Aprenent des de la pràctica Symfony2, aplicant mètodes i pràctiques àgils per construir SegundoUso.org

Memòria de Projecte Final de Diplomatura

Enginyeria Tècnica en Informàtica de Gestió

Desenvolupament Web

Autor: Javier Seixas Pérez

Consultor: Ignasi Lorente Puchades

09/01/2014

Crèdits/Copyright

Abstract

El present es divideix en dos grans blocs. Per una banda està el bloc tècnic i per l'altre el bloc de negoci. Aquest darrer té com a objectiu el desenvolupament d'una plataforma web que esdevingui un espai on les persones puguin oferir objectes que ja no utilitzen a altres persones, amb la intenció de fomentar o facilitar la reutilització en la societat. Pel que fa el bloc tècnic, es subdivideix en dos grans àrees. La primera es basa en l'aprenentatge del framework de PHP Symfony2. La segona pretén posar en pràctica un seguit de mètodes àgils molt interessants de conèixer professionalment, però que no s'ha tingut la oportunitat de posar en pràctica durant els estudis. A conseqüència d'això, també s'aprendrà la utilització del framework de PHP Behat, enfocat a la pràctica àgil de BDD (*Behaviour Driven Development*).

Al llarg d'aquest document es posarà de manifest tots els coneixements adquirits en aquestes noves tècniques, així com exemples de l'aprenentatge pràctic que s'ha dut a terme. Sense cap mena de dubte aquest projecte ha sigut molt enriquidor a nivell personal, tan pel fet de poder conèixer un framework reconegut dins de l'ecosistema PHP com és Symfony2, com per la oportunitat de poder posar en pràctica mètodes àgils, i la aplicació d'una eina tan útil com és el BDD.

Notacions i Convencions

Per la bona entesa del contingut d'aquest document s'ha creat un seguit de notacions que ajudin a la comprensió del contingut.

El lector trobarà diferents blocs negres com el l'exposat a continuació. Aquests quadres contenen codi font d'arxius del projecte o codi simulat a mode d'exemple.

```php
<?php

namespace SegundoUso\ExempleBundle\Controller

class ExempleController extends Controller
{
    public function indexAction()
    {
        $manager = $this->get('seguso.ad_manager');

        return $this->render('SegundoUsoExempleBundle:Exemple:index.html.twig');
    }
}
```

També es poden trobar dintre dels paràgrafs del cos del document paraules escrites en una tipografia com aquesta que fan referència a codi que apareix en un bloc negre anterior o posterior. Així doncs si s'hagués d'esmentar la funció de l'exemple anterior es referiria a ella com `indexAction()`.

A més, al document hi apareixeran instruccions escrites en terminal. Aquestes estaran incloses en blocs de color gris, per facilitar la diferenciació del blocs de codi de color negre.

```
sudo a2ensite segundo
sudo service apache2 reload
```

Índex

1. Introducció

El present Projecte de Final de Carrera parteix en dos grans inquietuds personals. El primer i més important és donar-me la oportunitat de programar utilitzant eines i metodologies noves, no estudiades durant la carrera però si molt demandades a nivell laboral dintre del sector de desenvolupament web en PHP. El segon bloc correspon a la creació d'una web d'ús social que pogués col·laborar en la construcció d'una societat més sostenible. La temàtica de la web és la d'un portal on les persones puguin oferir de manera gratuïta objectes que ja no utilitzen per a que altres persones interessades puguin quedar-se'ls i donar-hi un segon ús.

1.1 Justificació del projecte

Justificar la creació de SegundoUso.org és justificar el perquè dels dos grans motius esmentats anteriorment: l'aprenentatge tècnic i la creació d'una plataforma d'interès social.

1.1.1 Aprenentatge tècnic

Durant els estudis a la UOC s'estudia principalment el desenvolupament en cascada com a mètode de desenvolupament, on es planteja un projecte en un seguit de fases: requisits, planificació, disseny, desenvolupament, proves i manteniment. Aquest és un procés que data dels anys 70 i que diferents moviments han considerat, i consideren, poc adient pel desenvolupament de *software*, i plantegen altres alternatives. Durant els anys 90 va aparèixer el moviment *agile* que promovia un altre enfoc dels projectes proposant una altra forma de gestionar-los i programar-los. Amb aquest moviment va néixer l'*Extreme Programming.* A dia d'avui, moltes empreses consideren aquesta metodologia la més òptima, i apliquen els seus principis en més o menys mesura dins del procés de desenvolupament.

A partir d'aquesta realitat vaig considerar interessant posar en pràctica aquestes metodologies que fins ara, ni durant els estudis ni a nivell laboral, no havia tingut la oportunitat d'aplicar. Una motivació per això és que com s'esmentava anteriorment, moltes empreses demanen coneixements en aquestes mètodes. A partir d'aquí la decisió va ésser intentar focalitzar-se en tres de les més conegudes, diferents però complementàries: SCRUM, Ex*treme Programming* I *Behaviour Driven Development* (BDD).

També a nivell tècnic, deixant de banda les metodologies, resulta interessant tenir la oportunitat de treballar amb eines que també són demandades a nivell laboral. Una de les més interessants actualment dins de l'ecosistema de PHP és el framework Symfony2, possiblement un dels més complerts I avançats. També són demanades altres eines com Behat, un framework de test adaptat a la pràctica de BDD.

1.1.2 Creació d'una web per obra social

A nivell personal sempre he tingut molta inquietud per la ecologia i per la pràctica d'accions que ajudin a conservar i millorar el medi ambient. Actualment vivim en una societat consumista que abusa dels recursos del planeta. Part d'aquest abús és centra en la fabricació de tot tipus d'objectes que també es vincula a la societat consumista.

Com a veí de Barcelona setmanalment veig com es llencen munts de mobles, els quals moltes vegades poden ser aprofitats. Molt possiblement la reutilització d'aquests mobles evitaria compres de productes nous, reduint així el consum de recursos.

També en un context de crisi hi ha persones que ja no tenen l'oportunitat d'adquirir molts productes de necessitat, com pugui ser roba. Un portal d'aquestes característiques pot fomentar que persones amb pocs recursos puguin accedir de manera gratuïta al que necessiten.

Hi ha altres portals que compleixen la mateixa funció. El més conegut és freecycle, que es basa en la eina de Yahoo! Groups amb un important volum d'usuaris. Tot i que es pot considerar llavors que no existeix mercat per una web com SegundoUso.org, personalment considero que sí, perquè Yahoo! Groups no és una aplicació especialitzada per gestionar i emmagatzemar el tipus d'informació, que bàsicament són anuncis.

Per altra banda, també crec que crear un portal especialitzat pot permetre la facilitat d'ús de la eina, i per tant facilitar que la gent participi en iniciatives com aquesta.

1.2 Objectius

Com s'ha plantejat a l'inici d'aquesta introducció, els dos grans blocs entorn els quals gira aquest projecte, es poden considerar a l'hora els objectius principals del Projecte. Per aquest motiu s'han dividit els objectius en Tècnics i de Negoci.

1.2.1 Tècnics

Els objectius tècnics es basen principalment en l'ús d'unes determinades eines i mètodes denominats àgils.

- Desenvolupar l'aplicació a partir del framework Symfony2 de PHP i adquirir coneixements sobre ell mitjançant el seu ús.
- Aplicar arquitectura i bones pràctiques en el disseny
- Posar en pràctica el procés de desenvolupament BDD mitjançant la eina de PHP Behat.
- Desenvolupar el projecte fent servir tècniques de desenvolupament àgil com són Extreme Programming i Scrum, adaptant-les en lo possible al fet que el desenvolupament és unipersonal.

Cal esmentar que aquest TFC pretén encarar els mètodes i pràctiques àgils així com les diferents eines que s'empraran des d'un enfoc principalment pràctic.

1.2.2 De Negoci

Objectius de negoci en el TF.

- Crear una versió senzilla però funcional d'un portal que permeti publicar anuncis oferint objectes gratuïtament.
- Que el portal ofereixi la opció de participar-hi sense la necessitat de registrar-se, evitant el màxim elements que obstaculitzin la participació.

1.3 Eines utilitzades i justificació

L'elecció de les eines ha sigut determinada per l'ús i la demanda que aquestes tenen a nivell laboral, principalment a la ciutat de Barcelona, però també valorades a nivell mundial.

La principal eina utilitzada ha sigut el framework **Symfony2**. Symfony2 està 100% desenvolupat en PHP i utilitza el paradigma de la programació orientada a objectes, permetent distribuir diferents funcions en paquets independents, donant una gran flexibilitat al desenvolupament. Aquests paquets són els anomenats **Components** que permeten el funcionament de tot el sistema. També existeixen un altre tipus de components anomenats ***Bundles***. Un *Bundle* és un paquet que no és indispensable pel funcionament del framework, però que aporta una lògica de negoci que pot ser reutilitzada en diferents projectes. Symfony2 està considerat un dels tres frameworks més populars de PHP en l'actualitat i és utilitzat arreu del món. En l'apartat 3.1 s'aprofundeix sobre ell.

Behat és un altre framework utilitzat en el desenvolupament. És una eina enfocada a la creació de tests funcionals, i optimitzada per a fer-se servir en l'aplicació de BDD i amb mètodes àgils. Es basa en la implementació d'històries d'usuari a través del llenguatge **Gherkin**, que a transforma el flux descrit en la història en comprovacions lògiques sobre el navegador, i utilitzat amb altres frameworks de test com **Sahi** o **Selenium**, és possible validar el funcionament de la aplicació web de manera automàtica. En l'apartat 3.2 s'aprofundeix més sobre el funcionament de Behat.

Composer és una eina encarregada de la gestió de dependències. Configurant un simple arxiu i executant l'eina les terceres llibreries necessàries pel funcionament de l'aplicació són descarregades d'un repositori anomenat Packagist, que a l'hora també fa servir la plataforma Github per proveir-se. Composer simplifica molt l'ús i gestió de terceres llibreries.

Capifony és una eina desenvolupada amb el llenguatge Ruby i basada en Capistrano, que permet el desplegament automàtic de l'aplicació SegundoUso.org en l'entorn de producció. La necessitat d'ús

d'aquesta eina ve de la complexitat de desplegar el framework Symfony2 amb totes les seves llibreries i amb l'ús de Composer. Capifony s'encarrega d'aquesta tasca mitjançant una sola comanda.

L'eina triada per gestionar el control de versions ha sigut **Git**. Els motius de la tria són la seva creixent popularitat, la vinculació d'una plataforma gratuïta com Github on poder mantenir un repositori online, i la compatibilitat amb la eina Capifony per dur a terme el desplegament.

2. Mètodes i pràctiques àgils aplicades

El mètodes i pràctiques aplicats en aquest projecte són una part del que és conegut com **Desenvolupament àgil de programari**. Aquest és un grup de mètodes que premien la flexibilitat en l'evolució del projecte donant per suposat que aquesta no és pot predir a llarg plaç en un projecte de programari, degut a tots els factors que intervenen: tecnologies, clients, el propi equip de desenvolupadors, etc. Dintre del desenvolupament àgil hi ha moltes tècniques (XP, Scrum, entre d'altres) que en termes generals s'enfoquen en dividir els projectes en iteracions de poques setmanes on a cadascuna es construirà una part de l'aplicació final. Per altra banda, els desenvolupadors es dividiran en equips petits (8-10 persones). Aquestes dues característiques, evitant períodes de desenvolupament llargs i equips grans, intenten simplificar, i per tant optimitzar, la gestió de d'equips i projectes.

Una de les diferències més importants que tenen aquestes metodologies respecte de les clàssiques (desenvolupament en cascada) és que promouen la comunicació entre l'equip en comptes de la documentació del projecte (diagrames UML, documents de requeriments o d'estimació de temps). Per la banda de l'elaboració de diagrames, amb l'*Agile* s'intenta evitar la inversió de temps que suposa redactar la documentació d'un projecte informàtic, i de mantenir-la actualitzada a mesura que van havent canvis. De fet, segons els principis de l'XP, el codi de l'aplicació ha de ser prou **net** i llegible com per que per si mateix sigui la documentació de l'aplicació.

2.1 Mètodes àgils utilitzats

Existeixen gran multitud de mètodes àgils, dels quals en aquest projecte s'ha volgut aplicar principalment dos: Scrum, XP, a més d'una pràctica que pren molt de protagonisme durant el desenvolupament: BDD. Els pròxims paràgrafs pretenen ser una breu introducció a aquests mètodes i pràctica.

2.1.1 Scrum

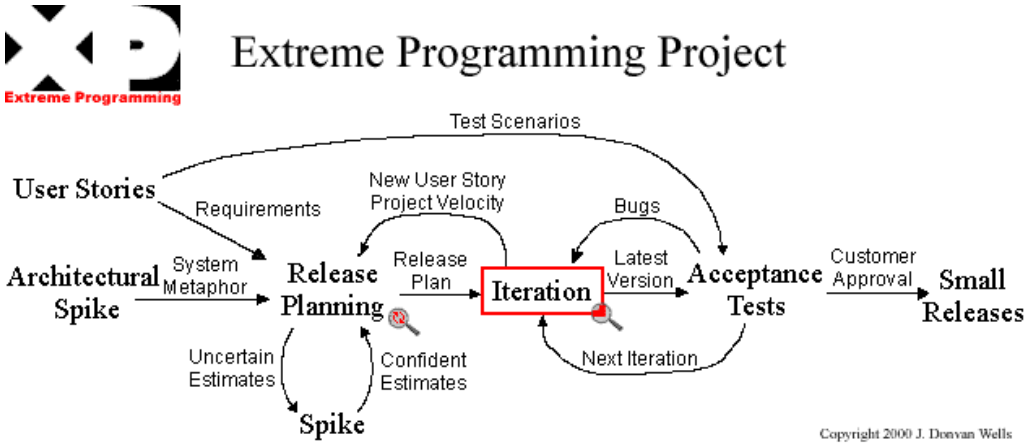
Scrum és un mètode enfocat la gestió projecte desenvolupat de manera àgil. Emfatitza la comunicació interna de l'equip, però també la importància de la implicació del client en aquesta. L'Scrum s'encarrega de definir una sèrie de rols entre les diferents persones participants en el projecte i promou a seva dinàmica. Els rols són l'equip de desenvolupament, el P*roduct Owner* (PO), que correspondria al client, i l'Scrum Master, persona encarregada de ser el pont entre els programadors i el PO.

En Scrum el transcurs del projecte es divideix en iteracions, també anomenat *sprints*. Són unitats de temps que engloben entre dos i quatre setmanes, on l'equip de desenvolupament té assignats una sèrie de punts d'esforç. Per altre banda, setmanalment totes les persones implicades en el projecte, desenvolupadors, POs i Scrum Master, determinarien les tasques a realitzar en la següent iteració. Les tasques són estimades pels

desenvolupadors, assignant punts d'esforç a cada una. En funció dels punts d'esforç que pugui assumir l'equip, i de la prioritat que els POs estableixin, es marca la llista de tasques que seran assignades dins l'sprint. Aquesta llista és anomenada *backlog*.

2.1.2 Extreme Programming (XP)

Extreme Programming és un mètode que té com a principal objectiu la satisfacció del client, però on prenen molta rellevància les pràctiques adaptades per aconseguir-ho. Promou la flexibilitat en el desenvolupament mitjançant la planificació de releases petites dins d'un desenvolupament incremental, donant forma al projecte per fases setmanals, i permetent al client modificar o adaptar els requisits sense que això hagi de d'afectar a l'avanç o els costos del projecte. A més de tenir una filosofia molt remarcada, una de les seves principals característiques es promoure bones pràctiques en la programació, com la simplicitat en el codi, el *testing*, el *refactoring*, entre d'altres.



Esquema del funcionament de l'Extreme Progamming. Font: http://www.extremeprogramming.org/map/project.html

En realitat, entre XP i Scrum i ha moltes més similituds que no pas diferències[1]. Es considera que Scrum acostuma a presentar iteracions més llargues que XP, entre dos i quatre setmanes, per entre una i dos setmanes d'XP. Scrum és més inflexible, degut a que teòricament no es pot modificar l'ordre o afegir tasques a les ja definides dins d'un sprint. Per altra banda XP enfoca el desenvolupament estrictament en l'ordre de prioritat definit pel client, mentre que en Scrum és l'equip de desenvolupadors qui prioritza dins del backlog d'un sprint. Potser la principal diferència entre ambdues metodologies és que XP s'ha pensat aplicant en ell implícitament certes pràctiques d'enginyeria (*testing, refactoring,* estàndards de programació, releases

1 MONTAIN GOAT SOFTWARE. *Differences between Scrum and Extreme Programming*. [en línia]. http://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming [data de consulta: 06/01/2014]

petites, etc) mentre que Scrum es focalitza principalment en la gestió del projecte sense especificar les pràctiques en el desenvolupament. Val a dir però que les pràctiques que XP promou poden ser també aplicades en Scrum.

2.1.3 BDD (Behaviour Driven Development)

El *Behaviour Driven Development* (Desenvolupament Guiat pel Comportament) no és un mètode pròpiament dit. Més aviat es podria considerar una pràctica que es pot aplicar dins dels mètodes de desenvolupament àgil. El BDD és una evolució del TDD (*Test Driven Development* o Desenvolupament Guiat per Test) on el primer que es desenvolupa a l'hora de programar, abans que el propi codi, és el codi del tests que recolzaran el programa. L'aportació d'aquesta pràctica és bàsicament que programant el test abans que el codi obliga a que la gran part de l'aplicació disposi de tests des del primer moment, donant solidesa a mig i llarg plaç a l'aplicació. A l'hora, especificar el test des del primer moment ajuda al programador a especificar el propi funcionament de la classe.

BDD és una evolució de TDD en la mesura que intenta aproximar els tests al valor de negoci, seguint el marc que donen les històries d'usuari. Aproximant ambdós llenguatges s'aconsegueixen especificacions més exactes a les necessitats de negoci, obtenint com a resultat un millor producte.

2.2. Diferències existents en la aplicació de mètodes àgils en l'entorn d'aquest projecte respecte d'un entorn real

La naturalesa d'aquest projecte fa impossible l'aplicació formal dels mètodes explicats anteriorment, començant pel fet que aquest és un treball unipersonal, i els mètodes estan pensats per entorns pluridisciplinaris amb equips de desenvolupament pluripersonals.

Per aquest projecte una sola persona ha complert el rol de Product Owner, de desenvolupador i d'usuari final de la aplicació. El rol com a client es va desenvolupar bàsicament en l'inici del projecte, durant la planificació, a l'hora d'especificar els requisits. Cal remarcar la consciència tinguda en no haver sigut possible posar en pràctica la comunicació entre negoci i desenvolupament, tan remarcada i emfatitzada en els mètodes àgils aplicats. Ha sigut el rol de desenvolupador el més aplicat durant el procés d'elaboració del projecte.

Una altra diferència que hi ha hagut és la falta de canvis de requisits durant el projecte. En entorns reals on els clients són persones diferents és molt més freqüent la re-priorització de requisits, o la inclusió d'alguns de nous. En el present cas, al tractar-se de la mateixa persona no hi havia intenció ni necessitat de variar els requisits respecte a lo determinat a l'inici del projecte.

3. Principals eines aplicades

Una part molt important dels objectius de la pràctica és la aplicació d'un seguit d'eines, tal i com s'explicava al punt 1.2 d'aquest document. A continuació es realitza una introducció al funcionament bàsic de les que han sigut les dos eines més importants en el projecte: Symfony2 i Behat.

3.1 Symfony

Symfony2 és la segona versió del primer framework que va aparèixer per PHP. És una versió totalment nova i incompatible amb l'anterior, que explota al límit les noves característiques de PHP 5.3. Ha sigut desenvolupada per l'empresa SensioLabs originària a França, però que ha rebut i rep multitud de contribucions per part de la comunitat de desenvolupadors que el fan servir arreu del món.

3.1.1 L'arquitectura

Symfony2 està 100% desenvolupat utilitzant el paradigma de la programació orientada a objectes aprofitant al màxim les possibilitats que donen els *Namespaces* a PHP, permetent distribuir diferents funcions en diferents paquets independents, donant una gran flexibilitat al desenvolupament. Aquests paquets són els anomenats Components que permeten el funcionament de tot el sistema.

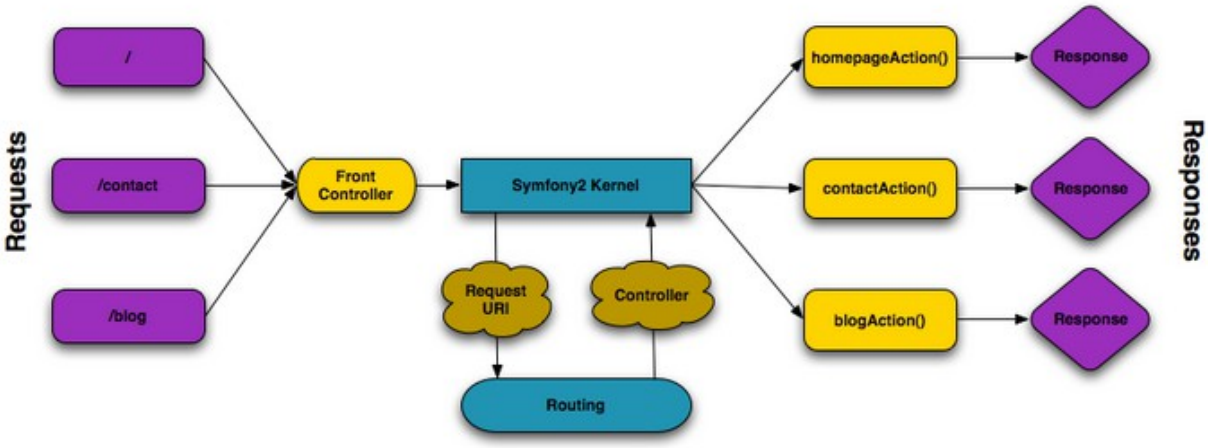
També existeixen un altre tipus de components anomenats *Bundles*. Un Bundle és un component que no és indispensable pel funcionament del framework, però que aporta una lògica de negoci que pot ser reutilitzada en diferents projectes, per exemple, un Bundle per realitzar estadístiques a partir d'una base de dades.

3.1.2 Els components

1) **HTTP**

   Aquest és el component bàsic de funcionament. És l'encarregat d'interpretar les peticions i de generar les respostes segons el protocol HTTP. Es fa servir el patró del *Front Controller*, el qual permet que totes les peticions HTTP que es fan a l'aplicació siguin gestionades pel mateix fitxer. És el principi bàsic per poder aplicar el patró MVC. A partir d'aquí existeixen diferents classes[2] que s'encarreguen de gestionar la petició al servidor i la resposta que aquest donarà al client.

2 Element principal del paradigma de la programació orientada a objectes. Simbolitza l'element bàsic a l'hora de distribuir una entitat lògica.



Esquema del procés d'una petició a Symfony2. Origen: http://symfony.com/doc/current/book/http_fundamentals.html

2) Enrutament

L'enrutament a Symfony2 té la mateixa base que a qualsevol altre framework. En realitzar una petició PHP es determina a quin Controlador i a quina de les seves accions s'estan fent referència. Si es troba, s'executa.

La configuració d'aquest enrutament es pot fer de diferents maneres, però sempre mantenint una lògica dins del codi.

3) Motor de plantilles

Symfony2 fa servir un motor de plantilles anomenat Twig, tot i que també existeix la opció d'utilitzar PHP per a la realització de les mateixes).

El motor de Twig serveix per evitar que hi hagi lògica de programació barrejada amb codi de presentació (HTML), facilitant així el manteniment del codi, aporta múltiples avantatges respecte de no fer servir-lo. La avantatge més destacada és la opció d'herència. Amb Twig una plantilla pot estendre una altra sobreescrivint alguns blocs de la plantilla estesa amb la informació de la que estén. Aquesta característica aporta una gran flexibilitat a l'hora de construir disseny i reaprofitar codi.

Twig també incorpora un seguit de funcions que són utilitzades a les plantilles i que permeten formatejar certs continguts de manera neta i amb el mínim esforç.

4) Bases dades i mapatge

Symfony2 permet treballar amb diferents llibreries d'ORM. La més utilitzada és una anomenada Doctrine2, que és l'encarregada de processar tota la lògica de mapatge i vinculant les taules de la base de dades amb les Entitats que es processaran dins del codi. En realitat Symfony2 no fa res de tot això, únicament facilita un component que fa de pont entre el framework i Doctrine2 per a poder integrar totes les funcionalitats de la llibreria.

5) Formularis

Una de les tasques més comunes i repetitives per un desenvolupador web és la creació de formularis, molt especialment a l'hora de crear *backoffices*[3] on són el mecanisme d'introducció de dades a una base de dades, i on pràcticament sempre és necessari crear dos formularis iguals, amb la diferència que un és per creació i l'altre per modificació de registres.

Com es mencionava, aquesta és una de les tasques més repetitives i és més normal invertir hores de codificació ja que en els formularis d'avui dia és requisit indispensable que hi hagi una correcta manipulació de les dades, i en cas d'error, que aquests es mostrin amigablement al usuari que ha introduït malament les dades.

El component de formularis el que permet és especificar en un únic arxiu com ha de ser un formulari, quins camps ha de tenir i com s'han de validar aquests camps, i partir d'aquesta informació el component s'encarregarà de presentar el formulari en codi HTML, i de validar-lo.

6) Validació

La validació de dades és quelcom molt freqüent en les aplicacions web. Per exemple, els formularis són el sistema més comú per enviar dades del client el servidor. Per tenir cura de la seguretat i del bon funcionament de l'aplicació, cal validar les dades que són enviades mitjançant formularis. En PHP era molt freqüent embrutar el codi fent comprovacions lògiques de les dades i molts cops de manera repetitiva en camps que tenien les mateixes regles de validació.

Aquest component, doncs, permet configurar de manera molt senzilla i neta les validacions que han de tenir els camps d'un formulari.

7) **Seguretat**

Symfony2 aporta eines per assegurar-se que un usuari accedeix als recursos pels qual realment hauria de tenir accés. Els dos passos bàsics són la autenticació i la autorització. El primer s'encarrega de verificar que l'usuari realment és qui diu ser. El segon pas s'encarrega de comprovar que efectivament aquest usuari pot accedir a les dades que vol.

Symfony2 ja integra un sistema que permet al desenvolupador configurar els diferents *Firewalls* i control d'accessos per l'aplicació, així com també un sistema per definir usuaris i rols per poder tenir més flexibilitat a l'hora de fer aquestes configuracions.

8) Contenidor de dependències

Aquest component segueix els principis del patró Injecció de dependències. Un contenidor de dependències s'encarrega de gestionar els serveis[4] dins d'un aplicació. El contenidor de

3 Terme utilitzat per referir-se a un panell privat per la gestió de continguts d'una pàgina web dinàmica

4 Un servei dins d'una aplicació es refereix a qualsevol objecte encarregat del funcionament d'una tasca concreta dins d'aquesta aplicació.

dependències és l'encarregat d'instanciar aquests objectes. En cas de que hi hagi objectes que tinguin dependència entre ells, s'encarregarà de vincular-los.

Per exemple, si a l'aplicació hi ha un servei encarregat d'enviar emails d'una manera determinada, serà necessari que aquest objecte tingui dins seu la instància d'un altre objecte que s'encarregui de crear aquest email.

3.1.3 Els Bundles

Com es mencionava anteriorment, els Bundles són paquets de Symfony2 que aporten una lògica més de negoci. Segons marca l'ús de Symfony, tot la lògica de negoci de l'aplicació estarà distribuïda en Bundles. Per exemple, si existeix la necessitat de desenvolupar una aplicació web amb un apartat on s'hagi de mostrar estadístiques per una serie de valors extrets de la base de dades, els desenvolupadors del projecte s'encarregarien de crear una serie de classes que tindrien aquesta funció.

A més d'això, els Bundles també es poden utilitzar com a components intercanviables entre projectes, i inclús podrien ser distribuïts per Internet i fer-se servir per diferents organitzacions. En el cas de l'exemple anterior, on seria necessari un Bundle per gestionar estadístiques, abans de començar el desenvolupament s'hauria de cercar a Internet si ja existeix un que compleixi aquests requisits, de manera que els desenvolupadors s'estalviarien programar-lo des de zero. En casos reals, és probable que s'hagués de fer algun ajustament al Bundle per poder implementar-lo a la aplicació en qüestió, però en qualsevol cas, l'objectiu és estalviar temps de desenvolupament i millorar la productivitat.

Els Bundles a Symfony2 són desenvolupats per la comunitat de programadors que envolten al framework i que l'utilitzen assíduament i que es poden trobar a la web knpbundles.com.

En l'actualitat existeixen molts tipus de Bundles que la comunitat posa a disposició de la resta. Alguns dels més coneguts estan enfocats a la gestió d'usuaris dins d'una aplicació, a la creació de Backoffices, o la creació d'una API REST utilitzant Symfony2 com a plataforma.

3.2 Behat

Behat és una eina que té l'objectiu de facilitar la pràctica del BDD dins del desenvolupament d'aplicacions web en PHP. Permet l'execució de tests funcionals escrits en un estil llegible per humans (*human-readable*) i que es basen en les històries d'usuaris definides al projecte, amb l'objectiu que l'àrea de negoci i la de desenvolupament utilitzin els mateixos termes i llenguatge . En Behat les històries d'usuari són anomenades *features,* que en aquest context voldria dir funcionalitat. A la pràctica un fitxer amb una feature conté un seguit d'instruccions interpretables que descriuen els passos a seguir per un usuari dintre de la interfície de l'aplicació.

Per altra banda, existeixen uns fitxers anomenats *Context*, els quals que contenen tota la lògica programàtica on es tradueixen les instruccions interpretables en funcions en llenguatge PHP, i són el pont entre les features i la aplicació principal del projecte.

3.2.1 Les Features

Aquestes instruccions interpretables s'escriuen mitjançant un llenguatge anomenat Gherkin. Aquest llenguatge llegeix el fitxer de manera que pot extreure diferents escenaris que s'hagin indicat. Cada escenari conté les condicions, accions i conseqüències que han de complir-se per determinar que l'escenari es compleix com està previst.

A continuació es pot veure l'estructura bàsica del que compondria un arxiu feature de Behat.

```
Feature: Aquí es descriu un breu resum de la funcionalitat
    Per tal de realitzar una tasca de negoci
    Com a un actor del sistema
    Necessito veure algun tipus de sortida que m'ajudi a complir el meu objectiu

    Scenario: Alguna situació de negoci determinada
        Given una precondició concreta
          And una altra precondició
         When faig alguna acció determinada
          And realitzo una altra acció
         Then aconsegueixo una resposta del sistema avaluable
          And i una altra cosa que es pot comprovar succeïx també

    Scenario: Una altra situació
          ...
```

Exemple de contigut d'un arxiu feature de Behat

És interessant esmentar que el text que acompanya la paraula Feature és un text merament informatiu i que descriu el que en el seu moment va ser indicat en el *post-it* que representa una història d'usuari.

La part més important vindria a ser les instruccions que acompanyen cadascun dels Escenaris (Scenario). Sota aquests es troben les condicions, accions i resultats avaluables esperats. Al començament de cadascun d'ells es troben les variables clau que marquen el tipus d'instrucció: *Given*, *When* i *Then*.

3.2.2 Els Contexts

Els Contexts són arxius PHP que contenen classes[5] amb la lògica que hi ha darrera de les instruccions dels arxius feature. Aquestes classes que estenen les pròpies classes de Behat poden comunicar-se amb Symfony2, i prenen principal rellevància amb el fet que poden utilitzar el contenidor de dependències de Symfony, de tal manera que facilita molt la creació de les condicions necessàries per crear els escenaris. Accedir al contenidor de dependències suposa tenir accés als diferents gestors del model, el qual permet de manera molt més simple crear els registres a la base de dades per fer els tests corresponents.

3.2.3 Mink

Mink és una extensió de Behat. El seu principal objectiu és oferir Contexts amb instruccions predefinides per la construcció de tests funcionals. Això significa que Mink aporta moltes de les instruccions incloses a les features, que són utilitzades per interpretar els continguts HTML d'una web.

3.2.4 Eines d'automatització de tests

Els tests funcionals que executa Behat no serien possibles sense un altre eina com és un automatitzador de tests. En el cas de SegundoUso.org la eina utilizada és anomenada Sahi, i és l'encarregat d'executar un navegador i que Behat pugui interactuar amb ell.

5 Classe entesa com element que defineix un Objecte dintre del paradigma de la programació orientada a objectes

4. Abast i planificació

El projecte SegundoUso.org parteix d'un objectiu de negoci clar, però dintre de totes les possibilitats que ofereix desenvolupar una web d'aquest tipus calia especificar les funcionalitats que aquesta tindria al finalitzar l'assignatura. Un cop definits calia ordenar-los en el temps dintre del marc temporal que donava l'assignatura.

4.1 Idea de negoci

Un dels objectius del projecte és desenvolupar una web anomenada SegundoUso.org. Els requisits bàsics amb els que partia aquesta idea eren la d'una plataforma on els usuaris poguessin publicar anuncis de manera gratuïta, i oferint objectes. Es volia oferir la possibilitat de poder fer-ho sense que l'usuari necessités registrar-se per participar amb l'objectiu d'evitar barreres que perjudiquessin l'activitat, tal com fan altres webs conegudes com loquo.com

A més, també es vol oferir la possibilitat de registrar-se, per tal que qui volgués podria tenir la seva compte, i així realitzar gestions des de la pròpia web. L'opció de tenir una compte dona moltes més opcions per que l'usuari pugui tenir un perfil i es pugui traçar més fàcilment la seva activitat a la web.

Un cop es té definida la idea de com ha de ser la web es pot passar a les següents etapes de la planificació. Denota que en el desenvolupament àgil no calen grans documents explicant un per un els requisits. En comptes d'això la manera de desenvolupar la planificació és emplaçar en una mateixa sala totes les persones que tindran quelcom a veure en el projecte: clients, desenvolupadors i *stakeholders*. En aquest punt és on es comencen a definir les idees del que haurà de tenir la web, i a partir d'aquí les històries d'usuari.

4.2 Etapes de la planificació

La primera tasca duta a terme ha sigut definir el que havia de complir aquesta web. A arrel d'aquí es produiran un seguit de passos que componen la planificació:

1. Definir l'abast: Correspon a determinar què ha de fer el projecte, i materialitzar aquest concepte en actors, entitats i tasques.
2. Definir les històries d'usuari: A partir dels conceptes definits en l'abast es detallen les històries d'usuari entrant en més detall en les tasques, l'actor que les durà a terme, i què correspondrà exactament a una tasca.
3. Estimar les històries d'usuari: Determinar l'esforç que necessitarà cadascuna de les històries d'usuari.

4. Definir les iteracions: Acabar de determinar les dates que ocuparan les iteracions, i els dies de release.

4.3. Abast

Per definir l'abast s'ha realitzat una *brain storming* on s'escrivís en papers els diferents conceptes que donarien forma a la web: **entitats**, **actors** i **tasques** dels actors en l'aplicació. Per cadascun d'aquests s'ha fet servir un color. Groc clar per les entitats, rosa pels actors i taronja per les tasques.

Post-its indicant el primer esquema conceptual de SegundoUso.org

El contingut de la imatge anterior es correspon al següent esquema:

Actors

Els diferents perfils d'usuaris que tindran protagonisme i tasques diferenciades són tres:

1. **Usuari visitant**: Es l'usuari que arriba a la web i pot navegar per les diferents pàgines on està continguda la informació dels anuncis. Pot cercar anuncis utilitzant els diferents filtres disponibles. El visitant podrà accedir a la pàgina de detall d'un anunci i contactar l'anunciant mitjançant un formulari.

   El visitant també podrà publicar anuncis, convertint-se virtualment en **anunciant**. Gestionarà els seus anuncis (editar i esborrar) des d'un email que rebrà en el moment de la publicació.

Podrà guardar els seus anuncis favorits, de manera que podrà tenir-los en un pàgina especial on podrà accedir-los tots fàcilment. Aquesta informació es guardarà en *cookies* en la màquina utilitzada, per tant no estarà disponible si l'usuari visitant accedeix des d'una màquina diferent d'on va guardar els favorits.

Per últim, l'usuari visitant podrà marcar anuncis que consideri fraudulents, una opció que donarà als usuaris una eina per mantenir el bon ús de la web.

2. **Usuari registrat**: És un usuari que podrà fer les mateixes funcions que l'usuari visitant, amb la diferència que haurà indicat un nom d'usuari, email i contrasenya, amb lo qual estarà creant una compte.

   Les avantatges que aporta estar registrat són que en cas de publicar anuncis podrà gestionar-los des de la pròpia web en comptes de mitjançant un email.

   Una altra avantatge és que els seus favorits també es guardaran vinculats a aquesta compte d'usuari, per la qual cosa també podrà visualitzar-los des de qualsevol ordinador sempre que s'accedeixi primer.

3. **Administrador**: És el perfil d'usuari intern que s'encarregarà de la gestió de la pàgina. Serà un usuari registrat a la base de dades amb permisos especials que li permetrà gestionar anuncis publicats i les categories dels anuncis. Accedirà a un *Backend* a través d'un formulari de login diferenciat del de la pàgina pública.
   Un cop dins del *Backend* tindrà les opcions per editar o eliminar anuncis, i categories. Disposarà d'una secció on podrà gestionar visualitzar anuncis marcats com fraudulents i decidir la seva eliminació.

Entitats

Les entitats definides correspondrien els conceptes de negoci i elements de la web que possiblement esdevindrien en les taules de la base de dades. Les entitats definides foren les següents:

1. **Anunci**: és la peça clau de la informació en la web ja que la informació principal d'aquesta són els anuncis.
2. **Categories d'anuncis**: permetrien la categorització dels anuncis per tal de permetre un filtratge dels resultats i la seva visualització més còmode en funció de l'interès dels usuaris.
3. **Fotos dels anuncis**: permetria vincular més d'una imatge als anuncis pujats de manera que l'objecte enunciat es pugui visualitzar òptimament.
4. **Etiquetatge d'anuncis**: permetria que els usuaris etiquetessin els objectes oferts de manera que donaria més filtres a l'hora de cercar.
   Aquesta entitat va quedar descartada durant el procés posterior.

5. **Localització d'anuncis**: donaria pas a una entitat del tipus ciutat, que permetria vincular l'anunci a una àrea geogràfica i d'aquesta manera servir com un filtre més de cara a la cerca d'anuncis.

Tasques

Com a tasques s'han definit les principals funcions que els actors definits anteriorment haurien de poder complir:

1. Crear anuncis sense estar registrat
2. Editar i esborrar anuncis sense estar registrat mitjançant opcions en un email.
3. Marcar anuncis com a favorits sense estar registrat
4. Marcar anuncis com a fraudulents
5. Contactar anunciants mitjançant un formulari
6. Registrar-se com a usuari
7. Crear anuncis com a usuari registrat
8. Gestionar anuncis com a usuari registrat
9. Marcar anuncis favorits com a usuari registrat
10. Cercar anuncis per paraula
11. Eliminar anuncis fraudulents com administrador
12. Gestionar categories de la web com administrador.

Totes aquestes tasques seran la base per crear les històries d'usuari.

Val la pena esmentar que durant aquesta pluja d'idees van sortir altres funcionalitats per la web, que posteriorment van quedar descartades per la planificació en aquest treball per falta de temps.

4.4 Històries d'usuari

Un cop definit l'abast del projecte, el següent pas era crear les històries del projecte a partir de les tasques.

4.4.1 Crear Anunci sense registrar-se

Història	Crear Anunci sense registrar-se
Resum	Com a **usuari visitant**

	per poder oferir un bé he de poder crear un anunci sense haver-me de registrar
Condicions prèvies	• Existeixen categories a la base de dades • L'usuari es troba a la home del *site*
Accions a desenvolupar	1. L'usuari visitant clica la opció de “Anunciar gratis” 2. Omplir el formulari amb les dades pertinents i clicar el botó de publicar. 3. L'usuari veurà un missatge informant que ha rebut un email on confirmarà la publicació de l'anunci. 4. L'usuari accedirà al seu email i clicarà l'enllaç de confirmació 5. L'enllaç el portarà a SegundoUso.org on veurà un missatge de confirmació
Resultat del sistema	• L'anunci estarà publicat a la categoria i municipi indicat per l'usuari

4.4.2 Veure anuncis sense registrar-se

Història	Veure anuncis sense registrar-se
Resum	Com a **usuari visitant** per poder trobar anuncis i bens del meu interès necessito poder llistar-los
Condicions prèvies	• Hi haurà anuncis publicats • L'usuari tindrà un municipi seleccionat des de la part superior del site
Accions a desenvolupar	1. Des de la pàgina d'inici del *site*, l'usuari podrà seleccionar una categoria sobre la que cercar anuncis. Si no selecciona una categoria es buscaran els anuncis per totes les categories. 2. L'usuari clicarà el botó “Buscar” 3. Es redireccionarà a una pàgina on es mostraran els anuncis pels criteris triats (ciutat i categoria) 4. L'usuari podrà clicar en un dels anuncis mostrats 5. Anirà a una pàgina amb el detall dels anuncis
Resultat del sistema	• L'usuari estarà visualitzant l'anunci

4.4.3 Contactar anunciant i recepció d'email

Història	Contactar anunciant i recepció email
Resum	Com a **usuari visitant** o **usuari registrat** per poder aconseguir contactar un anunciant necessito un sistema de contacte
Condicions prèvies	• L'usuari es troba en la pàgina de detall d'un anunci
Accions a desenvolupar	1. L'usuari omple el formulari de contacte amb les dades demanades i clica el botó d'enviar. 2. Es mostrarà un missatge de confirmació a l'usuari
Resultat del sistema	• L'anunciant rebrà un email amb el missatge de l'usuari interessat i podran iniciar el contacte.

4.4.5 Gestionar anuncis propis sense estar registrat

Història	Gestionar anuncis propis sense estar registrat
Resum	Com a **usuari visitant** per poder actualitzar un anunci meu he de poder modificar-lo
Condicions prèvies	• L'usuari ha de tenir al menys un anunci publicat
Accions a desenvolupar	1. L'usuari accedeix al formulari d'edició del seu anunci a través del email que va rebre en el moment de publicar-lo. 2. L'usuari edita les opcions de l'anunci que vol modificar i envia el formulari. 3. L'usuari veurà un missatge de confirmació o error en funció de si s'ha fet l'acció correctament.
Resultat del sistema	• L'anunci haurà sigut modificat

4.4.6 Marcar anuncis favorits sense registrar-se

Història	Marcar anuncis favorits sense registrar-se
Resum	Com a **usuari visitant**

	per poder accedir als meus anuncis favorits i facilitar així la meva interacció he de poder emmagatzemar-los
Condicions prèvies	• L'usuari ha de trobar-se a la pàgina de detall d'un anunci
Accions a desenvolupar	1. L'usuari clicarà la opció “Favorito” de l'anunci. Automàticament aquest passarà a ser emmagatzemat com a favorit. 2. L'usuari seleccionarà la opció “Favoritos” del menú superior de la pàgina. 3. L'usuari visualitzarà l'anunci marcat en el llistat de la pàgina.
Resultat del sistema	• L'anunci quedarà emmagatzemat com a favorit en l'ordinador que l'usuari ha utilitzat.

4.4.7 Introduir fotos als anuncis

Història	Introduir fotos als anuncis
Resum	Com a **usuari visitant** o **usuari registrat** per tal d'oferir més informació sobre un bé he de poder adjuntar fotos d'aquest a l'anunci
Condicions prèvies	• Dins del procés de crear o editar un anunci
Accions a desenvolupar	1. L'usuari podrà afegir imatges als anuncis clicant sobre la opció “Añadir imagen”, apareixent un camp d'arxiu al formulari.
Resultat del sistema	• A l'hora de guardar l'anunci la imatges o imatges adjuntades quedaran vinculades a l'anunci.

4.4.8 Marcar anunci com fraudulent

Història	Marcar anunci com a fraudulent
Resum	Com a **usuari visitant** i **usuari registrat** per poder contribuir al correcte ús de la web he de poder marcar anuncis que siguin fraudulents
Condicions prèvies	• Hi ha d'haver anuncis publicats
Accions a desenvolupar	1. L'usuari, trobant-se a la pàgina de detall d'un anunci, podrà prémer el botó “Fraudulento” per marcar l'anunci com a fraudulent.

Resultat del sistema	• L'anunci quedarà internament marcat com a fraudulent pel seu posterior control per part de l'administració de la web.

4.4.9 Crear Àrea privada (Administració)

Història	Accés a àrea privada
Resum	Com a **administrador** Per tal d'accedir a l'administració de manera privada necessito que sigui un àrea que demani credencials per accedir-hi
Condicions prèvies	• NA
Accions a desenvolupar	1. Accedir al formulari d'entrada intern per l'administració 2. Omplir-lo amb les dades d'un usuari administrador
Resultat del sistema	• En cas d'introduir les dades d'usuari correctament es crearà una sessió i l'usuari podrà accedir a les àrees privades internes • En cas d'introduir errròniament l'usuari mostrarà un error i n'impedirà l'accés

4.4.10 Eliminar anuncis

Història	Eliminar anuncis
Resum	Com a **administrador** per tal de gestionar els anuncis he de poder eliminar anuncis que consideri incorrectes
Condicions prèvies	• L'usuari està prèviament logat com a administrador
Accions a desenvolupar	1. L'administrador accedirà a la opció “Ads – All Ads” a través del menú superior. 2. Visualitzarà tots els anuncis publicats a la web. 3. Podrà eliminar anuncis a través del botó corresponent per cada anunci.
Resultat del sistema	• L'anunci quedarà esborrat del sistema.

4.4.11 Gestionar categories d'anuncis

Història	Gestionar categories d'anuncis
Resum	Com a **administrador** per tal de poder classificar anuncis he de poder gestionar categories on es puguin classificar
Condicions prèvies	• L'usuari està prèviament logat com a administrador
Accions a desenvolupar	• L'administrador accedirà a la opció “Categories” a través del menú superior. • L'administrador podrà crear categories noves a través del botó corresponent. • L'administrador podrà editar o eliminar categories existents.
Resultat del sistema	• Les categories quedaran creades, editades o esborrades de la base de dades en funció de la acció feta.

4.4.12 Navegar per ciutat

Història	Navegar per ciutat
Resum	Com a **usuari visitant** o **usuari registrat** per tal de trobar bens a prop meu necessito poder seleccionar per ciutat
Condicions prèvies	• Han d'haver municipis introduïts al sistema
Accions a desenvolupar	1. L'usuari triarà un municipi al menú superior. 2. Aquest serà el municipi per defecte a l'hora de fer les cerques d'objectes a la web
Resultat del sistema	• Els resultats d'anuncis mostrats a una cerca estaran filtrats pel municipi triat a la part superior

4.4.13 Crear anuncis per ciutats

Història	Crear anuncis per ciutats
Resum	Com a **usuari visitant** i **usuari registrat**

	per tal d'indicar millor on es pot recollir el bé que ofereixo necessito poder assignar la ciutat al crear editar l'anunci
Condicions prèvies	• Dins del procés de crear o editar un anunci
Accions a desenvolupar	1. L'usuari tindrà la obligatorietat de triar un municipi al que enllaçar l'anunci. A més continuarà disposant del camp localització per especificar l'àrea de recollida.
Resultat del sistema	• L'anunci quedarà enllaçat a un municipi disponible a la base de dades

4.4.14 Registrar-se com a usuari

Història	Registrar-se com a usuari
Resum	Com a **usuari visitant** per tal de tenir compte i més opcions a la web he de poder registrar-me
Condicions prèvies	• L'usuari no pot ser usuari registrat i logat prèviament
Accions a desenvolupar	1. L'usuari clicarà sobre la opció “Registrarse” en el menú superior. 2. L'usuari omplirà el formulari de registre i l'enviarà. 3. Apareixerà la pàgina de confirmació si les dades han sigut introduïdes correctament.
Resultat del sistema	• L'usuari quedarà registrat i logat automàticament si les dades són correctes. • Si les dades de l'usuari són incorrectes (usuari o email ja existeix, email no és vàlid o contrasenya i confirmació de contrasenya són incorrectes) l'usuari tornarà un error.

4.4.15 Opció per cercar anuncis

Història	Cercar anuncis per text
Resum	Com a **usuari visitant** o **usuari registrat** per tal de trobar més fàcilment anuncis del meu interès he de poder cercar-los pel text que pugui aparèixer en el títol o la descripció de l'anunci
Condicions prèvies	• Hi ha d'haver anuncis publicats prèviament
Accions a desenvolupar	1. L'usuari, des del camp de cercar de la part superior introduirà una o més d'una paraules clau i clicarà el botó amb la lupa. 2. Es redireccionarà l'usuari a una pàgina on llistarà tots els anuncis que continguin la paraula clau introduïda.
Resultat del sistema	• Mostrarà resultats d'anuncis per la paraula introduïda.

4.4.16 Gestionar anuncis fraudulents

Història	Gestionar anuncis fraudulents
Resum	Com a **administrador** per poder assabentar-me d'avisos d'anuncis fraudulents he de poder veure els anuncis marcats com a fraudulents
Condicions prèvies	• L'usuari està prèviament logat com a administrador
Accions a desenvolupar	1. L'administrador accedirà a la opció “Ads – Fraudulent Ads” a través del menú superior. 2. Visualitzarà tots els anuncis que han sigut marcats com a Fraudulents. 3. Podrà eliminar anuncis a través del botó corresponent per cada anunci.
Resultat del sistema	• Els anuncis que hagi esborrat com a fraudulents desapareixeran de la base de dades.

4.4.17 Crear anunci com a usuari

Història	Crear anunci com a usuari
Resum	Com a **usuari registrat**

	per tal de poder gestionar anuncis en el meu compte he de poder crear anuncis
Condicions prèvies	• L'usuari està prèviament logat com a usuari
Accions a desenvolupar	• El procés serà el mateix que l'indicat per als usuaris visitants.
Resultat del sistema	• L'anunci estarà publicat a la categoria i municipi indicat per l'usuari

4.4.18 Marcar anuncis preferits com a usuari registrat

Història	Marcar anuncis favorits com a usuari registrat
Resum	Com a **usuari registrat** per tal de gestionar els meus anuncis he de poder marcar-los i visualitzar-los
Condicions prèvies	• L'usuari està prèviament logat com a usuari • L'usuari es troba en la pàgina de detall d'un anunci
Accions a desenvolupar	1. L'usuari clicarà la opció “Favorito” de l'anunci. Automàticament aquest passarà a ser emmagatzemat com a favorit. 2. L'usuari seleccionarà la opció “Mi cuenta - Favoritos” del menú superior de la pàgina. 3. L'usuari visualitzarà l'anunci marcat en el llistat que apareixerà a la pàgina.
Resultat del sistema	• L'anunci quedarà emmagatzemat com a favorit en el sistema i enllaçat a l'usuari en qüestió, per lo que podrà veure'l des de qualsevol ordinador.

4.4.19 Gestionar els meus anuncis

Història	Gestionar els meus anuncis
Resum	Com a **usuari registrat** per tal poder gestionar anuncis en el meu compte he de poder accedir a ells
Condicions prèvies	• L'usuari està prèviament logat com a usuari • L'usuari ha de tenir al menys un anunci publicat
Accions a desenvolupar	• L'usuari selecciona la opció “Mi cuenta – Mis anuncios” del menú

	superior de la pàgina. • L'usuari visualitza el llistat amb els seus anuncis propis. Selecciona la opció editar d'un anunci. • L'usuari accedirà al formulari d'edició d'un anunci. Farà els canvis corresponents i premerà el botó d'editar.
Resultat del sistema	• L'anunci haurà sigut modificat

4.5 Mapatge de les històries d'usuari

Com s'acostuma a fer tradicionalment dins de la planificació en SCRUM, totes les històries d'usuaris plasmades en post-its s'enganxen en una paret on es crearà el mapatge d'històries d'usuari (User story mapping en anglès). Aquest mapa ajudarà a crear d'una manera molt visual les etapes del projecte i l'esforç que requerirà.

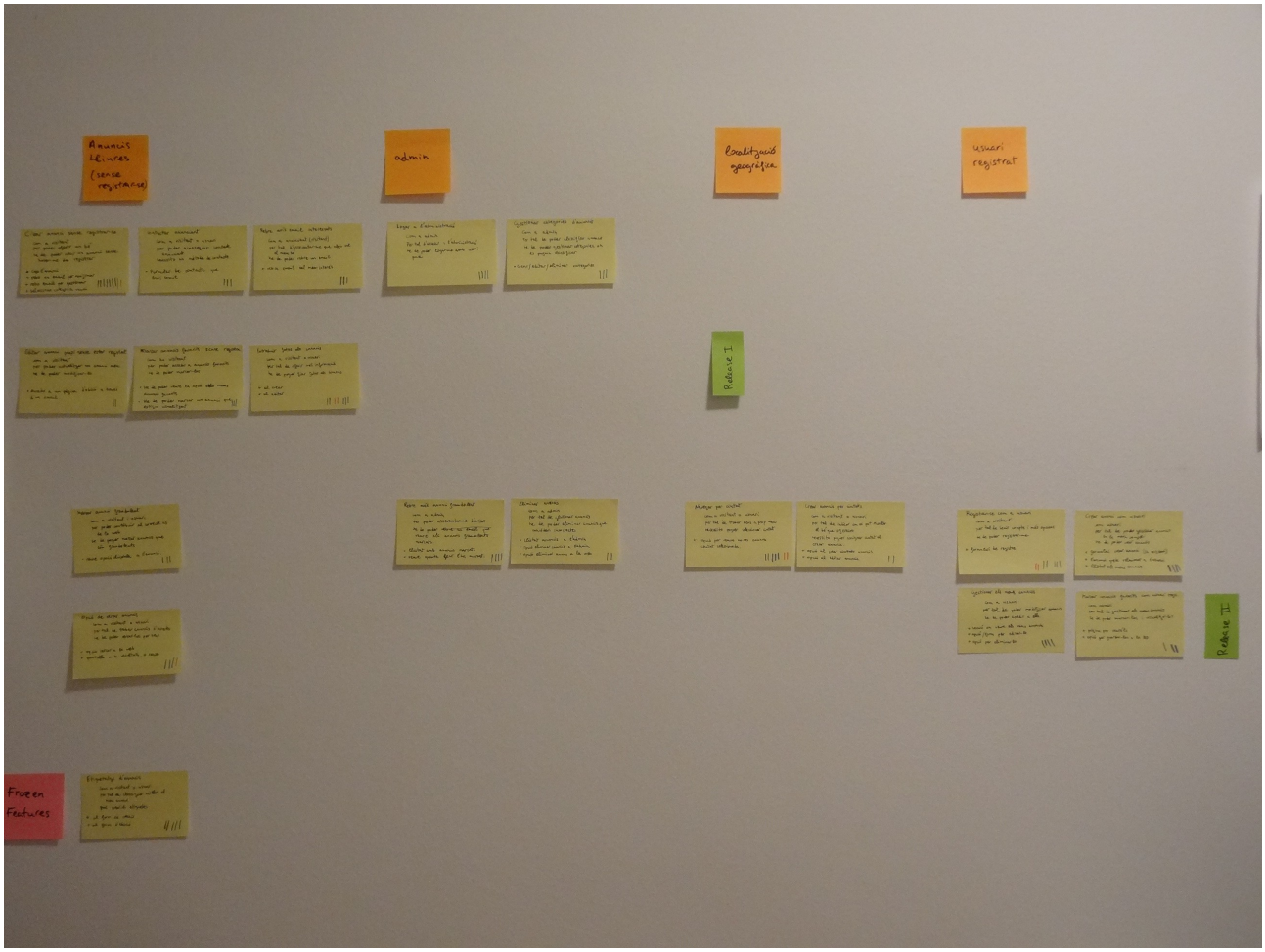
El primer que cal per crear el mapa és definir les gran àrees del projecte. En aquest cas, les gran àrees s'han definit a partir de les grans funcionalitats que requeriran les històries d'usuari:

- **Sistema d'anuncis** (per usuaris visitants): Engloba pràcticament la funcionalitat bàsica de la web, que correspon a la home, els formularis de creació, edició i eliminació d'anuncis, l'avís d'anuncis fraudulents, marcar anuncis com a favorits, i l'ampliació del sistema d'anuncis per incloure fotografies.
- **Administració**: Tota la part de backend des d'on l'usuari administrador podrà gestionar la web.
- **Localització geogràfica**: La part del sistema que permetrà vincular els anuncis a ciutats i cercar a partir d'elles.
- **Sistema d'usuaris registrats**: L'ampliació del sistema que permetrà als usuaris crear el seu compte i controlar els seus anuncis des de la web.

Un cop estan definides les històries d'usuari i les grans àrees que les engloben, es pot començar a concretar la planificació. Donat el marc del TF en que es troba aquest projecte, les dates de lliurament de les PACs marcaven un requeriment que s'havia de tenir en compte a l'hora de la planificació. A partir d'aquí, es va decidir establir dos releases coincidint amb el lliurament de les PACs 2 i 3. I donat que entre ambdues hi havia un mes de diferència, es va establir sprints de 2 setmanes, quedant dos sprints i la release I, seguit de dos sprints més i la release II.

Un cop estan definides les grans àrees, les històries d'usuari i els sprints de que disposarà el projecte, es pot començar a construir el mapatge. Seguint la filosofia d'SCRUM, les històries d'usuari a realitzar primer són les que responen a les funcionalitats més importants de l'aplicació, que normalment són les que responen a les característiques bàsiques que permeten dur a terme l'objectiu de la web. S'ha de tenir en compte que després d'un mes de desenvolupament (dos sprints de dos setmanes) s'ha de llençar una versió funcional de l'aplicació. Per tant, es va considerar que les característiques més importants eren pràcticament totes les funcionalitats que tenien a veure amb presentar un sistema d'anuncis funcional, i les funcionalitats bàsiques de l'administrador, que serien accedir a una administració i poder crear categories.

Com s'acostuma a fer tradicionalment, aquest mapatge es materialitza mitjançant post-its, marcant l'evolució que tindrà el projecte:



Mapatge d'històries d'usuari duta a terme pel projecte

En la imatge anterior es pot apreciar l'estructura del mapatge. En post-its de color taronja es veuen les grans àrees del projecte, les quals marquen verticalment el seu abast. En l'eix horitzontal, les 4 primeres línies de post-its grocs marquen els quatre sprints definits pel projecte. Els post-its de color verd marquen el moment de les releases.

La darrera línia és un espai anomenat *Frozen Features*, que correspon a funcionalitats que no han quedat descartades pel projecte, i per tant pot ser ignorada dins de l'actual mapatge.

4.6 Estimació

El mètode SCRUM determina que s'ha d'estimar l'esforç de desenvolupament que requerirà cada història d'usuari, per tal de veure el valor de negoci que aporta aquella funcionalitat en funció de la dificultat per desenvolupar-la. Aquesta visió permet al client final determinar si val la pena fer una història en un moment concret, o si és millor avançar-la o retardar-la dins del procés de desenvolupament.

En aquest punt aquest projecte partia amb la desavantatge en quant a que no existia experiència prèvia, o molt poca, amb les eines utilitzades en el projecte. Per tant era difícil poder estimar amb seguretat l'esforç.

4.7. Iteracions

A continuació es resumeix com van quedar les històries d'usuari dins les iteracions. Cada iteració es presenta en un quadre on s'indica el nom de la història, l'àrea que afecta, i l'esforç estimat. Pel que fa a aquest últim, es valora en dos números: El primer són els punts d'esforç, cadascun dels quals es correspon a 3 hores. El segon correspon al risc, un factor que pot fer que hi ha hagi sorpreses i que l'esforç necessari pugui augmentar.

4.7.1 Iteració I

La iteració I s'havia de dur a terme del 6 al 19 d'octubre. Es va dur a terme del 6 al 29 d'octubre.

Història d'usuari	**Àrea**	**Esforç**
Crear anuncis sense registrar-se	Sistema d'anuncis	8/1
Veure anuncis sense registrar-se	Sistema d'anuncis	3/0
Contactar anunciant	Sistema d'anuncis	3/0

4.7.2 Iteració II

La iteració II s'havia de dur a terme del 20 d'octubre al 2 de novembre. Es va dur a terme del 30 d'octubre al 15 de novembre

Història d'usuari	**Àrea**	**Esforç**
Editar anunci propi sense estar registrat	Sistema d'anuncis	3/0
Marcar anuncis favorits sense estar registrat	Sistema d'anuncis	5/0

Introduir fotos als anuncis	Sistema d'anuncis	5/3
Marcar anuncis com a fraudulents	Sistema d'anuncis	3/1

4.7.3 Iteració III

La iteració II s'havia de dur a terme del 3 al 16 de novembre. Es va dur a terme del 16 de novembre al 10 de desembre

Història d'usuari	**Àrea**	**Esforç**
Accés a àrea privada	Administració	4/2
Eliminar anuncis	Administració	2/1
Gestionar categories d'anuncis	Administració	3/1
Navegar per ciutat	Localització	5/3
Crear anuncis per ciutat	Localització	3/1
Registrar-se com a usuari	Sistema d'usuaris registrats	6/3

4.7.4 Iteració IV

La iteració II s'havia de dur a terme del 17 al 30 de novembre. Es va dur a terme del 11 al 31 de desembre

Història d'usuari	**Àrea**	**Esforç**
Cercar anuncis per text	Sistema d'anuncis	3/1
Gestionar anuncis fraudulents	Administració	4/1
Crear anunci com a usuari registrat	Sistema d'usuaris registrats	5/1
Marcar anuncis favorits com a usuari registrat	Sistema d'usuaris registrats	3/0
Gestionar els meus anuncis	Sistema d'usuaris registrats	4/1

5. Aplicació del BDD en el Projecte

Com s'ha vingut explicant al llarg del document, un dels grans objectius del projecte era desenvolupar-lo aplicant pràctiques de BDD. En aquest sentit PHP disposa de dos eines molt conegudes. La primera és **Behat**, que com s'ha descrit al llarg del document serveix per construir tests funcionals basats en les històries d'usuari. El segon és l'anomenat **phpSpec**, i és una eina per crear tests unitaris. En un primer moment es va considerar fer servir les dues eines, però finalment es va haver de desestimar per dos motius principals. El primer és que la demanda de temps en l'aprenentatge i ús de la primera, Behat, va consumir la majoria del temps que s'hi podia dedicar al BDD. El segon motiu, també important, és que degut a la naturalesa de l'arquitectura, la qual utilitzava en gran mesura el propi ecosistema de Symfony, no feia imprescindible la necessitat de tests unitaris per provar classes. Òbviament sempre és interessant disposar del màxim nombre de tests possibles, però la base de tests i de comprovació del funcionament de la aplicació ja es cobria prou amb els tests funcionals de Behat. En conclusió, l'aplicació del BDD ha sigut en la besant purament funcional dels tests.

5.1 De les històries a les *features*

La gran avantatge de Behat és que es “parla” l'idioma del desenvolupament àgil. El fet de disposar d'històries d'usuari fa pràcticament immediat la definició de les features, que no pas la creació de les mateixes. Resumidament, una història d'usuari es pot correspondre a una feature, i així s'ha procurat que fos en aquest projecte.

Però tot i això, la creació d'una feature requereix molta més feina degut a que la importància d'aquesta no està en la descripció de la història, si no en els **Escenaris** que conté, que són realment els encarregats de validar el funcionament de l'aplicació web. Per tal d'avançar en els features calia definir els diferents escenaris que permetessin provar el correcte funcionament de la funcionalitat definida a la història. Aquests escenaris es corresponen de precondicions, accions i respostes avaluables per Behat.

A continuació es mostra un exemple de feature de Behat:

```
Feature: Search ads by text
    In order to be able to find ads of my interest
    As a visitor or registered user
    I need search them from the text appearing in the title or the description

    Scenario: I search for items and I find them
         When I search for "billar"
         Then I should see "Mesa de billar"
```

```
        And I should see there 1 ads


  Scenario: I search for items and I do not find them
       When I search for "hervidor"
       Then I should see "No hay anuncios publicados"
```

Fragment de feature de Behat extreta del projecte SegundoUso.org

Aquest exemple correspon a una de les feature del projecte, concretament la que representa la història *Cercar anuncis per text*. S'aprecia com a la capçalera es troba la pròpia descripció de la Feature amb motiu bàsicament informatiu, i a continuació es despleguen dos escenaris.

El primer escenari es correspon a la possibilitat de que un usuari cerqui un anunci escrivint una paraula clau, i que existeixi algun anunci que inclogui una paraula clau al seu títol o la seva descripció. El segon escenari simula la possibilitat contrària, que l'usuari indiqui un text que no existeixi en els anuncis publicats. En ambdós escenaris trobem accions i resultats avaluables. En el primer escenari per exemple, quan diu `When I search for "billar"` respon a l'acció que s'ha introduït "billar" dintre del camp de cerca de la pàgina web. En el resultat esperat on indica `Then I should see "Mesa de billar"` es refereix a que un cop feta l'acció de cercar, la pàgina resultant hauria de mostrar aquest text.

Com s'explicava en l'apartat 3.2, la feature de Behat pretén ser el pont entre l'àrea de negoci i desenvolupament, i això fa que la intenció és que els arxius de features siguin llegibles i comprensibles per persones no tècniques. Aquest fet fa que calgui una capa més per a que una simple frase pugui ser interpretada per Behat per processar un test. Aquesta capa es defineix en els ja esmentats **Context**, dels quals se'n parlarà més endavant.

Abans però, queda un característica que cal mencionar. En l'exemple anterior s'ha mostrat accions i resultats esperats, però no hi ha cap precondició. En el cas de la feature tractada, totes les precondicions estan indicades en un Escenari especial anomenat `Background`. El Background pot presentar un o més precondicions i/o accions que són comunes per tots els escenaris definits en una feature. A continuació es mostra el Background definit a la feature que s'està tractant.

```
  Background:
      Given I am on the homepage
        And there are the following municipalities:
          | name              |
          | Barcelona         |
        And there are the following categories:
          | name              |
          | Muebles           |
```

```
            | Electrodomésticos |
            | Menaje            |
          And there are the following ads:
            | title             | category    | municipality |
            | Mesa de billar    | Muebles     | barcelona    |
            | Mecedora          | Muebles     | barcelona    |
```

Background de Behat extret del projecte SegundoUso.org

Es pot apreciar un seguit de precondicions. La primera és `I am on the homepage`, referint-se a que l'usuari que pren protagonisme en aquesta història d'usuari es troba a la home. Les següents precondicions són d'un tipus molt especial i molt important. Aquestes precondicions que contenen part de la informació en tabla es refereixen a informació que la pàgina web a de contenir en un hipotètic escenari. Behat s'encarregarà de guardar aquesta informació a la base de dades per a que quan s'executi l'escenari la web disposi d'un mínim d'informació de mostra en la que poder-se basar.

Per exemple, quan diu `there are the following categories` es refereix que ha la base de dades han d'existir les categories citades a la taula, en aquest cas Mobles i Electrodomèstics. Aquest tipus de precondicions són molt importants perquè són la manera que Behat té d'introduir informació a la base de dades, cosa imprescindible per poder posar en pràctica els tests.

Si s'ajunten els dos fragments indicats fins ara obtenim la següent feature:

```
Feature: Search ads by text
    In order to be able to find ads of my interest
    As a visitor or registered user
    I need search them from the text appearing in the title or the description

    Background:
        Given I am on the homepage
          And there are the following municipalities:
            | name              |
            | Barcelona         |
          And there are the following categories:
            | name              |
            | Muebles           |
            | Electrodomésticos |
            | Menaje            |
          And there are the following ads:
            | title             | category    | municipality |
            | Mesa de billar    | Muebles     | barcelona    |
```

```
          | Mecedora          | Muebles     | barcelona    |

  Scenario: I search for items and I find
       When I search for "billar"
       Then I should see "Mesa de billar"
        And I should see there 1 ads

  Scenario: I search for items and I do not find
       When I search for "hervidor"
       Then I should see "No hay anuncios publicados"
```

Feature de Behat extreta del projecte SegundoUso.org

En l'exemple complet es pot apreciar la lògica de funcionament. Veiem que a les precondicions del background s'insereixen a la base de dades dos anuncis, un amb el títol “Mesa de billar” i l'altre amb el títol “Mecedora”. Si analitzem els dos escenaris descrits, en el primer veiem que es busca anuncis que continguin la paraula “billar” i espera trobar-ne, degut a que s'ha creat un anunci anomenat “Mesa de billar”. En el segon escenari es cerca per la paraula “hervidor”, la qual s'espera no trobar. Cap dels dos anuncis creats al background conté aquesta paraula, i per tant no es trobarà. Quan s'executi aquest test, tots dos escenaris haurien de complir-se. Si no ho fan probablement serà perquè l'aplicació no està funcionant com s'espera.

5.2 La construcció dels Contexts

En l'apartat anterior s'esmentaven els contexts com la capa de lògica que hi ha entre la feature i el propi framework de Behat, i és on es programen realment les instruccions que s'han de dur a terme amb cada comprovació.

Behat disposa de diferents contexts, i dintre del propi projecte de SegundoUso.org s'utilitzen tres contexts diferents: el FeatureContext, el DataContext i el WebContext. Cadascun d'aquests encapsula un tipus de funcionalitat concreta. Serán explicats amb més detall en l'apartat 6.2.4.

A continuació es copia un fragment de WebContext:

```php
<?php

namespace SegundoUso\BehatBundle\Context;

use ...

class WebContext extends MinkContext implements KernelAwareInterface
{

    ...

    /**
     * @When /^I search for "([^"]*)"$/
     */
    public function iSearchFor($term)
    {
        return array(
            new When('I fill in "FLAG_search-input" with "'.$term.'"'),
            new When('I press "FLAG_search-button"'),
        );
    }

    ...

}
```

Fragment de la classe WebContext extreta del projecte SegundoUso.org

Amunt es pot observar un exemple de traducció de feature a un mètode de la classe WebContext que és l'encarregada de fer l'acció de cercar una paraula per trobar un anunci. En aquest exemple es distingeixen tres parts:

1. La anotació `@When /^I search for "([^"]*)"$/`: Aquesta anotació està escrita en llenguatge Gherkin i es correspon a la instrucció escrita en la feature, i que conté una expressió regular per interpretar qualsevol variable.
2. La definició de la funció PHP `public function iSearchFor($term)`: que materialitza la instrucció Gherkin i que emmagatzema la lògica a processar. Es veu que la funció té un paràmetre anomenat `$term` i que es correspon amb el valor que es passi a la instrucció Gherkin segons la expressió regular.
3. La pròpia lògica del mètode: En aquest cas, el mètode conté dos noves instruccions expressades en codi Gherkin `I fill in ...` i `I press ...`. Aquestes faran les accions d'omplir un camp concret de l'HTML i de prémer un botó concret, indicats com a paràmetre dins de la mateixa instrucció, `FLAG_search-input` i `FLAG_search-button` respectivament. Aquestes dues instruccions estan definides a l'hora en un altre Context, el MinkContext, del que s'ha parlat a l'apartat 3.2.3, el qual estén WebContext.

5.3 Posant-ho en pràctica

Com s'ha explicat en el punt 2 d'aquest document, en la introducció al BDD, aquesta pràctica neix des d'una altra pràctica, el TDD (Test Driven Development). Una de les màximes consignes del TDD és que abans que començar a programar cal començar a construir el test. El BDD està pensat per a que s'apliqui el mateix principi. A l'hora el BDD també és una pràctica iterativa, i l'objectiu quan es fa un test és que aquest sigui verd, és a dir, que totes les proves siguin satisfactòries.

Les passes que s'ha procurat seguir en tot moment en aquest procés han sigut les següents, iterativament per cada història d'usuari.

1. Crear la feature a partir de la història d'usuari
2. Determinar els escenaris mínims que verifiquin que els objectius de la història d'usuari es compleix. Per ara, només es definiran les precondicions, accions i resultats esperats per un escenari de la feature. A l'hora de definir-los, s'ha d'intentar reaprofitar instruccions que haguessin sigut creades anteriorment per altres features o escenaris. En cap moment s'haurà treballat amb el context encara.
3. Executar els tests. L'execució dels tests respondrà amb error, o més probablement, indicant que Behat no té cap funció per donar resposta a la instrucció de la feature. Behat proposarà l'estructura pel mètode del Context.
4. S'implementaran els mètodes necessaris en els Contexts corresponents, per tal de donar suport a les instruccions definides a la feature.

5. Paral·lelament s'anirà desenvolupament el codi corresponent a la aplicació pròpiament dita, creant els controladors, classes, vistes que siguin necessaris. En aquest punt és necessari moltes vegades treballar en paral·lel amb el Context, ja que assíduament el propi Context està molt vinculat amb la mateixa vista, o inclús amb el model del sistema.
6. Un cop es creu que tant el Context com el codi implementat a l'aplicació estan desenvolupats, es tornarà a executar el test esperant resultats. Si aquest passa correctament voldrà dir que les funcionalitats s'han implementat amb èxit. Si falla, s'ha d'investigar el motiu del problema i solucionar-ho. Es tornarà a executar el test fins que sigui satisfactori.
7. Quan un escenari sigui satisfactori, el procés tornarà a repetir-se per un nou escenari de la feature, en cas que en tingui.

5.4 Executant els tests

Per executar els tests és necessari disposar de features i dels contexts corresponents. També caldrà tenir un entorn preparat amb una eina d'automatització. En el cas de SegundoUso.org aquesta és Sahi.

A continuació es pot apreciar el resultat del test executat per la feature `Search ads by text` que s'està tractant.

```
Feature: Search ads by text
    In order to be able to find ads of my interest
    As a visitor or registered user
    I need search them from the text appearing in the title or the description

  Background:                                    # features/search_ads_by_text.f
eature:6
    Given I am on the homepage                   # SegundoUso\BehatBundle\Contex
t\WebContext::iAmOnHomepage()
    And there are the following municipalities: # SegundoUso\BehatBundle\Contex
t\DataContext::thereAreFollowingMunicipalities()
      | name      |
      | Barcelona |
    And there are the following categories:      # SegundoUso\BehatBundle\Contex
t\DataContext::thereAreFollowingCategories()
      | name              |
      | Muebles           |
      | Electrodomésticos |
      | Menaje            |
    And there are the following ads:             # SegundoUso\BehatBundle\Contex
t\DataContext::thereAreFollowingAds()
      | title          | category | municipality |
      | Mesa de billar | Muebles  | barcelona    |
      | Mecedora       | Muebles  | barcelona    |

  Scenario: I search for items and I find        # features/search_ads_by_text.f
eature:22
    When I search for "billar"                   # SegundoUso\BehatBundle\Contex
t\WebContext::iSearchFor()
    Then I should see "Mesa de billar"           # SegundoUso\BehatBundle\Contex
t\WebContext::assertPageContainsText()
    And I should see there 1 ads                 # SegundoUso\BehatBundle\Contex
t\WebContext::iShouldSeeThatMuchAds()

  Scenario: I search for items and I do not find   # features/search_ads_by_tex
t.feature:28
    When I search for "hervidor"                     # SegundoUso\BehatBundle\Con
text\WebContext::iSearchFor()
    Then I should see "No hay anuncios publicados" # SegundoUso\BehatBundle\Con
text\WebContext::assertPageContainsText()

2 scenarios (2 passed)
13 steps (13 passed)
0m35.175s
```

Resultat de l'execució de la feature Search ads by text

Quan s'executa un test podem veure les instruccions que es processen correctament i les que fallen per un codi de colors. Els instruccions satisfactòries es veuran en verd, i les que fallen es veuran en vermell. En cas de que alguna instrucció s'ignori (això passa quan no s'avalua una instrucció perquè ha fallat una anterior en un mateix escenari) es veurà de color blau clar.

A continuació es veurà el resultat en cas de que el primer escenari hagués fallat.

```
  Scenario: I search for items and I find        # features/search_ads_by_text.feature:2
2
    When I search for "billar"                   # SegundoUso\BehatBundle\Context\WebCon
text::iSearchFor()
    Then I should see "Mesa de billar"           # SegundoUso\BehatBundle\Context\WebCon
text::assertPageContainsText()
      The text "Mesa de billar" was not found anywhere in the text of the current page.
    And I should see there 1 ads                 # SegundoUso\BehatBundle\Context\WebCon
text::iShouldSeeThatMuchAds()

  Scenario: I search for items and I do not find   # features/search_ads_by_text.featur
e:28
    When I search for "hervidor"                     # SegundoUso\BehatBundle\Context\Web
Context::iSearchFor()
    Then I should see "No hay anuncios publicados" # SegundoUso\BehatBundle\Context\Web
Context::assertPageContainsText()

2 scenarios (1 passed, 1 failed)
13 steps (11 passed, 1 skipped, 1 failed)
0m31.71s
```

Resultat de l'execució de la feature `Search ads by text` en cas de fallida

5.5 Durant el procés de desenvolupament

Com s'ha exposat al llarg del document, i arrel de l'ús de mètodes àgils, SegundoUso.org ha tingut un procés de desenvolupament incremental. Exemple d'això és el fet que a la primera release és publiqués un producte funcional, però que durant iteracions posteriors el codi que funcionava hagués de ser modificat per introduir noves funcionalitats. Òbviament això és un risc, però eines com Behat ajuden a prevenir aquest tipus de problemes.

A l'hora de desenvolupar o implementar una nova funcionalitat es parteix de la base que tots els tests existents funcionen. Això permet que el desenvolupador tingui la seguretat que l'aplicació és estable. Si el desenvolupador comença a modificar codi i executa els tests, si aquests fallen sabrà que el que fa fallar el sistema és quelcom modificat per ell.

Aquest principi ha sigut molt útil a l'hora de desenvolupar de manera incremental. Després de la primera release s'havien de modificar paquets del codi que funcionaven correctament, amb el risc que deixessin de funcionar. Executar els tests i veure'ls fallar ha sigut l'alarma que ha permès adonar-se que alguna cosa havia deixat de funcionar.

5.6 Dificultats trobades a l'hora d'aplicar BDD

Començar a aplicar un nou mètode o pràctica des de zero és difícil. Aplicar BDD significa fer un canvi de mentalitat a l'hora d'enfocar els problemes que plantegen un desenvolupament. Especialment a l'inici del projecte, quan Behat no té cap Context, la sensació del programador és que inverteix tot el temps en crear els tests, especialment aquells que tenen a veure amb a la persistència a la base de dades. Tot i això, a mesura que el projecte avança i els Contexts van implementant mètodes la eficàcia millora.

Un altre motiu de dificultat ha sigut que per certes accions no ha sigut possible desenvolupar tests. Alguns casos ha sigut accions on prenien protagonisme els emails, o la interacció de l'usuari amb aquests. Behat no soporta, o al menys no ha sigut possible dur-ho a terme, el tests d'emails. Similar ha passat a l'hora de testejar la pujada d'imatges al servidor.

6. Arquitectura

En aquest apartat es detallen l'estructura de directoris i paquets de la aplicació en primer lloc, i el disseny final de la base de dades.

6.1 Estructura

La estructura de la aplicació web SegundoUso.org segueix les línies que marca el propi framework de Symfony2. En ella destaquen quatre carpetes principals: **app**, carpeta que emmagatzema configuracions i els arxius de Kernel de l'aplicació, així com arxius de cache i logs generats per la mateixa; **src**, carpeta que conté tots els bundles que formen el lògica de negoci de l'aplicació, i les vistes que donen la interfície del *site*; **vendor**, directori on Composer automàticament instal·la les llibreries definides que la aplicació necessita; **web**, és el directori arrel HTTP i conté tots els arxius accessibles mitjançant aquest protocol, així com l'arxiu PHP que és el *Front Controller* de tota la aplicació.

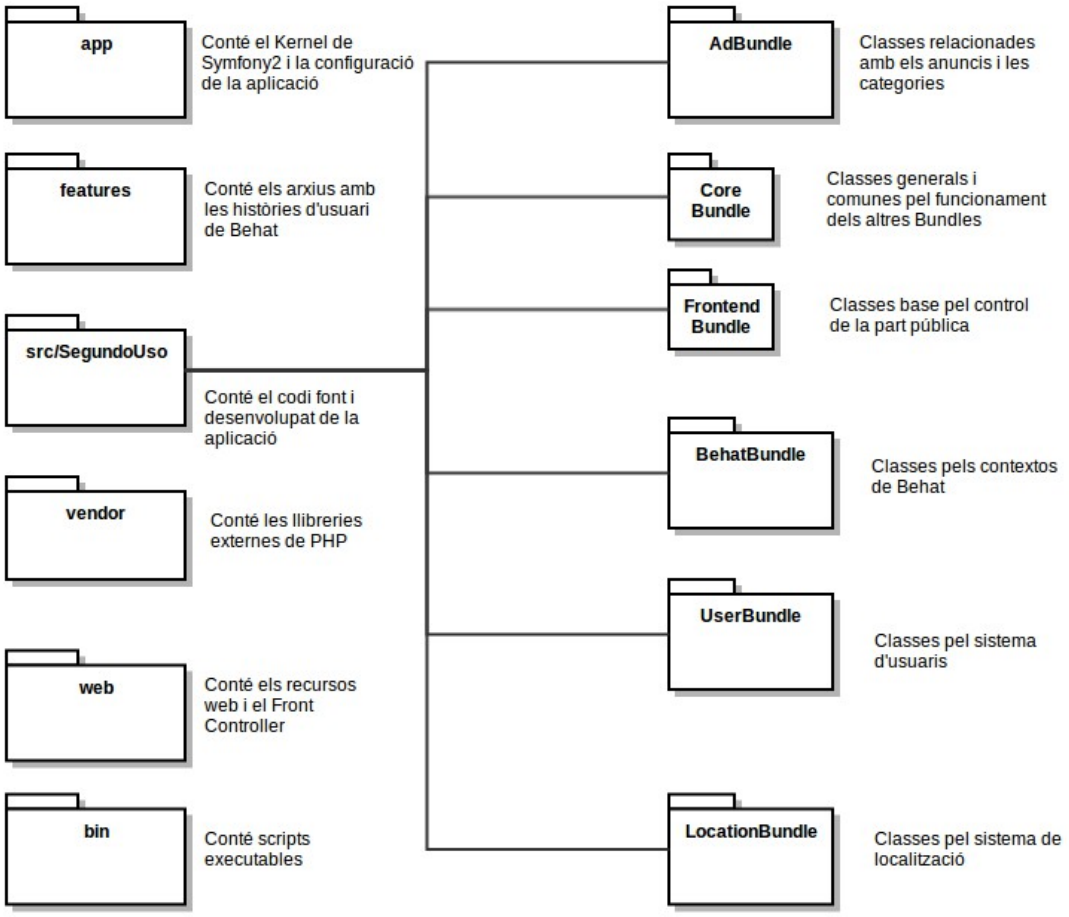


Diagrama de la estructura de directoris principals de l'aplicació SegundoUso.org

A continuació s'analitza amb més detall aquesta estructura:

- **app**: conté el *Kernel* de Symfony2 que és la base del funcionament del sistema. Aquí es defineixen els Bundles instal·lats a l'aplicació.

  Dins també es troben:

  - la carpeta **config** que conté els arxius de configuració. Aquests arxius estan en un format anomenat Yaml, amb l'extensió yml. Trobaríem els arxius de configuració del framework, dels bundles per cadascun del entorns de desenvolupament, a més dels arxius de seguretat i de *routing*.
  - la carpeta **logs** que conté els logs que genera la aplicació fent servir el component Monolog
  - la carpeta **cache** on s'emmagatzemen els arxius *caxejats* pel framework.
  - la carpeta **Resources** conté l'estructura d'HTML bàsica utilitzada en totes les pàgines de l'aplicació.
  - l'arxiu **console** el qual executa els comandaments de consola del framework.
- **features**: conté els arxius de Behat on s'especifiquen les històries d'usuari que seran utilitzades en els tests de validació.
- **bin**: és una carpeta generada durant la instal·lació de Composer que conté arxius binaris per l'execució de llibreries. En el cas del projecte és utilitzat principalment per l'execució de l'arxiu que arranca els tests de validació de Behat.
- **src/SegundoUso**: conté els arxius fonts desenvolupats pel projecte. Està estructurat en b*undles* seguint l'arquitectura definida per Symfony2. En un pròxim apartat s'entrarà en detall en el contingut dels bundles.
- **vendor**: conté totes les llibreries externes fetes servir en el projecte, des del propi framework Symfony2 o Behat, fins a llibreries per la gestió d'usuaris o manipulació d'imatges.

  Per la manipulació d'aquestes dependències es fa servir la llibreria Composer, que com s'ha explicat anteriorment, és una eina per gestionar les dependències d'altres llibreries d'una aplicació.

  Les llibreries més importantes utilitzades en el projecte són les següents:

  - Symfony2: framework de l'aplicació
  - Behat: framework per BDD
  - FosUserBundle: Eina per gestionar el sistema d'usuaris de l'aplicació

- Imagine: Llibreria per manipular imatges
- Doctrine: Llibreria molt lligada a Symfony utilitzada pel sistema de mapatge i accés a les Bases de Dades

- **web**: conté els diferents recursos accessibles per HTTP i que són necessaris per la correcte visualització de la web, com són els arxius d'estils css i els de javascript. També conté els diferents arxius Front Controller per cadascun dels entorns de Symfony: desenvolupament, test i producció.

6.2 Bundles de SegundoUso

Seguint l'estructura marcada per Symfony2, el codi específic de l'aplicació i que marca la lògica de negoci s'ha d'ubicar dins de la carpeta **src**, i que a l'hora ha de contenir una carpeta amb el nom de la organització, en aquest cas **SegundoUso.org**, per motius d'estructuració que no té rellevància tractar aquí. A dins d'aquesta carpeta es trobaran les altres grans estructures de codi, els *Bundles*, explicats en detall en el punt 3.1.3.

S'ha dividit el codi font de la aplicació segons una lògica de domini. Els dominis principals són els **Anuncis**, els **Usuaris**, la **Localització**, i els tests funcionals de BDD amb **Behat**. Cadascun d'aquests dominis té un Bundle propi. En el resum del contingut de cadascun es podrà apreciar que tots tenen una estructura de carpetes molt similar, la qual es descriu en profunditat a continuació. Val la pena esmentar que aquests bundles tenen tots dos carpetes: una anomenada DataFixtures i l'altra DependencyInjection. La primera conté unes classes que serveixen per omplir de continguts base la web durant el desenvolupament. Com poden ser categories, anuncis, municipis, usuaris, etc. La segona conté un seguit de classes que serveixen a Symfony per configurar cadascun dels bundles de manera específica.

6.2.1 AdBundle

L'AdBundle és el paquet que conté tota la lògica relacionada amb els **Anuncis** i és la base principal de l'aplicació. A més de tenir les classes que treballen directament amb anuncis també conté la lògica utilitzada per les **Categories**, que s'han considerat una Entitat estretament lligada als Anuncis. Les carpetes que componen aquest Bundle són:

- **Controller**: conté les classes que l'hora són els controladors de l'aplicació. Trobem tan controladors utilitzats tan en el Frontend com en el Backend.
- **Entity**: conté les classes que representen les taules de la base de dades en classes PHP, anomenades Entitats. També hi guarda els repositoris, que són classes que es comuniquen amb la base de dades per tornar els registres guardats en forma d'entitats. Les Entitats creades per aquest bundle són: *Ad*, pels anuncis; *Advertiser*, per usuaris no registrats; *Category,* per les categories;

Image, per les imatges dels anuncis; *Mark*, pels marcatges d'anuncis fraudulents. S'entrarà en detall sobre l'explicació de les Entitats al punt 6.4 sobre el disseny de la base de dades.

- **Event:** conté les classes del tipus Event que estan lligades al Component de Symfony Event Dispatcher. En l'apartat 7.2 d'aquest document s'aprofundeix sobre aquest tema.
- **EventListener**: conté les classes del tipus Listener. Com els Events, estan lligats al component Event Dispatcher. Els Listeners permeten executar una lògica concreta quan a l'aplicació es produeix un esdeveniment concret. En l'apartat 7.2 d'aquest document s'aprofundeix sobre aquest tema.
- **Exception**: carpeta destinada a emmagatzemar classes d'excepcions per aquest Bundle.
- **Form**: conté les classes que emmagatzemen la lògica de gran part dels formularis en l'aplicació. Symfony2 permet configurar formularis des de classes PHP. D'aquesta manera permet tres coses:
    1. Disposar d'un sistema de configuració de formularis lligat a Entitats i així facilitant la validació.
    2. Separar capes de lògica de manera que la vista només rebrà un objecte formulari i quedarà més neta i serà més mantenible.
    3. Facilitar la reutilització de formularis en diferents vistes evitant duplicació de codi.
- **Model**: conté classes molt vinculades a les entitats però que són una capa abstracte més que facilita l'obtenció d'Entitats des dels controladors.
- **Resources**: Conté diferents recursos:
    - **config**: aquesta carpeta guarda els arxius de configuració i de mapatge de la base de dades.
        - **doctrine**: conté els arxius yaml de mapatge que vincula les Entitats amb les taules de la bases de dades.
        - **services**: conté els arxius yaml on es defineixen els serveis de Symfony. S'entra en més detall sobre aquests en l'apartat 7.1,
        - **routing**: conté els arxius que configuren les rutes de la URL per les pàgines que es corresponen amb el Backend i que estan vinculades a la gestió d'anuncis.
    - **public**: conté arxius que seran portats a la carpeta web de l'arrel i que seran accessibles via HTTP.
    - **views**: conté els arxius de maquetació HTML en un sistema de plantilles anomenat Twig.

- **Util**: Conté classes que no tenen vinculació a cap altre carpeta i que presenta funcions d'utilitat per a altres classes.

6.2.2 UserBundle

El UserBundle és una extensió del paquet **FOSUserBundle**, una de les terceres llibreries utilitzades. La llibreria completa es trobarà dins de la carpeta **vendor**, però aquí s'hi troba també com a Bundle ja que s'encarrega d'estendre la llibreria base adaptant el seu funcionament i disseny a les necessitats del projecte actual. Dintre del d'aquest Bundle està contingut tot el que correspon al login d'usuaris, ja sigui al Frontend o al Backend, i l'àrea privada de cada usuari registrat.

Les carpetes que el componen són:

- **Controller**: Conté dos controladors. El Controlador **AccountController** que s'encarrega de gestionar les accions que tenen a veure amb la secció del compte de l'usuari registrat quan aquest està logat, i el Controlador **SecurityController** que hereta de la mateixa classe en el Bundle original per gestionar els dos formularis de login, tant el del Backend com el de la zona privada de l'usuari en el Frontend.
- **Entity**: Conté les entitats i repositoris relacionades amb aquest Bundle. En aquest cas, existeixen dues entitats, User i Group, però només l'entitat User s'està utilitzant. Aquesta hereta la gran majoria d'atributs de la classe base al FosUserBundle.
- **Resources**: Conté diferents recursos:
    - **config**: aquesta carpeta guarda els arxius de configuració i de mapatge de la base de dades.
        - **doctrine**: conté els arxius yaml de mapatge que vincula les Entitats amb les taules de la bases de dades.
        - **routing**: arxiu que configura les diferents rutes del FosUserBundle dins l'aplicació.
    - **views**: conté els arxius de maquetació HTML en un sistema de plantilles anomenat Twig.

En l'apartat 6.3.1 es parla amb més detall de l'ús de la llibreria FosUserBundle i justifica la estructura d'aquest Bundle.

6.2.3 LocationBundle

El LocationBundle és un paquet que conté la lògica aplicada a la localització dels anuncis. Les carpetes que el componen són:

- **Controller**: Conté un controlador per defecte que és l'encarregat de gestionar el canvi de municipi que disposa la web.
- **Entity**: Conté l'entitat de municipis anomenada Municipality.
- **Model**: Conté el gestor de municipis.
- **Resources**:
    - **config**: conté el mapatge de la classe Municipality i l'arxiu de configuració de les rutes.
    - **views**: conté una vista que es correspon amb el selector de municipi.

El LocationBundle és molt petit, però es va decidir separar-lo en un Bundle independent amb per mantindre els diferents dominis el més desacoblats possibles. En cas de creixement del projecte i les opcions per localitzar anuncis geogràficament, pot fer fer créixer aquest Bundle molt ràpidament.

6.2.4 BehatBundle

El BehatBundle és un paquet específicament creat per contenir les classes on implementar la lògica de Behat per les proves de validació. Dins de la carpeta **Context** es troben els diferents contexts que gestionen les accions. Hi ha tres contexts:

- **FeatureContext**: és un context bàsic que permet implementar Behat dins de Symfony i gestionar el resseteig de les dades a la base de dades cada cop que s'executa un test.
- **DataContext**: és un classe que s'encarrega de tota la lògica que manipula dades entre la base de dades i Behat, com per exemple crear anuncis, usuaris o categories pels tests.
- **WebContext**: és la classe que conté totes les accions definides a les features de la carpeta feature a l'arrel de l'aplicació. WebContext estén la classe MinkContext (veure punt 3.3).

L'objectiu de crear un Bundle diferenciat és el fet de poder encapsular aquesta lògica fora dels altres Bundles.

6.2.5 Altres Bundles

Existeixen dos Bundles més, **FrontendBundle** i **CoreBundle**. Tots dos es van definir al començament del projecte sense encara tenir clar quin paper desenvoluparien en l'arquitectura.

FrontendBundle conté arxius comuns que jugarien un paper dins del que seria Frontend estandard o part pública de la web. Finalment bàsicament conté la configuració de les rutes per cridar les accions dels controladors i algunes vistes.

El CoreBundle va nèixer amb l'objectiu contenir els arxius que fossin utilitzats per diferents Bundles. En l'estat actual de l'aplicació només conté la llibreria **Twitter Bootstrap** com a Framework de maquetació HTML i CSS.

6.2.6 Aclaracions sobre l'estructura de Bundles

És important esmentar que l'estructura de Bundles s'ha pensat tenint en compte que aquest és un projecte en expansió. Probablement ara el FrontendBundle i el CoreBundle no tenen una raó de ser, però amb el creixement de la aplicació molt probablement així acabi sent.

L'objectiu d'aquests Bundles és que en el futur siguin el més independents possibles entre ells, segons la filosofia per la qual es van pensar els bundles a Symfony.

6.3 Terceres llibreries

En l'ecosistema de PHP actual, i més des del naixement de Composer, s'ha estès molt l'ús de terceres llibreries, precisament per les facilitats que Composer aporta en la seva gestió. En el cas de SegundoUso.org s'ha fet servir diferents llibreries més o menys lligades a Symfony2.

6.3.1 FosUserBundle

FosUserBundle és una llibreria OpenSource que s'integra dins de Symfony aportant un seguit de funcionalitats molt comunes a tot tipus de web pel que fa a la gestió d'usuaris. Alguns exemples podrien ser el registre d'usuaris, formulari de login, l'edició de les dades, recordatori de contrasenya, entre d'altres.

Com tota dependència de Composer s'instal·la dins de la carpeta vendor.

La forma d'integrar les opcions que porta dins de l'aplicació Symfony es realitza mitjançant la creació d'un nou bundle amb, en aquest cas UserBundle, que hereti la classe FosUserBundle original. A partir d'aquí es podran estendre entitats i controladors per tal d'adaptar-se a les necessitats del projecte.

Adaptacions de FosUserBundle a SegundoUso

Pel que fa als controladors, com es veu en el fragment de codi a continuació, s'ha estès el controlador SecurityController i s'ha reescrit el mètode `renderLogin()` permetent que en funció de la ruta (`_route`) de la pàgina on es troba l'usuari, es retorni una vista o una altre. Això permet mostrar dos formularis diferents, un pel login de l'administració i un altra pel login ordinari dels usuaris.

```
<?php
namespace SegundoUso\UserBundle\Controller;
```

```php
use FOS\UserBundle\Controller\SecurityController as BaseController;

class SecurityController extends BaseController
{
    public function renderLogin(array $data)
    {
        $requestAttributes = $this->container->get('request')->attributes;
        if ('admin_login' === $requestAttributes->get('_route')) {
            $template = sprintf('SegundoUsoUserBundle:Admin:login.html.twig');
        } else {
            $template = sprintf('SegundoUsoUserBundle:Security:login.html.twig');
        }
        return $this->container->get('templating')->renderResponse($template, $data);
    }
}
```

Exemple de codi del UserBundle estenen el FosUserBundle

Un altra exemple d'extensió es produeix en les Entitats. Com es veu en l'exemple següent la Entitat User es estesa per la entitat del mateix nom del FosUserBundle. En aquest cas l'adaptació que es fa d'aquesta entitat és la inclusió de nous atributs en la classe, com són `$ads` i `$favouriteAds`. Aquests contindran objectes de la classe Ad i que estaran vinculats a través d'una relació de la classe de dades amb la instància corresponent de l'objecte User.

```php
<?php
namespace SegundoUso\UserBundle\Entity;

use FOS\UserBundle\Model\User as BaseUser;
use Doctrine\Common\Collections\ArrayCollection;
use SegundoUso\AdBundle\Entity\Ad;

class User extends BaseUser
{
    protected $id;
    protected $ads;
    protected $favouriteAds;

    public function __construct()
    {
        parent::__construct();
        $this->roles = array('ROLE_USER');
```

```
        $this->ads = new ArrayCollection();
        $this->favouriteAds = new ArrayCollection();
    }
    ...
}
```

Exemple de codi del UserBundle estenen el FosUserBundle

6.4 Disseny de la base de dades

Com s'ha explicat al llarg d'aquest apartat, el bundle amb més importància de la aplicació és l'AdBundle. El diagrama de relacions entre entitats deixa constància:

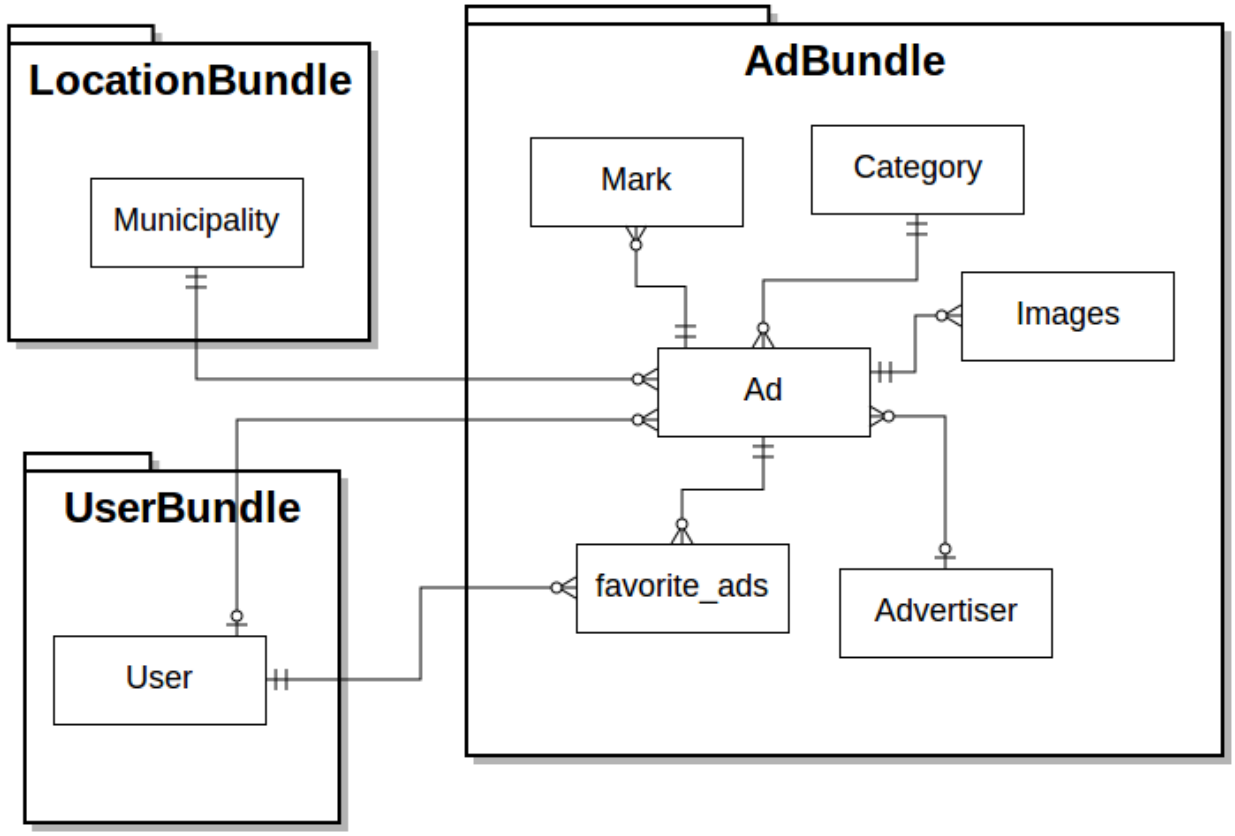


Diagrama de relacions entre entitats a SegundoUso.org

En el diagrama anterior es veuen totes les relacions entre entitats. El diagrama és prou clar, però es poden fer els següents aclariments:

- **favorite_ads** es correspon a una taula relacional a la base de dades que relaciona anuncis i users, però no es correspon a cap Entitat de PHP. Doctrine s'encarrega de gestionar la relació N:M que tenen.
- Un Anunci pot tenir un **Advertiser**, que és una entitat que guarda un usuari visitant, o un **User**, un usuari registrat, però no totes dues. Es va crear l'entitat Advertiser per emmagatzemar l'email d'un usuari visitant que anuncia per tal d'evitar afegir un camp email a la Entitat **Ad**, ja que quan aquesta es vincula amb User aquest camp seria redundant.

7. Desenvolupament amb Symfony2 i bones pràctiques

Symfony2 i els components que el formen s'han desenvolupat fent servir l'ús de bones pràctiques de programació i patrons. Una de les avantatges d'utilitzar aquest framework és que aquest obliga en certa mesura a també fer servir bones pràctiques, fet que a la llarga és molt beneficiós pel projecte.

En aquesta secció es volen posar de relleu algunes de les bones tècniques que aplica Symfony2 i que per conseqüent s'han aplicat en el projecte. Els següents apartats no pretenen ser un tutorial sobre cap component, però si volen demostrar part dels coneixements tècnics adquirits en el framework.

7.1 Injecció de Dependències i serveis

Injecció de Dependències, o Dependency Injection en anglès, és un patró de desenvolupament dissenyat per evitar la instanciació d'objectes dins d'altres objectes i evitar així l'acoblament entre classes.

Per gestionar les dependències a Symfony2 es fa servir un Contenidor d'Injecció de Dependències o **Contenidor de Serveis**. Aquest és un objecte PHP que s'encarrega d'emmagatzemar instàncies d'altres d'objectes que podran ser utilitzades al llarg de l'aplicació. A l'hora permet que tots els objectes definits dins del contenidor siguin passats com a paràmetres a altres objectes del propi contenidor. L'objectiu permet que els objectes que es defineixen únicament un cop. Les classes definides al contenidor de dependències s'anomenen S**erveis**.

```
# src/SegundoUso/AdBundle/Resources/config/services/model.yml
parameters:
    seguso.advertiser_manager.class: SegundoUso\AdBundle\Model\AdvertiserManager
    seguso.advertiser.class: SegundoUso\AdBundle\Entity\Advertiser
services:
    seguso.advertiser_manager:
        class: %seguso.advertiser_manager.class%
        arguments: [@doctrine.orm.entity_manager, %seguso.advertiser.class%]
```

Exemple de de creació d'un servei mitjançant a Symfony2

L'arxiu yaml anterior defineix un servei anomenat `seguso.advertiser_manager`. També defineix dos paràmetres que ajudaran a definir el servei. Així aquest servei serà una instància de la classe `SegundoUso\AdBundle\Model\AdveriserManager`, i tindrà dos arguments en la seva instanciació: el primer serà un servei ja configurat al Contenidor, `@doctrine.orm.entity_manager`, que és un servei que crea la llibreria Doctrine per gestionar la persistència de les entitats. El segon serà una cadena de text,

definit al mateix arxiu. S'ha de remarcar que el tipus d'argument depèn del prefix amb el que vingui. Els serveis van precedits d'una @, mentre que les cadenes de text van entre %.

7.1.1 Beneficis

Els Beneficis de fer servir un Contenidor de serveis amb injecció de dependències són:

1. Centralitza l'accés a les funcionalitats concretes de les classes.
2. Només instancia els objectes si són requerits durant el procés, evitant així un consum innecessari de recursos.

7.2 Event Dispatcher i listeners (Observer Pattern)

Symfony2 utilitza un patró molt conegut: l'*Observer.* En aquest cas el *Subject* és anomenat Disparador d'esdeveniments o *Event Dispatcher.* És un servei de Symfony2 que conté els *Observers*, anomenats *Listeners* o Escoltadors.

Els Escoltadors són afegits al Disparador indicant un nombre d'esdeveniment, de manera que aquest sabrà qui escolta què. El Disparador com a servei pont ser cridat per una gran quantitat d'objectes. Quan algún objecte el cridi per disparar li indicarà un esdeveniment. El Disparador llavors buscarà entre tots els Escoltadors associats, i els cridarà.

Aquesta utilitat s'ha utilitzat diferents cops en aquest projecte. Un cas ha sigut quan es vol enviar un email de confirmació a l'usuari que crea un anunci.

El següent codi mostra una classe amb un mètode que s'encarrega d'enviar aquest email.

```php
<?php
namespace SegundoUso\AdBundle\EventListener;

class AdConfirmationEmailListener
{
    public function onNewAdCreated(AdEvent $event)
    {
        $ad = $event->getAd();
        $advertiser = $ad->getAdvertiser();
        $message = \Swift_Message::newInstance()
            ->setSubject('Confirmación de anuncio en SegundoUso.org')
            ->setFrom('dev@segundouso.org')
            ->setTo($this->getEmail($ad))
            ->setBody(
                $this->twig->render(
                    'SegundoUsoAdBundle:Email:ad_confirmation_email.html.twig',
```

```
                    array('ad' => $ad)
                )
            )
            ->setContentType('text/html')
        ;
        $this->mailer->send($message);
    }
}
```

Exemple de Listener a SegundoUso.org

Es vol que aquest Listener quedi registrat a l'Event Dispatcher, per a que aquest última s'encarregui de notificar al Listener quan esdevé l'esdeveniment. Per fer-ho cal enregistrat el Listener com a servei indicant un paràmetre tags. En aquest exemple el Listener s'ha registrat per escoltar l'esdeveniment anomenat segudo.ad_create.completed.

```
# src/SegundoUso/AdBundle/Resources/config/services/listener.yml
services:
    seguso.listener.ad_conf_email:
        class: %seguso.listener.ad_conf_email.class%
        arguments: [@mailer, @twig]
        tags:
            - { name: kernel.event_listener, event: seguso.ad_create.completed, method:
onNewAdCreated }
```

Definició de listener com a servei a SegundoUso.org

En aquest moment el Listener podrà escoltar quan es produeix l'esdeveniment. Ara cal que l'Event Dispatcher avisi quan s'executa l'esdeveniment. Aquest fet tindrà lloc en quan l'anunci ha quedat correctament guardat a la base de dades. En el fragment següent es pot veure com el Dispatcher llança l'esdeveniment.

```
class DefaultController extends Controller
{
    public function createAction(Request $request)
    {
        ...
        if ($form->isValid()) {
            ...
            $adManager->updateAd($ad);

            $dispatcher->dispatch(SegundoUsoAdEvents::AD_CREATE_COMPLETED, new
AdEvent($ad));
```

```
            return $this->redirect($this-
>generateUrl('segundo_uso_frontend_ad_waiting_confirmation'));
        }

        return $this->render('SegundoUsoAdBundle:Default:create.html.twig', array(
            'form' => $form->createView()
        ));
    }
}
```

Exemple d'ús de Event Dispatcher a SegundoUso.org

7.2.1 Beneficis

L'Event Dispatcher és un servei molt poderós. Permet afegir lògica de negoci en punts clau d'un procés sense haver d'alterar la classe que dispara l'esdeveniment, ajudant a mantenir classes desacoblades. En l'exemple anterior es pot apreciar com s'ha afegit un lògica que no afecta el codi de l'acció que la llença. Si es vol afegir més lògica en aquest punt només caldria crear un nou Listener amb els corresponents processos.

Si es defineixen punts clau en qualsevol procés de l'aplicació i es llencen esdeveniments en aquests punts, es podria fer créixer la lògica de l'aplicació sense alterar les classes base, com són els controladors. Val a dir però, que l'Event Dispatcher també pot fer-se servir a qualsevol classe a la que se l'injecti mitjançant el contenidor de serveis.

7.3 Model i Gestors (Managers)

Symfony no és un framework MVC[6]. Tot i que sí implementa Controladors i Vistes, no implementa un Model clar. Tot i això, definir una capa de model pot ser molt útil. Encara que es pot treballar directament sobre els repositoris i els controladors, a l'hora de persistir els objectes s'estaria implementant lògica en el controlador que podria perfectament ser encapsulada en una altra classe. Aquesta classe seria el Gestor o *Manager*.

El Manager és una classe que encapsula tota la lògica que té a veure amb la persistència i la obtenció d'entitat. Seria doncs el Manager el que manipularia el repositori en comptes del Controlador.

```
<?php
namespace SegundoUso\AdBundle\Model;
use ...
class AdManager implements AdManagerInterface
```

6 FABIEN POTENCIER. *What is Symfony2*. [en línia]. http://fabien.potencier.org/article/49/what-is-symfony2 [data de consulta [09/01/2014]

```
{
    const AD_TOKEN_LENGTH = 16;
    const AD_PID_LENGTH = 64;

    protected $pidGenerator;
    protected $objectManager;
    protected $class;
    protected $repository;

    public function __construct(RandomStringGeneratorInterface $pidGenerator, ObjectManager $om,
$class)
    {
        $this->pidGenerator = $pidGenerator;

        $this->objectManager = $om;
        $this->repository = $this->objectManager->getRepository($class);

        $metadata = $om->getClassMetadata($class);
        $this->class = $metadata->getName();
    }

    public function createAd()
    {
        $class = $this->class;
        $ad = new $class;

        return $ad;
    }

    public function updateAd(AdInterface $ad, $andFlush = true)
    {
        if (null === $ad->getPid()) $this->setPublicId($ad);
        if (null === $ad->getToken()) $this->setToken($ad);

        $this->objectManager->persist($ad);
        if ($andFlush) {
            $this->objectManager->flush();
        }
    }
}
```

Fragment de la classe AdManager

En el fragment anterior podem veure com a la funció `__construct()` s'assignen les depèndencies del manager, necessari per després definir-lo com a servei. S'observen també dos funcions `createAd()` i

updateAd(), la primera encarregada de crear la entitat i la segona per persistir-la. UpdateAd() a més crida a altres funciones privades de la classe AdManager.

La conseqüència d'això és que el Controlador presenta un aspecte com el següent:

```
    public function createAction(Request $request)
    {
        /** @var $adManager \SegundoUso\AdBundle\Model\AdManager */
        $adManager = $this->get('seguso.ad_manager');
        /** @var $dispatcher \Symfony\Component\EventDispatcher\EventDispatcherInterface */
        $dispatcher = $this->get('event_dispatcher');

        $ad = $adManager->createAd();
        $ad->setPublished(false);

        $form = $this->createForm(new AdType($this->get('security.context')), $ad);

        $form->handleRequest($request);

        if ($form->isValid()) {

            $dispatcher->dispatch(SegundoUsoAdEvents::AD_CREATE_SUCCESS, new FormEvent($form,
$request));

            $adManager->updateAd($ad);

            $dispatcher->dispatch(SegundoUsoAdEvents::AD_CREATE_COMPLETED, new AdEvent($ad));

            return $this->redirect($this-
>generateUrl('segundo_uso_frontend_ad_waiting_confirmation'));
        }

        return $this->render('SegundoUsoAdBundle:Default:create.html.twig', array(
            'form' => $form->createView()
        ));
    }
```

Codi de l'acció createAction al DefaultController de l'AdBundle

En 16 línies de codi s'obté un controlador fàcil de llegir i d'interpretar. Primer obté els serveis que necessitarà, entre ells l'AdManager. Crea l'objecte Ad, el formulari, i retorna la vista l'objecte Response amb la vista. Té una condicional que valida si el formulari s'ha enviat i el processa. L'acció presenta gran flexibilitat ja que durant tot el procés es llencen 3 esdeveniments diferents.

7.3.1 Beneficis

L'exemple de controlador anterior és un exemple dels beneficis que aporta. Aconseguir tenir una aplicació amb codi net i fàcilment interpretable és una necessitat en les aplicacions web, i frameworks rigorosos com Symfony ajuden a aconseguir-ho.

A més, l'abstracció d'aquest lògica en Gestors facilitaria l'intercanvi de proveïdor de persistència. Per exemple, en el cas d'aquesta aplicació es fa servir Doctrine com a eina, però si en un futur es volgués fer un canvi a altres eines similars com Propel, gràcies a que l'`AdManager` implementa una interfície concreta, l'`AdManagerInterface`, es podria crear un altre AdManager que implementés aquesta interfície, i l'aplicació de SegundoUso.org continuaria funcionant, ja que tota la comunicació de persistència es fa a través del Gestor.

8. Entorn de desenvolupament i instal·lació

8.1 Requisits per l'entorn de desenvolupament

A continuació es detallen els requisits per tal de disposar de l'entorn de desenvolupament necessari que permeti visualitzar l'aplicació i els seus tests a l'ordinador local.

Les eines i programes necessaris per l'entorn són les següents:

- SO Linux Ubuntu 12.04 o superior
  (No s'ha testejat el funcionament de l'aplicació en altres sistemes operatius)
- Servidor Apache 2
- PHP 5.3.8 o superior
- MySQL 5.0 o superior
- git (Control de Versions)
- Composer (Gestor de paquets i dependències de PHP)
- Sahi (Eina per automatització de tests)

NOTA: Per poder procedir a la instal·lació de l'entorn serà necessari que l'usuari utilitzat tingui permisos d'administrador sobre l'ordinador.

8.2 Instal·lació de les eines

8.2.1 Instal·lació de l'entorn

El primer pas és disposar de l'entorn de desenvolupament anomenat LAMP (Linux-Apache-MySQL-PHP). L'objectiu d'aquesta secció no és explicar pas a pas com s'han d'instal·lar cadascun d'aquest. Per aquest motiu s'inclou un enllaç amb les instruccions corresponents:

- Instal·lació entorn LAMP : http://setupguides.blogspot.com.es/2012/04/install-lamp-in-ubuntu-1204.html

8.2.2 Instal·lació de git

Un cop es disposa de l'entorn, el següent pas és instal·lar git com a Sistem de Control de Versions. Es facilita un enllaç amb les corresponents instruccions pas a pas:

- Instal·lació de git : https://www.digitalocean.com/community/articles/how-to-install-git-on-ubuntu-12-04

  NOTA; L'enllaç presenta dos maneres d'instal·lar git. Per l'entorn creat per aquest projecte s'ha utilitzat la opció d'instal·lar git des dels arxius fonts. Tot i així, sembla més recomanable seguir la opció mitjançant `apt-get`.

8.2.3 Importar arxius de l'aplicació

Si s'ha seguit el primer enllaç en la instal·lació de l'entorn, la ruta del projecte estarà al directori `/var/www`. L'aplicació es guardarà en una carpeta anomenada `segundouso` dins del directori anterior. Per procedir a és necessari git. S'ha de procedir a obrir un Terminal (Crtl+Alt+T) i executar les següents comandes:

```
cd /var/www
git clone git@github.com:javierseixas/segundouso.git
```

Automàticament es crearà la carpeta `segundouso` i descarregarà el codi font de l'aplicació des de la plataforma Github.

8.2.4 Configurar el Virtual Host d'Apache

El següent pas és configurar el Servidor Apache per poder visualitzar la web en local mitjançant la adreça http://segundouso.local.

Cal executar les següents comandes per iniciar la creació del *Virtual Host* d'Apache.

```
cd /etc/apache2/sites-available
```

```
sudo vim segundouso
```

En el document que es crearà s'haurà de copiar el contingut següent:

```
<VirtualHost *:80>
        DocumentRoot "/home/javierseixas/www/segundouso/web/"
        DirectoryIndex app_dev.php
        ServerName segundouso.local
        ServerAlias segundouso.local

        <Directory "/home/javierseixas/www/segundouso/web/">
                AllowOverride All
                Allow from All
        </Directory>

        php_flag short_open_tag off
        php_value date.timezone "Europe/Madrid"

</VirtualHost>
```

A continuació s'ha d'activar el virtual host i es recarregar la configuració d'Apache:

```
sudo a2ensite segundo
sudo service apache2 reload
```

En aquest punt cal configurar els hosts:

```
sudo vim /etc/hosts
```

On s'ha d'incloure la següent línia::

```
127.0.0.1       segundouso.local
```

8.2.5 Crear la Base de Dades

En aquest punt caldrà crear la Base de Dades en Mysql. S'han d'executar les següents comandes:

```
mysql -u root -p
```

NOTA: L'usuari root és el per defecte però caldrà indicar l'usuari configurat a l'hora d'instal·lar MySQL.

El terminal demanarà la contrasenya de l'usuari. Un cop introduïda i arrancat MySQL:

```
CREATE DATABASE segundouso CHARSET utf8 COLLATE utf8_unicode_ci;
```

8.2.6 Instal·lació de les llibreries externes

SegundoUso.org funciona amb llibreries Open Source creades per tercers. La instal·lació es procedeix a fer-se a través de Composer. Per instal·lar Composer mateix i les llibreries s'han d'executar les següents instruccions:

```
cd /var/www/segundouso
curl -sS https://getcomposer.org/installer | php
php composer.phar update
```

Composer instal·larà totes les llibreries externes inclós el Symfony2. Al final de la instal·lació es requerirà la configuració de Symfony, explicada en el següent apartat.

8.2.7 Configurar Symfony2

Al final de la instal·lació de les llibreries Symfony requerirà la configuració dels paràmetres bàsics per les connexions de base de dades i smtp per enviar emails.

Symfony2 és el Framework sobre el qual s'està construint segundouso.org. Pel seu funcionament cal editar un fitxer per configurar la connexió a la Base de Dades:

```
Creating the "app/config/parameters.yml" file
Some parameters are missing. Please provide them.
database_driver (pdo_mysql):
database_host (127.0.0.1):
database_port (null):
database_name (symfony): segundouso
database_user (root):
database_password (null): <segonsusuari>
mailer_transport (smtp):
mailer_host (127.0.0.1):
mailer_user (null):
mailer_password (null):
```

```
locale (en): es
secret (ThisTokenIsNotSoSecretChangeIt):
```

NOTA: Per als paràmetres on no hi ha resposta s'ha de prémer ENTER.

NOTA: El valor a introduir a `database_password` dependrà de la contrasenya indicada per l'usuari en el moment d'instal·lar MySQL.

NOTA: Per un funcionament efectiu de l'aplicació en local caldrà incloure paràmetres propis de l'usuari de configuració SMTP per l'enviament de email. Aquest document no es proveïxen.

També caldrà donar permisos a dos carpetes on Symfony2 emmagatzema la cache i els logs de l'aplicació:

```
sudo chmod -R 777 /var/www/segundouso/app/cache
sudo chmod -R 777 /var/www/segundouso/app/logs
```

Per últim, caldrà crear l'esquema de la base de dades. Es farà servir un dels components de Symfony, el Console Component que permet executar accions des del terminal.

```
cd /var/www/segundo
app/console doctrine:schema:update –force
```

Arribat aquest punt, la aplicació hauria de ser operativa obrint un navegador amb la url http://segundouso.local/app_dev.php.

8.2..8 Instal·lar l'entorn de testing

Per l'entorn de testing són necessàries dos eines. La primera és el framework Behat, que ja ha sigut instal·lat a través de Composer anteriorment. La segona és Sahi, un eina que permet l'automatització de tests que pot treballar juntament amb Behat.

Per instal·lar Sahi, és recomanable seguir les instruccions al següent enllaç de la web oficial: http://sahi.co.in/w/using-sahi

El següent pas correspon a configurar Behat.

1. Dins del directori /var/www/segundouso, obrir l'arxiu behat.yml.dist amb qualsevol editor de text

2. On apareix el paràmetre base_url, canviar el valor per http://segundouso.local/app_test.php
3. Guardar el fitxer amb el nom behat.yml.

Per executar el tests de Behat caldrà:

1. Executar Sahi amb les següents comandes:

```
cd <sahi_root>/userdata/bin
./start_dashboard.sh
```

NOTA: En l'espai <sahi_root> s'ha d'indicar la ruta on s'ha instal·lat Sahi.

2. Y executar els tests mitjançant la següent comanda:

```
cd /var/www/segundo
app/console bin/behat
```

9. Projecció a futur

L'objectiu és que SegundoUso.org sigui una web que fomenti la reutilització d'objectes en una societat consumista. Existeix un inici de consciència social que comença a preocupar-se d'aquest tipus d'afers, factor que fa pensar que el portal té un públic potencial. Es vol continuar desenvolupant la web implementant noves funcionalitats amb l'objectiu a curt plaç de crear una aliança amb altres webs similars.

En un futur pròxim la idea seria crear una API que vincules la web amb una aplicació mòbil, per poder gestionar totes les funcionalitats des de qualsevol *smartphone*.

El desig final és que SegundoUso s'acabi convertint en una organització sense ànim de lucre, present a diferents països i que fomenti la seva filosofia del reciclatge.

9.1 Màrqueting

Com a projecte online el canal de comunicació natural és Internet. La intenció és iniciar campanya a través del compte de twitter ja existent (@segundouso_org) i possiblement un de facebook. S'investigaran associacions de locals que fomentin la reutilització, i s'oferirà opcions de col·laboració.

10. Conclusions i experiències de desenvolupament

Desenvolupar un projecte aplicant eines poc conegudes i sense experiència en els mètodes utilitzats no és fàcil. Per crear un fons de coneixement s'ha recorregut a multitud de fons parlant sobre el tema i s'ha adquirit un coneixement molt ben valorat personalment. A partir d'aquí posar-los en pràctica ha sigut un desafiament.

La planificació àgil mitjançant el mapatge de les històries d'usuari és una tècnica interessant. Molt probablement existeix una certa dificultat a l'inici, degut a que a vegades és difícil identificar on una història d'usuari passa a convertir-se en una altra. També és una arma de doble fil centrar-se únicament en les parts funcionals d'una aplicació perquè és fàcil oblidar els requisits no funcionals[7]. Donar-se compte un cop iniciat el projecte dóna problemes de planificació i de temps.

Un cop enllestida la planificació s'ha començat a aplicar el BDD. Aquest potser ha sigut un dels aspectes més difícils de complir. Aplicar una pràctica de "*Tests First*" és un canvi de mentalitat molt gran i que costa d'assumir. Més encara quan aquesta s'aplica a partir d'una eina poc coneguda com es Behat. L'inici de desenvolupament amb Symfony ha sigut en comparació més senzill, tot i que assimilar el funcionament d'elements com l'Event Dispatcher o l'ús de formularis ha sigut de les parts més crítiques.

A mesura que el projecte avançava traduir les històries d'usuari a features de Behat es feia més senzill. També la implementació de tests, a mesura que s'anaven formant els contexts bàsics per persistir les entitats. La part més crítica del projecte ha sigut fins a la primera release, tal que la segona release ha començat amb una base sòlida.

Ha sigut també en la segona release quan s'ha comprovat una de les grans virtuts del desenvolupament incremental, i en especial l'ús de BDD. A partir de la tercera iteració els requisits demanaven aplicar canvis sobre gran part del controladors i formularis. Gràcies a l'execució de les proves de validació de Behat s'ha pogut detectar ràpidament els errors en el funcionament causats pels canvis, i corregir-los. Molt possiblement molts d'aquests errors s'haguessin publicat en producció si no hagués sigut per les proves de Behat.

Es pot concloure de manera personal l'assoliment reeixit dels objectius: assimilar i familiaritzar-se amb l'ús i filosofia dels mètodes àgils, aprendre el funcionament de BDD i posar-lo en pràctica, i aprendre i aprofundir en el coneixement d'un framework com Symfony i el seu ecosistema.

7 Requisits no funcionals: http://www.methodsandtools.com/archive/archive.php?id=113

10.1 Retrospectiva

Siguen fidel a una de les pràctiques del mètode Scrum, es fa a continuació una breu retrospectiva per tal de trobar els punts que s'haguessin pogut millorar en la pràctica.

Per ordre cronològic d'aplicació, un dels primers punts a millorar seria la presa de consciència de planificar també els requisits no funcionals, que en aquest projecte han provocat una desfase de temps força important. També en quan a planificació, s'hagués hagut de posar més èmfasi en el control i compliment de les estimacions en l'esforç de les històries. En defensa d'aquest factor val a dir la dificultat d'estimar de forma fiable les històries degut a la falta d'experiència en les eines treballades. L'aplicació de Behat podria haver sigut millorable, ja que han quedat alguns requisits sense estar correctament coberts pels tests.

Es pot mencionar també que després de la indagació sobre XP i Scrum, s'arriba a la conclusió que l'aplicació en aquest projecte és una petita part de tot el que impliquen aquestes eines. A més, es pot considerar que l'aplicació d'Scrum es pot considerar molt minsa, ja que les tècniques aplicades poden pertànyer perfectament a l'Extreme Programming.

Per últim, hagués sigut interessant haver planificat en un inici del projecte un simulacre de canvi de requisits. Una de les grans avantatges d'*agile*, en especial de l'XP, és la gran flexibilitat davant de les noves demandes dels clients. En aquest projecte, donat que no hi havia un client real per canviar els requisits, no s'ha pogut posar de manifest la teòricament bona resposta dels mètodes àgils davant aquest tipus d'imprevistos.

Annex 1. Glossari

A continuació es defineixen breument termes i acrònims utilitzats en el present document necessaris per a la seva correcte comprensió.

Abast: es refereix a l'obtenció d'informació requerida per començar un projecte, amb les funcionalitats i característiques requerides pels actors.

Actor: Perfil d'usuari que utilitzarà una eina, en aquest cas la web desenvolupada.

Agile: Terme curt per referir-se a Desenvolupament àgil de programari

Backend: Espai privat d'una pàgina o aplicació web des d'on usuaris administradors poden gestionar diferents continguts d'aquella aplicació.

Background: En Behat es refereix a una o més precondicions i/o accions que són comunes per tots els escenaris definits en una feature.

BDD: Veure Behaviour Driven Development.

Behat: Framework PHP aplicat a la creació de tests funcionals i optimitzat per a la pràctica de BDD.

Behaviour Driven Development: De l'anglès Desenvolupament Guiat per Comportament, és una pràctica dintre dels mètodes àgils, i evolucionat del TDD, que promou l'ús del mateix llenguatge entre la part de desenvolupament i la part de negoci en un projecte, motivant l'ús d'eines per test com fa TDD.

Bundle: En Symfony2 es refereix a un paquet de programari que conté lògica del programari, entre altres arxius, i estructura el funcionament del programari en construcció.

Capifony: Eina desenvolupada en Ruby i basada en Capistrano que permet el desplegament d'una aplicació en un altre entorn com és el de producció.

Composer: eina emprada al llenguatge PHP per gestionar les dependències d'una llibreria o aplicació envers altres llibreries.

Controlador: Classe de PHP contenidora de mètodes que són els encarregats de processar una lògica determinada per una acció determinada, normalment cridada a partir d'una URL concreta.

Cookie: Informació emmagatzemada en els navegadors web per les pàgines web.

Desenvolupament àgil de programari: grup de mètodes pel desenvolupament de programari que es basen en el desenvolupament de software de manera iterativa i incremental i que cerca solucionar els problemes existents per altres mètodes com el de desenvolupament en cascada.

Doctrine: Llibreria de PHP encarregada de comunicar una aplicació amb les bases de dades intentant imitar el paradigma de la programació orientada a objectes a partir del mapatge de taules en classes.

Escenari: En Behat es refereix a la simulació d'una situació on a partir d'unes precondicions i l'execució d'unes accions, s'espera un tipus de resultat avaluable.

Extreme Programming: Mètode de desenvolupament de programari dintre del grup de mètodes àgils que té l'objectiu de produir *software* de millor qualitat i millorant la productivitat a partir d'un seguit de pràctiques.

Feature: En Behat es refereix a una funcionalitat de la aplicació i que conté un conjunt d'escenaris que s'han de complir, amb l'objectiu d'avaluar el correcte funcionament de la mateixa.

Framework: és una plataforma dissenyada per simplificar el desenvolupament de programari. Porten incorporats diferents components, llibreries, compiladors i altres funcions que ajuden a un més ràpid desenvolupament de *software*, evitant repetir tasques en l'elaboració de diferents projectes, i per tant, permetent que els programadors es puguin centrar en desenvolupar les funcions de negoci que requereix el projecte.

Frontend: Espai públic d'una pàgina web on qualsevol usuari té accés.

Gherkin: Llenguatge utilitzat per Behat en la redacció de les *features*.i que permet la vinculació entre classes PHP amb lògica i un llenguatge llegible i comprensible per persones no tècniques.

Git: Sistema de Control de Versions.

Github: Servei web que ofereix emmagatzemar repositoris Git.

Història d'usuari: és un conjunt de frases expressades en llenguatge de diari que expliquen una tasca que ha de desenvolupar l'usuari d'una aplicació. Aquestes frases acostumen a respondre les preguntes “Qui”, “Què” i Per què”. L'objectiu de les històries d'usuari és definir les funcionalitats del sistema. A la pràctica i de manera tradicional s'acostuma a definir-les en *post-its*.

Iteració: Unitat de desenvolupament pròpia de l'Scrum. Espai temporal on s'assignen les tasques a fer pel desenvolupament d'un projecte de programari.

Mapatge d'històries d'usuari: Sistema que pretén ajudar a l'estructuració i millor avaluació de les històries d'usuari. Es basa en col·locar les històries al llarg de dos eixos, on l'eix horitzontal intenta organitzar el funcionament del sistema, i l'eix vertical defineix l'ordre d'acció i prioritat de les històries.

Namespace: En PHP, sistema nadiu del llenguatge a partir de la seva versió 5.3 que serveix per agrupar o encapsular arxius a nivell lògic.

PHP: Llenguatge de programació de part del servidor utilitzat per construir aplicacions web dinàmiques.

Product Owner: En Scrum persona o persones que representen la part del client dins d'un projecte.

Refactoring: Veure refactorització.

Refactorització: Acció de revisar un codi amb l'objectiu de millorar-lo, ja sigui optimitzant el seu comportament com millorant la seva llegibilitat.

Release: Estat d'un programari en el qual s'entén que està preparat per ser publicat.

Servei: A Symfony2 un servei es refereix a la instància d'una classe que conté una funció específica i que ha sigut definida i és gestionada pel contenidor de serveis i que pot ser requerida en qualsevol part del sistema que tingui accés al contenidor.

Scrum: Dintre del desenvolupament àgil de programari, mètode enfocat a la gestió de projectes.

Sistema de Control de Versions: Sistema que permet emmagatzemar les diferents versions d'un programari, facilitant la seva gestió, control i desenvolupament.

Site: Anglicisme que fa referència a l'espai web desenvolupat.

Sprint: Veure iteració.

Stakeholder: Veure actor.

Symfony2: Framework PHP que utilitza el paradigma de programació orientat a objectes i el patró vista-controlador

Tasca: Es refereix a les accions i funcionalitats que els actors que utilitzaran la aplicació hauran de realitzar.

TDD: Veure *Test Driven Development*.

Test: Procediment que avalua el funcionament d'una aplicació o d'una part de la mateix a partir de la obtenció d'un resultat avaluable extret del codi a provar.

Testing: Pràctica que consisteix en la creació de tests o proves.

Test Driven Development: De l'anglès Desenvolupament Guiat per Tests, és una pràctica pròpia de l'Extreme Programming que promou el desenvolupament de programari partint de la creació d'un test per provar una funcionalitat, i continuant pel desenvolupament del codi en sí mateix. Un tercer pas correspondria a la refactorització del codi per tal d'optimitzar-lo.

Twitter Bootstrap: Col·lecció d'eines per la creació visual de pàgines i aplicacions web, les més utilitzades són un sistema de plantilles per la maquetació HTML. Estils visuals mitjançant CSS i una llibreria de javascript que facilita certes tasques comunes.

XP: Veure Extreme Programming.

Yaml: És un format de serialització de dades fàcilment llegible i comprensible per humans, molt utilitzat per estructurar els diferents arxius de configuració i mapatge en Symfony2.

Annex 2. Bibliografia

PROYECTOSAGILES.ORG. *Vídeos cortos sobre planificación ágil*. [en línia]. http://www.proyectosagiles.org/videos-cortos-planificacion-agil [data de consulta: 26/09/2013]

WIKIPEDIA. *Behaviour-driven development.* [en línia]. http://en.wikipedia.org/wiki/Behavior-driven_development [data de consulta: 27/09/2013]

AGILEALLIANCE.ORG. *Story Mapping*. [en línia]. http://guide.agilealliance.org/guide/storymap.html [data de consulta: 01/10/2013]

WIKIPEDIA. Scrum. [en línia]. http://en.wikipedia.org/wiki/Scrum_%28development%29 [data de consulta: 28/09/2013]

WIKIPEDIA. *Software testing*. [en línia]. http://en.wikipedia.org/wiki/Software_testing [data de consulta: 05/10/2013]

BEHAT. *Behat documentation*. [en línia]. http://docs.behat.org/ [data de consulta: 05/10/2013]

SYMFONY BOOK. *Symfony documentation*. [en línia]. http://symfony.com/doc/current/index.html [data de consulta: 01/10/2013]

EXTREMEPROGRAMMING.ORG. *The rules of Extreme Programming*. [en línia]. http://www.extremeprogramming.org/rules.html [data de consulta: 28/10/2013]

MONTAIN GOAT SOFTWARE. *Differences between Scrum and Extreme Programming*. [en línia]. http://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming [data de consulta: 06/01/2014]

WIKIPEDIA. *Dependency Injection*. [en línia]. http://en.wikipedia.org/wiki/Dependency_injection [data de consulta: 08/01/2014]

FABIEN POTENCIER. *What is Symfony2*. [en línia]. http://fabien.potencier.org/article/49/what-is-symfony2 [data de consulta [09/01/2014]

DIGITAL OCEAN. *How to install git on ubuntu 12.04*. [en línia]. https://www.digitalocean.com/community/articles/how-to-install-git-on-ubuntu-12-04 [data de consulta: 09/01/2014]

SETUP GUIDES. Install LAMP in Ubuntu 12.04. [en línia]. http://setupguides.blogspot.com.es/2012/04/install-lamp-in-ubuntu-1204.html [data de consulta: 09/01/2014]

SAHI. *Using Sahi*. [en línia]. http://sahi.co.in/w/using-sahi [data de consulta: 09/01/2014]

METHODS AND TOOLS. *Non-Functional Requirements: Do User Stories Really Help?* [en línia] http://www.methodsandtools.com/archive/archive.php?id=113 [data de consulta: 09/01/2014]

Annex 3. Vita

Javier Seixas és desenvolupador web. Va iniciar-se en el món de la programació l'any 2004 com a passatemps i va formalitzar el seu interès per aquesta disciplina iniciant estudis universitaris a la UOC l'any 2005. Durant aquest temps ha treballat en vàries empreses dedicades al desenvolupament web, en primer lloc amb el llenguatge ASP, fins que al 2007 va iniciar la transició cap a PHP degut a la seva popularitat. Des de llavors ha viscut l'evolució d'Internet en general i del llenguatge PHP en particular. Siguen conscient de l'especialització que ha s'estat visquen Internet, el seu objectiu és especialitzar-se en la programació de qualitat en la filosofia *Clean Code* i en el desenvolupament àgil de programari. En 2012 inicia el projecte SegundoUso.org on pretén agrupar la seva passió pel desenvolupament amb la seva inquietud i interès social i mediambiental.