UNIVERSITAT DE BARCELONA

Treball final del grau

D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Eina de govern SOA de la UPC

Autor: Pol Bieto Luengo

Director:	Josep Vañó Chic
Tutor a l'empresa:	Miguel González
Realitzat a:	UPCNet

Barcelona, 1 de febrer de 2019

Abstract

This project was created with the aim to contribute into the consolidation of UPC's SOA model (service-oriented architecture). This model and its definitive implementation represent a strategic goal within the ICT architecture of the UPC. Despite of that, one of the biggest deficiencies, until the development of this project, was to provide the model with a tool that could have effective agency and centralize all the information related to the web services offered by the UPC. So far, these control and government tasks have been almost done manually and not systematically, in contradiction with the philosophy of the model.

This challenge has been tackled using the latest technologies available in Java's web development ecosystem, paying special attention to an enterprise-oriented development in a continuous integration environment, with a strong DevOps culture. An agile methodology has also been used to face the development of the project, making it iterative and adaptable to the needs evolved. This agile development has been based on Scrum and Kanban, yet not implementing these frameworks in a strict mode, rather using those elements that have been proven to be more useful. Furthermore, it is important to remark that, is a user-centred design.

This software possess as main functionality the management of different types of application forms: application for publication of a web service, publication of a version of a web service and application for subscription of a version. It also has an expository part that we call catalogue and centralizes all available information on published services and versions. In addition, we have sections to manage the services published and the services subscribed. The software also has an administrative section, for managing the users and the states of the applications. Login is accessed through the CAS authentication of the UPC. The administrator can also pretend to be another user and view the application as if he was navigating.

Most of the objectives set up at the beginning have been successfully achieved. A stable product with well-defined features has been developed, going through all the design and implementation stages of software.

Resum

Aquest projecte neix amb la vocació d'aportar un granet de sorra a la consolidació del model SOA (arquitectura orientada als serveis, de les seves sigles en anglès) de la UPC. Aquest model i la seva implementació definitiva representa un objectiu estratègic dins de l'arquitectura de les TIC a la UPC. Malgrat això, una de les grans carències fins al desenvolupament d'aquest projecte, era la de disposar d'una eina que pogués fer efectiu aquest govern i centralitzes tota la informació relativa als serveis web que ofereix la UPC. Fins al moment, aquestes tasques de control i govern s'han fet de forma gairebé manual i no sistematitzada, en contradicció amb la filosofia del model.

S'ha afrontat aquest repte utilitzant les darreres tecnologies disponibles en el desenvolupament web en Java, donant especial importància al desenvolupament amb caràcter empresarial de l'aplicació i en un entorn d'integració continua amb un forta cultura DevOps. També s'ha utilitzat metodologia àgil per afrontar el desenvolupament del projecte, fent-lo iteratiu i adaptable a necessitats que han anat evolucionant. Aquest desenvolupament àgil s'ha basat en Scrum i Kanban, no fent una implementació estricta d'aquests marcs de treball sinó utilitzant aquells elements que han resultat ser més útils. També cal destacar un disseny centrat en l'usuari.

L'aplicació té com a principal funcionalitat la gestió de diferents tipus de formulari de sol·licitud: sol·licitud de publicació d'un servei web, de publicació d'una versió d'un servei web i sol·licitud de subscripció d'una versió. També disposa d'una part expositiva que anomenem catàleg i centralitza tota la informació disponible dels serveis i versions publicades. A més a més, disposem d'apartats per a gestionar els serveis publicats i els serveis subscrits. L'aplicació també té una part d'administració, on gestionar els usuaris i els estats de les sol·licituds. S'accedeix a l'aplicació a través de l'autentificació CAS de la UPC. L'administrador també pot simular ser un altre usuari i visualitzar l'aplicació com si aquest estigués navegant.

La majoria dels objectius plantejats a l'inici s'ha acomplit amb èxit. S'ha desenvolupat un producte estable i amb funcionalitats ben definides, passant per totes les etapes de disseny i implementació de programari.

Agraïments

Vull agrair, en primera instància, a la meva família, i molt especialment a la meva mare, per donar-me l'energia i la confiança per tirar endavant aquest projecte

També al meu tutor de la universitat, pels seus advertiments i comentaris que han estat presents al llarg del desenvolupament.

Per ultim, i no menys important, al meu tutor a l'empresa i a la meva cap d'equip per fer el paper de Product Owner i donar-me la confiança i l'autonomia necessaris per abordar aquest projecte.

Finalment, agrair a UPCNet la seva vocació de servei públic i la seva aposta pel programari lliure, molt necessari en els temps que corren avui en dia.

Índex de figures

Índex

1 Introducció

1.1 Motivació i context

Aquest treball final té com a objectiu consolidar els coneixements adquirits al llarg de meva trajectòria acadèmica i professional en el sector del disseny i el desenvolupament d'aplicacions web. En el darrer any he estat fent pràctiques a UPCNet, empresa que treballa per cobrir les necessitats tecnològiques de la Universitat Politècnica de Catalunya. Front a la idea de realitzar el treball de final de grau amb ells, vàrem estudiar quines necessitats tenia l'empresa i què podria ser útil. D'aquí va néixer la idea de realitzar l'eina de govern SOA de la UPC.

La UPC ofereix les seves dades a través d'una arquitectura orientada als serveis per tal de crear sistemes escalables, fàcilment modificables i integrables. D'altra banda, per tal d'acomplir amb aquest objectius, s'ha de garantir, a més a més de la infraestructura necessària per a donar-li suport i mantenir-ho, un centralització dels serveis disponibles i de tota la documentació i informació necessària per a poder donar-lis ús i poder realitzar-ne una gestió adequada. Deguda la naturalesa de la lògica del negoci a la que vol donar suport el programari que resulti d'aquest TFG, s'ha optat per desenvolupar una aplicació personalitzada, escalable i millorable en el temps.

Fins ara la centralització de les dades SOA s'ha realitzat en una pàgina web estàtica, on es recull el catàleg de serveis, les estadístiques d'ús, la documentació general de l'estructura SOA, la governança i la seva arquitectura, a més a més de la documentació específica dels serveis oferits. També s'ofereixen dos formularis a través dels quals poder sol·licitar la publicació de nous serveis o la subscripció als existents. Aquests formularis genèrics generen un tiquet de suport a l'aplicatiu d'Atenció als usuaris.

Al no disposar d'una eina integral que aglutini totes aquestes funcionalitats la gestió resulta feixuga i no sistematitzable, alhora que no permet un visió global de l'estat de l'arquitectura SOA i del seu ús, fet que contradiu la filosofia i la intencionalitat de proveir una arquitectura d'aquest tipus.

Així, aquest treball de final de grau pretén donar solució a les qüestions plantejades i, al mateix temps, fer ús de les ultimes tecnologies disponibles en tot el procés de desenvolupament del programari. Es vol posar en el centre del desenvolupament, la pràctica de metodologies àgils i un disseny centrat en l'usuari.

2 Descripció del projecte

Aquest projecte té com a finalitat el desenvolupament d'un programari web capaç de centralitzar la informació relacionada amb l'arquitectura SOA de la UPC i al mateix temps, poder gestionar-ne les accions associades al seu govern. En aquest darrer cas les principals funcions seran el procés de proveïment d'un nou servei o versió, la subscripció a un servei existent, el catàleg dels serveis disponibles, la gestió de les sol·licituds i els serveis, la notificació i publicació d'incidències i/o novetats i el monitoratge del seu l'ús. Aquest programari es basarà en un solució web ja que ha de permetre l'accés a qualsevol usuari de la comunitat UPC independentment de l'especificitat del maquinari que utilitzi.

2.1 Objectiu general

L'objectiu general no es altre que proveir d'un producte usable, fàcil de mantenir, escalable i amb prestacions ben definides per a l'empresa UPCNet per tal de millorar i facilitar la gestió de govern SOA de la UPC, d'igual forma que centralitzar-ne la informació i oferir un canal de comunicació i interacció còmode i àgil als usuaris de l'arquitectura.

2.2 Objectius específics

Els objectius específics propis del projecte són consolidar i aprofundir en els coneixements que fan possible el desenvolupament d'aplicacions web modernes, tant a nivell tècnic, referent a utilitzar les darreres tecnologies disponibles, com a nivell metodològic, al incorporar les eines més recents alhora d'afrontar el desenvolupament d'un projecte.

Concretant, a nivell tecnològic tenim:

- Aprenentatge i comprensió del framework Spring Boot i Java 8
- Disseny del model de dades i posada a punt en una base de dades PostgreSQL
- Ús de HTML5, CSS i JavaScrip per tal de dissenyar una interfície centrada en l'usuari. Domini del framework Boostrap 4.
- Utilització de serveis web.
- Aprenentatge de l'entorn JUnit
- Eines DevOps com GitLab, Maven, Docker i Rundeck.

A nivell de metodologia tenim:

- Desenvolupament guiat per proves (Test-driven development)
- Filosofia SCRUM i Kanban
- Filosofia DevOps i integració continua.

3 Metodologia

La metodologia pren especial rellevància en aquest TFG. El desenvolupament de projectes a UPCNet s'afronta des de una perspectiva àgil i així s'encara aquest projecte, que vol ser també, una posada en pràctica d'aquest tipus de tècniques.

En concret s'ha fet servir una metodologia basada en Scrum i també en Kanban. A la figura 1 podem veure el cicle de desenvolupament propi d'aquest tipus de metodologies.

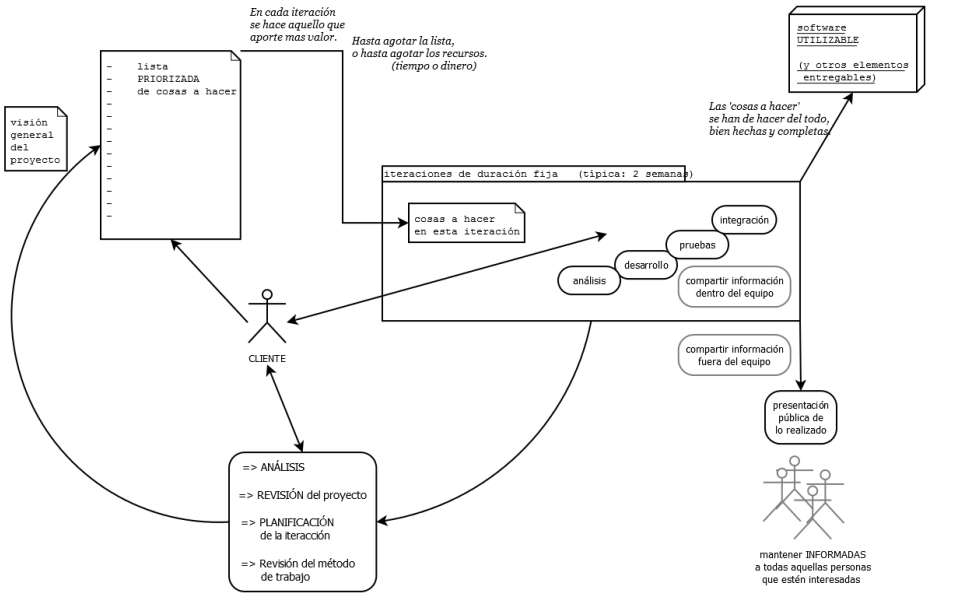

Figura 1: Esquema general de metodologia àgil (font: es.wikipedia.org)

Aquesta forma d'encarar un projecte inverteix l'enfocament en cascada clàssic. Mentre que en aquest darrer el procés d'anàlisis de requisits funcionals i no funcionals, d'igual forma que l'anàlisi de riscos, es produeix en una fase prèvia al desenvolupament, en entorns àgils aquests anàlisis es produeix en cada cicle d'iteració. Es prefereix optar per aquesta metodologia per que s'ha considerat que l'abstracció que suposa un anàlisis previ, complet i profund del producte a desenvolupar no es eficient ni realista, ja que sovint els requeriments del producte van canviant al llarg del seu desenvolupament, o es concreten en aquest. L'únic anàlisis de riscos real és aquell que es produeix en el moment pròxim a la implementació. En aquest cas, tenim un marc general al voltant del qual desenvolupar l'aplicació, però s'aniran definint i concretant les seves característiques a mesura que avanci la implementació i abordem les diferents necessitats.

Com ja s'ha dit, mentre que l'anàlisi de requeriments es fa en la definició dels sprints, prioritzant allò que més valor de negoci aporta al producte, l'anàlisi de riscos es materialitza alhora d'afrontar el seu desenvolupament. Així, un cop definit l'objectiu i les històries d'usuari del sprint, es procedeix a dissenyar les proves que guiaran el desenvolupament. Aquestes proves són la materialització concreta

de l'anàlisi de riscos per a una funcionalitat. Amb la retrospectiva de l'sprint, s'eixampla aquest anàlisis de riscos aportant la perspectiva del Product Owner que coneix bé la lògica del negoci i els diferents casos d'ús.

També s'utilitza Kanban com a metodologia complementaria per a que el Product Owner vagi afegint requeriments i/o millores en el desenvolupament del sprint. Els items del Kanban són contextualitzats i convertits en històries d'usuaris del sprint actual o queden com a base per a la planificació del següent sprint. Amb tot això s'ha de puntualitzar que és molt difícil utilitzar el model scrum al cent per cent en un equip que a la fi estarà conformat per un únic desenvolupador. Així doncs, aquest projecte pretén utilitzar una metodologia àgil i utilitzar aquells elements que realment aportin valor i siguin útils de les diferents filosofies existents. Al que més importància li donarem serà a la comunicació de l'equip, l'avaluació periòdica a partir dels resultats dels sprints i el desenvolupament incremental i iteratiu.

A més a més de cal mencionar l'òptica DevOps que també vol recollir el projecte. La idea principal es crear una arquitectura i una dinàmica de treball que permeti integrar les noves implementacions de manera continua i segura. Així, un cop realitzades les proves i el desenvolupament, tant aviat com sigui possible pujarem els canvis al repositori de Git remot. Aquí s'iniciarà el procés d'integració, amb les proves i les mètriques del codi. Si el canvis superen aquest procés satisfactòriament, es procedirà a publicar la nova versió als diferents servidors en els que estigui desplegat el projecte. A la secció 5.5 de compilació i desplegament concretarem i explicarem aquest procés.

4 Desenvolupament del projecte

En aquesta secció es procedirà a explicar el procés de desenvolupament integrat del programari. No es realitzarà una separació explícita de disseny, anàlisis de requisits i riscos, sinó que s'explicarà fase a fase al llarg del desenvolupament com s'ha tractat cada punt en les respectives iteracions. L'esquema general de cada sprint té com a primer pas definir les històries d'usuari que més valor de negoci aporten en cada moment; a continuació, es puntuen segons la complexitat o el temps previst estimat per a resoldre-les. En aquest projecte s'ha decidit puntuar les històries seguint la successió de Fibonacci, per tal de poder diferenciar les tasques senzilles de les realment difícils. Els sprint són de tres setmanes aproximadament, i en cada un s'aborden històries per un valor total d'entre 30 i 40 punts.

4.1 Punt de partida (1/10 - 1/11)

Es té una primera reunió amb el Product Owner on s'explica el context des del que es parteix i les necessitats bàsiques. Aquestes venen definides a partir de la següent llista de desitjos:

Com a subscriptor vull:

- Trobar serveis web que em proporcionin els atributs que desitjo
- Saber sobre els nous serveis
- Contactar persones que hagin subscrit el mateix servei
- Contactar el proveïdor del servei en qualsevol moment
- Que se'm notifiqui si hi ha problemes en el servei que subscric
- Que se'm notifiqui si hi ha noves versions del servei que subscric
- Fer proves del servei abans de subscriure'l
- Subscriure el servei introduint el mínim d'informació
- Sentir-me acompanyat en tot el procés de subscripció

Com a proveïdor vull:

- Donar d'alta el servei amb la introducció mínima d'informació
- Assabentar-me en el moment que un subscriptor no està complint els terminis del contracte
- Veure si el meu servei està essent utilitzat
- Rebre les peticions de millora dels serveis

- Rebre les consultes sobre el meu servei
- Veure tots els contactes SOA

Com a govern SOA vull:

- Poder notificar a tots els contactes SOA
- Donar la informació SOA en el moment que es necessita
- Poder veure les incidències SOA
- Poder veure l'ús que es fa dels serveis SOA
- Poder comparar la situació actual amb la de fa uns mesos
- Rebre les peticions que realitzen altres usuaris
- Veure les preguntes sense resoldre i quan fa que estan sense resoldre.

D'aquí es valora l'abast del projecte, el temps disponible i la capacitat per a desenvolupar-lo. Així, es prioritza quins són els items que més valor de negoci aporten al futur aplicatiu, i es decideix seleccionar només alguns dels items:

Com a subscriptor vull:

- Saber sobre els nous serveis
- Que se'm notifiqui si hi ha problemes en el servei que subscric
- Que se'm notifiqui si hi ha noves versions del servei que subscric
- Subscriure el servei introduint el mínim d'informació
- Sentir-me acompanyat en tot el procés de subscripció

Com a proveïdor vull:

- Donar d'alta el servei amb la introducció mínima d'informació
- Veure si el meu servei està essent utilitzat
- Veure tots els contactes SOA

Com a govern SOA vull:

- Poder notificar a tots els contactes SOA
- Poder veure les incidències SOA
- Poder veure l'ús que es fa dels serveis SOA

- Rebre les peticions que realitzen altres usuaris

Un cop acotada la magnitud del projecte va començar-se la fase documentació i aprenentatge. D'igual forma es va crear l'entorn de treball, vinculant-lo amb GitLab, es va realitzar la configuració inicial del projecte i es va preparar l'entorn de proves. També es va definir la versió inicial de la base de dades, atenent a les necessitats del primer sprint. A la figura 2 veiem l'esquema inicial.

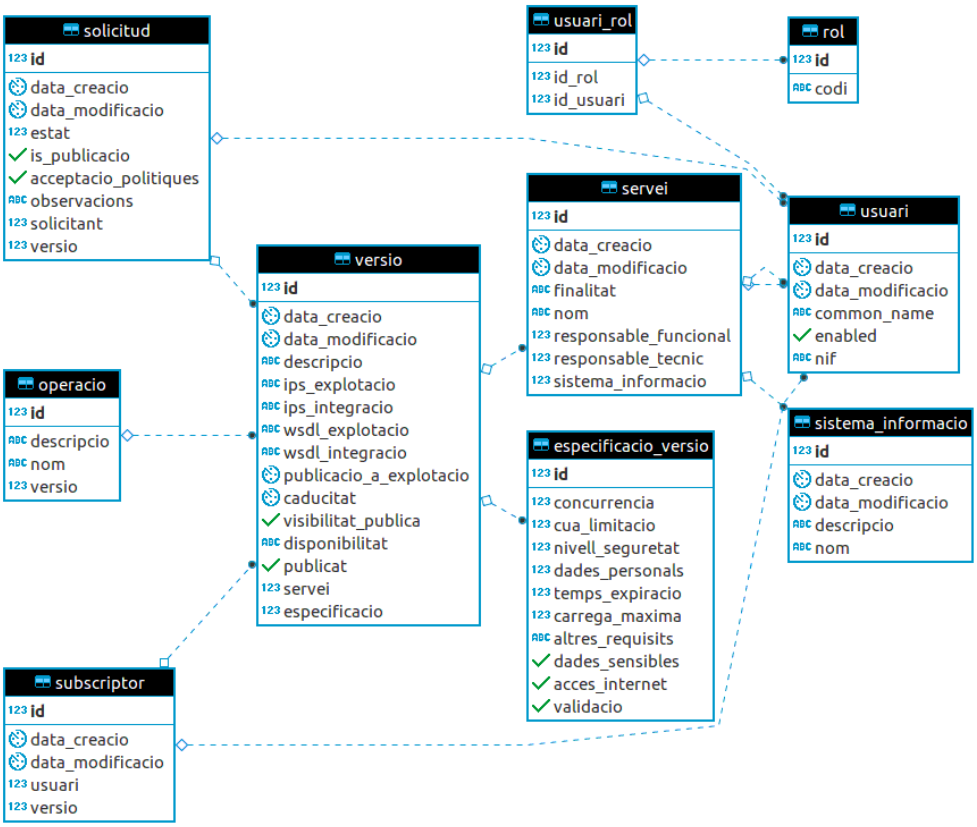

Figura 2: Primera versió de la base de dades

També es van crear un disseny bàsic per poder començar a treballar. A les següents figures es pot veure la proposta.

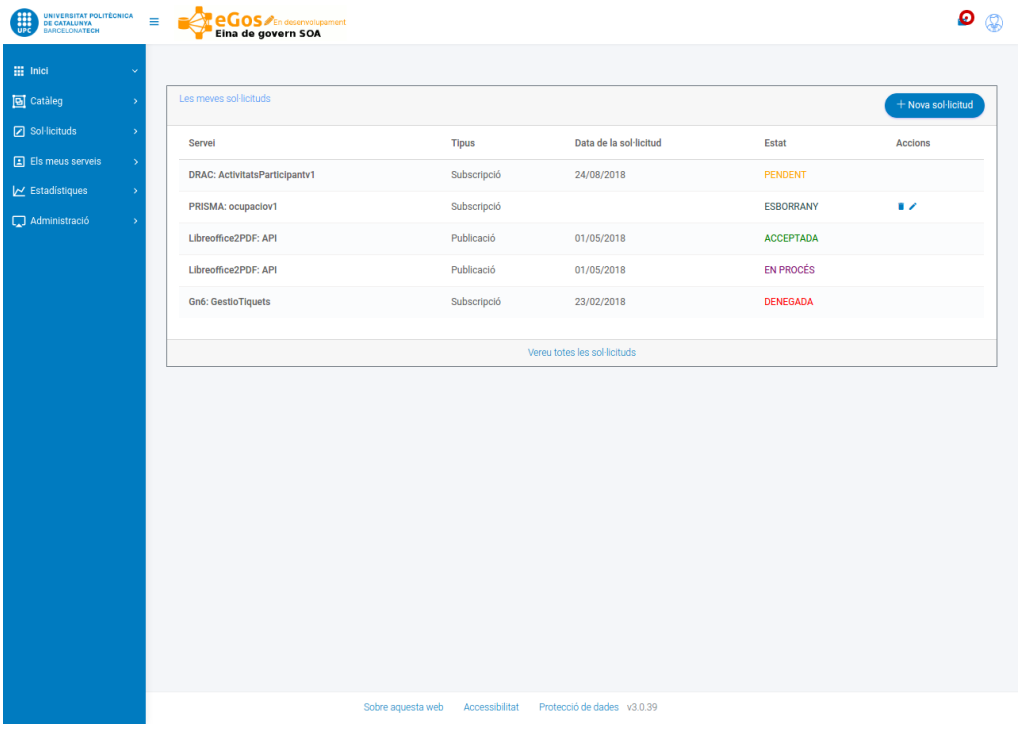

Figura 3: Prototip de la pàgina inicial

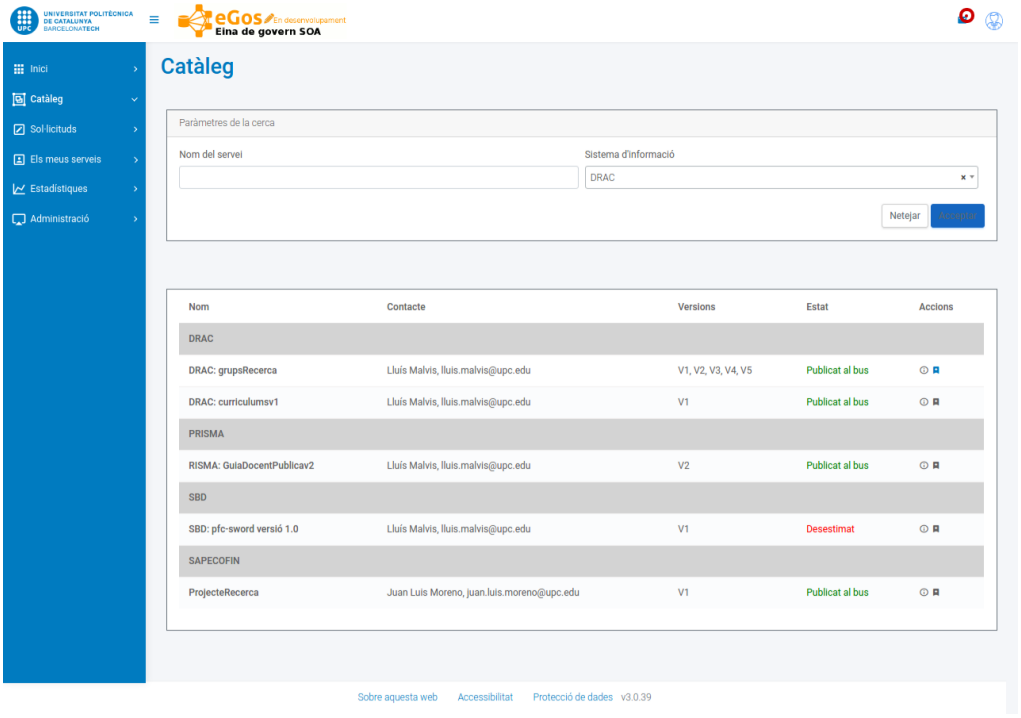

Figura 4: Prototip pel catàleg

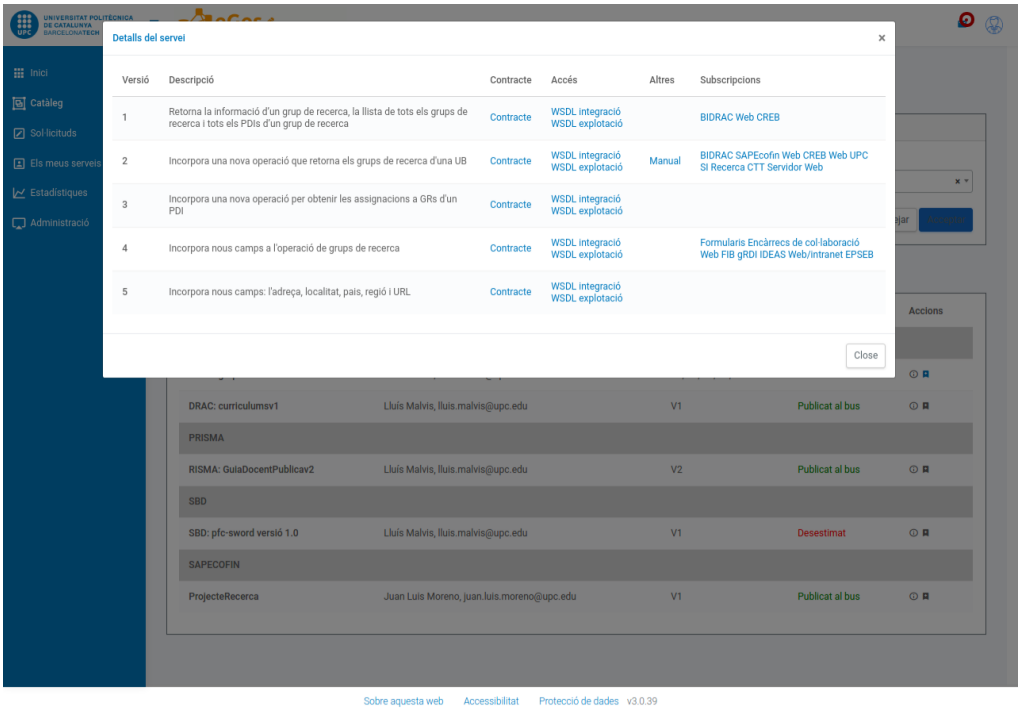

Figura 5: Prototip del detall del catàleg al clicar en un dels serveis

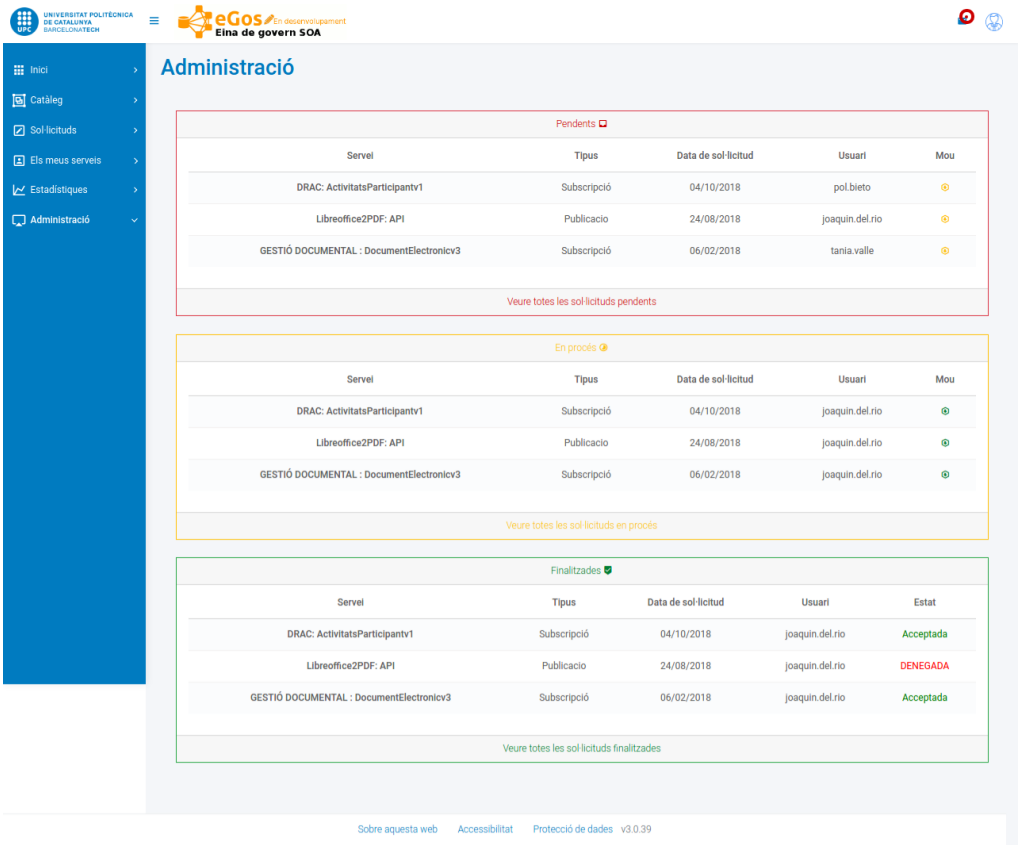

Figura 6: Prototip de plana inicial de l'administració

A l'anàlisi de riscos es contempla:

- La transformació i la correspondència entre l'objecte que dona suport al formulari de la sol·licitud i la pròpia sol·licitud a base de dades pot ser una gran font d'errors i de fallades. Per a minimitzar aquest risc es decideix utilitzar una eina de generació de codi ideada per aquest cas d'ús. S'escolleix **MapStruct** per a fer aquesta correspondència entre DTO (data transfer object) i la entitat, que representa la relació a base de dades.
- Les validacions també s'analitzen com a possible font de riscos. Per minimitzar-los, s'aposta per utilitzar tant validacions al front-end, per a que l'usuari pugui interactuar d'una forma natural amb els camps, com a back-end, per a garantir la seguretat i la integritat de les dades.
- Es decideix delegar el login al CAS de la UPC, per oferir una autentificació estàndard i també per desocupar-nos d'aquesta funcionalitat.

4.2 Primer sprint (1/11 - 22/11)

El primer sprint comença un cop l'entorn de desenvolupament està preparat. L'objectiu d'aquest sprint es implementar l'entorn d'usuaris i realitzar el formulari de sol·licitud de publicació d'un servei web. Es decideixen abordar les següents històries d'usuari, amb un valor total de **37 punts**:

- Login usuari mitjançant el cas de la UPC **(2 punts)**

  Com a usuari

  Vull logar-me al sistema utilitzant l'autentificació UPC

  Per poder utilitzar-lo

- Rol d'administració **(3 punts)**

  Com a usuari administrador

  Vull logar-me al sistema

  Per exercir les tasques associades a l'administració

- Suplantar usuari **(3 punts)**

  Com a usuari administrador

  Vull simular ser un altre usuari

  Per poder interaccionar amb el sistema com si fos aquell usuari

- Formulari sol·licitud publicació **(13 punts)**

  Com a usuari

  Vull accedir al formulari de sol·licitud de publicació

  Per poder introduir-hi les dades pròpies del servei que vull donar d'alta

- Llista de les meves sol·licituds **(3 punts)**

  Com a usuari

  Vull veure la llista de les meves sol·licituds

  Per poder-ne fer un seguiment adequat

- Validacions formulari sol·licitud de publicació **(8 punts)**

  Com a usuari que vol publicar un nou servei

  Vull que el formulari mostri els errors de validació

  Per poder enviar una sol·licitud vàlida a ser considerada

- Cercador de persones als camps responsable **(5 punts)**

  Com a usuari que vol publica un servei

  Vull un cercador de persones als camps responsable funcional i responsable tècnic

  Per no haver de recordar i escriure el common.name de la persona

En la retrospectiva d'aquest sprint es van fer les següents valoracions i observacions de les tasques realitzades:

- Tant en la cerca del responsable tècnic com funcional, si les persones buscades no tenen segon cognom, que no apareix-hi el text 'null'
- Separar la relació del responsable amb un Usuari de l'aplicació i crear la taula Persona per emmagatzemar-hi aquestes dades.
- El sistemes d'informació han de sortir ordenats alfabèticament.
- Canviar el nom del camp "IP's d'accés explotació accés intranet"per "IP's d'accés explotació". D'igual forma amb "IP's d'accés test accés intranet"

La resta de consideracions van servir per a prioritzar i re-definir el backlog del següent sprint.

A l'anàlisi de possibles riscos es contempla el següent:

- Es necessari usar una eina de migració dels canvis a base de dades. Donat que estem treballant de forma àgil i la base de dades pot patir canvis a cada iteració, es necessari poder migrar estats anterior a nous models, fent aquest procés automàtic i estandarditzat. Es decideix incorporar FlyWay al projecte.
- Al haver-hi tants camps al formulari de sol·licitud, es possible que l'usuari es perdi i no es vegi capaç de completar-lo. Per minimitzar això serà necessari afegir-hi ajudes contextuals i valors per defecte que facilitin la entrada d'informació. En la mateixa línia, s'ha de poder gestionar les operacions o el sistema d'informació des de la mateixa pantalla.

Pel que fa a a l'evolució de la feina al llarg del spring, a la figura 7 podem veure el burndown chart.

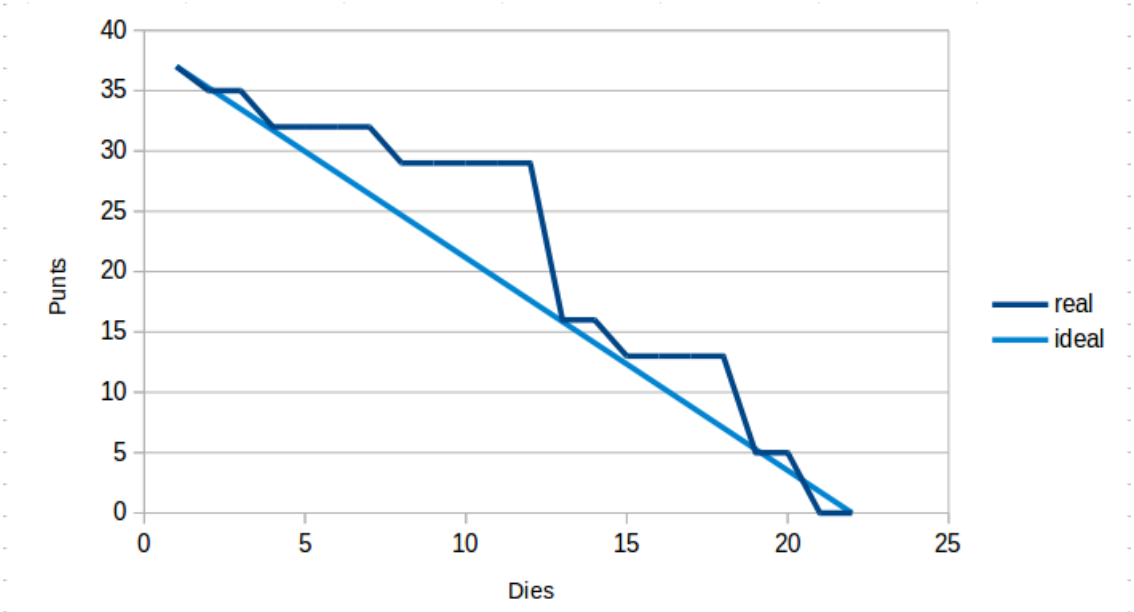

Figura 7: Burndown chart del primer sprint

Veiem que un punt d'inflexió notable sobre el dia 13, que coincideix amb el moment en que s'acaba la tasca de realitzar el formulari de sol·licitud de publicació d'un nou servei. També veiem, doncs, que al principi de la iteració es realitzen les tasques més fàcils, com el login o el rol de l'administrador, i no és fins la meitat final que s'aborda tota la complexitat del formulari. Es fa la reflexió que de cara al següent sprint cal equilibrar una mica la realització de tasques complexes amb d'altres més fàcils i ràpides.

4.3 Segon sprint (26/11 - 17/12)

En aquest sprint es marca com a fita implementar el formulari de publicació d'una versió d'un servei i el de subscripció, d'igual forma que completar el de publicació d'un servei. S'aborden les següents històries d'usuari per un valor total de **41 punts**:

- Formulari sol·licitud de subscripció **(13 punts)**

  Com a usuari

  Vull accedir al formulari de sol·licitud de subscripció

  Per poder introduir-hi les dades pròpies del servei al que vull subscriure'm

- Validacions formulari sol·licitud de subscripció **(8 punts)**

  Com a usuari que vol subscriure's a un nou servei

Vull que el formulari mostri els errors de validació

Per poder enviar una sol·licitud vàlida a ser considerada

- Afegir gestió d'operacions al mateix formulari de publicació. **(8 punts)**

  Com a usuari que vol publicar un servei

  Vull gestionar les operacions del servei en el formulari de publicació

  Per poder crear-ne de noves o eliminar-les sense haver de canviar de pantalla

- Diferència entre publicació de nou servei i publicació de nova versió. **(5 punts)**

  Com a usuari que vol realitzar una sol·licitud de publicació

  Vull dos formularis diferenciats entre nou servei i nova versió

  Per no haver d'introduir informació que ja està al sistema

- Afegir valors per defecte i ajudes contextuals a tots els formularis. **(2 punts)**

  Com a usuari que vol fer una sol·licitud

  Vull que se'm mostrin ajudes contextuals i valors per defecte als formularis

  Per tal de sentir-me acompanyat i introduir el mínim d'informació possible

- Donar d'alta nous sistemes de la informació **(5 punts)**

  Com a usuari

  Vull donar d'alta nous sistemes de la informació

  Per tal de poder triar-los a les sol·licituds

Per poder abordar aquest canvis s'ha modificat la base de dades. En concret s'ha fet el següent:

1. Afegir la relació persona i canviar el camp responsable_funcional i responsable_tecnic de la relació servei, per a que enlloc d'apuntar a la taula d'usuaris apuntin a aquesta nova. D'igual forma es crea una referència d'usuari a persona.

2. Afegir la entitat solicitud_subscripcio i solicitud_subscripcio_operacions, que manté la relació entre la subscripció i les operacions del servei que es volen utilitzar. Així, solicitud_subscripcio es configura com una especialització de la la entitat solicitud, ja que hi ha una relació 1 a 1 única entre solicitud_subscripcio i solicitud.

3. Es canvia el camp is_publicacio de solicitud per tipus_solicitud, que es correspon amb una enumeració on podem diferenciar entre "Sol·licitud de publicació d'un nou servei", "Sol·licitud de publicació d'una nova versió d'un servei"o "Sol·licitud de subscripció"

Amb aquests canvis l'estructura de la Base de dades queda tal que així:

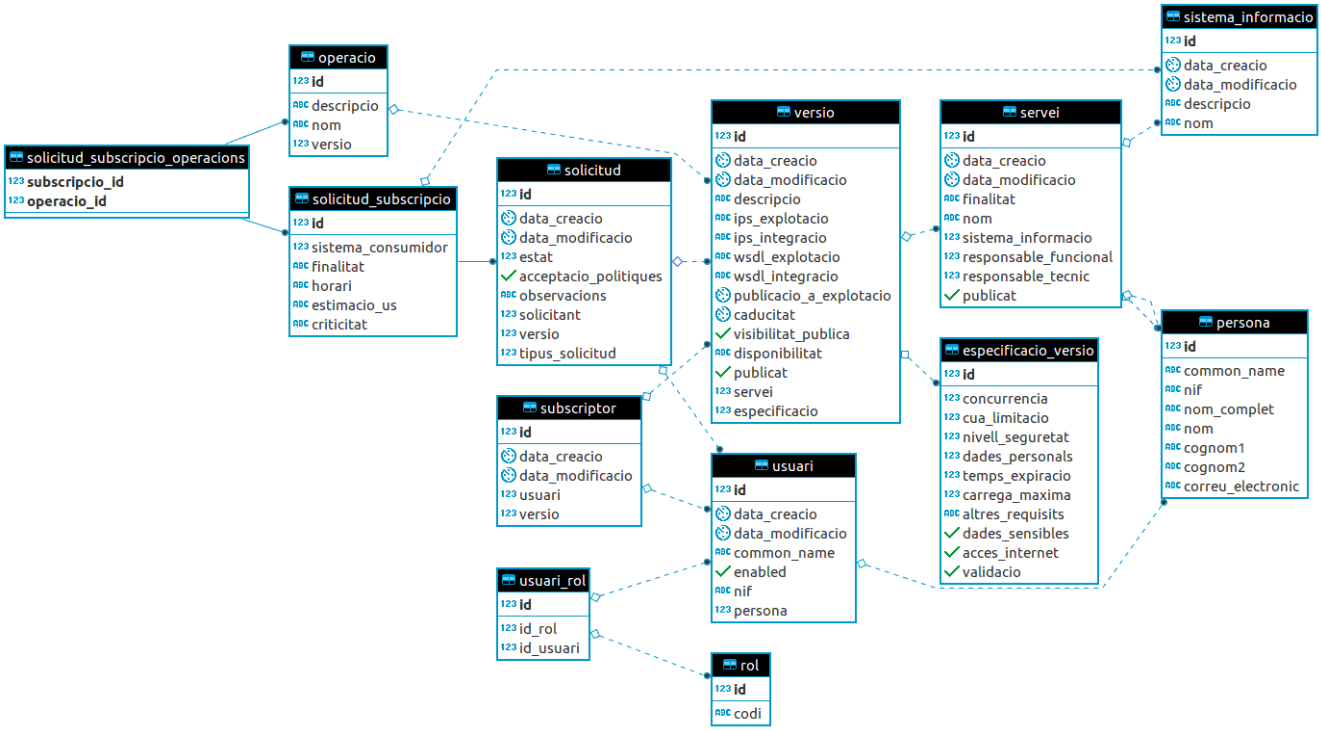

Figura 8: Segona versió de la base de dades

En la retrospectiva d'aquest sprint es van fer les següent valoracions i observacions:

- A les vistes dels formularis s'han de marcar els camps obligatoris abans de la validació del formulari.
- Afegir un camp a solicitud_subscripcio que apunti a sistema_informacio i que s'anomeni sistema_consumidor.
- Que s'han de poder de gestionar els usuaris des d'una part d'administració i poder atorgar permisos d'administració.
- Que s'han de poder gestionar les sol·licituds des d'una part d'administració on s'ha de poder canviar l'estat de la sol·licitud. Aquest canvis han de tindre efecte sobre l'estat de la versió.
- Que a la secció "Serveis que gestiono"hi han d'aparéixer els serveis dels quals sóc responsable funcional o responsable tècnic.
- Que a la secció "Subscripcions que gestiono"han sortir els serveis dels quals hi ha subscripcions on hi apareix-ho com a "responsable funcional". S'ha de crear aquest camp, responsable_funcional.

Pel que fa a a l'evolució de la feina pendent al llarg del sprint, a continuació podem veure la burndown chart.

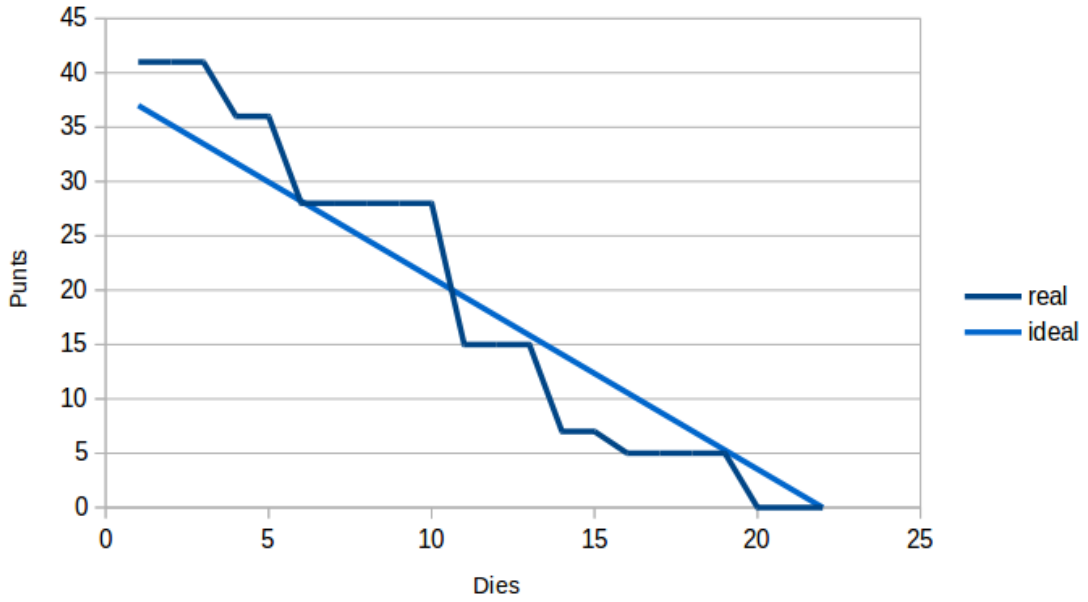

Figura 9: Burndown chart del segon sprint

En aquesta iteració s'ha intentat balancejar una mica més la càrrega de treball. D'igual forma la implementació del formulari de subscripció torna a generar un comportament en el gràfic similar al del anterior sprint. Es fa la reflexió que potser hauria estat preferible separar la història d'usuari en diverses històries per a abordar-les de forma més incremental.

4.4 Tercer sprint (17/12 - 7/1)

L'objectiu d'aquest sprint, un cop s'han implementat els diferents formularis, que són la font d'entrada de dades principal, es decideix apostar per la visualització i estructuració d'aquestes dades. Així, es vol abordar les vistes de les diferents entitats i afegir una part bàsica d'administració per a poder gestionar les sol·licituds i els usuaris. S'aborden les següents històries amb un valor total de **41 punts**.

- Serveis que gestiono **(2 punts)**

  Com a usuari

  Vull veure una llista dels serveis que gestiono

  Per saber en quins sóc responsable funcional o responsable tècnic

- Subscripcions que gestiono **(2 punts)**

  Com a usuari

Vull veure una llista de les subscripcions que gestiono

Per saber en quines hi apareix-ho com a responsable funcional.

- Catàleg **(5 punts)**

  Com a usuari

  Vull veure una llista de totes les versions publicades i organitzades per sistema d'informació

  Per saber quins serveis s'estan oferint

- Cerca del catàleg **(5 punts)**

  Com a usuari

  Vull cercar pels camps "nom del servei", "sistema d'informació", "responsable funcionalï "descripció de la versió"

  Per poder trobar el servei que m'interessa dins del conjunt de serveis

- Detall del servei **(5 punts)**

  Com a usuari

  Vull veure el conjunt d'informació d'un servei

  Per poder conèixer totes les seves característiques.

- Detall de la versió **(5 punts)**

  Com a usuari

  Vull veure el conjunt d'informació del detall d'una versió i les seves operacions

  Per poder conèixer totes les seves característiques.

- Administració de les sol·licituds **(8 punts)**

  Com a usuari administrador

  Vull poder canviar l'estat de les sol·licituds entre els següents: "esborrany", "procés", "pendent de completar", "acceptadaï "descartada"

  Per poder saber en quin punt es troba la sol·licitud en el seu cicle de vida

- Administració dels usuaris **(5 punts)**

  Com a usuari administrador

  Vull poder veure una llista d'usuaris

  Per poder localitzar-ne un i atorgar-li o revocar-li permís d'administrador

- Històric de la sol·licitud **(3 punts)**

  Com a usuari

  Vull veure un històric dels canvis d'estat d'una sol·licitud

  Per poder saber com va el procés i si es requereix d'intervenció

- Manual de la versió **(1 punts)**

  Com a usuari

  Vull veure un document PDF associat a cada versió que descrigui el seu comportament i ús

  Per a saber com funciona

Per poder abordar aquests canvis es van fer les següents modificacions a la base de dades:

1. Es canvia el camp booleà 'publicat' de la relació versió, per un camp estat que fa referència a la enumeració EstatVersio que pren valors en relació a l'estat de la sol·licitud.
2. S'afegeix un camp bytea a especificacio_versio per poder-hi desar el PDF del manual.
3. S'afegeix un camp número a versio per poder saber quin número de versió és.
4. S'afegeix el camp responsable_funcional a solicitud_subscripcio que apunta a la taula persona.
5. S'afegeix la relació historic_solicitud.
6. S'elimina la taula subscriptor ja que la informació queda recollida a solicitud_subscripcio

El resultat d'aquesta nova versió de l'esquema de la base de dades es pot veure a la figura 10.

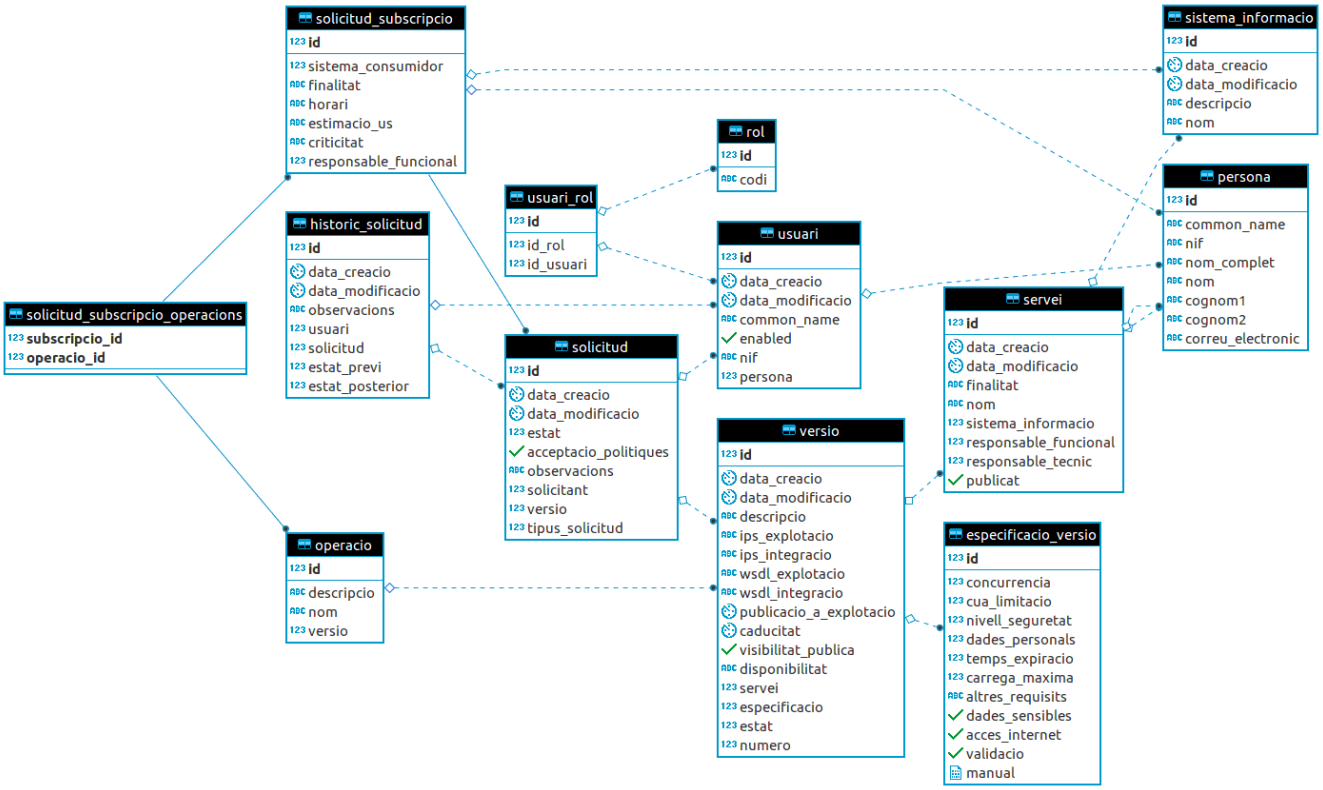

Figura 10: Tercera versió de la base de dades

A l'anàlisi de riscos destaquem:

- S'ha de limitar la mida del manual per evitar problemes d'espai a disc o de temps grans de transferència.
- El número de la versió s'ha d'assignar quan la versió es publica, no abans, per evitar incoherències en quant al número que faria confondre a l'usuari.
- S'han de protegir alguns camps exposats en els DTO afegint validacions i comprovacions globals en els mappers.

Pel que fa a a l'evolució de la feina pendent al llarg del sprint, a continuació podem veure la burndown chart.

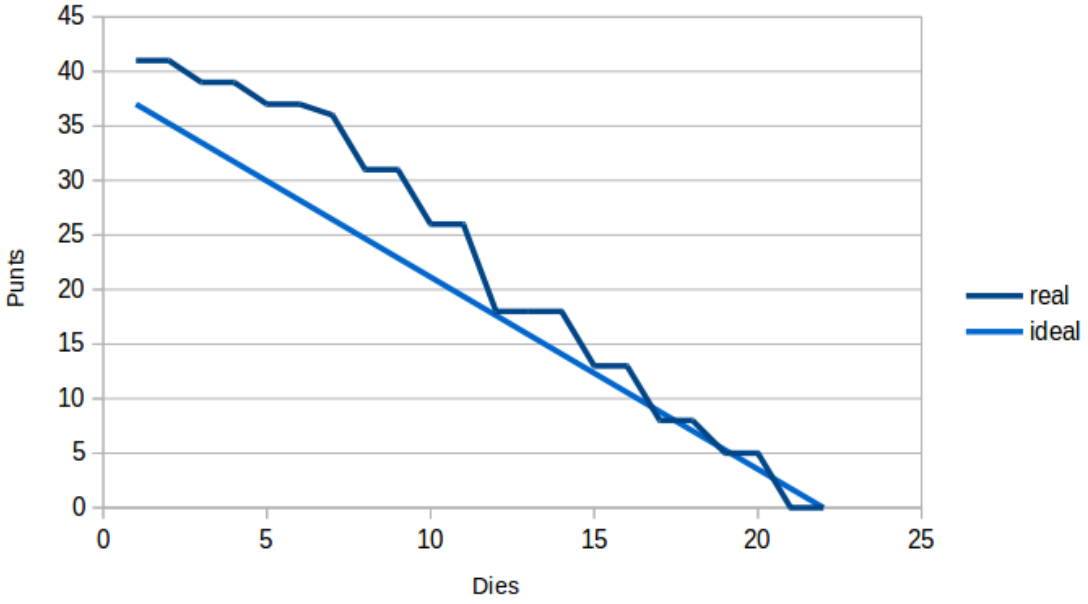

Figura 11: Burndown chart del tercer sprint

Veiem que al tindre un major nombre de tasques amb ponderacions més ajustades el descens real és més suau que en els anteriors casos. D'altra banda s'aprecia que el gruix de feina s'ha abordat a la segona meitat de la iteració. En aquest cas es lògic ja que coincideix amb les vacances de Nadal, on hi vaig poder dedicar-hi força més temps.

4.5 Quart sprint (7/1 - 22/1)

En el darrer sprint review es decideix prioritzar la seguretat i l'estabilitat de l'aplicació en contraposició a afegir-hi noves funcionalitats. En l'actual fase del projecte, i amb la intenció de començar a afegir dades reals a l'aplicació en pre-producció es considera que el que aporta més valor en aquesta etapa és l'estabilitat de la aplicació i millores en el disseny. Així doncs, s'aborden històries per un valor de **32 punts** en un sprint que es decideix que sigui lleugerament més curt.

- Veure sol·licituds **(8 punts)**

  Com a usuari

  Vull veure els formularis de sol·licitud encara que no puguin ser editats

  Per poder accedir en tot moment a les dades que he introduït

- Pre-carregar els camps del formulari de publicació d'una nova versió **(8 punts)**

  Com a usuari

  Vull veure els valors de la darrera versió en el formulari de publicació

  Per a minimitzar la quantitat d'informació que s'ha d'afegir

- Confirmació al esborrar sol·licitud **(2 punts)**

  Com a usuari

  Vull que l'aplicació em demani confirmació al esborrar una sol·licitud

  Per a no esborrar-ne una per error

- Gestió dels estats de la sol·licitud **(8 punts)**

  Com a usuari administrador

  Vull un canvi coherent dels estats de la sol·licitud

  Per poder gestionar el cicle de vida d'aquestes i de les versions associades

- Estat obsolet de la versió **(3 punts)**

  Com a usuari administrador

  Vull poder marcar una versió com a obsoleta

  Per deixar constància de que ja no es manté

- Camp publicat al bus a la versió **(2 punts)**

  Com a usuari

  Vull poder marcar si la versió està al bus

  Per tindre el màxim d'informació possible de la versió

- Estadístiques **(1 punts)**

  Com a usuari

  Vull veure un enllaç a les estadístiques de l'ús dels serveis web

  Per centralitzar la informació dins de l'aplicació

A la retrospectiva del sprint és fa la reflexió de que les funcionalitats base del cor de l'aplicació ja estan acabades, són usables i orientades a l'experiència d'usuari. Es decideix aturar el desenvolupament fins a la presentació del TFG, i centrar els esforços en millorar i augmentar el nombre de test i la documentació de l'aplicació.

Pel que fa a a l'evolució de la feina pendent al llarg del sprint, a continuació podem veure la burndown chart.

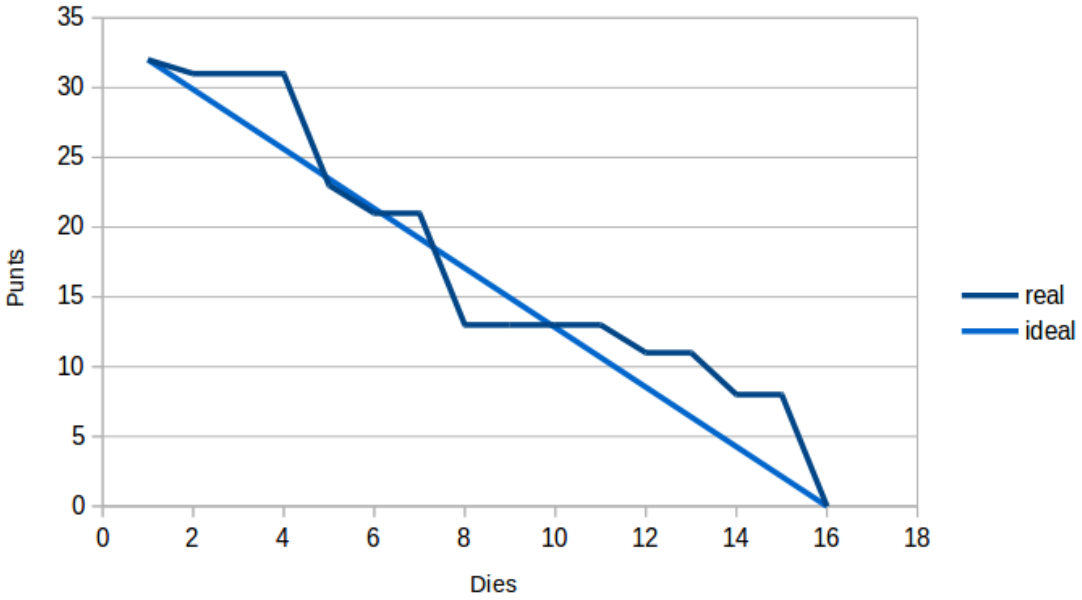

Figura 12: Burndown chart del quart sprint

Veiem a la figura 12 que hi ha moments d'estancament coincidint amb els dies d'examen, encara que en general, s'ha anat seguint el ritme ideal més o menys.

5 Implementació

En aquest apartat exposarem l'estructura del projecte, i cada un dels detalls de la implementació i de la tecnologia utilitzada. A la figura 13 podem veure l'estructura general.

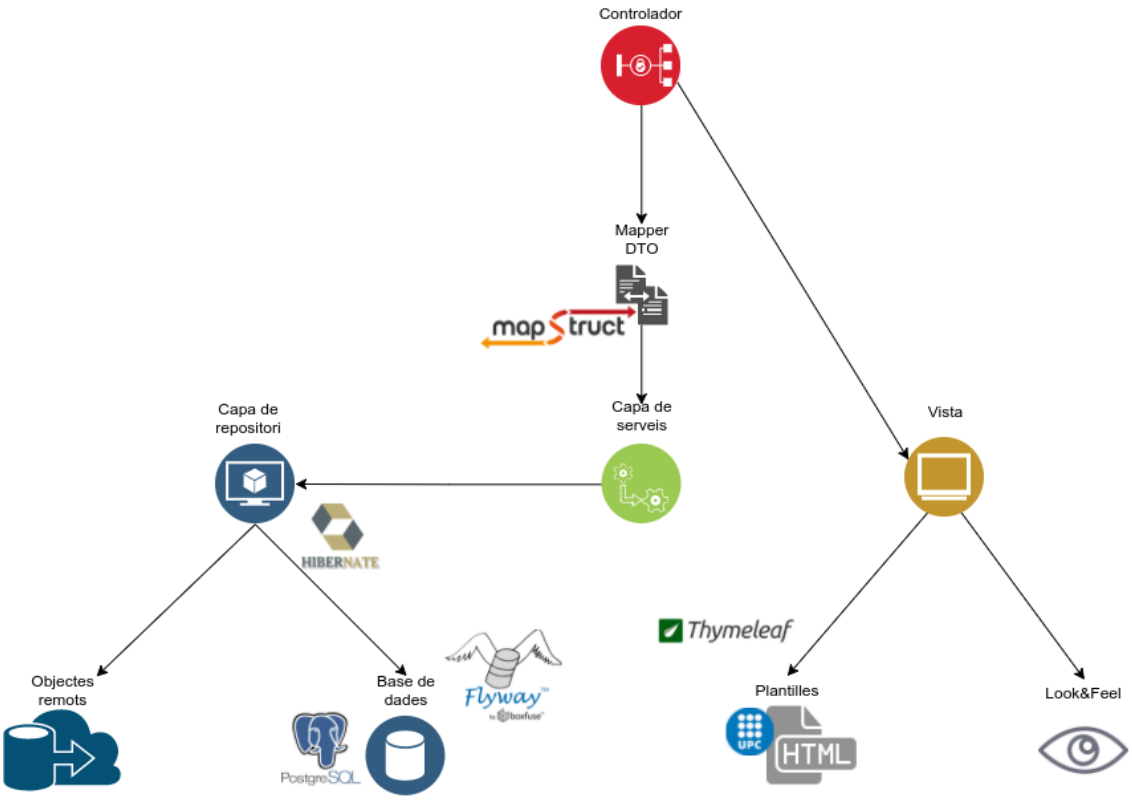

Figura 13: Estructura general del projecte

Aquesta es basa en una estructura per capes, en contret, en tenim tres. Les capes són les que es corresponen amb el patró model, vista i controlador. Així, tindrem que el paper de la vista o presentació rau en les plantilles HTML, i el JavaScript i el CSS que completen l'experiència del Look&Feel. Per altra banda tenim el model, que estaria conformat per la capa de repositori, la capa de serveis i la dualitat entitat/DTO (data transfer object). Per ultim tenim els controladors, que atenen les peticions HTTP i serveixen el contingut demanat preparant el model per a conformar la vista. A continuació anem a exposar la implementació de cada capa i el programari utilitzat.

5.1 Spring Boot

Spring és un framework per al desenvolupament d'aplicacions i és també un contenidor d'inversió de control per a la plataforma Java. Encara que Spring Framework no

força cap model de programació, ha esdevingut amplament popular dintre de la comunitat Java primerament com una alternativa que a desplaçat el model Enterprise JavaBean. Les seves característiques principals són:

- Gestió de la configuració basada en JavaBeans, aplicant-hi principis d'Inversió de Control amb injecció de dependències.
- Una factoria de Beans central, que és usada globalment.
- Capa genèrica d'abstracció per la gestió de transaccions de la base de dades.
- Estratègies preincorporades per la JTA i un sol DataSource de JDBC.
- Integració amb entorns de persistència i napeig objecte-relació.
- Entorn per al desenvolupament d'aplicacions web MVC, que constitueix al nucli de la funcionalitat de Spring, suportant moltes tecnologies per generar vistes i treballar amb motors de plantilles, d'igual forma que aportant implementacions a problemes comuns.
- Entorn extensiu de programació orientada a aspectes per proveir serveis, com ara gestió de transaccions.

Spring Boot per altre banda, es presenta com a un sub-conjunt del framework. Segons la pròpia definició de la web:

> Spring Boot fa que sigui fàcil crear aplicacions autònomes, preparades per a entorns de producció i basades en Spring, que podem "simplement executar". Prenem una visó balancejada de la plataforma de Spring i de les biblioteques de tercers perquè pugueu començara treballar amb un mínim cost. La majoria de les aplicacions d'Spring Boot requereixen una mínima configuració.

Donat que el framework d'Spring conforma un enorme sistema que pot esser utilitzat en multitud de contextos, la configuració inicial d'aquest per arrancar un projecte és costosa. La voluntat d'Spring Boot es encapsular una configuració bàsica per començar a desenvolupar aplicacions web de form ràpida i àgil. Spring està dissenyat de forma que el desenvolupador es centri en la capa de negoci original del projecte, poden utilitzar multitud de mòduls amb solucions a implementacions usuals, com la seguretat, els serveis web, la persistència etc.

5.2 Model

A la capa del model hi tenim quatre elements importants: la base de dades, la capa de repositori, la capa de serveis i els mappers entitat-DTO. El model de dades definitiu és el següent:

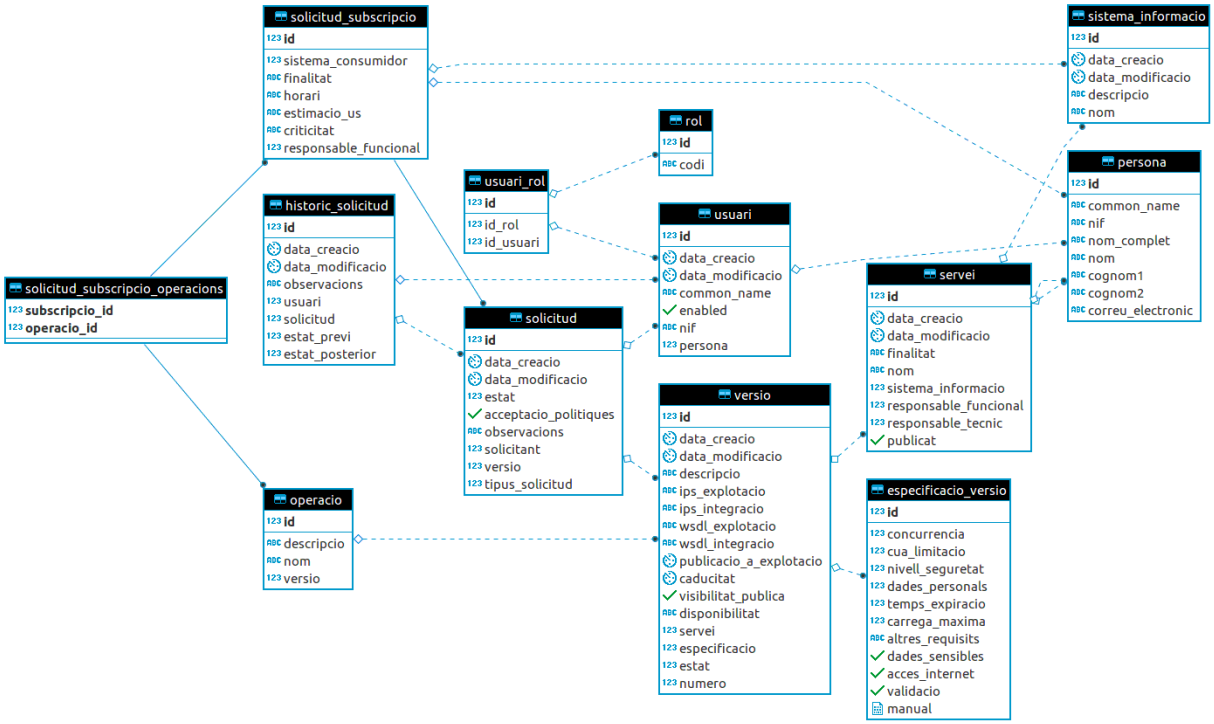

Figura 14: Model definitiu del projecte

A la figura 15 veiem els diferents elements involucrats a la capa del model. Per una banda tenim la capa de repositoris, que treballa amb objectes tipus entitat. Les entitats són la implementació d'un patró de disseny que permet afegir una capa d'abstracció entre la font de les dades i els objectes del model. Així, tant si les dades provenen de base de dades com si provenen de fonts remotes, a nivell intern els tractem igual.

Per tal de mapejar les entitats del model amb les relacions de la base de dades utilitzem Hibernate. **Hibernate** és una solució pel mapeig objecte-relacional (ORM), que ens permet fer aquesta correspondència de manera transparent. És mitjançant anotacions que configurem el mapeig entre taules i objectes. Desacoblem així el tipus de base de dades, o d'altres formes de persistència, del model. D'igual forma en aquest projecte utilitzem **flyway**, que és una eina de migració de versions. Encara que explicarem amb més detall el seu funcionament a la part *Compilació i desplegament*, aquest programari ens permet fer canvis, i materialitzar-los a la resta d'entorns on estigui funcionant l'aplicació.

Cal mencionar que Hibernate és una implementació de **JPA**, Java Persistence API. JPA especifica la API per a la persistència de dades, defineix el llenguatge de consulta JPQL i modela el mapeig objecte-relacional. En aquest sentit Hibernate és només una implementació, igual que ho són ObjectDB, OpenJPA, EclipseLink o altres.

D'igual forma tenim objectes remots com a una altre font de dades. En aquest

projecte s'utilitza PersonesV8, que es un servei SOAP que retorna informació sobre les dades d'aquells que estan vinculats a la UPC. En el procés de compilació es descarrega la definició de l'objecte amb l'ajuda de **JAX-WS**. A cada mètode de PersonesV8, JAX-WS fa el mapeig a la sol·licitud corresponent del servei web que representa.

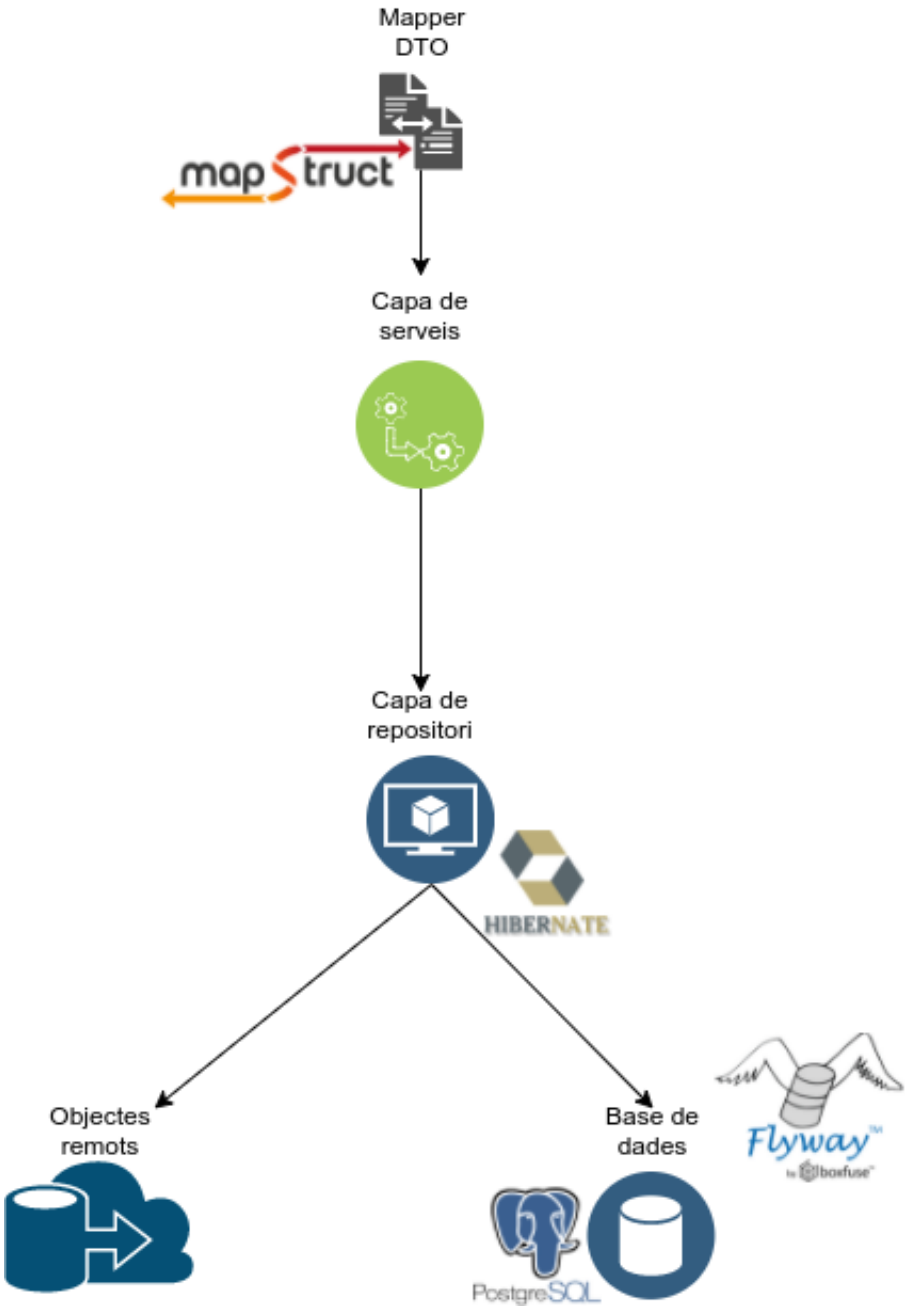

Figura 15: Elements de la capa de model

La capa de repositoris no s'utilitza de forma directa, sinó que són les classes de la capa de Serveis les que en fan ús. Aquesta capa de serveis està conformada

pels serveis corresponents a cada entitat i també pels mappers que fan la conversió d'entitat a DTO i viceversa. Quina és la intencionalitat d'utilitzar aquest tipus d'objectes? Els DTO, Data Transfer Object, ens serveixen per:

- Exposar només els camps que necessitem de la base de dades
- Transformar la informació de la base de dades per adaptar-la a la vista.
- Agrupar informació diversa en una única classe.
- Afegir validacions i restriccions en quant al contingut.

Anem a exemplificar això a partir de l'exemple concret de les sol·licituds. Per la sol·licitud de publicació d'un nou servei web o una nova versió web tenim un únic DTO. Aquest l'hem anomenat SolicitudDTO. D'igual forma s'han creat les classes DTO dels atributs que apunten a altres objectes, com ara ServeiDTO, VersioDTO, EspecificacioVersioDTO etc.

```
public class EspecificacioVersioDTO {
    public static Double MIDA_MAXIMA_MANUAL = 4e+6;

    private Long id;

    @Min(value=1, message="{formulari.valor.minim.un}",groups = {SolicitudDTO.NouServei.class,SolicitudDTO.NovaVersio.class})
    @NotNull(message = "{formulari.obligatori}",groups = {SolicitudDTO.NouServei.class,SolicitudDTO.NovaVersio.class})
    @Builder.Default private Integer concurrencia =  2;

    @Min(value=1, message="{formulari.valor.minim.un}",groups = {SolicitudDTO.NouServei.class,SolicitudDTO.NovaVersio.class})
    @NotNull(message = "{formulari.obligatori}",groups = {SolicitudDTO.NouServei.class,SolicitudDTO.NovaVersio.class})
    @Builder.Default private Integer cuaLimitacio = 5;

    @Min(value=1, message="{formulari.valor.minim.un}",groups = {SolicitudDTO.NouServei.class,SolicitudDTO.NovaVersio.class})
    @NotNull(message = "{formulari.obligatori}",groups = {SolicitudDTO.NouServei.class,SolicitudDTO.NovaVersio.class})
    @Builder.Default private Integer tempsExpiracio = 10;

    @Min(value=1, message="{formulari.valor.minim.un}",groups = {SolicitudDTO.NouServei.class,SolicitudDTO.NovaVersio.class})
    @NotNull(message = "{formulari.obligatori}",groups = {SolicitudDTO.NouServei.class,SolicitudDTO.NovaVersio.class})
    @Builder.Default private Integer carregaMaxima = 1;

    @NotNull(message = "{formulari.obligatori}",groups = {SolicitudDTO.NouServei.class,SolicitudDTO.NovaVersio.class})
    private Boolean validacio;
```

Figura 16: Fragment de EspecificacioVersioDTO

A la figura 16 veiem un fragment de EspecificacioVersioDTO. Podem observar que a la mateixa classe hi tenim els valors per defecte dels camps que es mostraran al formulari, i també les validacions. Els valors per defecte són els que assignem als atributs marcats amb l'anotació @Builder.Default. Aquesta anotació de lombok inicialitza els atributs amb els valors assignats al constructor. Per altre banda les validacions venen també anotades, per exemple usem: @NotNull, @Min(1), @Email, @FutureOrPresent, @NotEmpty etc. Per a validacions més complexes tenim l'anotació @Pattern, que ens permet comprovar si un string segueix una expressió regular. Per a validacions de més d'un camp, o per les quals no hi hagi una anotació explícita, usem mètodes adhoc que retornen un boleà i els anotem com a @AssertTrue. D'igual formà, si anotem un objecte amb @Valid, quan es validi l'objecte pare, també es validaran els atributs de l'objecte fill. A la figura 16

veiem, a més a més, que cada una de les anotacions ve acompanyada de *message* i *groups*. El primer fa referència al missatge d'error associat en cas de que no es compleixi la condició; el segon ens serveix per organitzar grups de validació. Mentre que SolicitudDTO ens serveix tant per a gestionar el formulari de sol·licitud d'un nou servei com per al de sol·licitud d'una nova versió, les validacions associades a cada camp no són les mateixes, així que amb l'atribut *groups* podem fer aquesta diferenciació.

Ara que ja tenim clars quins són els objectes amb els que treballarem a la vista (DTO) i com emmagatzemem les dades (entitats i base de dades), anem a centrar-nos en la lògica del negoci que trobem a la capa de serveis. Per fer això hem de tornar a fe referència al model de dades de la figura 14. Aquest model resulta altament acoblat, i és així ja que per a cada sol·licitud ha d'haver-hi una versió, que al seu torn ha de tindre un servei. A nivell conceptual tenim tres tipus de sol·licituds:

- Sol·licitud de publicació d'un nou servei
- Sol·licitud de publicació d'una nova versió d'un servei
- Sol·licitud de subscripció a una nova versió

Aquestes tres sol·licituds, però, es materialitzen en dos taules a la base de dades. La relació solicitud, que conté els camps comuns als tres tipus. Després tenim solicitud_subscripcio, que hereta de la primera i completa la informació que hi falta per a les subscripcions. Tenim, per ultim, un camp tipus_solicitud que és al final el que ens permet classificar-les.

Anem a reflexionar sobre la sol·licitud de publicació d'un nou servei. Amb el model que tenim, al crear aquesta sol·licitud, hem de crear alhora la versió inicial i el la definició del servei, d'igual forma que la seva primera versió. Aquesta implementació simplifica el model de dades però també requereix d'una coordinació absoluta entre l'estat de la sol·licitud i el del servei i la versió. Veiem a la figura 17 aquesta relació. Així, no és fins que la sol·licitud finalitza, que la versió passa a l'estat publicat, o bé es rebutja i acaba a l'estat desestimat. L'estat obsolet és manual, i és l'únic que no va lligat a l'estat de la sol·licitud. Pel que fa al servei, aquest té dos estats, publicat o no. Un servei estarà publicat sempre que tingui alguna versió publicada.

La lògica dels estats es gestionada per la capa de serveis. A més a més de les validacions dels camps, també tenim algunes comprovacions alhora de convertir la SolicitudDTO (o qualsevol altre objecte DTO) a una entitat del domini. Això es degut a que exposem alguns camps que ens són útils alhora de visualitzar les dades però no al editar-les. Podríem crear diferents objectes DTO, però s'ha optat per controlar-ho a partir dels mappers. Per fer aquesta conversió utilitzem **MapStruct**, que genera codi automàticament a partir d'una definició bàsica de la correspondència entre els camps del DTO i la entitat. Anem a veure un exemple senzill.

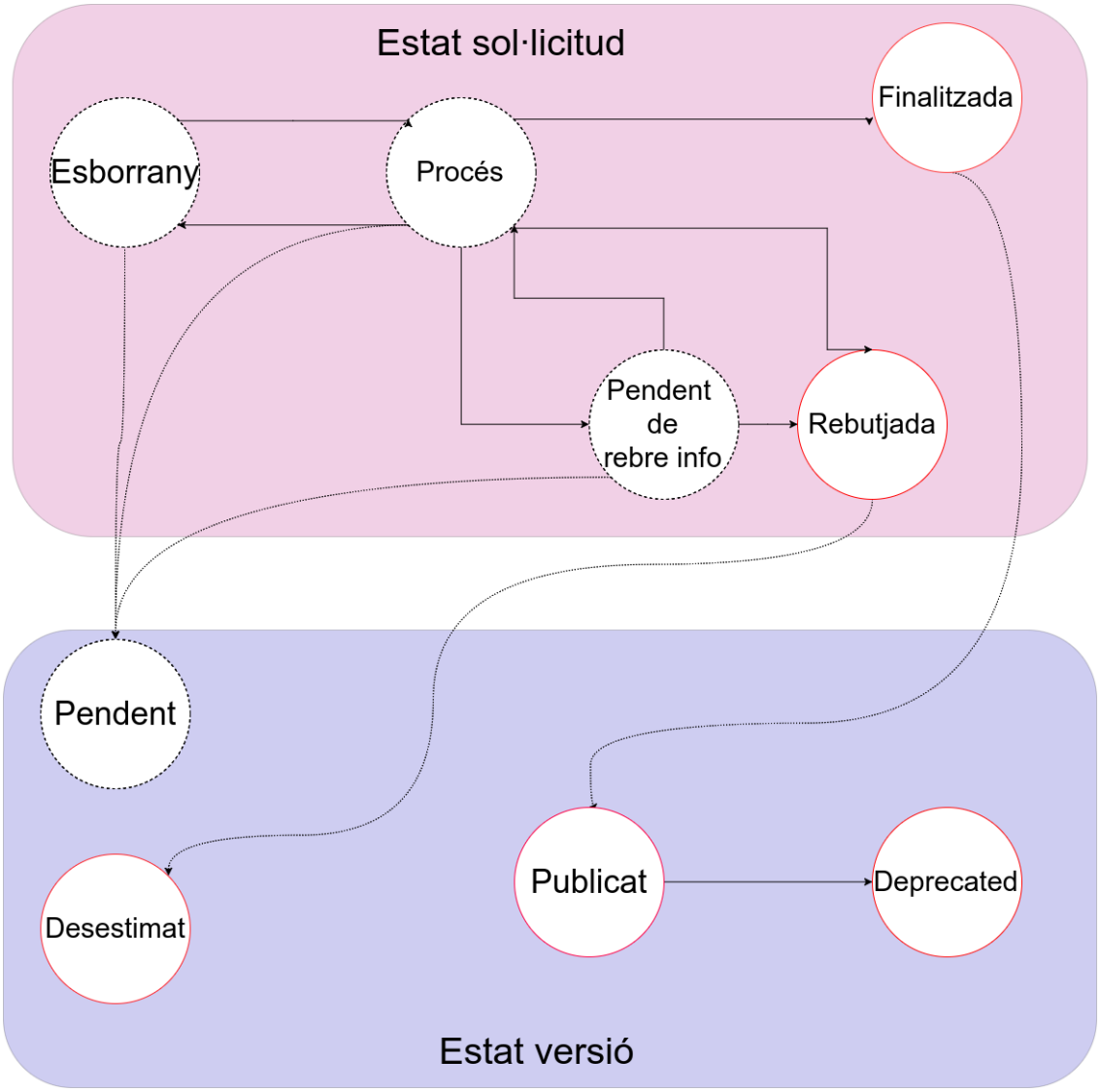

Figura 17: Correspondència entre els estats de la versió i els del servei. Els estats amb vora vermella són finals

Veiem a la figura 18 les classes involucrades en aquest exemple de l'entitat Usuari. A l'esquerra tenim la classe que representa la taula usuari, amb les anotacions corresponents d'Hibernate per gestionar la relació amb la base de dades. A la dreta veiem l'objecte DTO, on només hi tenim els atributs nif i nomComplet. Fins ara l'aplicació no ha necessita exposar més camps. Llavors, la transformació de l'un a l'altre ve donada per la classe UsuariMapper. Aquesta classe genera el codi de manera automàtica per a fer la conversió. Per exemple, la conversió de DTO a entitat ve donada per fromDTO. L'anotació *@mapping* ens serveix per a fer la correspondència dels camps. En aquest cas li diem que ignori l'atribut nif, ja que aquesta informació no es editable. Quan una persona intenta accedir a l'aplicació, si es de la comunitat UPC, recuperem les seves dades del servei web PersonesV8 i així construïm l'objecte usuari. Pe altra banda tenim el mètode toDTO; en aquest cas es

fa el mapping amb l'atribut nif de manera automàtica, ja que tenen el mateix nom, i addicionalment, fem el mapping del camp nomComplet, de l'objecte Persona, a un string a UsuariDTO. A més a més veiem una altre anotació a la part superior de UsuariMapper on hi indiquem que volem utilitzar la classe UsuariResolver. Això fa que, abans de fer el mapping de DTO a entitat, comprovi si a base de dades ja existia l'usuari, i així recuperem els camps on ja tenim dades. En aquest cas concret, quan passem de UsuariDTO a Usuari, realment no estem mapejant cap camp, així que simplement es va a buscar a base de dades l'entitat segons el nif.

Figura 18: Correspondència entre els estats de la versió i els del servei. Els estats amb vora vermella són finals

També podem definir mètodes supletoris anotats amb @BeforeMapping o @AfterMapping, que tal com indica el nom, s'executaran abans i després de fer les operacions de correspondència de camps. Aquest mètodes poden modificar tant l'objecte DTO inicial, com l'entitat en construcció o rescatada de la base de dades. Aquí és on podem fer certes comprovacions, com esmentava anteriorment amb el cas de les sol·licituds. Per exemple, en el cas de la sol·licitud de subscripció, comprovem si les operacions subscrites pertanyen realment a la versió a la que l'usuari es vol subscriure. Si veiem que no, llancem una excepció que serà atesa pel controlador.

MapStruct ens evita la tediosa feina d'escriure la correspondència dels atributs, amb la comprovació sistemàtica de que els objectes estiguin definits i amb la facilitat de simplement afegir o treure anotacions si els camps canvien. Això minimitza els

errors i el risc de fallada.

5.3 Vista

Pasem ara a parlar de la vsita. El component essencial d'aquesta capa, sense dubte, són les plantilles HTML. S'ha utilitzat el motor de plantilles **thymeleaf** que ens permet:

- Definir una jerarquia de plantilles i reutilitzar components.
- Utilitzar de forma transparent i directa els objectes DTO afegits pel controlador.
- Cridar funcions, iterar sobre col·leccions, escriure expressions condicionals etc.
- Centrar-nos en escriure l'HTML d'una forma natural.
- Internacionalització dels textos.

Primer anem a veure aquesta jerarquia de plantilles. A fragments.html tenim els elements comuns a totes les pantalles, aquest són els següents:

```
<head data-th-fragment="head-fragment">...</head>
<header data-th-fragment="header-fragment" class="topbar">...</header>
<aside class="left-sidebar"
    data-th-fragment="left-sidebar-fragment">...</aside>
<footer class="footer" id="footer-fragment"
    data-th-fragment="footer-fragment (mainJs)" >...</footer>
```

Veiem que cada un dels elements té un atribut anomenat data-th-fragment. Això és un identificador que és el que ens permet la reutilització dels components. En aquest cas tenim un fragment pel head, el header, una barra lateral i un peu. A més a més, per tal d'organitzar aquests components i definir una estructura base per a qualsevol pantalla, definim un *layout*. En el cas de l'aplicació desenvolupada, es té la següent estructura:

```
<!DOCTYPE html>
<html>
<head data-th-replace="fragments :: head-fragment"></head>

<body class="fix-header fix-sidebar ">
    <div id="main-wrapper">
        <header data-th-replace="fragments :: header-fragment"></header>
        <aside data-th-replace="fragments ::
            left-sidebar-fragment"></aside>
    </div>
    <div class="page-wrapper">
        <div class="js-titol-seccio d-none d-md-block row ">
```

```
            <div layout:fragment="page-title">
                <!-- Title goes here -->
            </div>
        </div>
        <div class="js-container row">
            <div class="container-fluid " layout:fragment="content">
                <!-- Content goes here -->
            </div>
        </div>
    </div>
</body>

<footer data-th-replace="fragments :: footer-fragment
    (${mainJs})"></footer>
</html>
```

Aquesta estructura respon a la nova imatge corporativa de les aplicacions de la UPC, adaptada per a ser utilitzada en el projecte. La representació visual la podem veure a continuació:

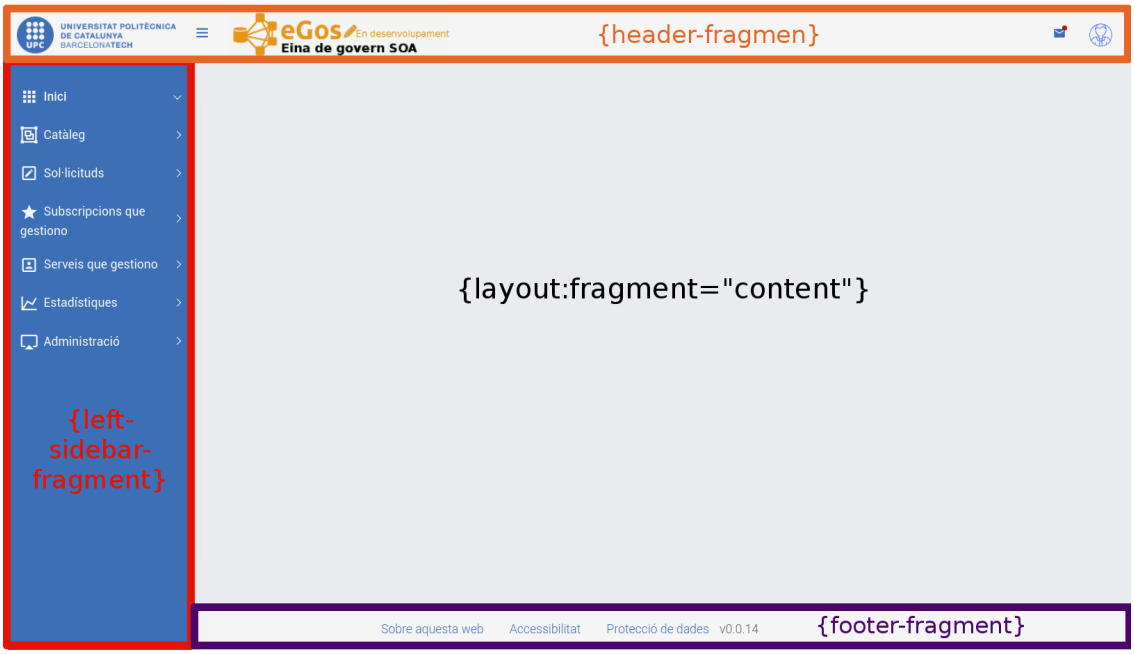

Figura 19: Estructura bàsica de la imatge de l'aplicació

Aquesta modularitat ens permet canviar la imatge base de forma fàcil i ràpida. Anem a veure també la capacitat que ens atorgo **thymeleaf** per a fer plantilles amb llenguatge natural, i al mateix temps treballar amb els objectes del model. A continuació es mostra el cas de la llista dels serveis que gestiono:

```
<table class="display nowrap table table-hover table-striped"
    id='js-llista-serveis' data-th-if="${!serveis.empty}">
```

```
   <thead...>
   <tbody>
      <tr data-th-each="servei, iterStat : ${serveis}">
         <td data-th-text="${servei.nom}">
            grupsRecerca
         </td>
         <td data-th-text="${servei.sistemaInformacio.nom}">
            DRAC
         </td>
         <td  data-th-text="${#lists.size(servei.versions)}">
            3
         </td>
         <td  data-th-text="${servei.responsableTecnic?.nomComplet}">
            Lluís Malvis
         </td>
         <td  data-th-text="${servei.responsableFuncional?.nomComplet}">
            Lluís Malvis
         </td>
      </tr>
   </tbody>
</table>
<div class="text-center" data-th-text="#{serveis.no.dades}"
    data-th-if="${serveis.empty}">
               No hi ha serveis a mostrar
</div>
```

Veiem que el codi és pràcticament HTML correcte, però amb una sèrie d'atributs no estàndards que són els que gestiona thymeleaf. El primer que s'ha de fer notar és l'atribut *data-th-if* de l'element *table*, que comprova si la llista *serveis* té elements. Si no en té, no es mostra la taula i mostra la div que hi ha continuació, amb el text de la referència *serveis.no.dades*, que tenim definit a les propietats del projecte. Així, el literal que veiem a la div on hi posa "No hi ha serveis a mostrar"serà substituït per el de la referència, que en aquest cas és el mateix, però l'explicitem a la plantilla també per mantenir un HTML natural. Si tenim serveis a mostrar, veiem que al primer tr hi tenim la propietat *data-th-each*, que ens permetrà iterar el contingut de la llista. Així, a cada columna hi posem com a text el contingut d'un dels atributs del servei.

A més a més del codi HTML, utilitzem **JavaScript** per a introduir dinamisme a l'aplicació i millorar el Look&Feel d'aquesta. Hem organitzat els fitxers de JavaScript, d'igual forma que els plugins i les dependències, de manera modular per poder propiciar la reutilització de components. La principal eina de la que fem ús per acomplir amb aquest objectiu és *require.js*. Podem veure a la figura 20 l'estructura interna dels fitxers JS dins el projecte.

Així, a l'inici de cada plantilla afegim l'atribut *data-th-with* al tag d'html, i li assignem el nom d'un fitxer JavaScript, que es correspon amb algun dels que tenim a l'arrel l'estructura de fitxers Js i que anomenem de forma sistemàtica com main-*.js. Aquest fitxers carreguen la configuració de *require.js* i criden alguns dels fitxers

main.js que tenim a les diferents carpetes.

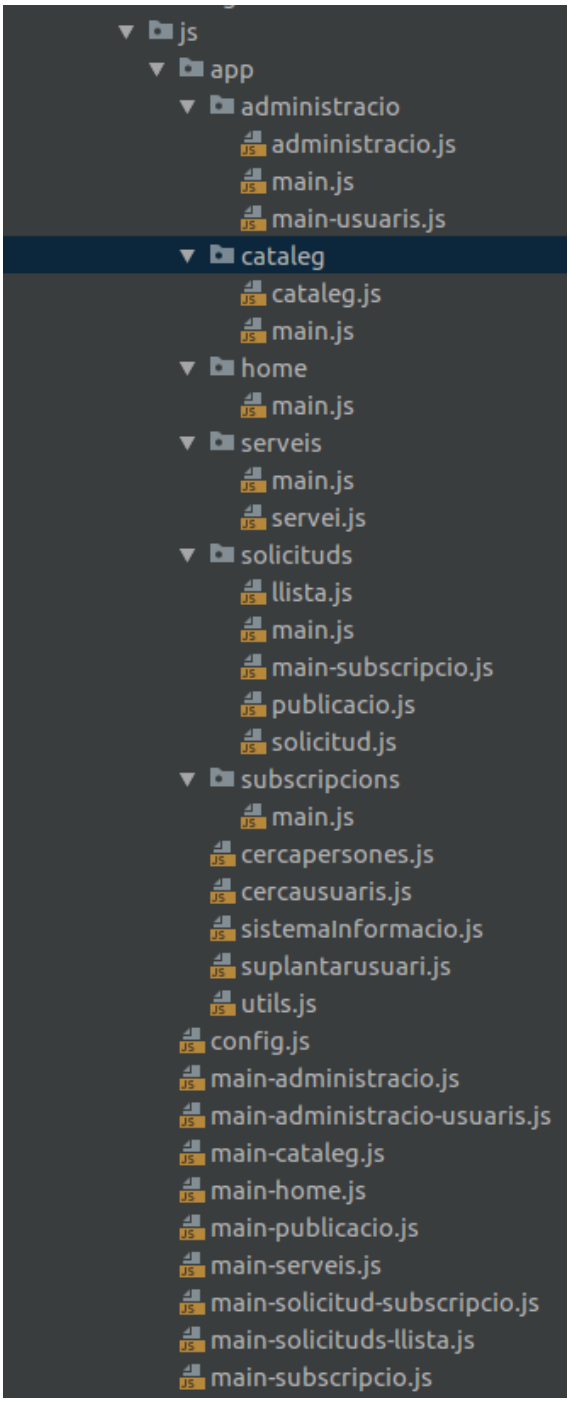

Figura 20: Estructura dels fitxers JavaScript

Podem observar en el codi de la figura 21 com des del fitxer main fem el require de totes les dependències i plugins que utilitzarem a la pantalla. Require carregarà els fitxers i a més a més les dependències que haguem explicitat a config.js.

```
define(function(require) {
    var $ = require('jquery'),
        bootstrap = require('bootstrap'),
        scrollbar= require('perfect_scrollbar'),
        waves = require('waves'),
        datatables = require('datatables.net'),
        sidebarmenu = require('sidebarmenu'),
        stickykit = require('stickykit'),
        sparkline = require('sparkline'),
        theme = require('theme'),
        suplantarusuari = require('app/suplantarusuari'),
        select2 = require('select2'),
        select2Ca = require('select2-ca'),
        utils = require('app/utils')
    ;
    $(document).ready(function () {
      //CONTINGUT A EXECUTAR
    });
});
```

Figura 21: Estructura dels fitxers main.js

A més a més podem cridar els fitxers JS propis, com en el cas de *suplantarusuari* que veiem a la fig.21. Així, a *cataleg.js*, *administracio.js*, *servei.js* etc hi posem les funcions pròpies d'una part del model de l'aplicatiu, però que poden ser reutilitzables i cridades des de qualsevol altre javascript.

Anem ara a mencionar tres plugins més que ens són de gran utilitat i utilitzem a tota l'aplicació de forma recurrent.

- **DataTable**: aquest pulguin ens permet transformar les taules HTML estàtiques en taules dinàmiques, on l'usuari pot fer cerques, ordenar i paginar. Ens permet millorar notablement el Look&Feel d'una forma ràpida i fàcil de configurar.
- **Toastr**: llibreria que ens facilita l'ús de notificacions no bloquejants. Molt fàcil d'utilitzar, ja que per defecte ens permet mostrar diferents tipus de notificacions: d'error, d'advertència, de confirmació i d'informació. És fàcilment extensible per afegir-hi nous tipus. Es utilitzat majoritàriament per donar resposta a l'usuari a les accions que es duen a terme mitjançant AJAX. Ho podem veure a la fig.22

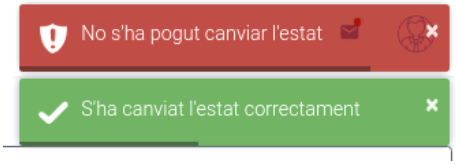

Figura 22: Missatges d'èxit i d'error amb Toastr després de realitzar accions amb AJAX

- **select2**: plugin que ens aporta dinamisme en l'ús de desplegables d'HTML, permetent-nos fer cerques, selecció múltiple, cerca remotes per AJAX, validacions etc. Podem veure dos exemples a la figura 23.

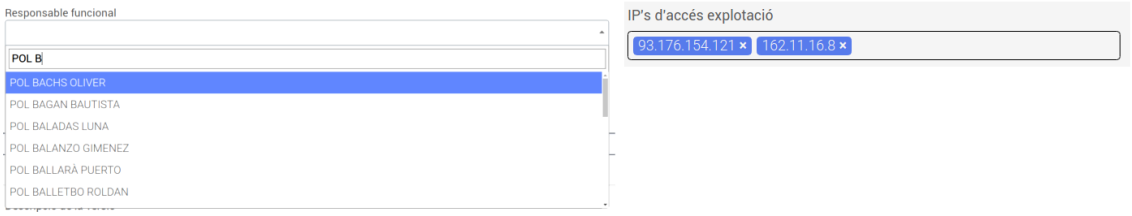

Figura 23: Usos del plugin select2. A l'esquerra hi tenim una cerca remota per AJAX. Els resultats es mostren com una llista seleccionable. A la dreta tenim un select múltiple on s'afegeixen elements de forma dinàmica. Aquests elements han de complir una expressió regular, que en aquest cas es correspon amb una IP.

Cal esmentar que aquest pluguins no funcionen sense el ja indispensable **JQuery**. Aquesta llibreria simplifica enormement les tasques d'interacció amb l'HTML, donant un ampli suport per a la manipulació del DOM (Document Object Model), treballar amb events, desenvolupar animacions i fer crides AJAX. Al mateix temps, simplifica la notació de JavaScript i la fa més natural i llegible.

Per ultim em queda parlar del CSS. Com la plantilla base és corporativa i unificada per les noves aplicacions web de la UPC, en gran mesura s'utilitzen els estils propis del tema escollit. Així, no s'ha realitzat pràcticament cap canvi dels estils que ja venen per defecte, encara que s'ha afegit un fitxer extra, anomenat estils.css, on hi posem els estils propis. Com a font d'icones utilitzem essencialment els de Font-awesome i Material Icons.

Cal fer menció de **Boostrap**, un framework per treballar el front-end, que ens proporciona un conjunt d'HTML, JavaScript i CSS per a maquetar de forma ràpida i còmode vistes i components web. Boostrap m'ha permés prototipar ràpid, i evolucionar els dissenys sense gran esforç.

5.4 Controlador

La capa de controlador, és el punt de síntesis del model MVC, ja que es l'encarregat d'atendre les peticions i servir la vista segons la lògica de negoci.

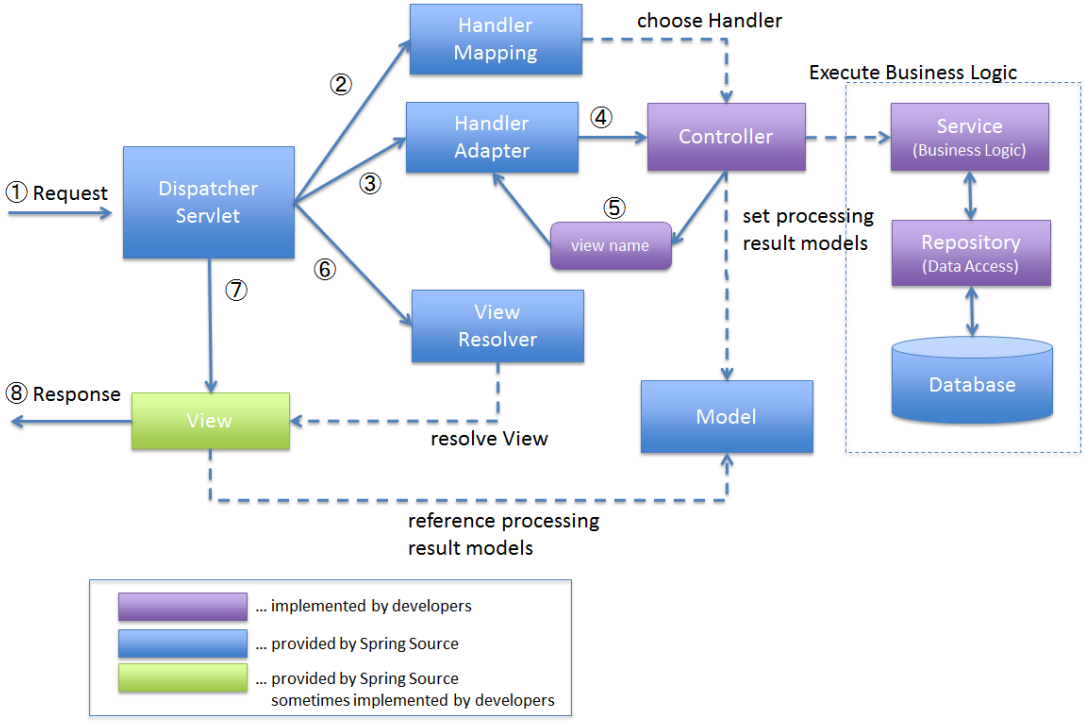

Figura 24: Esquema general del procés d'atendre una petició i servir el contingut a Spring[3]

A la figura 24 podem veure l'esquema general de funcionament. Veiem que el primer punt d'entrada es el Dispatcher Servlet, que delega la tasca de trobar el controlador adequat a HandlerMapping. Aquest darrer compara la URL de la petició amb les mapejades als controladors, i en retorna l'adequat. El Dispatcher Servlet delega l'execució del la lògica del controlador al Handler Adapter. El controlador executa la lògica del negoci, afegint els resultats d'aquesta al model i escollint el nom de la vista. View Resolver retorna la vista a partir del nom, i per ultim, Dispatcher Servlet delega a la vista la integració del model, que acabarà retornant com a resposta a la petició original.

A nivell programàtic anotem amb *@Controller* els controladors, i amb *@RequestMapping* fixem la url que atenen. Els controlador de l'aplicació sempre segueixen la següent estructura:

- Declaració dels noms de les vistes com a atributs de la classe, estàtics i finals .

[3]Font: https://terasolunaorg.github.io/guideline/1.0.1.RELEASE/en/Overview/SpringMVCOverview.html

- Referències als objectes de la capa de serveis que necessitem. La injecció de dependències d'Spring s'encarrega d'inicialitzar els objectes al constructor de la classe.

- Mètodes per a gestionar peticions a diferents urls. Cada mètode ve amb *@RequestMapping* i la url que até. Per defecte, si no en té, atent la url del *@RequestMapping* definit a la classe. A més a més de la url, també si ha d'especificar el mètode HTTP (get, post, delete...). També s'especifica com es retorna el contingut del mètode, usualment amb *@ResponseBody*, que indica que la resposta de la petició estarà al body. També s'especifiquen els paràmetres de la petició. Aquest poden trobar-se a la url de la petició o al cos de la mateixa. Quan venen al cos, simplement apareixen com a paràmetres d'entrada del mètode. Quan es troben a la url, a més a més, els hem d'anotar amb *@RequestParam*. A tot mètode d'un controlador, hi ha un atribut d'entrada que sempre hi és, i ve anotat amb *@AuthenticationPrincipal*, i fa referència l'objecte que ens proporciona la informació sobre la sessió i l'usuari connectat.

A més a més podem limitar l'accés als controladors segons els rols d'usuari. Per a fer això anotem la classe o el mètode que volem protegir amb *@PreAuthorize*, especificant el rol que garanteix l'accés. Si l'usuari intenta accedir a una url protegida, y no té el rol necessari, salta una excepció que acabarà mostrant una pàgina que informa de l'error 403 Forbidden d'HTTP.

Més enllà del funcionament general anem a desenvolupar un cas força comú: una sol·licitud POST on s'envia un formulari. Aquest es el cas, per exemple, dels formularis de les sol·licituds de publicació. La url per enviar el formulari és /solicitudsPublicacio/enviar. La petició és atesa pel mètode processaFormulariEnviar de la classe SolicitudPublicacioController. Aquest mètode te com a variables d'entrada, un objecte SolicitudDTO, que es el que fa el mapping amb els camps del formulari. El primer que fa el mètode es escollir la plantilla, segons el tipus de sol·licitud. A continuació valida l'objecte SolicitudDTO, aplicant les validacions també segons el tipus de sol·licitud. Si el formulari té errors s'afegeixen al model de retorn per a que siguin visualitzats. Si no hi ha errors, la capa de serveis procedeix a gravar la informació a base de dades. Primer es passa l'objecte DTO a entitat i a continuació es persisteix. En aquest procés es poden produir diferents excepcions, que són llançades cap enredera fins a arribar al mètode del controlador, on seran tractades. En l'exemple que s'exposava, tenim SolicitudNoModificablePerUsuariException i SolicitudPublicacioException; la primera es genera si la sol·licitud no es modificable per l'usuari, i la darrera en el cas de que les dades de la sol·licitud no siguin coherents, probablement per que s'ha manipulat el formulari. Si no s'ha generat cap excepció, es redirigeix la petició al formulari d'edició de la sol·licitud ja persistida.

5.5 Compilació i desplegament

Per a realitzar el procés de compilació ens hem servit de **Maven**. Aquesta eina es desenvolupada per l'Apache Software Foundation i ens serveix per a configurar de forma fàcil i clara el procés de compilació. Per a fer això s'utilitza un Project Object Model (POM), un fitxer xml per descriure el procés de compilació i explicitar les dependències. La part principal del cicle de vida de Maven és:

- compile: generar els fitxers .class compilant les fonts .java
- test: executar els test automàtics de JUnit existents, avortant el procés si algun d'ells falla.
- package: generar el fitxer .jar amb els .class compilats

Al POM del projecte hi tenim:

- El tipus d'empaquetat que genera.
- La versió i descriptors del projecte.
- El pare del projecte, en aquest cas, es tracta d'un projecte fill d'Spring Boot.
- Les dependències del projecte (flyway, mapstructs, lombok, thymeleaf, postgresql...)
- La configuració del build. Aquí configurem lombok, mapstructs i JAX-WS.

Pel que fa al desplegament, aquest es duu a terme a GitLab-CI, i es configura en les següents etapes:

1. Build: GitLabCI descarrega la imatge de Docker amb l'entorn corresponent requerit pel procés de compilació i s'instancia el contenidor per iniciar-lo. Es descarrega les dependències i llibreries i es compila amb Maven, passant les proves unitàries.
2. Publish: si ha anat tot bé i s'han passat totes les proves de manera satisfactòria, es publica la imatge Docker amb l'aplicació i les se-ves dependències (infraestructura) al repositori d'Artifactory.
3. Deploy: Rundeck desplega la imatge de l'artifactory al servidor que correspongui coma ultima etapa, completant el cicle de desplegament continu.

Aquest procés garanteix la qualitat del codi gràcies a la bateria de proves unitàries i també una alta disponibilitat dels canvis realitzats per a que siguin incorporats a producció sota demanada en qualsevol moment. El procés d'integració podria millorar-se incorporant inspecció automàtica del codi per a trobar bugs, erros potencials, duplicat de codi etc. Això es podria fer afegint al pipeline les mètriques de sortida del SonarQube, una eina d'inspecció continua. Malgrat això no s'ha fet, sí s'ha treballat amb el plugin de Sonar per l'IDE, que proporciona la majoria de característiques.

6 Proves i resultats

En aquesta secció procedirem a exposar els resultats del projecte, mostrant algunes de les pantalles més important de l'aplicació desenvolupada.

Catàleg És aquesta una de les parts fonamentals de l'aplicació. Aquí es pretén centralitzar la informació disponible sobre els diferents serveis web i les seves versions.

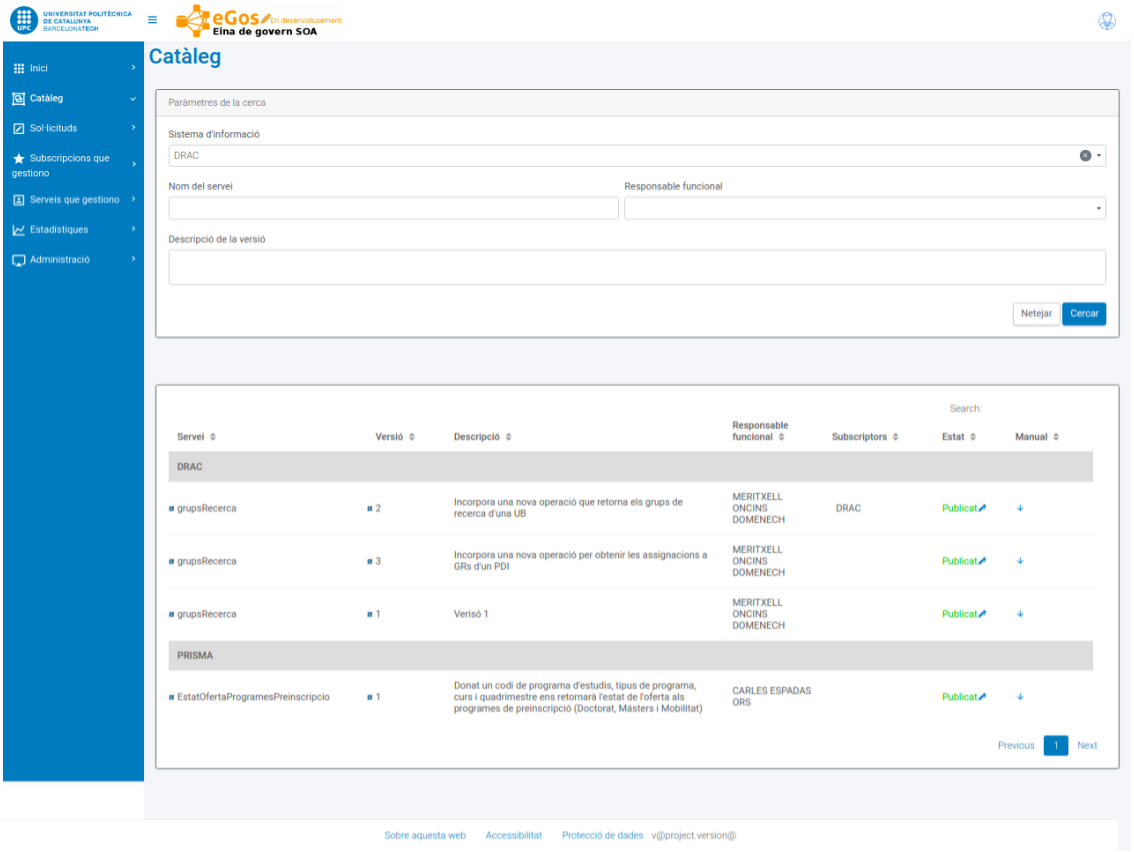

Figura 25: Vista del catàleg

Veiem a la figura 25 la vista del catàleg. A la part superior hi ha el cercador, on podem buscar per sistema d'informació del servei, nom parcial, responsable funcional o descripció de la versió. Si no introduïm cap paràmetre a la cerca, la llista mostra totes les versions. Les versions es visualitzen classificades per sistema d'informació, fent així més còmode i clara la vista per a l'usuari. Els botons de cercar i netejar el formulari retornen el resultat de la cerca via AJAX, fent així que no sigui necessari actualitzar la pàgina.

Veiem que a la llista, el nom del servei i el número de la versió són enllaços. Al fer-hi clic, s'obre una modal que via AJAX carrega la informació del servei i de la versió respectivament. Al detall del servei, que el podem veure a la figura 26,

hi visualitzem tots els camps propis i també una llista amb les diferents versions. Aquí, de nou, podem clicar sobre la versió i navegar-hi al seu detall, d'igual forma que féiem a la llista del catàleg.

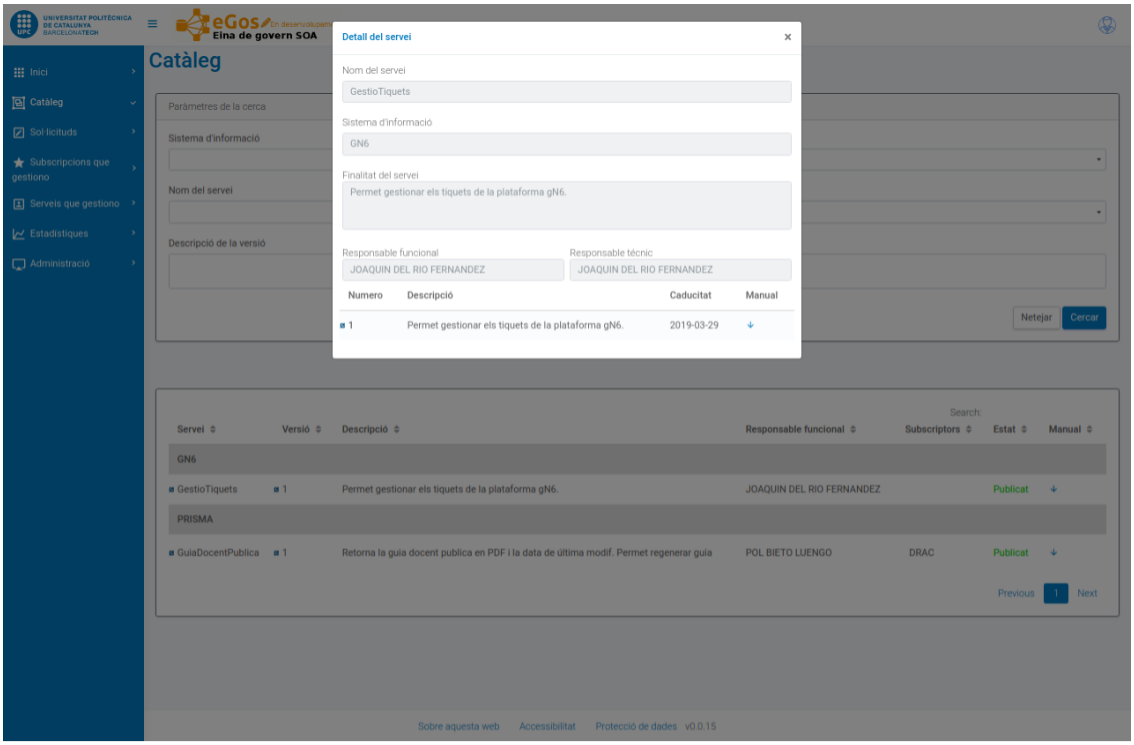

Figura 26: Detall del servei

Navegant al detall de la versió, a la figura 27, veiem tots els camps propis, i a la part superior l'estat i el manual. Al costat de l'estat es pot veure una icona d'un llapis. Aquesta icona només la visualitzen els administradors i permet canviar via AJAX l'estat de la versió. Tal com s'ha dit a la figura 17, el canvi possible de l'estat de la versió és de Publicat a Obsolet i viceversa. Aquesta acció AJAX converteix l'estat en un desplegable per efectuar el canvi, i al seleccionar-ne un dels disponibles, desapareix per mostrar el nou estat. El canvi també afecta a l'estat que es visualitza a la llista del catàleg, sota la modal, on també podem fer aquesta acció sense necessitat d'obrir el detall. A més a més, també hi veiem un icona per a tornar enrere, prop del títol de la modal. Fent clic en aquest icona naveguem de nou cap al detall del servei pare. A la part inferior de la nodal, hi tenim la llista d'operacions.

Comparant això amb la figura 5 veiem que, encara que la idea del prototip inicial s'ha conservat prou, ha evolucionat força, donant més pes al conjunt de camps propis de cada entitat. Una fita del disseny és que totes les accions del catàleg es fan via AJAX, així que un usuari que simplement vulgui explorar els serveis i les versions publicades ho pot fer sense necessitat de re-carregar la pàgina i navegant de forma natural amb un quants clics.

Per ultim, veiem a la figura 28 la vista de quan no s'obtenen resultats pels criteris de cerca introduïts. El disseny d'aquesta vista es comuna a totes les llistes quan no hi ha resultats.

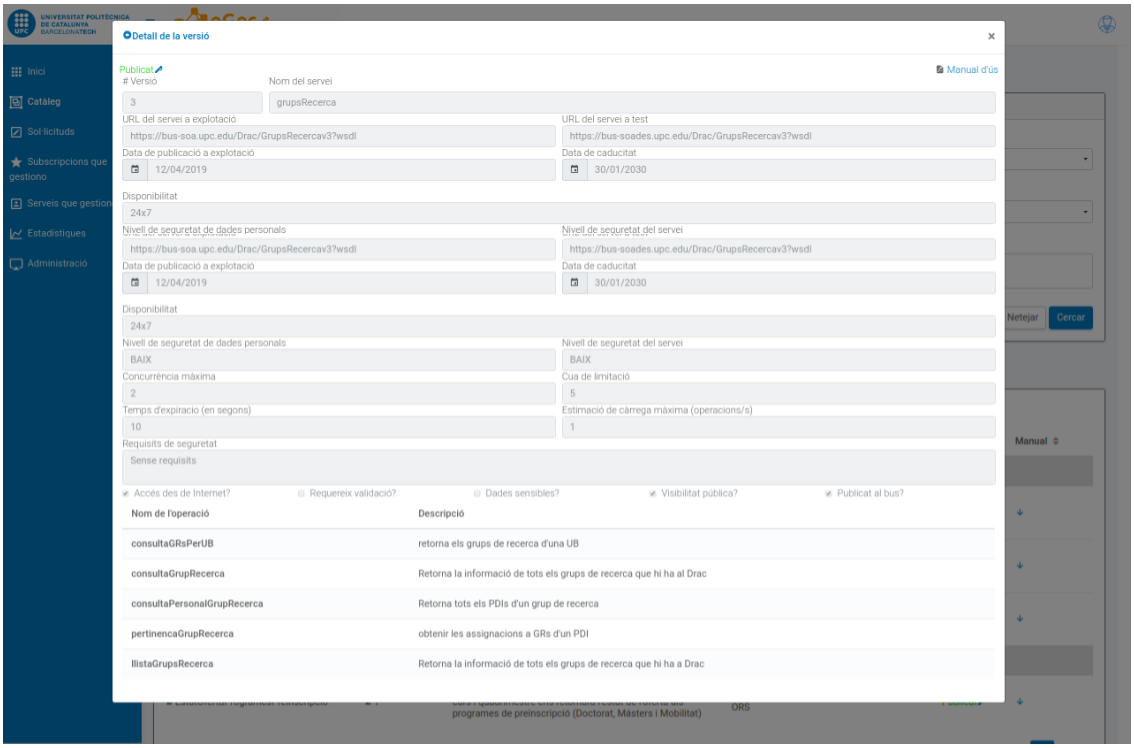

Figura 27: Detall de la versió

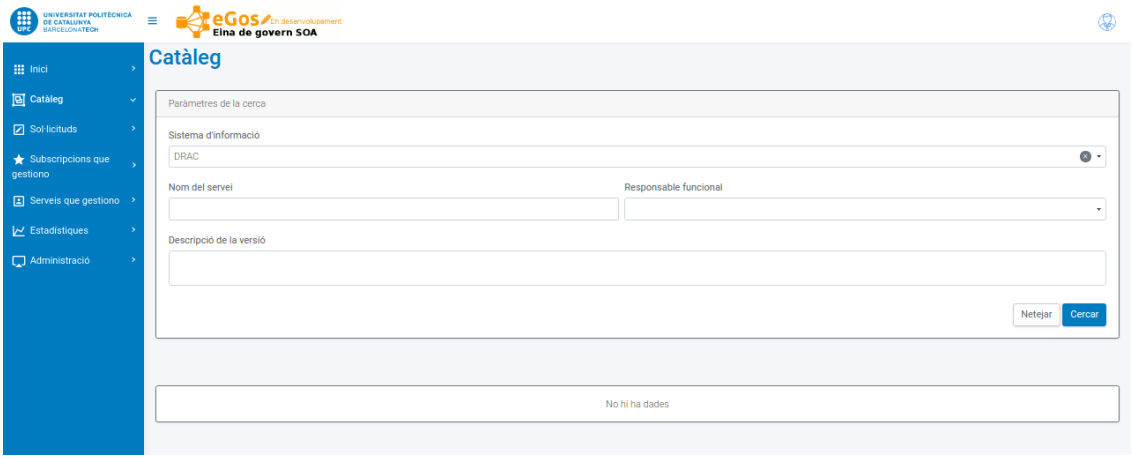

Figura 28: Vista del catàleg quan la cerca no proporciona resultats

Serveis i subscripcions que gestiono. Veiem a la figura 29 la vista dels serveis i les subscripcions que gestiono. A la primera sortiran els serveis on sóc responsable

funcional o responsable tècnic. A la darrera, les subscripcions on hi aparec com a responsable funcional. Veiem que ambdues vistes són similars. Aquí, igual que en el cas del catàleg, podem navegar al detall del servei o de la versió clicant sobre l'enllaç corresponent.

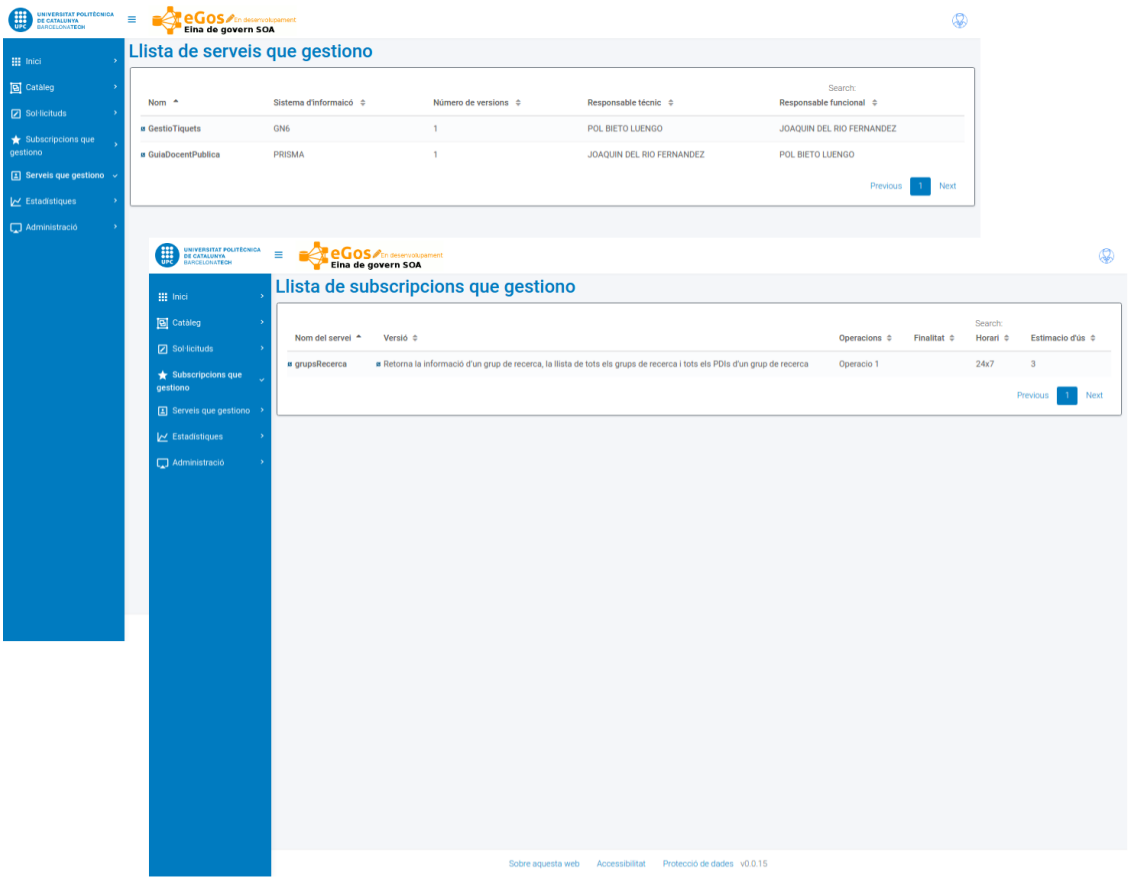

Figura 29: A la dreta, la llista dels serveis que gestiono, a l'esquerra, les subscripcions

Llista de sol·licituds. A la figura 30 veiem la llista de la sol·licituds, ordenades per la data de modificació, amb les més recents al principi de la llista. A la darrera columna de la taula, la que s'anomena "Accions", hi veiem diferents icones per cada una de les files. Si la sol·licitud està en estat esborrany, podem esborrar-la, editar-la o consultar l'històric de modificacions. En qualsevol altre cas, tant sols podrem veure la sol·licitud o consultar l'històric (es pot veure a la figura 31, on es carrega el contingut via AJAX). També hi podem veure el menú desplegable per afegir una nova sol·licitud, amb les opcions per a crear sol·licituds de publicació d'un nou servei, d'una nova versió o sol·licituds de subscripció.

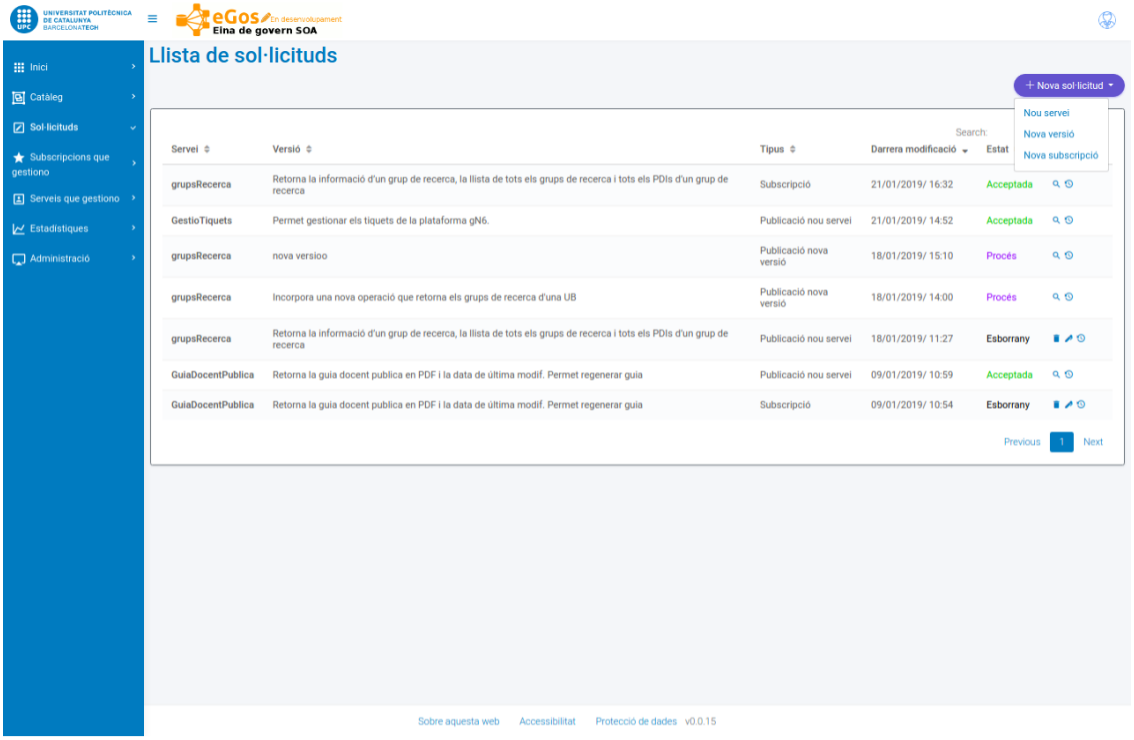

Figura 30: Vista de la llista de sol·licituds

Figura 31: Vista de l'històric de la sol·licitud

Formularis de sol·licitud. Aquest formularis, que són la part de l'aplicació que recopila més dades, tenen un gran nombre de camps i una estructura complexa. Com que aquí no els podríem visualitzar sencers, anem a fer un resum d'algunes de les seves característiques. A la figura 32 veiem elements del formulari de publicació d'un nou servei, també presents als altres formularis. A la part superior esquerra hi veiem la modal per afegir un nou sistema de la informació. Si el sistema sobre

el qual volem publicar el servei web no apareix al desplegable, podem afegir-lo a través d'aquesta modal, que es carrega per AJAX. Un cop s'ha comprovat que els camps estan correctament validats, es tanca la modal i al desplegable hi queda automàticament seleccionada la nova opció. A la part superior dreta hi veiem els camps responsable tècnic (que ja s'ha comentat a la figura 23 que en aquest tipus de camp es produeix una cerca en línia via AJAX), i el bloc d'operacions. Les operacions és poden afegir de manera dinàmica, amb la icona que apareix a la dreta, o eliminar-les, amb el símbol d'un cubell de deixalles que apareix al costat de la caixa de text de la descripció de l'operació. Per ultim, a la part inferior de la figura, hi veiem un tooltip amb una ajuda contextual i un calendari. Les ajudes són presents a tots els camps, i han de servir per guiar a l'usuari a posar la informació adequada a cada camp. El calendari utilitza un plugin de JavaScript i s'aplica sobre camps que requereixen introduir una data (en aquest cas, la da de publicació del servei web a explotació), per evitar formats erronis i facilitar la feina a l'usuari amb un interfície molt comuna.

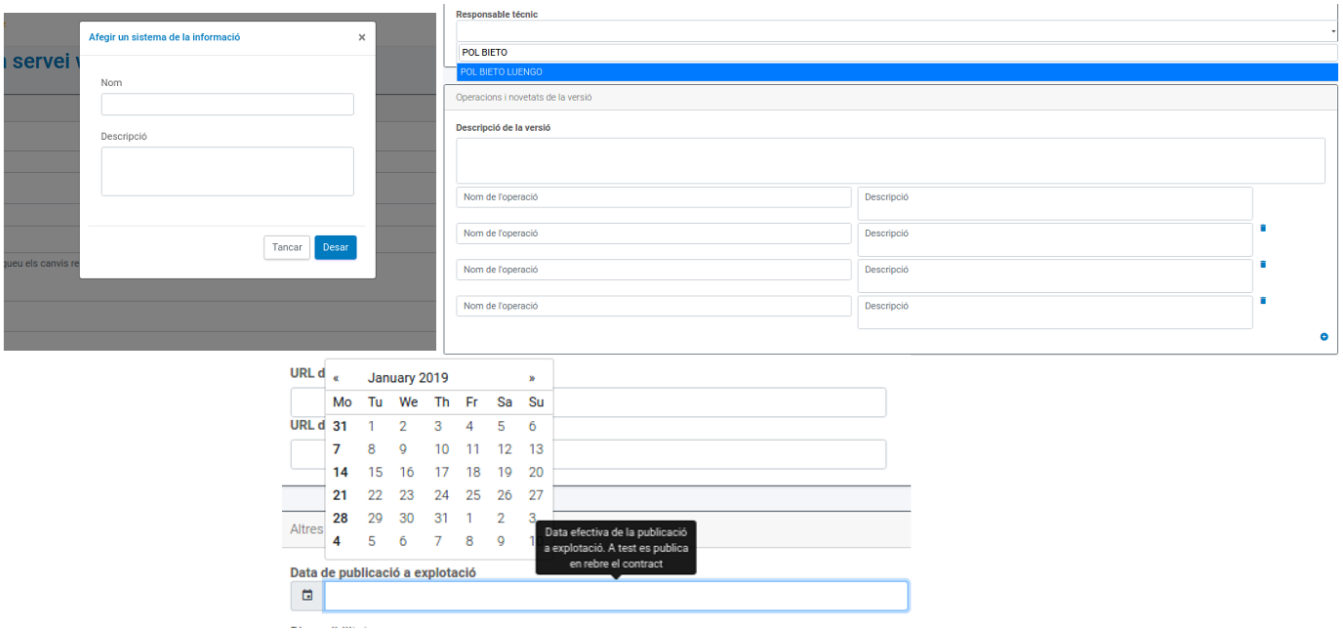

Figura 32: Elements del formulari de publicació d'un nou servei

A la figura 33 veiem una part del formulari, un cop validat. S'avisa a l'usuari amb un missatge que el formulari no es correcte (el podem veure a la part superior dreta de la imatge). També, a sota de cada camp, mostrem els missatges d'error particulars. A la imatge podem veure missatges com "El camp és obligatori", "El format de la url no és correcte", L·La data no pot ser del passat"o "Has de marcar aquest camp". Aquesta part del formulari es troba fent *scroll down* de la pàgina. Quan fem scroll, el títol de la pàgina es queda flotant, fixat a la part superior, per a que l'usuari tingui present en tot moment on es troba. Això ho fem a totes les pantalles de l'aplicació.

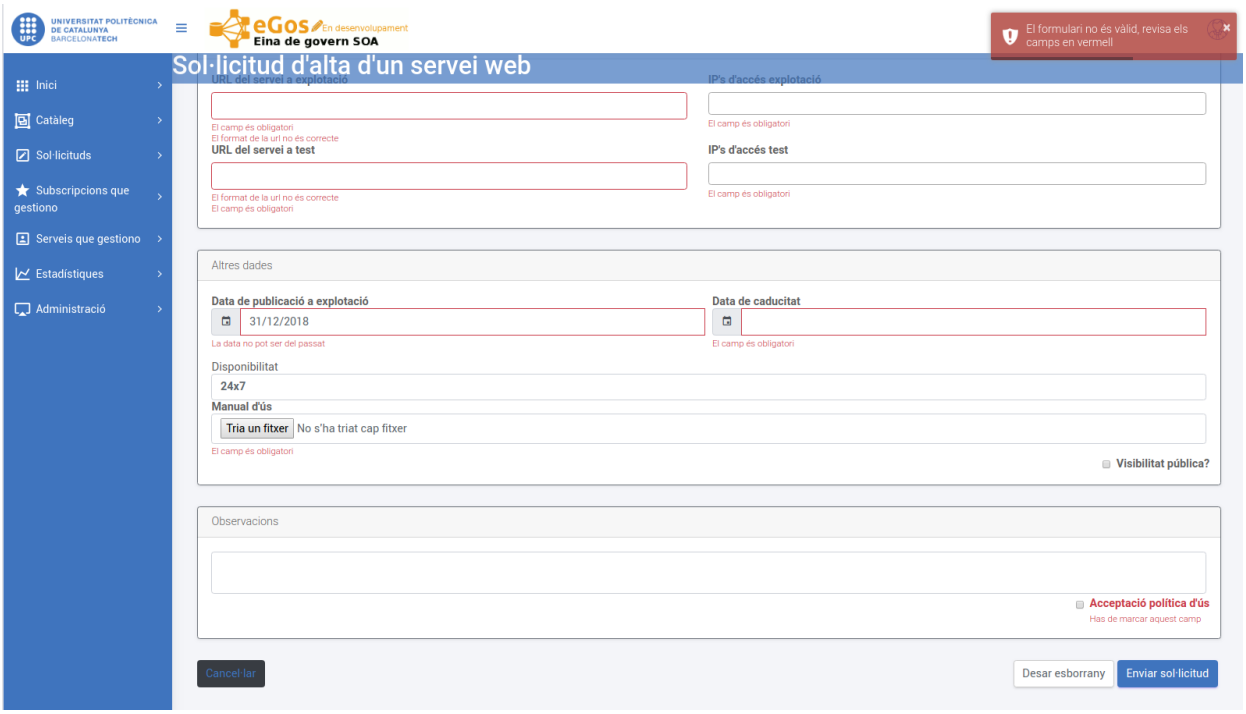

Figura 33: Formulari amb errors

A la figura 34 veiem la part corresponent al servei web del formulari de sol·licitud de subscripció. En aquest formulari hem d'indicar quines operacions, de quina versió i de quin servei ens volem subscriure. Per facilitar aquesta feina a l'usuari, s'ha fet aquesta vista, que es correspon amb una selecció en cascada. El primer camp ve pre-carregat amb els serveis publicats. Un cop seleccionem el servei, via AJAX carreguem les versions disponibles. Al marcar la versió, carreguem les operacions en el darrer desplegable, que és múltiple.

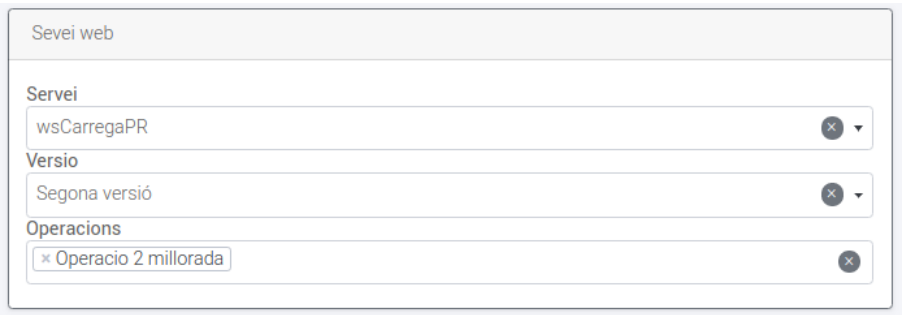

Figura 34: Selecció del servei, la versió i les operacions a subscriure

Pel que fa al formulari de publicació d'una versió, a la figura 35 podem veure que l'usuari pot escollir el servei pare d'un desplegable. S'ha situat un botó al costat del desplegable, que via AJAX ens permet recuperar els camps de la darrera

versió, estalviant així temps a l'usuari i facilitant donar d'alta una versió introduït el mínim nombre de dades possible.

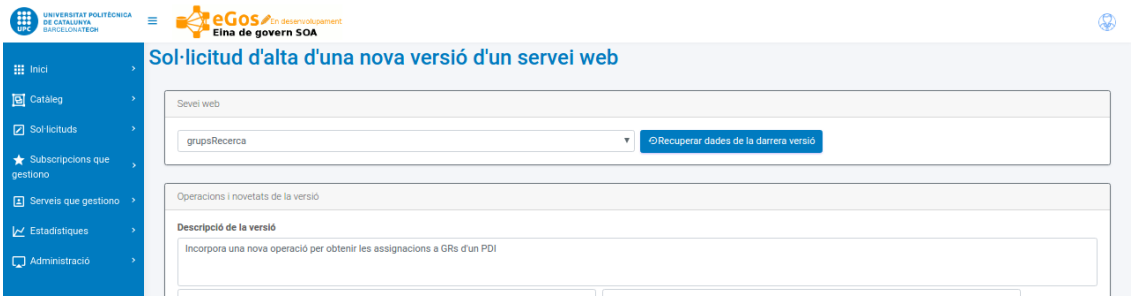

Figura 35: Formulari de publicació d'una nova versió d'un servei, on veiem que podem recuperar els camps de la darrera versió

Per ultim, a la figura 36 hi veiem una part del formulari d'alta d'un servei web un cop aquest ha passat de la fase esborrany i ja no es pot editar, aquest cas, amb el menú lateral plegat.

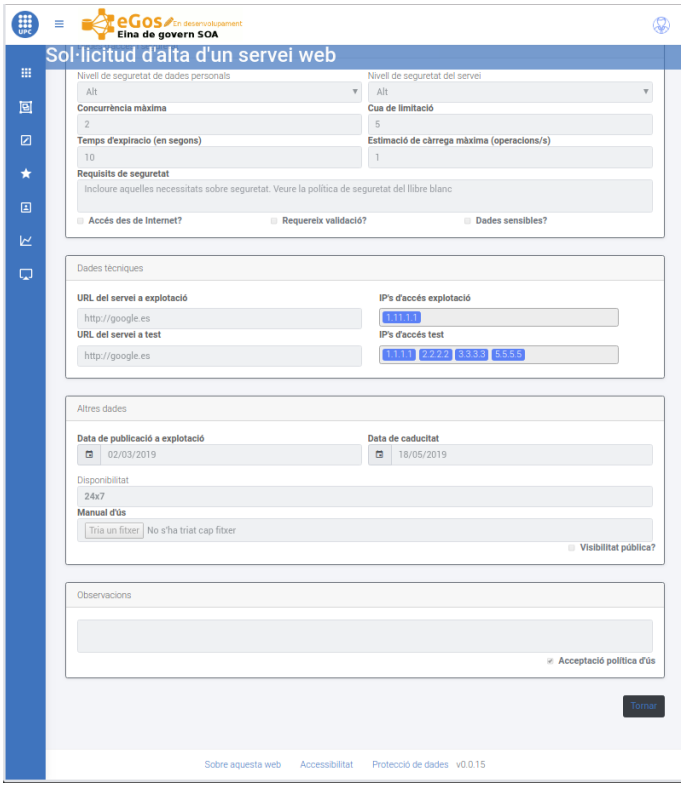

Figura 36: Vista no editable de la sol·licitud d'alta d'un servei web

Administració. Aquesta secció es força simple i només s'ha implementat allò més bàsic i necessari. Hi ha dues pantalles, la de l'administració dels usuaris i la de

les sol·licituds. La primera la podem veure a la figura 37. Hi podem veure una llista dels usuaris que han entrat a l'aplicació i podem atorgar permisos d'administració a qui volem marcant la casella de la columna "És administrador?". El canvi s'efectua mitjançant AJAX, i es retorna un missatge de confirmació o error segons si la operació s'ha efectuat amb èxit o no.

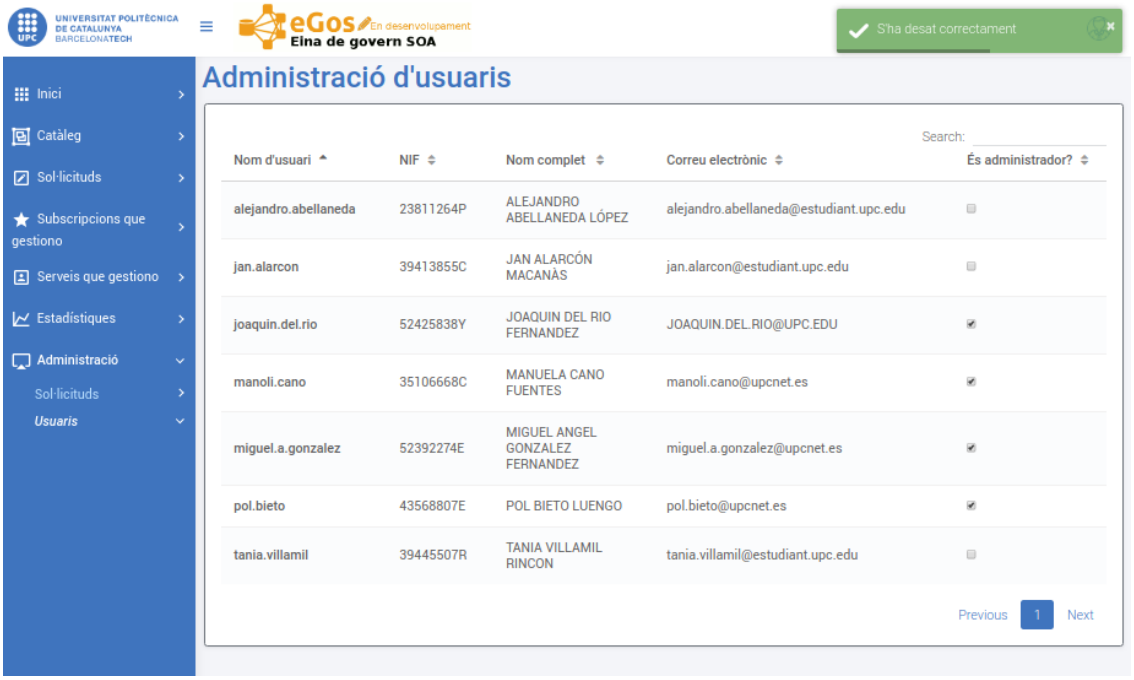

Figura 37: Vista de l'administració dels usuaris

A la figura 38 veiem el cas de l'administració de les sol·licituds. En aquesta pantalla es permet fer totes les accions disponibles sobre les sol·licituds a l'administrador. També es pot gestionar el canvi d'estat d'aquestes. Per fer-ho, s'ha de clicar sobre l'estat actual de la sol·licitud i sobre el desplegable amb les opcions. Només aquelles opcions que segueixin el diagrama d'estats de la figura 17 són vàlids. Si no es selecciona un a opció vàlida, es posa l'estat original i s'adverteix a l'usuari que no s'ha pogut canviar l'estat. Si el canvi es correcte, es mostra el nou estat i un missatge de confirmació, tot això via AJAX.

Cal destacar que al final la imatge d'aquesta secció ha estat molt diferent a la de la figura 6. Això s'ha degut a que el Product Owner s'ha acabat decantant per una imatge més clàssica i clara.

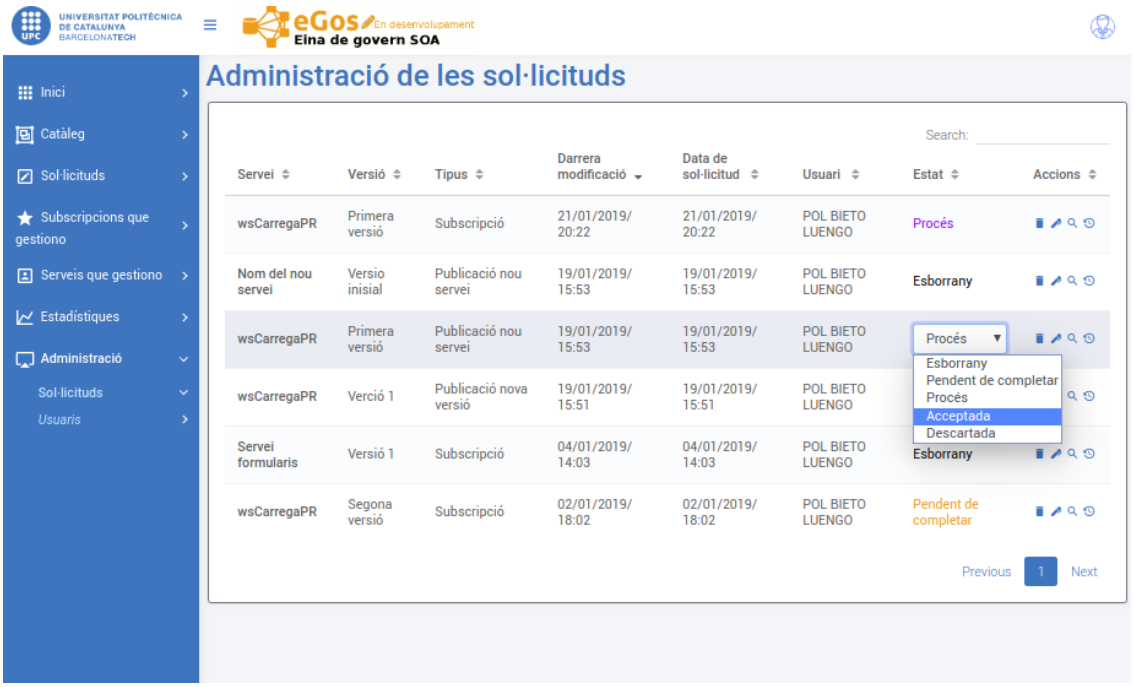

Figura 38: Vista de l'administració de les sol·licituds

Login i suplantació. El login, com ja s'ha comentat, s'ha implementat mitjançant el sistema d'identificació CAS de la UPC. La pantalla de login la podem veure a la figura 39. Un cop introduïdes les credencials, si són vàlides, es redirecció a la pàgina inicial de l'aplicació. Aquest tipus d'implementació garanteix que qualsevol usuari UPC pugui accedir a l'aplicació, així que un cop logat es crea un usuari intern amb les dades provinents del servei web PersonesV8.

Figura 39: Login mitjançant el CAS de la UPC

A més a més, si som usuaris administradors, podem fer-nos passar per un altre usuari. Aquest mecanisme, que anomenem suplantació, pot ser força útil per atendre incidències d'usuari i veure les seves dades. La suplantació es realitza clicant a la part superior dreta de qualsevol pantalla, i s'obre una modal com la de la figura 40. A la modal hi tenim un cercador de persones, com el ja esmentats a la figura 32 o 23. Un cop seleccionat l'usuari, es recarrega la pàgina.

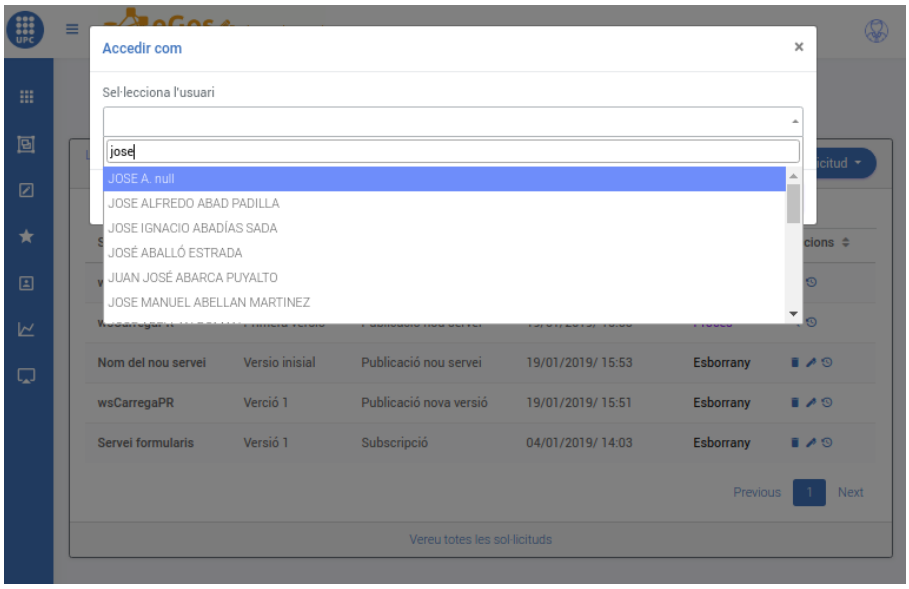

Figura 40: Formulari de suplantació amb cercador de persones

6.1 Proves

Pel que fa a les proves, s'ha intentat fer un desenvolupament guiat per proves i realitzar aquestes de manera sistemàtica. Encara amb això, no s'ha assolit l'objectiu completament, un 70% les línies del projecte estiguin cobertes. Pel que fa a la capa de serveis, la cobertura és del 60%. Sens dubte aquest valor s'ha de millorar fins a arribar a prop del 100%.

Les proves s'han realitzat mitjançant test unitaris de JUnit. S'ha definit una classe base, que em anomenat BaseTest, on inicialitzem la configuració de l'entorn de proves. En aquesta configuració inicial destaca l'esquema de la base de dades. Per a simular la presencia de la base de dades s'utilitza **H2**, que ens permet treballar en memòria aïllant l'entorn de proves. A més a més, també tenim scripts propis que s'executen a H2 per a les diferents proves. Per simular les crides HTTP i la interacció amb diferents classes s'utilitzen els Mocks d'Spring i **Mockito**, un framework de proves per Java. Així, les proves és fan punt a punt. Es testeja un endpoint de la capa del controlador, i es comprova la sortida. El conjunt dels test han de recórrer tots els camins interiors, de la capa de serveis i la del repositori.

7 Conclusions

Pel que fa als objectius marcats a l'inici del projecte es pot afirmar que la majoria s'han assolit amb relatiu èxit. L'objectiu general, de desenvolupar una aplicació usable, fàcil de mantenir, escalable i amb prestacions ben definides s'ha aconseguit. S'ha de matisar, potser que no s'ha implementat tot allò que a l'inici va semblar viable, valorant les capacitats de temps i habilitat. Això mereix esser analitzat per a tindre-ho en compte de cara a futurs projectes. Els principals motius semblen ser:

- Un procés massa llarg de formació i enteniment del marc de treball. Es va començar el projecte des de 0, sense gaire suport més enllà de la documentació disponible. No va estar fàcil configurar el projecte i modelar el disseny inicial. Malgrat això, aquest procés ha resultat satisfactori ja que m'ha permés aprendre molt i entendre de forma global l'ecosistema amb el que he desenvolupat el projecte. També m'ha donat una gran autonomia i confiança en mi mateix alhora d'encarar nous reptes. Encara així, en certs moments hauria d'haver demanat més suport i ajuda a companys experimentats.

- Un manca de definició i un estudi incomplert dels requeriments de l'aplicació i del seu model per part del Product Owmer. Això a generat certa indefinició alhora de prioritzar quines característiques aportaven major valor a la lògica del negoci i, com a conseqüència, alguns canvis que han perjudicat l'eficiència en el desenvolupament. Per altre banda, aquest punt demostra que s'ha fet bé escollint un tipus de desenvolupament àgil i que finalment ha resultat satisfactori per afrontar els canvis de criteri i portar a bon port el projecte per tindre un producte funcional.

Pel que fa als objectius específics aquí sí es pot afirmar que s'han treballat tots i s'ha arribat a un bon grau de coneixement i domini de les tecnologies i metodologies de treball que s'havien plantejat. Potser el que queda més fluix és el desenvolupament guiat per proves. Malgrat s'ha assolit un nivell de cobertura de proves decent, sovint s'ha desenvolupat sense abans fer els test. Això s'ha de corregir de cara al futur. Es possible que un millor plantejament dels requeriments hagués facilitat aquesta feina.

També m'agradaria destacar que s'ha tingut molt present un disseny centrat en l'usuari. A les retrospectives de cada un dels sprints s'han valorat les característiques incorporades sempre des del punt de vista de l'ús que li donarà l'usuari. Malgrat això, m'hagués agradat poder aprofundir en aquesta filosofia de disseny realitzant proves d'usuari en les diferents fases del disseny i la implementació. També crec, fent ara una valoració global del projecte, que hagués estat interessant treballar amb eines que faciliten aquesta feina i que són tendència al mercat, amb frameworks de JavaScript com Angular o React. De totes maneres, haver utilitzat aquest tipus d'eines hagués incrementat el temps de formació i investigació, fet que podria haver estat contraproduent.

Per ultim, destacar tot allò que m'ha aportat el projecte en quant a treballar en un entorn empresarial, amb l'ús d'eines corporatives i de tot l'ecosistema d'em-

presa. Crec que aquesta experiència em serà d'allò més útil per afrontar els reptes professionals del futur.

7.1 Continuïtat del projecte

La naturalesa àgil i el marc d'abast d'aquest projecte fan que, encara que amb la fi d'aquest treball final de grau es presenta una versió estable i amb funcionalitats acabades i ben definides, el procés de desenvolupament i millora no acaba aquí, més aviat entra en una nova fase on ara s'hauran d'incorporar dades i usuaris reals que facin ús de l'aplicació a producció. Així, el desenvolupament segueix, per a incorporar algunes de les funcionalitats que no ha donat temps d'implementar, com les notificacions o la integració de les estadístiques gestionades per la pròpia aplicació. També es planteja com a repte de cara al futur, treballar el disseny i millorar-lo segons l'ús que li donin els usuaris.

S'ha de destacar però, que l'ús de bones pràctiques i d'una metodologia corporativa de treball permeten que el projecte tingui continuïtat dins de l'empresa per a incorporar més desenvolupadors, i així, aconseguir fer créixer l'aplicació actual per a que realment pugui abordar totes les necessitats d'una eina de govern SOA.

Referències

[1] Boostrap Team. (s.d.). *Boostrap documentation v.4.1.* Recuperat de https://getbootstrap.com/docs/4.1/getting-started/introduction/

[2] Boxfuse GmbH. (s.d.). *Documentation. Welcome to Flyway, database migrations made easy.* Recuperat de https://flywaydb.org/documentation/

[3] Ferentschik, Hardy, Morling, Gunnar (s.d.). *Hibernate Validator 5.4.2.Final - JSR. 349 Reference Implementation.* Recuperat de https://docs.jboss.org/hibernate/stable/validator/reference/en-US/pdf/hibernate_validator_reference.pdf

[4] Mihalcea, V. , Ebersole. S., Boriero, A., Morling, G., Badner, G., Cranford, C., Bernard, E., Grinovero, S., Meyer, B., Ferentschik, H., King, G., Bauer, C., Rydahl Andersen, M. ,Maesen, K., Vansa, R., Jacomet, L. (s.d.). *Hibernate ORM 5.4.1.Final User Guide.* Recuperat de http://docs.jboss.org/hibernate/orm/5.4/userguide/html_single/Hibernate_User_Guide.html

[5] Morling, G., Gudian, A., Sjaak, D., Hrisafov, F. and the MapStruct community. (2007). *MapStruct 1.2.0.Final Reference Guide.* Recuperat de http://mapstruct.org/documentation/stable/reference/pdf/mapstruct-reference-guide.pdf

[6] Pivotal Software, Inc. (2018). *Spring Boot Reference Guide. 2.1.2.RELEASE.* Recuperat de https://docs.spring.io/spring-boot/docs/2.1.2.RELEASE/reference/pdf/spring-boot-reference.pdf

[7] Schwaber, Ken & Sutherland, Jeff . (2017). *The Definitive Guide to Scrum: The Rules of the Game.* Recuperat de https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[8] The Apache Software Foundation. (s.d.). *Maven Documentation.* Recuperat de https://maven.apache.org/guides/

[9] The jQuery Foundation. (s.d.). *jQuery API Documentation.* Recuperat de http://api.jquery.com

[10] The PostgreSQL Global Development Group. (s.d.). *PostgreSQL 11.1 Documentation.* Recuperat de https://www.postgresql.org/files/documentation/pdf/11/postgresql-11-A4.pdf

[11] Thymeleaf Team. (s.d.). *Tutorial: Using Thymeleaf.* Recuperat de https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.pdf