Aplicació web HelpDesk

Memòria de Projecte Final de Màster

Màster universitari en Aplicacions Multimèdia

Itinerari professional

Autor: Josep Ramon Camps Lladó

Consultor: Sergio Schvarstein Liuboschetz

Professor: David García Solórzano

04 de gener de 2016

Crèdits/Copyright

Documentació

Aplicacions

Les tres aplicacions que componen el projecte (API, aplicació d'usuaris i aplicació d'administradors) estan subjectes a la següent llicència:

Llibreries, frameworks i altres recursos

Les aplicacions fan ús d'un nombre elevat de llibreries, *frameworks* i d'altres recursos de tercers. Totes aquestes llibreries estan distribuïdes sota llicències *opensource* tipus MIT, Apache o similars (en cap cas GPL o de cap altre tipus que impliqui *copyleft*). Aquestes dependències estan detallades en el capítol *3.4 Llenguatges de programació i APIs utilitzades,* i no es detallen aquí per la seva extensió.

Cita

“Always keep in mind the old retail adage: Customers remember the service a lot longer than they remember the price.”

Lauren Freedman, President The E-tailing Group

Abstract

Providing a good customer service is a key aspect for any company or service. There's a wide range of tools and methods to deliver help to the clients and attend their demands, ranging from the traditional phone or email based systems, to issue trackers, help desks and other systems.

However, this field is not overly saturated, and there's room for experimentation and innovation. The Master's Final Project consists on the design and development of a web based help desk, using modern methodologies and architectures in web development, such as Single Page Applications, RESTful APIs, NodeJS and NoSQL databases, implemented using new programming languages as CoffeeScript, Stylus or ES6, prioritizing the usability and performance of all the components of the system to provide an optimal experience on any kind of devices.

Resum

Oferir un bon servei d'atenció als usuaris és un aspecte fonamental que ha de complir qualsevol empresa o servei. Existeixen una gran varietat de mètodes i eines d'atenció als clients, des dels tradicionals sistemes d'atenció telefònica o per correu electrònic fins als *issue trackers*, *helpdesks* i d'altres sistemes, el ventall d'opcions existents és força elevat.

Tot i així, no és un camp excessivament saturat, i hi ha espai per a la experimentació i la innovació. Per al Projecte de Final de Màster s'ha decidit implementar un sistema *Helpdesk* que ofereixi diverses eines de resolució d'incidències utilitzant metodologies i arquitectures modernes en el desenvolupament web, com són les Single Page Applications, les Restful APIs, NodeJS i bases de dades NOSQL i llenguatges de programació nous com Coffeescript, Stylus o ES6, però posant davant de tot la usabilitat i el rendiment de tots els components el sistema, per garantir una experiència òptima en tot tipus de dispositius.

Paraules clau

Helpdesk, Web app, Responsive, SPA (Single page application), RESTful API, NodeJS, NoSQL, Treball de Final de Màster

Notacions i Convencions

En l'elaboració de la memòria s'ha utilitzat la tipografia Arial amb una mida 10 punts per als continguts d'aquesta. En el redactat apareixen nombrosos termes en d'altres llengües, relacionats principalment amb conceptes tècnics. Aquests termes es presenten en cursiva.

Al llarg de la documentació també apareixen nombroses referències a codi informàtic o a algun directori dins d'un sistema d'arxius. Totes aquestes referències es presenten utilitzant la tipografia `Courier`. Si el codi forma part del text, és a dir, si només apareix una referència a una variable o instrucció dins d'una explicació, aquest es mostra en línia amb la resta del text. Per exemple: *la variable `foobar` s'utilitza per a...*

Si el bloc de codi presentat és més extens es mostra com en el següent exemple:

```
{
  “redis”: {
    "host": "localhost",
    "port": 6379,
    “db”: 4,
    “password”: “w9f376lefe445H/%E$&BTERTGgkfduy475t”
  }
}
```

En alguns capítols també apareixen instruccions que cal teclejar en la línia de comandes del sistema. El format és:

```
$  npm install grunt –g
```

Índex

Figures i taules

Índex de figures

Índex de taules

Capítol 1: Introducció

1.Introducció/Prefaci

L'atenció al client és un aspecte fonamental de qualsevol tipus de servei, presencial o no. Poder atendre les consultes i demandes dels clients d'una forma ràpida, eficient i organitzada és important per tal de fidelitzar els clients o usuaris i donar una imatge de marca positiva.

Els sistemes de *customer service* no són res nou. Des dels tradicionals sistemes d'atenció telefònica o per correu electrònic fins als *issue trackers*, *helpdesks* i d'altres sistemes, el ventall d'opcions existents és força elevat.

Tot i així, no és un camp excessivament saturat, ni en el que alguna solució en particular s'hagi establert com a estàndard *de facto*; per tant hi ha espai per a la experimentació i la innovació.

L'elecció de la temàtica del Treball Final de Màster ha estat motivada per la necessitat d'un sistema de les característiques plantejades per a un altre projecte en el que estic implicat. Aquest altre projecte és un sistema de gestió de continguts (un CMS), que un cop desenvolupat funcionarà com un servei que permetrà als usuaris sense coneixements tècnics implementar llocs web, amb blog, comerç electrònic i d'altres funcionalitats, enfocat a la facilitat d'ús, i plantejat tenint en compte la possibilitat de ser utilitzat des de qualsevol tipus de dispositiu. Per a aquest servei era necessari un sistema d'atenció als usuaris per tal de resoldre qualsevol tipus d'incidència dels clients.

S'han explorat diverses alternatives, com ara altres projectes de codi obert o d'altres serveis comercials preexistents. Els projectes de codi obert explorats no s'adeqüen a les necessitats del projecte, ja sigui per limitacions en relació a les prestacions requerides, o per qüestions tècniques, pel fet d'estar desenvolupats en llenguatges de programació diferents als utilitzats en el CMS, o dificultat d'integració amb aquest.

Després d'analitzar les necessitats en relació a l'atenció als usuaris, s'ha considerat que era més convenient desenvolupar una solució a mida, donat que la complexitat d'un projecte d'aquestes característiques no és molt elevat.

La complexitat d'un projecte com el plantejat i les competències necessàries per a desenvolupar-lo fan del projecte una bona opció per a ser desenvolupat com a Projecte Final de Màster, perquè és un projecte viable de realitzar en el temps que es disposa, requereix d'una varietat de coneixements i competències força ampli, com ara el disseny d'interfícies per a diferents tipus de dispositius, la usabilitat, o la programació web *frontend* i *backend* entre d'altres.

D'una banda s'ha considerat que una aplicació web *responsive* permet explorar a fons qüestions de disseny d'interfícies i usabilitat en diferents tipus de dispositius, adaptant el flux i els mecanismes d'interacció en diferents tipus de dispositius.

El fet de plantejar les aplicacions com Single Page Applications permet oferir una experiència més fluïda als usuaris al realitzar la càrrega del codi HTML, CSS i Javascript a l'inici i carregant posteriorment només les dades *pures* a partir de la API. Aquest fet, junt amb polítiques de *cache* agressives en el servidor, millora notablement la velocitat de càrrega percebuda per l'usuari.

Aquesta arquitectura permet explorar una metodologia alternativa a la tradicional en el desenvolupament web, basada en 'pàgines' individuals. També possibilita mitjans alternatius de comunicació amb el servidor, com ara els *websockets* o els *server sent events* que potencien la bidireccionalitat de les comunicacions.

2. Descripció/Definició

El treball de final de màster consisteix en el desenvolupament d'un HelpDesk (1), una aplicació de suport genèrica per als usuaris d'un determinat producte o servei, què oferirà una sèrie de vies de comunicació entre els usuaris i el personal d'atenció:

- **Sistema de tiquets:** Permet als usuaris obrir incidències i seguir-ne la seva resolució, amb respostes del personal d'atenció.

  L'aplicació permetrà als usuaris, un cop registrats, crear un tiquet, una incidència, en la que plantegi un problema amb un producte o servei, adjuntant una descripció, i opcionalment diversos recursos adjunts, com ara documents o imatges. El personal d'atenció al client rebrà una notificació i assignarà la incidència a algun dels seus agents. Un cop assignada la incidència, l'agent pot contestar a l'usuari proposant-li una possible solució o sol·licitar detalls addicionals. El sistema permet que l'usuari i l'agent intercanviïn tants missatges com sigui necessari fins a la resolució de la incidència.

  Quan l'usuari creï una incidència podrà assignar-li una categoria de les preestablertes pel personal de suport, un títol i unes paraules clau que permetran al personal de suport determinar quin és l'agent més adequat per resoldre la incidència, o cercar tiquets similars per a facilitar-ne la resolució.

  El personal de suport podrà assignar a les incidències un nivell de prioritat en funció de les característiques o la naturalesa d'aquesta.

  Les incidències passaran per un seguit d'estats, com ara 'pendent d'assignar', 'en tràmit', 'pendent de més detalls' o 'tancada'. Els administradors podran gestionar aquests estats i crear-ne de nous per adaptar el funcionament del sistema al seu flux de treball.

  Es podran configurar notificacions automatitzades per tal que les parts implicades (usuaris i agents) siguin informats quan una incidència canviï d'estat. Aquestes notificacions podran ser per correu electrònic o mitjançant indicadors en les aplicacions.

- **Knowledge base** (2)**:** Repositori d'articles de suport organitzats per temàtica. Permet als administradors i personal de suport publicar articles sobre l'ús d'un determinat producte o component, preguntes freqüents i d'altre tipus de material què pugui resultar d'ajuda als usuaris o clients.

  Els articles podran contenir textos, imatges, vídeos i s'hi podrà incorporar qualsevol tipus d'arxiu adjunt, disponible per a descàrrega per part dels usuaris.

  Aquest apartat és d'accés públic, no requerint el registre dels usuaris, i el seu objectiu és reduir la càrrega de treball del departament d'atenció al client permetent als usuaris l'accés a tutorials i respostes a problemes freqüents.

  A més de la categorització dels articles, l'usuari disposarà d'altres mecanismes per trobar els articles rellevants al seu problema, com ara un cercador o un núvol d'etiquetes.

- **Xat:** inicialment s'havia plantejat que el sistema disposés d'un *widget* integrable en qualsevol pàgina web que permet als usuaris comunicar-se en temps real amb el personal d'atenció. Durant l'evolució del projecte s'ha considerat apropiat eliminar aquesta funcionalitat del

projecte per manca de temps, i es desenvoluparà a posteriori, fora de l'àmbit del projecte de final de màster.

A nivell tècnic, el sistema estarà composat per diferents components:

- **Aplicació per als usuaris:** És una aplicació web, *responsive*, què permet als usuaris accedir als articles del *Knowledge base*, i registrar-se i crear tiquets, així con seguir-ne la seva evolució. Estarà desenvolupada com una *SPA*, una *Single Page Application* (3) (4), i es comunica amb una *RESTful API* (5).

- **Aplicació per al personal de suport:** Aplicació per al personal de suport: Com en el cas anterior, és una aplicació web, *responsive*, estructurada també com una *SPA*, que permet al personal d'atenció al client gestionar els tiquets i publicar articles en el *Knowledge base*.

  També disposarà d'un apartat d'estadístiques i informes que permetrà obtenir una visió global del sistema d'atenció, i obtenir un informe sobre la quantitat d'incidències en cada un dels seus estats, estadístiques d'incidències obertes per usuari o producte/àrea, informes de resolució d'incidències per agent, etcètera.

  Com en el cas anterior, és completament estàtica (no requereix cap component en el servidor) i es comunica amb la *RESTful API*.

- **API**: És el component *backend* del sistema, que permet la interacció dels diferents components, i l'accés a la base de dades. La API s'implementarà en Node, amb Mongo com a base de dades i Redis per a l'emmagatzemament de sessions i per temes de *cache*.

El sistema està plantejat d'una forma modular per tal de facilitar-ne la integració amb d'altres aplicacions web, tot i què es pot utilitzar de forma autònoma.

Les dues aplicacions web estan plantejades com a *SPA* per tal de facilitar-ne la integració en qualsevol web, i perquè aquest plantejament permet *encapsular* les aplicacions per convertir-les en aplicacions per a dispositius mòbils, utilitzant tecnologies com PhoneGap. El desenvolupament d'aquestes aplicacions mòbils no forma part del projecte, però s'ha tingut en compte el seu possible futur desenvolupament a l'hora de plantejar com es desenvoluparan les aplicacions web.

L'objectiu és desenvolupar un sistema de suport als usuaris fàcilment integrable en d'altres webs i alliberar els diversos components sota una llicència de codi obert, tot i què no es descarta la possibilitat d'explotar-ho de forma autònoma més endavant, oferint un servei similar a Freshdesk (6) o Zendesk (7).

3. Objectius generals

3.1 Objectius principals

Objectius de l'aplicació/producte/servei:

- Oferir diversos mecanismes d'atenció i resolució d'incidències als usuaris o clients d'una determinada empresa o servei.
- Fer accessible aquest servei en qualsevol tipus de dispositiu, de la forma més adequada en cada un dels mitjans.

Objectius per al client/usuari:

- Accés a documentació i material de suport de diferents tipus, com ara articles, tutorials o preguntes freqüents sobre un servei d'una forma àgil i còmoda.
- Accés a un canal de comunicació amb l'empresa o servei què li permeti expressar dubtes o notificar incidències, així com realitzar un seguiment del procés de resolució d'aquestes.

Objectius personals de l'autor del TF:

- Implementar el sistema amb una arquitectura modular i oberta de forma que resulti senzill la integració amb d'altres sistemes i una futura ampliació de les seves funcionalitats.
- Obtenir un sistema d'atenció als usuaris per a un projecte en el què estic implicat.
- Aprofundir en el coneixement de tecnologies i tècniques de desenvolupament web alternatives a les "tradicionals", com l'arquitectura *SPA*, les bases de dades no relacionals o el desenvolupament *serverside* amb Node.
- Aplicar el conjunt de coneixements adquirits durant el màster i el grau en el desenvolupament d'una aplicació, especialment aspectes com ara el disseny d'interfícies per a diferents tipus de dispositius, la usabilitat, o la programació web *frontend* i *backend* entre d'altres.

3.2 Objectius secundaris

Objectius addicionals que enriqueixen el TF.

- Contribuir a la comunitat *opensource* de la què he estat beneficiari durant molts anys alliberant les aplicacions que componen el sistema de forma que tercers se'n puguin beneficiar sense cap tipus de limitació.

4. Metodologia i procés de treball

Per afrontar el projecte s'ha considerat més adient desenvolupar-lo des de zero que partir d'un projecte preexistent perquè no s'han trobat projectes adients i perquè implementar-ho des de zero dona més flexibilitat i és més motivador, ja que implica moltes més responsabilitats al haver de dissenyar el sistema complert, i permet experimentar i innovar en major grau. En realitat, per a les aplicacions web no s'ha partit completament des de zero, sinó que s'ha aprofitat el disseny de l'arquitectura de l'aplicació i diverses llibreries d'un altre projecte què he estat desenvolupant en paral·lel.

En el desenvolupament del projecte s'ha utilitzat una metodologia àgil, concretament *scrum* (8). A l'inici del projecte s'ha realitzat la presa de requisits, l'anàlisi a alt nivell, el disseny general de l'arquitectura del sistema i la planificació global. El projecte s'ha dividit en diversos *sprints,* un per a cada funcionalitat a implementar (com ara la autenticació, la pujada d'arxius, el *knowledge base*, etc.).

En cada *sprint* s'ha realitzat l'anàlisi detallat de la funcionalitat a implementar, el seu disseny, la implementació i les proves corresponents.

Tot i què el projecte està composat per tres aplicacions diferenciades més o menys independents, el seu desenvolupament s'ha afrontat de forma global, en paral·lel, per funcionalitats.

Els *sprints* s'han agrupat en *milestones,* que han coincidit amb les PAC (veure planificació del projecte).

Per a gestionar el projecte s'ha creat un projecte a Github per a cada un dels components del sistema[1] (la API, l'aplicació dels usuaris, la dels administradors, i la documentació). Els projectes de Github s'han utilitzat per al control de versions, i per a organitzar les tasques (en forma de *issues* i *milestones*). A més, s'ha creat un *scrum board* per a cada projecte en un servei anomenat waffle.io (9), i també s'ha utilitzat el *Project* amb la planificació del projecte desenvolupat a l'inici d'aquest per a fer el seguiment.

Per tal d'optimitzar el temps s'han automatitzat totes les tasques possibles amb una eina anomenada Grunt (10), des de la compilació del codi (les aplicacions dels usuaris i els administradors estan desenvolupades en Coffeescript, Stylus i Handlebars, i per tant és necessari *transpilar-les* (11) (12) a Javascript, CSS i HTML), la execució de tests unitaris o d'altres tasques auxiliars.

Pel què fa al testeig, aquest s'ha desenvolupat en finalitzar la implementació de cada una de les funcionalitats. Inicialment s'han implementat tests unitaris i d'integració, però a mesura que ha avançat el projecte ha estat necessari deixar de banda l'escriptura de nous tests per manca de temps,

[1] Els projectes ara mateix són privats, d'accés restringit. En concloure el màster i acabar d'implementar els tests que manquen s'obriran al públic general.

i la manca de tests automatitzats s'ha suplert amb tests manuals, utilitzant diversos navegadors i dispositius diferents, en el cas de les aplicacions web, i desenvolupant una col·lecció de peticions a la API amb totes les peticions que aquesta accepta amb una eina anomenada Postman (13).

Finalment, per verificar el funcionament en un entorn *net,* s'ha configurat un servidor de proves, només amb els components imprescindibles per a fer funcionar les aplicacions (a més dels mecanismes de seguretat necessaris com un tallafoc o un certificat SSL) per tal de validar el correcte funcionament del sistema en un entorn diferent del de desenvolupament.

En aquest servidor s'han configurat dos grups de llocs web amb bases de dades independents, els de *staging*, per a realitzar proves, i els de producció, on s'ha publicat les aplicacions per a la avaluació de les PAC.

El procés complert de desenvolupament per a cada una de les funcionalitats ha estat el següent:

- Anàlisi detallat i disseny de la implementació
- Definició de les subtasques necessàries (i creació dels *issues* corresponents a Github)
- Disseny UI/UX
- Desenvolupament i testeig de cada subtasca (per a cada tasca s'ha creat una branca independent en Git, que s'han anat fusionant amb una branca anomenada *develop* al completar la subtasca)
- Proves (automatitzades i manuals) en local de tot el conjunt de tasques
- Testeig en el servidor de proves, en els llocs de *staging*
- *Release*: fusió de la branca *develop* amb la branca *master*
- Testeig automatizat en el cas de la API en un sistema d'integració contínua anomenat Travis CI (14)
- Publicació en els llocs web de producció i proves manuals

Com es pot apreciar, les fases de testeig i documentació no estan diferenciades de les de desenvolupament, sinó que formen una unitat.

5. Planificació

Inicialment el sistema havia d'estar composat per quatre aplicacions diferenciades (API, aplicació dels usuaris, aplicació dels agents/administradors i *widget* de xat), i es va considerar que seria més convenient estructurar el projecte basant-se en les funcionalitats principals, ja que totes elles impliquen la majoria de les aplicacions.

En la fase inicial del projecte s'han desenvolupat algunes tasques que afecten a tot el sistema, com la presa de requisits, el disseny a alt nivell del sistema, la base de les aplicacions i de la API.

La resta del projecte, que inclou la implementació de les funcionalitats del sistema (*knowledge base*, tiquets, i d'altres funcionalitats menors) s'han afrontat com a mini projectes (en quant a les tasques, pel que fa a la gestió és global), cada un amb les seves tasques de disseny lògic, UI/UX, la implementació de les funcionalitats corresponents en cada una de les aplicacions implicades, el testeig i la documentació, dedicant la segona etapa del projecte a la implementació del *knowledge base*, la tercera al sistema de tiquets i part del xat, i per a la última s'havia previst acabar el desenvolupament del *xat* i completar la memòria.

Per tant, la divisió del treball està orientada a fites, agrupant les funcionalitats i processos en funció del calendari d'entregues.

Donada la complexitat del projecte s'han pres algunes precaucions en la planificació, deixant espais de temps entre algunes de les etapes per a solucionar petites desviacions, i s'havia planificat per a l'última etapa el component del xat, la part menys important del projecte, per tal que si la desviació és més important aquest s'eliminaria del projecte.

Finalment, durant la tercera etapa s'ha decidit suprimir el component del *xat*, perquè el sistema de tiquets ha resultat ser més complex del previst, i s'ha considerat més convenient concentrar els recursos de temps disponibles en aquest component i en acabar de refinar la resta del sistema.

Les etapes en les què s'ha desenvolupat el projecte han estat:

- **Primera etapa: Proposta**
  En aquesta etapa s'han proposat diverses alternatives per al projecte i s'ha definit el concepte de l'aplicació i les seves característiques a alt nivell.
- **Segona etapa: Mandat del projecte i planificació**
  Durant la segona etapa s'han completat les tasques d'iniciació del projecte. Durant aquesta fase s'ha realitzat un estudi de la competència, s'ha fet la presa de requisits i s'han definit els objectius i abast del projecte. També s'ha elaborat la planificació del projecte, determinant els recursos necessaris per a cada tasca i la seqüenciació i temporització d'aquestes.

Finalment s'ha realitzat un disseny a alt nivell de tot el sistema, deixant per a cada etapa el disseny detallat de cada una de les funcionalitats a implementar.

- **Tercera etapa: PAC3 – Entrega 1**
  Aquesta fase ha representat l'inici de les tasques de desenvolupament. Les fites més importants de l'etapa han estat la implementació de la estructura de les dues aplicacions web i la API, i el desenvolupament del *knowledge base*.
- **Quarta etapa: PAC4 – Entrega 2**
  Durant la quarta etapa del projecte s'ha desenvolupat el sistema de tiquets, i també s'han desenvolupat d'altres tasques menors com la pujada d'arxius. En aquesta etapa s'ha produït una desviació de temps important derivada de la incorrecta estimació de la complexitat del sistema de tiquets, que ha suposat una replanificació de la temporització i l'abast del projecte.
- **Cinquena etapa: Tancament**
  En aquesta etapa estava previst completar el sistema de xat, que s'hauria d'haver iniciat en l'etapa anterior, i completar la resta de tasques obertes, així com completar la redacció de la memòria. Degut al canvi en la planificació sorgit en la fase anterior, aquesta etapa ha estat dedicada a completar el sistema de tiquets, implementar diverses millores sobre diverses mancances de les aplicacions detectades durant el desenvolupament, i a completar la redacció de la memòria.

El calendari de fites del projecte, després dels ajustos en la planificació de la quarta etapa és el següent:

Nom	Durada	Inici	Final
Gestió projecte	91 dies	05/10/2015	03/01/2016
Iniciació projecte	31 dies	30/09/2015	08/10/2015
Estudi competència/productes existents	03 dies	30/09/2015	03/10/2015
Definició projecte	04 dies	30/09/2015	04/10/2015
Presa de requisits	< 1 dia	06/10/2015	06/10/2015
Disseny a alt nivell del sistema	< 1 dia	06/10/2015	07/10/2015
Disseny a alt nivell API i web apps	< 1 dia	10/10/2015	10/10/2015
PAC3 - Entrega 1	28 dies	20/10/2015	16/11/2015
Estructura base API	04 dies	10/10/2015	10/10/2015
Estructura base aplicacions web	13 dies	10/10/2015	10/10/2015
Autenticació	11 dies	10/10/2015	10/10/2015
Knowledge base	12 dies	10/10/2015	10/10/2015

Nom	Durada	Inici	Final
PAC4 - Entrega 2	28 dies	17/11/2015	14/12/2015
Uploads	05 dies	16/11/2015	20/11/2015
Tickets	21 dies	21/11/2015	11/12/2015
Tancament	20 dies	15/12/2015	04/01/2016
Millores (dashboard, erdenació articles, etc.)	9 dies	11/12/2015	20/12/2015
Memòria	88 dies	05/10/2015	31/12/2015
Redacció i correcció de la memòria	88 dies	05/10/2015	31/12/2015
Presentació	6 dies	26/12/2015	01/01/2016

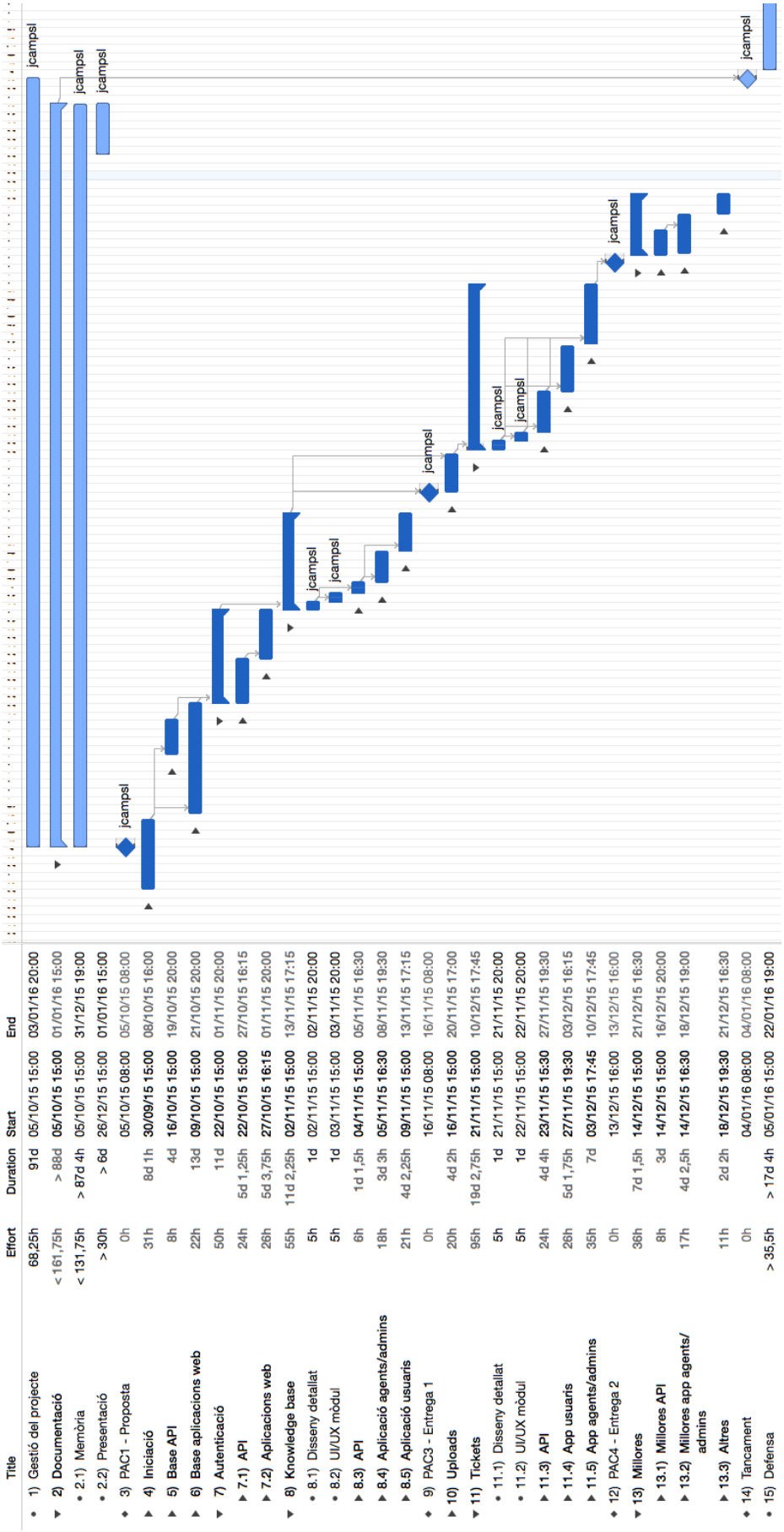
Taula 1 - Fites del projecte

En la pàgina següent es mostra el diagrama de Gantt del projecte amb la seqüenciació dels grups de tasques, així com les durades i dates d'inici i finalització.

La durada d'algunes de les tasques poden semblar poc realistes, concretament el disseny de l'arquitectura de les aplicacions i la implementació d'alguns components tenen unes durades baixes.

Aquest fet s'explica perquè el disseny de l'estructura de les aplicacions i la API es basa en uns dissenys anteriors per al sistema de gestió de continguts per al que s'ha plantejat el *Helpdesk*. En realitat és una arquitectura força genèrica, no específica del CMS, però que resulta molt adequada per al projecte com a base. Pel què fa a algunes tasques de programació amb estimacions baixes, la justificació de nou és l'experiència anterior amb l'altre projecte, i al fet de construir alguns d'aquests components sobre una base (llibreries i mòduls) prèviament desenvolupada.

El calendari està dissenyat partint de la premissa que es dedicarien cinc hores diàries al projecte, els set dies de la setmana, tot i que a la pràctica la dedicació diària ha estat superior.



Title	Effort	Duration	Start	End
1) Gestió del projecte	68,25h	91d	05/10/15 15:00	03/01/16 20:00
2) Documentació	< 161,75h	> 88d	05/10/15 15:00	01/01/16 15:00
2.1) Memòria	< 131,75h	> 87d 4h	05/10/15 15:00	31/12/15 19:00
2.2) Presentació	> 30h	> 6d	26/12/15 15:00	01/01/16 15:00
3) PAC1 - Proposta	0h		05/10/15 08:00	05/10/15 08:00
4) Iniciació	31h	8d 1h	30/09/15 15:00	08/10/15 16:00
5) Base API	8h	4d	16/10/15 15:00	19/10/15 20:00
6) Base aplicacions web	22h	13d	09/10/15 15:00	21/10/15 20:00
7) Autenticació	50h	11d	22/10/15 15:00	01/11/15 20:00
7.1) API	24h	5d 1,25h	22/10/15 15:00	27/10/15 16:15
7.2) Aplicacions web	26h	5d 3,75h	27/10/15 16:15	01/11/15 20:00
8) Knowledge base	55h	11d 2,25h	02/11/15 15:00	13/11/15 17:15
8.1) Disseny detallat	5h	1d	02/11/15 15:00	02/11/15 20:00
8.2) UI/UX mòdul	5h	1d	03/11/15 15:00	03/11/15 20:00
8.3) API	6h	1d 1,5h	04/11/15 15:00	05/11/15 16:30
8.4) Aplicació agents/admins	18h	3d 3h	05/11/15 16:30	08/11/15 19:30
8.5) Aplicació usuaris	21h	4d 2,25h	09/11/15 15:00	13/11/15 17:15
9) PAC3 - Entrega 1	0h		16/11/15 08:00	16/11/15 08:00
10) Uploads	20h	4d 2h	16/11/15 15:00	20/11/15 17:00
11) Tickets	95h	19d 2,75h	21/11/15 15:00	10/12/15 17:45
11.1) Disseny detallat	5h	1d	21/11/15 15:00	21/11/15 20:00
11.2) UI/UX mòdul	5h	1d	22/11/15 15:00	22/11/15 20:00
11.3) API	24h	4d 4h	23/11/15 15:30	27/11/15 19:30
11.4) App usuaris	26h	5d 1,75h	27/11/15 19:30	03/12/15 16:15
11.5) App agents/admins	35h	7d	03/12/15 17:45	10/12/15 17:45
12) PAC4 - Entrega 2	0h		13/12/15 16:00	13/12/15 16:00
13) Millores	36h	7d 1,5h	14/12/15 15:00	21/12/15 16:30
13.1) Millores API	8h	3d	14/12/15 15:00	16/12/15 20:00
13.2) Millores app agents/admins	17h	4d 2,5h	14/12/15 16:30	18/12/15 19:00
13.3) Altres	11h	2d 2h	18/12/15 19:30	21/12/15 16:30
14) Tancament	0h		04/01/16 08:00	04/01/16 08:00
15) Defensa	> 35,5h	> 17d 4h	05/01/16 15:00	22/01/16 19:00

Figura 1 - Diagrama de Gantt del projecte

6. Pressupost

El càlcul del pressupost es basa en la dedicació al projecte d'una persona al llarg del període que ha durat el projecte. Tot i què el sistema es pot considerar complet i és completament funcional, un cop finalitzat el màster, aquest continuarà evolucionant, inicialment per a implementar tests automatitzats i acabar de netejar el codi abans d'alliberar els components com un projecte de codi lliure, i posteriorment per afegir noves funcionalitats o millorar les existents.

Donada la dificultat de l'estimació per endavant del cost econòmic d'un projecte de codi obert, sobretot tenint en compte que encara no s'ha definit un pla detallat sobre l'evolució del projecte en el futur, el pressupost només contempla els costos de la part desenvolupada en el marc del projecte de final de màster.

El projecte, excloent les tasques de redacció de la memòria, elaboració de la presentació i proposta inicial ha representat unes 350 hores de feina. En aquest cas no té massa sentit separar les hores per tipus de tasca, ja que totes tenen el mateix valor, i han estat realitzades per una mateixa persona.

Concepte	Hores	€/hora	Total
Definició del projecte	30	40	1.200€
Desenvolupament del sistema	320	40	12.800€
Total			**14.000€**

Taula 2 - Pressupost

La definició del projecte inclou les tasques d'anàlisi, estudi de la competència i disseny conceptual del sistema.

Les tasques de desenvolupament inclouen el disseny detallat de cada component, el disseny gràfic, la programació, testeig i altres tasques necessàries. No s'ha considerat oportú desglossar-ho perquè totes les tasques tenen el mateix cost econòmic, i per tant aquesta informació no té cap valor en aquest cas.

Pel que fa als recursos dedicats al projecte, es pot considerar que el cost ha estat nul perquè a part de l'ordenador utilitzat per a desenvolupar el sistema, la resta de recursos utilitzats no tenien cap cost econòmic associat. Concretament, tot el software utilitzat (editor de codi, llibreries i *frameworks*) és de codi lliure, i l'accés als recursos necessaris per a la *demo* (servidor, domini, certificat SSL, enviament de correus a través del servei Sendgrid (15) i allotjament massiu d'arxius a Amazon S3 (16) (17)) s'ha obtingut gratuïtament a través del programa Github Education (18).

En quant al cost de posar en producció el sistema, l'import depèn del volum d'usuaris que han de servir les aplicacions. L'aplicació dels usuaris i la dels administradors no requereixen d'un servidor amb unes característiques concretes, ja que són aplicacions web completament estàtiques. Per tant, qualsevol servei de *hosting* és vàlid.
La API si té uns requisits particulars, i la opció d'allotjament més adequada és un servidor dedicat, que pot ser virtual. Tot i així té uns requisits de hardware molt modestos, de forma que els plans més bàsics són suficients en la majoria dels casos.
Per exemple, la *demo* està allotjada en un servidor virtual amb 512 MB de memòria i 20 GB de disc dur en una companyia anomenada Digital Ocean (19). El cost normal (en el cas del projecte ha estat zero per la promoció d'estudiants de Github) d'un servidor d'aquestes característiques és de 5 dòlars mensuals.

Opcionalment, la API es pot configurar de forma que utilitzi sistemes de tercers per a l'enviament de correus i emmagatzematge dels arxius. En el cas del correu suporta diversos proveïdors i en el cas de l'allotjament d'arxius només Amazon S3.

Sendgrid, el servei utilitzat en la demo, ofereix un pla gratuït que permet enviar 12000 correus mensuals, i diversos plans de pagament a partir de 10 dòlars mensuals (20).

En el cas de Amazon S3 els preus varien en funció de la quantitat d'espai requerit, i a partir de 0.03 dòlars per GB (21).

7. Estructura de la resta del document

El Capítol 2 presenta un anàlisi de solucions existents, defineix el públic objectiu del sistema i en descriu detalladament les seves característiques.

El Capítol 3 descriu l'arquitectura del sistema i exposa el procés de disseny d'aquest i les tecnologies utilitzades.

En el Capítol 4 es descriuen els requisits tècnics del sistema i es descriuen els passos necessaris per a la seva instal·lació i configuració.

El Capítol 5 presenta les instruccions d'ús de l'aplicació, i proporciona els detalls d'accés a la demostració que s'ha preparat. També descriu els prototips creats durant la fase de desenvolupament i detalla els tests realitzats.

El Capítol 6 presenta les conclusions del projecte i possibles línies de futur per a l'evolució del sistema.

Capítol 2: Anàlisi

1. Estat de l'art

Els sistemes de *customer service* no són res nou. Des dels tradicionals sistemes d'atenció telefònica o per correu electrònic fins als *issue trackers*, *helpdesks* i d'altres sistemes, el ventall d'opcions existents és força elevat.

En relació al projecte, els sistemes més rellevants són els tipus *helpdesk* i els *issue trackers*. En el fons el funcionament dels dos tipus de sistemes és força similar, basats principalment en sistemes de *tickets* o incidències, obertes pels usuaris (els clients en el primer cas, desenvolupadors en el segon), i que permeten seguir el procés de resolució d'aquesta incidència.

Els serveis tipus *helpdesk* poden incorporar funcionalitats addicionals per resoldre les incidències, com ara *knowledge bases*, és a dir, repositoris d'articles d'ajuda, com ara preguntes freqüents, tutorials i d'altre tipus de material, així com fòrums, sistemes d'atenció en temps real (com ara atenció telefònica o *xats*), formularis de contacte i d'altres.

1.1 Serveis Help desk

Existeix diverses solucions tipus Help Desk, la majoria dels quals són serveis comercials que operen sota un model SAAS (22).

Alguns dels més rellevants són:

- **Zendesk** (23) (7)
  Zendesk ofereix comercialment diverses solucions d'atenció al client. Ofereixen diversos paquets que inclouen diferents eines de suport, com ara un sistema de *tickets* o *knowledge bases*.

  Alguns dels aspectes a destacar de la seva oferta és la naturalesa multicanal del sistema de *tickets*, que permet obrir-ne a través de diversos canals (com ara web, xarxes socials, correu electrònic i altres), la integració amb sistemes de tercers, la usabilitat i la cura en els detalls.

- **Freshdesk** (6) (24)
  L'oferta de Freshdesk és força similar a la de Zendesk (de fet l'any 2011 es va produir una polèmica quan el CEO de Zendesk va acusar a Freshdesk de copiar-los-hi el producte (25).

Addicionalment ofereixen una altre servei anomenat *Mobihelp* (26), que permet integrar els seus serveis en aplicacions mòbils, subministrant *SDKs* per a IOS i Android.

- **Help scout** (27)
  Ofereix un paquet de solucions molt similar als dos anteriors, amb un sistema de *tickets*, un *knowledge base* i integracions amb tercers serveis.

- **Desk.com** (28) (29)
  Anteriorment anomenat *Assistly*, fou adquirit per *Salesforce* i posteriorment renombrat com *Desk.com*.
  Ofereix un paquet de serveis als descrits en els competidors anteriors, i addicionalment ofereix integracions amb els altres productes de *Salesforce*.

- **osTicket** (30)
  osTicket és una aplicació web basada en *tickets*, *opensource*, distribuïda sota la GNU General Public License ("GPL"), desenvolupada en PHP i mantinguda principalment per *Enhancesoft*.
  L'empresa també explota la solució *allotjant* ells l'aplicació per als clients que prefereixen no allotjar-la en els seus propis servidors.

1.2 Issue trackers

Hi ha disponibles un gran nombre de solucions tipus *issue trackers*.
Pràcticament tots aquests serveis estan plantejats per ser utilitzats en l'àmbit del desenvolupament de software, i resulten poc adequats com a solució genèrica d'atenció al client en d'altres àmbits, per la seva complexitat o per la forma com estan plantejats.

En aquesta categoria existeix molta més diversitat que en l'anterior pel que fa al model de distribució. Existeixen diverses solucions *standalone* i d'altres integrades dins de sistemes que ofereixen d'altres funcionalitats, com ara allotjament de codi font en sistemes de control de versions.

Algunes d'aquestes solucions són:

- **JIRA** (31) (32)
  És un servei comercial desenvolupat per *Atlassian* que inclou funcionalitats de *issue tracking* i gestió de projectes, per a projectes de desenvolupament de software.

- **Github** (33)
  Github és un servei web què proporciona allotjament de repositoris Git.
  És un servei comercial (tot i què és gratuït per a projectes de codi obert) que funciona sota el

model SaaS.
Ofereix un conjunt de funcionalitats addicionals, com *wikis* i un *issue tracker* vinculat al repositori.

- **Bugzilla** (34) (35)
  És un *bug tracker* desenvolupat per la Fundació Mozilla, sota una llicència *Mozilla Public License*.

- **GNATS** (36) (37)
  És un *issue tracker* del projecte GNU, sota llicència GPL, creat l'any 1992, utilitzat per a projectes de desenvolupament de software, principalment projectes de codi obert, com alguns dels components de GNU, NetBSD o FreeBSD, entre d'altres.

- **Mantis Bug Tracker** (38) (39)
  És un *bug tracker* sota una llicència GNU General Public License, que a més de funcionar com a *issue tracker* també pot actuar com a *project management system*.

1.3 Sistemes de suport en temps real

La majoria d'aquests serveis proporcionen mecanismes per interactuar amb els agents de suport en temps real mitjançant *xats*.

Dos dels serveis d'aquest tipus més representatius són:

- **Zopim** (40) (41)
- **LiveChat** (42) (43)

Les solucions ofertes per les dues empreses són força similars. Ofereixen un servei de *live suport* sota un model *Software as a service*.

El sistema permet afegir una petita finestra flotant en qualsevol lloc web per tal què els usuaris puguin comunicar-se en temps real amb el personal de suport.

Les dues empreses ofereixen també la monitorització en temps real del tràfic al lloc i del comportament dels visitants en el web.

2. Públic objectiu i perfils d'usuari

A curt termini no existeix la intenció d'explotar el sistema comercialment. De moment aquest s'integrarà dins d'una aplicació més gran, un gestor de continguts, per a oferir atenció als potencials usuaris, i a mitjà termini un com implementats els tests que falta implementar, el *helpdesk* s'alliberarà com un projecte de codi lliure.

Inicialment, el públic objectiu del sistema en principi està format per desenvolupadors independents o agències, que necessitin integrar una solució *helpdesk* per a oferir atenció als usuaris finals d'un servei o producte.

Posteriorment, quan el sistema estigui més madur i estable, es desenvoluparà un lloc web per al sistema, amb detallant-ne les prestacions i amb instruccions d'instal·lació, ús i personalització per tal d'accedir a un públic més ampli, no limitat a desenvolupadors familiaritzats amb el món del codi lliure i Github.

Per tant, el públic objectiu de l'aplicació és qualsevol individual o empresa que necessiti oferir un sistema de suport a través d'internet als seus clients o usuaris.

Pel que fa als usuaris finals de les aplicacions, és convenient diferenciar entre els usuaris de l'aplicació dels administradors i agents i la dels usuaris finals. En els dos casos és molt difícil establir un perfil dels usuaris perquè la solució és molt genèrica, i per tant, el perfil dels usuaris finals depèn de l'empresa o servei en la que s'integri el sistema per oferir atenció als seus usuaris o clients.

En el primer cas, els seus usuaris finals són el personal encarregat de l'atenció als clients de l'empresa o servei que integri la solució. No és un grup amb un perfil molt definit a nivell demogràfic o cultural, però es pot assumir que tenen un mínim nivell de competències tècniques i estan familiaritzats amb diverses metodologies d'atenció al públic i de resolució d'incidències.

El públic final de l'aplicació dels usuaris és encara més indefinit i heterogeni. Són usuaris o clients d'un producte o servei que necessiten informació sobre aquest o requereixen atenció per tal de resoldre algun dubte o incidència.

3. Definició d'objectius/especificacions del producte

L'objectiu principal del sistema és oferir diversos mecanismes de resolució d'incidències als usuaris o clients d'un servei, ja sigui de forma autònoma, mitjançant el *knowledge base,* com de forma assistida per un agent, mitjançant els sistema de tiquets.

3.1 Especificacions de l'aplicació dels usuaris

El *helpdesk* ha de permetre als usuaris accedir a diversos tipus de material de suport, com ara articles, tutorials o preguntes freqüents, formats bàsicament per text, però què poden incloure també imatges, vídeos o arxius adjunts de qualsevol tipus.

El sistema permet categoritzar aquests material, així com afegir-hi etiquetes per facilitar als usuaris la cerca de continguts similars. El *knowledge base* inclou també un cercador per agilitzar l'accés als continguts rellevants.

L'accés al *knowledge base* és obert, és a dir, no requereix registre per part dels usuaris.

Per accedir al sistema de tiquets els usuaris s'han de registrar. El registre és obert, qualsevol usuari es pot registrar introduint només un usuari, una contrasenya i una direcció de correu electrònic, que ha de ser vàlida, perquè un cop completat el registre, el compte de l'usuari no s'activarà fins que aquest validi la direcció accedint a una url especial que se li enviarà a la direcció de registre.

Un cop registrat l'usuari pot modificar la direcció de correu i credencials d'accés, així com completar el seu perfil amb detalls addicionals, com el nom complert, la ubicació o una fotografia. De moment només els agents tenen informació aquesta informació, per tal que puguin oferir un servei més personal als usuaris, tot i que part d'aquesta, com ara la fotografia o el nom es podria utilitzar més endavant si s'implementen millores o nous mòduls al sistema, com ara comentaris, un fòrum o un *wiki.*

En el cas que l'usuari oblidi les credencials d'accés, el sistema permet establir una nova contrasenya accedint a una url que se li envia a la direcció de correu vinculada al compte.

Tant la url per resetejar la contrasenya com per activar el compte són temporals; només es poden utilitzar un cop i caduquen automàticament al cap de N hores de crear-se.

Els usuaris identificats poden crear *tiquets*, és a dir, poden crear una incidència per tal d'obtenir informació o assistència d'un agent. Poden crear tantes incidències com vulguin, i per a fer-ho només han d'introduir un títol i una descripció. En el cas que els administradors del sistema hagin definit diferents categories o departaments, també n'hauran de seleccionar una. Opcionalment els usuaris poden adjuntar imatges o arxius de qualsevol mena al tiquet.

Els usuaris poden llistar totes les incidències que han creat, categoritzades entre obertes, és a dir en procés de resolució, o tancades. Inicialment els tiquets es mostren ordenats per la data de modificació, és a dir, l'últim cop que l'usuari o l'agent va modificar el tiquet, tot i que l'usuari pot ordenar el llistat en funció de qualsevol dels paràmetres del tiquet.

El sistema també ofereix un cercador per als tiquets, que permet cercar per l'id dels tiquets, el títol, la descripció o els comentaris.

Es usuaris poden afegir nous comentaris amb indicacions o observacions addicionals als tiquets oberts. Els comentaris són bàsicament textuals, i opcionalment poden contenir arxius adjunts.

Els tiquets oberts poden ser tancats pels usuaris, introduint un comentari per tal que l'agent assignat pugui ser notificat del motiu de tancament. Un cop tancats, els tiquets poden ser reoberts en cas que es consideri que la incidència no ha estat resolta del tot.

Els usuaris poden visualitzar l'evolució del tiquet, és a dir, per quins estats ha passat, en quin moment ha canviat d'estat, i si l'agent ha introduït el motiu el perquè.

Quan un agent actualitzi un tiquet (afegint-hi un comentari o canviant-ne l'estat) l'usuari serà notificat per correu electrònic.

3.2 Especificacions de l'aplicació dels administradors i agents

El *dashboard*, o aplicació dels administradors i agents és d'accés restringit, i només ha de permetre l'accés als usuaris identificats.

L'aplicació no permet el registre dels administradors i els agents. L'administrador es crea durant el procés de configuració de l'aplicació, i posteriorment aquest pot donar d'alta nous agents, que hauran d'activar el comte seguint un procés similar al detallat en l'aplicació anterior. L'administrador també pot eliminar els agents.

Els administradors i els agents poden gestionar les seves dades amb un mecanisme idèntic al de l'aplicació del usuaris finals.

Tant els administradors com els agents poden gestionar el *knowlege base*. Poden crear i modificar categories i articles, i determinar-ne l'ordre d'aparició.

En la creació i edició dels articles, poden definir-ne el títol i els continguts, així com d'altres característiques, com ara la categoria o les etiquetes. Els articles es poden definir com a publicats (visibles per als usuaris finals), no publicats, i també es pot definir la data de publicació, de forma que l'article romandrà com a no publicat fins a la data especificada, em que es publicarà automàticament.

L'aplicació proporciona un editor simple del tipus proporciona un editor simple del tipus *wysiwyg* (44) per a editar els continguts de l'article, que permet als autors definir el format del text i introduir imatges.

Els autors també poden adjuntar documents de qualsevol tipus als articles que podran ser descarregats pels usuaris.

Els administradors poden personalitzar el flux de resolució dels tiquets definint diverses categories o departaments i assignant-ne els agents, i definir els estats per els poden passar els tiquets.

Els agents poden llistar els tiquets sense assignar dels departaments als que estiguin vinculats i autoassignar-se'ls. També poden llistar els tiquets que tenen assignats, classificats entre els que estan en procés de resolució i els tancats.

Els agents poden editar alguns paràmetres dels tiquets assignats oberts, com ara canviar-ne la prioritat, l'estat o assignar-hi etiquetes. També poden afegir comentaris visibles per als usuaris (addicionalment l'usuari rebrà un correu) i notes privades.

Tant els administradors com els agents tenen accés a un apartat d'estadístiques dels tiquets que permet formar-se una idea de l'estat del sistema en el moment actual i la seva evolució.

3.3 Especificacions comuns de les aplicacions web

Les dues aplicacions web són estàtiques, és a dir, no requereixen programació *serverside* ni base de dades, i es comuniquen amb una API Restful, de forma que es poden publicar en qualsevol tipus de servidor per facilitar-ne la integració amb qualsevol tipus de web. Aquest fet juntament a la seva arquitectura *Single Page Application* possibilitaria el desenvolupament d'aplicacions per a mòbil utilitzant tecnologies com PhoneGap (45) (46).

Les aplicacions són multiidioma. Tots els textos estan separats en *locales* independents, fàcilment editables. De moment les llengües disponibles són el català, el castellà i l'anglès[2]. Durant el procés de configuració de les aplicacions, l'administrador defineix la llengua de les aplicacions.

Les aplicacions web són *responsive*, amb *layouts* i mecanismes d'interacció optimitzats per a cada tipus de dispositiu.

Les aplicacions tenen una arquitectura modular per tal de facilitar-ne una futura expansió amb nous mòduls o funcionalitats.

[2] S'ha detectat que el cercador no torna correctament els resultats al realizar cerques amb caràcters especials, com ara accents. S'estan realitzant proves amb la última versió de Mongo en la que s'ha modificat el comportament dels índex textuals per tal de trobar una solució.

Capítol 3: Disseny

1. Arquitectura general del sistema

El sistema està format per tres aplicacions: una API Restful, implementada en Node, i dues aplicacions web estàtiques. El sistema en general segueix el principis del *Thin server architecture* (47) (48), i el *API First Design* (49), és a dir, la API és completament independent de la resta de components del sistema (el *helpdesk* i el *dashboard*), és només una capa per sobre de les bases de dades que gestiona l'accés a les dades. A part de la gestió dels documents la API només s'encarrega de l'autenticació, autorització i validació per protegir la integritat de les dades, delegant a les aplicacions clients d'altres tasques com mantenir l'estat, renderitzat de vistes, i d'altres operacions complexes.
Aquesta arquitectura permet reduir significativament la càrrega del servidor, perquè gran part de les tasques que aquest hauria de desenvolupar en d'altres arquitectures ara es mouen a les aplicacions client. Per tant proporciona una major modularització i separació de responsabilitats.
El fet que les aplicacions clients siguin completament independents del *backend* dona molta més flexibilitat a l'hora d'implementar-les, tant en el disseny i funcionament d'aquestes. És a dir, en un futur es podria desenvolupar per exemple una aplicació nativa per a telèfons mòbils o per a qualsevol altre tipus de dispositiu, o es podria permetre l'accés a la API a aplicacions de tercers que volguessin integrar el servei. sense necessitat de realitzar cap canvi en el *backend*.

3.1.1 API

La API és el component central i gestiona l'autenticació dels usuaris, l'accés i manipulació de les dades, l'emmagatzemament i altres activitats secundàries com l'enviament de notificacions per correu electrònic.

És una aplicació implementada en javascript (fet que possiblilita la reutilització de part del codi entre el *backend* i el *frontend*) utilitzant una *framework* anomenada Express (50) i que exposa una sèrie d'entitats o tipus de documents (com ara articles, categories, tiquets, etc.) seguint una arquitectura REST. Té una estructura completament modular per tal de poder integrar fàcilment i de forma independent altres funcionalitats com els *websockets* o la possibilitat que també serveixi pàgines HTML. La pròpia API té una estructura modular, amb cada grup de rutes encarregades de gestionar una determinada entitat implementades com un mòdul independent, per tal de facilitar-ne l'extensió i la escalabilitat.

Les dades persistents s'emmagatzemen en Mongo (51) (52), una base de dades no relacional, *schemaless* (53) (54), i les dades de caràcter temporal s'emmagatzemen en Redis, un *key-value Store* què funciona en memòria. Redis també és utilitzat com a mecanisme de *cache*.

S'ha decidit implementar la API en Node.js (55) (56) i MongoDB per diversos motius:

- L'experiència prèvia personal en el desenvolupament *server side* ha estat sobretot amb PHP/MySQL, i s'ha considerat que utilitzar una tecnologia alternativa podria resultar motivant pel fet d'adquirir nous coneixements i experimentar amb d'altres tecnologies. El

desenvolupament en node és força diferent del desenvolupament amb PHP, i el fet d'utilitzar una base de dades, orientada a documents (57), en comptes d'una base de dades relacional tradicional implica un paradigma de desenvolupament diferent al habitual.

- *Full stack* (58): el desenvolupament en Node i Mongo implica utilitzar javascript en totes les capes de l'aplicació, des del *frontend* fins a la base de dades, passant pel *backend*. Aquest fet permet reutilitzar codi entre els diferents components.

- Velocitat: L'arquitectura triada ofereix grans avantatges de velocitat i rendiment respecte a d'altres arquitectures. Abans d'implementar l'aplicació definitiva es van implementar dos prototips de la API en node i PHP, i es van realitzar diversos *benchmarks*, i el rendiment mostrat per l'aplicació en node va ser molt superior (més de 10x) al de l'aplicació en PHP (veure capítol *5.3 Tests*).

Ja que l'arquitectura Restful (5) és *stateless* i no inclou el concepte de sessions, l'autenticació està implementada mitjançant *JSON web tokens* (59), que s'adjunten com una capçalera HTTP en cada petició per tal d'accedir a recursos protegits.

Els diversos tipus d'usuaris tenen diferents privilegis assignats, i cada un dels *endpoints* de la API pot requerir diferents privilegis per cada una de les operacions.

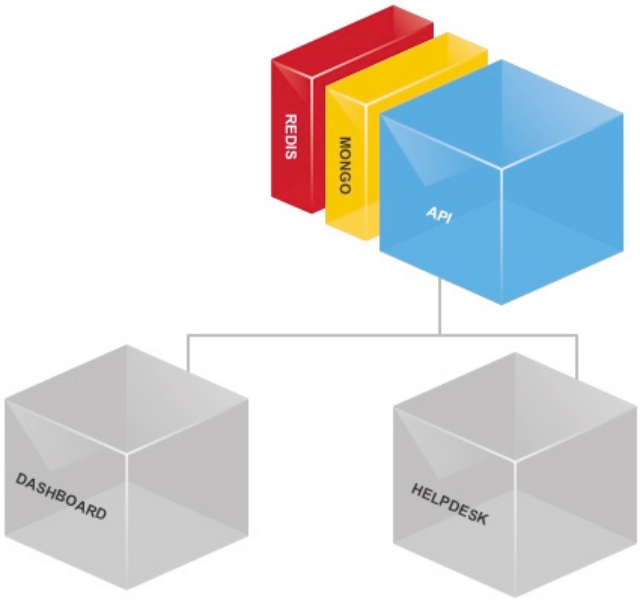


Figura 2 - Arquitectura bàsica del sistema

Tot i què la API és un servidor *HTTP,* i per si sola pot desenvolupar totes les tasques necessàries per a l'operatòria del sistema, incloent l'enviament de correus i les pujades d'arxius i el seu posterior accés, per operar l'aplicació en un entorn de producció real és recomanable no exposar directament la API a internet, i posar davant un servidor HTTP robust i àmpliament testejat, com per exemple NGINX (60) o Apache (61) (62) actuant com a *reverse proxy,* per motius de seguretat, rendiment i robustesa. Aquesta configuració permet tenir diverses instàncies de la API funcionant en paral·lel en un o diversos servidors i que NGINX actuant com a *load balancer.*

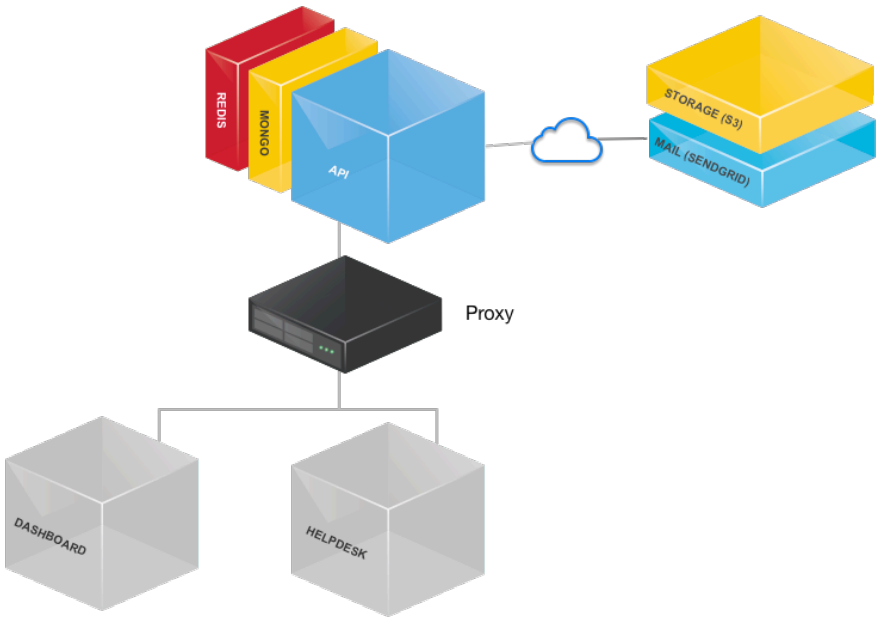


Figura 3 - Arquitectura recomanada en producció

La API està configurada per acceptar peticions de des de dominis tercers (*cross domain requests*). En el cas de configurar un *proxy* davant de la API pot ser necessari ajustar-ne la configuració per tal de permetre les peticions CORS (Cross-origin resource sharing) (63).

En producció tampoc es recomanable gestionar l'emmagatzemament d'arxius (i el seu posterior accés) des de la API, perquè aquest no és el seu paper. L'aplicació implementa la possibilitat d'utilitzar Amazon S3.

La llibreria utilitzada per a l'enviament de correus suporta diversos mecanismes d'enviament o *transports*. En producció es recomana utilitzar un servei especialitzat en l'enviament de correus transaccionals, com ara Sendgrid (15).

3.2.2 Aplicacions web

Com ja s'ha esmentat, les dues aplicacions web són completament estàtiques i no tenen cap requeriment especial, es poden allotjar en qualsevol servidor web. Per a la *demo* s'han creat dos *virtualhosts* independents, cada un amb el seu subdomini, però aquesta és només una de les configuracions possibles, i les aplicacions podrien funcionar sense problemes en un mateix *virtualhost*, en directoris diferents.

Cada una de les aplicacions està implementada com una *Single Page Application* (SPA). El fet de plantejar les aplicacions com SPA permet oferir una experència més fluïda als usuaris al realitzar la càrrega del codi HTML, CSS i Javascript a l'inici i carregant posteriorment només les dades pures a partir de la API. Aquesta arquitectura permet carregar les dades en segon pla i *cachejar-les* en local utilitzant tecnologies com *localStorage* o *sessionStorage*, i junt amb polítiques de cache agressives en el servidor, millora notablement la velocitat de càrrega percebuda per l'usuari.

Les aplicacions segueixen una arquitectura MVC, altament modularitzada, per facilitar-ne la escalabilitat i el manteniment. Tot i què l'aplicació dels usuaris i el *dashboard* són significativament diferents, comparteixen la mateixa arquitectura i la major part del codi.

Tots els components de l'aplicació són completament independents, i no existeixen referencies directes entre ells. La comunicació entre ells es realitza mitjançant *canals* o *event buses*, segons el patró *PubSub* (64), fet que facilita la modularització i escalabilitat de l'aplicació. Les aplicacions tenen una estructura jerarquitzada de mòduls, és a dir, estan compostes per una sèrie de mòduls principals, que al seu torn poden estar compostos per més mòduls. Cada un dels mòduls té una única finalitat, i si aquesta és molt complexa el mòdul s'ha dividit en submòduls més específics encarregats d'una tasca específica, i el mòdul *pare* simplement coordina els submòduls.

No s'inclou un diagrama de les classes perquè les aplicacions estan composades per centenars de classes, tot i què l'estructura en realitat no és molt complexa.

2. Arquitectura de la informació i diagrames de navegació

2.1 Disseny de la base de dades

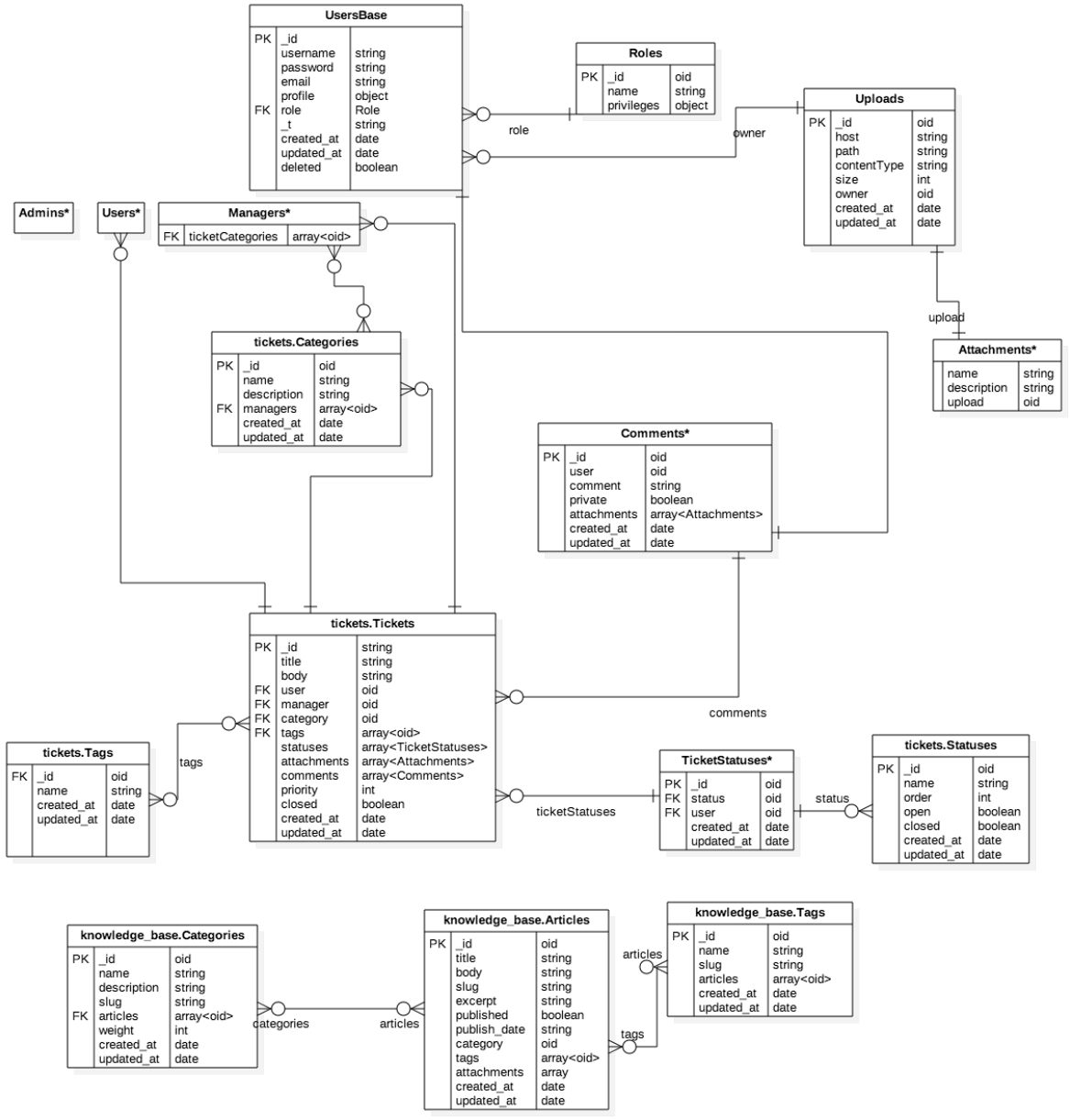


Figura 4 - Model de dades

En la figura anterior es mostra el model de dades del sistema, i en part es correspon a l'estructura dels documents emmagatzemats en Mongo. Cal mencionar que Mongo no és una base de dades relacional, sinó orientada a documents, per tant no existeix el concepte de taula, sinó de col·lecció de documents JSON. A més, Mongo és *schemaless*, per tant a nivell de la base de dades l'estructura dels documents d'una col·lecció determinada no està prefixada, tot i que l'ODM (Object Data Mapping) utilitzat, Mongoose (65), permet definir l'*schema* de les col·leccions.

Per tant és necessari aclarir alguns detalls del diagrama:

- Algunes de les entitats definides (les marcades amb un asterisc) no existeixen com a col·leccions de documents independents. Algunes d'aquestes entitats (*TicketStatuses*, *Comments*, *Attachments*) són documents niats, mentre que n'hi ha d'altres (*Users*, *Managers* i *Admins*) que són *subclasses*[3].
- En Mongo no existeix el concepte de claus foranes, tot i què s'han indicat en el diagrama per fer-lo més entenedor. La implementació de les relacions entre documents també és radicalment diferent que en les bases de dades relacionals, existint diverses formes d'implementar-les en funció de les circumstàncies. Per tant, hi ha algunes relacions que només s'han definit en un dels costats de la relació i en d'altres en els dos, en funció de diversos condicionants. Finalment, com ja s'ha mencionat, algunes de les entitats relacionades no existeixen com a documents independents, sinó que són documents niats. En general, el criteri seguit per decidir si per a un determinat tipus d'entitat es definiria una col·lecció de documents o s'utilitzarien documents niats és si la existència del document per si sol té sentit. En aquest cas com a norma general (en funció del cas poden existir condicionants addicionals) s'utilitza una col·lecció de documents independents. En cas contrari s'utilitzen estructures niades. Un clar exemple per il·lustrar aquest fet són els comentaris dels tiquets, la existència dels quals no té sentit per si sola, només si estan vinculats a un tiquet.
- Un altre punt que cal aclarir és el tipus de dades del camp *_id* de la col·lecció *tickets*. En la resta de col·leccions els *id* són del tipus *ObjectId*, una cadena hexadecimal de 24 caràcters, però en aquest cas és simplement un *String*. Els valors per al camp es generen amb una llibreria anomenada shortid (66), que genera identificadors únics no seqüencials molt més curts i fàcils de recordar o teclejar. L'elecció d'aquest tipus d'identificador en comptes dels que genera Mongo per defecte és per usabilitat, perquè els identificadors no només s'utilitzen internament en l'aplicació, sinó que en aquest cas també es mostren als usuaris com a identificador únic de cada incidència.

2.2 Comunicació amb el servidor

Les aplicacions web es comuniquen amb la API REST utilitzant AJAX. La API és *stateless*, per tant són les aplicacions web les encarregades de mantenir l'estat.

S'han implementat diversos mecanismes per minimitzar al màxim el volum de dades transmeses entre les aplicacions web i la API per tal d'optimitzar els recursos del servidor i millorar l'experiència dels usuaris. De la banda del servidor, les respostes de la API estan minimitzades i les respostes utilitzen compressió HTTP (gzip). A més, la API utilitza Redis com a mecanisme de cache (sobretot en qüestions referents a l'autenticació i autorització) per minimitzar l'accés a la base de dades i millorar el temps de resposta. Les peticions a la API que retornen múltiples entitats estan totes paginades per defecte per evitar transferir informació innecessària. Per la banda de les aplicacions clients, aquestes emmagatzemen les dades rebudes (són *cachejades*) i les conserven en memòria o en la *sessionStorage*, i només les recarreguen en determinades circumstàncies, i per actualitzar les entitats en la API sempre que es possible es realitzen actualitzacions parcials (PATCH) només amb els atributs que han canviat en comptes de actualitzacions completes (PUT).

[3] Aquest polimorfisme dels documents a partir d'un atribut discriminador és una funcionalitat aportada per l'ODM, ja que Mongo és *schemaless*.

Alguns dels recursos que exposa la API requereixen autenticació i determinats privilegis. L'autenticació està implementada mitjançant JSON web tokens (59). En el moment en el que l'usuari s'identifica en les aplicacions client, aquestes realitzen una petició a la API per obtenir el *token*. Si les credencials són correctes la API retorna un *token* a l'aplicació, que aquesta adjuntarà en les capçaleres HTTP de les següents peticions. Per evitar accessos continuats a la base de dades i costoses operacions criptogràfiques per descodificar els *tokens* en cada petició, un cop la API emet un *token* el desa temporalment durant el temps de vida d'aquest en Redis, un *key-value Store* en memòria, de forma que el procés d'autorització no requereix cap accés a la base de dades.

Degut al fet que els *tokens* són temporals, és a dir, que tenen un període de validesa determinat, les aplicacions client tenen la responsabilitat de controlar el temps restant de vida del *token* i renovar-lo automàticament abans que expiri per evitar que l'usuari es tingui que identificar de nou.

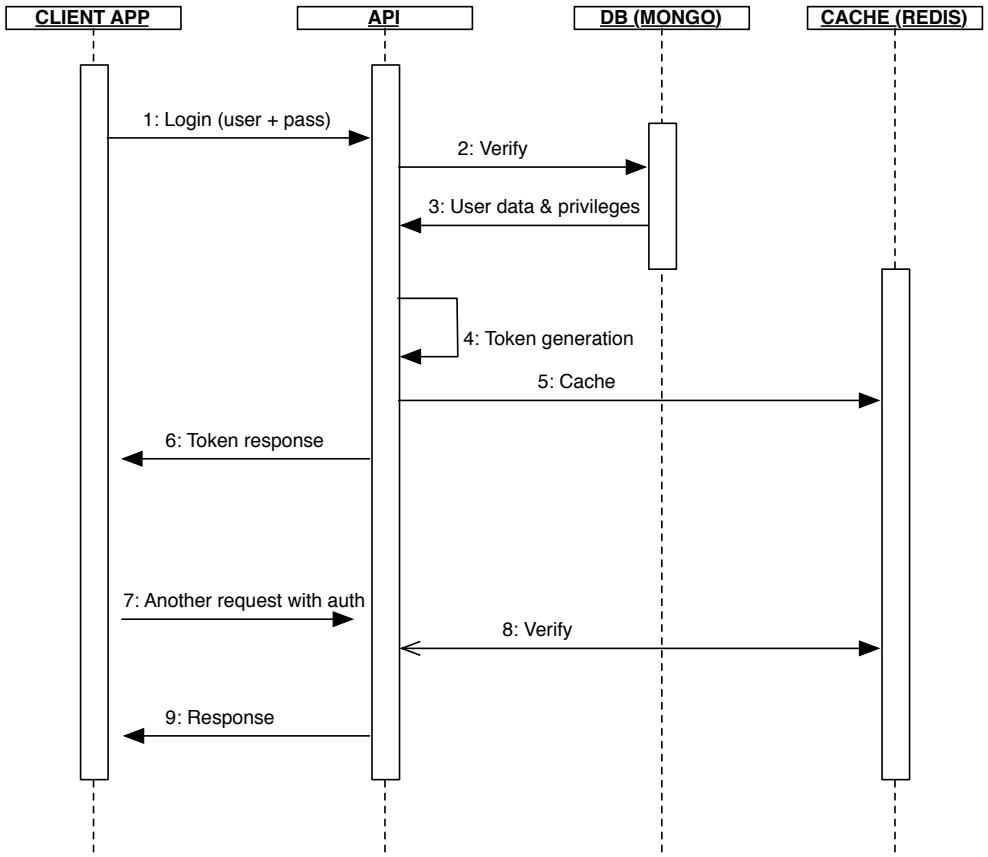


Figura 5 - Diagrama de seqüència procés autenticació

2.3 Estructura dels continguts

S'ha dissenyat la navegació de les aplicacions per tal que aquesta resulti el més simple possible, amb elements com menús i d'altres que permeten a l'usuari accedir a qualsevol de les funcionalitats en el menor número de passos entremitjos (màxim 3).

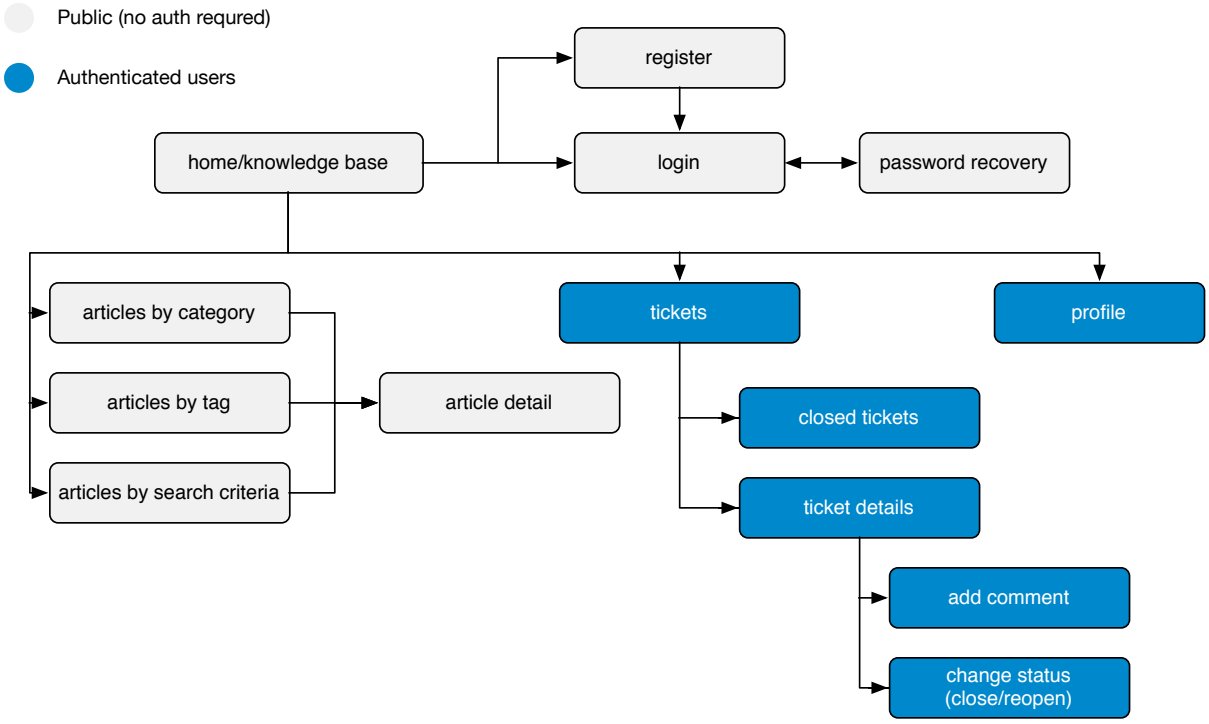


Figura 6 - Diagrama de navegació Helpdesk

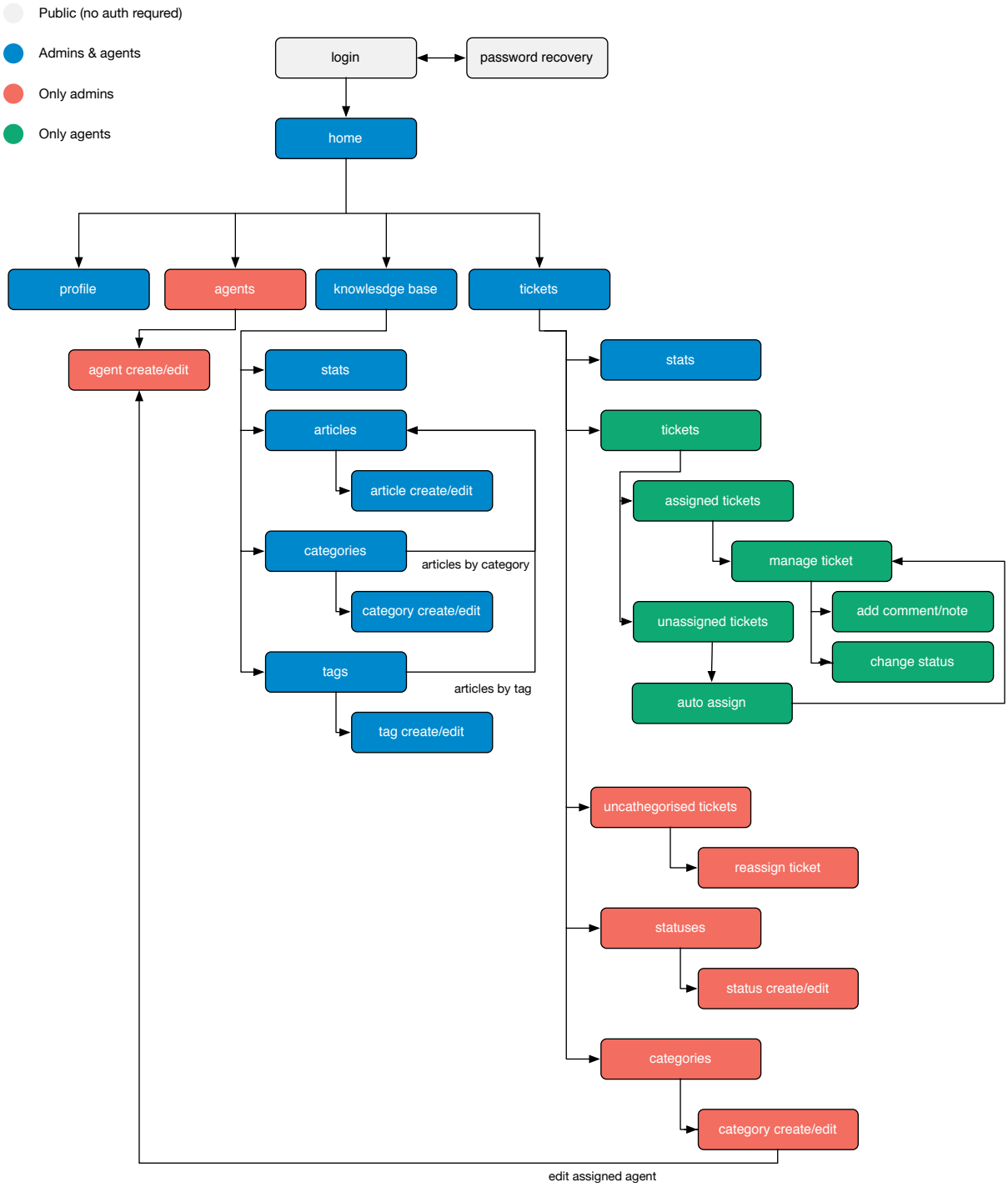


Figura 7 - Diagrama navegació Dashboard

3. Disseny gràfic i interfícies

3.1 Estils

Les dues aplicacions web que formen el sistema tenen una línia gràfica diferenciada, especialment pel que fa al *layout*, perquè estan orientades a públics i tasques i objectius diferents, tot i què són consistents en l'ús de paletes de colors, tipografies, logotips i elements de la UI.

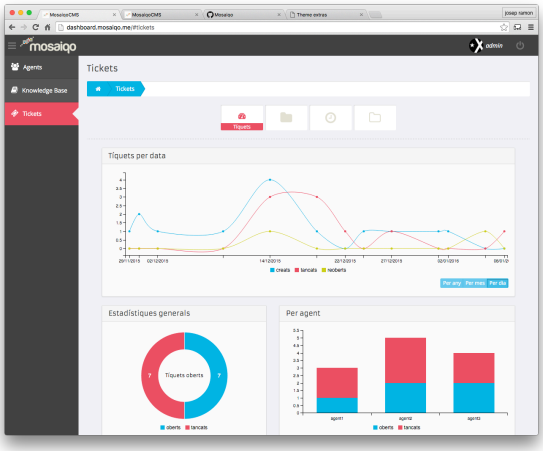


Figura 8 - Dashboard: stats

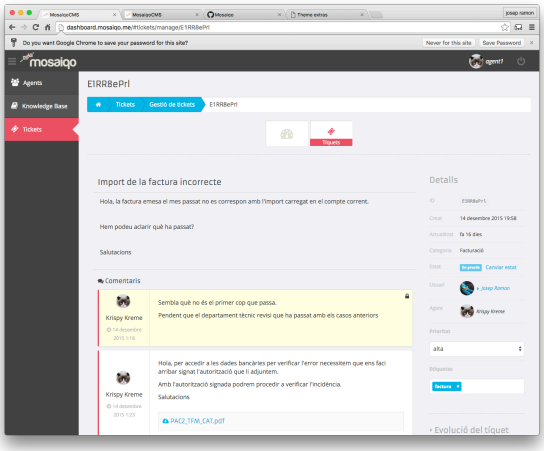


Figura 9 - Dashboard: Ticket detail

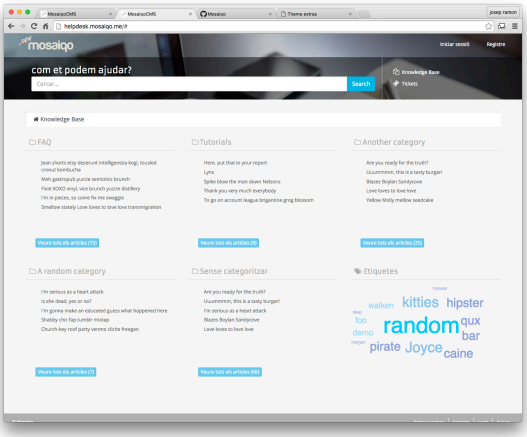


Figura 10 - Helpdesk: home

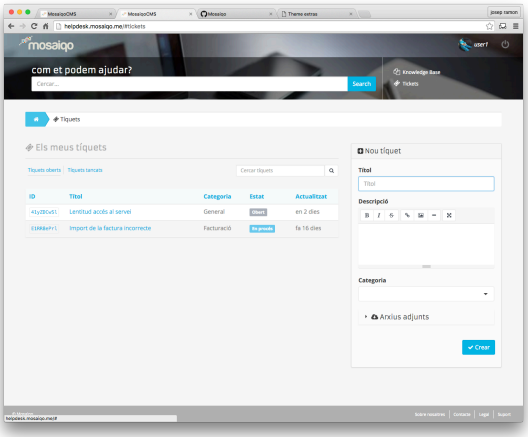


Figura 11 - Helpdesk: Tickets list

No s'ha considerat necessari desenvolupar un nou logotip per a l'aplicació, perquè com s'ha comentat amb anterioritat, el sistema d'una banda s'utilitzarà com a component de suport als usuaris d'un servei que s'està desenvolupant en paral·lel, i a part, s'alliberarà com a un projecte de codi lliure. En el moment que s'alliberi el sistema, no es farà a nivell personal, sinó sota la marca del servei que s'està desenvolupant, junt amb d'altres components i subsistemes que el composen.

S'ha considerat adequat utilitzar el mateix logotip o variacions d'aquest en tots els projectes de codi lliure derivats del servei per tal de crear imatge de marca.

En qualsevol cas el logotip de l'aplicació no és un element important, perquè els usuaris/empreses que decideixin utilitzar el sistema per a oferir el servei d'atenció als seus usuaris o clients substituiran el logotip per defecte per el seu propi.

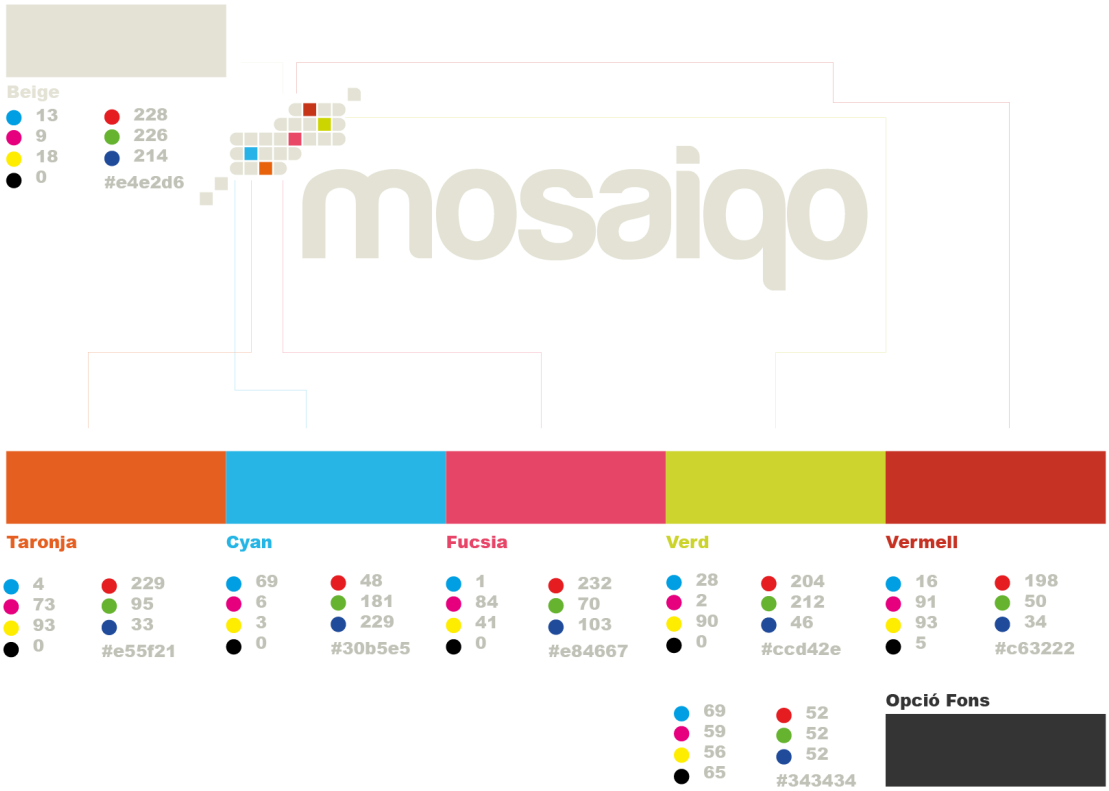


Figura 12 - Logotip aplicacions

El logotip ha estat dissenyat per Boudy de Geer, un altre dels components de l'equip amb qui estic desenvolupant el servei del CMS, i s'utilitza amb la seva autorització.

S'ha optat per una paleta de grisos com a base pel fet que les aplicacions estan pensades per ser explotades per tercers, integrant-les en els seus webs. Per tant, una paleta neutra s'ha considerat que era la opció més adequada. El *dashboard* utilitza algunes varietats més fosques per a alguns elements de la interfície, com ara la capçalera, el peu i la barra de navegació per diferenciar els elements de la UI de l'àrea de treball.
La resta de colors utilitzats venen determinats per el *tema* de la llibreria de CSS utilitzada (Twitter Bootstrap (67). El *tema* s'anomena Kaleidoscope (68), i ha estat desenvolupat per l'autor del projecte, al marge d'aquest. L'ús d'aquests colors és en funció de la seva càrrega semàntica, així:

- #30B5E5: utilitzat per als enllaços i per a indicar que una determinada acció és la primària.
- #7EB641: utilitzat per a indicar èxit, per exemple quan una acció s'ha completat adequadament.
- #FE7600: indica una advertència a l'usuari sobre algun fet que requereix atenció.
- #FF2417: indica perill, per exemple quan alguna acció pot tenir efectes negatius (per exemple l'eliminació d'algun element).

En general no s'han utilitzat imatges en les aplicacions amb l'objectiu d'obtenir una interfície el més plana i minimalista possible. Només se n'ha utilitzat una en la capçalera del *helpdesk* i una altra com a fons de les vistes d'autenticació del *dashboard.* En els dos casos són imatges força neutres, i els usuaris que decideixin integrar el sistema en els seus webs les poden canviar fàcilment per adaptar-ho a la seva imatge corporativa. Les dues imatges s'han obtingut en el lloc web Negativespace (69) sota una llicència CC0 (70).

Les tipografies utilitzades són com en el cas dels colors les que determina el *tema.* Aquestes són:

- **Armata**, creada per Viktoriya Grabowska, i distribuïda sota una llicència SIL Open Font License, 1.1 (71), és utilitzada per als títols.

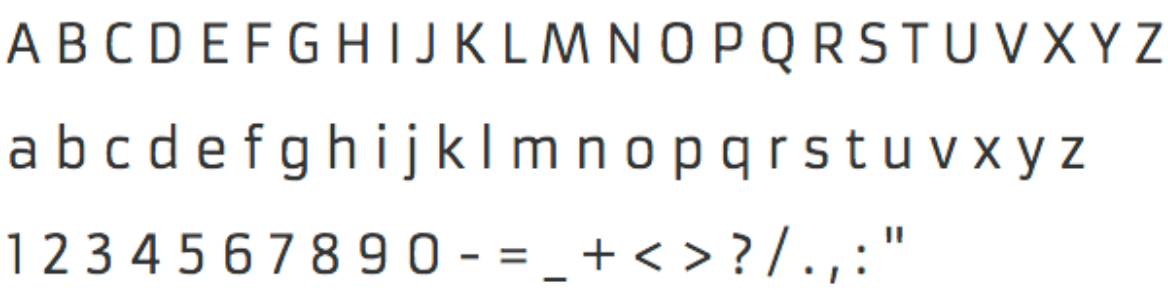


Figura 13 - Font Armata

- **Open Sans**, creada per Steve Matteson, i distribuïda ota una llicència Apache License, version 2.0 (72), utilitzada en la resta de casos.

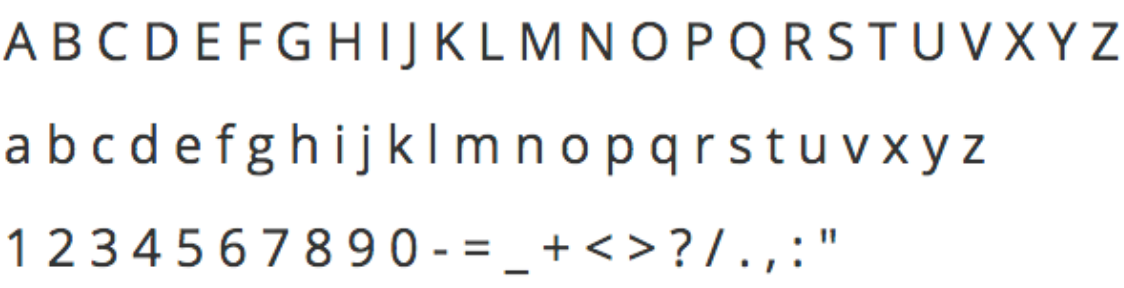


Figura 14 - Font Open Sans

Les aplicacions carreguen aquestes tipografies a través del servei Google Fonts (73).

La llibreria de icones utilitzada és Font Awesome (74), disponible sota una llicència SIL OFL 1.1.

3.2 Usabilitat/UX

La usabilitat és un dels factors fonamentals que s'han pres en compte durant el procés de disseny i desenvolupament del sistema, i s'han dissenyat tots els processos centrats en l'usuari. S'han tingut en compte els principis descrits en diverses obres, com *10 Usability Heuristics for User Interface Design* de Jakob Nielsen (75), *Principles for Usable Design* (76), o *Don't Make Me Think* de Steve Krug (77).

Les aplicacions són *responsive,* i per a les diverses resolucions de pantalla s'ha adaptat no només els *layouts*, sinó també el disseny i funcionament d'altres elements, com la mida de determinats botons, que s'ha incrementat per facilitar-ne l'ús en dispositius mòbils, o la presentació i comportament dels menús.

D'una forma resumida, els principis generals aplicats han estat:

- Consistència i estàndards: la consistència ha estat un dels factors principals que han determinat el disseny de cada una de les seccions i processos. Un dels motius pel qual s'ha decidit utilitzar Bootstrap com llibreria per a la UI i els *widgets* és el fet que és una llibreria àmpliament utilitzada, i per tant els usuaris estan familiaritzats amb els seus components.
- Rellevància i valor: en tot moment només es mostra la informació i les eines relacionades amb la tasca què s'està desenvolupant. S'ha optat per un disseny minimalista, no només a nivell visual, sinó també en quan a funcionalitats, implementant només les que aporten valor a l'usuari.
- Visibilitat del sistema: tot i què les aplicacions estan estructurades com a *Single Page Applications*, cada vista disposa d'una URL única. Addicionalment es mostra en tot moment (en la versió d'escriptori, en mòbils no) uns *breadcrumbs,* i en els menús sempre està destacada la secció en la que es troba l'usuari. Sempre que es produeix una comunicació amb el servidor, les aplicacions mostren un indicador de progrés per tal que els usuaris percebin que el procés s'està duent a terme, i que l'aplicació no s'ha *congelat.*
- Simplicitat, visibilitat i auto-evidència: El disseny i els processos s'han simplificat, eliminant qualsevol element superflu. L'accés a les opcions principals de l'aplicació són visibles en tot moment i les opcions secundàries estan només a un *click.* Totes les vistes i formularis s'han dissenyat de forma que el seu objectiu i funcionament sigui evident, sense necessitat de documentació addicional, tot i que en els processos que poden resultar més complexos o poden tenir un funcionament menys evident s'han afegit petits blocs informatius.
- Feedback: totes les accions que impliquen una interacció amb el servidor sempre donen lloc a una notificació per indicar l'èxit de la operació, o si ha fallat, els motius de l'error. Tots els formularis es validen per duplicat, en local i en el servidor. Si qualsevol de les validacions genera un error, el camp corresponent del formulari es destaca i és mostra el motiu de l'error.

- Estructura: Els continguts de les aplicacions estan clarament estructurats en diverses seccions (*knowledge base,* tiquets, perfil, etc.)
- Ajuda i documentació: Tot i que s'ha procurat dissenyar tots els processos i interaccions per tal que siguin completament evidents, en algunes circumstàncies s'ha optat per afegir textos d'ajuda addicionals per tal d'aclarir aspectes que puguin no resultar obvis.
- *Forgiveness* i *error prevention:* totes les accions potencialment destructives (per exemple eliminar algun element) visualment són diferents, en vermell per indicar que l'acció es potencialment perillosa. Un cop els usuaris premen aquests botons sempre es mostra un diàleg de confirmació abans d'executar la tasca.
- Reducció de la càrrega de memòria: s'ha procurat reduir al mínim els elements de la interfície i els formularis, eliminant tot el que no sigui imprescindible. Així, en els editors *wysiwyg* només es mostren les opcions necessàries en cada cas (per exemple, per a l'edició dels continguts dels articles les opcions disponibles són molt més nombroses que per a l'edició de la descripció de les categories). En formularis complexos amb molts camps, s'ha optat per agrupar els camps en grups, i ocultar tots els grups secundaris utilitzant *acordions.*

4. Llenguatges de programació i APIs utilitzades

4.1 API

La API s'ha desenvolupat en Javascript, sobre Nodejs i utilitzant Mongo i Redis com a base de dades. Aquesta elecció és deguda principalment al fet que la meva experiència anterior en el desenvolupament *backend* era bàsicament amb PHP/MySQL, i es va considerar que utilitzar un *stack* alternatiu podria resultar motivant i enriquidor. Tot i així, tenint en compte que la durada del projecte és limitada es va optar per Javascript en comptes d'altres altres alternatives com Python o Ruby per l'experiència prèvia amb el llenguatge en el desenvolupament *frontend*, i perquè s'havia previst que el sistema inclogués un sistema de xat en temps real, basat en *websockets*, i la quantitat de llibreries madures i documentació per implementar aquesta funcionalitat era més extensa. A més, el fet d'utilitzar Javascript en el servidor permetia reutilitzar components el les aplicacions client.

L'ús de bases de dades no relacionals ha estat també ha estat motivat principalment per l'experiència anterior amb bases de dades d'altres tipus, i per tant el desig d'experimentar amb noves tecnologies. Es van explorar altres alternatives, com *Cassandra* (78) (79) o*, CouchDB* (80) (81)*,* però finalment es va optar per Mongo i Redis perquè hi havia més documentació i llibreries per a Node disponibles.

L'aplicació està implementada utilitzant una *framwework* anomenada Express (50). Hi ha un gran nombre de *frameworks* alternatives, com Meteor (82), Hapi (83) o Sails.js (84), però es va optar per Express perquè aquesta està inspirada en Sinatra (85), una *framework* per a Ruby amb la que havia experimentat amb anterioritat, i que destaca pel seu minimalisme i elegància. A més, Express és provablement la *frameework* per a Node més popular, i per tant hi ha disponible una gran quantitat de documentació i *plugins.* Algún dels plugins utilitzats són *express-jwt* (86) o *express-paginate* (87).

Per a l'accés i manipulació de les dades emmagatzemades en Mongo s'ha utilitzat Mongoose (65), l'ODM (Object Document Mapper) estàndard *de facto* en Node, junt amb diversos plugins com *mongoose-deep-populate* (88)*, mongoose-deleted* (89)*, mongoose-paginate* (90) o *mongoose-time* (91) entre d'altres.

D'altres llibreries rellevants utilitzades en l'aplicació són *multer* (92) per a gestionar les pujades d'arxius i *nodemailer* (93) per a l'enviament de correus.

Només s'han mencionat les dependències més rellevants. El llistat complert està en l'arxiu *package.json* del projecte.

Les APIs i serveis de tercers utilitzats per la API són:

- Gravatar (94): en el moment en que els usuaris es registren només han d'introduir el nom d'usuari, la contrasenya i la direcció de correu electrònic. Per tal que la seva imatge de perfil

no estigui buida fins que pugi una imatge, s'utilitza per defecte la imatge que pugui tenir associada la direcció del correu al servei Gravatar. Si el email no està donat d'alta a Gravatar, el servei retorna una imatge per defecte.

- Amazon S3 (opcional): la API permet utilitzar Amazon S3 com a sistema d'emmagatzemament per als arxius que pugen ells usuaris. Si l'aplicació està configurada per a utilitzar aquest servei, un cop l'usuari ha pujat un arxiu, aquest es mou a Amazon S3 utilitzant la seva API. La implementació és completament transparent, perquè *multer,* la llibreria encarregada de la pujada dels arxius, disposa d'un *adapter* per utilitzar Amazon S3 en lloc del *filesystem.*
- Sendgrid (15) (opcional): *nodemailer*, la llibreria utilitzada per a l'enviament de correus disposa de diversos *transports*, o mecanismes per a enviar els correus. En la *demo*, s'ha utilitzat el servei proporcionat per Sendgrid, un servei d'enviament de correus transaccionals per a enviar les notificacions. S'han realitzat proves amb diverses configuracions i proveïdors de correu i aquest servei ha estat el més fàcil de configurar i el que ha ofert un millor rendiment. Altres serveis provats, com Gmail, han resultat força problemàtics en qüestions d'autenticació.

4.2 Aplicacions client

Les aplicacions estan programades en Coffeescript (95) (96), un llenguatge que es transpila a javascript, i què aporta algunes avantatges sobre aquest, com la brevetat, la llegibilitat. Un dels motius que han condicionat l'elecció d'aquest llenguatge és que les aplicacions estan desenvolupades sobre una llibreria anomenada *Mosaiqo FrontendAppLib* (97) desenvolupada en aquest llenguatge per l'autor del projecte, que aporta un conjunt de components i patrons per al desenvolupament d'aplicacions web basades en les *frameworks* Backbone (98) i Marionnette (99).

Tot i que existeixen un gran nombre de *frameworks* alternatives com Ember (100), Angular (101) o React (102) s'ha optat per la solució basada en Backbone, Marionette i FrontendAppLib per diversos motius. D'una banda el fet de tenir experiència prèvia amb aquestes *frameworks* agilitza el procés de desenvolupament. Aquest ha estat un factor important, tenint en compte que per a la API s'han escollit tecnologies amb les que no es tenia experiència prèvia. D'una altra banda, la decisió també ha estat condicionada pel fet de voler aprofitar la feina prèvia en el desenvolupament de Mosaiqo FrontendAppLib, i finalment perquè un dels objectius del projecte és poder integrar el sistema en un altre projecte en el que estic treballant, per tant, per facilitar la integració les aplicacions tenen una arquitectura i estan basades en unes llibreries comunes.

Com que Backbone és una *framework* molt minimalista en comparació a altres *frameworks* com Ember, s'ha utilitzat un elevat nombre de llibreries addicionals o *plugins,* com l'anteriorment mencionada Marionette, que aporta classes base per a implementar diferents tipus de vistes, Backbone-associations (103) per a les relacions entre models, Backbone.ComputedFields (104) per a definir atributs virtuals així com *getters* i *setters*, Backbone.Radio (105) per a la comunicació entre

mòduls mitjançant el patró PubSub, Backbone.Syphon (106) per a la serialització/deserialització de formularis, i d'altres.

D'altres llibreries rellevants utilitzades són:

- jQuery (107): principalment utilitzada per a la manipulació del DOM, i també per a la comunicació amb la API mitajnçant XMLHttpRequest (108).
- Handlebars.js (109): *logicless templating language* utilitzat per definir l'estructura de les vistes, que permet separar l'estructura i presentació d'aquestes del comportament.
- I18next (110): utilitzada com a mecanisme d'internacionalització de les aplicacions, permet separar tots els continguts textuals del codi, movent-los a arxius *locales* independents, i gestiona la detecció de l'idioma i la càrrega de l'arxiu amb les traduccions en la llengua corresponent.
- Moment.js (111): llibreria per a la manipulació de dates, permet *parsejar*, manipular i mostrar-les dates en diferents formats, amb suport multi idioma.
- C3.js (112): C3 és un *wrapper* per a la llibreria de generació de gràfiques D3 (113), i què facilita la creació de gràfiques interactives amb aquesta llibreria.

La capa de presentació s'ha implementat utilitzant Stylus (114) (115), un metallenguatge que es transpila a CSS, que ofereix variables, funcions i d'altres avantatges, com una sintaxi més compacta o herència i d'altres mecanismes per a reutilitzar codi.
També s'ha utilitzat Autoprefixer (116), un potprocessador de CSS que automàticament afegeix els *vendor prefixes* (117) necessaris per garantir la compatibilitat amb els diferents navegadors.

D'altres llibreries rellevants utilitzades en el projecte relacionades amb la capa de presentació són Kaleidoscope (68), un tema per a Twitter Bootstrap (67), de l'autor del projecte, i la llibreria d'icones Font Awesome (74).

El llistat de dependències i llibreries utilitzades per les aplicacions és força més extens. El llistat complert està en l'arxiu *package.json* del projecte.

Totes les llibreries, frameworks i d'altres dependències tenen llicències lliberals, tipus MIT, Apache i similars, i s'han evitat les llicències tipus *copyleft,* perquè aquestes haurien condicionat la llicència dels components del projecte.

4.3 Altres eines i serveis emprats

Una llibreria que s'ha utilitzat extensivament tant en la API com en les aplicacions client és underscore.js (118), una llibreria que proporciona un nombre elevat de *helpers* de tot tipus, com ara iteradors, manipuladors d'objectes, *function binding,* testeig de tipus de dades i altres.

Per als tests automatitzats, tant de la API com les aplicacions client, s'han utilitzat diverses llibreries. Les més importants són la *testing framework* Mocha (119), la *BDD/TDD assertion library* Chai (120) i Sinon.js (121), una llibreria que proporciona *test spies, stubs* i *mocks* per als tests.

Tot i que els tests s'han executat principalment en local, en la màquina utilitzada per al desenvolupament, en el cas de la API també s'han executat en Travis CI (14), un sistema d'integració continua, en determinats moments.
El repositori de Github es va configurar de forma que cada cop que es realitzi un un *push* al repositori del projecte, Github notifiqui a Travis, mitjançant un *webhook,* que hi han hagut canvis i què executi tots els tests definits en el projecte.

En els tres projectes s'han automatitzat totes les tasques possibles (*build,* execució de tests, monitorització d'arxius, arrancada de serveis, etc.) utilitzant un *task runner* anomenat Grunt (10). De les tasques definides en els tres projectes caldria destacar l'anomenada `dev`. Aquesta tasca, que ha resultat vital en el procés de desenvolupament i ha permès estalviar molt de temps realitza les següents tasques:

- Aplicacions client:
    - Genera els arxius HTML a partir de *templates.*
    - *Transplila* els arxius Stylus a CSS.
    - Realitza el *build* dels scripts (*linting* del codi, *transpilació* coffeescript a javascript, resolució de dependències, concatenació, minimització i generació de *source maps*)
    - Optimitza automàticament les imatges del projecte.
    - Monitoritza els arxius que componen el projecte, de forma que quan algun canvia automàticament s'executen les tasques necessàries.
    - Arrenca un servidor web, que a més injecta automàticament un *script* a les pàgines de forma que quan qualsevol arxiu font canvia els navegadors el recarreguen automàticament, i que permet sincronitzar la navegació i els esdeveniments entre diversos navegadors i dispositius diferents.
- API:
    - Arrenca els serveis Mongo i Redis
    - Configura determinades variables d'entorn
    - Arrenca l'aplicació
    - Monitoritza els arxius que componen l'aplicació, i cada cop que es detecta un canvi automàticament valida la sintaxi i l'estil dels arxius, reinicia l'aplicació i executa els tests.

Capítol 4: Implementació

1. Requisits d'instal·lació

Les aplicacions dels usuaris i els administradors no tenen cap requisit especial d'instal·lació per al seu ús. Només requereixen un servidor web, qualsevol és vàlid. Els projectes n'inclouen un basat en node què és adequat per al desenvolupament i proves, però per a posar les aplicacions en producció és recomanable utilitzar un servidor web *normal,* com ara Nginx o Apache.

En principi no són necessaris cap tipus de coneixements o formació específica.

Per tal d'editar el codi per modificar-ne el comportament o l'aspecte visual és necessari instal·lar node prèviament en el sistema (durant el desenvolupament s'ha utilitzat la versió 4.2.1).

Són necessaris uns coneixements relativament avançats de javascript, Coffeescript i certa familiaritat amb la framework Backbone, així com comoditat amb l'ús de la línia de comandes del sistema (és recomanable l'ús d'un sistema tipus Unix, com ara Linux o OSX, tot i què en principi també es pot utilitzar Windows) per tal de *compilar* les aplicacions.

La API és el component més complex del sistema i té més requisits d'instal·lació:

Node: durant el desenvolupament del projecte s'ha utilitzat la versió 4.2.1. Actualment l'última versió estable és la 5.3.0, però no alguna de les dependències de l'aplicació no és compatible amb la branca 5.x, per tant és recomanable utilitzar-ne una de la branca 4.2.x, què en aquest moment està etiquetada com LTS (long term support) (122).
Mongo i Redis: la API requereix Mongo i Redis per emmagatzemar les dades de forma persistent i com a mecanisme de *cache.* Aquests serveis poden estar instal·lats en el mateix sistema en el què s'executa la API o en un altre (també és pot utilitzar un proveïdor extern, com ara Mongolab (51) o Redislabs (123)).
Servidor web: tot i què l'aplicació no requereix cap servidor web perquè n'implementa un, no és aconsellable exposar directament aquest servidor en un entorn de producció, i es recomana utilitzar un servidor web davant de la aplicació per motius de seguretat i rendiment, com Nginx, actuant com a proxy invers.
L'API envia notificacions per correu electrònic als usuaris davant determinats esdeveniments. Tot i què es pot utilitzar un compte de correu electrònic de qualsevol proveïdor, en producció és convenient utilitzar un servei extern especialitzat en l'enviament de correus transaccionals. L'aplicació ofereix suport per a un servei anomenat Sendgrid (15), tot i què la llibreria per enviar els correus suporta altres proveïdors (en aquest cas seria necessari instal·lar un adaptador addicional).
Per a les pujades d'arxius és necessari disposar d'un directori amb permisos d'escriptura en el servidor on s'executi la API. Per motius d'escalabilitat, en un entorn de producció pot resultar

convenient utilitzar un servei d'emmagatzemament massiu com Amazon S3, tot i què és completament opcional.

Són necessaris uns mínims coneixements d'administració de servidors per a posar la API en producció de forma òptima a nivell de rendiment i seguretat, pel fet què hi ha diversos components a instal·lar, i existeixen diverses formes de configurar l'aplicació en node darrera d'un servidor web.

2. Instruccions d'instal·lació

2.1 API

Per instal·lar la API és necessari instal·lar i tenir funcionant els prerequisits de la aplicació. Aquests requisits són Node, concretament la versió 4.2.x, actualment etiquetada com LTS, i el seu gestor de paquets NPM, Mongo i Redis.

El procediment d'instal·lació varia d'un sistema operatiu a un altre. En tots els casos les aplicacions estan disponibles per a ser descarregades com a binari i en forma de codi font, i també estan disponibles en els repositoris dels gestors de paquets com APT, Yum o Brew.

En el cas de Node, els binaris es poden descarregar a través de l'apartat de descàrregues del lloc del projecte (124), on també existeix una pàgina amb instruccions per a la instal·lació amb gestors de paquets (125).

Els binaris de Mongo es poden descarregar de la pàgina de descarregues del projecte (126), que també disposa d'un apartat amb instruccions detallades per a la seva instal·lació a partir del codi font o mitjançant gestors de paquets (127).

Redis només ofereix descarregues en forma de codi font a través de la seva pàgina oficial de descarregues (128), tot i què també està disponible a través de la majoria de gestors de paquets.

Un cop instal·lades aquestes dependències, és necessari instal·lar les llibreries que requereix l'aplicació executant `npm install` en el directori arrel del projecte. Totes les dependències s'instal·len en el directori *node_modules* del projecte. Una de les dependències, Grunt, s'utilitza per a executar diferents tasques relacionades amb el projecte i pot resultar més còmode instal·lar-lo globalment en el sistema per no haver de introduir la ruta complerta a l'executable cada vegada que sigui necessari executar una tasca. Per a instal·lar Grunt globalment és necessari executar:

```
$ npm install grunt -g
```

A continuació s'ha de definir els paràmetres de configuració de la API. En l'arrel del projecte hi ha un arxiu anomenat *env.json* amb la configuració per defecte de l'aplicació. Aquest arxiu no s'hauria d'editar, sinó que s'ha de crear un nou arxiu en el mateix directori anomenat *env.json*, i redefinir només els paràmetres que es desitgi sobreescriure.

Aquests paràmetres són:

- `serverPort`: aquesta directiva determina el port en el què estarà disponible la API
- `jwtSecret`: clau criptogràfica utilitzada per a signar els *tokens* utilitzats en l'autenticació. Pot ser una cadena aleatòria.
- `cryptKey`: clau criptogràfica utilitzada per a encriptar i desencriptar determinades dades, per exemple algunes dades emmagatzemades temporalment a Redis.
- `redis`: paràmetres de connexió a Redis. Per defecte la API es connecta intenta connectar-se a una instància de Redis executant-se en la mateixa màquina que la API, en el port per defecte, sense utilitzar autenticació, i utilitzant la base de dades per defecte. Tots els paràmetres es poden sobreescriure. Per exemple:

```
{
  “redis”: {
    "host": "localhost",
    "port": 6379,
    “db”: 4,
    “password”: “w9f376lefe445H/%E$&BTERTGgkfduy475t”
  }
}
```

- `mongo`: Aquest node defineix els paràmetres de connexió a Mongo, com ara el *host*, el port i els paràmetres d'autenticació. En l'arxiu de configuració hi ha definits dos entorns, el per defecte i un per als tests, de forma que durant els tests les dades reals no siguin alterades.
- `mail`: l'enviament de correus en la API utilitza la llibreria nodemailer (93). Aquesta llibreria implementa diferents mecanismes o estratègies per enviar els correus que es poden definir en l'arxiu de configuració. Per a l'enviament utilitzant una connexió a un servidor de SMT, la configuració hauria de ser similar a la següent:

```
{
  "mail": {
    "host": "localhost",
    "port": 25,
    "auth": {
      "user": "username",
      "pass": "password"
    },
    "sender": "Some name <sender@gmail.com"
  }
}
```

La directiva `sender` determina l'origen dels correus enviats per la API.

En les proves realitzades, la configuració d'alguns proveïdors de correu com Gmail o iCloud ha resultat força problemàtica, i la solució més senzilla i que ha donat millors resultats ha estat utilitzar un servei especialitzat per a l'enviament de correus transaccionals anomenat Sendgrid. En aquest cas, la configuració hauria de ser similar a la següent:

```
{
  "mail": {
    "service": "sendgrid",
    "auth": {
      "api_key": "your_api_key"
    },
    "sender": "Some name <sender@gmail.com"
}
```

- `sites`: aquesta directiva permet definir les *url* de les aplicacions dels administradors i els usuaris. Aquestes *url* s'utilitzen en els enllaços què contenen alguns dels correus que envia l'aplicació davant determinats esdeveniments (per exemple quan un usuari es registra).
- `uploads`: conté la configuració per a la pujada d'arxius. Les pujades es poden allotjar en el mateix servidor de la API o a Amazon S3 (recomanable en producció). Per allotjar els arxius en el servidor de la API la configuració hauria de ser similar a la següent:

```
{
  "uploads": {
     "mode":  "local",
     "dest":  "/path/to/the/upload/dir",
     "host":  "localhost:9999",
     "path":  "uploads",
     "host":  null
   }
}
```

on:

- `dest`: directori on s'emmagatzemaran els arxius.

- `host`: host a partir del qual es serviran els arxius. En producció no es recomanable servir els arxius directament des de la API. És millor servir-los directament des de Nginx o Apache. Durant el desenvolupament, la API pot servir els arxius establint com a `null` la directiva. En aquest cas, els arxius estaran disponibles en la ruta definida en la directiva `path`.

- `path`: ruta a partir d'on es serviran els arxius.

Amb la configuració anterior, un arxiu pujat al servidor *whatever.jpg* estarà disponible a *http://localhost:9999/uploads/whatever.jpg.*

Per a allotjar els arxius a Amazon S3, la configuració hauria de similar a la següent:

```
{
  "uploads": {
    "mode":         "s3",
    "path":         "uploads",
    "host":         "your-bucket.s3.amazonaws.com",
    "bucket":       "the-bucket-name",
    "region":       "eu-central-1"
  },
  "s3": {
    "accessKey":    "your-S3-accessKey",
    "accessKeyId": "your-S3-accessKeyId"
  }
}
```

on:

- `uploads.path`: 'directori' per als arxius.
- `uploads.host`: host utilitzat per recuperar els arxius. El valor ha de ser del tipus `your-bucket.3.amazonaws.com` o un domini personalitzat com ara `files.yourdomain.com` (en aquest cas és necessari configurar un registre DNS que apunti a AWS) (129).
- `uploads.bucket`: S3 bucket en el que s'emmagatzemaran les pujades.
- `s3.accessKey`: S3 access key
- `s3.accessKeyId`: S3 access key id
- `uploads.region`: Regió AWS on estan allotjats els *buckets.*

Les directives `bucket`, `region` i `host` estan definides dins del node `uploads` en comptes del node s3 per tal què es puguin utilitzar diferents *buckets* i configuracions per a diferents tasques.

Un cop configurada l'aplicació és necessari carregar les dades inicials i crear un usuari de tipus administrador.

Per a carregar les dades és necessari executar en el directori on està l'aplicació la següent comanda:

```
$  grunt setup
```

Per a crear l'administrador cal executar:

```
$  node ./util/createUser.js
```

i seguir les instruccions que l'script va proporcionant.

Amb l'aplicació configurada i les dades carregades es pot arrencar l'aplicació executant:

```
$ npm start
```

La API també pot ser arrancada en mode *desenvolupament.* Quan funciona en aquestes condicions, els arxius de l'aplicació estan monitoritzats, i quan es produeix qualsevol canvi automàticament s'executen els tests unitaris i es reinicia l'aplicació. Per iniciar la API en mode desenvolupament cal executar:

```
$ grunt dev
```

2.2 Aplicacions web (usuaris i administradors)

Les aplicacions web dels usuaris i els administradors, un com compilades, no requereixen cap tipus d'instal·lació especial, només cal publicar-les a un directori accessible d'un servidor web.

La *url* de la API i l'idioma de les aplicacions es pot personalitzar editant els arxius *index.html* de cada aplicació.
Per canviar l'idioma només cal editar l'atribut `lang` de l'etiqueta `html`. Els valors acceptats són `ca` per al català, `es` per al castellà i `en` per l'anglés. Si el valor és un altre o no s'especifica, les aplicacions per defecte es mostraran en anglès.
Per modificar la url de la API cal editar la variable de javascript `MOSAIQO_API_ROOT_URL` de l'arxiu. La url ha de ser absoluta i ha d'incloure el protocol, per exemple `http://localhost/api`.

Qualsevol altre canvi o parametrització de les aplicacions requereix recompilar-les.

Per compilar les aplicacions primer és necessari instal·lar les dependències. Amb node i npm prèviament instal·lats, cal executar:

```
$ npm install
```

Alguna de les dependències tenen prerequisits, llibreries i executables que han d'estar instal·lats al sistema. Algunes d'aquestes dependències venen preinstal·lades en determinats sistemes operatius però en d'altres és necessària una instal·lació. Per exemple, en Ubuntu 14.04.3 LTS les següents llibreries s'han hagut d'instal·lar:

- imagemagick
- libpng-dev
- optipng

Un cop instal·lades, cal crear un arxiu de configuració a l'arrel del projecte anomenat *env.json* amb la url de la API:

```
{
  "api": {
    "rootURL": "http://192.168.1.10:8000/api"
  }
}
```

En aquest arxiu també és pot definir l'idioma de l'aplicació, per exemple:

```
{
  "lang": "ca"
}
```

Els enllaços que apareixen en el peu de les aplicacions es poden personalitzar afegint a l'arxiu un bloc com el següent:

```
{
  "footer": [
    { "text": "About us", "url": "http://whatever.com/about" },
    { "text": "Legal", "url": "http://whatever.com/legal" },
    { "text": "contact", "url": "contact.html" }
  ]
}
```

Un cop definida la configuració, per compilar les aplicacions només cal executar `grunt` en l'arrel del projecte.

Hi ha definida una tasca per al desenvolupament, que es pot invocar executant `grunt dev`, que monitoritza els arxius de l'aplicació i la recompila quan es produeix qualsevol canvi. Aquesta tasca també arrenca un servidor HTTP que injecta en les pàgines un script que actualitza automàticament el navegador quan es produeix un canvi en el codi. Els arxius generats per aquesta tasca no són adequats per a producció, perquè els arxius resultants no estan *minimitzats,* i incorporen *source maps*, directives i codi addicional per facilitar la depuració.

Capítol 5: Demostració

1. Instruccions d'ús

S'ha configurat un servidor de proves amb una *demo* del sistema per tal de facilitar-ne l'avaluació.

Les *url* són:

- Aplicació dels usuaris: http://helpdesk.mosaiqo.me
- Aplicació dels administradors i agents: http://dashboard.mosaiqo.me/

L'accés a la aplicació dels administradors és restringida als usuaris registrats. S'ha creat un administrador i diversos gestors de prova. Les dades d'accés són:

- user: `admin`, password: `admin1234`
- user: `agent1`, password: `agent1234`
- user: `agent2`, password: `agent1234`
- user: `agent3`, password: `agent1234`

La aplicació dels usuaris té alguns apartats d'accés restringit, com la creació de tiquets. Hi ha donats d'alta uns quants usuaris amb diversos tiquets cada un:

- user: `user1`, password: `user1234`
- user: `user2`, password: `user1234`
- user: `user3`, password: `user1234`
- user: `user4`, password: `user1234`
- user: `user5`, password: `user1234`
- user: `user6`, password: `user1234`
- user: `user7`, password: `user1234`
- user: `user8`, password: `user1234`

Les aplicacions accedeixen a la API en la url https://mosaiqo.me/api

També s'hi pot accedir utilitzant cURL (130) (131) o qualsevol client REST (com POSTMAN (13)). La documentació de la major part de les peticions acceptades per la API està disponible a http://josepramon.github.io/tfm-api/apidoc

En el directori `doc` del codi font del la API hi ha un directori anomenat `api-postman-conf`, amb arxius de configuració per aquest client REST amb totes les peticions a la API preconfigurades per testejar-ne el funcionament.

El funcionament de les aplicacions és prou senzill i intuïtiu, i en alguns casos realment no fa falta cap mena d'instruccions, però a continuació es descriuen els passos de les activitats principals què s'hi poden realitzar.

1.1 Aplicació dels usuaris

1.1.1 Knowledge base:

Al accedir a l'aplicació es mostra un llistat de les categories d'articles d'ajuda, mostrant per a cada una els últims articles publicats (i/o els articles als que els administradors els hagin assignat més pes), un núvol d'etiquetes dels articles i un cercador.

Un cop s'accedeix al llistat d'articles d'una categoria (o al llistat de resultats d'una cerca) articles apareixen paginats. No hi ha cap mecanisme de paginació explícit, a mesura que l'usuari va fent *scroll* es van carregant més articles automàticament.

Per tornar enrere és poden utilitzar els botons estàndard de navegació (endavant/enrere) del navegador, punxar en qualsevol part els *breadcrumbs* o algun dels enllaços del menú principal.

1.1.2 Registre/dades d'usuari:

Per a accedir a l'apartat dels tiquets, és necessari identificar-se en l'aplicació. En la part superior d'aquesta hi ha els enllaços per iniciar sessió i crear un compte nou.
Durant el registre només es sol·licita a l'usuari les dades imprescindibles (nom d'usuari, contrasenya i direcció de correu electrònic). Un cop completat el formulari l'aplicació enviarà a l'usuari un correu amb un enllaç per activar el compte.

Un cop identificat, l'usuari pot editar les seves dades accedint al seu perfil punxant al seu nom d'usuari en la part superior de l'aplicació. Per defecte, la API assigna a l'usuari com a imatge de perfil la imatge vinculada al correu electrònic en el servei Gravatar (94) (132), i si la direcció de correu no està vinculada al servei una per defecte. A través de l'aplicació, l'usuari pot pujar una nova imatge per utilitzar en comptes de la provinent de Gravatar. Per pujar una imatge nova cal punxar en el camp corresponent o arrossegar-hi a sobre l'arxiu.

1.1.3 Tickets:

Quan l'usuari accedeix a l'apartat dels tiquets se li mostra inicialment un llistat amb els tiquets en procés de resolució i un formulari per crear-ne un de nou. El llistat apareix paginat en cas d'haver molts tiquets, i l'usuari pot ordenar el llistat *clickant* en les capçaleres de la taula. En la part superior

del llistat hi ha un menú per passar al llistat de tiquets tancats (resolts).

També pot cercar un tiquet concret a partir del seu codi identificador o pel text què conté aquest.

Per crear un nou tiquets l'usuari ha d'emplenar com a mínim els camps *títol* i *descripció*. El desplegable de la categoria també és obligatori però només es mostra si en el sistema hi ha donades d'alta més d'una categoria. L'usuari pot adjuntar diversos arxius punxant en el camp corresponent o arrossegant-los a sobre.

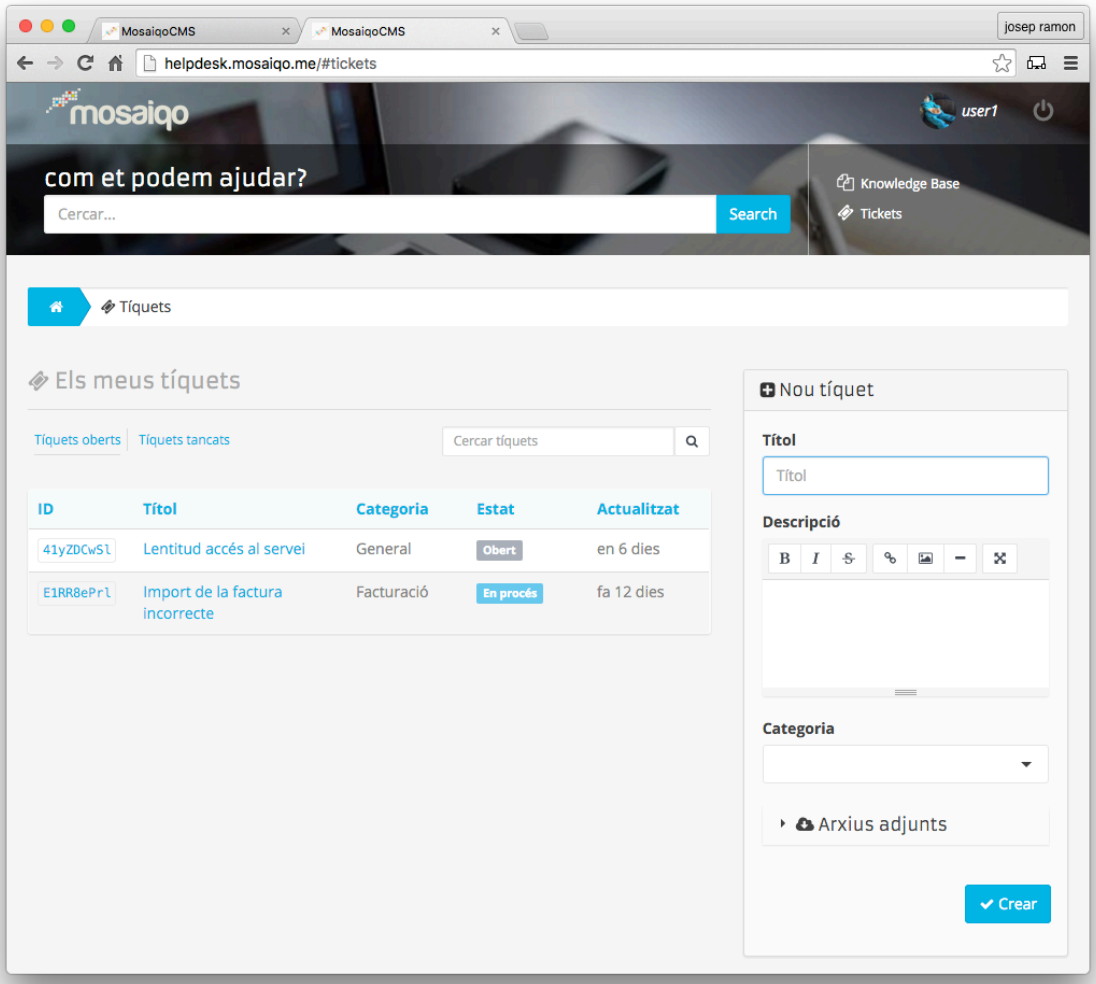


Figura 15 - Tickets: pàgina inicial

Un cop creada la incidència, l'usuari es redirigit a la vista de detall d'aquesta, on pot afegir nous comentaris o tancar el tiquet. Per tancar la incidència l'usuari ha d'introduir un comentari, com ara el motiu o alguna observació.

En el detall de la incidència es mostra la informació introduïda en el moment de crear-la, els comentaris que hagin pogut incorporar posteriorment l'usuari i l'agent assignat, i diversos detalls d'aquesta, com ara la categoria, l'estat, l'agent assignat, les dates d'obertura i última actualització, i

l'historial del tiquet, és a dir, per quins estats a passat, en quin moment, i opcionalment amb comentaris de qui va canviar l'estat (com ara el motiu).

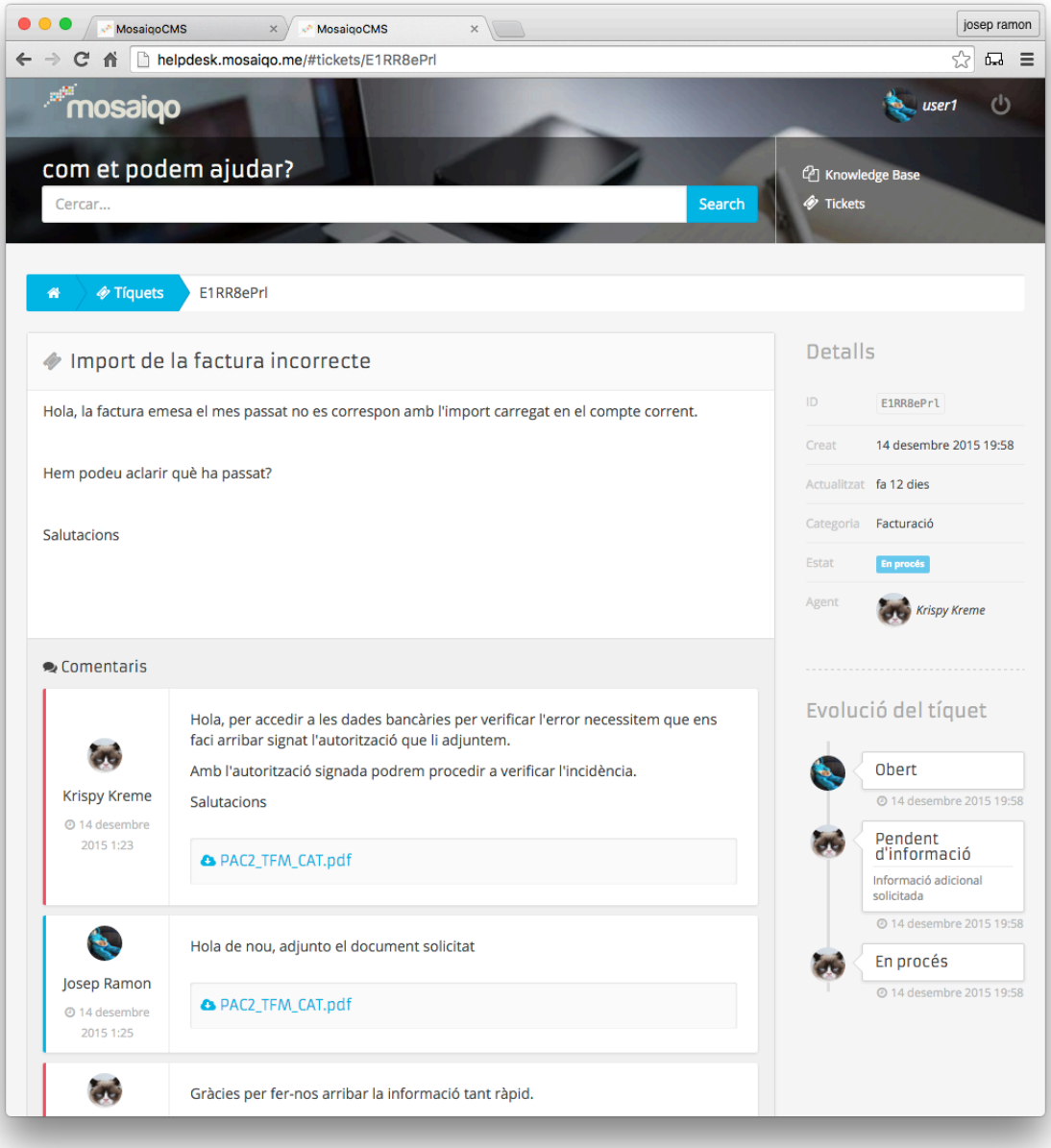


Figura 16 - Tickets - detall ticket obert

En la vista de detall dels tiquets tancats, l'usuari no pot introduir nous comentaris. L'única acció que pot realitzar l'usuari és reobrir la incidència. Per a fer-ho a d'introduir el motiu pel qual reobre el tiquet.

1.2 Aplicació dels administradors i agents

1.2.1 Accés/dades usuari:

L'aplicació dels administradors no disposa d'un formulari de registre. L'administrador es crea en el moment de configurar l'aplicació (tot i què se'n poden crear de nous posteriorment), i els agents són donats d'alta per un dels administradors.
La gestió de les dades del compte i el perfil funciona exactament igual que en l'aplicació dels usuaris.

1.2.2 Gestió d'agents:

A aquesta secció només hi poden accedir els administradors. Si un agent hi intenta accedir l'aplicació mostrarà un error d'autorització. Aquí els administradors poden crear nous agents, editar-ne alguns detalls i eliminar-los. També hi ha un enllaç a l'apartat de gestió de categories del sistema de tiquets, on l'administrador pot assignar els agents a cada una de les categories.

1.2.3 Knowledge Base:

A aquest apartat hi poden accedir els administradors i els agents. Permet i editar crear articles, categories i etiquetes.
El procediment normal per crear etiquetes és simplement introduint la paraula desitjada en el camp etiquetes en el moment de crear o editar un article, però també es poden crear directament en l'apartat d'etiquetes, tot i què aquest està concebut principalment per a la seva gestió, es a dir, per a editar-les (per exemple per afegir una descripció o personalitzar la url), o per eliminar-les de cop de tots els articles que les tinguin assignades.

Tant el llistat d'etiquetes com el de categories mostren el total d'articles assignats. Al punxar al número es mostren els articles vinculats.

La secció dels articles mostra el llistat, amb una paginació del tipus *infinite scrolling,* és a dir, que es van carregant automàticament nous articles al fer *scroll*, i un cercador, junt a l'article que està sent editat. Des del llistat es poden eliminar i publicar/despublicar directament els articles sense necessitat de carregar l'article. Tots els paràmetres dels articles són prou clars, i els únics que pot resultar adequat explicar són les opcions de publicació i els paràmetres avançats.

Quan es crea un article, inicialment, i a no ser que l'autor indiqui el contrari, aquest es crea com a no publicat, és a dir, no es visible pels usuaris finals. L'autor pot publicar l'article en qualsevol moment, o pot definir el dia i la hora en la que es publicarà automàticament.

Dins de les opcions avançades hi ha tres camps:

- Slug (133): Forma part de la url de l'article. El genera automàticament la API al donar d'alta un nou article (en el moment de crear l'article el camp no està disponible), i els administradors i agents el poden canviar posteriorment.
- Extracte: L'extracte de l'article és un resum opcional d'aquest. En els llistats d'articles (per categoria o les cerques, per exemple) normalment es mostra el títol i un fragment (el principi) del cos dels articles. Si es defineix l'extracte, aquest es mostrarà en comptes del fragment de l'article.
- Pes: Serveix per alterar l'ordre dels articles en els llistats. Aquests es mostren en principi per la data de publicació. Assignant un pes als articles aquests apareixeran abans que els que no el tenen assignat.

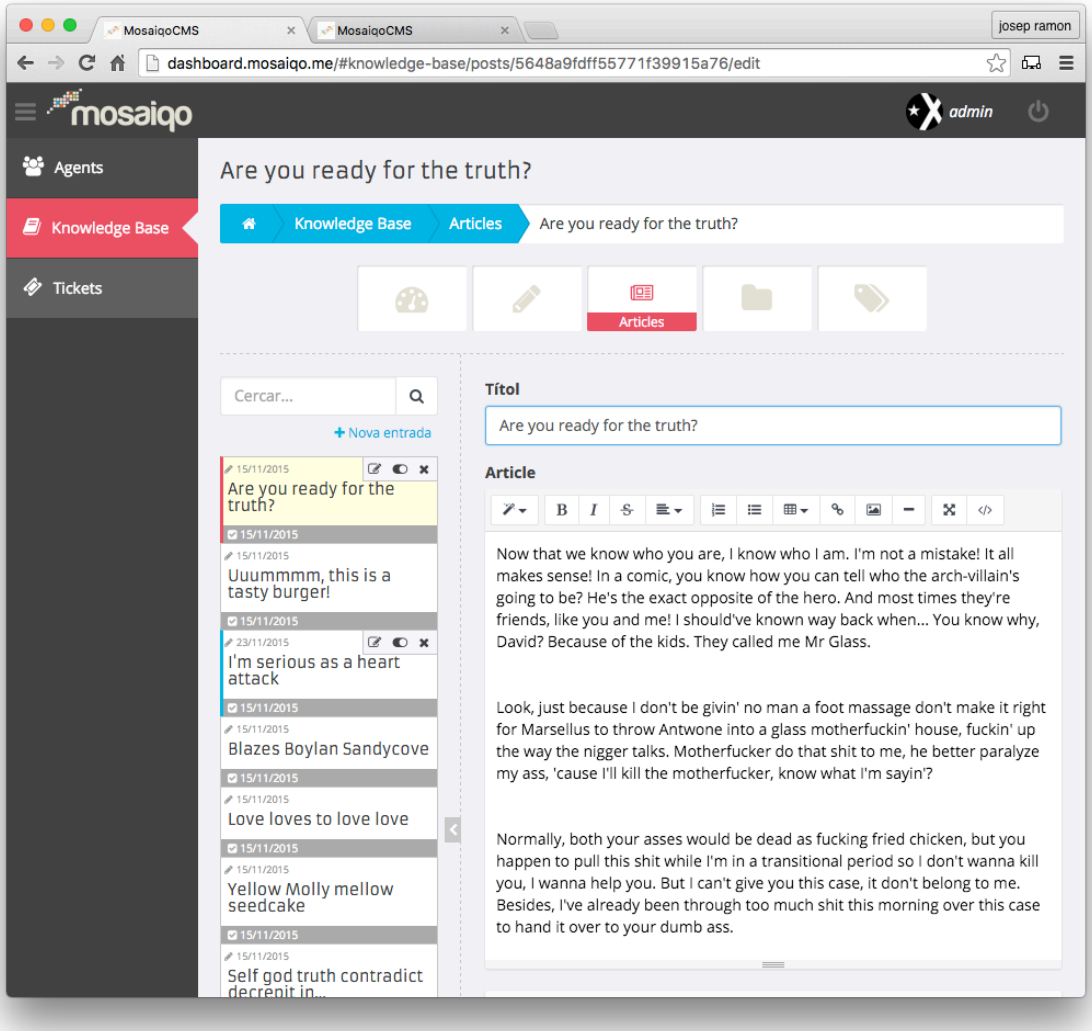


Figura 17 - Knowledge base: gestió d'articles

1.2.4 Tiquets:

La informació que es mostra en aquest apartat és diferent en funció del perfil de l'usuari.
Tant els administradors com els gestors tenen accés a un apartat d'estadístiques generals sobre el sistema de tiquets. Al passar el cursor per sobre les gràfiques (o prémer-hi en el cas de dispositius tàctils) es mostra informació addicional. En les gràfiques que està indicat, al punxar-hi es mostren més dades relacionades.

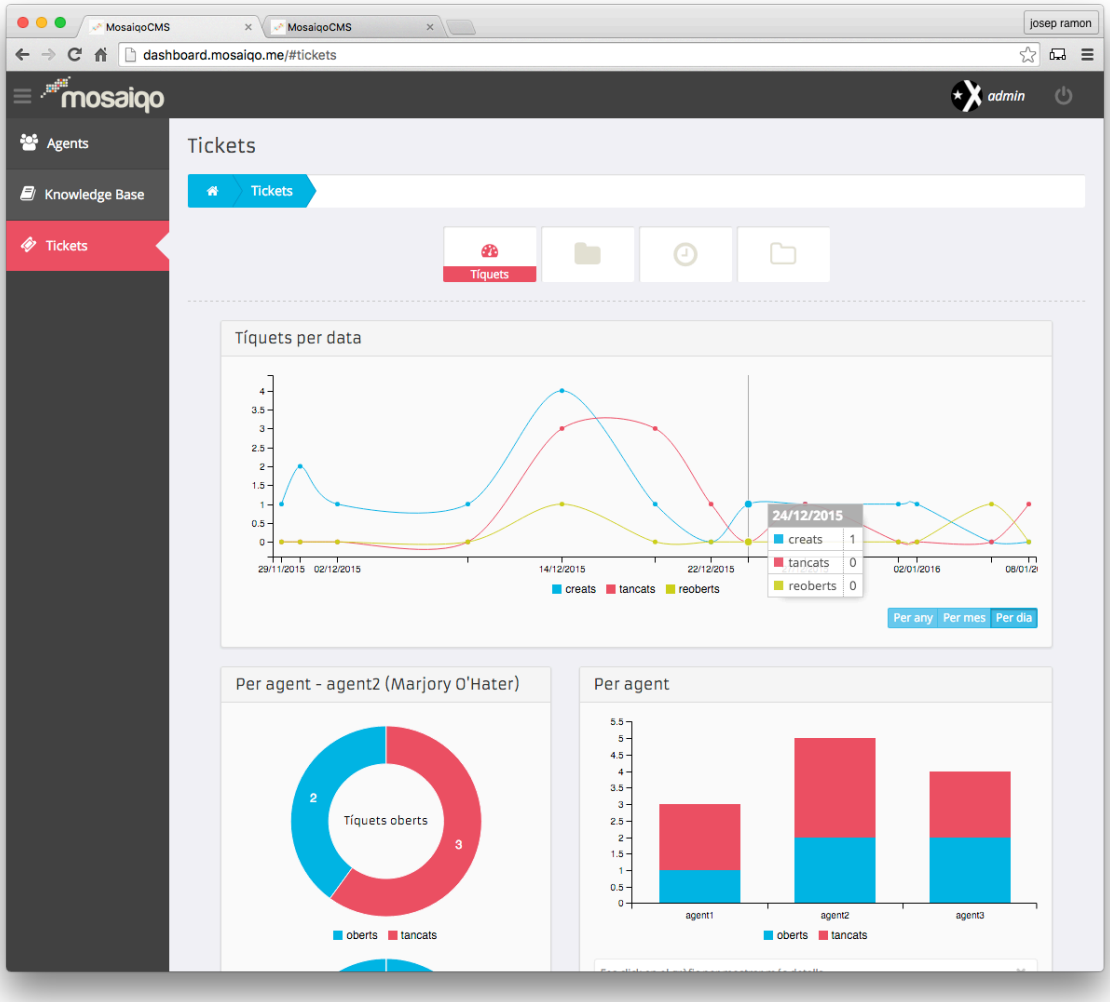


Figura 18 - Tickets: detall estadístiques

Els administradors tenen accés a la gestió de categories i estats per tal de configurar com ha de ser el procés de resolució d'incidències.

Els administradors també tenen accés a un llistat de tiquets sense categoria assignada. La categoria és un atribut obligatori dels tiquets, i aquests no es poden crear sense assignar-ne una. Si un cop el sistema ja està en funcionament l'administrador decideix eliminar una categoria amb tiquets assignats, aquests no s'eliminen però se'ls desvincula la categoria. Com els agents només tenen accés als tiquets de les categories que tenen assignats, és necessari que un dels administradors reassigni els

tiquets *orfes* després d'eliminar una categoria. Aquesta és la única interacció possible dels administradors amb els tiquets, ja que la seva gestió és responsabilitat dels agents.

Els agents poden accedir al llistat de tiquets assignats i els que estan sense agent assignar de les categories que tingui vinculades l'agent. És a dir, un agent vinculat només a la categoria *Suport tècnic* no tindrà accés als tiquets sense assignar de la categoria *Facturació.* Els agents s'han d'autoassignar els tiquets.

Els filtres *tiquets oberts/tiquets tancats, els meus tiquets/tiquets sense assignar* i el cercador són acumulatius, es a dir, que si agent té seleccionat *tiquets oberts* i *els meus tiquets* i realitza una cerca, els resultats només mostraran tiquets que coincideixin amb el criteri de cerca que estiguin oberts i assignats a l'agent.

En els tiquets assignats, els agents poden editar-ne alguns paràmetres, com canviar la prioritat, l'estat o afegir-hi etiquetes. També hi poden afegir comentaris visibles per l'usuari que ha obert la incidència (que a més rebrà una notificació per correu electrònic), o comentaris privats (que només poden veure els agents). Si l'usuari afegeix un comentari a un tiquet assignat a un agent, aquest rebrà una notificació.

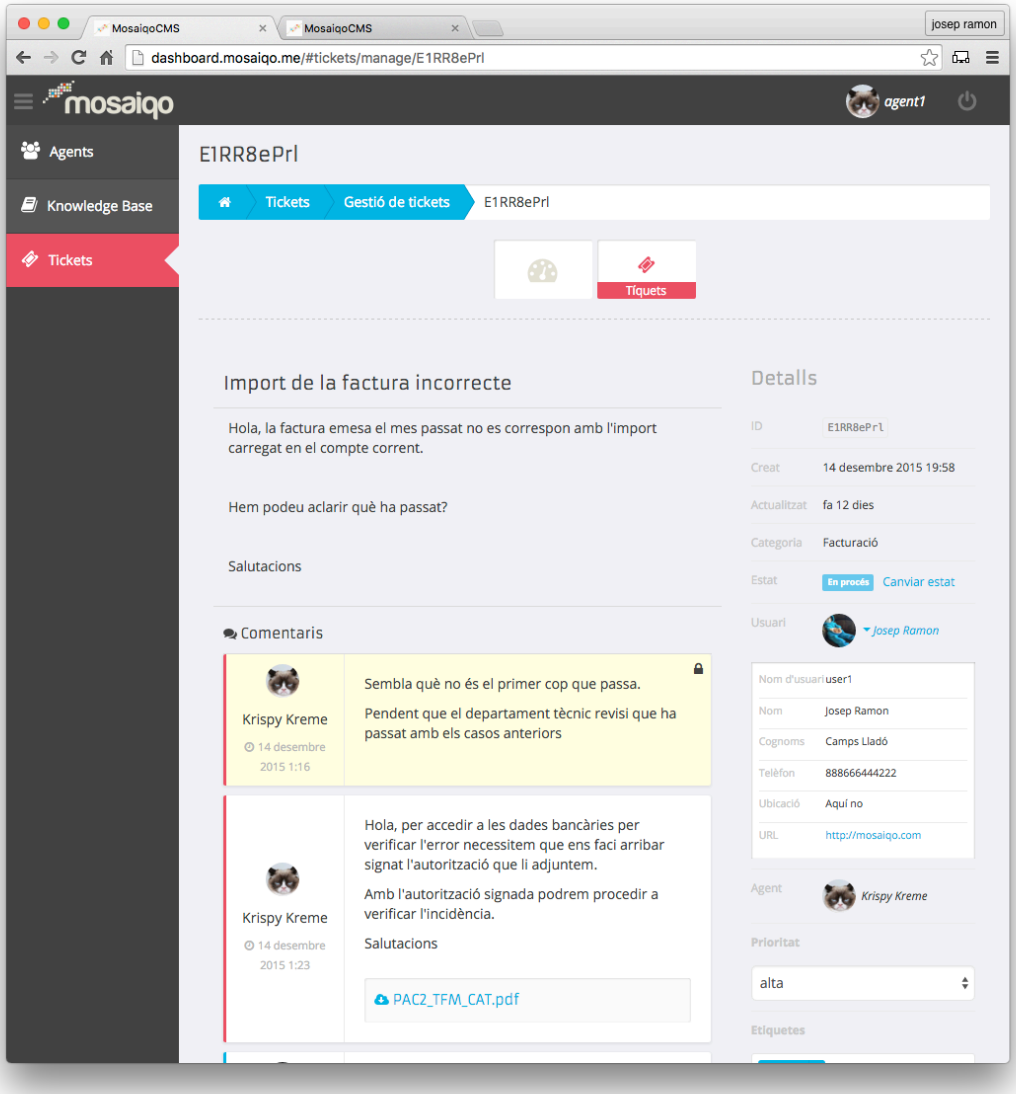


Figura 19 - Tiquets: detall tiquet per als agents

2. Prototips

2.1 Aplicacions web

En el procés de desenvolupament de les aplicacions s'han elaborat *wireframes* només de l'aplicació dels usuaris, perquè l'aplicació dels administradors té la mateixa estructura què una aplicació desenvolupada recentment amb característiques similars, i per tant no s'ha considerat necessari.

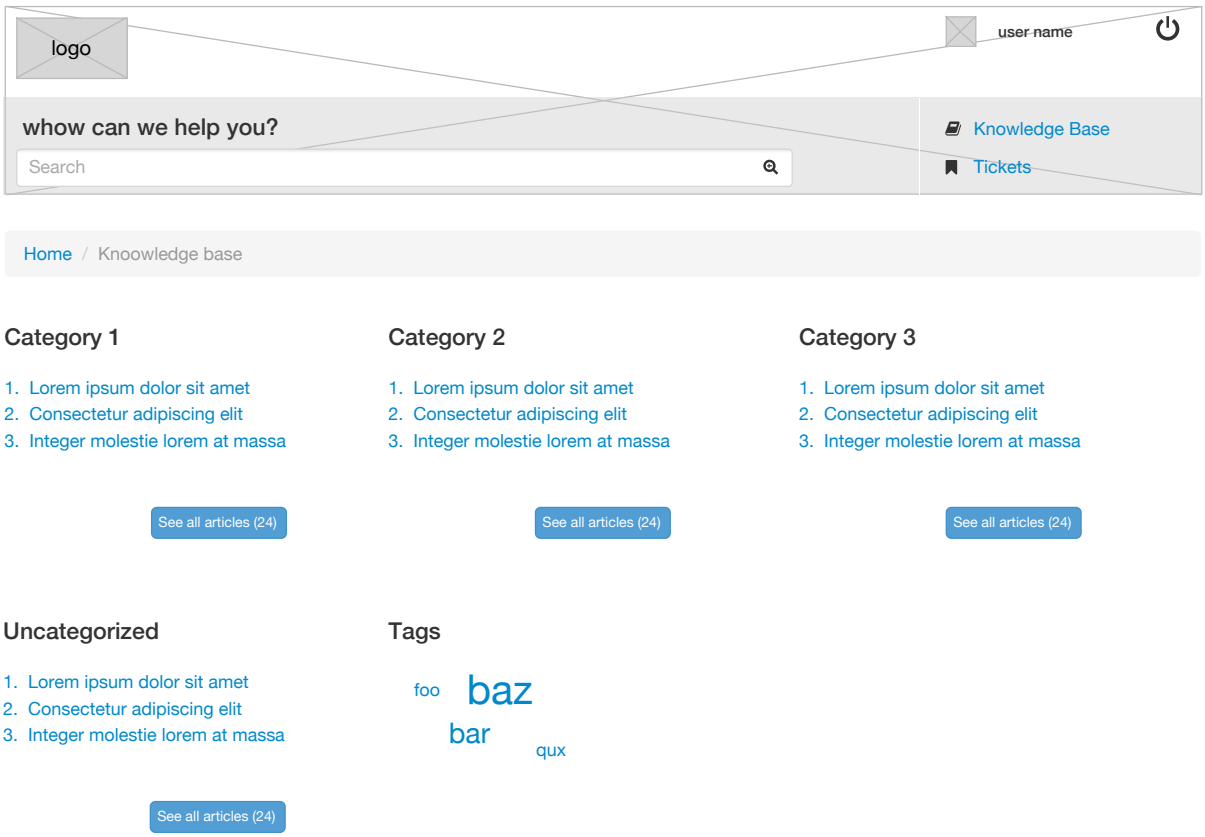


Figura 20 - Knowledge base - Home (versió desktop)

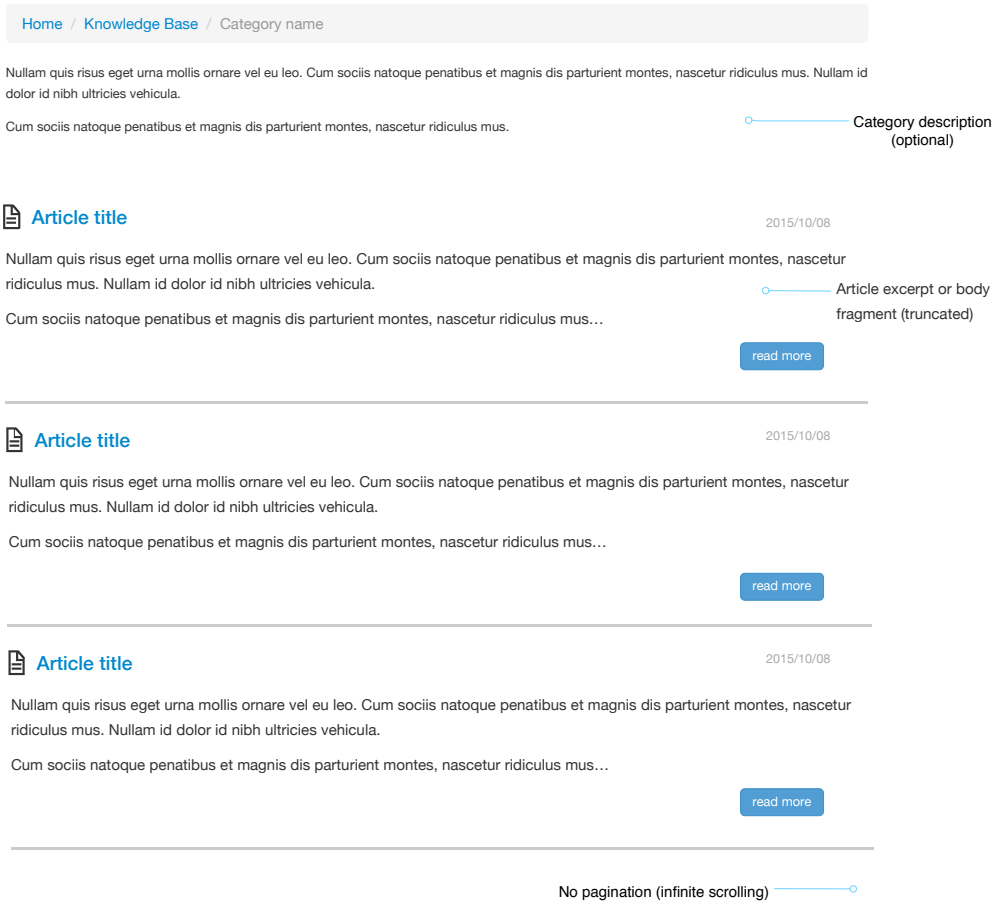


Figura 21 - Knowledge Base - Llistat articles (versió desktop)

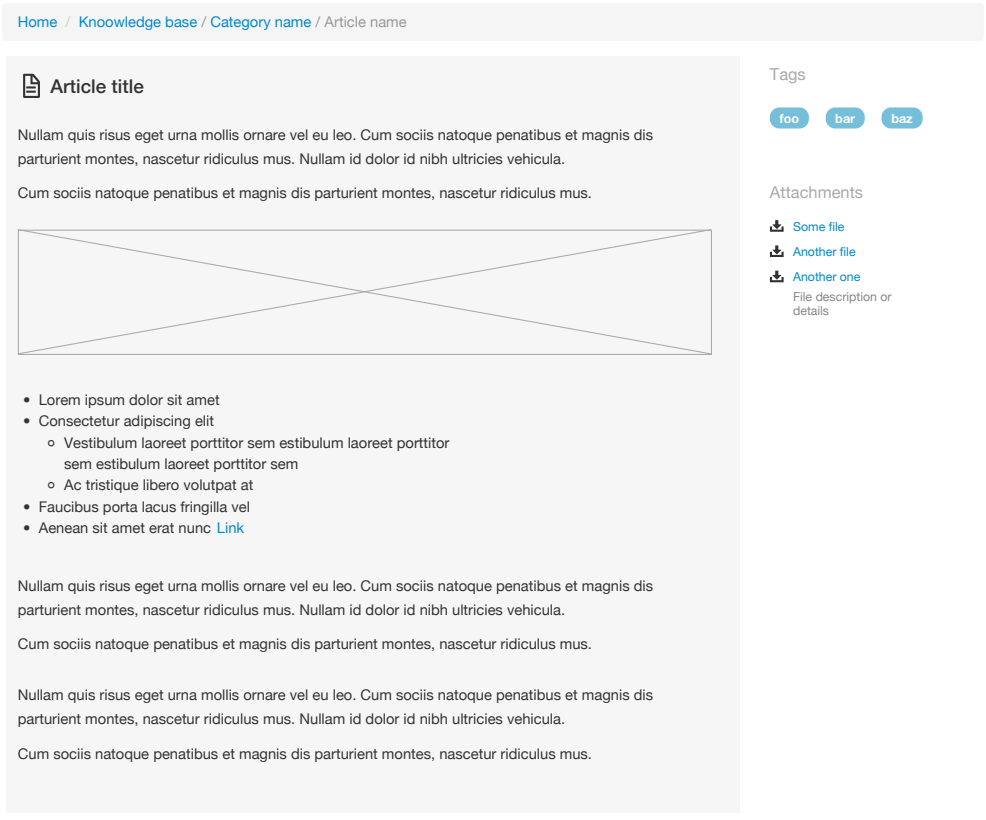
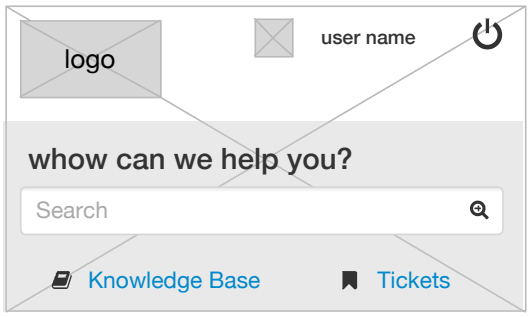


Figura 22 - Knowledge Base – Detall article (versió desktop)



Category 1

1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa

See all articles (24)

Category 2

1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa

See all articles (24)

Uncategorized

1. Lorem ipsum dolor sit amet
2. Consectetur adipiscing elit
3. Integer molestie lorem at massa

See all articles (24)

Tags

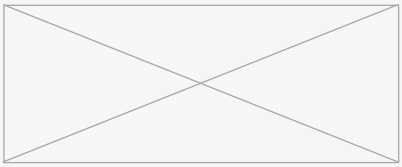
foo baz bar qux

Figura 23 - Knowledge Base - Detall article (versió mòbil)

Article title

Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.



- Lorem ipsum dolor sit amet
- Consectetur adipiscing elit
  - Vestibulum laoreet porttitor sem estibulum laoreet porttitor sem estibulum laoreet porttitor sem
  - Ac tristique libero volutpat at
- Faucibus porta lacus fringilla vel Link
- Aenean sit amet erat nunc

Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula.

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Tags

foo bar baz

Attachments

Some file
Another file
Another one
File description or details

Figura 24 - Knowledge Base - Detall article (versió mòbil)

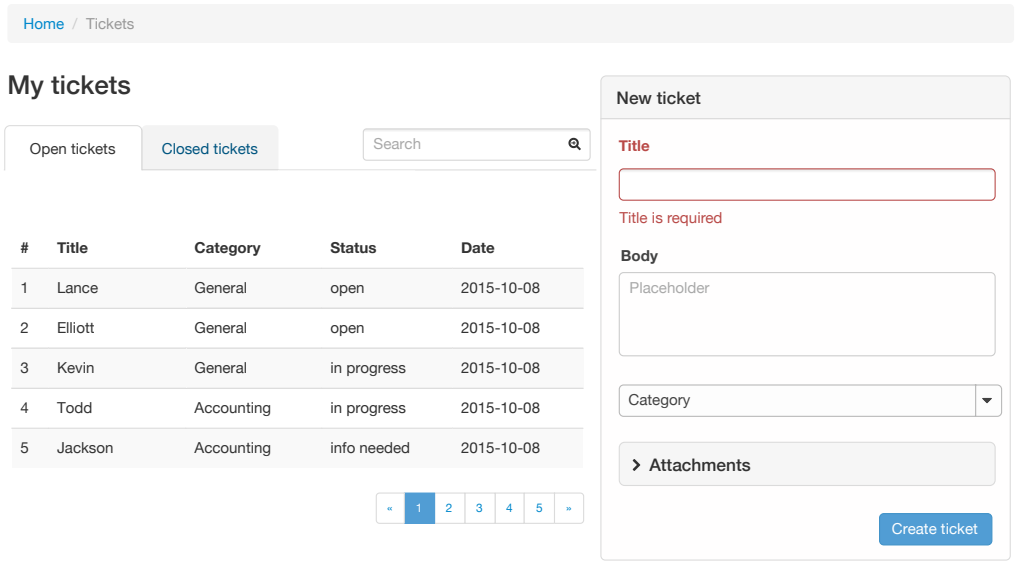


Figura 25 - Tickets - Home (versió desktop)

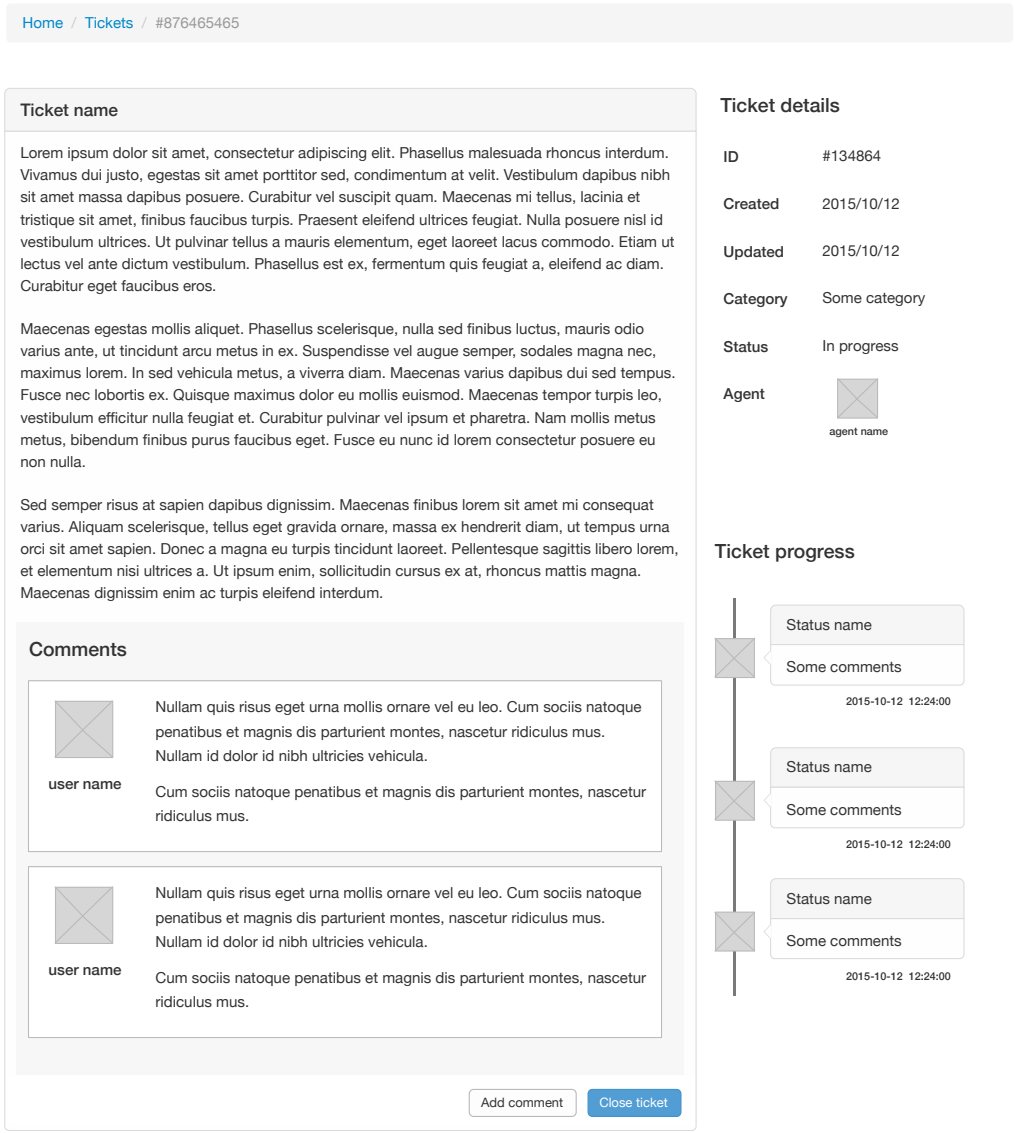


Figura 26 - Tickets - Detall (versió desktop)

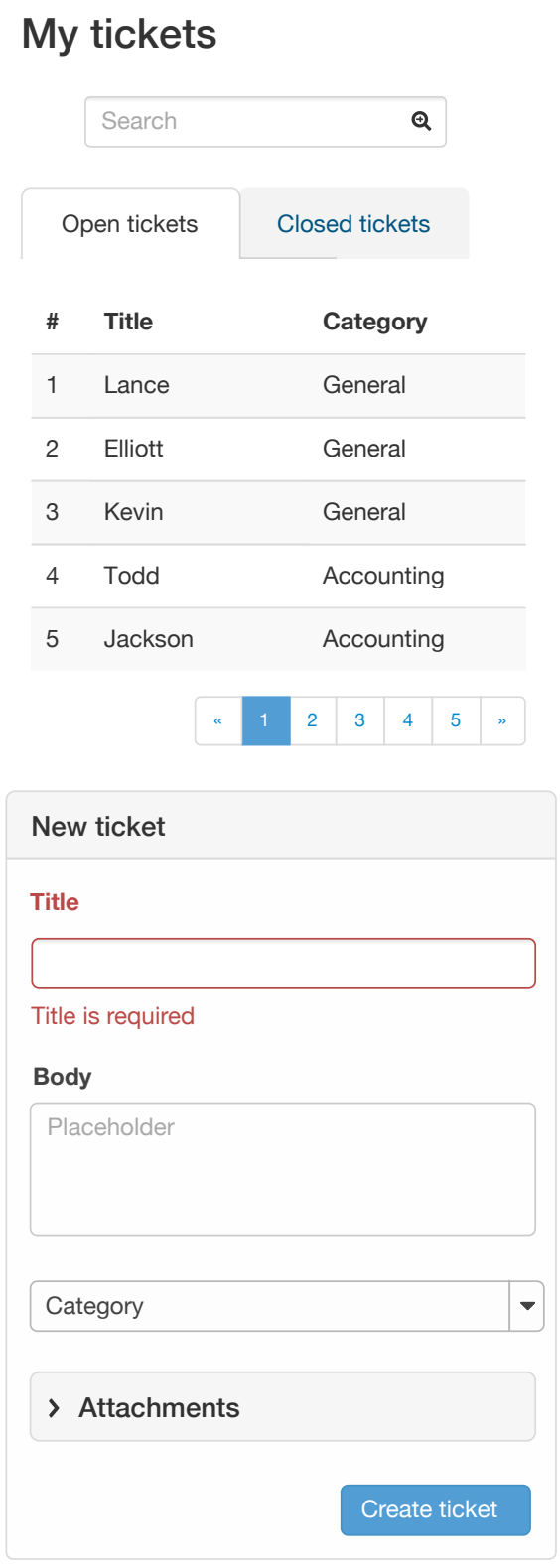


Figura 27 - Tickets - Home (versió mòbil)

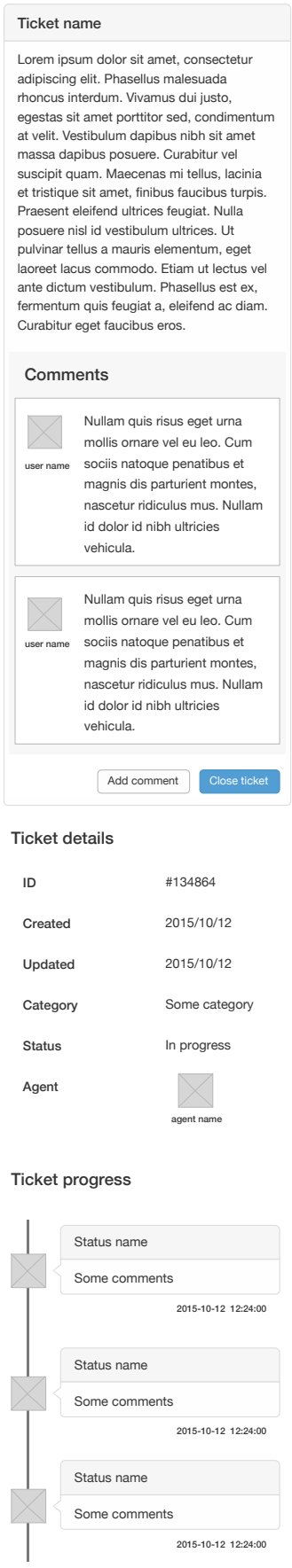


Figura 28 - Tickets - Detall (versió mòbil)

2.2 API

Abans d'iniciar la implementació de la API es va desenvolupar un prototip en node i un altre en PHP perquè l'experiència anterior en el desenvolupament d'aplicacions *serverside* era bàsicament amb PHP, i per al projecte es va considerar que resultaria enriquidor i motivant experimentar amb una tecnologia alternativa. Per tant es va desenvolupar un prototip de la API en els dos llenguatges per tal de valorar-ne la viabilitat i comparar el rendiment.

Un cop desenvolupats els prototips en els dos llenguatges i verificada la viabilitat i idoneïtat de la implementació utilitzant node, es va configurar un servidor remot, una màquina virtual en un proveïdor de *hosting,* exclusivament per a la prova de rendiment, instal·lant exclusivament els components imprescindibles per al funcionament de les aplicacions. El servidor en el que s'executaven les aplicacions disposava de 2 cores, 2 GB Ram, i 40 GB SSD Disk, i els tests es van iniciar des d'una màquina remota, utilitzant ApacheBench (134).

En general el rendiment de la implementació en node ha estat molt superior que la solució implementada en PHP, fins a 10vegades més ràpid en alguna de les proves. Els resultats del *bechmark són els següents:*

Mongo + Node:

```
ab -c 100 -n 500 http://localhost:9999/api/articles

Server Software:
Server Hostname:        localhost
Server Port:            9999

Document Path:          /api/articles
Document Length:        17596 bytes

Concurrency Level:      100
Time taken for tests:   1.942 seconds
Complete requests:      500
Failed requests:        0
Total transferred:      8942503 bytes
HTML transferred:       8798000 bytes
Requests per second:    257.43 [#/sec] (mean)
Time per request:       388.458 [ms] (mean)
Time per request:       3.885 [ms] (mean, across all concurrent requests)
Transfer rate:          4496.19 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.9      0       5
Processing:    41  357  74.6    348     582
Waiting:       41  356  74.7    347     582
Total:         46  357  74.2    348     583

Percentage of the requests served within a certain time (ms)
  50%    348
  66%    377
  75%    398
  80%    408
  90%    450
  95%    484
  98%    524
  99%    551
 100%    583 (longest request)
```

Taula 3 - Resultats Benchmark - Mongo + Node

Nginx + Mongo + PHP (FastCGI):

```
ab -c 100 -n 500 http://localhost/api/articles
```

```
Server Software:        nginx/1.8.0
Server Hostname:        localhost
Server Port:            80

Document Path:          /api/articles
Document Length:        25586 bytes

Concurrency Level:      100
Time taken for tests:   21.539 seconds
Complete requests:      500
Failed requests:        0
Total transferred:      12893000 bytes
HTML transferred:       12793000 bytes
Requests per second:    23.21 [#/sec] (mean)
Time per request:       4307.855 [ms] (mean)
Time per request:       43.079 [ms] (mean, across all concurrent
requests)
Transfer rate:          584.55 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   1.8      0       7
Processing:   160 3923 967.8   4176    5055
Waiting:      158 3923 967.8   4176    5054
Total:        165 3924 966.1   4177    5055

Percentage of the requests served within a certain time (ms)
  50%   4177
  66%   4420
  75%   4522
  80%   4567
  90%   4717
  95%   4835
  98%   4897
  99%   4953
 100%   5055 (longest request)
```

Taula 4 - Resultats Benchmark - Nginx + Mongo + PHP (FastCGI)

Nginx + Mongo + PHP (HHVM (135) (136)**):**

```
ab -c 100 -n 500 http://localhost/api/articles
```

```
Server Software:        nginx/1.8.0
Server Hostname:        localhost
Server Port:            80

Document Path:          /api/articles
Document Length:        25586 bytes

Concurrency Level:      100
Time taken for tests:   12.720 seconds
Complete requests:      500
Failed requests:        0
Total transferred:      12917500 bytes
HTML transferred:       12793000 bytes
Requests per second:    39.31 [#/sec] (mean)
Time per request:       2543.936 [ms] (mean)
Time per request:       25.439 [ms] (mean, across all concurrent requests)
Transfer rate:          991.75 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    3   5.9      0      17
Processing:   282 2296 731.8   2210   12701
Waiting:      281 2296 731.8   2210   12697
Total:        296 2299 730.1   2212   12718

Percentage of the requests served within a certain time (ms)
  50%   2212
  66%   2539
  75%   2644
  80%   2707
  90%   2993
  95%   3108
  98%   3145
  99%   3246
 100%  12718 (longest request)
```

Taula 5 - Resultats Benchmark - Nginx + Mongo + PHP (HHVM)

3. Tests

La metodologia seguida per als tests en les aplicacions dels usuaris i administradors d'una banda i la API per l'altra, han estat força diferents.

Inicialment per a la API es van escriure tests automatitzats de diversos tipus, com tests unitaris, funcionals i d'integració. A mesura que ha avançat el projecte ha estat impossible seguir escrivint tests per manca de temps necessari i a partir d'aquí les noves funcionalitats s'han testejat creant un catàleg de totes les peticions que accepta la API, i regularment executant bateries de proves amb aquests tests i verificant els resultats.

Els tests desenvolupats inicialment no s'han eliminat en cap moment, i s'han executat automàticament cada vegada què s'ha produït un canvi en el codi per verificar que no s'ha produït cap regressió.

Aquests tests automatitzats s'han executat localment de forma continuada (a cada canvi en el codi) durant el desenvolupament, en un sistema d'integració continua (Travis CI (14)) cada cop que s'ha realitzat un *push* al repositori de Github, i en el servidor de proves cada cop que s'ha actualitzat l'aplicació.

Addicionalment s'han realitzat proves de rendiment en diferents punts del desenvolupament, utilitzant Apache AB (137) (134) per verificar el temps de resposta i la quantitat de peticions concurrents que suporta la API.

En les aplicacions web no hi ha hagut temps de desenvolupar tests unitaris, tot i què l'arquitectura d'aquests és molt similar a la utilitzada en un altre projecte que estic desenvolupant en paral·lel, i què compta amb tests d'aquest tipus, per tant s'ha considerat que davant dels recursos escassos de temps era més convenient centrar els recursos en altres tasques i realitzar només tests manuals, en diferents navegadors i dispositius.

Les proves amb usuaris no han estat molt exhaustives per limitacions de calendari. Tot i així s'han realitzat proves amb tres coneguts amb perfils i graus de coneixements tecnològics força diferents per tal de validar les decisions preses.

En els tres casos s'ha realitzat una introducció prèvia del projecte, amb explicacions no exhaustives de les característiques i funcionalitats de les aplicacions, i se'ls hi ha sol·licitat que realitzin una sèrie de tasques, com cercar un article, crear una incidència o realitzar-ne el seguiment.

S'ha observat el comportament dels usuaris i les seves reaccions, i un cop conclosa la prova s'ha realitzat una petita entrevista informal per obtenir-ne la opinió general, quins problemes han pogut trobar i aspectes que canviarien.

Tot i que s'ha tingut en compte totes les aportacions dels usuaris, aquestes han donat lloc a molts pocs canvis perquè en general han estat positives, i en relació als canvis proposats els que s'han considerat oportuns s'han implementat però alguns s'han descartat perquè la mostra d'usuaris era molt petita, i el canvi sol·licitat no s'ha considerat apropiat.

Capítol 6: Conclusions i línies de futur

1. Conclusions

Els objectius plantejats a l'inici del projecte s'han assolit en general, amb un grau de satisfacció força elevat del procés i els resultats. El projecte ha permès experimentar amb noves tecnologies i metodologies de desenvolupament en les que es tenia poca o cap experiència i ha resultat un procés molt enriquidor a nivell personal.

El desenvolupament d'aplicacions web *frontend* com a SPA, completament autocontingudes, desenvolupades com a aplicacions complertes, i que només es comuniquin amb el servidor per recuperar les dades és un trencament important respecte a la metodologia tradicional de desenvolupament d'aplicacions web basades en *pàgines,* i tot i que no és la solució més adequada per a totes les situacions, els avantatges què aporta aquesta arquitectura en les situacions en les que si és adequada són molt importants.
El fet de desenvolupar el *backend* en Javascript utilitzant Nodejs i bases de dades no relacionals com Mongo o Redis ha permés explorar unes metodologies, tècniques i principis completament nous per mi, ja que l'experiència prèvia havia estat en PHP, MySQL i PostgreSQL. Tot i tenir un perfil de desenvolupador *híbrid,* en els darrers anys havia concentrat l'interés en el desenvolupament *frontend* perquè considerava que oferia un repte major, més varietat i més dinamisme. Després d'experimentar amb aquestes noves tecnologies he recuperat l'interès en el desenvolupament *backend.*

Un dels components del sistema, el xat entre usuaris i agents s'ha hagut d'eliminar per manca de temps. Tot i què quan es va realitzar la planificació inicial del projecte ja es va percebre que seria difícil complir amb tots els elements del sistema per manca de temps, la temporització de les tasques es va ajustar per tal d'encabir-les en el temps disponible. Òbviament aquesta decisió va ser un error, perquè el desenvolupament d'un sistema complex requereix temps, i mai s'ha d'estimar el temps per sota de l'esforç necessari.
A part del component del sistema que s'ha eliminat, el calendari en general s'ha seguit escrupolosament fins a l'última etapa del projecte, quan es va decidir suprimir el component del xat, possibilitat que ja s'havia previst des de l'inici del projecte, per tal de poder mitigar unes petites desviacions en el calendari (3 dies) i concentrar els recursos de temps restants en acabar de polir la resta de components.
En relació al calendari, també es convenient comentar que tot i que s'han respectat les dates fins pràcticament el final, ha estat necessari dedicar al projecte més hores de les inicialment previstes per complir amb les dates, degut a que la estimació de la complexitat d'alguna de les tasques no va ser prou acurada, i a que a mesura que ha avançat el projecte han sorgit tasques que no s'havien previst.

2. Línies de futur

Com s'ha mencionat en apartats anteriors, el sistema d'una banda s'utilitzarà per a un projecte en el que estic treballant, i a part, el codi s'alliberarà sota una llicència *opensource*. Tot i què el sistema és completament funcional, considero que resten alguns aspectes a millorar abans d'alliberar el projecte. Les mancances que cal adreçar abans d'obrir el projecte són el baix nombre de tests automatitzats (unitaris, d'integració, etc.) i la documentació, que hauria de de ser molt més extensa i detallada.
Un cop solucionats aquests detalls s'obrirà l'accés al projecte a Github.

En una segona etapa, es desenvoluparan una sèrie de millores i ampliacions al sistema per tal d'incrementar-ne la seva utilitat.

D'una banda ha quedat pendent el component del xat, un *widget* integrable en qualsevol pàgina web (independent de l'aplicació dels usuaris), que permet als usuaris contactar en temps real amb els agents del sistema per tal de resoldre incidències o dubtes.

Durant l'estudi de la competència es va detectar la importància de poder oferir el servei de tiquets utilitzant múltiples canals. Els principals competidors permeten obrir nous tiquets no només a través de les aplicacions web, sinó també a través d'altres canals, com el correu electrònic o les xarxes socials. Durant el disseny i la implementació de la API ja es va tenir en compte aquesta possibilitat, i està preparada per afegir-hi la nova funcionalitat.

Un altra futura millora contemplada del sistema és la integració amb aplicacions de tercers mitjançant un sistema de *hooks*, és a dir, accions que s'executen automàticament davant de determinats esdeveniments. Per exemple, enviar notificacions de qualsevol tipus o crear *issues* en un *bug tracker* al crear un tiquet.

També s'ha considerat el futur desenvolupament d'aplicacions per a dispositius mòbils utilitzant tecnologies com PhoneGap, i tant el *helpdesk* com el *dashboard* s'han dissenyat i implementat tenint en compte aquestes futures aplicacions.

Finalment s'ha considerat la possibilitat d'explotar comercialment en algun moment el sistema, sota un model SAAS, però és una idea que encara s'ha d'estudiar a fons, i que en qualsevol cas no és una prioritat a mitjà termini.

Bibliografia

1. HelpDesk.com | What is Help Desk Software? [Online]. [cited 2015 10 04]. Available from: http://www.helpdesk.com/help-desk-software-overview/.
2. Wikipedia, the free encyclopedia | Knowledge Base. [Online]. [cited 2015 10 04]. Available from: https://en.wikipedia.org/wiki/Knowledge_base.
3. Wikipedia, the free encyclopedia | Single-page application. [Online]. [cited 2015 10 04]. Available from: https://en.wikipedia.org/wiki/Single-page_application.
4. Takada M. Modern web applications: an overview | Single page apps in depth. [Online]. [cited 2015 10 04]. Available from: http://singlepageappbook.com/goal.html.
5. Wikipedia, the free encyclopedia | Representational state transfer. [Online]. [cited 2015 10 04]. Available from: https://en.wikipedia.org/wiki/Representational_state_transfer.
6. Freshdesk - Online customer support software and helpdesk solution. [Online]. [cited 2015 10 18]. Available from: http://freshdesk.com.
7. Wikipedia, the free encyclopedia | Zendesk. [Online]. [cited 2015 10 18]. Available from: https://en.wikipedia.org/wiki/Zendesk.
8. Scrum Alliance | What is Scrum? An Agile Framework for Completing Complex Projects - Scrum Alliance. [Online]. [cited 2015 12 24]. Available from: https://www.scrumalliance.org/why-scrum.
9. Waffle.io · Work Better on GitHub Issues. [Online]. [cited 2015 12 24]. Available from: https://waffle.io/.
10. Grunt: The JavaScript Task Runner. [Online]. [cited 2015 12 24]. Available from: http://gruntjs.com/.
11. Fenton S. Steve Fenton | Compiling vs transpiling. [Online].; 2012 [cited 2015 12 26]. Available from: https://www.stevefenton.co.uk/2012/11/compiling-vs-transpiling/.
12. Wikipedia, the free encyclopedia | Source-to-source compiler. [Online]. [cited 2015 12 26]. Available from: https://en.wikipedia.org/wiki/Source-to-source_compiler.
13. Postman | Supercharge your API workflow. [Online]. [cited 2015 12 24]. Available from: https://www.getpostman.com/.
14. Travis CI - Test and Deploy with Confidence. [Online]. [cited 2015 12 24]. Available from: https://travis-ci.com/.
15. Sendgrid. [Online]. Available from: https://sendgrid.com/.
16. Amazon Simple Storage Service (S3) - Object Storage. [Online]. [cited 2015 12 23]. Available from: https://aws.amazon.com/s3/.
17. Wikipedia, the free encyclopedia | Amazon S3. [Online]. [cited 2015 12 23]. Available from: https://en.wikipedia.org/wiki/Amazon_S3.
18. GitHub Education. [Online]. [cited 2015 12 23]. Available from: https://education.github.com/.
19. Simple Cloud Infrastructure for Developers | DigitalOcean. [Online]. [cited 2015 12 23]. Available

from: https://www.digitalocean.com/.

20. SendGrid | Email Plans & Pricing | SendGrid. [Online]. [cited 2015 12 23]. Available from: https://sendgrid.com/pricing.
21. Amazon Simple Storage Service (S3) - Object Storage | Cloud Storage Pricing - Amazon Simple Storage Service (S3). [Online]. [cited 2015 12 23]. Available from: https://aws.amazon.com/s3/pricing/.
22. Wikipedia.org | Software as a service. [Online]. [cited 2015 10 18]. Available from: https://en.wikipedia.org/wiki/Software_as_a_service.
23. Zendesk | Customer Service Software & Support Ticket System. [Online]. [cited 2015 10 18]. Available from: https://www.zendesk.com.
24. Wikipedia, the free encyclopedia | Freshdesk. [Online]. [cited 2015 10 18]. Available from: https://en.wikipedia.org/wiki/Freshdesk.
25. Sinha A. NextBigWhat | Zendesk CEO Calls Freshdesk a "FREAKING-RIP-OFF". [Online]. [cited 2015 10 18]. Available from: http://www.nextbigwhat.com/zendesk-ceo-calls-freshdesk-a-rip-off-297/.
26. Freshdesk - Online customer support software and helpdesk solution | In app customer support for android apps | Freshdesk MobiHelp. [Online]. [cited 2015 10 18]. Available from: http://freshdesk.com/mobihelp.
27. Help Desk Software by Help Scout. [Online]. [cited 2015 10 18]. Available from: http://www.helpscout.net.
28. Customer service app and help desk support ticket software for growing businesses | Desk.com. [Online]. [cited 2015 10 18]. Available from: https://www.desk.com.
29. Wikipedia, the free encyclopedia | Salesforce.com. [Online]. [cited 2015 10 18]. Available from: https://en.wikipedia.org/wiki/Salesforce.com#Desk.com.
30. osTicket : Support Ticket System. [Online]. [cited 2015 10 18]. Available from: http://osticket.com.
31. Software Development and Collaboration Tools | Atlassian. [Online]. [cited 2015 10 18]. Available from: https://www.atlassian.com/software/jira.
32. Wikipedia, the free encyclopedia | JIRA. [Online]. [cited 2015 10 18]. Available from: https://en.wikipedia.org/wiki/JIRA.
33. GitHub · Where software is built. [Online]. [cited 2015 10 18]. Available from: https://github.com.
34. Bugzilla : bugzilla.org. [Online]. [cited 2015 10 18]. Available from: https://www.bugzilla.org.
35. Wikipedia, the free encyclopedia | Bugzilla. [Online]. [cited 2015 10 18]. Available from: https://en.wikipedia.org/wiki/Bugzilla.
36. GNATS - GNU Project - Free Software Foundation (FSF). [Online]. [cited 2015 10 18]. Available from: https://www.gnu.org/software/gnats/.
37. Wikipedia, the free encyclopedia | GNATS. [Online]. [cited 2015 10 18]. Available from: https://en.wikipedia.org/wiki/GNATS.

38. Mantis Bug Tracker. [Online]. [cited 2015 10 18]. Available from: https://www.mantisbt.org.
39. Wikipedia, the free encyclopedia | Mantis Bug Tracker. [Online].; 2015. Available from: https://en.wikipedia.org/wiki/Mantis_Bug_Tracker.
40. Zopim Live Chat Software | Engage your Customers | Live Support. [Online]. [cited 2015 10 18]. Available from: https://www.zopim.com.
41. Wikipedia, the free encyclopedia | Zopim. [Online]. [cited 2015 10 18]. Available from: https://en.wikipedia.org/wiki/Zopim.
42. LiveChat. [Online].; 2015. Available from: https://www.livechatinc.com.
43. Wikipedia, the free encyclopedia | LiveChat. [Online]. [cited 2015 10 18]. Available from: https://en.wikipedia.org/wiki/LiveChat.
44. Wikipedia, the free encyclopedia | WYSIWYG. [Online]. [cited 2015 12 26]. Available from: https://en.wikipedia.org/wiki/WYSIWYG.
45. PhoneGap. [Online]. [cited 2015 12 26]. Available from: http://phonegap.com/.
46. Wikipedia, the free encyclopedia | Apache Cordova. [Online]. [cited 2015 12 26]. Available from: https://en.wikipedia.org/wiki/Apache_Cordova.
47. Driscoll J. Jim Driscoll's Blog | Notes on Technology and the Web | Thin Server Architecture. [Online].; 2013 [cited 2015 12 26]. Available from: https://jamesgdriscoll.wordpress.com/2013/12/20/thin-server-architecture/.
48. van der Schee M. LeaseWeb labs | API first architecture or the fat vs thin server debate. [Online].; 2013 [cited 2015 12 26]. Available from: https://www.leaseweb.com/labs/2013/10/api-first-architecture-fat-vs-thin-server-debate/.
49. Mashape - Powering API Driven Software | Three Ways API-First Development is the Future of Web Development. [Online].; 2014 [cited 2015 12 26]. Available from: http://blog.mashape.com/three-ways-api-first-development-is-the-future-of-web/.
50. Express - Node.js web application framework. [Online]. Available from: http://expressjs.com/.
51. MongoDB Hosting: Database-as-a-Service by MongoLab. [Online]. [cited 2015 12 24]. Available from: https://mongolab.com/.
52. Wikipedia, the free encyclopedia | MongoDB. [Online]. [cited 2015 11 18]. Available from: https://en.wikipedia.org/wiki/MongoDB.
53. Fowler M. Martin Fowler. [Online]. Available from: http://martinfowler.com/articles/schemaless/.
54. Wikipedia, the free encyclopedia | NoSQL. [Online]. [cited 2015 10 18].
55. Node.js. [Online]. [cited 2015 10 18]. Available from: https://nodejs.org.
56. Wikipedia, the free encyclopedia | Node.js. [Online]. [cited 2015 10 18]. Available from: https://en.wikipedia.org/wiki/Node.js.
57. Wikipedia, the free encyclopedia | Document-oriented database. [Online]. [cited 2015 11 18]. Available from: https://en.wikipedia.org/wiki/Document-oriented_database.

58. Hernandez A. Smashing Magazine | An Introduction To Full-Stack JavaScript. [Online]. [cited 2015 10 15]. Available from: http://www.smashingmagazine.com/2013/11/introduction-to-full-stack-javascript/.
59. Jones M, Bradley J, Sakimura N. Internet Engineering Task Force (IETF) | RFC 7519 - JSON Web Token (JWT). [Online].; 2015 [cited 2015 11 15]. Available from: https://tools.ietf.org/html/rfc7519.
60. Wikipedia, the free encyclopedia | Nginx. [Online].; 2015. Available from: https://en.wikipedia.org/wiki/Nginx.
61. The Apache HTTP Server Project. [Online]. [cited 2015 12 26]. Available from: https://httpd.apache.org/.
62. Wikipedia, the free encyclopedia | Apache HTTP Server. [Online]. [cited 2015 12 26]. Available from: https://en.wikipedia.org/wiki/Apache_HTTP_Server.
63. Wikipedia, the free encyclopedia | Cross-origin resource sharing. [Online]. [cited 2015 12 26]. Available from: https://en.wikipedia.org/wiki/Cross-origin_resource_sharing.
64. Wikipedia, the free encyclopedia | Publish–subscribe pattern. [Online]. [cited 2015 12 26]. Available from: https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern.
65. Mongoose. Elegant mongodb object modeling for node.js. [Online]. [cited 2015 12 26]. Available from: http://mongoosejs.com/.
66. NPM: Node package manager | shortid: Amazingly short non-sequential url-friendly unique id generator. [Online]. [cited 2015 12 26]. Available from: https://www.npmjs.com/package/shortid.
67. Bootstrap · The world's most popular mobile-first and responsive front-end framework. [Online]. [cited 2015 12 26]. Available from: http://getbootstrap.com/.
68. Kaleidoscope. [Online]. [cited 2015 12 26]. Available from: http://mosaiqo.github.io/kaleidoscope/.
69. Negative Space - Free stock photos, no copyright restrictions. [Online]. [cited 2015 12 26]. Available from: http://negativespace.co/.
70. Negative Space - Free stock photos, no copyright restrictions. | License - Negative Space. [Online]. [cited 2015 12 26]. Available from: http://negativespace.co/license/.
71. Spalinger N, Gaultney V. SIL International. [Online].; 2007 [cited 2015 12 26]. Available from: http://scripts.sil.org/OFL.
72. Foundation TAS. The Apache Software Foundation | Apache License, Version 2.0. [Online].; 2004 [cited 2015 12 26]. Available from: http://www.apache.org/licenses/LICENSE-2.0.html.
73. Google Fonts. [Online]. [cited 2015 12 26]. Available from: https://www.google.com/fonts.
74. Font Awesome, the iconic font and CSS toolkit. [Online]. [cited 2015 12 26]. Available from: http://fontawesome.io/.
75. Nielsen J. Nielsen Norman Group: UX Training, Consulting, & Research | 10 Usability Heuristics for User Interface Design. [Online].; 1995 [cited 2015 12 26]. Available from: https://www.nngroup.com/articles/ten-usability-heuristics/.

76. Association UEP. Usability Body of Knowledge | Principles for Usable Design. [Online]. [cited 2015 12 26]. Available from: http://www.usabilitybok.org/principles-for-usable-design.
77. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed.: New Riders; 2014.
78. The Apache Cassandra Project. [Online]. [cited 2015 12 26]. Available from: http://cassandra.apache.org/.
79. Wikipedia, the free encyclopedia | Apache Cassandra. [Online]. [cited 2015 12 26]. Available from: https://en.wikipedia.org/wiki/Apache_Cassandra.
80. Apache CouchDB. [Online]. [cited 2015 12 26]. Available from: http://couchdb.apache.org/.
81. Wikipedia, the free encyclopedia | CouchDB. [Online]. [cited 2015 12 26]. Available from: https://en.wikipedia.org/wiki/CouchDB.
82. Meteor: The JavaScript App Platform. [Online]. [cited 2015 12 26]. Available from: https://www.meteor.com/.
83. hapi.js: A rich framework for building applications and services. [Online]. [cited 2015 12 26]. Available from: http://hapijs.com/.
84. Sails.js | Realtime MVC Framework for Node.js. [Online]. [cited 2015 12 26]. Available from: http://sailsjs.org/.
85. Sinatra: Classy web-development dressed in a DSL. [Online]. [cited 2015 12 26]. Available from: https://github.com/sinatra/sinatra.
86. auth0/express-jwt. [Online]. [cited 2015 12 26]. Available from: https://github.com/auth0/express-jwt.
87. expressjs/express-paginate: Paginate middleware. [Online]. [cited 2015 12 26]. Available from: https://github.com/expressjs/express-paginate.
88. buunguyen/mongoose-deep-populate: Mongoose plugin to enable deep population of nested models. [Online]. [cited 2015 12 26]. Available from: https://github.com/buunguyen/mongoose-deep-populate.
89. weisjohn/mongoose-deleted: a soft-delete implementation utilizing mongoose middleware. [Online]. [cited 2015 12 26]. Available from: https://github.com/weisjohn/mongoose-deleted.
90. edwardhotchkiss/mongoose-paginate: Mongoose.js (Node.js & MongoDB) Document Query Pagination. [Online]. [cited 2015 12 26]. Available from: https://github.com/edwardhotchkiss/mongoose-paginate.
91. yields/mongoose-time: timestamps for mongoose schemas. [Online]. [cited 2015 12 26]. Available from: https://github.com/yields/mongoose-time.
92. expressjs/multer: Node.js middleware for handling `multipart/form-data`. [Online]. [cited 2015 12 26]. Available from: https://github.com/expressjs/multer.
93. Nodemailer | Nodemailer README. [Online]. [cited 2015 12 26]. Available from: http://nodemailer.com/.
94. Wikipedia, the free encyclopedia | Gravatar. [Online]. [cited 2015 12 26]. Available from:

https://en.wikipedia.org/wiki/Gravatar.

95. CoffeeScript. [Online]. [cited 2015 11 15]. Available from: http://coffeescript.org.
96. Wikipedia, the free encyclopedia | CoffeeScript. [Online]. [cited 2015 11 15]. Available from: https://en.wikipedia.org/wiki/CoffeeScript.
97. mosaiqo/frontendAppLib: Library to build fronted apps using Marionnette. [Online]. [cited 2015 12 26]. Available from: https://github.com/mosaiqo/frontendAppLib.
98. Backbone.js. [Online]. [cited 2015 11 15]. Available from: http://backbonejs.org.
99. Marionette.js – The Backbone Framework. [Online]. [cited 2015 11 15]. Available from: http://marionettejs.com.
100. Ember.js - A framework for creating ambitious web applications. [Online]. [cited 2015 12 26]. Available from: http://emberjs.com/.
101. AngularJS — Superheroic JavaScript MVW Framework. [Online]. [cited 2015 12 26]. Available from: https://angularjs.org/.
102. A JavaScript library for building user interfaces | React. [Online]. [cited 2015 12 26]. Available from: https://facebook.github.io/react/.
103. Backbone-Associations. [Online]. [cited 2015 12 26]. Available from: http://dhruvaray.github.io/backbone-associations/.
104. alexbeletsky/backbone-computedfields: Computed fields for Backbone.Model. [Online]. [cited 2015 12 26]. Available from: https://github.com/alexbeletsky/backbone-computedfields.
105. marionettejs/backbone.radio: Messaging patterns for Backbone applications. [Online]. [cited 2015 12 26]. Available from: https://github.com/marionettejs/backbone.radio.
106. marionettejs/backbone.syphon: Serialize a Backbone.View in to a JavaScript object. [Online]. [cited 2015 12 26]. Available from: https://github.com/marionettejs/backbone.syphon.
107. jQuery: write less, do more. [Online]. [cited 2015 12 26]. Available from: https://jquery.com/.
108. Mozilla Developer Network | XMLHttpRequest - Web APIs | MDN. [Online]. [cited 2015 12 26]. Available from: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest.
109. Handlebars.js: Minimal Templating on Steroids. [Online]. [cited 2015 12 26]. Available from: http://handlebarsjs.com/.
110. i18next: internationalization framework | i18next. [Online]. [cited 2015 12 26]. Available from: http://i18next.com/.
111. Moment.js. [Online]. [cited 2015 12 26]. Available from: http://momentjs.com/.
112. C3.js | D3-based reusable chart library. [Online]. [cited 2015 12 26]. Available from: http://c3js.org/.
113. [Online]. [cited 2015 12 26]. Available from: http://d3js.org/.
114. Stylus: Expressive, dynamic, robust CSS — expressive, robust, feature-rich CSS preprocessor. [Online]. [cited 2015 12 26]. Available from: http://stylus-lang.com/.
115. Wikipedia, the free encyclopedia | Stylus (stylesheet language). [Online]. [cited 2015 12 26].

Available from: https://en.wikipedia.org/wiki/Stylus_(stylesheet_language).

116. postcss/autoprefixer: Parse CSS and add vendor prefixes to rules by Can I Use. [Online]. [cited 2015 12 26]. Available from: https://github.com/postcss/autoprefixer.
117. Mozilla Developer Network | Vendor Prefix. [Online]. [cited 2015 12 26]. Available from: https://developer.mozilla.org/en-US/docs/Glossary/Vendor_Prefix.
118. Underscore.js. [Online]. [cited 2015 12 26]. Available from: http://underscorejs.org/.
119. Mocha - the fun, simple, flexible JavaScript test framework. [Online]. [cited 2015 12 26]. Available from: https://mochajs.org/.
120. Chai Assertion Library. [Online]. [cited 2015 12 26]. Available from: http://chaijs.com/.
121. Sinon.JS. [Online]. [cited 2015 12 26]. Available from: http://sinonjs.org/.
122. Wikipedia, the free encyclopedia | Long-term support. [Online]. [cited 2015 12 24]. Available from: https://en.wikipedia.org/wiki/Long-term_support.
123. Home of Open Source and Enterprise-Class Redis. [Online].; 2015. Available from: https://redislabs.com/.
124. Node.js | Download | Node.js. [Online]. [cited 2015 12 26]. Available from: https://nodejs.org/en/download/.
125. Node.js | Installing Node.js via package manager. [Online]. [cited 2015 12 26]. Available from: https://nodejs.org/en/download/package-manager/.
126. MongoDB for GIANT Ideas | MongoDB | Downloads | MongoDB. [Online].; 2015. Available from: https://www.mongodb.org/downloads.
127. MongoDB for GIANT Ideas | MongoDB | Install MongoDB. [Online]. [cited 2015 12 26]. Available from: https://docs.mongodb.org/v3.0/installation/.
128. Redis | Redis - Download. [Online]. [cited 2015 12 26]. Available from: http://redis.io/download.
129. Amazon Web Services (AWS) - Cloud Computing Services | Virtual Hosting of Buckets - Amazon Simple Storage Service. [Online]. [cited 2015 12 26]. Available from: http://docs.aws.amazon.com/AmazonS3/latest/dev/VirtualHosting.html.
130. curl and libcurl. [Online]. [cited 2015 11 15]. Available from: http://curl.haxx.se.
131. Wikipedia, the free encyclopedia | cURL. [Online]. [cited 2015 11 15]. Available from: https://en.wikipedia.org/wiki/CURL.
132. Gravatar - Globally Recognized Avatars. [Online]. [cited 2015 12 26]. Available from: https://en.gravatar.com/.
133. Wikipedia, the free encyclopedia | Semantic URL - Slug. [Online]. [cited 2015 12 26]. Available from: https://en.wikipedia.org/wiki/Semantic_URL#Slug.
134. Wikipedia, the free encyclopedia | ApacheBench. [Online]. [cited 2015 11 14]. Available from: https://en.wikipedia.org/wiki/ApacheBench.
135. Facebook. HHVM. [Online]. [cited 2015 11 14]. Available from: http://hhvm.com.

136. Wikipedia, the free encyclopedia | HipHop Virtual Machine. [Online]. [cited 2015 11 14]. Available from: https://en.wikipedia.org/wiki/HipHop_Virtual_Machine.

137. Apache Software Foundation | ab - Apache HTTP server benchmarking tool. [Online]. [cited 2015 11 14]. Available from: https://httpd.apache.org/docs/2.2/programs/ab.html.

Annexos

Annex A: Glossari

- Administrador: propietari o responsable de l'operatòria del servei.
- Agent: usuari amb permisos especials en el sistema, encarregat d'oferir suport als usuaris *normals*.
- AJAX: (*asynchronous JavaScript and XML*) conjunt de tècniques de desenvolupament utilitzades en el desenvolupament web *client side* que permeten la comunicació amb servidors remots de forma asíncrona.
- Amazon S3: servei d'emmagatzemament *on line* ofert per Amazon Web Services.
- API: (Aplication Programing Interface): conjunt de rutines, protocols i eines que exposa un determinat sistema o llibreria.
- Backend: en enginyeria del software, *backend* és la especialitat encarregada del desenvolupament d'aplicacions que s'executen en el servidor.
- Breadcrumbs: Control gràfic utilitzat en el disseny d'interfícies amb l'objectiu de mostrar als usuaris on es troven dons d'una estructura jerarquitzada, com ara un sistema d'arxius.
- Coffeescript: llenguatge de programació que es transcompila a Javascript, i que ofereix una major brevetat i claredat que aquest, afegint , i que ofereix una major brevetat i claredat que aquest, afegint *syntactic sugar* inspirat en llenguatges com Ruby, Python o Haskell.
- CORS:
- Dashboard: un *dashboard* o panell de control és una aplicació que permet visualitzar l'estat d'un sistema i gestionar-ne el funcionament.
- Framework: conjunt de llibreries i components genèrics per al desenvolupament en software. Sovint aquestes *frameworks* també *imposen* una determinada estructura, metodologia i patrons arquitecturals.
- Frontend: en enginyeria del software, *frontend* és la especialitat encarregada del desenvolupament d'aplicacions que s'executen el dispositiu client, per exemple en el navegador dels usuaris.
- Grunt: és un *task runner* implementat en Javascript i que funciona sobre NodeJS utilitzat per automatitzar tasques de diferent tipus, utilitzat especialment en el desenvolupament web.
- Helpdesk: aplicació o sistema informàtic dissenyat per a oferir informació i suport als usuaris d'un servei o empresa.
- Infinite scrolling: tècnica utilitzada en el desenvolupament d'interfícies en la que es van carregant continguts a mesura que l'usuari va fent *scroll* en una vista. S'utilitza com a alternativa a la paginació.
- Knowledge base: Sistema que permet emmagatzemar i posteriorment recuperar informació de diversa naturalesa. Una implementació habitual d'aquest tipus de sistemes són els repositoris d'articles categoritzats segons la seva naturalesa o temàtica.

- Locale: arxiu o conjunt d'arxius amb paràmetres i traduccions per a una determinada llengua, utilitzat en aplicacions multi idioma per a mostrar els textos en l'idioma corresponent, formatar correctament les dates, etcètera.
- Mongo: Base de dades *open source* orientada a documents, que emmagatzema documents en format similar a JSON (BSON o Binari JSON) en comptes d'estructures tabulars com les bases de dades relacionals.
- MVC:
- Nodejs: *Runtime* basat en Google V8 (el motor de Javascript utilitzat per Google Chrome) utilitzat per a desenvolupar aplicacions en Javascript principalment per a ser executades en servidors, tot i que també s'utilitza per al desenvolupament d'aplicacions d'escriptori.
- NOSQL: terme utilitzat per referir-se a les bases de dades no relacionals.
- Opensource: Model de desenvolupament i distribució de *software*, que promou l'accés universal al codi mitjançant llicències permissives.
- Redis: Servidor de base de dades NOSQL, del tipus *key-value store,* que funciona en memòria (amb persistència a disc opcional).
- Responsive web design: conjunt de tècniques en el desenvolupament web amb l'objectiu d'oferir una experiència òptima en tot tipus de dispositius i mides de pantalla adaptant el disseny i la disposició dels elments que componen les pàgines.
- REST: (Representational state transfer) Arquitectura de software per a sistemes distribuïts modelada segons el funcionament del World Wide Web, basada en els mètodes definits en l'estàndar HTTP.
- Schemaless: una base de dades és *schemaless* quan els registres o documents que formen una determinada col·lecció (o taula) no tenen una estructura rígida predefinida, i per tant cada un d'aquests documents o registres poden tenir diferents camps.
- Scrum: Metodologia àgil iterativa i incremental de desenvolupament de software.
- Single Page Application (SPA): Tipus d'aplicació web no basada en la càrrega de pàgines individuals a partir de diferents URLs, sinó que carrega tot el codi de l'aplicació de cop i va actualitzant la presentació en funció de les interaccions de l'usuari, només comunicant-se amb el servidor per carregar dades addicionals, generalment utilitzant AJAX.
- Slug: els *slug* són una part de les URL, comprensible per als humans, utilitzades per a millorar la usabilitat de les URL i per a millorar-ne el posicionament en cercadors.
- Source map: un *source-map* és un arxiu que permet revertir un procés de compressió o minimització per tal de fer llegible de nou el codi font per al procés de *debug.* En el desenvolupament web és habitual optimitzar els arxius Javascript i CSS mitjançant diversos processos de concatenació, supressió dels espais en blanc i salts de línia, i altres tipus de tècniques de compressió. Al generar un *source map* durant aquest procés, es possible identificar exactament en quin punt del codi original es produeix un determinat problema, encara que l'arxiu carregat en el navegador tingui una estructura completament diferent.
- Stylus: llenguatge de fulles d'estils, compilable a CSS, inspirat en d'altres llenguatges com Sass o LESS.

- Transpilació: procés mitjançant el qual el codi font escrit en un determinat llenguatge es transforma a un altre llenguatge amb un nivell general d'abstracció, en contraposició a la compilació, on el codi es transforma a un llenguatge amb un nivell d'abstracció molt diferent (per exemple de C++ a *codi màquina*)
- Tiquet: notificació d'un problema, incidència o dubte, creada per un usuari, i assignada a un agent responsable de resoldre el problema.

Annex B: Lliurables del projecte

Junt a la memòria, s'adjunten els següents fitxers:

- **Presentació del projecte, en diversos formats:**

  `presentacio/jcampsl-TFM-Presentacio.pptx`

  `presentacio/jcampsl-TFM-Presentacio.pdf`

- **Codi font de les aplicacions:**
  - API:

    `entregables/api`
  - *Helpdesk* (aplicació usuaris):

    `entregables/usersApp-src`
  - *Dashboard* (aplicació administradors i agents):

    `entregables/adminApp-src`

- **Aplicacions client compilades:**
  - *Helpdesk:*

    `entregables/usersApp-dist`
  - *Dashboard:*

    `entregables/adminApp-dist`

Annex C: Captures de pantalla

Captures de pantalla tant del producte/servei/aplicació realitzat així com del procés de treball. Aquest annex també pot utilitzar-se per recopilar les captures mostrades en altres seccions, en major grandària per a la seva millor visualització, o no ser necessari el seu ús pel tipus de treball realitzat.

C.1 Helpdesk

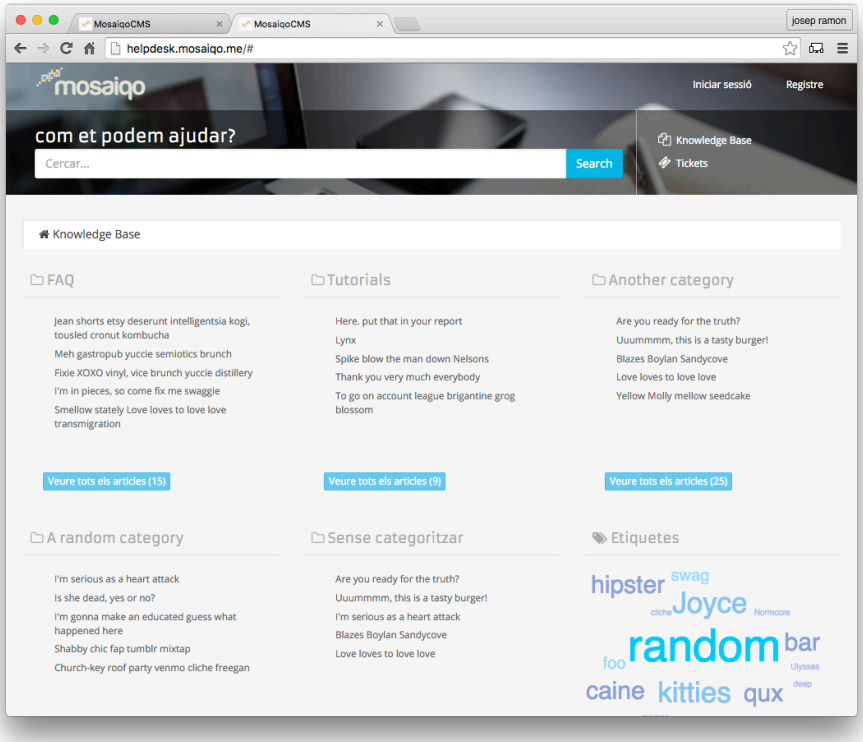


Figura 29 - Helpdesk: home

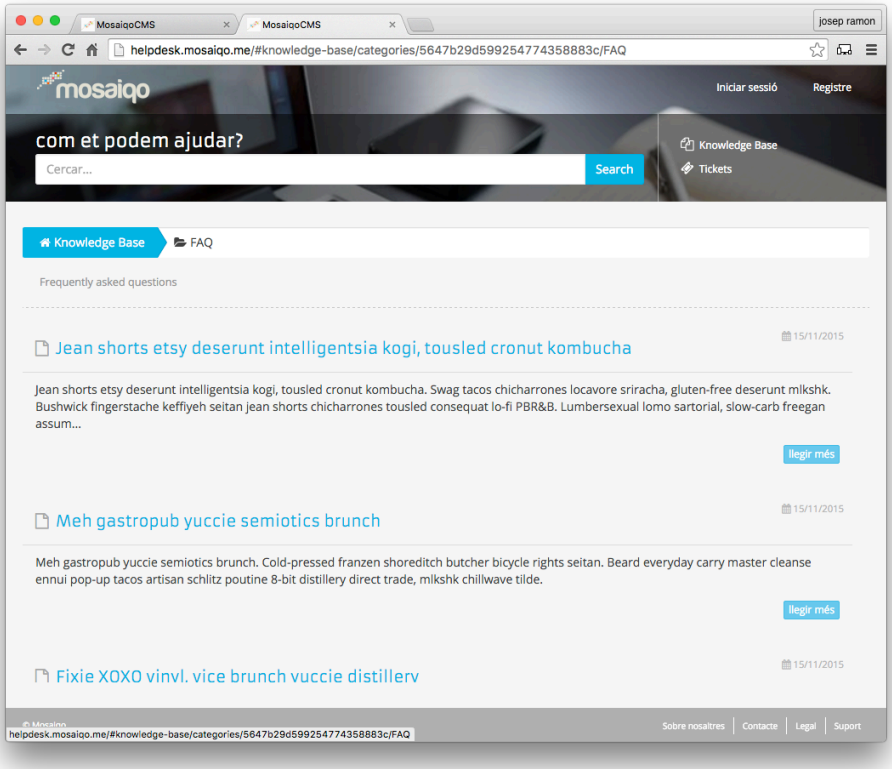


Figura 30 - Helpdesk: Articles per categoría

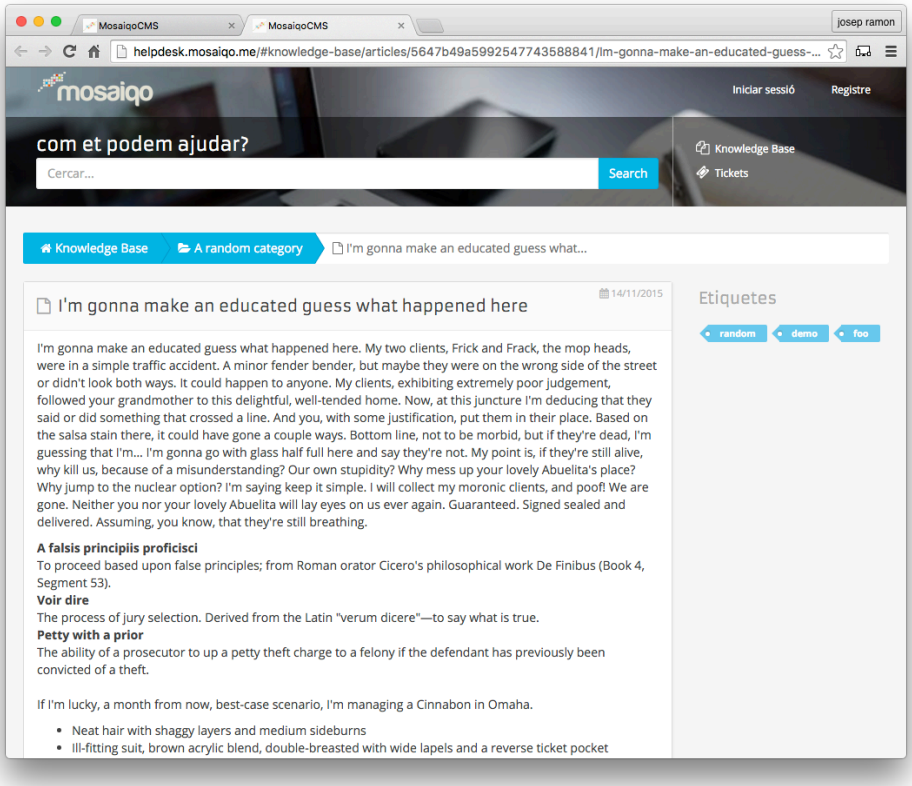


Figura 31 - Helpdesk: Detall article

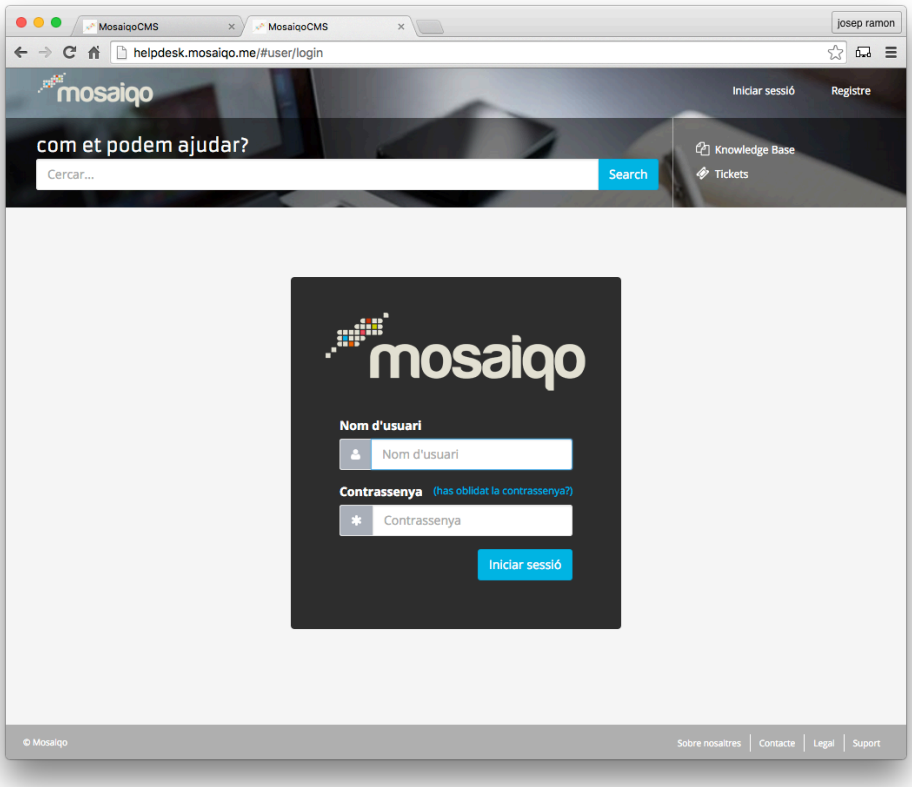


Figura 32 - Helpdesk: Inici sessió

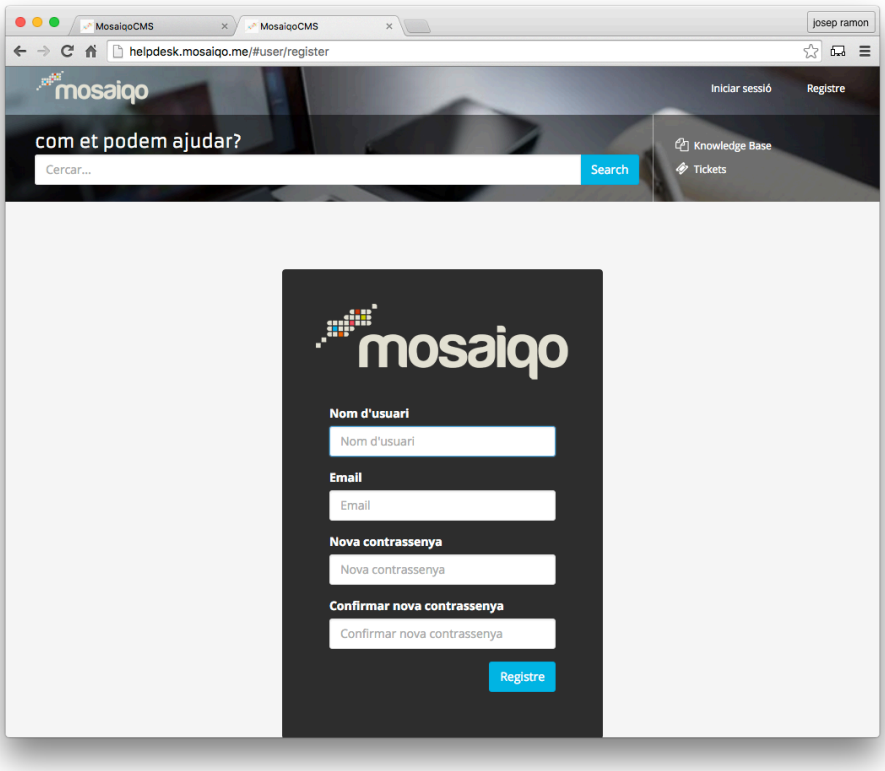


Figura 33 - Helpdesk: Registre

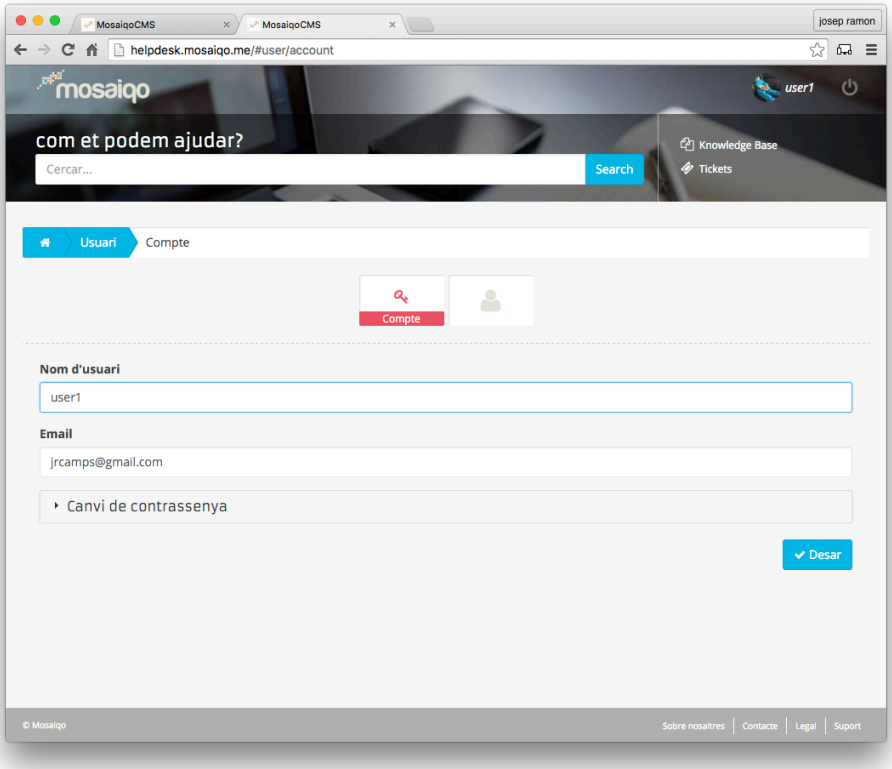


Figura 34 - Helpdesk: Detalls usuari

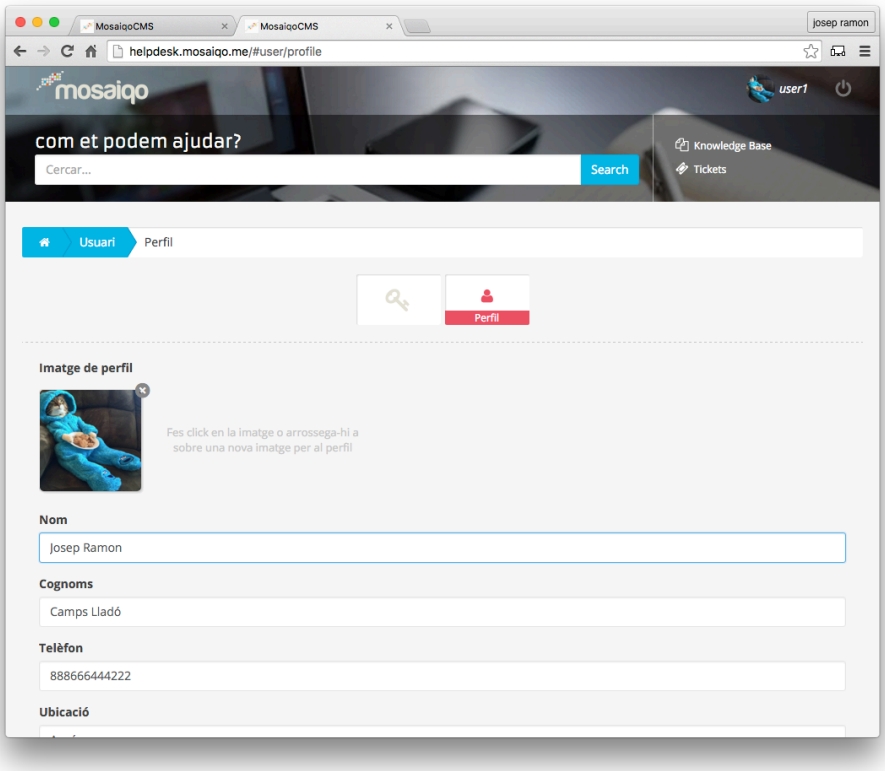


Figura 35 - Helpdesk: Perfil

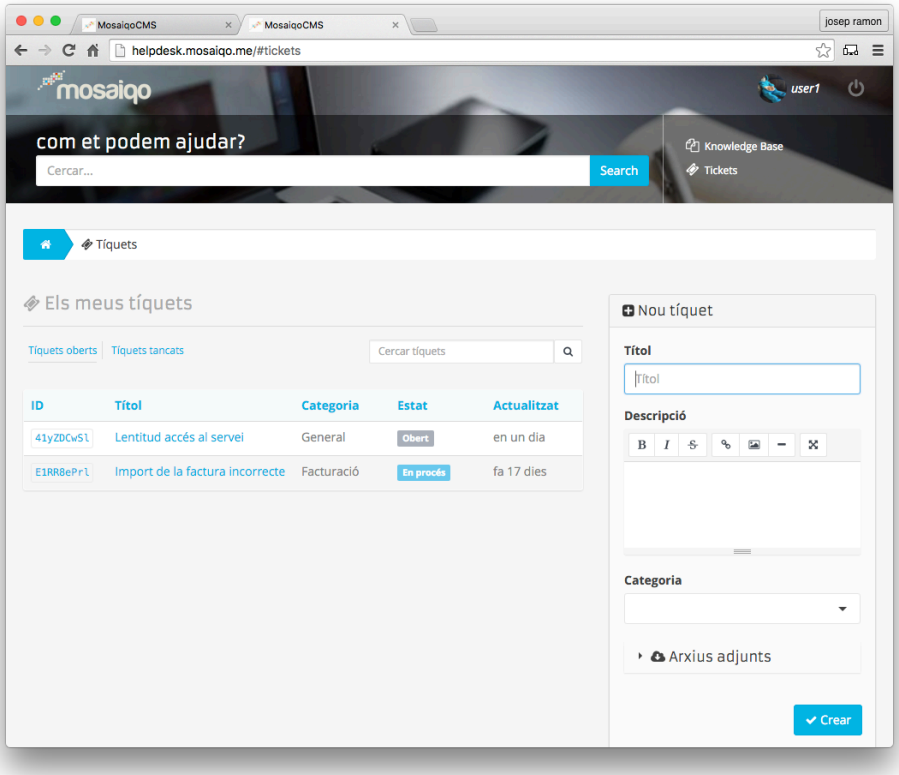


Figura 36 - Helpdesk: Llistat tiquets

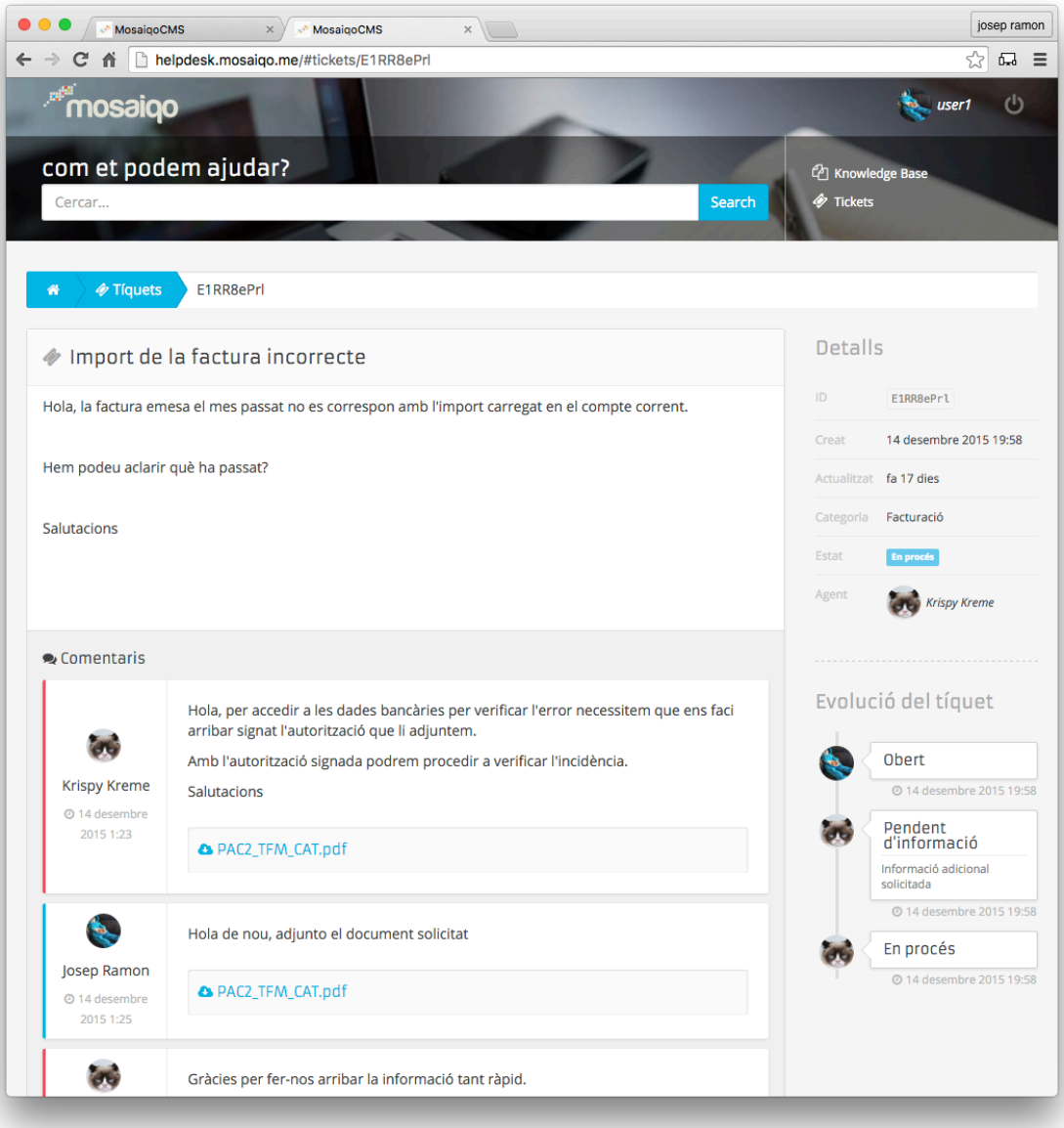


Figura 37 - Helpdesk: Detall tiquet

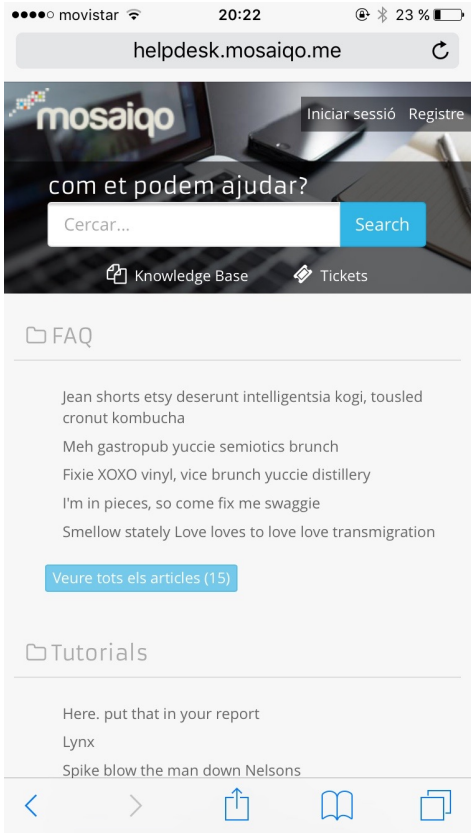


Figura 38 - Helpdesk: Home (mòbil)

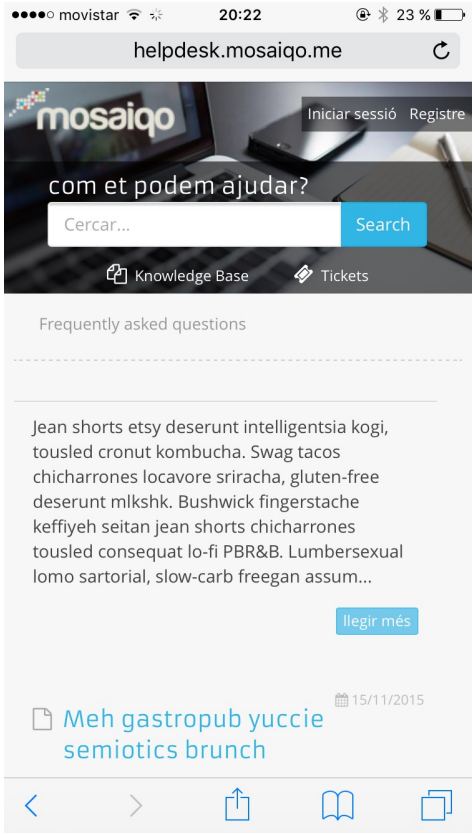


Figura 39 - Helpdesk: Articles per categoria (mòbil)

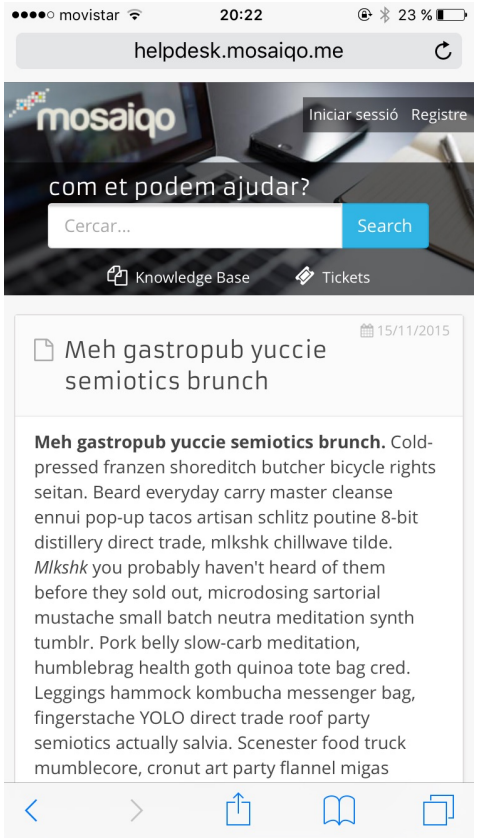


Figura 40 - Helpdesk: Detall article (mòbil)

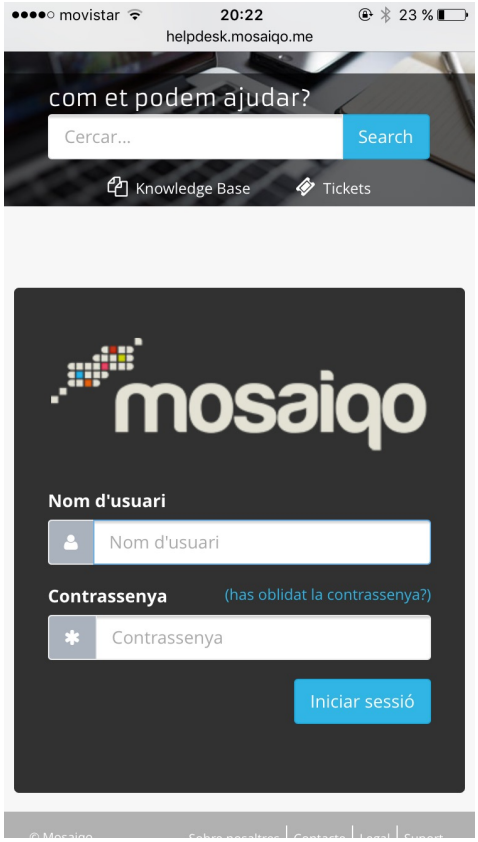


Figura 41- Helpdesk: Inici sessió (mòbil)

Figura 42 - Helpdesk: Registre (mòbil)

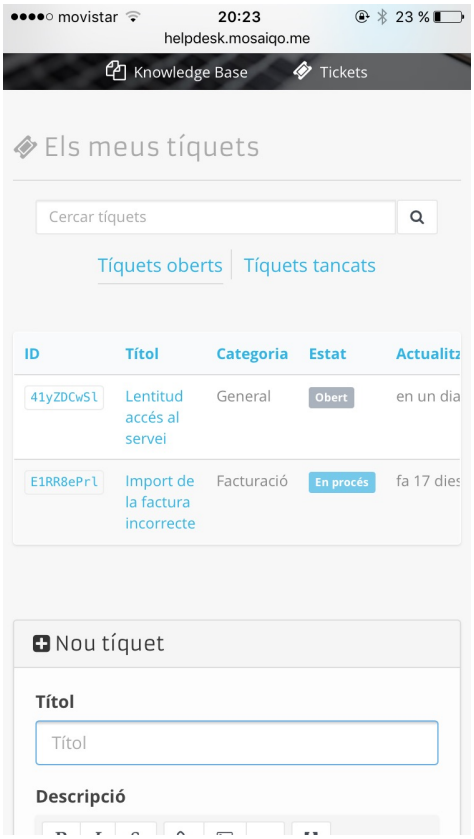


Figura 43 - Helpdesk: Llistat tiquets (mòbil)

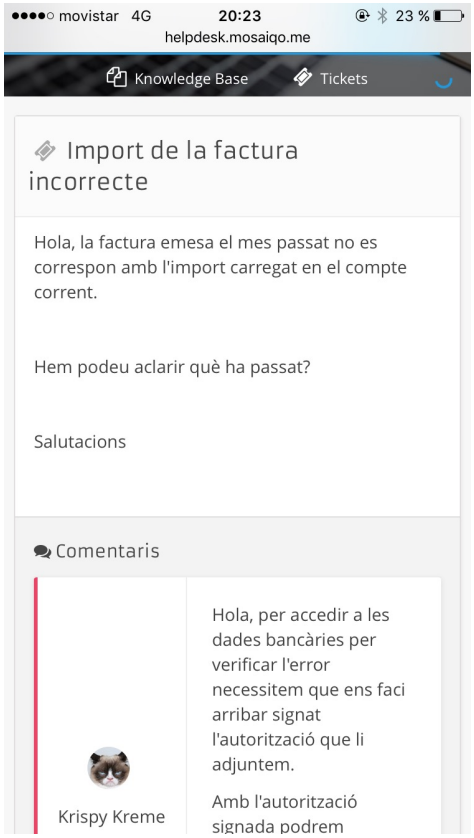


Figura 44 - Helpdesk: Detall tiquet (mòbil)

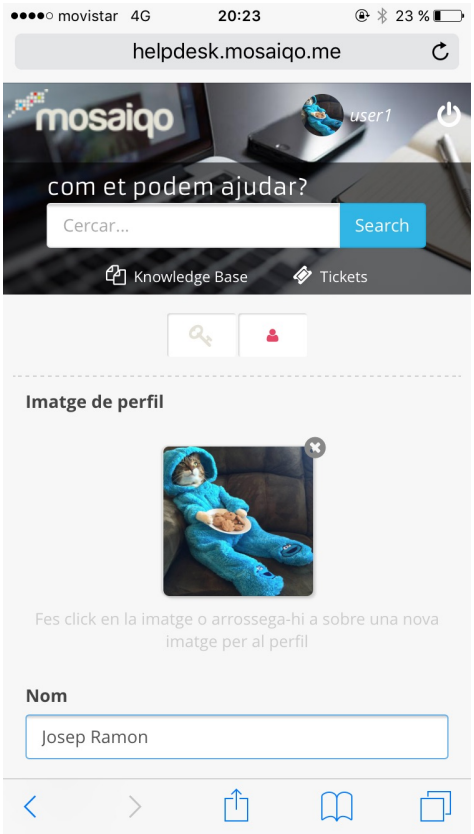


Figura 45 - Helpdesk: Perfil (mòbil)

C.2 Dashboard

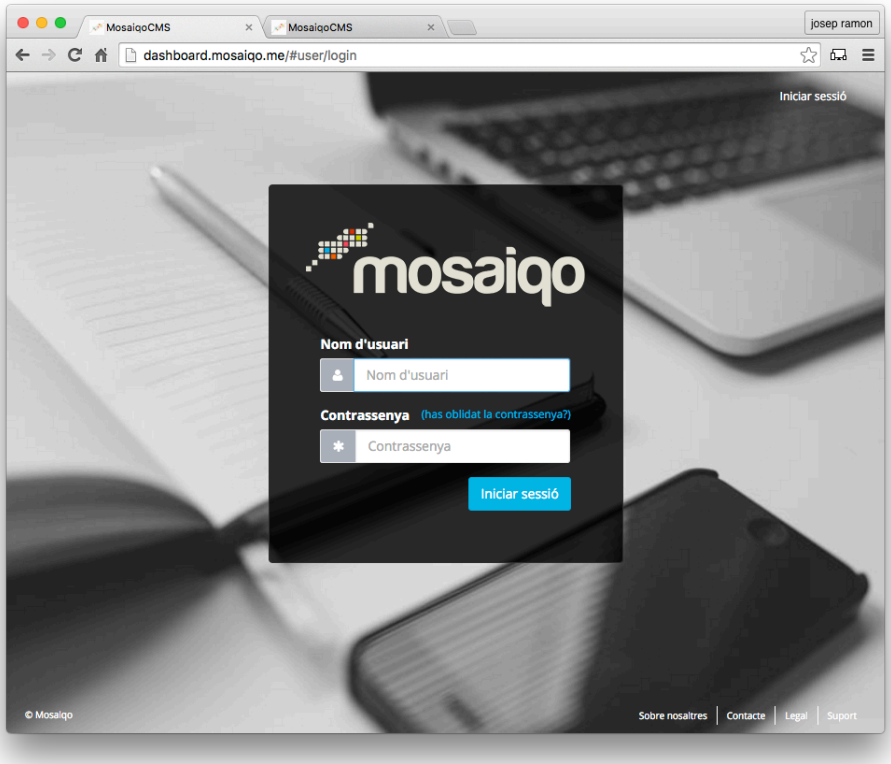


Figura 46 - Dashboard: Inici sessió

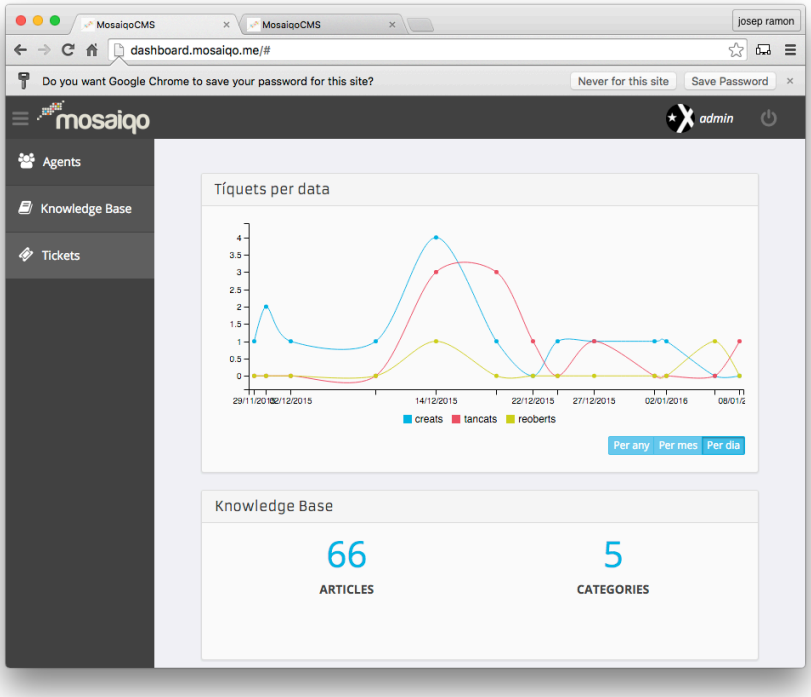


Figura 47 - Dashboard: Home

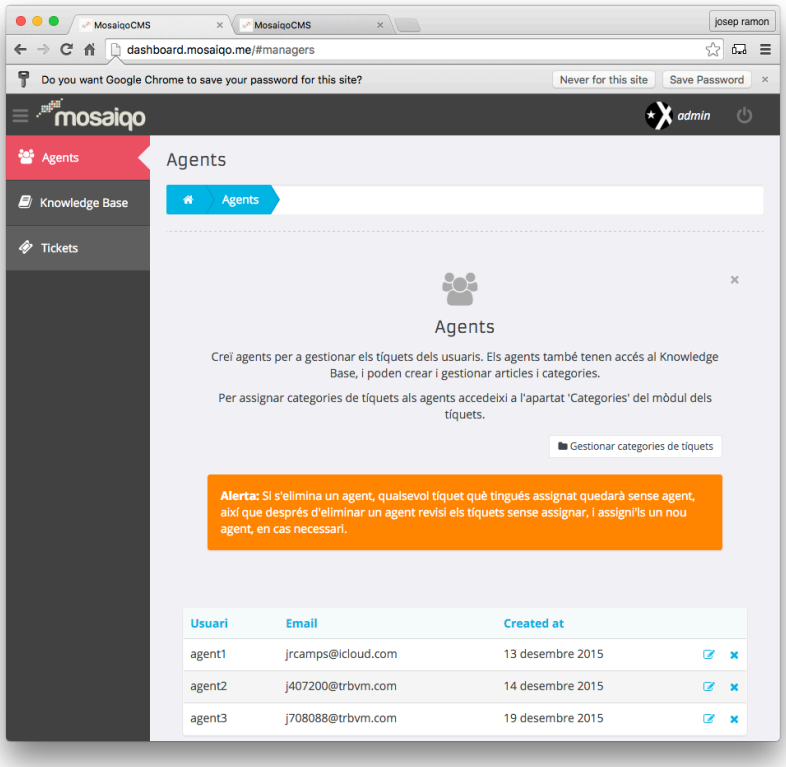


Figura 48 - Dashboard: Gestió agents

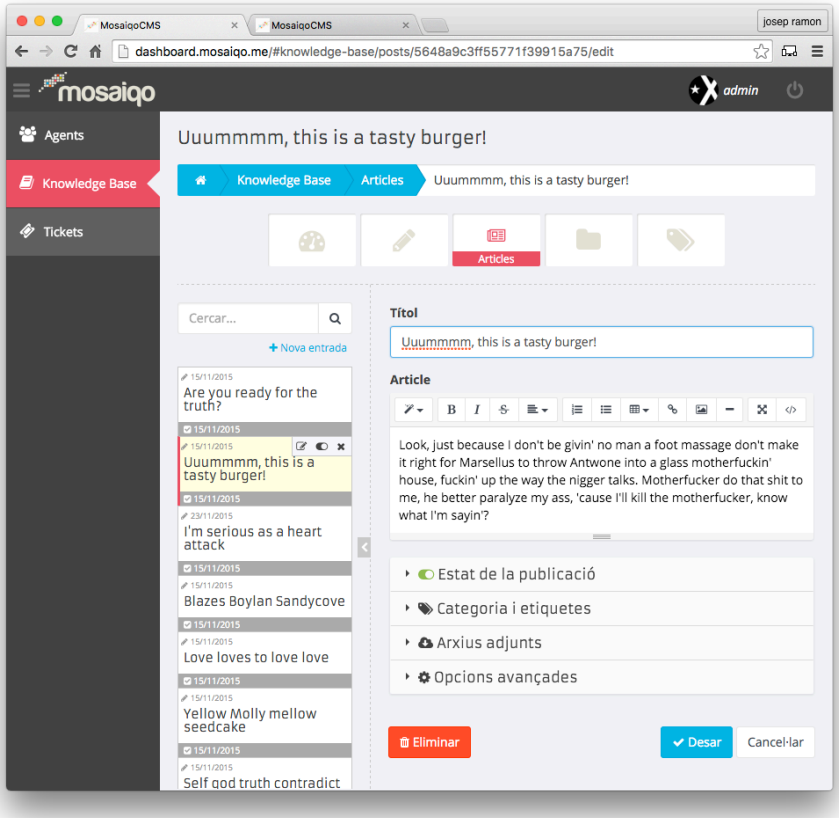


Figura 49 - Dashboard: Knowledge base – articles

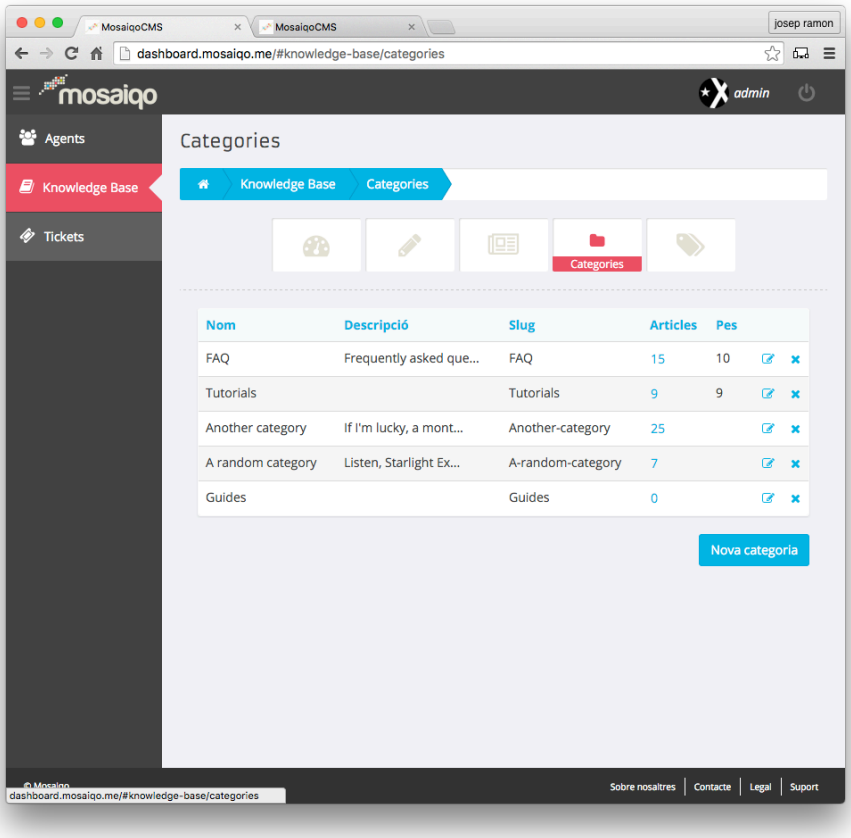


Figura 50 - Dashboard: Knowledge base - Gestió categories

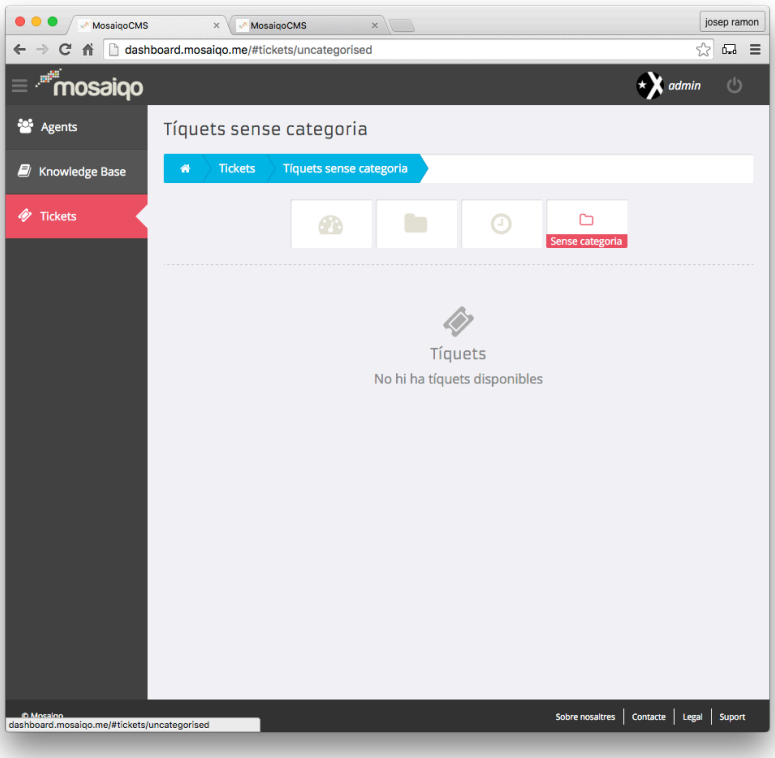


Figura 51 - Dashboard: Gestió tíquets (buit)

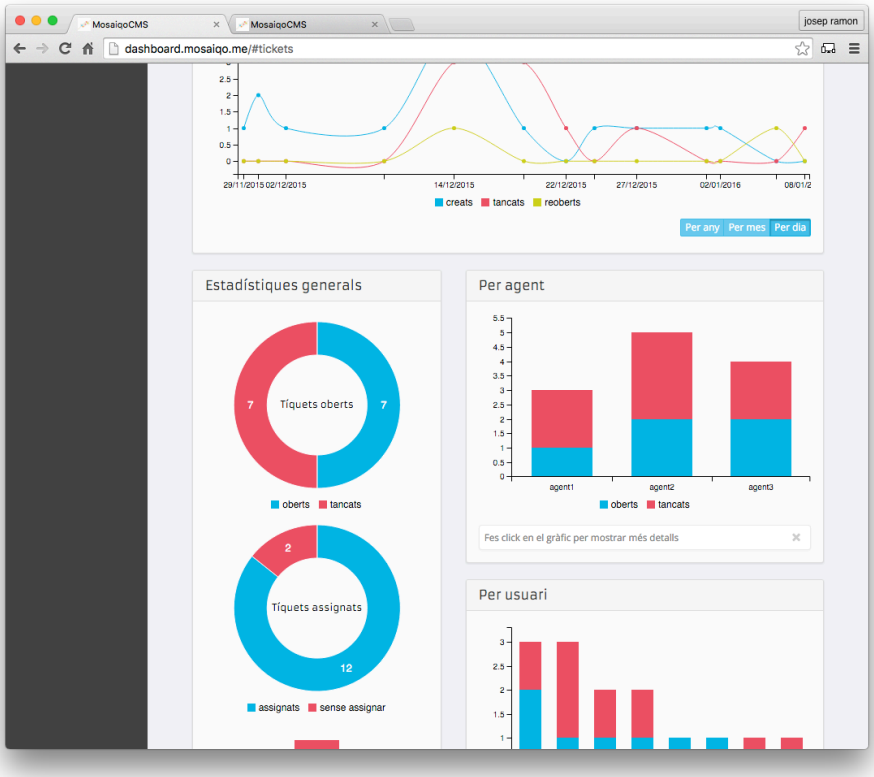


Figura 52 - Dashboard: Tiquets – estadístiques

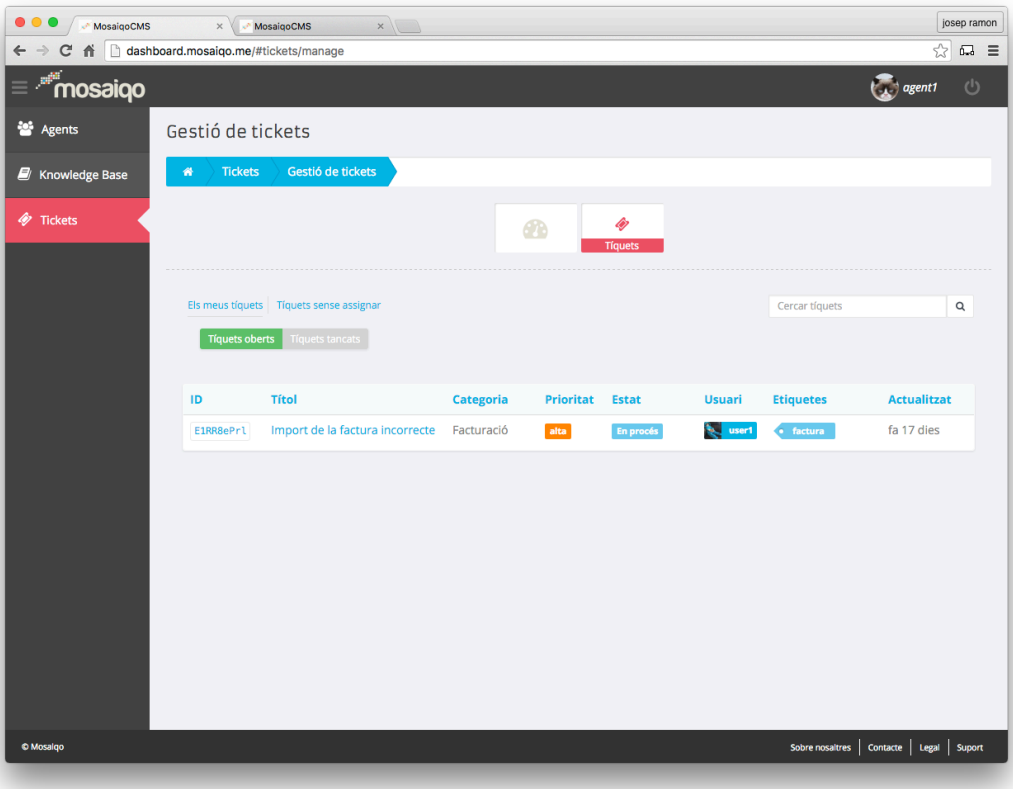


Figura 53 - Dashboard: Llistat tiquets assignats

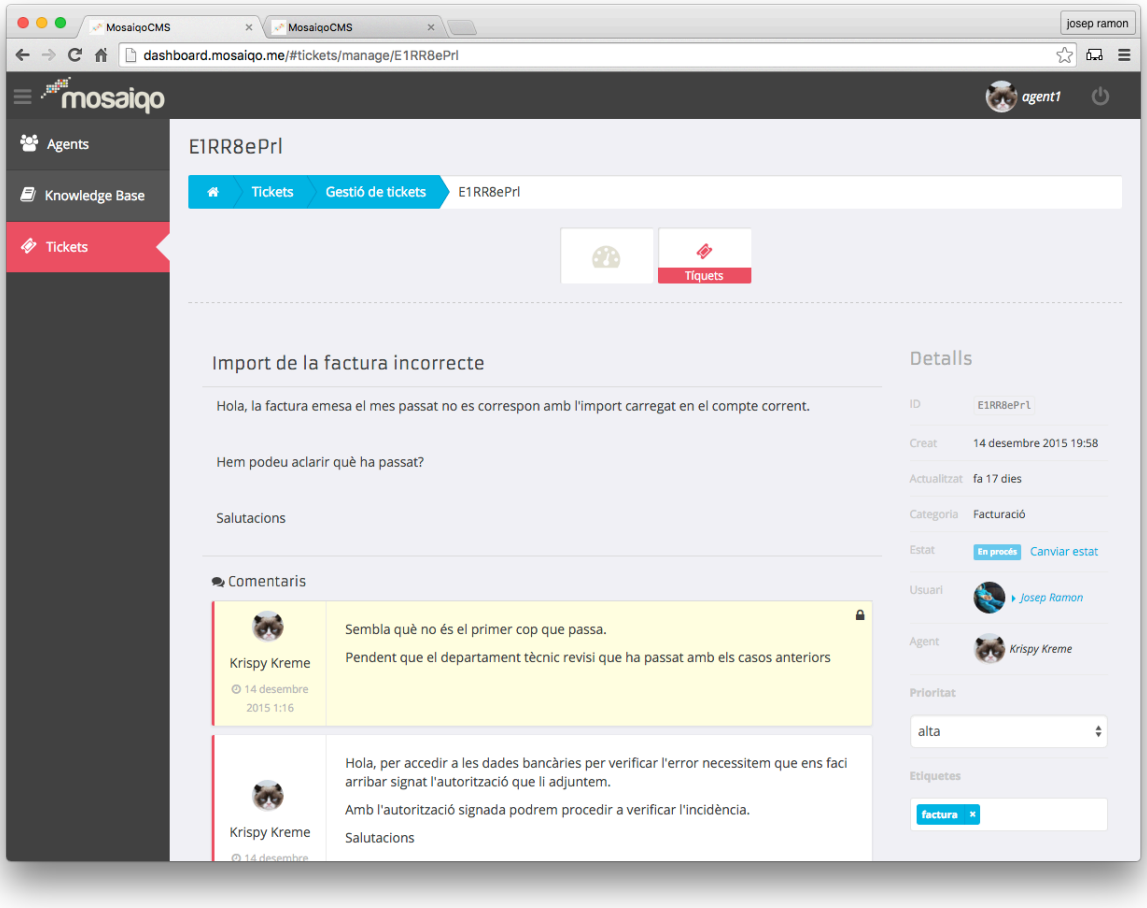


Figura 54 - Dashboard: Tiquets – Detall

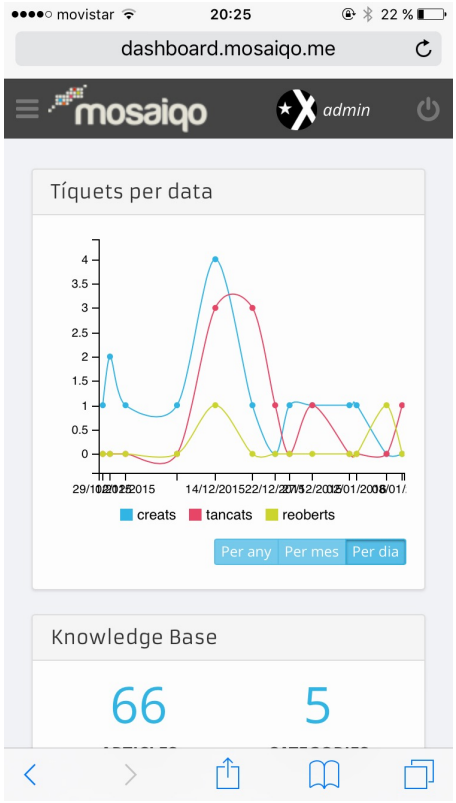


Figura 55 - Dashboard: Home (mòbil)

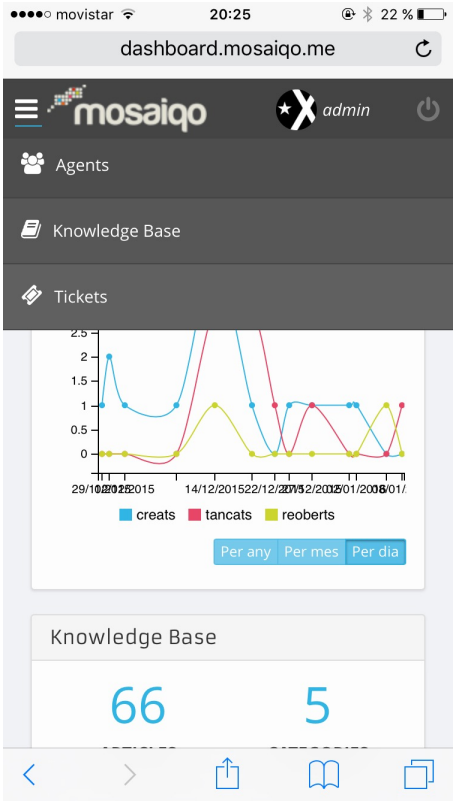


Figura 56 - Dashboard: Menu (mòbil)

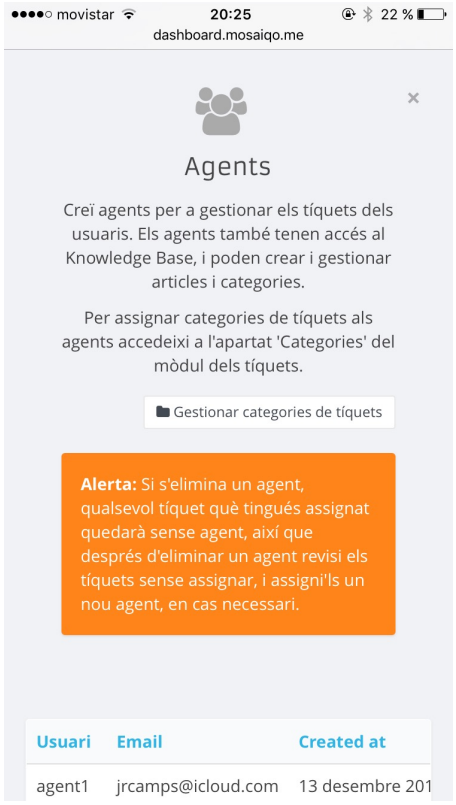


Figura 57 - Dashboard: Gestió agents (mòbil)

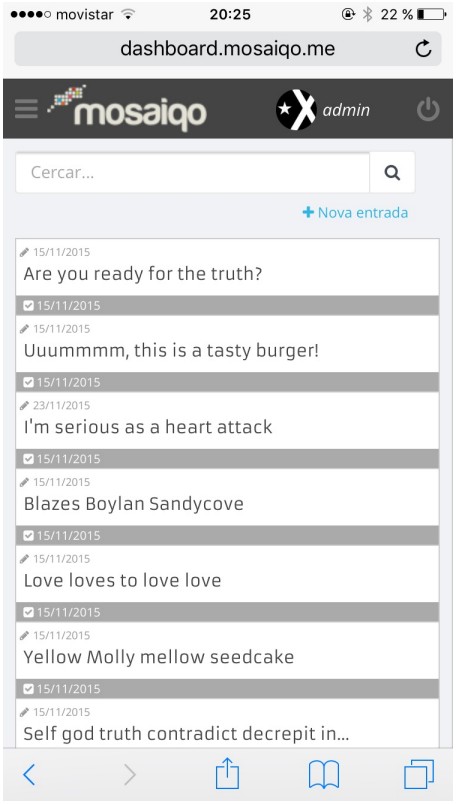


Figura 58 - Dashboard: Knowledge base: Llistat articles (mòbil)

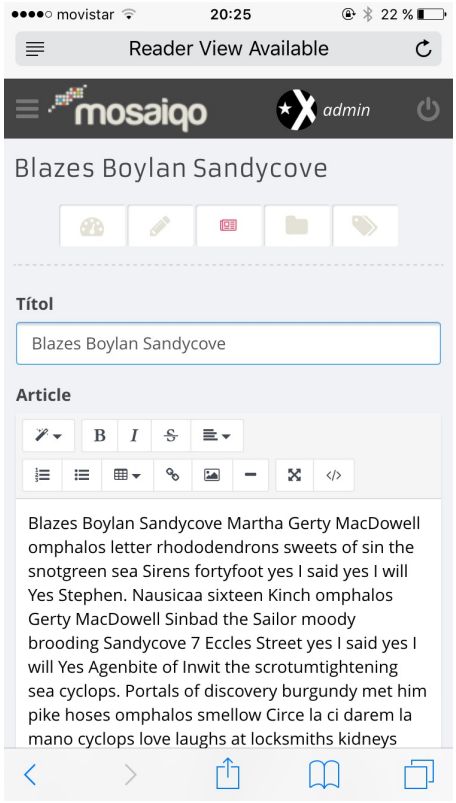


Figura 59 - Dashboard: Knowledge base: Edició article (mòbil)

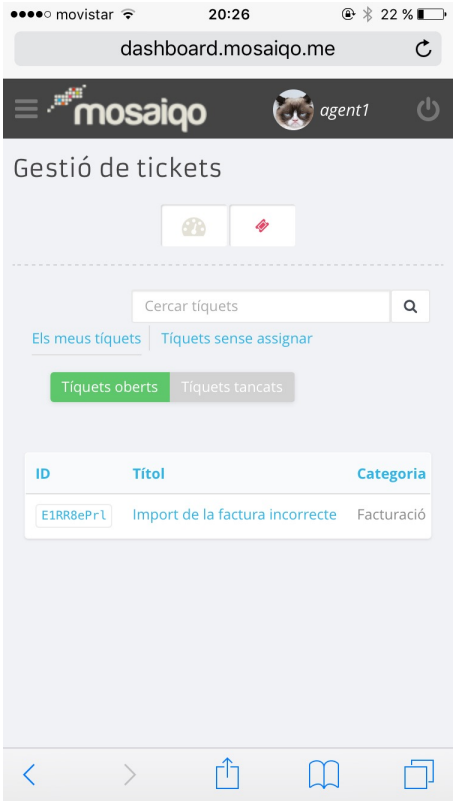


Figura 60 - Dashboard: Llistat tiquets assignats (mòbil)

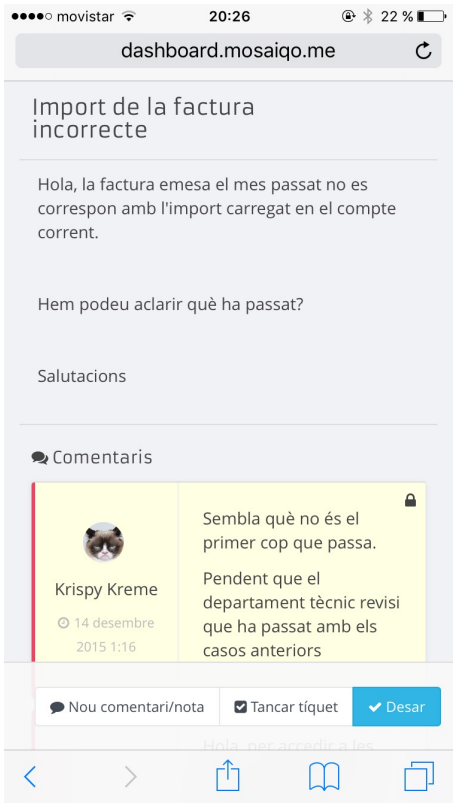


Figura 61 - Dashboard: Tiquets: Detall (mòbil)

C.3 Procés de treball

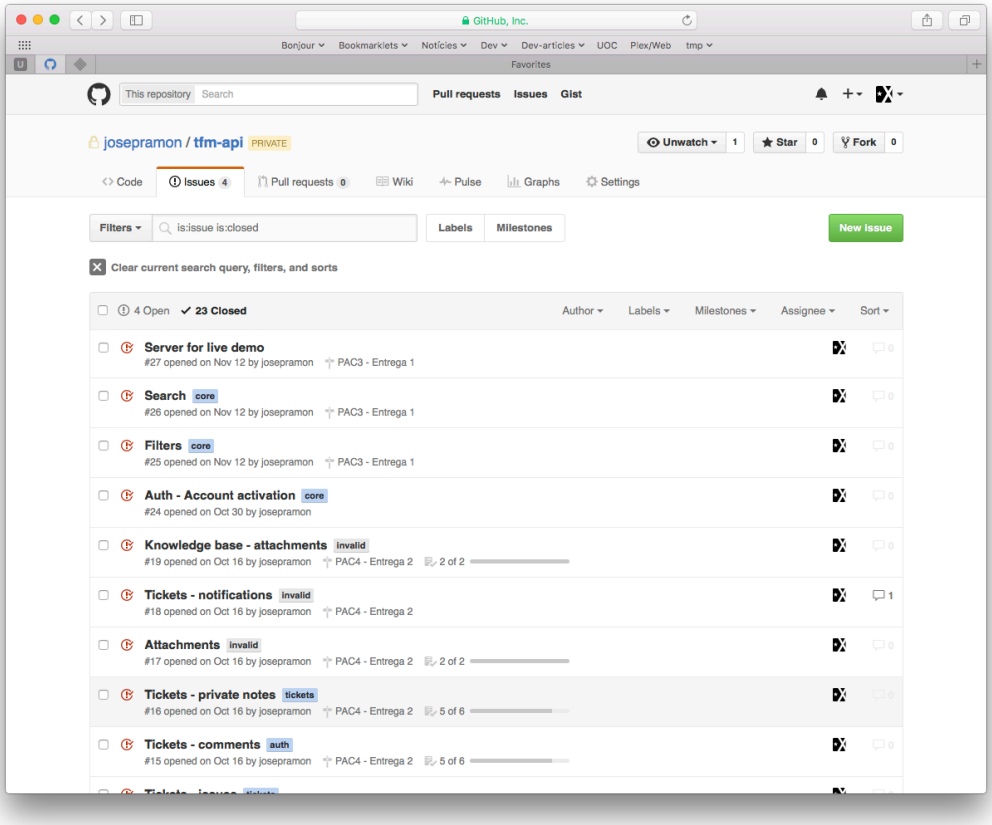


Figura 62 - Github: Issues en un dels repositoris del projecte

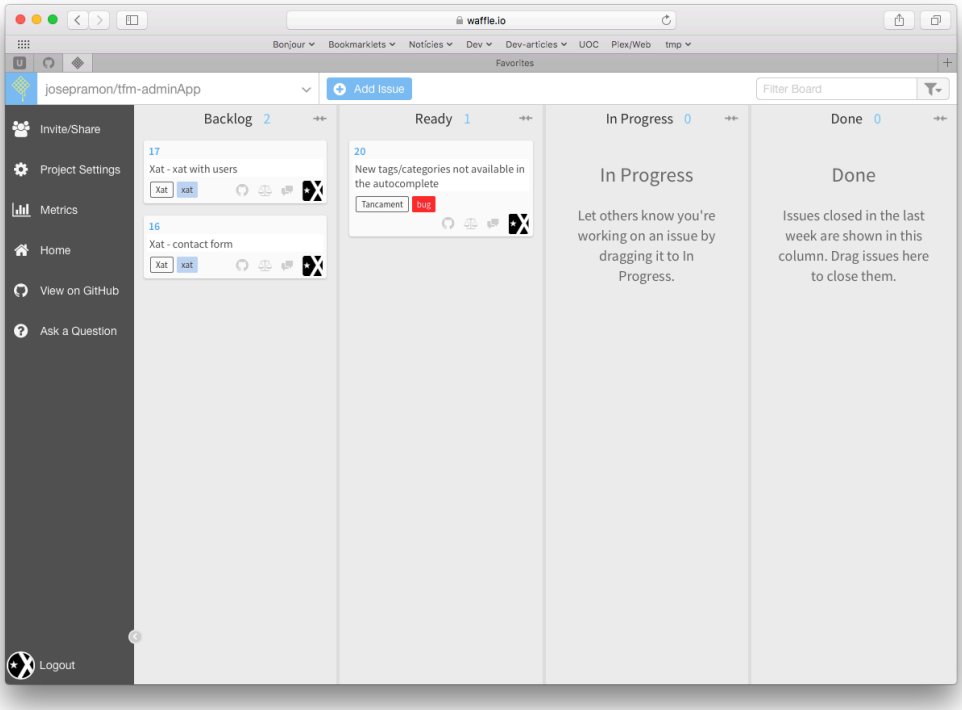


Figura 63 - Scrum board d'un dels components del sistema

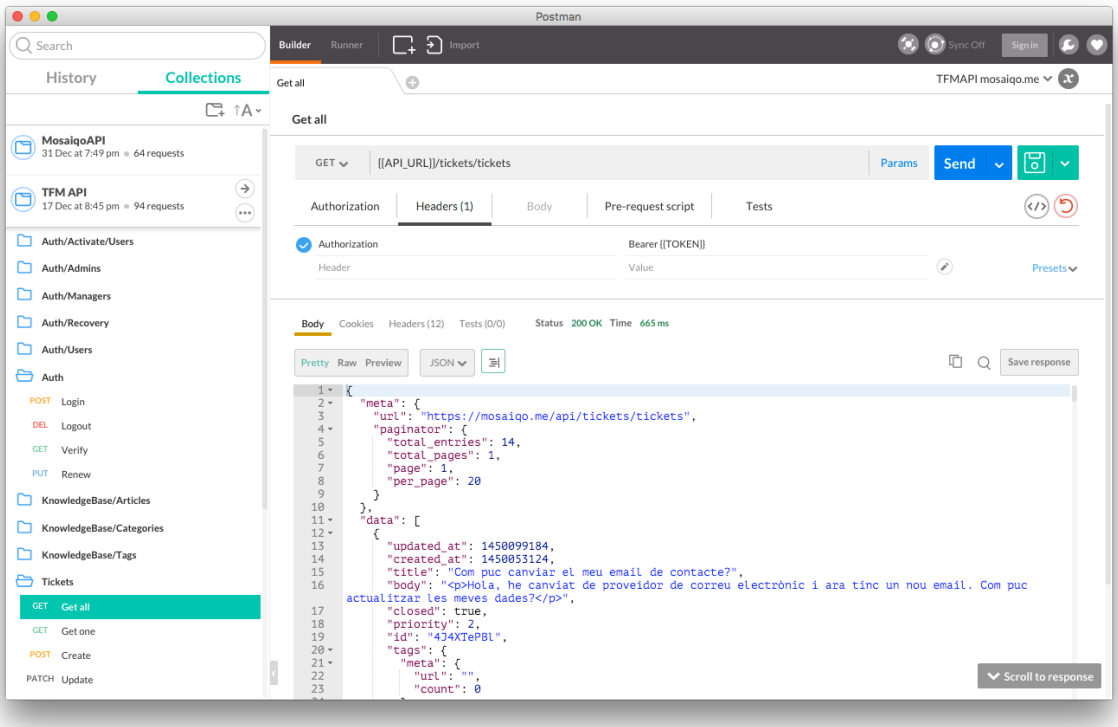


Figura 64 - Testeig dels paràmetres i resposta de la API amb Postman

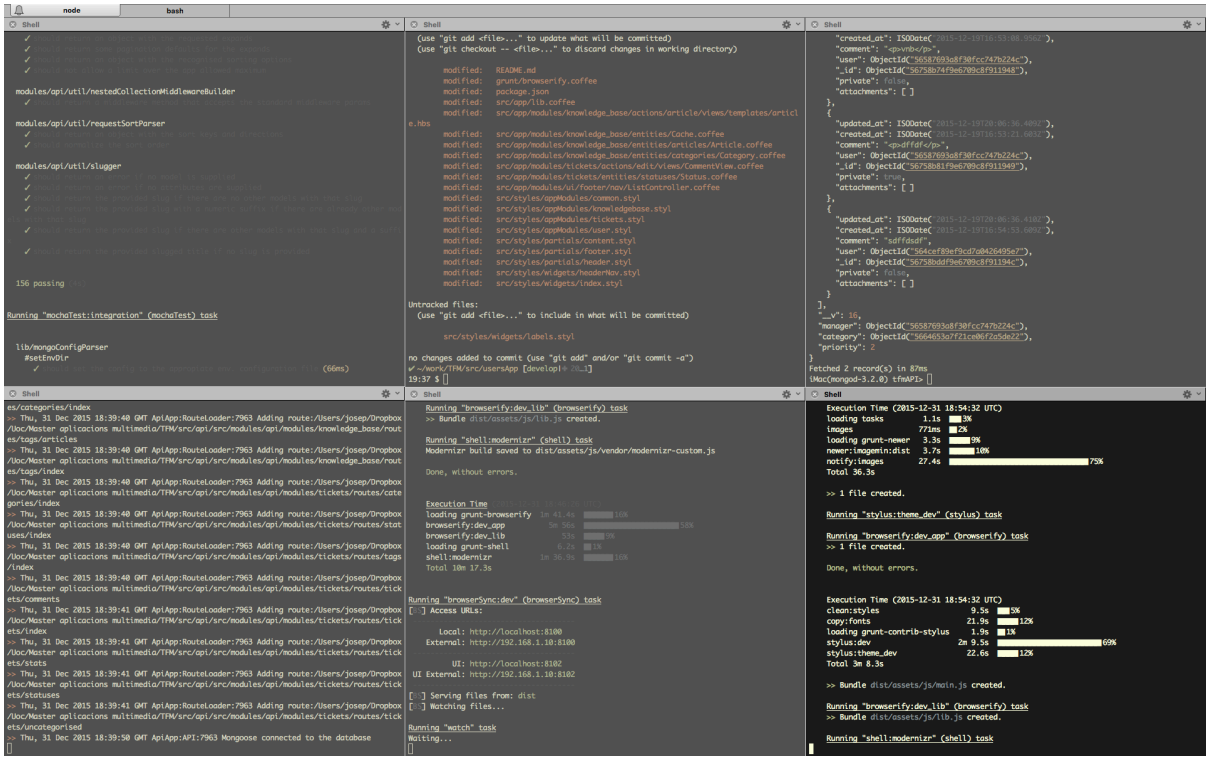


Figura 65 - Dev. aplicacions (monitorització i execució automatitzada de tasques com la compilació o execució de tests)

Annex D: Currículum Vitae

Josep Ramon Camps Lladó (Tàrrega, 1979)

Front end web developer, although i don't limit myself to just front-end, and also love from time to time to write some backend code, preferably in Node, but also in PHP, but where i really enjoy performing my work is writing clean, semantic and standards compliant front-end code, with a strong focus on the performance, scalability and the inner beauty of the code, and also on other aspects like the usability and accessibility in order to provide a great user experience.

I love learning new technologies and working with other people who love their work, in challenging and motivating projects.

https://github.com/josepramon
https://github.com/mosaiqo
https://www.linkedin.com/in/jrcamps

Education

- Multimedia applications: design and development of smart content Master's degree. Universitat Oberta de Catalunya (UOC) (2014-Present)
- Grau en Multimedia (Multimedia bachelor's degree). Universitat Oberta de Catalunya (UOC) (2008-2010)
- Graduat Multimèdia. Undergraduate degree from Universitat Politècnica de Catalunya (UPC) and Universitat Oberta de Catalunya (UOC). (2000-2006)
- Workshop: Network Advertising. Universitat Oberta de Catalunya (UOC) (2003)
- Workshop: Technological resources at the service of a virtual community. Universitat Oberta de Catalunya (UOC) (2003)
- Workshop: Net.art experiments. Universitat Oberta de Catalunya (UOC) (2002)
- Workshop: 3d virtual environments team building with 3D Studio Max MAXScript. Universitat Oberta de Catalunya (UOC) (2001)
- Degree in Psychology, University of Barcelona (UB) (1998-2000) (unfinished studies)
- Multitude of online courses on multimedia and programming. (2001-Present)