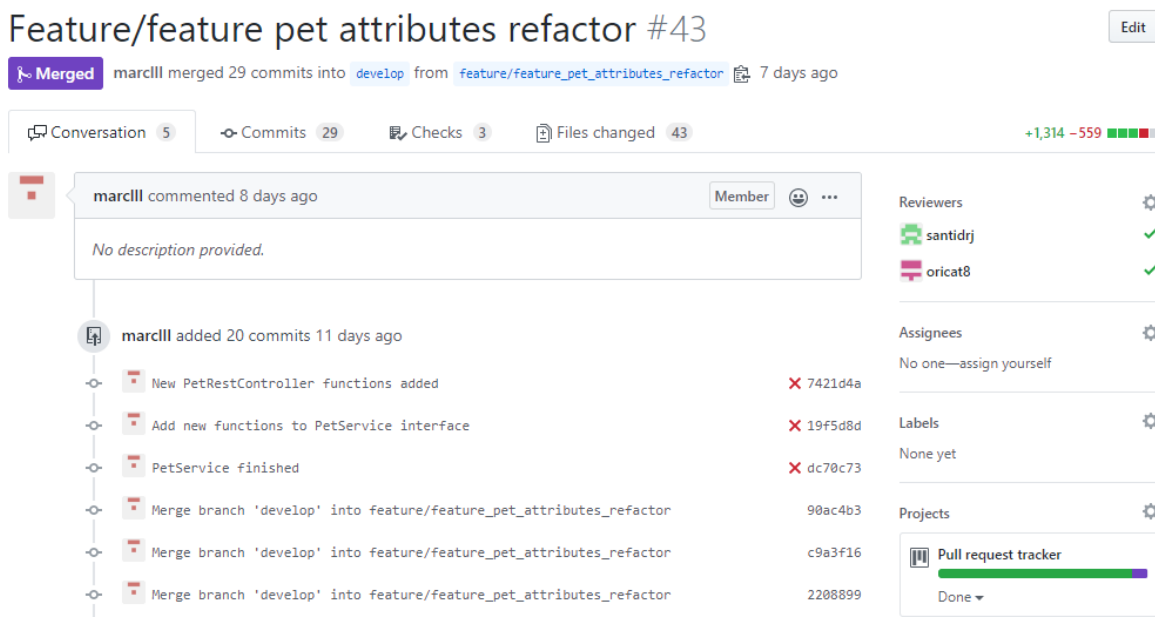


Figura 12. Pull request del servei web

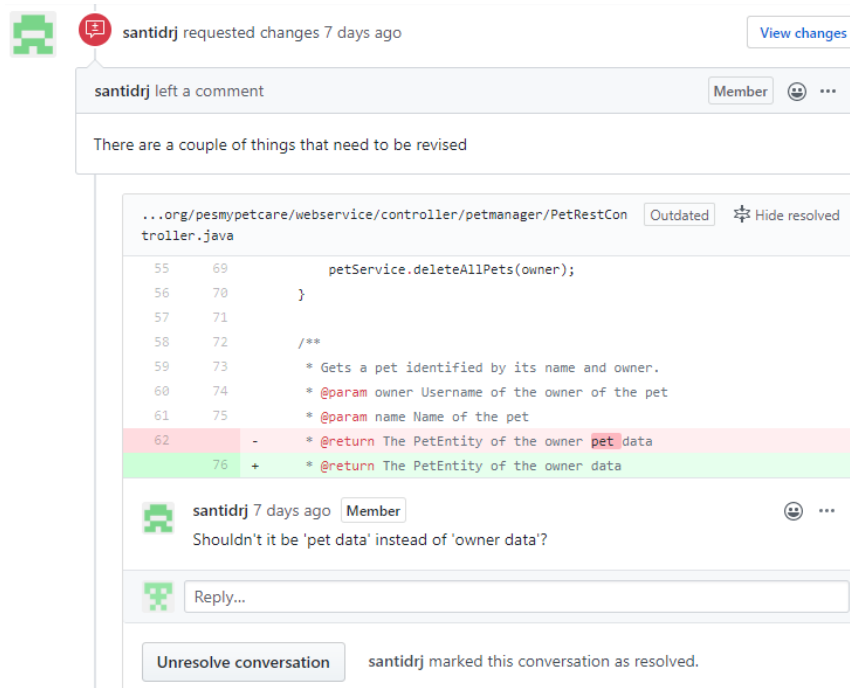


Figura 13. Exemple d'un canvi requerit en un pull request

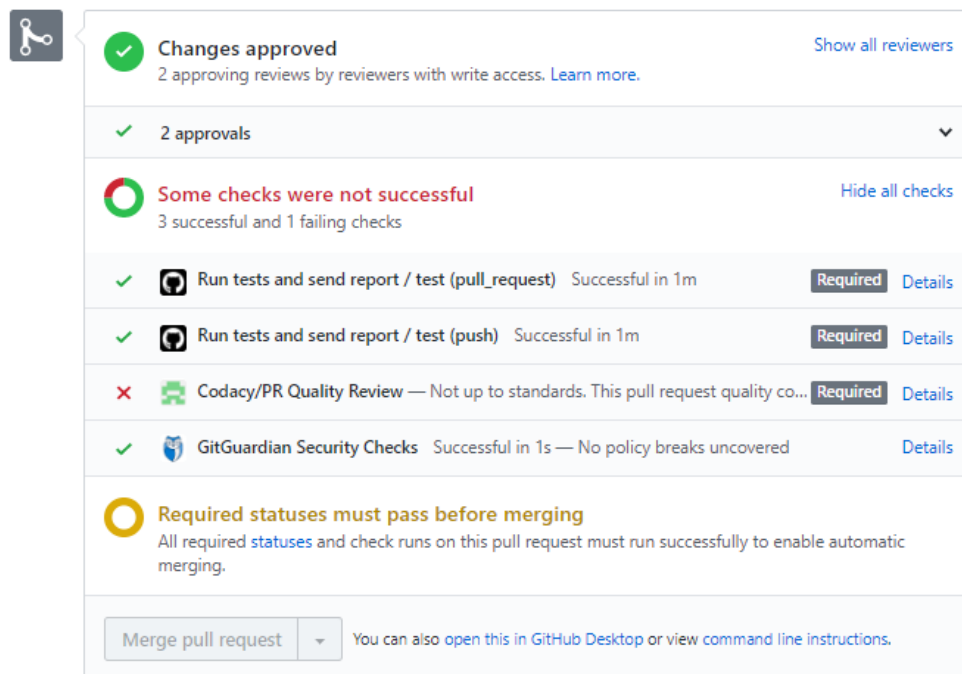
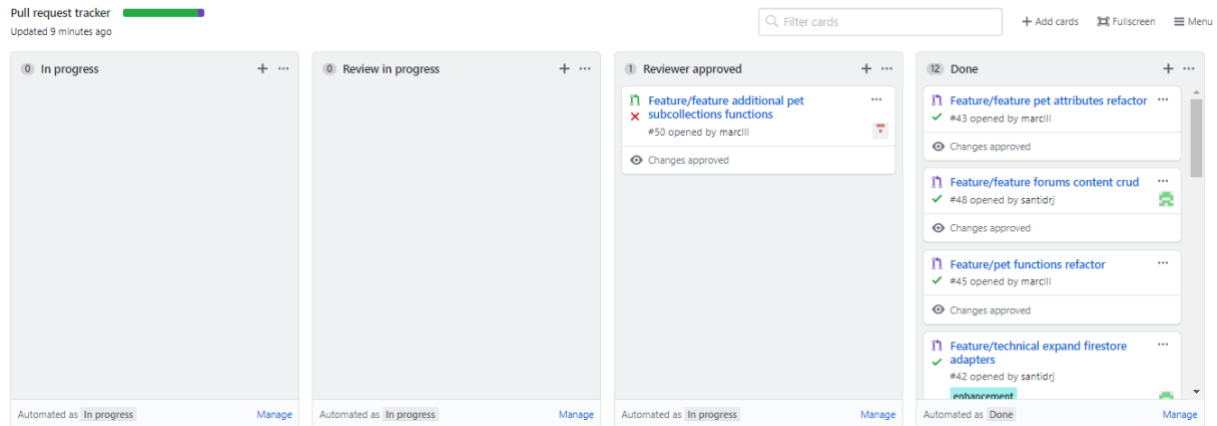


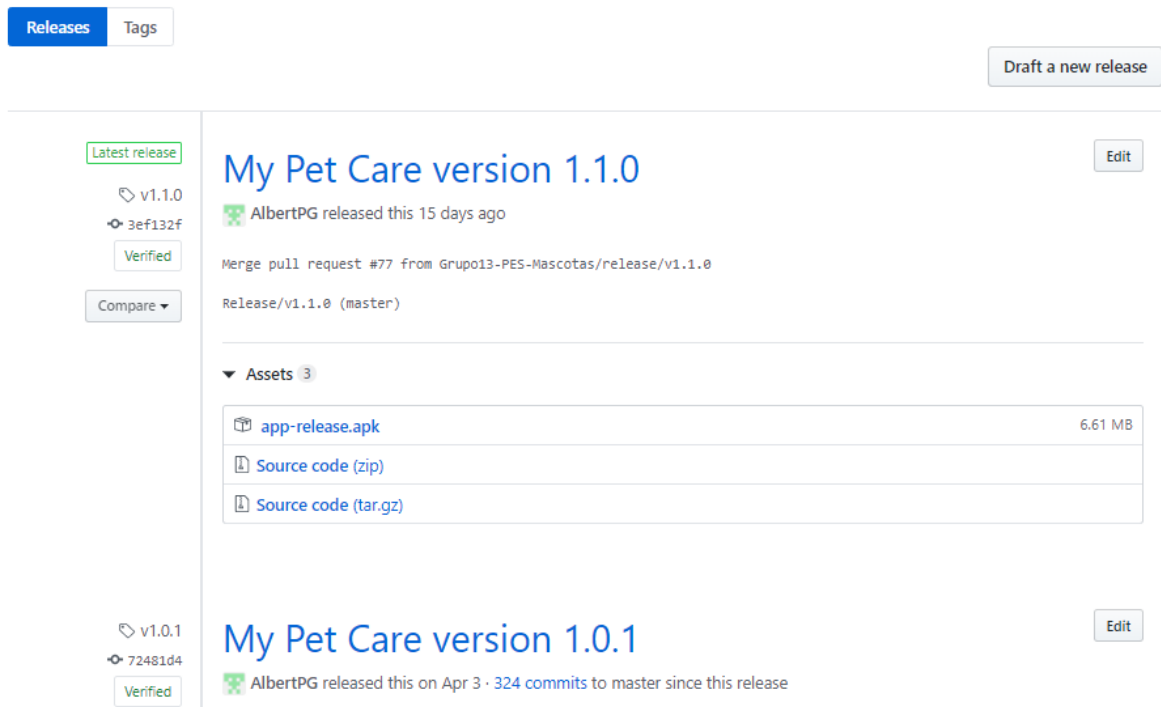
Figura 14. Exemple d'un pull request bloquejat

Acte seguit, per tal de controlar l'estat dels pull request i de les issues, hem utilitzat dos projectes de GitHub en cada repositoris. Aquests són assignats en els pull request i s'actualitzen automàticament. En la figura 15, podem observar l'evolució de l'estat dels pull requests al servei web.



**Figura 15. Pull request tracker en el servei web**

Finalment, en el GitHub portem un control de les releases que es realitzen en cada repositoris. En el cas del de l'aplicació, per exemple, s'afegeix l'instal·lador de l'aplicació, tal i com es pot observar en la figura 16.



**Figura 16. Releases del repositori de l'aplicació**

### 4.3.3. Project record track (nova divisió)

Per tal de controlar les hores de dedicació personal al projecte, hem creat un full de càlcul anomenat Project record track. En aquest, hem d'afegir les diferents activitats vinculades al projecte que anem realitzant, com per exemple, la realització del codi o la redacció de la documentació.

#### Nou

En la fase d'iniciació i en les dues primeres iteracions del desenvolupament hem apuntat les hores realitzades agrupant diferents tasques o parts de la documentació. Malgrat tot, a partir de la tercera iteració hem optat per modificar els criteris que utilitzem. Actualment, la documentació no l'apuntem en el full de càlcul i indiquem els identificadors de les tasques del Taiga que hem realitzat, si existeix. D'aquesta manera controlarem millor els progressos en termes de desenvolupament del projecte. En la figura 17, podem observar una fragment del project record track.

01/05/2020	Tasca #413 - Afegir imatge a un post	Albert Pinto i Gil	▼	0,40	
01/05/2020	Tasca #477 - Execució en paral·lel de l'obtenció de grups	Albert Pinto i Gil	▼	0,40	
01/05/2020	Tasca #485 - Adaptació de ImageZoomFragment	Albert Pinto i Gil	▼	0,40	
01/05/2020	Tasca #486 - Transacció per afegir imatge a un grup	Xavier Campos Diaz	▼	0,50	
01/05/2020	Tasca #487 - Transacció per eliminar la imatge d'un grup	Xavier Campos Diaz	▼	0,50	
01/05/2020	Refactor de l'accés a la base de dades	Santiago Del Rey Juárez	▼	0,50	
02/05/2020	Tasca #483 i #484 - Compartir App	Enric Hernando Puerta	▼	1,00	
04/05/2020	CRUD rentats	Enric Hernando Puerta	▼	2,00	#452, #453, #454, #455, #456, #457, #458, #491
04/05/2020	Expansió dels adaptadors per la base de dades	Santiago Del Rey Juárez	▼	2,00	
04/05/2020	CRUD Visites Veterinari	Xavier Campos Diaz	▼	2,00	#469, #470, #471, #472, #473, #475
04/05/2020	SubColeccions de Mascota al servidor i a la llibreria	Marc Simó Guzmán	▼	2,00	
04/05/2020	CRUD Exercici	Albert Pinto i Gil	▼	2,00	#445, #449, #447, #492, #448, #446

**Figura 17. Fragment del project record track**

#### 4.4. Convencions

Per tal de gestionar correctament les iteracions hem definit un conjunt de convencions de nomenclatura.<sup>1</sup> Els convenis que tenen un asterisc entre parèntesis “(\*)” són verificades pel *Codacy*:

- Tot allò que no sigui documentació o un comentari s’ha de realitzar en anglès.
- Les release tindran un nom en el format vX.Y.Z, començant en la v1.0.0, tal que:
  - Per a cada nova release s’incrementa la X.
  - Per a cada hotfix solucionat s’incrementa la Z.
  - Si la solució d’un hotfix ha provocat un gran canvi en l’aplicació, s’incrementa la Y.
- Els *pull request* realitzats s’anomenaran com la branca des d’on provenen, amb una breu descripció si es creu convenient.
- Els missatges dels *commits* s’escriuen en present.
- Exceptuant el *project record track*, cal iniciar sessió en totes les eines amb el compte de GitHub, si és possible.
- Els noms de les classes han de seguir un dels següents patrons, en funció de quin sigui el seu contingut:
  - Activitat: *NomActivity*
  - Fragment: *NomFragment*
  - Gateway: *NomGateway*
  - Vista (component): *NomView*
  - Intent: *targetActivityIntent*
  - JUnit test: *NomClassePerTestejarTest*
  - Altres: poden tenir el nom que es consideri oportú.
- En tots els fitxers en java s’utilitzarà camel case en tots els noms (\*); malgrat tot, en xml s’escriurà tot en minúscula separant les paraules amb guions baixos, excepte amb els identificadors els quals hauran d’estar amb camel case.
- Els noms de les variables i dels mètodes han de ser significatius.
- Les interfícies no poden començar amb el prefix “I”.

<sup>1</sup> Per consultar tots els convenis establerts, consulteu el següent [enllaç](#).

- Els identificadors dels components han de començar amb un prefix significatiu, els quals s'aniran concretant sota demanda. Per exemple, *btn*Nom pels botons o *lb*/Nom per les etiquetes de text.
- Les funcions no poden tenir més de 15 línies de codi, sense comptar els comentaris (\*).
- Les funcions no poden tenir més de 5 paràmetres (\*).
- L'idioma utilitzat en el codi és l'anglès, excepte el contingut dels elements del fitxer de recursos Strings.xml amb un locale assignat.
- Les diferents pantalles de l'aplicació, excepte el log in, han de ser implementades utilitzant fragments. Per tal de poder comprovar el funcionament n'hi ha prou amb crear una activitat i incloure el fragment, tot i que aquesta no pot estar en el *pull request* realitzat.
- S'ha de respectar l'arquitectura en tres capes establerta.
- Per accedir als diferents elements de la interfície gràfica, s'ha de fer servir el View Binding, introduït a Android Studio en la versió 3.6.
- Els noms de les branques han de seguir els següents criteris:
  - Si la branca està relacionada amb una nova funcionalitat, aleshores s'anomenarà *feature\_user\_story\_name*.
  - Si la branca implementa una llibreria per accedir al servei, aleshores s'anomenarà *library\_service\_name*.
  - Si la branca implementa un aspecte tècnic que no estigui en cap història d'usuari, aleshores s'anomenarà *technical\_action\_name*.
  - Si es tracta d'un altre tipus de branca s'anomenarà *type\_reason\_name*.
- Per a crear un intent des d'una activitat a una altre, s'ha de posar com a primer paràmetre "*NomActivity.this*" en lloc de "*this*".
- Els noms dels mètodes de les classes dels tests unitaris (JUnit) han de ser autoexplicatius.

#### 4.5. Gitflow (actualitzat)



Per tal d'utilitzar el repositori de codi eficientment i evitar problemes d'incompatibilitat de versions, hem decidit utilitzar una metodologia de treball anomenada gitflow. Aquesta estableix en quin tipus de branca del repositori s'ha d'incorporar el codi que hem realitzat i quin és el procediment a seguir en cada cas. En total distingim 5 tipus de branques: *master*, *develop*, *feature*, *release* i *hotfix*.

Primerament, en la branca de *master* es troba el codi de l'aplicació que forma part d'una versió publicada de l'aplicació, és a dir, el codi resultant al final d'una iteració. Per tant, no es pot realitzar cap *commit* que provingui directament de l'entorn de desenvolupament en aquesta branca. En el nostre projecte, hem acordat que l'únic cas on està permès fer un *commit* sense passar pel procediment habitual és quan incorporem al repositori la documentació d'aquest, la qual es troba únicament en aquesta branca.

Acte seguit, trobem la branca *develop*, la qual prové de la *master*. En aquesta s'aniran incorporant les diferents funcionalitats que es vagin desenvolupant durant el projecte, un cop hagin passat els diferents controls de qualitat establerts. De forma similar a la branca *master*, no es pot realitzar un *commit* directament en aquesta, sinó cal passar abans per un altre tipus de branca.

A continuació, disposem de les branques de tipus *feature*, les quals contenen el codi de les noves funcionalitats de l'aplicació o que solucionen problemes trobats durant el desenvolupament. Aquestes branques provenen de la branca *develop* i es on es realitzen tots els *commits* des de l'entorn de desenvolupament. D'aquesta manera el codi nou queda aïllat del codi comprovat i potencialment funcional de la branca original. Per tal de poder incorporar aquests canvis a la *develop*, és a dir, realitzar la fusió entre les branques, és necessari realitzar un *pull request*, en el qual es passaran un seguit de tests d'estil de forma automàtica i una comprovació de compatibilitat entre classes. A més, hem establert que el codi ha de ser verificat totalment per una altre membre del grup, el qual no hagi participat en el desenvolupament d'aquesta part. Malgrat tot, tots els membres poden comentar el codi i suggerir millores d'aquest. Un cop tots els tests han passat i el codi ha estat verificat, es pot realitzar la fusió amb la branca *develop* i, acte seguit, s'eliminarà la branca original.

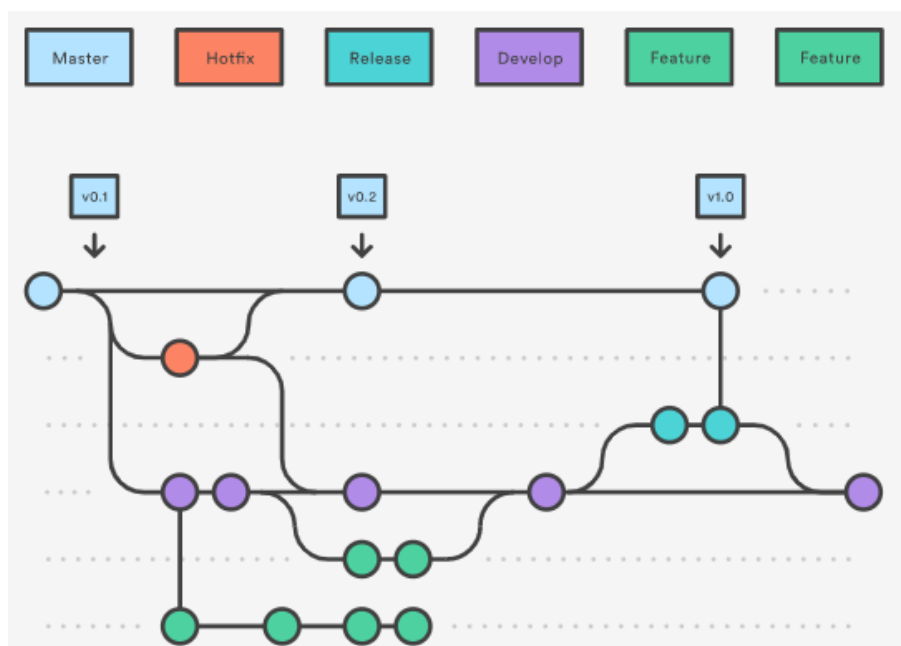


Quan s'acosta el final de la iteració, es decideix finalitzar la incorporació de noves funcionalitats i iniciar una branca del tipus *release*, la qual prové de la branca *develop*. Totes les funcionalitats que no han estat incorporades a la branca *develop* abans de l'inici d'aquesta nova branca queden fora de la versió actual de l'aplicació. En aquesta, es solucionaran petits errors o *bugs* que s'hagin detectat i es prepararà el codi per a ser lliurat com una versió de l'aplicació. Aquestes comprovacions es realitzaran mitjançant un *pull request* en el qual tots els membres del grup hauran de revisar alguna part del codi per tal de trobar inconsistències o detectar problemes. Un cop ha estat verificat es realitza primer la fusió amb la branca *master* i, acte seguit, amb la *develop*.

Finalment, si un cop s'ha publicat la versió de l'aplicació és detecta algun problema amb l'aplicació, sobretot provinent de les proves amb usuaris reals, s'ha d'iniciar una branca anomenada *hotfix* des de la *master*. Aquesta s'encarrega de solucionar aquests problemes i, acte seguit, s'inicia un *pull request*. Un cop el codi ha estat verificat i s'ha comprovat que aquests han estat solventats, es realitza la fusió amb la branca *master* i, a continuació, amb la *develop*.

**Nou**

En la figura 18, podem observar les diferents branques del Gitflow.



**Figura 18. Tipus de branques del Gitflow**

#### 4.6. Testeig (actualitzat)

Durant el desenvolupament del projecte, hem de realitzar un conjunt de tests per assegurar la integritat i el correcte funcionament de l'aplicació. Per a dur-los a terme, farem servir tres *frameworks* diferents: JUnit, el qual permet realitzar tests unitaris, Mockito, que permet realitzar *mocks* en tests unitaris, i Espresso, que permet realitzar tests instrumentals. L'ús d'aquests *frameworks* permetrà que puguem aplicar el TDD per a desenvolupar el codi relacionat amb la lògica de l'aplicació.

Per tal de calcular quin percentatge de les classes ha estat provat, utilitzem la llibreria Jacoco. Aquesta, genera un report detallat on s'indica el percentatge d'instruccions i branques que disposen de tests. A més, indica la quantitat total de línies, mètodes i classes que no estan sent testejades.

Per aprofitar al màxim aquests tests i la utilització de Jacoco, utilitzem les *pipelines* de GitHub per córrer els tests a cada push a les branques *feature* y en cada *pull request* cap a *develop* i *master*. Un cop es realitzen els tests i es genera el report, aquest s'envia a Codacy, el qual ens indicarà si el repositori compleix amb l'estàndard de cobertura que hem decidit.

**Nou**

En la figura 19, podem observar l'evolució de la cobertura en una de les branques del repositori del servei web.



Figura 19. Evolució de la cobertura d'una de les branques

### 4.7. GitHub Actions (actualitzat)

Per tal d'aplicar els principis de *continuous integration* i *continuous deployment*, farem servir les *pipelines* que ens proporciona GitHub, ja que és el nostre sistema de control de versions i, per tant, tindrà una millor integració amb el repositori. Aquestes *pipelines* ens proporcionen un mètode per automatitzar la realització de tests, la creació de paquets i muntar o desplegar la nostra aplicació.

Primerament, es defineix el nom que portarà la *pipe* i els esdeveniments que l'activen. Per a cadascun d'aquests, es pot escollir en quina o quines branques es pot iniciar. A continuació, es creen les tasques a realitzar, les quals s'executaran en paral·lel si no s'indica el contrari. Per a la creació de cadascuna d'aquestes s'indica el seu nom i, després, s'especifica en quin o quins sistemes operatius s'executarà, podent escollir entre diferents versions de Linux, mac Os o Windows. Un cop arribats a aquest punt, es defineixen les etapes que ha de seguir la tasca. Aquestes poden ser comandos del terminal o altres accions definides per GitHub o per altres usuaris.

**Nou**

En la figura 20, podem observar com s'han realitzat els tests amb les Github actions per un *commit* al repositoris de l'aplicació.

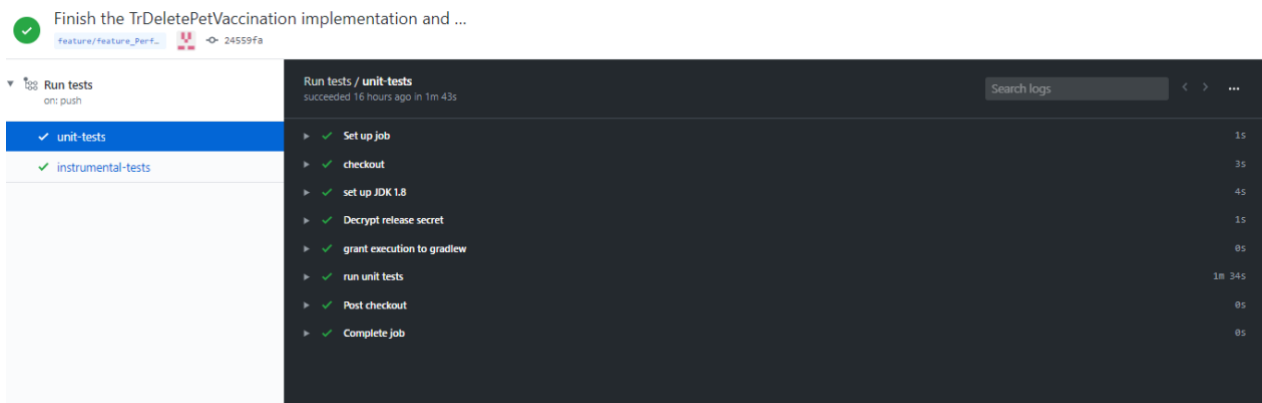


Figura 20. GitHub actions en un commit

#### 4.8. Dependabot (actualitzat)

Per tal de mantenir les dependències actualitzades i evitar així problemes de compatibilitat o seguretat, incorporem Dependabot. Aquesta és una eina que revisa el codi en busca de dependències desactualitzades. En cas de trobar-ne alguna, Dependabot obre un *pull request* informant de quina és la versió que ha trobat i quina és la que hauria d'haver-hi i, en el moment que detecta que s'ha resolt el problema, aquest tanca automàticament el *pull request*.

Nou

En la figura 21, podem observar un pull request creat pel Dependabot.

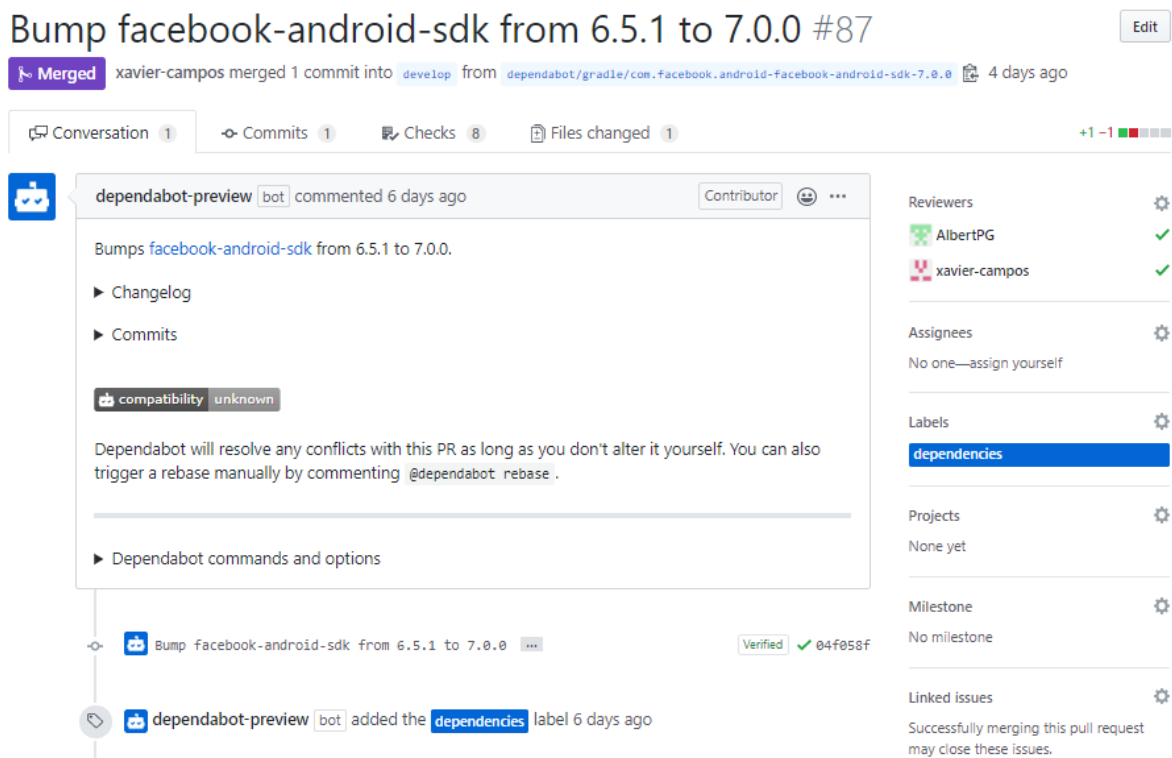


Figura 21. Pull request iniciat pel Dependabot

#### 4.9. GitGuardian (actualitzat)

Tenint en compte que el nostre projecte es troba a un repositori públic, on qualsevol pot veure el contingut del nostre codi, s'ha de tenir cura de no pujar informació que pugui comprometre la integritat de la nostra aplicació, com podrien ser les API keys de Google Maps o Firebase. Per aquest motiu, hem incorporat GitGuardian als nostre repositoris. Aquest eina de monitorització per a repositoris, s'encarrega d'analitzar tots els fitxers ubicats dins d'un repositori i alertar de possibles bretxes de seguretat cada cop que es realitza un push. En cas de trobar-ne alguna, s'envia un correu a la o les persones especificades indicant quin fitxer conté informació que pot ser considerada sensible.

**Nou**

En la figura 22 podem veure el correu que GitGuardian ens va enviar avisant-nos de la filtració de la nostra api key de Google maps. Actualment, aquesta qüestió ja està resolta.

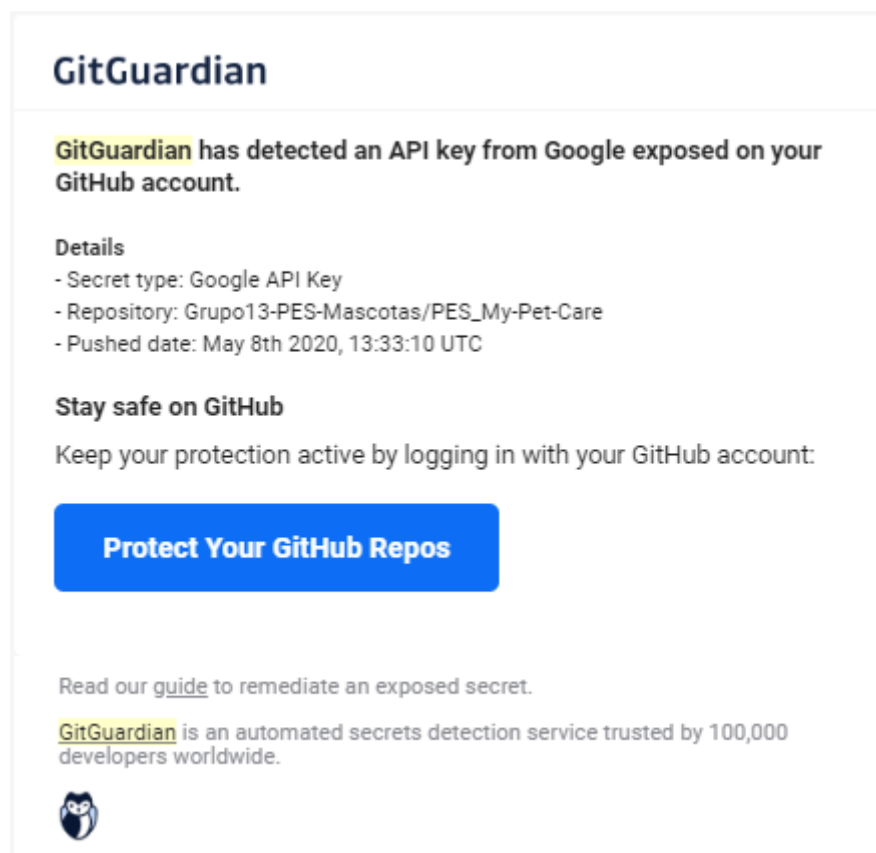


Figura 22. Correu electrònic de GitGuardian

#### 4.10. Qualitat del codi (actualitzat)

Per tal de mantenir el nostre codi net i sense *code smells*, utilitzarem una eina de revisió i anàlisi de codi anomenada Codacy. Aquesta eina disposa d'un seguit de regles, les quals pots activar i personalitzar, per tal de crear uns criteris de revisió pel codi. Amb aquestes, s'intenta que tot el codi desenvolupat dins del projecte segueixi els mateixos criteris d'estil, és a dir, que estigui estructurat i definit de la mateixa manera independentment de qui l'hagi escrit.

Primerament, l'eina porta un control dels problemes detectats als codis, com poden ser errors en l'estil, zones de codi duplicat o no utilitzat, o parts de codi que poden ser propenses a errors, entre altres. Tots aquests problemes es mostren en una taula on apareix la quantitat de cadascun. Un altre lloc on els podem trobar és a la gràfica de qualitat, en la qual es mostra l'evolució de la qualitat del projecte. Aquesta s'obté de mesurar la quantitat d'errors en el codi, la complexitat d'aquest i la quantitat de codi repetit.

Acte seguit, podem veure el llistat de *commits* i *pull requests* que s'han anat realitzant juntament amb la quantitat d'errors que aquest ha generat i quins han siguts arreglats. També ens permet veure informació més detallada dels nostres fitxers mostrant-nos, per exemple, la seva complexitat i els errors que es troben en aquests. En el cas dels errors, aquests apareixen ressaltats sobre el codi, juntament amb el motiu i una breu explicació d'aquest, cosa que facilita molt la seva correcció.

Finalment, Codacy disposa d'integració amb GitHub, permetent així enllaçar el nostre repositori amb el revisor. D'aquesta manera, podem configurar que s'iniciïn revisions del codi cada cop que es realitzi un *commit* o un *pull request* a una determinada branca. A més, Codacy s'encarrega d'enviar els resultats de les revisions com a resposta de l'acció que l'ha activat. Això ens permetrà detectar i corregir ràpidament possibles problemes al nou codi, sense haver de perdre temps en que un dels revisors el llegeixi sencer.



**Nou**

En la figura 23, podem observar les issues detectades per Codacy en un commit al repositori de web services.

src/main/java/org/pesmypetcare/webservice/entity/petmanager/PetEntity.java

More than 5 parameters (found 8).	▼
38 public PetEntity(GenderType gender, String breed, String birth, String pathologies, String needs,	
Avoid unused constructor parameters such as 'profileImageLocation'.	▼
39 Double recommendedKcal, String profileImageLocation, String calendarId) {	
Assigning an Object to null is a code smell. Consider refactoring.	▼
47 this.profileImageLocation = null;	
Assigning an Object to null is a code smell. Consider refactoring.	▼
48 this.calendarId = null;	
Boolean expression complexity is 5 (max allowed is 3).	▼
56 if (!GENDER.equals(field) && !BREED.equals(field) && !BIRTH.equals(field)	

Figura 23. Issues detectades pel Codacy en un commit

## 4.11. Entorns integrats de desenvolupament

Per tal de desenvolupar el nostre projecte, utilitzarem dues de les eines més completes que hi actualment per a treballar amb Java i, sobretot, amb android: Android Studio i IntelliJ IDEA.

### 4.11.1. Android Studio

Per a realitzar l'aplicació mòbil que es connectarà amb el nostre servidor, utilitzarem Android Studio. Aquest IDE, especialitzat en aplicacions android, disposa d'un conjunt d'eines per a realitzar tant la interfície gràfica de l'aplicació com la lògica d'aquesta, de forma senzilla i fàcilment usable.

Per una banda, les aplicacions android utilitzen XML per tal de dissenyar l'aparença de les pantalles. Malgrat tot, aquest llenguatge pot ser difícil d'utilitzar, sobretot si no s'hi està acostumat. Per aquest motiu, aquest entorn de desenvolupament ofereix la possibilitat de dissenyar-les utilitzant un entorn gràfic, en el qual els components es poden col·locar arrossegant-los a la posició desitjada dins d'un contenidor d'elements i es poden modificar cadascuna de les seves propietats fàcilment.

Per l'altra banda, per tal de dissenyar la lògica de l'aplicació, aquest IDE disposa d'unes eines eficients per a mantenir el codi organitzat, llegible i eficient. A més, es poden afegir eines de testeig, com per exemple JUnit, de forma senzilla i s'ofereix una vista de testeig pròpia per aquestes.

Finalment, a part de poder desenvolupar l'aplicació, aquest entorn de desenvolupament es pot sincronitzar amb una eina de control de versions, com per exemple GitHub. Des del mateix programa, és possible crear noves branques, fer *commits* i *pushes* al repositori i iniciar els *pull request* per tal de realitzar la fusió entre les branques del projecte. D'aquesta manera no és necessari utilitzar un intèrpret de comandes per tal de realitzar aquestes accions en el repositori.

#### 4.11.2. IntelliJ IDEA

En el nostre projecte, tota la lògica es troba en el nostre servidor web. Per tal de configurar-lo, utilitzarem IntelliJ IDEA, un dels IDEs més destacats actualment. Aquest disposa d'un gran conjunt d'integracions amb els *frameworks* més utilitzats, entre ells Spring, el qual utilitzarem per a desenvolupar el servidor. A més, com que és dels mateixos creadors que Android Studio, segueix un mètode de funcionament similar.

L'entorn de desenvolupament integrat d'IntelliJ IDEA, disposa d'un conjunt d'eines per tal de millorar la qualitat del codi, les quals es troben també disponibles a l'Android Studio. A més, degut a la seva integració amb Spring podem gestionar el servidor i actualitzar-lo si és convenient. A més, també disposem de diverses opcions per a connectar-se amb el sistema de control de versions, és a dir, amb el nostre repositori de GitHub específic pel servidor. D'aquesta manera, podem aplicar la metodologia de treball del Gitflow sense cap dificultat.

#### 4.12. Comunicació (actualitzat)



Durant la realització del projecte, és important que tots els membres estiguem en contacte. Per tant, hem decidit utilitzar un conjunt d'eines de comunicació específiques per a determinades situacions.

Per una banda, al principi del projecte hem utilitzat sobretot el WhatsApp, ja que és l'eina més còmoda per comunicar-se actualment. Malgrat tot, hem començat a utilitzar Slack, una aplicació que permet crear canals de text independents per a cadascun dels equips que formen el projecte, a part del canal general. En el nostre cas, hem creat un canal específic pel *front end* i un altre pel *back end*. Aquests canals, més el general, estan sincronitzats als seus respectius repositoris de GitHub, al Codacy i al

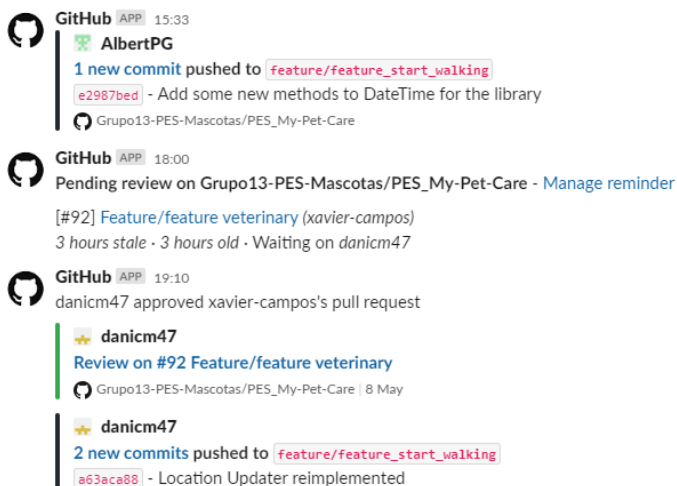


Taiga, permetent doncs estar informats de qualsevol canvi que es produeixi en aquests.

Per l'altre banda, per a la realització de reunions virtuals hem decidit utilitzar dues eines diferents, en funció de la finalitat de la reunió. Si en la reunió ha de participar el *product owner*, aleshores fem servir Skype, en el qual tenim un grup específic per a parlar amb ell. En canvi, si la reunió és per avançar en el desenvolupament de l'aplicació aleshores utilitzem Discord, en el qual hem creat un servidor amb diferents canals de veu en funció de l'equip. D'aquesta manera, els diferents equips poden parlar sense que l'altre estigui de fons i si s'ha de tractar alguna qüestió de forma general disposem del canal general i d'altres canals secundaris per a grups reduïts.

**Nou**

En la figura 24, podem observar un dels missatges enviats per GitHub en un dels canals de slack.



**Figura 24. Missatges del GitHub en un dels canals de slack**

### 4.13. Gestió de bugs (Actualitzat)

En el moment de detectar un *bug* ja sigui en l'aplicació per Android o en el servidor es necessari obrir una *issue* al GitHub. Atès que el Taiga està sincronitzat amb GitHub, la *issue* també s'obrirà en aquest; malgrat tot, és necessari accedir-hi per tal d'assignar el revisor de la *issue* i establir el tipus, la prioritat i la gravetat d'aquesta.

Un cop la *issue* ha estat especificada correctament, el revisor obre una branca del tipus *hotfix* des de *master*. En aquesta es realitzen les modificacions necessàries per solucionar el *bug* detectat i, en el cas de que sigui possible, s'actualitza la versió de l'aplicació i s'obren dos *pull requests* a GitHub: un a *master* i l'altre a *develop*. Un cop els dos han estat aprovats, es prossegueix a eliminar la branca creada i es tanca la *issue* al GitHub i, de forma automàtica, al Taiga.

**Nou**

En la figura 25, podem observar les *issues* tancades en el repositori de l'aplicació.

<input type="checkbox"/>	3 Open  8 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>	<b>Change mail not working</b> #51 by enrichp was closed on Apr 10						
<input type="checkbox"/>	<b>Requires api Marshmallow</b> <span>enhancement</span> #50 by enrichp was closed 15 days ago						1
<input type="checkbox"/>	<b>Problem with calendar translation</b> <span>bug</span> #42 by AlbertPG was closed 15 days ago						1
<input type="checkbox"/>	<b>Floating button for registering a new pet is not hiding on the beginning</b> #39 by AlbertPG was closed on Apr 3						1
<input type="checkbox"/>	<b>The back button sets the title to false and doesn't change</b> #32 by AlbertPG was closed on Mar 25						1
<input type="checkbox"/>	<b>Problem with CircularImageView Before version 9 (not included)</b> #20 by AlbertPG was closed on Apr 3						2
<input type="checkbox"/>	<b>Override annotation in MainActivity are missing</b> #9 by AlbertPG was closed on Mar 20						
<input type="checkbox"/>	<b>Document User class</b> #5 by santidjr was closed on Mar 25						1

Figura 25. Issues tancades en l'aplicació

#### 4.14. JitPack (Nou)

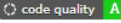
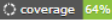

JitPack es un repositori de paquets per a projectes Android i JVM. Aquest munta projectes Git sota demanda i proveeix als seus usuaris amb artefactes *ready-to-use* (jar, aar).

Degut a que es un servei gratuït per a projectes públics i està integrat amb GitHub, hem optat per utilitzar-lo per fer més fàcil la distribució de les diferents llibreries que hem creat. D'aquesta manera, l'equip que es troba desenvolupant l'aplicació per mòbils, només ha d'afegir a les dependències de l'aplicació les llibreries que volen fer servir i automàticament Gradle descarregarà de JitPack els paquets demanats i els afegirà al projecte.

Un altre motiu ha estat la possibilitat que ens ofereix JitPack d'allotjar els diferents Javadoc de cada una de les llibreries i projectes en una ubicació web, sent així més senzill accedir i compartir aquestes amb la resta de membres de l'equip o tot aquell qui estigui interessat.

En la figura 26, podem observar el read me de la llibreria on s'explica com afegir el JitPack en el Gradle del projecte d'Android.

**Pes\_My-Pet-Care-Libraries**

A software project from FIB (Facultat d'Informàtica de Barcelona)   

**Download**

**Gradle:**

Add to your project build.gradle:

```
allprojects {
    repositories {
        maven { url "https://jitpack.io" }
    }
}
```

To add a specific module:

```
dependencies {
    implementation 'com.github.Grupo13-PES-Mascotas:PES_My-Pet-Care-Libraries:{latest version}'
}
```

To add all modules:

```
dependencies {
    implementation 'com.github.Grupo13-PES-Mascotas:PES_My-Pet-Care-Libraries:{latest version}'
}
```

**Figura 26. Read me de la llibreria amb la informació del JitPack**

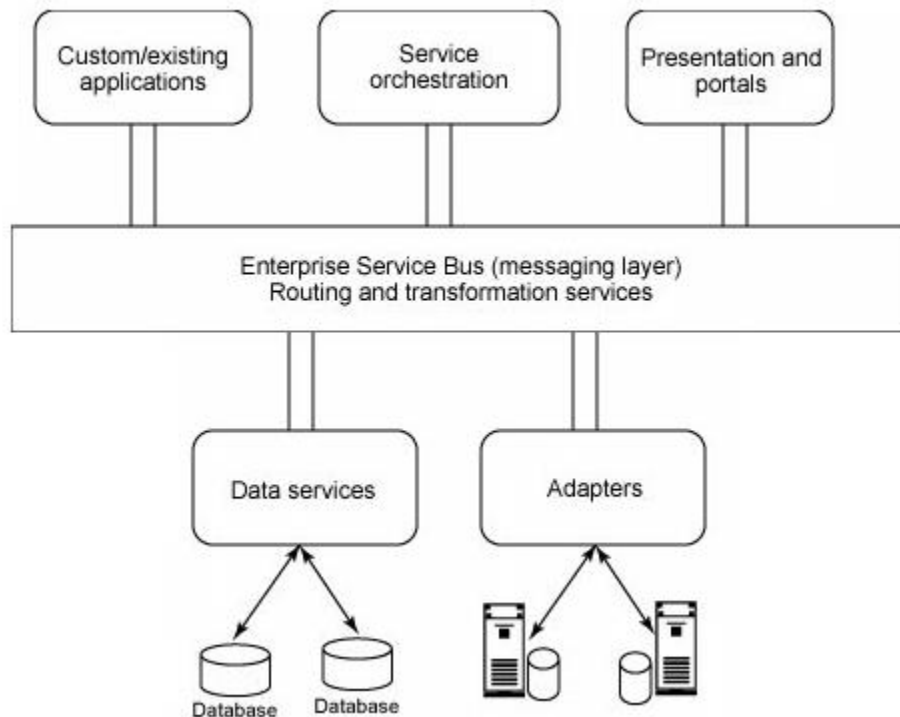
## 5. Descripció tècnica

En aquest apartat es dona una explicació dels patrons i architectures utilitzades per realitzar el projecte.

### 5.1. Concepció general de l'arquitectura

La nostra arquitectura es basa en la utilització d'APIs que nosaltres mateixos generem en la part del servidor i també aquelles que ens proporciona Google, Whatsapp o Facebook. Sobretot utilitzarem la API de Google ja que volem sincronitzar Google Calendar amb el nostre usuari i també farem ús de Google Maps. Aquesta arquitectura orientada a serveis ens permet ser més flexibles, poder reutilitzar més fàcilment i també reduir l'acoblament ja que cada API funciona de manera independent. També ens permet modificar, quan veiem necessari, més fàcilment alguna part del nostre projecte. Creiem que el fet d'utilitzar aquesta arquitectura no només ens serà útil per desenvolupar la nostra aplicació, sinó que també serà de gran utilitat per poder col·laborar amb els altres projectes, tant per a que un altre grup pugui integrar en el seu projecte una de les nostres APIs com per a que puguem integrar nosaltres un servei d'un altre grup. Un altre aspecte a destacar és la rapidesa que ens atorga aquesta arquitectura ja que és un aspecte que els usuaris valoren positivament ja que millora la experiència d'usuari.

Per la comunicació entre la interfície i la base de dades utilitzarem una arquitectura de tipus ESB (*Enterprise service bus*), que unifica les peticions de l'usuari cap a la base de dades a través de les APIs. ESB consisteix en una integració distribuïda gràcies a la utilització de contenidors de serveis. Amb aquesta arquitectura aconseguim crear un intermediari entre les diferents plataformes i APIs de la nostra app, fent així la comunicació molt més fàcil, ja que el propi bus s'encarrega de traduir les peticions per a cada servei (la part del bus que realitza aquestes transformacions es coneix com un adaptador), encara que vinguin d'un que inicialment no sigui compatible. Un altre avantatge que ofereix ESB, és que els diferents serveis i APIs es connecten directament al bus i no entre elles, simplificant i reduint les unions entre elles i fent que sigui molt més fàcil la seva utilització.



**Figura 27. Arquitectura Enterprise Service Bus**

Altres avantatges que ofereix l'arquitectura ESB és la simplicitat de l'encaminament, la seguretat en el lliurament de missatges, el suport per a protocols tant síncrons com asíncrons i també una coordinació de serveis d'aplicació múltiples encapsulats con un de sol.

Si utilitzem aquesta arquitectura es per aprofitar els beneficis que ens ofereix, com per exemple l'acomodació de sistemes existents molt més ràpida, la creació de serveis *ready-to-use* i la facilitat d'exportació a altres aplicacions.

## 5.2. Diagrames arquitectònics (Actualitzat)

A continuació veurem el disseny físic del nostre projecte, el qual es divideix en tres capes. En el front-end, es troba l'aplicació per a dispositius Android que es comunicarà via peticions HTTP amb el servidor. A continuació, ens trobem amb el mid-tier, on es troba ubicat el nostre servidor el qual farà d'adaptador per a resoldre les peticions cap als diferents serveis que s'ofereixen als usuaris, tant els nostres propis com el de tercers com Google o Facebook. Finalment arribem al *back-end*, on es troba ubicada la nostra base de dades, contra la qual el servidor realitzarà les diferents consultes o modificacions que calgui segons les peticions que rebí.

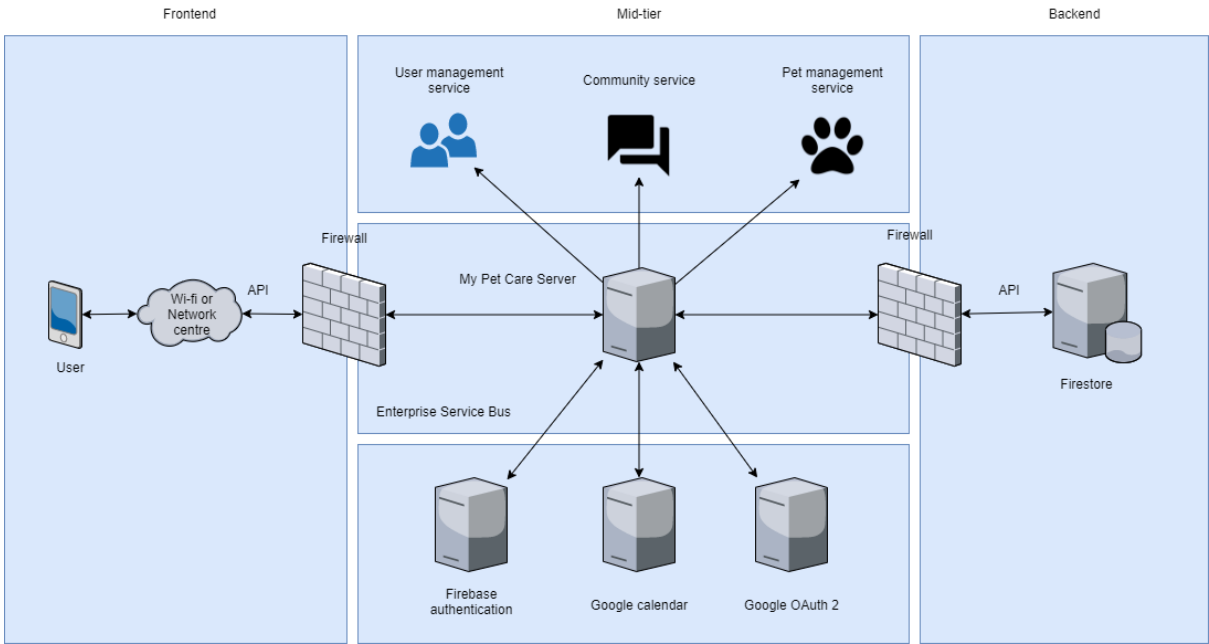


Figura 28. Representació de la capa física (Original)

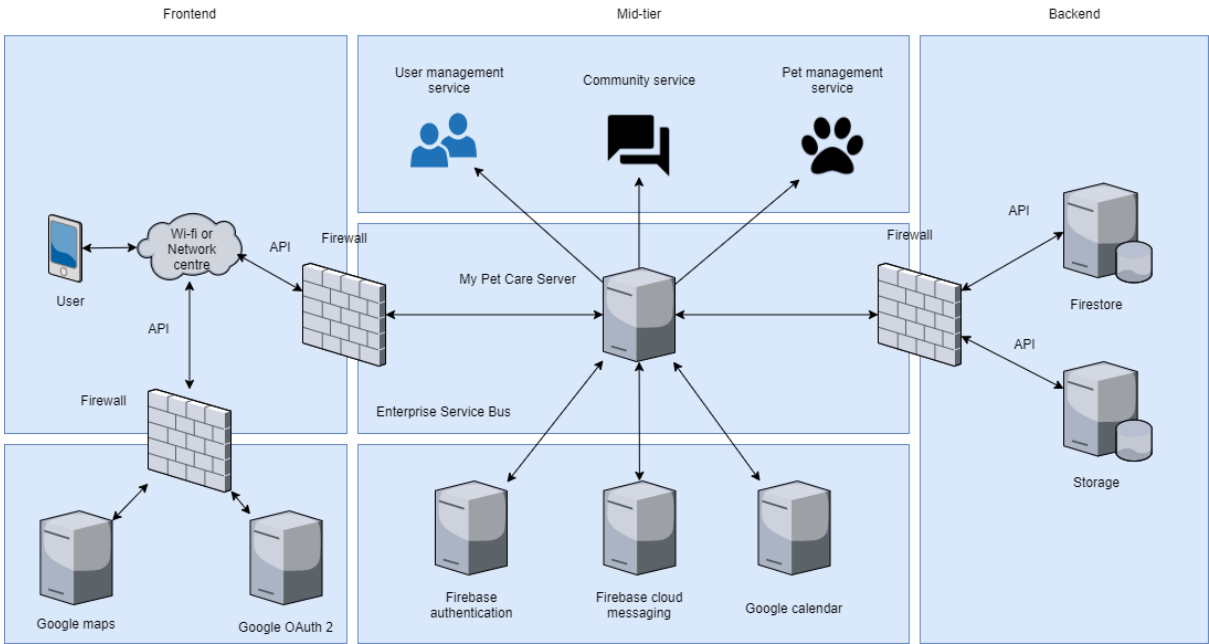


Figura 29. Representació de la capa física (Nova versió)

Acte seguit, es mostra el diagrama de components del projecte en el qual es poden observar amb més detall la comunicació entre el dispositiu i els diferents serveis que hem desenvolupat en el projecte.

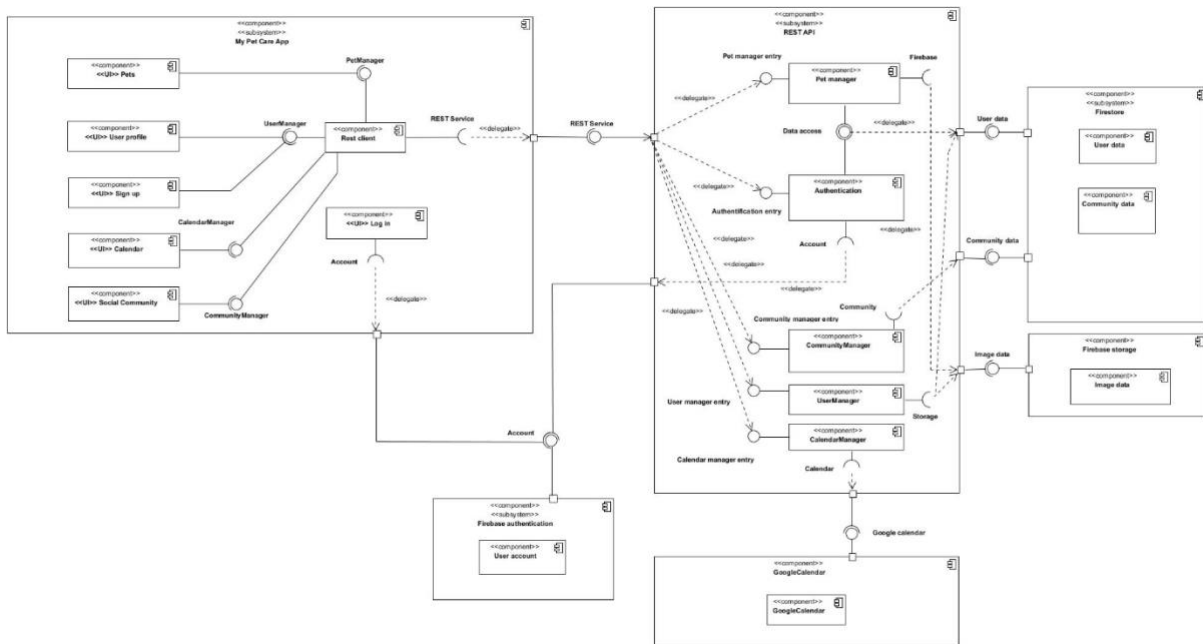


Figura 30. Diagrama de components (Versió sprint 2)

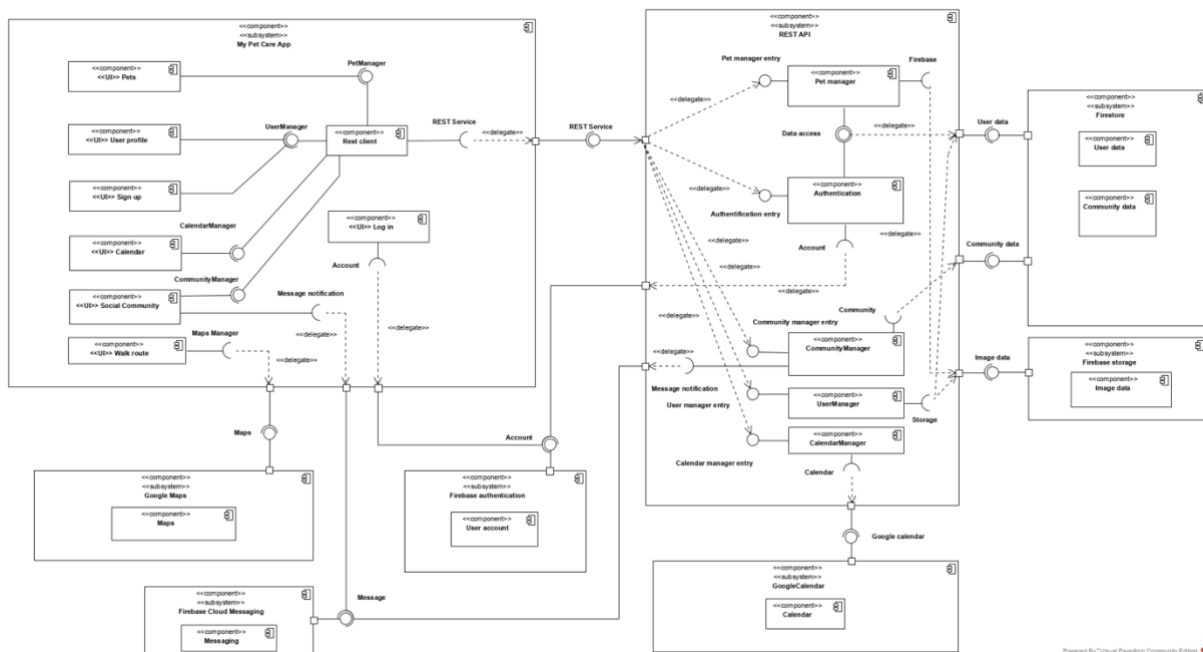


Figura 31. Diagrama de components (Nova versió)

### 5.3. Patrons aplicats

A continuació es descriuen els patrons aplicats en el projecte fins al moment.

#### 5.3.1. Factoria abstracta

El patró factoria abstracta és un patró de disseny per el desenvolupament software que soluciona el problema de crear diferents famílies d'objectes.

S'aplica quan:

- Es preveu que s'inclouran noves famílies d'objectes.
- Existeix una dependència entre els tipus d'objectes.

Consisteix en elaborar una interfície per crear famílies d'objectes relacionats sense especificar les seves classes concretes, de la següent forma:

- Factoria abstracta: Defineix les interfícies de les factories i proveeix un mètode per l'obtenció de cada objecte que pugui crear.
- Factoria concreta: Representa les diferents famílies de productes, proveeix la instància concreta que s'encarrega crear.

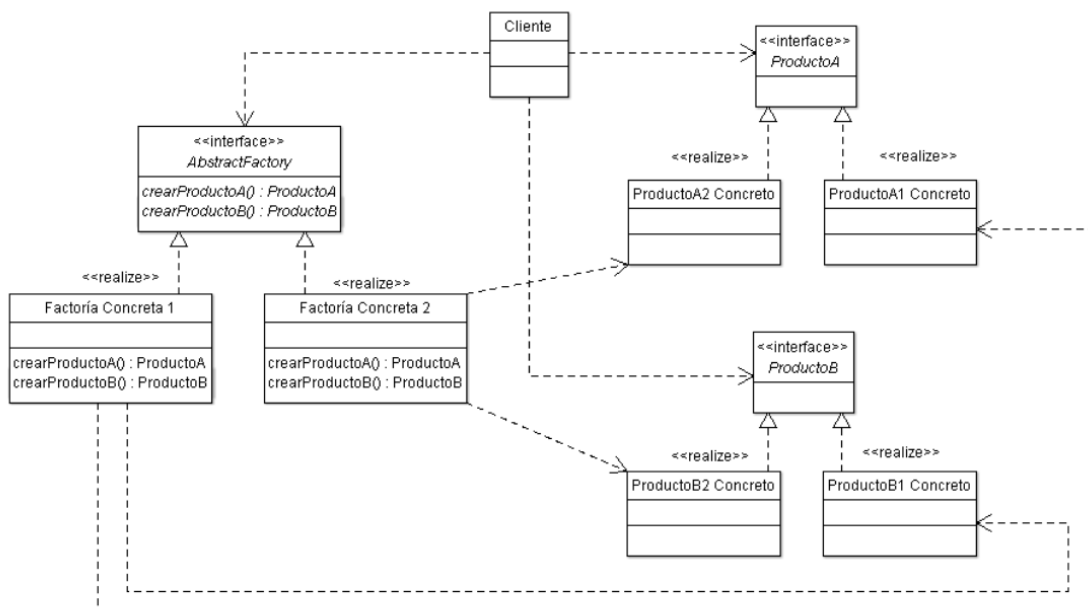


Figura 32. Exemple del disseny UML d'una factoria abstracta



Pel que respecta l'ús, aquest patró s'utilitza per la creació dels controladors de transacció, i l'utilitzem perquè es compleixen els requisits per utilitzar-lo, es preveu que s'inclouran noves famílies d'objectes i existeix una dependència entre els tipus d'objectes, i per tant ens facilita la creació de diferents famílies d'objectes.

### 5.3.2. Factoria simple

És una simplificació del patró de Factoria abstracta. Consisteix en utilitzar una classe constructora abstracta amb uns quants mètodes definits i d'un altre abstracte dedicat a la construcció d'objectes d'un subtipus d'un tipus determinat.

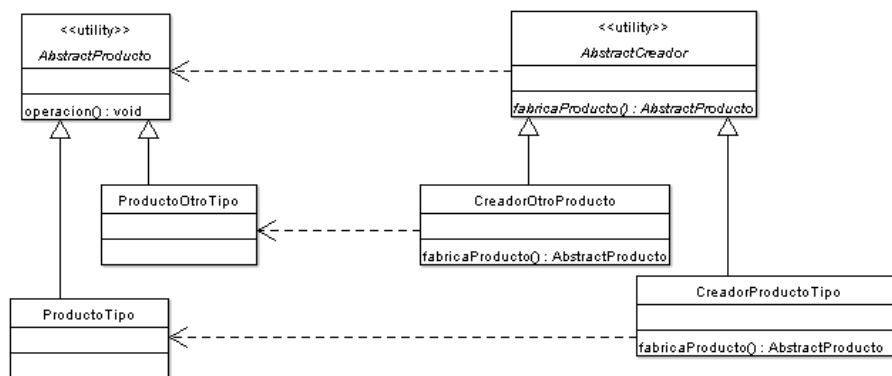


Figura 33. Exemple del disseny UML d'una factoria simple

A la nostra aplicació fem ús d'aquest patró per crear instàncies dels serveis Firebase.

### 5.3.3. Adaptador

El patró adaptador ens permet que dues classes amb diferents interfícies puguin treballar de manera conjunta a partir de la creació d'un objecte que les comunicarà i per tant, que permetrà que s'utilitzin els mètodes de la classe a adaptar.

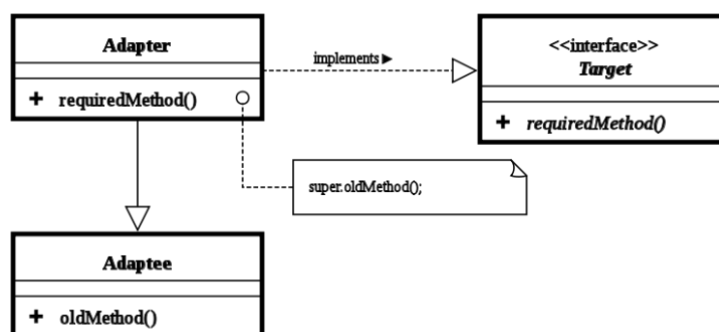


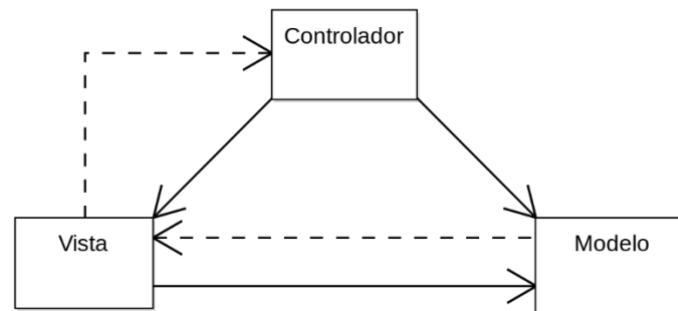
Figura 34. Exemple del disseny UML d'un adaptador

S'aplica quan es vol utilitzar una classe però la seva interfície no concorda amb la que necessitem, o quan es vol reutilitzar una classe.

Pel que respecta a l'ús, aquest patró s'utilitza per realitzar l'accés als serveis i així poder utilitzar les seves funcionalitats sense connectar-nos directament al servidor.

#### 5.3.4. Model, vista, controlador

L'arquitectura Model–Vista–Controlador (MVC) és un patró de disseny utilitzat per la implementació d'interfícies d'usuari. Aquest patró de desenvolupament de programari divideix l'aplicació en tres parts interconnectades: el model de dades, la interfície d'usuari i la lògica de control.

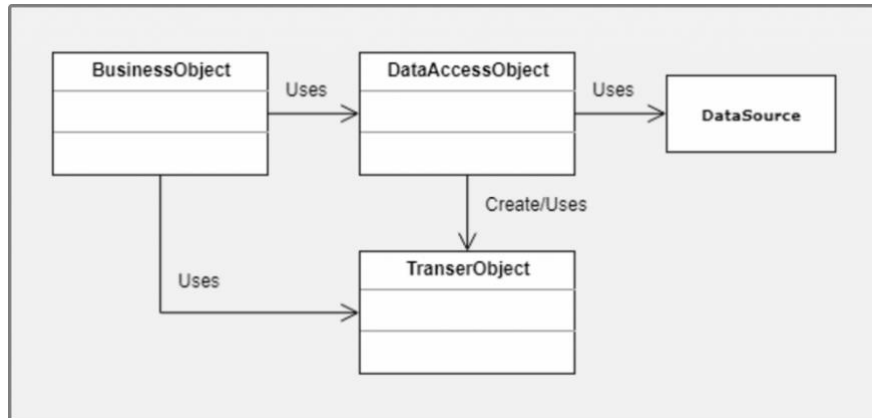


**Figura 35. Exemple del disseny UML del MVC**

Aquest està basat en les idees de reutilització de codi i separació de conceptes, característiques que busquen facilitar la tasca de desenvolupament d'aplicacions i el seu posterior manteniment, sent aquest el motiu de la seva utilització en el nostre projecte. El patró ha sigut utilitzat en el front-end de l'aplicació ja que és on s'implementa la interfície d'usuari.

#### 5.3.5. DAO

El patró Data Access Object (DAO) proposa separar del tot la lògica de negoci de la lògica per accedir a les dades, d'aquesta manera, el DAO proporcionarà els mètodes necessaris per inserir, actualitzar, esborrar i consultar la informació; d'altra banda, la capa de negoci només es preocupa de la lògica de negoci i utilitza el DAO per interactuar amb la font de dades.



**Figura 36. Exemple del disseny UML del DAO**

L'avantatge d'utilitzar el patró DAO és l'aïllament entre lògica de negoci i la font d'informació (generalment, una base de dades), d'aquesta manera el DAO no requereix coneixement directe del destí de la informació que manipula i la lògica de negoci és independent del tipus de font d'informació utilitzada ja que el DAO s'encarrega d'interactuar amb aquesta. Per aquest motiu hem aplicat aquest patró per l'accés a la base de dades des del servidor.

### 5.3.6. Service Locator

Aquest patró és utilitzat per a encapsular els serveis en una capa abstracta. Rep el nom pel component central *Service Locator* que retorna una instància dels serveis en ser sol·licitades pels clients.

Principalment té 4 components:

1. Service Locator: Abstrau tota la complexitat de fer ús d'un servei i li ofereix al client una interfície senzilla. Això, a més de l'avantatge que dona la senzillesa, permet també al client reutilitzar-lo.
2. InitialContext: És l'objecte del punt de sortida en el procés de cerca i creació. Els proveïdors de serveis proporcionen aquest objecte, que varia depenent del tipus de servei proporcionats.
3. Service Factory: És l'objecte que gestiona el cicle de vida de l'objecte Business Service.

4. Business Service: És un objecte que compleix el rol del servei que el client ha sol·licitat.

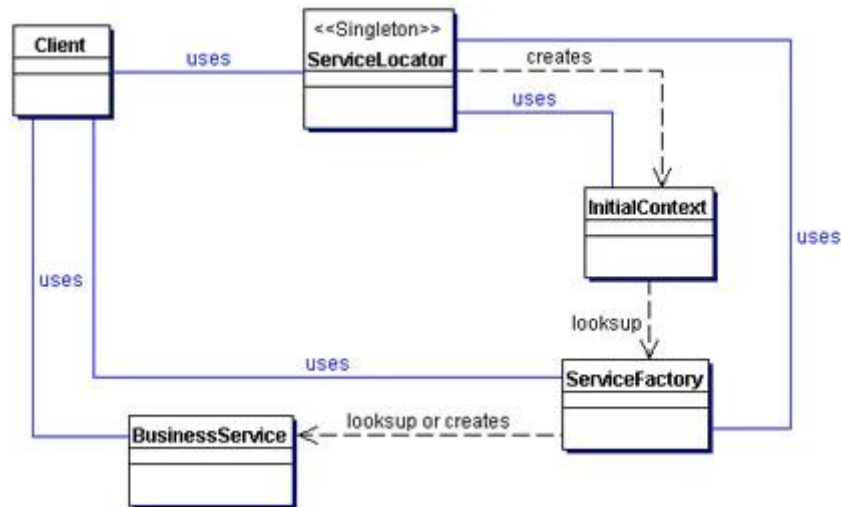


Figura 37. Exemple del disseny UML del Service Locator

Aquest patró és utilitzat per l'aplicació mòbil per accedir als serveis que proporciona el servidor.

### 5.3.7. Singletó

Aquest és un patró que restringeix la creació d'objectes. El seu rol és permetre l'existència de només una única instància de l'objecte i proporcionar una referència global a aquest. Hi ha diverses maneres d'implementar-lo, però la idea principal és la de la creació d'un mètode públic que s'encarregui de cridar a un constructor privat per crear la instància si no existeix cap altre. La classe, a més, disposarà d'un mètode públic que servirà per obtenir aquesta instància.

En la nostra aplicació, aquest patró s'utilitza sobretot per la creació de les factories Abstractes.

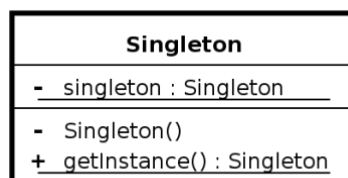


Figura 38. Representació d'un singletó en UML

### 5.3.8. Expert

Aquest és un patró lligat amb les bones pràctiques de programació que consisteix en assignar responsabilitats a classes. La idea d'aquest patró és analitzar les responsabilitats i assignar-les a la classe corresponent en funció de la informació necessària per a complir-les.

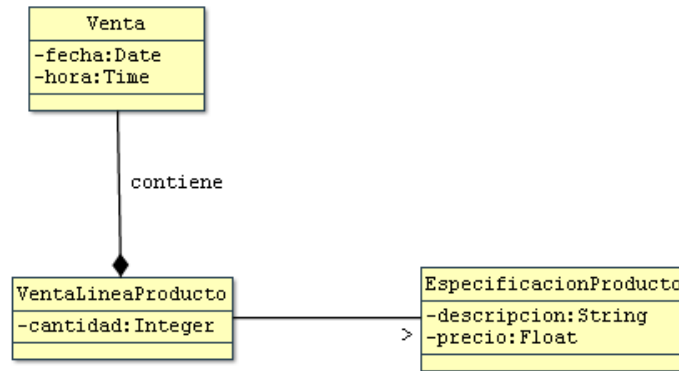


Figura 39. Exemple d'aplicació del patró expert

Aquest patró és utilitzat per totes les classes del nostre sistema per tal de mantenir un software coherent.

### 5.3.9. Controlador de transacció

Aquest patró fa d'intermediari entre la interfície i l'algoritme que implementa. Així, rep les dades de l'usuari i les envia a les diverses classes en funció del mètode cridat. A la nostra aplicació, els nostres controladors són cridats des de front per accedir als serveis.

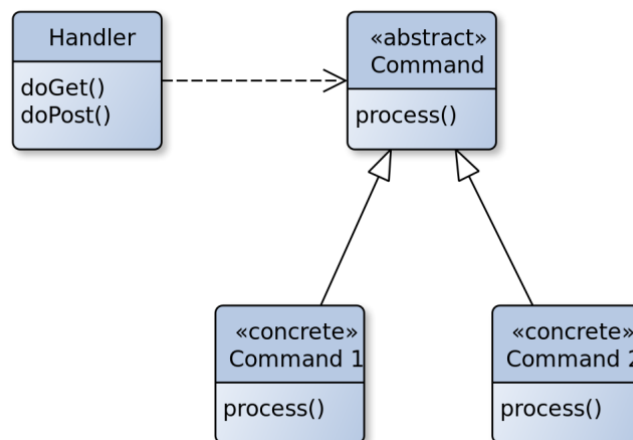


Figura 40. Exemple del disseny UML del controlador de transacció

### 5.3.10. Patró Plantilla

El patró plantilla és un patró de disseny de comportament que defineix l'esquelet d'un algoritme en un mètode que, en ser heretat per les subclasses, difereix en alguns dels seus passos.

Un exemple seria el següent:



**Figura 41. Patró Plantilla**

En un principi, és un mètode dissenyat per a macros, on cadascun implementa parts invariables i deixa oberts certs “placeholders” per a personalitzar les opcions.

Utilitzar aquest patró aporta una sèrie d'avantatges: primerament, deixa que les classes que s'implementen tinguin un comportament que pugui variar, evita la duplicació de codi i, per últim, permet evitar una sobrecàrrega polimòrfica: en comptes d'heretar el mètode i reescriure'l completament, permet canviar només aquelles parts canviabls.

En el nostre codi, fem ús del patró plantilla per a tal d'accedir a la informació reduïda d'una mascota.

### 5.3.11. Patró estratègia

Aquest és una patró de disseny de comportament que ofereix diversos algorismes al client i li dona la possibilitat de canviar entre ells segons les seves necessitats.

Un exemple:

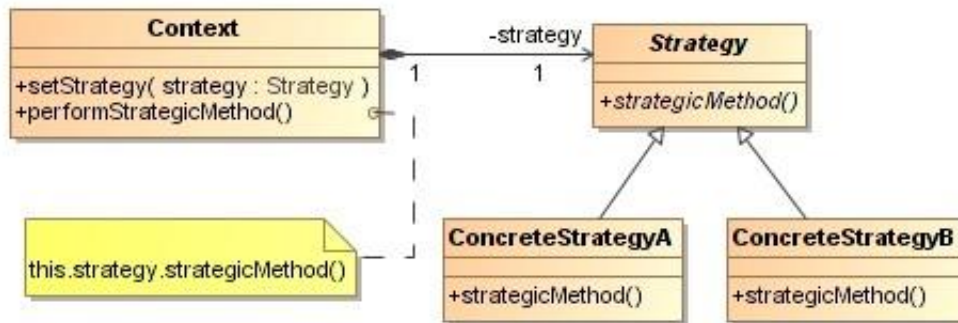


Figura 42. Patró estratègia

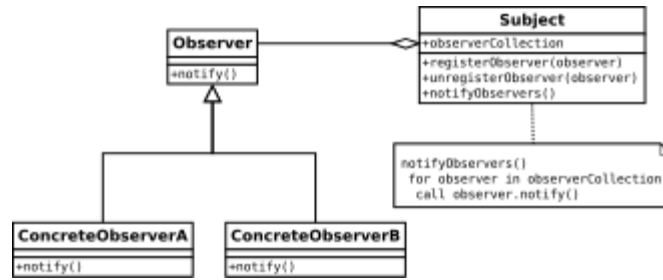
El patró estratègia és molt útil per a situacions en les que, per exemple, el client no necessita tots els algorismes en tots els cassos, o si tinguéssim diversos clients fent ús del mateix codi: en comptes de duplicar-lo, oferim estratègies diferents a cadascun.

A la nostra aplicació, fem us del patró estratègia per a alternar entre els diferents estadístics del barchart, al Front End.

### 5.3.12. Patró observador

El patró observador és un patró de disseny utilitzat en programació d'ordinadors per a observar l'estat d'un objecte en un programa. Està relacionat amb el principi d'invocació implícita.

Aquest patró es fa servir principalment per a implementar sistemes de tractament d'esdeveniments distribuïts. En alguns llenguatges de programació, els problemes tractats per aquest patró, són tractats a la sintaxi de tractament d'esdeveniments nativa. Aquesta és una funcionalitat molt interessant en termes de desenvolupament d'aplicacions en temps real.



**Figura 43. Patró observador**

L'essència d'aquest patró és que un o més objectes (anomenats observadors o escoltadors) són registrats (o es registren ells mateixos) per a observar un esdeveniment que pot ser llençat per l'objecte observat (el subjecte). L'objecte que pot llençar un esdeveniments generalment manté una col·lecció d'observadors.

En el nostre cas utilitzem aquest patró per escoltar els canvis en la base de dades, més concretament per escoltar canvis en els posts dels fòrums.



## 5.4. Model conceptual de les dades (Actualitzat)

A continuació es mostren els diagrames del model de dades implementat fins al moment i el que es pretén obtenir del projecte finalitzat.

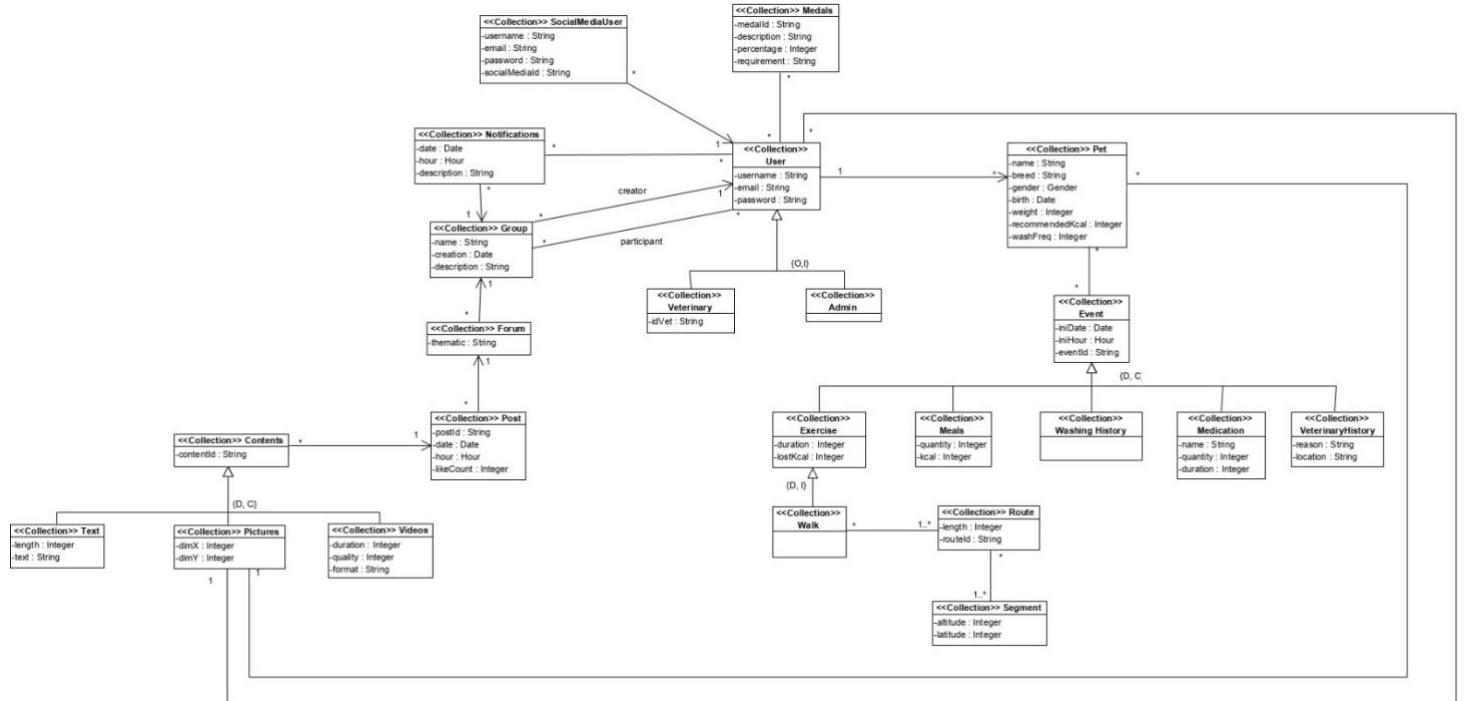


Figura 44. Model de dades final (Original)

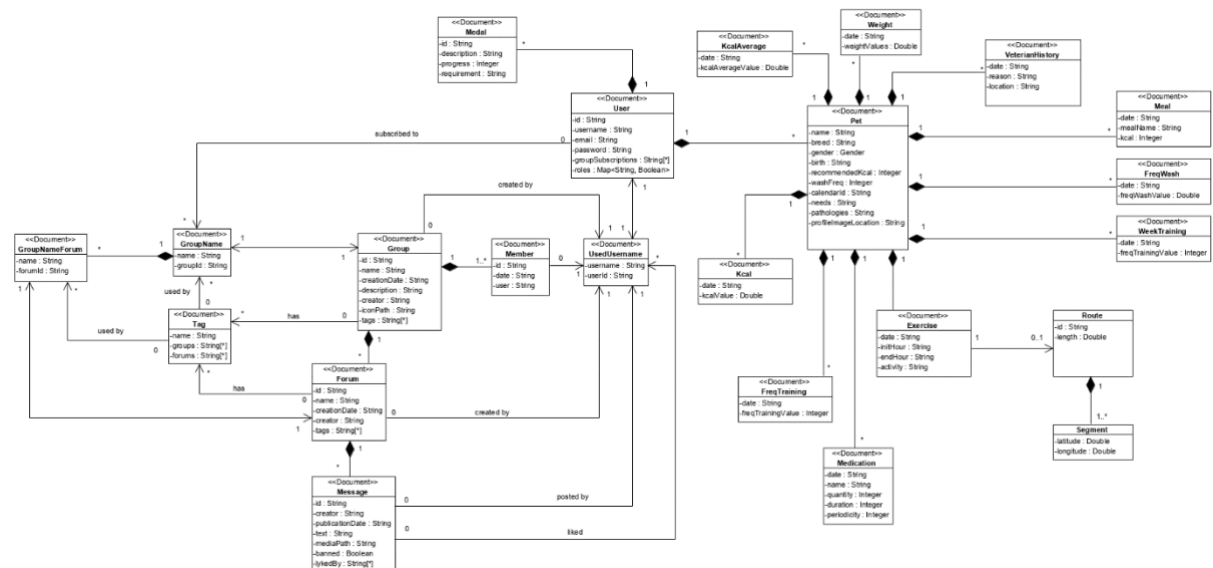


Figura 45. Diagrama de classes final (Versió sprint 2)

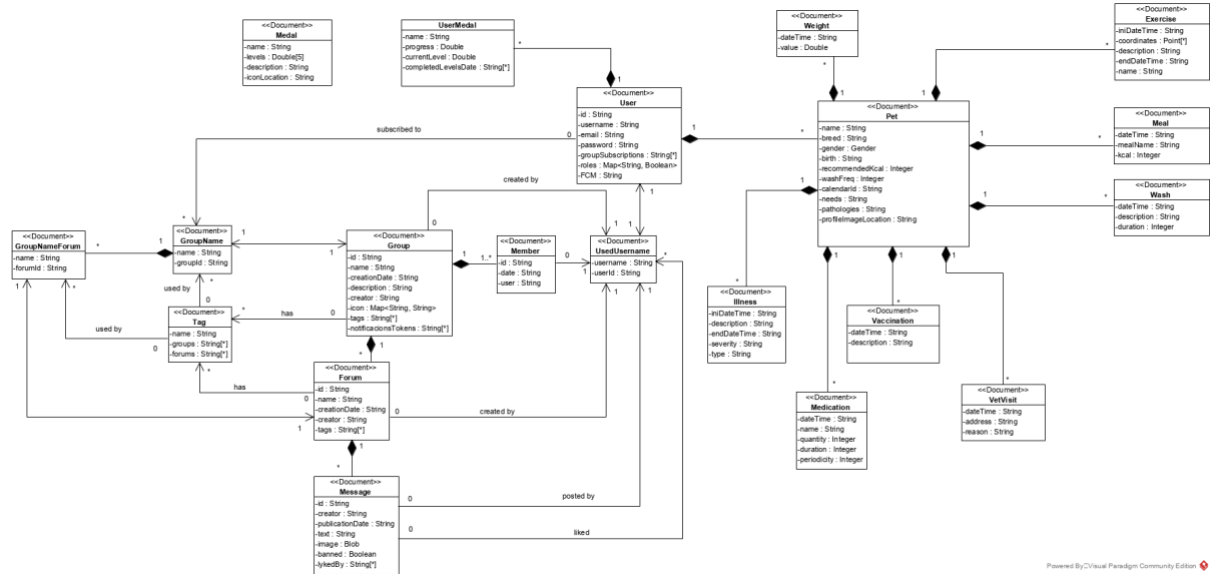


Figura 46. Diagrama de classes final (Nova versió)



Figura 47. Diagrama de classes actual (Original)

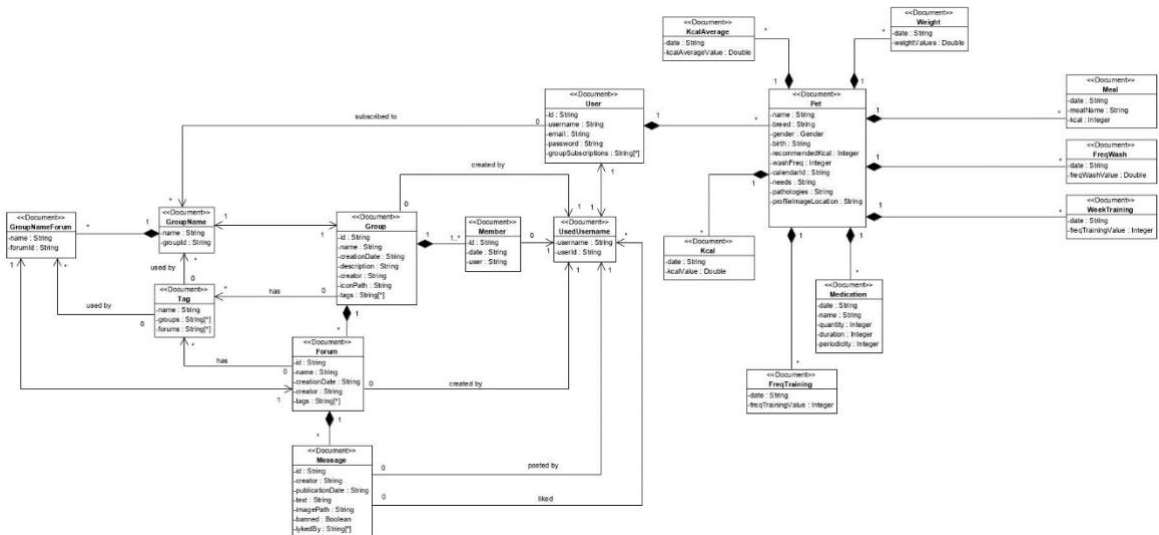


Figura 48. Diagrama de classes actual (Versió sprint 2)

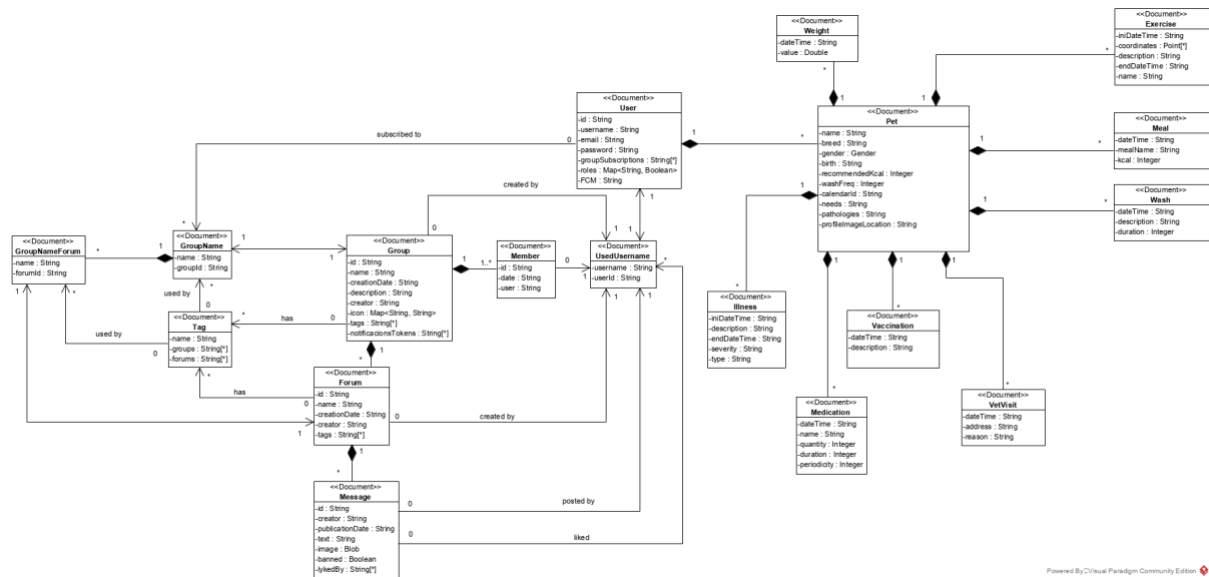


Figura 49. Diagrama de classes actual (Nova versió)

## 5.5. Altres aspectes tecnològics

A continuació es fa una breu explicació dels diferents aspectes tecnològics inclosos en el projecte.

### 5.5.1. Servidors

Disposem de dos servidors, un públic ubicat a Heroku i un altre de privat proporcionat per la FIB. El servidor públic s'utilitza per llençar les versions funcionals de l'aplicació, ja que aquest serà al que accediran els usuaris. D'altra banda, tenim el servidor de la FIB, on es van implementant les noves funcionalitats i es testegen abans de porta-les al servidor públic.



### 5.5.2. Base de dades

Utilitzem una única base de dades de tipus NoSQL gestionada per Firebase, que es un conjunt de serveis de *back-end* que proporciona Google. El versionat d'aquesta es fa mitjançant el servei de Firestore, un servei ofert per Firebase, per tant, és totalment transparent a nosaltres i se'n encarrega exclusivament el servei de Firestore.

Tenint en compte el temps del que disposem per fer el projecte, hem escollit aquesta solució degut a la seva facilitat d'implementació respecte de les altres alternatives. Un

altre motiu és el nostre enfoc dels accessos a dades de la nostra aplicació. Degut a que una de les principals característiques de la nostra aplicació seran els fòrums, la nostra aplicació tindrà una alta càrrega de lectures de la base de dades. Per aquest motiu, considerem que una base de dades NoSQL ens oferirà un major rendiment a l'hora de realitzar aquestes peticions.

### 5.5.3. Nombre de llenguatges

Per a la creació de l'aplicació Android utilitzem Java i XML, Java per la lògica de l'aplicació i XML per la interfície, mentre que per a la implementació del servidor s'utilitza únicament Java.

### 5.5.4. APIs



Cadascuna de les nostres principals funcionalitats conformarà una API, facilitant la comunicació entre les diferents parts de l'aplicació i també agilitzant el desenvolupament. Actualment disposem de dos APIs, una per a la gestió dels usuaris i una per a la gestió de les mascotes. Cal afegir, que actualment ambdues es troben empaquetades en una sola llibreria.

### 5.5.5. Frameworks

Primerament, utilitzem Spring Boot per crear el nostre servidor i les API REST ja que és un dels frameworks més utilitzats a Java i facilita tota la creació d'aquest.

En segon lloc, utilitzem JUnit per a la realització dels tests unitaris. Aquest ens permet testejar els diferents escenaris que es poden produir en l'execució dels mètodes d'una classe.

En tercer lloc, utilitzem Mockito per a la realització dels tests unitaris, que ens proporciona un conjunt de mètodes per a la implementació de mocks, facilitant així l'aïllament dels tests.

Finalment, utilitzem Espresso per a la realització dels tests instrumentals de l'aplicació Android, ja que ens proporciona un conjunt de mètodes per a simular les accions d'un usuari i automatitzar-les.

### 5.5.7. Desplegament

Per al desplegament del servidor fem servir Heroku, un servei de *hosting* que ens proporciona servidors en el núvol. Aquest ens permet enllaçar el repositori que conté la implementació del servidor, facilitant així que es puguin desplegar noves versions del servidor de manera automàtica, un cop ha passat tots els tests de GitHub Actions i Codacy.