UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Clean Architecture amb PHP

Treball de Final de Grau

–

Memòria

Autor:
Ferran Martín Sànchez

Titulació:
Grau en Enginyeria Informàtica
Enginyeria del Software

Directora:
Gemma Casamajó

Empresa:
l'Apòstrof SCCL

Ponent:
Ernest Teniente

Departament:
Enginyeria de Serveis i Sistemes
d'Informació (ESSI)

26 d'octubre de 2017

Agraïments

Vull començar aquest document donant les gràcies a totes aquelles persones que han participat en aquest projecte o que s'hi han interessat d'alguna forma, en especial a:

Les cooperativistes de l'Apòstrof, per oferir-me aquesta oportunitat i confiar en mi per realitzar aquest projecte, i sobretot a la Gemma i a en Martí que han format la *Comissió Intranet* i tots junts hem estat fent el seguiment i organització d'aquest projecte.

Al meu ponent *Ernest Teniente*, per tota l'ajuda i consells que m'ha donat, no només durant aquest projecte, sinó durant total la carrera.

A la meva família, per tot el suport donat, i molt especialment a la meva parella Mariona, sense el seu suport i ajut no hauria sigut possible realitzar aquest projecte.

Resum

Des de sempre ha sigut important portar un control dels processos en les empreses. I a partir de l'auge dels Sistemes d'Informació es va fer més evident que era un factor clau per aconseguir avantatges competitius. Avui en dia totes (o quasi totes) les grans empreses ja tenen algun tipus de software per controlar els seus processos interns, per tant el fet de tenir-ho ja no aporta un avantatge competitiu respecte als altres, sinó que és un requisit.

Però això canvia amb les mitjanes i petites empreses, les quals molts cops no tenen els recursos necessaris per accedir a aquests softwares. Per tant arribar a aconseguir implantar un software per controlar els seus processos, sí que pot significar un avantatge respecte als altres. A més, si ens centrem en empreses del tercer sector i del cooperativisme, es fa quasi impossible trobar algun software que reflecteixi la forma d'entendre l'economia, i el paper d'aquesta i de les empreses dins la societat.

És per això que l'objectiu del present treball és el de crear una primera versió d'un software de gestió de feines per a la cooperativa l'Apòstrof SCCL que també inclogui conceptes d'àmbit humà i social. El projecte té la visió d'oferir l'eina a cooperatives i empreses del tercer sector.

Resumen

Desde siempre ha sido importante tener un control de los procesos internos en las empresas. I a partir del auge de los Sistemas de Información se hizo más evidente que era un factor clave para conseguir ventajas competitivas. Hoy en día todas (o casi todas) las grandes empresas ya tienen algún tipo de software para llevar el control de sus procesos internos, por lo que el echo de tener este software ya no aporta una ventaja, sino que se ha convertido en un requisito.

Pero esto cambia con la medianas y pequeñas empresas, que muchas veces no tienen los recursos necesarios para acceder a estos softwares. Por lo que conseguir implantar un software para llevar el control de sus procesos sí que puede significar tener una ventaja competitiva respeto a los demás. A parte, si nos centramos en empresas del tercer sector i del cooperativismo, se hace casi imposible encontrar un software que refleje la forma de entender la economía, i el papel de ésta i las empresas dentro la sociedad.

Es por esto que el objetivo del presente trabajo es el de crear una primera versión de un software de gestión de trabajo para la cooperativa l'Apòstrof SCCL que también incluía conceptos de ámbito humano i social. El proyecto tiene la visión de ofrecer la herramienta a otras cooperativas i empresas del tercer sector.

Abstract

It has always been important to keep track of processes in companies. Starting with the Information Systems boom it has become even more obvious that it's a key factor in having competitive advantages. Today, most large companies already have some type of software to control their internal processes, therefore, it is not an advantage but a requirement to stay competitive.

The paradigm changes with small and medium-sized businesses, which often do not have the resources to access these software. Therefore, being able to implement a software to control their processes can in fact mean a competitive edge with respect to the others. In addition, if we focus on companies in the third sector and cooperativism, it is almost impossible to find some software that reflects how to understand their business economics and its role in relationship to their sector and society.

That is why the objective of this paper is to create a first version of a work management software for the cooperative l'Apòstrof SCCL that also includes concepts of human and social scope. The project has the vision of offering the tool to cooperatives and companies of the third sector.

Índex

1. Context

1.1 Contextualització

Aquest és un treball que es realitza en un conveni de cooperació educativa a l'empresa L'Apòstrof. Aquesta és una cooperativa de comunicació integrada per periodistes, lingüistes, dissenyadors, docents i desenvolupadors. Es treballa en equip i en xarxa, potenciant així la creativitat. Podem veure l'organització interna de l'Apòstrof de dues formes diferents; la primera seria veure que té dos nivells, els socis i els treballadors. I la segona seria veient que internament s'organitza amb àrees de feina (comercial, comunicació, text, disseny i desenvolupament). En conjunt, aquest funcionament prioritza l'horitzontalitat. Cada projecte té assignat un responsable i les decisions importants que afecten tota l'empresa es prenen en assemblea formada pels socis.

El present projecte sorgeix de la necessitat d'actualitzar i millorar l'actual intranet que es fa servir dins l'Apòstrof. La intranet de la qual es parteix està feta a mida, però ha quedat obsoleta, ja que es va crear fa uns 10 anys. A causa del fet que la base inicial no es va fer seguint uns bons estàndards de programació i qualitat, que no es va fer pensant a facilitar el seu manteniment i extensió a la llarga, i que els canvis que s'han anat fent ha estat de manera poc organitzada i sense previsió, aquesta intranet actualment és una eina poc fiable i estable. En l'actualitat, doncs, des de l'empresa tenen la necessitat de renovar aquesta eina per tal de crear-ne una que pugui ser fiable, estable, mantenible i extensible.

1.1.1 Motivació personal

Durant la meva última experiència laboral, abans de l'Apòstrof, vaig participar en el projecte de crear i mantenir una app força gran i complexa per a iOS. Durant aquest projecte, en un punt on teníem una app que fallava bastant, es va decidir refer-la i utilitzar una arquitectura neta. El resultat va ser una app molt estable (amb uns 800 usuaris diaris, només un 0.1% d'aquests tenien algun error que feia aturar la app) i fàcilment mantenible i

extensible.

Vist el resultat d'implementar una arquitectura neta en una app de iOS, vaig voler provar si una implementació d'aquesta arquitectura en un projecte web amb PHP donaria els mateixos resultats, i si valdria la pena.

1.2 Actors

1. **L'Apòstrof**
   L'Apòstrof (socis i treballadors) és el principal StakeHolder del projecte, ja que en seran els beneficiaris directes.
   Per tal d'agilitzar la presa de decisions, en aquest projecte s'ha creat una comissió interna de dues persones formada per la Gemma Casamajó (coordinadora de l'Apòstrof i directora del present TFG) i en Martí Làzaro (dissenyador i soci de l'Apòstrof), que faran el rol de *Product Owner*[1].
   La resta de treballadors de l'Apòstrof provaran regularment la intranet nova per tal de poder donar feedback com més aviat millor.

2. **Equip de Desenvolupament**
   Està format per dues persones, l'Asier Illarramendi (desenvolupador i treballador de l'Apòstrof) i jo, Ferran Martin (desenvolupador i treballador de l'Apòstrof). Tot i així, el projecte de la intranet el desenvoluparé principalment jo, ja que l'Asier haurà de seguir principalment realitzant projectes per a clients de l'Apòstrof.
   Per tant, les funcions de l'Asier dins aquest projecte seran principalment les de programar algunes parts del projecte, participar en la presa de decisions tècniques i realitzar *Code Reviews*[2] (tècnica per a assegurar la qualitat del codi, que s'explica més endavant).

3. **Clients de l'Apòstrof**
   Els clients també són un *StakeHolder* important, però de manera indirecta, ja que tot i que ells no utilitzaran la intranet, en la mesura

[1]Product Ower: `https://www.mountaingoatsoftware.com/agile/scrum/roles/product-owner`

[2]Code Reviews: `http://blogs.atlassian.com/2009/11/code_review_in_agile_teams_part_i/`

que aquesta ajudi a l'Apòstrof a millorar els seus processos interns, els clients se'n veuran beneficiats.

4. **Altres cooperatives**
   L'Apòstrof està situat dins d'un grup de cooperatives (Grup ECOS). Dins d'aquest grup es potencia la intercooperativitat. l'Àpostrof és de les úniques cooperatives del grup que actualment té una intranet feta a mida i que respon a bastants de les seves necessitats (tot i tenir les mancances que ja s'han comentat). Des del moment en què es va saber que l'Apòstof volia renovar i millorar la seva intranet, altres cooperatives del grup ja van demostrar el seu interès a poder utilitzar-la ells també. Això queda molt lluny de l'abast d'aquest projecte (com es veurà més endavant), però cal tenir-ho present a l'hora de dissenyar la intranet. És per això que s'han inclòs com a *StakeHolders*.

5. **Ponent del projecte**
   El ponent del projecte és l'Ernest Teniente Lopez que pertany a l'especialitat d'Enginyeria de Software, i és professor de la FIB. El seu rol serà el de comprovar que es compleixin els objectius establerts al final de cada una de les fases.

2. Formulació del problema

Tot i que com hem vist a l'apartat anterior la intranet actual no és fiable i seria molt difícil modificar-la i ampliar-la, sí que la utilitzarem per analitzar les funcionalitats que té implementades actualment.

Tot seguit es descriuran quines d'aquestes funcionalitats es mantindran tal com estan i quines s'hauran de millorar i/o corregir. També es descriuran noves funcionalitats que es volen incorporar a la nova intranet, però que la intranet actual no té. Alhora, es farà una neteja de funcionalitats innecessàries, ja que hi ha certes funcionalitats de la intranet actual que ja no es fan servir que no s'implementaran. Aquestes funcionalitats no es descriuen en aquest document, ja que no tenen cap impacte en el projecte.

2.1 Intranet actual

La intranet actual està organitzada per àrees que agrupen certes funcionalitats. Tot seguit descrivim aquestes àrees amb les funcionalitats que agrupen.

1. **Àrea d'administració**
   Aquesta àrea engloba totes les funcionalitats necessàries per a l'administració de la intranet.

   - **Treballadors:** Permet crear, editar i esborrar treballadors.
   - **Usuaris:** Permet crear, editar i esborrar els usuaris que tenen accés a la intranet i amb quins permisos. Poden estar o no vinculats a un treballador.
   - **Avisos:** Permet crear avisos amb un missatge i usuaris destinataris. Aquests avisos es mostraran als destinataris a l'entrar a la intranet.

   També s'haurà de millorar molt la funcionalitat de **Bases horàries**. Aquesta permet la creació i edició de les bases horàries dels empleats, que actualment és molt rudimentària.

2. **Àrea de negoci**
   Aquesta àrea conté funcionalitats que permeten definir com s'organitzen les feines del personal de l'Apòstrof.

   - **Àrees:** Permet definir en quines àrees s'organitza l'Apòstrof. Actualment són comunicació, text, disseny, economia solidària, docència i desenvolupament.
   - **Habilitats:** Permet definir un llistat d'habilitats les quals els usuaris podran marcar a l'entrar hores.

3. **Àrea de gestió de feina**
   Aquí és on es troben totes les funcionalitats que els empleats de l'Apòstrof necessiten al dia a dia.

   - **Clients:** Permet gestionar tota la informació sobre els clients.
   - **Feines Productives:** Permet gestionar tota la informació de les feines que es realitzen pels clients de l'Apòstrof. A les feines se'ls hi poden associar despeses.
   - **Feines Reproductives:** Permet gestionar tota la informació de les feines que es realitzen de forma interna i no aporten un benefici directe a l'empresa. Per a les cooperativistes de l'Apòstrof poder analitzar dades d'aquestes feines és igual d'important que de les productives.
   - **Hores:** Permet als treballadors entrar les hores que treballes. Es poden entrar hores tant de feines Reproductives com de feines Productives. En el cas de les feines Productives, també s'han d'indicar les habilitats usades per a realitzar la tasca.
   - **Factures:** Amb aquesta funcionalitat es poden crear les factures per les feines realitzades.

   La funcionalitat de **Feines Productives** es vol ampliar, i introduir el concepte de Partides, que permetrà definir vàries partides per a cada feina, i associar aquestes partides a una àrea de feina. Les despeses també s'associarien a les partides. Amb tot això es podran extreure dades millor organitzades.

   Una altra funcionalitat actual que es vol millorar és la de **Feines Reproductives**. Com s'ha explicat anteriorment, les feines reproductives

són les feines internes de l'Apòstrof que no aporten un benefici econòmic directe a l'empresa. Actualment aquestes feines només tenen un identificador, un nom i una descripció, i tot i que els treballadors poden entrar hores dedicades a aquestes, no poden definir a quina habilitat i/o àrea fan referència. Amb la nova intranet es vol incorporar que algunes feines reproductives (les que són prou grans) es puguin gestionar com una feina productiva; és a dir, associar-hi despeses, donar-hi una facturació estimada, entrar hores amb detalls, etc.

4. **Àrea de xarxa de treball**
   Totes les funcionalfuncionalitatsitas per gestionar la xarxa de treball de l'Apòstrof estan localitzades en aquesta àrea. Bàsicament serà on es tractin totes les dades dels **proveïdors** que fa servir l'Apòstrof. Actualment també hi ha la funcionalitat de col·laboradors, però s'ha decidit unificar les duos funcionalitats en una (proveïdors), ja que en realitat eren exactament el mateix.

5. **Àrea d'informes**
   Aquí es poden localitzar tots els informes que es fan servir per veure l'estat de l'empresa. Molts d'aquests informes actuals s'han de repensar, ja que no ofereixen dades útils, o no es generen correctament.

   - **Facturació:** informes sobre la facturació neta i total per anys, facturació entre mesos i facturació per clients.
   - **Despeses:** informes amb les despeses de feines tancades, i previsions sobre les despeses obertes.
   - **Productivitat:** informes amb les dades de productivitat; diferents resums d'hores segons alguns paràmetres. És important aclarir que tot i que el nom d'aquests informes és *productivitat*, en ells també s'analitzen les feines reproductives.
   - **Rendibilitat:** càlculs de preus per hora estimats i reals.

   Els informes en general es volen millorar força, i en concret hi ha una sèrie d'informes que ara mateix s'estan generant manualment (amb excel) que es volen automatitzar. Una millora important que es vol fer és poder analitzar millor les feines reproductives.

2.2 Noves funcionalitats

A part de les funcionalitats que la intranet ja té implementades, es volen afegir algunes més. Aquestes noves funcionalitats inclouen poder rectificar factures, afegir el concepte de "partides" a les feines, afegir un sistema de comentaris per les feines i poder gestionar els perfils d'usuaris (per ara estan predefinits i no es poden modificar).

També s'està pensant a desenvolupar noves funcionalitats més grans, que només es faran si abans es finalitzen les anteriors. Aquestes són:

- funcionalitat de gestió de projectes per tal de millorar aquesta dins l'empresa i controlar millor la càrrega de treball dels treballadors.
- capa multi cooperativa amb la qual es voldria oferir la intranet com a servei a altres cooperatives.

2.3 Objectius del projecte

Com es pot veure en els apartats anteriors, el software que es vol desenvolupar és una Intranet amb moltes funcionalitats que permetrà a l'Apòstrof portar el control dels seus processos interns referents a les feines i que ajudarà a la presa de decisions mitjançant els informes generats.

Per això l'objectiu principal d'aquest projecte és el de desenvolupar una primera versió d'aquest software. Aquesta primera versió ha de tenir implementades les funcionalitats mínimes necessàries per a poder deixar de fer servir la Intranet actual, i es deixaran la resta de funcionalitats per a versions posteriors.

Per tal de poder arribar a aquest objectiu principal, s'han definit quines haurien de ser aquestes funcionalitats mínimes:

- Funcionalitats bàsiques del sistema tals com Login, Logout i gestió d'usuaris.
- Poder crear fitxes de feines i introduir hores associades a aquestes feines i treballadors.

- Poder crear factures per les feines per ser enviades als clients.

2.4 Possibles obstacles

Com es pot veure, es pretén crear una intranet amb forces funcionalitats, per tant podem considerar que és un projecte força gran i ampli. En ser un projecte amb aquestes característiques serà fàcil que surtin molts obstacles durant la realització d'aquest. Per tant és difícil preveure'ls tots. Tot i així, sí que poden dir que els principals problemes per assolir els objectius del projecte segurament seran deguts a la falta de coneixement i experiència en el tema, a una data límit ajustada i a canvis en els requisits de les funcionalitats del sistema.

2.4.1 Obstacles amb la implementació de l'arquitectura Clean

Com el nom del TFG indica, i com s'ha explicat anteriorment. Es vol crear aquesta intranet utilitzant unai implementació d'una arquitectura de software Clean. Per tant ens podem trobar que dediquem més hores de les esperades a pensar i decidir respecte a això, ja que no és una arquitectura típica que s'ha estudiat a la carrera.

2.4.2 Obstacles de temps

Per diferents motius, podem necessitar més temps del que inicialment ens pensem. Per exemple, a l'Apòstrof no es tenen clar del tot tots els requisits del sistema, i es vol poder fer modificacions durant el desenvolupament a aquests. També ens podem trobar amb problemes de temps pel fet que per raons econòmiques de l'empresa s'hagin de dedicar més hores a altres projectes. En qualsevol dels casos, com s'explicarà més endavant, s'ha triat una metodologia de feina àgil, per tal de minimitzar aquests obstacles. I també es faran les hores extres necessàries per tal d'arribar als objectius mínims.

2.5 Abast

Un cop definits els objectius que volem assolir amb aquest projecte, queda clar que l'abast d'aquests queda delimitat a fins que es compleixin aquests objectius.

Per altra banda queda fora de l'abast del projecte el fet de desenvolupar més funcionalitats que no siguin les mínimes descrites. A causa del fet que l'Apòstrof és una empresa on es mira molt el detall dels dissenys gràfics, també s'ha acordat de forma explícita que queda fora de l'abast d'aquest projecte el fet d'aconseguir un nivell d'acabat del disseny alt. Així i tot sí que s'hi haurà d'aconseguir que el sistema sigui fàcilment usable.

També, el fet que en aquest projecte es desenvolupa una primera versió d'un software amb el qual es continuarà treballant i s'hauran de crear més versions, queda implícit que aquest software haurà de ser el màxim de mantenible i extensible.

3. Estat de l'art

En el capítol de Formulació del Problema ja s'ha analitzat quines són les funcionalitats que té Intranet actual, però també és important fer una anàlisi d'altres sistemes ja existents per veure si hi ha algun que ja compleixi amb aquestes necessitats o no.

Ja que l'arquitectura que es vol fer servir és força nova, és convenient analitzar de quina documentació podrem disposar per dur a terme el projecte. Com a documentació es vol dir tota la bibliografia escrita en l'àmbit teòric, i també buscar si ja hi ha exemples de projectes que posin en pràctica aquesta arquitectura en un context similar (aplicació web en PHP).

3.1 Solucions ja existents

És evident que en el mercat ja hi ha moltes eines de gestió de feines, ja que és un dels primers *problemes* que es van començar a informatitzar en les empreses. Però també és cert, que fins fa pocs anys aquests sistemes només han estat a l'abast de les grans empreses, i no ha sigut fins que ca començar el *boom* de les aplicacions i serveis web que aquests sistemes no han arribat a les mitjanes i petites empreses. A més, en general totes les solucions ja creades se centren únicament en els aspectes econòmics i de productivitat deixant de costat altres com els aspectes humans i socials.

Tot i que sembla obvi que és difícil trobar una solució ja existent que abordi tots els aspectes que es vol, continua sent important analitzar aquestes solucions. A continuació es mostren alguns d'aquests sistemes:

- **Billage** *(https://www.billage.es/es/)* Billage és un servei multiplataforma (web, Android i iOS) per a empreses i autònoms que facilita la gestió en els àmbits comercial, gestió de projectes i facturació. Com a punts forts té diverses coses; per començar ofereix moltes integracions amb altres sistemes àmpliament utilitzats com serveis de Google (Gmail, Calendar i Drive), Mailchimp i Typeform, cosa que pot facilitar moltes coses a empreses amb pocs recursos. I en l'apartat de gestió de projectes cal destacar les funcionalitats de gestió de tasques amb un

taule Kanban i una interfície molt ben aconseguida per introduir hores que facilita molt aquesta tasca.

- **Anfix** *(https://anfix.com)* Un altre servei multiplataforma (web, Android i iOS) centrat principalment amb la gestió econòmica (despeses i factures). També incorpora una petita gestió de projectes per poder organitzar aquestes despeses i factures en projectes. Una funcionalitat que cal destacar és el fet de poder personalitzar la factura mitjançant plantilles editables.
- **Teamleader** *https://www.teamleader.es* Teamleader és una plataforma web que ofereix funcionalitats per a la gestió de diferents àrees d'una empresa, entre elles, les de gestió de projectes i facturació. Com a funcionalitat destacable té el fet de poder donar accés als projectes als clients, i així facilitar la comunicació.

Com s'ha dit, en el mercat hi ha moltes altres solucions ja fetes, les anteriors mencionades s'han destacat, ja que ofereixen alguna funcionalitat destacada que podria agafar-se d'exemple per un futur.

El problema principal que fa molt difícil trobar un sistema ja fet al qual es pogués fer servir és el fet que cap permet la gestió i anàlisi de feines internes que no aporten un benefici directe (que no es facturen), i aquest és un punt molt important per a l'Apòstrof.

3.2 Clean Architecture

Com indica el títol d'aquest projecte, es vol crear aquesta intranet utilitzant una implementació d'una arquitectura neta[1]amb PHP.

El concepte de *Clean Architecure*[1] va ser introduït per en Robert C. Martin[2] l'Agost del 2012 en una entrada al seu blog. Anteriorment ja havia escrit una altra entrada on començava a dibuixar un dels conceptes diferenciadors d'aquesta arquitectura; el concepte de *Screaming Architecture*[3].

[1]Clean Architecture: `https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html`

[2]Robert C. Martin: `https://en.wikipedia.org/wiki/Robert_Cecil_Martin`

[3]Screaming Architecture: `https://8thlight.com/blog/uncle-bob/2011/09/30/Screaming-Architecture.html`

A part de les dues entrades anteriors mencionades, no hi ha hagut més bibliografia escrita sobre aquest tema en l'àmbit teòric. Sí que es poden trobar vídeos de conferències on en Robert C. Martin parla i exemplifica una mica més aquesta arquitectura. Però no ha sigut fins al 2017 que no s'ha publicat cap llibre. El Setembre de 2017 es va publicar el llibre *Clean Architecture: A Craftsman's Guide to Software Structure and Design*[4]. Aquest llibre no s'ha pogut utilitzar com a referència per aquest projecte, ja que no ha sigut possible aconseguir un exemplar.

Si ens fixem en com implementar aquesta arquitectura en un servei web amb PHP en concret, sí que podem trobar algunes referències. La més significativa és el llibre *The Clean Architecture in PHP*[5]. També hi ha algunes entrades de blogs que expliquen una mica per sobre com fer una implementació amb PHP de l'arquitectura, però no entren gaire a fons.

Tot i que sembla que fins fa poc no hi ha hagut gaire bibliografia escrita sobre aquest tema, i que per tant podríem dir que no genera gaire interès, això no és cert. Per començar aquesta arquitectura està fortament lligada als conceptes de *Clean Code*[6] (que ja fa més temps que ha pres rellevància dins el sector i te força bibliografia escrita) i *principis SOLID*[7]. També és fàcil observar com cada cop més es poden trobar articles, conferències i xerrades que aborden aquests temes. I per últim, l'interès sobre aquesta arquitectura ha quedat demostrat pel fet que el nou llibre publicat aquest 2017 per Robert C. Martin ja havia aconseguit el distintiu de "best seller" a Amazon abans de sortir a la venda, només amb les prevendes.

I perquè pot arribar a generar aquest interès aquesta "nova" arquitectura? Segurament pels beneficis que busca aportar i d'on prové, ja que no és una "nova" arquitectura en si mateixa, sinó que és un intent d'unificar i alinear altres tipus d'arquitectures quedant-se amb el millor de cada una.

Aquestes arquitectures són; Hexagonal architecture[8], Onion architectu-

[4]Clean Architecture: A Craftsman's Guide to Software Structure and Design: `https://www.amazon.es/dp/0134494164/`

[5]The Clean Architecture in PHP: `https://leanpub.com/cleanphp`

[6]Clean Code: `https://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882`

[7]SOLID: `https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)`

[8]Hexagonal architecture: `http://alistair.cockburn.us/Hexagonal+architecture`

re[9], Screaming architecture[10], Lean architecture[11] i el concepte de Use Case Driven Approach[12].

Segons l'autor, com a resultat s'aconsegueix una arquitectura que aporta els següents beneficis:

1. **Independència de Frameworks[13]:** Molts sistemes estan desenvolupats utilitzant frameworks, ja que aquests faciliten la feina als desenvolupadors. El problema és quan el nivell d'acoblament amb aquests frameworks és elevat, ja que si per alguna raó s'ha de canviar o actualitzar el framework, no serà una tasca fàcil. Per això és important poder mantenir el nivell d'acoblament amb els frameworks o eines al mínim possible.

2. **Independència de la interfície gràfica (UI):** És habitual que la UI canviï constantment, sense que realment es modifiqui la resta del sistema. Per tant és important que la base del sistema (lògiques de negoci[14]) no estigui acoblat a la UI.

3. **Independència de la Base de Dades:** De la mateixa manera que amb la UI, és important mantenir un nivell d'acoblament baix amb el sistema que es faci servir per guardar les dades.

4. **Testable:** És important poder testejar al màxim els sistemes per verificar que funcionen correctament, i sobretot les parts centrals d'aquests. Per tant, gràcies al fet que podem desacoblar-nos fàcilment de tots els detalls externs (com ara la UI i les Bases de Dades) és molt més fàcil testejar aquestes parts importants.

[9]Onion architecture: `http://jeffreypalermo.com/blog/the-onion-architecture-part-1/`

[10]Screaming architecture: `https://8thlight.com/blog/uncle-bob/2011/09/30/Screaming-Architecture.html`

[11]Lean architecture: `http://www.leansoftwarearchitecture.com/`

[12]Use Case Driven Approach: `http://www.interface.ru/rational/rup51/manuals/intro/im_feat2.htm`

[13]Framework: `https://en.wikipedia.org/wiki/Software_framework`

[14]Lògiques de negoci: `https://en.wikipedia.org/wiki/Business_rule`

4. Metodologia i rigor

4.1 Mètodes de treball

Ja que aquest projecte serà dut a terme principalment només per a mi no s'usarà cap metodologia popular en si. Ara bé, sí que ens fixarem en metodologies que ens puguin ser útils i s'adaptaran a les necessitats de l'equip i projecte.

Ja que aquest és un projecte on els requisits no estan perfectament definits, es vol poder fer canvis durant el desenvolupament i es vol poder provar els resultats tan aviat com es pugui per poder decidir aquests canvis, és un entorn perfecte per usar metodologies *àgils*.

Les dues metodologies més conegudes són el *Kanban*[1] i el *Scrum*[2]. Per un costat amb *Kanban* s'intenta tenir un control més precís del treball al llarg del temps, mentre que per l'altre costat, amb *Scrum* sobretot s'intenta poder donar una flexibilitat per poder treballar amb requisits canviants. Per tant s'ha decidit basar-se més amb la metodologia del *Scrum*.

4.1.1 Mètode final

El procés que s'ha decidit que se seguirà serà el següent:

- Es crearà una *Comissió Intranet*, composta per dos socis de l'Apòstrof, que realitzarà el rol de *ProductOwner*[3] i seran els encarregats de decidir els requisits del sistema i les prioritats del projecte.
- A l'inici del projecte es crearà un *BackLog*[4] inicial amb totes les *Històries*

[1] Kanban: `https://es.wikipedia.org/wiki/Kanban_(desarrollo)`

[2] Scrum: `https://proyectosagiles.org/que-es-scrum/`

[3] Product Ower: `https://www.mountaingoatsoftware.com/agile/scrum/roles/product-owner`

[4] BackLog: `https://www.mountaingoatsoftware.com/agile/scrum/scrum-tools/product-backlog`

d'Usuari[5] que es puguin pensar. Aquestes històries d'usuaris hauran de ser el màxim de petites possibles i independents unes de les altres.

- Es treballarà en iteracions de feina incrementals (definides més endavant). Així, a poc a poc s'aniran incorporant funcionalitats al sistema, i al dinal de cada iteració hi haurà un sistema funcional.
- Aquestes iteracions seran de dues setmanes. A l'inici de cada iteració es farà una reunió amb la *Comissió Intranet* per decidir quines històries d'usuari implementaran. I al final de les iteracions es comprovarà quines s'han pogut implementar i quines no.
- Puntualment es faran "jornades de testing" amb la resta de persones de l'Apòstrof perquè provin tot el que hi ha desenvolupat fins al moment i puguin aportar un feedback.

4.2 Eines de segiment

S'utilitzarà Trello[6] com a software per ajudar a la planificació d'aquestes iteracions i fer-ne el seguiment. Per tal de fer el seguiment i control de versions del projecte s'utilitzarà un repositori de git[7] allotjat a GitHub[8]. Així també es podran monitoritzar els canvis que es produeixen durant el desenvolupament.

4.3 Mètodes de validació

4.3.1 Mètodes de validació manuals

Com s'ha explicat en la metodologia de treball, es realitzaran reunions al finalitzar cada iteració amb la *Comissió Intranet* (*Product Owner*) perquè

[5]Histories d'Usuari: `https://www.mountaingoatsoftware.com/agile/user-stories`

[6]Trello: `https://trello.com/`

[7]Git: `https://git-scm.com/`

[8] GitHub: `https://github.com/`

verifiquin que les funcionalitats implementades funcionen bé i fan tot el que han de fer. A part, també hi haurà una versió de prova de la Intranet disponible per als treballadors perquè puguin provar-la quan vulguin. I es marcaran dates de testing cada certes iteracions per tal que els treballadors provin tot el sistema implementat fins al moment.

També es realitzaran revisions del codi (*Code Review*) entre els dos desenvolupadors del projecte, per així assegurar la qualitat d'aquest. Per a fer això, per a cada funcionalitat que s'implementi es crearà una branca al repositori git i un cop finalitzada es crearà una petició d'integració (*Pull Request*) cap a la branca principal. Així serà fàcil tots els canvis que involucren una funcionalitat.

4.3.2 Mètodes de validació automàtics

La base principal seran els Tests Unitaris, que es realitzaran en totes les classes/parts possibles del codi. Per fer això es farà ús de la pràctica del TDD. També es crearan Tests d'Integració per verificar que tot el sistema funciona correctament junt.

5. Anàlisi de Requisits

5.1 Requisits funcionals

Com s'ha dit en l'apartat d'Objectius del projecte, s'han d'implementar un mínim de funcionalitats per tal de poder deixar de fer servir la Intranet actual i assolir l'objectiu principal d'aquest projecte.

En aquest apartat descriurem tots els requisits funcionals necessaris per aconseguir aquestes funcionalitats.

Podem dir que les funcionalitats principals són les directament relacionades amb la gestió de les feines, sobretot per a poder associar en aquestes feines uns còmputs d'hores dedicades, despeses i facturació. Aquest últim apartat de factures (o facturació) també és força ampli. Per tant podem agrupar aquestes funcionalitats en 4 grans grups; un primer referent a totes les funcionalitats bàsiques necessàries per interactuar amb el sistema, un segon grup amb funcionalitats referents a gestiona les dades del sistema bàsiques, un tercer grup que compren les funcionalitats per a la gestió de les feines en si mateixes i un quart grup on entren totes les funcionalitats de facturació.

5.1.1 Sistema

Aquest grup compren bàsicament dues funcionalitats molt bàsiques, les de **iniciar i finalitzar sessió** dels usuaris.

5.1.2 Administració del sistema:

Ja que s'haurà d'iniciar sessió per a utilitzar el sistema, és evident que la primera funcionalitat d'aquest grup ha de ser la de gestionar els usuaris del sistema; en concret el sistema hi haurà de permetra **crear, editar, esborrar i llistar els usuaris del sistema**.

Com hem dit les feines tindran un còmput d'hores dedicades, per tant també s'haurà de poder **crear, editar, esborrar i llistar treballadors**, als quals s'associaran les hores computades a les feines. Els treballadors només es podran esborrar en el cas que no tinguin cap hora associada. En cas contrari no es podran esborrar, però sí desactivar. També es podrà **associar un treballador a un usuari**, així aquest usuari podrà introduir hores del treballador.

Les feines s'organitzaran en partides, que a la vegada estaran relacionades amb una àrea de l'empresa. Per tant el sistema hi haurà de permetre la gestió d'aquestes àrees, en concret **crear, editar i llistar les àrees** l'empresa.

A les feines, a través de les partides, també s'associaran un seguit de despeses. Aquestes despeses s'hauran de classificar, per tant uns altres requisits seran els de poder **crear, editar, esborrar i llistar els diferents tipus de despeses**. Aquestes despeses també es volen associar a col·laboradors o proveïdors per, en un futur, poder analitzar aquestes dades. Per tant és obvi que també són necessaris els requisits funcionals de **crear, editar, esborrar i llistar els proveïdors** de l'empresa.

Per acabar amb les funcionalitats d'aquest grup, cada feina es realitzarà per un client, per tant també es vol poder **crear, editar, esborrar i llistar els clients** de l'empresa, per poder-los associar a les feines i factures.

5.1.3 Gestions de Feines:

Totes les funcionalitats descrites anteriorment són les necessàries per a poder, finalment, dur a terme les funcionalitats descrites a continuació en aquest grup.

Evidentment, les funcionalitats més importants seran les de **veure, crear, editar, esborrar i llistar les feines**. Com s'ha dit anteriorment, les feines s'organitzaran en partides, per tant també s'haurà de poder **crear, editar, esborrar i llistar les partides d'una feina**, i per a cada partida s'haurà de poder **crear, editar, esborrar i llistar despeses**.

Per acabar, per a poder tenir un còmput d'hores dedicades a les feines, els usuaris podran **introduir hores associades a una partida d'una**

feina i un treballador, també s'han de poder **llistar i esborrar aquestes hores**.

5.1.4 Facturació:

Aquest últim grup conté totes les funcionalitats referents a la facturació de les feines. Bàsicament per a cada feina es vol poder portar un control de les factures que s'envien als clients.

A part de les dades d'una factura (dades del client, imports, lineas, etc), les factures podran passar per un total de 4 estats:

1. El primer estat serà el de *demanada*. Això significarà que la factura ha estat donada d'alta al sistema amb les dades bàsiques, però que falta revisar-la i enviar-la al client.
2. El següent estat serà el de *acceptada*. Una factura passarà a aquest estat quant a estigui revisada i enviada al client.
3. A continuació una factura passarà a l'estat de *pagada* en quant el client la pagui.
4. L'últim estat possible serà el de *rectificada*. Això passarà quan es vulgui editar una factura, però ja no és possible fer-ho donat que ja s'ha presentat a hisenda. En aquest cas la factura es marcarà com a rectificada i se'n crearà una altra d'idèntica que "rectificarà" l'anterior.

Les factures hauran d'estar numerades seguint una sèrie incremental per a cada any, i les factures rectificades hauran de seguir una sèrie paral·lela per cada any.

Com es pot veure, la "vida" d'una factura és una mica complexa, per tant en aquest cas seran necessaris més funcionalitats per a facilitar la gestió d'aquestes. En concret es podrà **demanar una factura** (per crear-la), **editar i imprimir una factura** (per revisar-la i enviar-la) i **rectificar una factura**. A part, és obvi que també s'haurà de poder **llistar i esborrar les factures**.

5.2 Requisits no funcionals

A part dels requisits funcionals descrits anteriorment, el sistema també haurà de complir amb una sèrie de requisits no funcionals per garantir el correcte funcionament d'aquest.

1. **Disponibilitat:** El servidor ha d'estar sempre disponible perquè els usuaris puguin utilitzar la Intranet i dur el control de les feines de la cooperativa.
2. **Temps de resposta:** El temps de resposta del servidor quan l'aplicació fa una petició hauria de ser inferior als 3 segons.
3. **Usabilitat:** La interfície de la intranet hauria de ser senzilla i fàcil d'usar per a qualsevol nivell d'usuari. També hauria de tenir una corba d'aprenentatge ràpida per tal que els usuaris la sàpiguen fer servir ràpidament.
4. **Robustesa:** Les dades introduïdes pels usuaris no poden corrompre el sistema i s'haurà d'indicar a aquests quan introdueixin dades incorrectes de la millor forma i rapidesa possible.
5. **Testabilitat:** S'hauran de poder crear tests del sistema fàcilment per tal que es pugui mantenir i aplicar modificacions.

6. Especificació

6.1 Actors

Com es pot veure en l'apartat de Formulació del Problema, la Intranet actual té diversos tipus d'usuaris (usuaris no identificats, administradors, treballadors i socis), però per aquesta primera versió només hi haurà dos tipus bàsics. Els diferents actors que interactuaran amb el sistema seran els següents:

Usuari no identificat: tota persona que accedeixi al sistema, però encara no s'ha identificat, i que per tant l'únic que podran fer és identificar-se.

Usuari identificat: els usuaris identificats seran els que faran servir la Intranet cada dia. Seran els encarregats d'introduir totes les dades necessàries al sistema per tal de poder fer servir la Intranet i dur un control de les feines de la cooperativa. En general els treballadors responsables de cada feina seran principalment els que gestionaran les dades de les feines.

6.2 Casos d'ús

En aquest apartat es mostren tots els casos d'ús necessari per a poder complir amb tots els requisists funcionals que ha de tenir el sistema. En concret es mostren els diagrames de tots els casos d'ús i una explicació en detall de cada un d'ells.

6.2.1 Sistema:

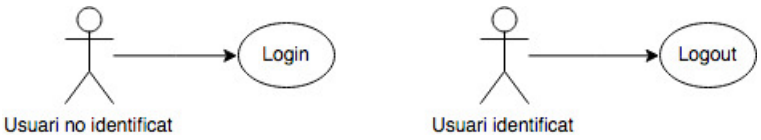


Figura 6.1: Casos d'ús bàsics de sistema

Cas d'ús 1 - Login

L'usuari introdueix el email i la contrasenya per tal de poder accedir i usar el sistema com a usuari.

Actor principal: Usuari no identificat
Disparador: Un usuari no identificat vol identificar-se per entrar al sistema.
Precondicions:

1. L'usuari està donat d'alta al sistema.

Escenari principal:

1. L'usuari obre la web per primera vegada.
2. El sistema mostra el formulari per introduir les credencials (email i contrasenya).
3. L'usuari omple el formulari i envia les dades.
4. El sistema valida les dades i mostra la pàgina principal.

Extensions:

4.1. El sistema detecta credencials incorrectes
   (a) El sistema mostra un altre cop el formulari per introduir les

credencials, però amb un error indicant quin és el problema.
(b) Tornar al pas 3.

Cas d'ús 2 - Logout

L'usuari clica el botó de desconnexió per deixar d'usar el sistema.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol finalitzar la sessió en un navegador.
Precondicions:
1. L'usuari està identificat.
2. L'usuari té la web oberta en alguna pàgina.

Escenari principal:
1. L'usuari clica al botó de *logout*.
2. El sistema finalitza la sessió de l'usuari i mostra el formulari per introduir les credencials.

6.2.2 Administració del sistema:

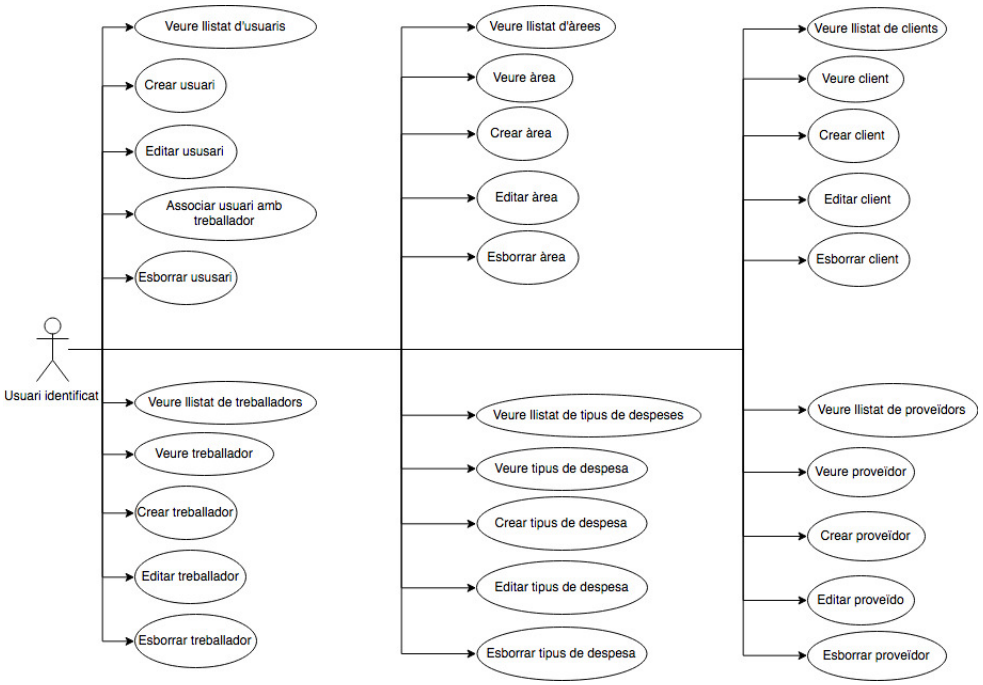


Figura 6.2: Casos d'ús d'administració del sistema

Com es pot veure en l'anterior gràfic, en aquest apartat hi ha molts casos d'ús. Fàcilment es pot veure que podem agrupar els casos d'ús amb 6 subgrups; els relacionats amb usuaris, treballadors, àrees, tipus de despeses, clients i proveïdors.
Per tant, tot seguit, es definiran seguint aquest ordre.

6.2.2.1 Usuaris

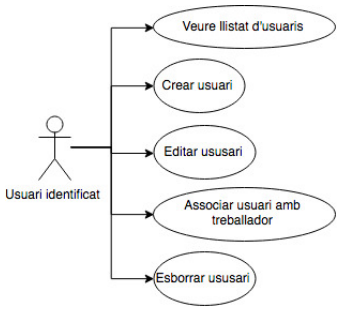


Figura 6.3: Casos d'ús d'administració d'Usuaris

Cas d'ús 3 - Llistat d'usuaris

Veure un llistat de tots els usuaris de la cooperativa per tal de poder accedir a tota la informació necessària.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure un llistat de tots els usuaris registrats al sistema.
Precondicions:

1. L'usuari està identificat.

Escenari principal:

1. L'usuari obre la web a la pàgina de llistat d'usuaris.
2. El sistema mostra el llistat de tots els usuaris registrats al sistema.

Cas d'ús 4 - Crear usuari

Donar d'alta un usuari per tal que la informació d'aquest estigui en el sistema, es pugui consultar i aquest pugui iniciar sessió

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol donar d'alta un altre usuari

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat d'usuaris.

Escenari principal:

1. L'usuari clica al link de "crear usuaris".
2. El sistema mostra el formulari d'alta d'usuari (email i contrasenya).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i dona d'alta el nou usuari.
5. El servidor mostra un altre cop el llistat d'usuaris actualitzat.

Extensions:

4.1. Ja hi ha un usuari registrat amb el mateix email
   (a) El sistema torna a mostrar el formulari d'alta d'usuari mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

4.2. La contrasenya no és valida
   (a) El sistema torna a mostrar el formulari d'alta d'usuari mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 5 - Editar usuari

Editar la informació d'un usuari per tal d'ampliar-la o corregir-la.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol editar la informació d'un altre usuari

Precondicions:

1.
2. L'usuari està identificat.
3. L'usuari està veient la pàgina de llistat d'usuaris.

Escenari principal:

1. L'usuari clica al link de "editar usuari" d'una fila del llistat.
2. El sistema mostra el formulari d'usuari (email i contrasenya).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i actualitza la informació de l'usuari.

5. El servidor mostra un altre cop el llistat d'usuaris actualitzat.

Extensions:

4.1. Ja hi ha un usuari registrat amb el mateix email
   (a) El sistema torna a mostrar el formulari d'alta d'usuari mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

4.2. La contrasenya no és vàlida
   (a) El sistema torna a mostrar el formulari d'alta d'usuari mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 6 - Associar usuari amb treballador

Associar un usuari amb un treballador per tal que quan una persona identificada amb aquest usuari entri hores es faci en nom del treballador associat.

Actor principal: Usuari identificat

Disparador: Un usuari vol associar un treballador amb un usuari del sistema

Precondicions:

1.
2. L'usuari està identificat.
3. L'usuari està veient la pàgina de llistat d'usuaris.

Escenari principal:

1. L'usuari clica al link de "associar treballador" d'una fila del llistat.
2. El servidor mostra un llistat de treballadors actius i no associats a cap usuari encara.
3. L'usuari selecciona un treballador i envia les dades.
4. El servidor actualitza la informació de l'usuari.
5. El servidor mostra un altre cop el llistat d'usuaris actualitzat.

Cas d'ús 7 - Esborrar usuari

Esborrar un usuari per tal d'eliminar la informació sobre aquest del

sistema.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol esborrar un altre usuari
Precondicions:

1.
2. L'usuari està identificat.
3. L'usuari està veient la pàgina de llistat d'usuaris.

Escenari principal:

1. L'usuari clica al link de "eliminar usuari" d'una fila del llistat.
2. El servidor elimina l'usuari.
3. El servidor mostra un altre cop el llistat d'usuaris actualitzat.

Extensions:

3.1 L'usuari esborrar el seu propi usuari
   (a) El sistema tanca la sessió de l'usuari i mostra el formulari de Login.

6.2.2.2 Treballadors

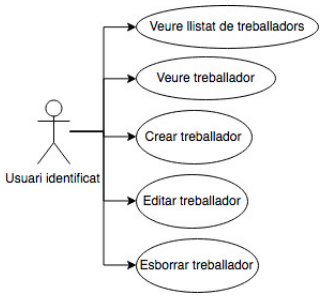


Figura 6.4: Casos d'ús d'administració de Treballadors

Cas d'ús 8 - Veure llistat de treballadors

Veure un llistat de tots els treballadors de la cooperativa per tal de poder accedir a tota la informació necessària.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure un llistat de tots els treballadors registrats al sistema.
Precondicions:

1. L'usuari està identificat.

Escenari principal:

1. L'usuari obre la web a la pàgina de llistat de treballadors.
2. El sistema mostra el llistat de tots els treballadors registrats.

Cas d'ús 9 - Crear treballador

Donar d'alta un treballador per tal que la informació d'aquest estigui en el sistema i es pugui consultar i introduir hores en el seu nom.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol donar d'alta un nou treballador.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de treballadors.

Escenari principal:

1. L'usuari clica al link de "crear treballador".
2. El sistema mostra el formulari d'alta de treballador (nom i cognoms).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i dona d'alta el nou treballador amb estat *actiu*.
5. El servidor mostra un altre cop el llistat de treballadors actualitzat.

Extensions:

4.1. Dades buides
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 10 - Veure treballador

Veure la fitxa d'un treballador per tal de poder consultar tota la informació d'aquest.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol consultar la fitxa d'un treballador.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de treballadors.

Escenari principal:

1. L'usuari clica al link de "veure treballador" d'una fila del llistat.
2. El sistema mostra la fitxa de treballador amb tota la informació.

Cas d'ús 11 - Editar treballador

Editar la informació d'un treballador per tal d'ampliar-la o corregir-la.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol editar la informació d'un treballador.
Precondicions:
1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de treballadors.

Escenari principal:
1. L'usuari clica al link de "editar treballador" d'una fila del llistat.
2. El sistema mostra el formulari de treballador (nom, cognoms i actiu/inactiu).
3. L'usuari edita les dades del formulari i les envia.
4. El servidor processa les dades i actualitza la informació del treballador.
5. El servidor mostra un altre cop el llistat de treballadors actualitzat.

Extensions:
4.1. Dades buides
   (a) El sistema torna a mostrar el formulari de treballador mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 12 - Esborrar treballador

Esborrar un treballador per tal d'eliminar la informació sobre aquest del sistema.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol esborrar un treballador del sistema.
Precondicions:
1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de treballadors.

Escenari principal:
1. L'usuari clica al link de "esborrar treballador" d'una fila del llistat.
2. El servidor elimina el treballador.

3. El servidor mostra un altre cop el llistat de treballadors actualitzat.

Extensions:

2.1 El treballador ja té hores assignades
   (a) El servidor no elimina el treballador.
   (b) El servidor torna a mostrar el llistat de treballadors indicant que no es pot esborrar el treballador.

6.2.2.3 Àrees

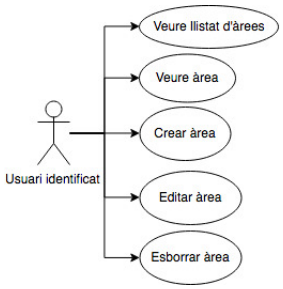


Figura 6.5: Casos d'ús d'administració d'Àrees

Cas d'ús 13 - Veure llistat d'àrees

Veure un llistat de les àrees de la cooperativa per tal de poder accedir a tota la informació necessària.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure un llistat de totes les àrees.
Precondicions:

1. L'usuari està identificat.

Escenari principal:

1. L'usuari obre la web a la pàgina de llistat d'àrees.
2. El sistema mostra el llistat amb totes les àrees.

Cas d'ús 14 - Crear àrea

Donar d'alta una àrea per tal que la informació d'aquesta estigui en el sistema i es puguin associar partides de feines a aquesta àrea.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol donar d'alta una nova àrea.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat d'àrees.

Escenari principal:

1. L'usuari clica al link de "crear àrea".
2. El sistema mostra el formulari d'alta d'àrea (nom i descripció).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i dona d'alta la nova àrea amb estat activa.
5. El servidor mostra un altre cop el llistat d'àrees actualitzat.

Extensions:

4.1. Nom buit
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

4.1. Nom duplicat
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 15 - Veure àrea

Veure la fitxa d'una àrea per tal de poder consultar a tota la informació d'aquesta.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol consultar la fitxa d'una àrea.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat d'àrees.

Escenari principal:

1. L'usuari clica al link de "veure àrea" d'una fila del llistat.
2. El sistema mostra la fitxa d'àrea amb tota la informació.

Cas d'ús 16 - Editar àrea

Editar la informació d'una àrea per tal d'ampliar-la o corregir-la.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol editar la informació d'una àrea.
Precondicions:
1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat d'àrees.

Escenari principal:
1. L'usuari clica al link de "editar àrea" d'una fila del llistat.
2. El sistema mostra el formulari d'àrea (nom, descripció i actiu/inactiu).
3. L'usuari edita les dades del formulari i les envia.
4. El servidor processa les dades i actualitza la informació de l'àrea.
5. El servidor mostra un altre cop el llistat d'àrees actualitzat.

Extensions:
4.1. Nom buit
   (a) El sistema torna a mostrar el formulari d'àrea mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

4.1. Nom duplicat
   (a) El sistema torna a mostrar el formulari d'àrea mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 17 - Esborrar àrea

Esborrar una àrea per tal d'eliminar la informació sobre aquesta del sistema.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol esborrar una àrea del sistema.
Precondicions:
1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat d'àrees.

Escenari principal:
1. L'usuari clica al link de "esborrar àrea" d'una fila del llistat.
2. El servidor elimina l'àrea.
3. El servidor mostra un altre cop el llistat d'àrees actualitzat.

Extensions:

2.1 L'àrea ja té partides associades
   (a) El servidor no elimina l'àrea.
   (b) El servidor torna a mostrar el llistat d'àrees indicant que no es pot esborrar l'àrea.

6.2.2.4 Tipus de despeses

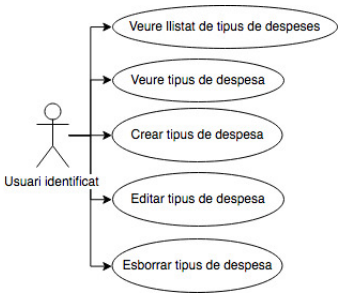


Figura 6.6: Casos d'ús d'administració de Tipus de despeses

Cas d'ús 18 - Veure llistat de tipus de despeses

Veure un llistat de tots els tipus de despeses que es fan servir a la cooperativa per tal de poder accedir a tota la informació necessària.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure un llistat de tots els tipus de despeses.
Precondicions:

1. L'usuari està identificat.

Escenari principal:

1. L'usuari obre la web a la pàgina de llistat de tipus de despeses.
2. El sistema mostra el llistat amb tots els tipus de despeses.

Cas d'ús 19 - Crear tipus de despesa

Donar d'alta un tipus de despesa per tal que la informació d'aquesta estigui en el sistema i es puguin associar despeses d'una feina a aquest tipus.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol donar d'alta un nou tipus de despesa.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de tipus de despesa.

Escenari principal:

1. L'usuari clica al link de "crear tipus de despesa".
2. El sistema mostra el formulari d'alta de tipus de despesa (nom).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i dona d'alta el nou tipus de despesa amb estat actiu.
5. El servidor mostra un altre cop el llistat de tipus de despesa actualitzat.

Extensions:

4.1. Nom buit
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

4.1. Nom duplicat
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 20 - Veure tipus de despesa

Veure la fitxa d'un tipus de despesa per tal de poder consultar a tota la informació d'aquest.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol consultar la fitxa d'un tipus de despesa.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de tipus de despesa.

Escenari principal:

1. L'usuari clica al link de "veure tipus de despesa" d'una fila del llistat.

2. El sistema mostra la fitxa de tipus de despesa amb tota la informació.

Cas d'ús 21 - Editar tipus de despesa

Editar la informació d'un tipus de despesa per tal d'ampliar-la o corregir-la.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol editar la informació d'un tipus de despesa.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de tipus de despesa.

Escenari principal:

1. L'usuari clica al link de "editar tipus de despesa" d'una fila del llistat.
2. El sistema mostra el formulari de tipus de despesa (nom i actiu/inactiu).
3. L'usuari edita les dades del formulari i les envia.
4. El servidor processa les dades i actualitza la informació del tipus de despesa.
5. El servidor mostra un altre cop el llistat de tipus de despeses actualitzat.

Extensions:

4.1. Nom buit
   (a) El sistema torna a mostrar el formulari de tipus de despesa mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

4.1. Nom duplicat
   (a) El sistema torna a mostrar el formulari de tipus de despesa mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 22 - Esborrar tipus de despesa

Esborrar un tipus de despesa per tal d'eliminar la informació sobre aquest del sistema.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol esborrar un tipus de despesa del sistema.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de tipus de despesa.

Escenari principal:

1. L'usuari clica al link de "esborrar tipus de despesa" d'una fila del llistat.
2. El servidor elimina el tipus de despesa.
3. El servidor mostra un altre cop el llistat de tipus de despeses actualitzat.

Extensions:

2.1 El tipus de despesa ja té despeses associades
   (a) El servidor no elimina el tipus de despesa.
   (b) El servidor torna a mostrar el llistat de tipus de despeses indicant que no es pot esborrar el tipus de despesa desitjat.

6.2.2.5 Clients

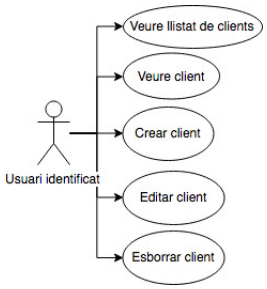


Figura 6.7: Casos d'ús d'administració de Clients

Cas d'ús 23 - Veure llistat de clients

Veure un llistat de tots els clients que es fan servir a la cooperativa per tal de poder accedir a tota la informació necessària.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure un llistat de tots els clients.
Precondicions:

1. L'usuari està identificat.

Escenari principal:

1. L'usuari obre la web a la pàgina de llistat de clients.
2. El sistema mostra el llistat amb tots els clients.

Cas d'ús 24 - Crear client

Donar d'alta un client per tal que la informació d'aquest estigui en el sistema i es puguin associar feines i factures a aquest client.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol donar d'alta un nou client.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de clients.

Escenari principal:

1. L'usuari clica al link de "crear client".
2. El sistema mostra el formulari d'alta de client (nom, cif, adreça, codi postal i ciutat).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i dona d'alta el nou client amb estat actiu.
5. El servidor mostra un altre cop el llistat de clients actualitzat.

Extensions:

4.1. Nom buit
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 25 - Veure client

Veure la fitxa d'un client per tal de poder consultar a tota la informació d'aquest.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol consultar la fitxa d'un client.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de clients.

Escenari principal:

1. L'usuari clica al link de "veure client" d'una fila del llistat.
2. El sistema mostra la fitxa de client amb tota la informació.

Cas d'ús 26 - Editar client

Editar la informació d'un client per tal d'ampliar-la o corregir-la.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol editar la informació d'un client.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de clients.

Escenari principal:

1. L'usuari clica al link de "editar client" d'una fila del llistat.
2. El sistema mostra el formulari de client (nom, cif, adreça, codi postal, ciutat i actiu/inactiu).
3. L'usuari edita les dades del formulari i les envia.
4. El servidor processa les dades i actualitza la informació del client.
5. El servidor mostra un altre cop el llistat de clients actualitzat.

Extensions:

4.1. Nom buit
   (a) El sistema torna a mostrar el formulari mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 27 - Esborrar client

Esborrar un client per tal d'eliminar la informació sobre aquest del sistema.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol esborrar un client del sistema.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de clients.

Escenari principal:

1. L'usuari clica al link de "esborrar client" d'una fila del llistat.
2. El servidor elimina el client.
3. El servidor mostra un altre cop el llistat de clients actualitzat.

Extensions:

2.1 El client ja té feines o factures associades
   (a) El servidor no elimina el client.
   (b) El servidor torna a mostrar el llistat de clients indicant que no es pot esborrar el client desitjat.

6.2.2.6 Proveïdors

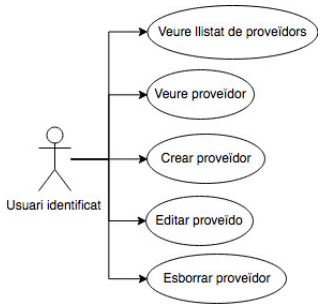


Figura 6.8: Casos d'ús d'administració de Proveïdors

Cas d'ús 28 - Veure llistat de proveïdors

Veure un llistat de tots els proveïdors que es fan servir a la cooperativa per tal de poder accedir a tota la informació necessària.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure un llistat de tots els proveïdors.
Precondicions:

1. L'usuari està identificat.

Escenari principal:

1. L'usuari obre la web a la pàgina de llistat de proveïdors.
2. El sistema mostra el llistat amb tots els proveïdors.

Cas d'ús 29 - Crear proveïdor

Donar d'alta un proveïdor per tal que la informació d'aquest estigui en el sistema i es puguin associar despeses a aquest proveïdor.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol donar d'alta un nou proveïdor.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de proveïdors.

Escenari principal:

1. L'usuari clica al link de "crear proveïdor".
2. El sistema mostra el formulari d'alta de proveïdor (nom).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i dona d'alta el nou proveïdor amb estat actiu.
5. El servidor mostra un altre cop el llistat de proveïdors actualitzat.

Extensions:

4.1. Nom buit
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

4.1. Nom duplicat
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 30 - Veure proveïdor

Veure la fitxa d'un proveïdor per tal de poder consultar a tota la informació d'aquest.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol consultar la fitxa d'un proveïdor.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de proveïdors.

Escenari principal:

1. L'usuari clica al link de "veure proveïdor" d'una fila del llistat.
2. El sistema mostra la fitxa de proveïdor amb tota la informació.

Cas d'ús 31 - Editar proveïdor

Editar la informació d'un proveïdor per tal d'ampliar-la o corregir-la.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol editar la informació d'un proveïdor.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de proveïdors.

Escenari principal:

1. L'usuari clica al link de "editar proveïdor" d'una fila del llistat.
2. El sistema mostra el formulari de proveïdor (nom i actiu/inactiu).
3. L'usuari edita les dades del formulari i les envia.
4. El servidor processa les dades i actualitza la informació del proveïdor.
5. El servidor mostra un altre cop el llistat de proveïdors actualitzat.

Extensions:

4.1. Nom buit
   (a) El sistema torna a mostrar el formulari mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

4.1. Nom duplicat
   (a) El sistema torna a mostrar el formulari mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 32 - Esborrar proveïdor

Esborrar un proveïdor per tal d'eliminar la informació sobre aquest del sistema.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol esborrar un proveïdor del sistema.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de proveïdors.

Escenari principal:

1. L'usuari clica al link de "esborrar proveïdor" d'una fila del llistat.
2. El servidor elimina el proveïdor.
3. El servidor mostra un altre cop el llistat de proveïdors actualitzat.

Extensions:

2.1 El proveïdor ja té despeses associades
   (a) El servidor no elimina el proveïdor.
   (b) El servidor torna a mostrar el llistat de proveïdors indicant que no es pot esborrar el proveïdor desitjat.

6.2.3 Gestions de Feines:

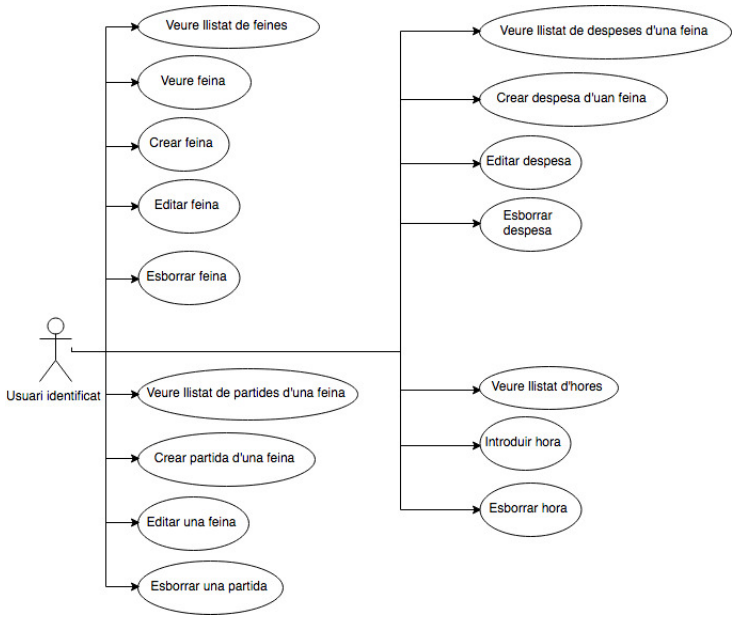


Figura 6.9: Casos d'ús d'administració de Feines i dades relacionades

Com es pot veure en la Figura 6.9, en aquest apartat també hi ha molts casos d'ús. També es pot veure facilment que podem agrupar els casos d'ús amb 4 subgrups; els relacionats amb les factures en sí, partides, despeses i hores.
Per tant, tot seguit, es definiran seguint aquest ordre.

6.2.3.1 Feines

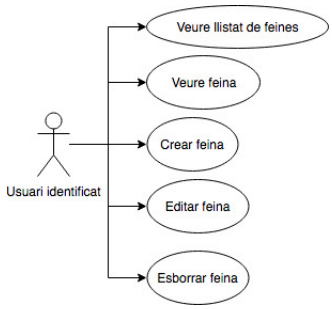


Figura 6.10: Casos d'ús d'administració de Feines

Cas d'ús 33 - Veure llistat de feines

Veure un llistat de totes les feines que hi ha a la cooperativa per tal de poder accedir a tota la informació necessària.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure un llistat de totes les feines.
Precondicions:

1. L'usuari està identificat.

Escenari principal:

1. L'usuari obre la web a la pàgina de llistat de feines.
2. El sistema mostra el llistat amb totes les feines.

Cas d'ús 34 - Crear feina

Donar d'alta una feina per tal que la informació d'aquesta estigui en el sistema i es puguin portar un control sobre ella.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol donar d'alta una nova feina.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de feines.

Escenari principal:

1. L'usuari clica al link de "crear feina".
2. El sistema mostra el formulari d'alta de feina (nom, descripció, client, responsable, nom partida, àrea partida i pressupost partida).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i dona d'alta la nova feina associada al client i responsable donats. L'hi dóna un preu hora de 32€per defecte i una data d'entrada del dia actual. També crea una

partida amb nom, àrea i pressupost donats i l'associa a la feina.
5. El servidor mostra el formulari d'edició de la feina.

Extensions:

4.1. Nom, client, responsable o dades de la partida incorrectes
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 35 - Veure feina

Veure la fitxa d'una feina per tal de poder consultar a tota la informació d'aquesta.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol consultar la fitxa d'una feina.
Precondicions:
1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de feines.

Escenari principal:
1. L'usuari clica al link de "veure feina" d'una fila del llistat.
2. El sistema mostra la fitxa de la feina amb tota la informació.

Cas d'ús 36 - Editar feina

Editar la informació d'una feina per tal d'ampliar-la o corregir-la.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol editar la informació d'una feina.
Precondicions:
1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de feines.

Escenari principal:
1. L'usuari clica al link de "editar feina" d'una fila del llistat.
2. El sistema mostra el formulari de feina (nom, descripció, estat, preu hora, client, responsable, data d'entrada, data de sortida,

observacions de la feina, observacions del client i observacions per l'administrador).
3. L'usuari edita les dades del formulari i les envia.
4. El servidor processa les dades i actualitza la informació de la feina.
5. El servidor mostra un altre cop el llistat de feines actualitzat.

Extensions:

4.1. Algunes dades obligatories estan buides (nom, client, responsable, preu hora, d'ata d'entrada, estat)
   (a) El sistema torna a mostrar el formulari mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

4.1. Data de sortida anterior a la d'entrada
   (a) El sistema torna a mostrar el formulari mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 37 - Esborrar feina

Esborrar una feina per tal d'eliminar la informació sobre aquesta del sistema.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol esborrar una feina del sistema.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la pàgina de llistat de feines.

Escenari principal:

1. L'usuari clica al link de "esborrar feina" d'una fila del llistat.
2. El servidor elimina la feina.
3. El servidor mostra un altre cop el llistat de feines actualitzat.

Extensions:

2.1 La feina ja té despeses, factures o hores associades
   (a) El servidor no elimina la feina.
   (b) El servidor torna a mostrar el llistat de feines indicant que no es pot esborrar la feina desitjada.

6.2.3.2 Partides

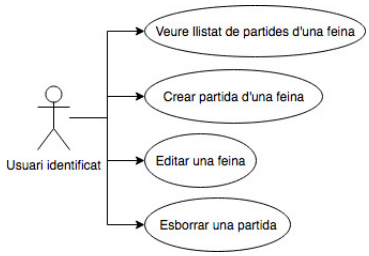


Figura 6.11: Casos d'ús d'administració de Partides

Cas d'ús 38 - Veure llistat de partides

Veure un llistat de totes les partides que hi ha en una feina per visualitzar tota la informació necessària.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure un llistat de totes les partides.
Precondicions:
1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:
1. El sistema mostra el llistat amb totes les partides que té la feina.

Cas d'ús 39 - Crear partida

Donar d'alta una partida per tal que la informació d'aquesta estigui en el sistema i es puguin associar despeses a ella.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol donar d'alta una nova partida.
Precondicions:
1. L'usuari està identificat.

2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. L'usuari clica al link de "crear partida".
2. El sistema mostra el formulari d'alta de partida (nom, àrea i pressupost).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i dona d'alta la nova partida associada a la feina.
5. El servidor mostra la fitxa de la feina actualitzada.

Extensions:

4.1. Nom, àrea o pressupost buits
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 40 - Editar partida

Editar la informació d'una partida per tal de corregir-la.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol editar la informació d'una partida.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. L'usuari clica al link de "editar partida" d'una fila del llistat de partides.
2. El sistema mostra el formulari de partida (nom, àrea i pressupost).
3. L'usuari edita les dades del formulari i les envia.
4. El servidor processa les dades i actualitza la informació de la partida.
5. El servidor mostra la fitxa de la feina actualitzada.

Extensions:

4.1. El nom, l'àrea o el pressupost estan buits
   (a) El sistema torna a mostrar el formulari mostrant un missat-

ge que explica l'error.
(b) Tornar al pas 3.

Cas d'ús 41 - Esborrar partida

Esborrar una partida per tal d'eliminar la informació sobre aquesta del sistema.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol esborrar la informació d'una partida.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. L'usuari clica al link de "esborrar partida" d'una fila del llistat de partides.
2. El servidor elimina la partida.
3. El servidor mostra la fitxa de la feina actualitzada.

Extensions:

2.1 La partida ja té despeses o hores associades
   (a) El servidor no elimina la partida.
   (b) El servidor mostra la fitxa de la feina indicant que no es pot esborrar la partida desitjada.

6.2.3.3 Despeses

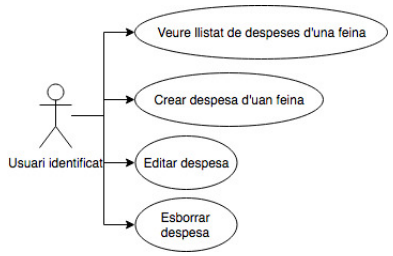


Figura 6.12: Casos d'ús d'administració de Despeses

Cas d'ús 42 - Veure llistat de despeses

Veure un llistat de totes les despeses que hi ha en una feina per visualitzar tota la informació necessària.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure un llistat de totes les despeses.
Precondicions:
1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:
1. El sistema mostra el llistat amb totes les despeses que té la feina.

Cas d'ús 43 - Crear despesa

Donar d'alta una despesa per tal que la informació d'aquesta estigui en el sistema.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol donar d'alta una nova despesa.
Precondicions:
1. L'usuari està identificat.

2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. L'usuari clica al link de "crear despesa".
2. El sistema mostra el formulari d'alta de despesa (partida, tipus de despesa, proveïdor, import estimat, marge i import real).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i dona d'alta la nova despesa associada a la partida de la feina i al proveïdor.
5. El servidor mostra la fitxa de la feina actualitzada.

Extensions:

4.1. Partida, tipus de despesa, proveïdor, import estimat, marge o import real buits
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 44 - Editar despesa

Editar la informació d'una despesa per tal de corregir-la.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol editar la informació d'una despesa.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. L'usuari clica al link de "editar despesa" d'una fila del llistat de despeses.
2. El sistema mostra el formulari de despesa (partida, tipus de despesa, proveïdor, import estimat, marge i import real).
3. L'usuari edita les dades del formulari i les envia.
4. El servidor processa les dades i actualitza la informació de la despesa.
5. El servidor mostra la fitxa de la feina actualitzada.

Extensions:

4.1. Partida, tipus de despesa, proveïdor, import estimat, marge o

import real buits
   (a) El sistema torna a mostrar el formulari mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 45 - Esborrar despesa

Esborrar una despesa per tal d'eliminar la informació sobre aquesta del sistema.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol esborrar la informació d'una despesa.
Precondicions:
1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:
1. L'usuari clica al link de "esborrar despesa" d'una fila del llistat de despeses.
2. El servidor elimina la despesa.
3. El servidor mostra la fitxa de la feina actualitzada.

6.2.3.4 Hores

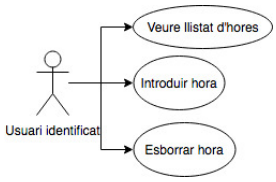


Figura 6.13: Casos d'ús d'administració d'Hores

Cas d'ús 46 - Veure llistat de d'hores

Veure un llistat de totes les hores entrades per un dia i un treballador.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure el llistat d'hores en un dia pel treballador que té associat.
Precondicions:
1. L'usuari està identificat.

Escenari principal:
1. L'usuari accedeix a la pàgina de llistat d'hores.
2. El sistema mostra el llistat amb totes les hores pel dia seleccionat (per defecte l'actual) i el treballador associat que té l'usuari.

Extensions:
2.1. Usuari sense treballador associat
   (a) El sistema mostra el llistat buit.

Cas d'ús 47 - Introduir hora

Introduir una hora d'un treballador en un dia per una feina.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol introduir una hora en nom del treballador que té associat.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està a la pàgina de llistat d'hores.

Escenari principal:

1. El sistema mostra un formulari al final del llistat d'hores per introduir hores (dia, feina, partida i minuts).
2. L'usuari omple el formulari i envia les dades.
3. El sistema introdueix una hora en la data seleccionada, per al treballador que té associat l'usuari, per a la feina i partida seleccionades i amb un total de minuts.
4. El sistema mostra el llistat d'hores actualitzat.

Extensions:

3.1. Data, feina, partida o minuts buits
   (a) El sistema mostra el llistat amb el formulari amb un missatge que explica l'error.
   (b) Tornar al pas 2.

1.1. Usuari sense treballador associat
   (a) El sistema no mostra cap formulari per introduir hores.

Cas d'ús 48 - Esborrar hora

Esborrar una hora d'un treballador en un dia per una feina.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol esborrar una hora en nom del treballador que té associat.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està a la pàgina de llistat d'hores.

Escenari principal:

1. L'usuari selecciona el link "esborrar hora" d'una fila del llistat d'hores.
2. El sistema esborra l'hora.
3. El sistema mostra el llistat d'hores actualitzat.

6.2.4 Facturació:

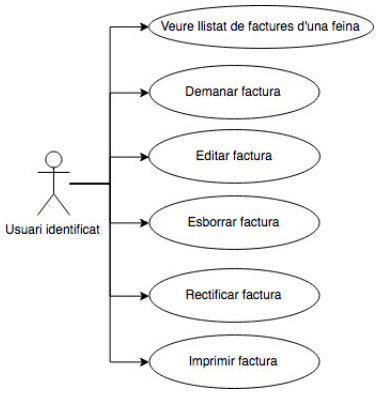


Figura 6.14: Casos d'ús de facturació

Cas d'ús 49 - Veure llistat de factures

Veure un llistat de totes les factures que hi ha en una feina per poder accedir a tota la informació necessària.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol veure el llistat de factures d'una feina.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. El sistema mostra el llistat amb totes les factures que té la feina.

Cas d'ús 50 - Demanar factura

Donar d'alta una factura per tal que la informació d'aquesta estigui en el sistema.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol donar d'alta una nova factura d'una feina.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. L'usuari clica al link de "crear factura".
2. El sistema mostra el formulari d'alta de factura (data i imports per partides).
3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i dona d'alta la nova factura associada a la feina i amb el mateix client que ja té la feina. La factura tindrà l'estat de *demanada* i un iva i irpf per defecte. També és crearà un número de factura seguint la sèrie anual.
5. El servidor mostra la fitxa de la feina actualitzada.

Extensions:

4.1. Data o algun import de partides buits
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

Cas d'ús 51 - Editar factura

Editar una factura per tal d'ampliar o corregir la informació d'aquesta en el sistema.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol revisar una factura d'una feina per enviar-la al client.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. L'usuari clica al link de "revisar factura" en una fila del llistat de factures.
2. El sistema mostra el formulari de factura (data, iva, irpf, client, imports per partides i estat).

3. L'usuari omple el formulari i envia les dades.
4. El servidor processa les dades i actualitza la informació de la factura.
5. El servidor mostra la fitxa de la feina actualitzada.

Extensions:

4.1. Data, iva, irp, client o algun import de partides buits
   (a) El sistema torna a mostrar el formulari d'alta mostrant un missatge que explica l'error.
   (b) Tornar al pas 3.

5.1. Any de la data modificat
   (a) El sistema generarà un nou número de factura seguint la sèrie del nou any.
   (b) Tornar al pas 6.

Cas d'ús 52 - Esborrar factura

Esborrar una factura per tal d'eliminar la informació d'aquesta en el sistema.

Actor principal: Usuari identificat

Disparador: Un usuari identificat vol esborrar una factura d'una feina.

Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. L'usuari clica al link de "esborrar factura" en una fila del llistat de factures.
2. El sistema elimina la factura.
3. El servidor mostra la fitxa de la feina actualitzada.

Extensions:

2.1. Factura ja revisada, pagada o rectificada
   (a) El sistema no elimina la factura.
   (b) El servidor mostra la fitxa de la feina mostrant un missatge que explica l'error.

Cas d'ús 53 - Rectificar factura

Rectificar una factura per tal de corregir-la un cop ja ha estat enviada a hisenda i no es pot editar.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol rectificar una factura d'una feina.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. L'usuari clica al link de "rectificar factura" en una fila del llistat de factures.
2. El sistema marca com a *rectificada* la factura.
3. El sistema crear una factura idèntica a la rectificada. Aquesta nova factura té un altre número que seguirà la sèrie paral·lela per a les factures rectificades.
4. El servidor mostra el formulari de la factura nova.

Extensions:

2.1. Factura no està enviada o pagada
   (a) El sistema no rectifica la factura.
   (b) El servidor mostra la fitxa de la feina mostrant un missatge que explica l'error.

Cas d'ús 54 - Imprimir factura

Imprimir una factura per tal d'enviar-la al client.

Actor principal: Usuari identificat
Disparador: Un usuari identificat vol imprimir una factura d'una feina.
Precondicions:

1. L'usuari està identificat.
2. L'usuari està veient la fitxa d'una feina.

Escenari principal:

1. L'usuari clica al link de "imprimir factura" en una fila del llistat de factures.
2. El sistema mostra una finestra amb tota la informació de la factura preparada per ser impresa.
3. L'usuari selecciona l'opció d'imprimir del navegador.

6.3 Model conceptual

Els models conceptuals descriuen les estructures de dades i restriccions que té un sistema. Utilitzant un model conceptual podem veure quins elements intervenen en el sistema i les relacions que tenen entre ells.

A continuació es mostrarà l'esquema conceptual general de la nova Intranet amb les seves restriccions textuals. També es farà una petita descripció més detallada de cada part de l'esquema per aclarir totes les relacions.

6.3.1 Esquema conceptual

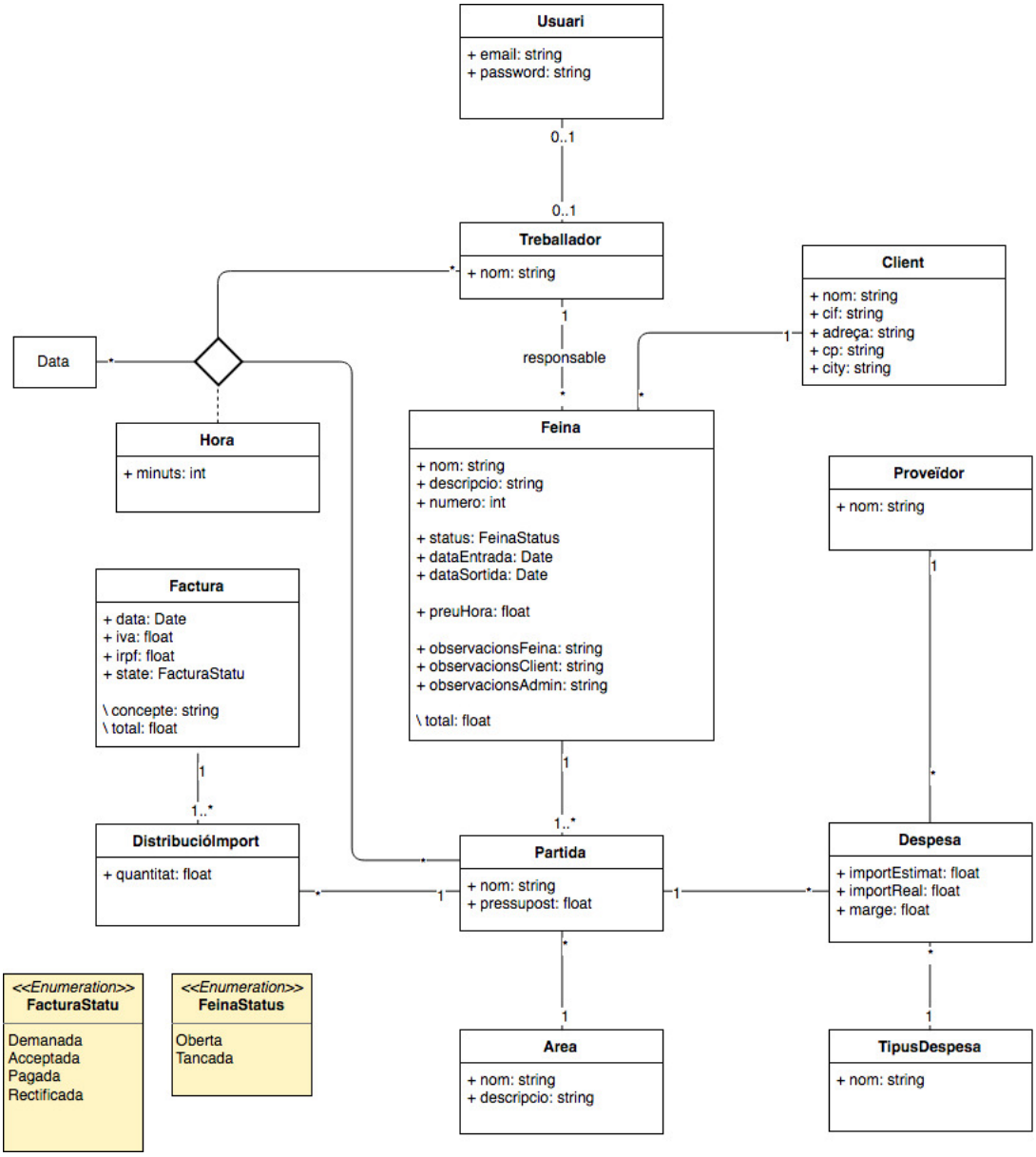
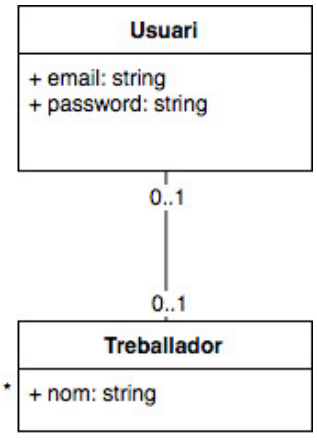


Figura 6.15: Esquema conceptual

6.3.2 Restriccions d'integritat

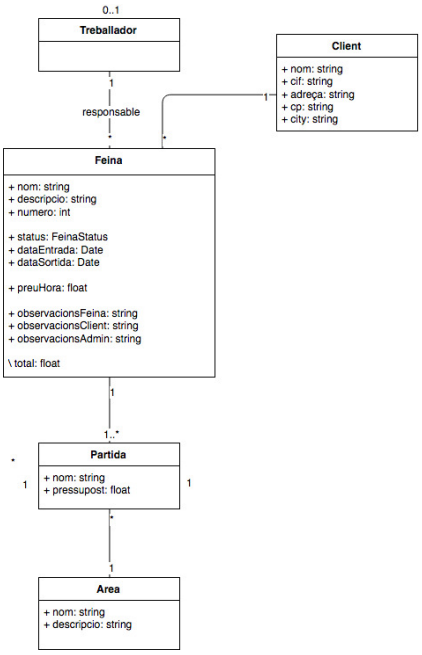
1. La *dataEntrada* d'una **Feina** ha de ser anterior a la *dataSortida*.
2. El *concepte* d'una **Factura** és un string compost pel *numero* i *nom* de la **Feina**, més la suma de *quantitat*s de les **DistribucionsImport**s.
3. Per a cada **Factura** hi ha tantes **DistribucionsImports** com **Partides** de la **Feina** corresponent.
4. El *total* d'una **Factura** és la suma de *quantitat*s de les **DistribucionsImport**s.
5. El *total* d'una **Feina** és la suma de *pressupost*s de les **Partides** relacionades.

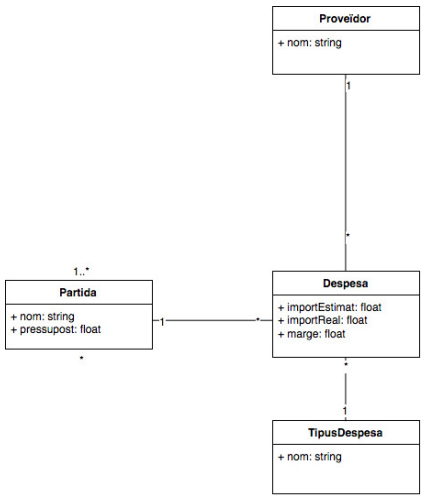
6.3.3 Descripció



Com es pot veure, en el sistema es separen els **Usuari**s dels **Treballador**s. En general cada **Usuari** tindrà un **Treballador** assignat, però es pot donar el cas que un **Usuari** no en tingui cap, per tant aquest **Usuari** no podrà entrar hores. També es pot donar el cas que un **Treballador** no estigui associat a cap **Usuari**, això voldrà dir que ja no es podran entrar més hores per aquest treballador (per exemple, perquè ja no treballa a la cooperativa). A part, en un futur, està previst qui hi hagui diferents rols d'**Usuari**s, i en concret hi haurà els rols d'administrador i soci, que podran entrar hores per a qualsevol treballador encara que els socis només tindran associat un **Treballador** i l'administrador cap (ja que és un comptable extern).

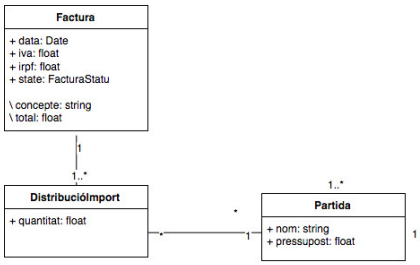
En la següent part de l'esquema es pot veure la figura més important amb les seves relacions més directes. Aquesta és la **Feina**, que tindrà un **Treballador** assignat com a responsable. A la vegada cada feina es realitza per un**Client**, i una **Feina** està organitzada en una o més **Partide**s. També es pot veure que les **Partide**s estan relacionades amb una **Àrea** de l'empresa. Això serà útil en un futur quan es vulguin extreure informes.

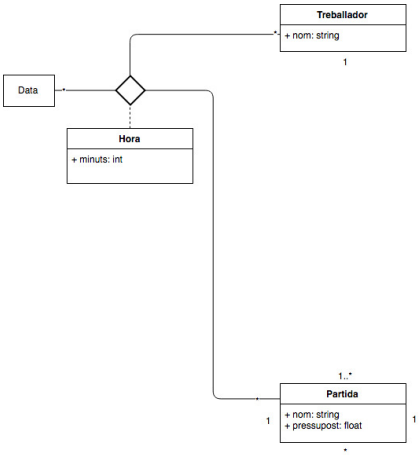




Una altra relació que es fa amb les **Feines** són les **Despeses**. Aquestes no estan directament relacionades amb les **Feines**, sinó que s'associen a les **Partides** d'una **Feina**, així més endavant es podran consultar els gastos per **Partides** i per tant també per **Àrees**. Com es pot comprovar, una **Despesa** es paga a un **Proveïdor** i té un **TipusDespesa** associat.

Una altra estructura important en el sistema són les **Factures**. Com es pot veure, les **Factures** estan compostes per una o més **DistribucióImport**. Les **DistribucionsImports** a l'estar relacionades amb **Partides** defineixen quina quantitat del total de la **Factura** s'ha de "comptabilitzar" a cada **Partida** de la **Feina**. Així, en última instància, es podrà saber la facturació per cada **Àrea**.



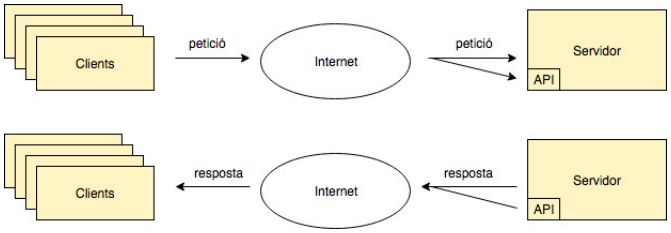


Per acabar, donada una **Partida** d'una **Feina** i un **Treballador**, es podran registrar el temps dedicat a la **Feina** per aquell **Treballador** en un dia concret. Com es veu, en un mateix dia, per un mateix **Treballador** i per una mateixa **Partida** es podran entrar més d'una **Hora**.

7. Disseny

7.1 Arquitectura física

En el context d'aquesta Intranet, s'utilitzarà la típica arquitectura física web basada en *client-servidor*.



Com es pot veure, un client (navegador web d'un ordinador) fa una petició d'una pàgina en concret de la Intranet al servidor i aquest respon a la petició. La resposta és una plana web HTML amb el corresponent CSS i Javascript.

També es pot observar, que el servidor disposa una API, en concret una *REST API*. Això és degut al fet que hi ha certes interfícies de la Intranet que són més dinàmiques i per tant s'han creat amb el framework de Javascript ReactJs. Per tant, en aquests casos, el client en la primera petició es descarrega el codi HTML, CSS i Javascript (tal com s'ha dit abans), però en les pàgines on s'executa un component de React, aquest fa peticions posteriors de forma asíncrona a través de la API per consultar i enviar dades al servidor. Les dades d'aquestes peticions són enviades en format *JSON*.

Gràcies a l'arquitectura lògica que s'ha utilitzat (explicada en el següent apartat), resulta molt senzill mantenir els dos tipus de peticions (normals i API).

7.2 Arquitectura lògica

En aquest apartat s'explicarà com s'organitzen i interactuen els diferents components del sistema per tal d'aconseguir el funcionament desitjat. Hi

ha moltes formes d'organització, i aquestes componen les diferents arquitectures conegudes. Aquestes arquitectures poden anar des d'un clàssic com *l'arquitectura en 3 capes*, fins a conceptes més evolucionats com *l'arquitectura hexagonal*. Podríem creure que són conceptes totalment diferents, però en general totes les arquitectures comparteixen moltes coses, com ara la divisió i agrupació dels diferents components del sistema en capes. El que varia entre cada arquitectura és justament on es troben aquestes divisions i quantes n'hi ha. També hi ha "regles" de com s'han de comunicar els components que varien entre les arquitectures. Per tant podem definir una arquitectura com un conjunt de regles que defineixen com separar i organitzar en capes els diferents components del sistema i com sa de dur a terme la comunicació entre aquests elements i capes.

A continuació s'explicarà quines són aquestes regles en una arquitectura neta, i es podrà comprovar que en veritat moltes d'elles vénen d'altres arquitectures anteriors, i que l'arquitectura neta el que fa és quedar-se i combinar el "millor" de les arquitectures anteriors i emfatitzar algunes regles de comunicació entre components que en altres arquitectures no es donava tanta importància.

7.2.1 Clean arquitecture

7.2.1.1 Regles bàsiques

Abans d'explicar com funciona una arquitectura neta és necessari explicar algunes regles o principis bàsics amb els quals es fonamenta aquesta arquitectura.

Primer de tot s'han d'explicar els principis **SOLID**. SOLID és un acrònim, introduït pel mateix autor (Robert C. Martin), que agrupa 5 principis bàsics de la programació orientada a objectes. L'objectiu d'això és que els desenvolupadors utilitzin aquests 5 principis en la mesura del possible per així aconseguir millors dissenys de software. Estan molt estesos en entorns de desenvolupament amb metodologies àgils i amb pràctiques com el TDD.
De forma molt breu, els 5 principis SOLID són els següents:

- **Principi de responsabilitat única** (*Single responsability principle*). Que diu que un objecte només hauria de tenir una sola responsabilitat.

- **Principi d'obert/tancat** (*Open/closed principle*).
  El concepte que les entitats de software (classes) han d'estar obertes a (permetre) la seva extensió, però tancades a la modificació.
- **Principi de substitució de Liskov** (*Liskov substitution principle*).
  Aques principi diu que els objectes d'un programa haurien poder de ser reemplaçats per instàncies de subtipus d'aquests sense alterar el correcte funcionament del programa.
- **Principi de segregació de la interfície** (*Interface segregation principle*).
  La idea que moltes interfícies de client específiques són millor que una interfície genèrica.
- **Principi d'inversió de dependències** (*Dependency inversion principle*).
  Que diu que les entitats d'un software han de dependre d'abstraccions més que d'implementacions.

Com s'ha dit, tots els principis són importants, però el principi d'Open/Close molts cops és considerat el més important. Si analitzem el que diu aquest principi, veurem que busca que un sistema software es pugui ampliar sense haver de modificar el que ja hi ha fet. Objectiu que, si es compleix, facilita molt el fet de treballar amb metodologies àgils, les quals van creant productes de forma incremental. Per tant, també podem veure la resta de principis com un mitjà per a facilitar aquest.

Per poder explicar el funcionament d'una arquitectura neta, també és necessari destacar el principi d'**inversió de dependències**, ja que té un paper clau en aquesta. Una forma d'aconseguir complir aquest principi és aplicant el patró d'**injecció de dependències**.
Bàsicament el que defineix aquest patró és que les dependències que pugui tenir una classe del sistema no les pot "generar" la mateixa classe, sinó que se l'hi ha de passar des de fora; injectar. Aquesta injecció es pot fer de moltes formes, depenent de les característiques del llenguatge de programació i de les eines o frameworks que s'utilitzin, però les dues formes més bàsiques són a través dels mètodes constructors de les classes o com a paràmetres en les crides.

A part dels principis SOLID, una altra regla que cal explicar abans d'entrar a definir una arquitectura neta, és la **llei de la dependència**.

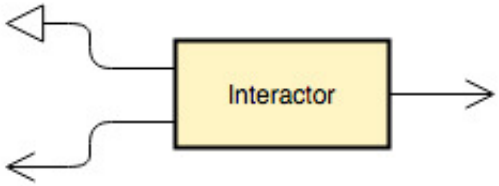
Com s'ha dit anteriorment, un sistema software té un seguit de capes amb les quals s'organitzen els objectes del sistema, i cada arquitectura defineix quines capes s'han de crear i on estan les fronteres d'aquestes. Però en el cas de l'arquitectura neta també es defineix de forma molt explícita quines dependències pot haver-hi entre aquestes capes. Com es veurà més endavant, l'arquitectura neta defineix un seguit de capes concèntriques, on en el centre hi ha els objectes més "importants" i en els exteriors els "detalls". Aleshores es defineix que les dependències entre objectes d'altres capes sempre han d'anar cap endins. Per tant, i com a exemple, cap objecte de la capa més cèntrica pot saber res d'objectes externs a aquesta capa. Aquí és on entra en joc el principi d'inversió de dependències, ja que ens permetrà aconseguir complir aquesta llei.

7.2.1.2 Interactors, Entities i Boundaries

Com s'ha explicat en l'apartat de Estat de l'art, abans de parlar de *Clean Architecture*, Robert C. Martin va escriure sobre el concepte de ***Screaming Arquitecture***.

La idea més bàsica i resumida que vol transmetre l'autor amb aquest concepte, és el fet que una persona hauria de ser capaç de saber que fa una aplicació simplement mirant els noms de les carpetes i arxius del codi font. En canvi, avui en dia si mires els noms de les carpetes i arxius, generalment, sabràs de quin llenguatge de programació es fa ús, quin tipus d'aplicació és (web, mobil, escriptori), quines eines fan servir per desenvolupar-la (IDEs i/o frameworks), inclús pots detectar patrons de disseny i arquitectures fàcilment, però no podràs saber el més important que és; que fa aquesta aplicació?

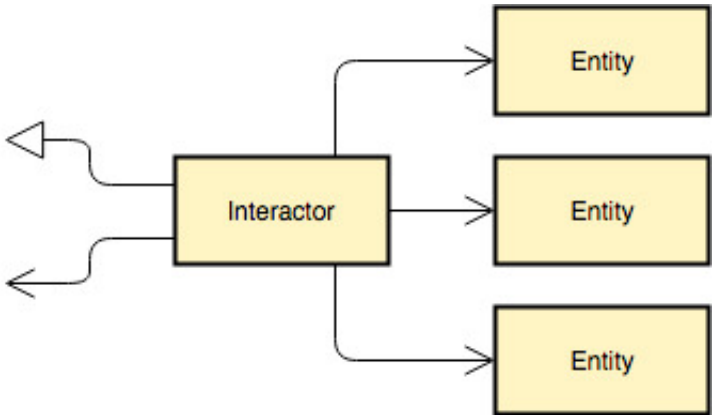
És per això que Robert C. Martin parla dels **Interactors**.



En l'arquitectura neta, els Interactors són un element central d'aquesta. Bàsicament són objectes que "codifiquen" les *lògiques de negocis* descrites en els casos d'ús, o podem dir que els Interacors contenen les *lògiques de negocis específiques de l'aplicació.*
Per tant, si mirem el conjunt d'arxius dels Interactors amb noms tals com *GetUsersListInteractor, CreateUserInteractor, etc*, podrem fer-nos una idea ràpida del que fa aquesta aplicació.

Com s'ha dit, en una arquitectura neta, les lògiques de negoci específiques de l'aplicació estan en els Interactors. En general aquestes es refereixen a les interaccions entre els usuaris i el sistema, però no a lògiques de negocis més genèriques o d'alt nivell.

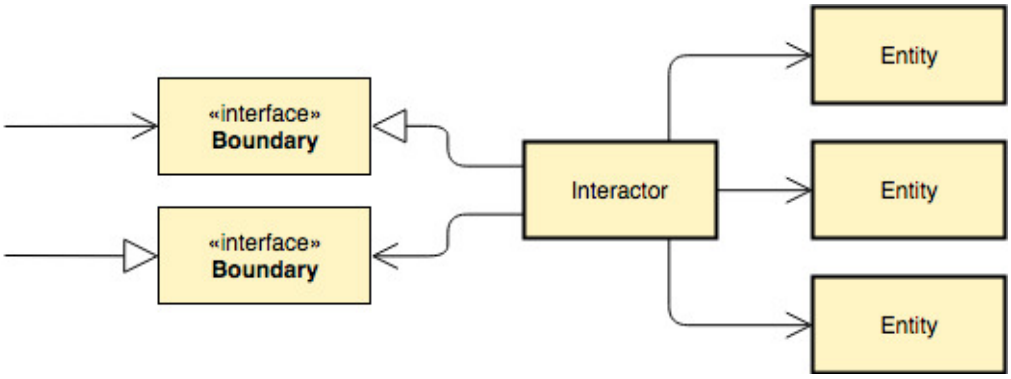
En una arquitectura neta, aquestes lògiques de negocis més genèriques es troben en les **Entities**.



Aquestes Entities són objectes que com s'ha dit contenen les lògiques de negocis genèriques, i són la part més cèntrica de l'arquitectura, cosa que fa que cap Entity pot conèixer res fora d'aquesta capa.
Per tant, seran els Interactors els encarregats de fer servir i coordinar les diferents Entities per aconseguir els seus objectius.

Amb aquests dos artefactes (Interactors i Entities) tenim localitzada tota la lògica de negoci del nostre sistema, i hem aconseguit localitzar-la de forma molt concreta i aïllar-la per tal que sigui el més fàcil possible treballar amb ella.

Un cop tenim aquestes dues capes ens falta poder fer que els Interactors es comuniquin amb la resta del sistema. Un problema que tenim aquí és que, per la *llei de la dependència*, els Interactors no poden saber res de la resta del sistema que està més cap "enfora", o sigui, només tenen coneixement d'altres Interactors i de les Entities.
Per tant, per poder aconseguir aquesta comunicació, entren en joc els **Boundaries**.

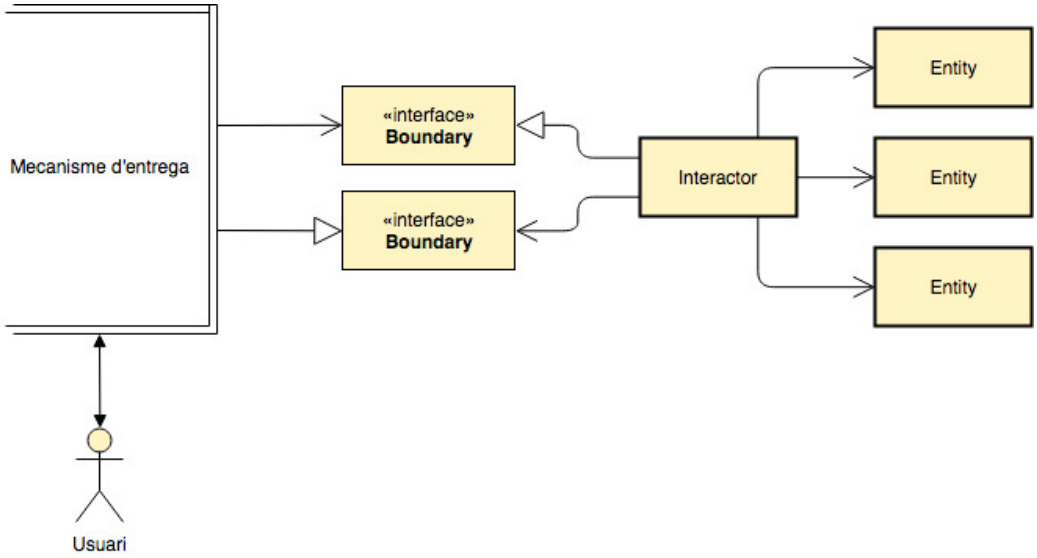


Aquí és on es veu l'ús de la llei d'inversió de dependències, ja que els Boundaries seran interfícies declarades en una capa i implementades en una altra, així s'aconsegueix una comunicació entre les capes externes i internes i a la vegada que les capes internes no tinguin cap dependència cap a fora, cosa que crea l'efecte de *plugin*; la capa externa esdevé un plugin de la interna que és fàcilment intercanviable per un altre.

En el cas dels Interactors en concret, en la figura anterior, es pot veure que s'utilitzarà una primera interfície, definida per l'Interactor i implementada per algun objecte de les capes externes, com a via per una comunicació de fora cap endins. I una segona interfície, definida en una capa exterior i implementada per l'Interactor, s'utilitzarà com a via de comunicació de dins cap enfora.

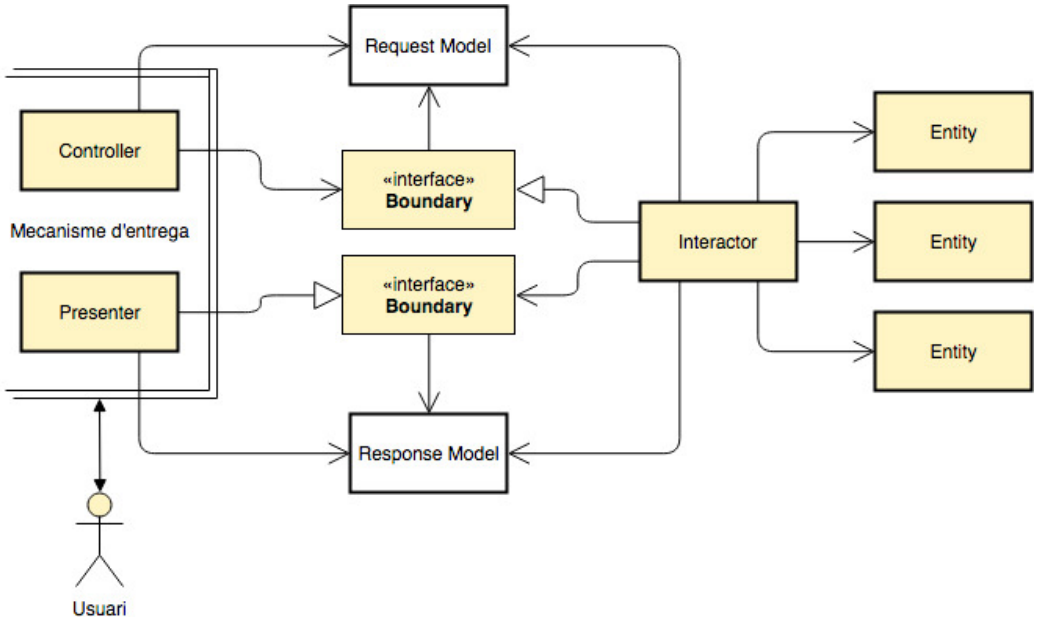
7.2.1.3 Mecanisme d'entrega

Amb tota l'arquitectura explicada fins ara tenim dues capes on estan localitzades totes les lògiques de negoci del sistema i una forma de comunicar-nos amb elles. Una part que tindrà qualsevol aplicació i que es comunicarà amb aquestes capes és el que en Robert C. Martin anomena *mecanisme d'etrega*.



El mecanisme d'etrega pot ser qualsevol, des d'una web, una aplicació mòbil fina a una linea de comandaments, i això ha de ser un "detall" dins la nostra aplicació, ja que segurament serà una de les parts que més canvis pateixi en el temps. Aquí és on segurament trobarem els típics patrons de MVC o MVP (Mocel View Controller o Presenter).

Una possible implementació d'aquest mecanisme d'entrega podria ser la seguent:

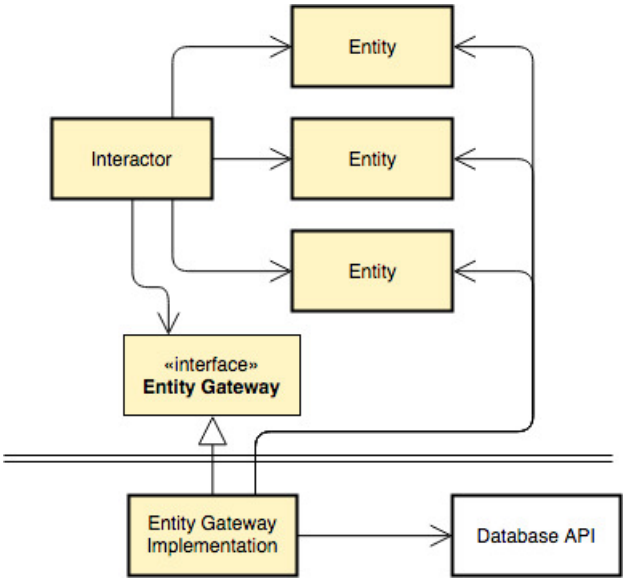


Com podem veure, tindriem un **Controller** que seria l'encarregat de rebre la petició d'un usuari, crear un **Request Model** (que sería una simple collecció de dades) i enviarlo cap a l'Interactor a través del Boundary. L'Interactor al rebre el Request Model el llegiria i executaria les seves lògiques de negoci juntament amb les de les Entities per acabar generant un **Response Model** que enviaria cap a un **Presenter** a través dels Boundaries un altre cop. I finalment el Presenter seria l'encarregat de mostrar el resultat final a l'usuari en el format corresponent.

7.2.1.4 Mecanisme de persistència

Una altra part molt comuna és la del mecanisme de persistència. Aquesta és la forma en la qual es guarden les dades per tal que estiguin sempre accessibles, generalment en una Base de Dades. Aquesta part potser és un dels grans trets diferenciadors d'aquesta arquitectura amb contraposició d'altres més clàssiques que on la Base de Dades té un paper més important i central. En canvi en una arquitectura neta el mecanisme de persistència passa a ser un detall d'implementació que pot ser fàcilment intercanviable per un altre.

La forma d'aconseguir això és molt semblant al mecanisme d'entrega; bàsicament tornant a fer ús de la llei de dependència, del principi d'inversió de dependències i del patró d'injecció de dependències, aconseguim el següent esquema:



Com es pot veure, a l'Interactor li tornem a injectar un Boundary (que en aquest cas s'ha anomenat **Entity Gateway**), el qual és només una interfície, la implementació de la qual està a la capa del mecanisme de persistència. Aquest objecte **Entity Gateway Implementation** serà l'encarregat d'accedir a les dades per generar o guardar les Entities necessàries per als Interactors. És evident que segurament hi haurà més d'una interfície i implementació per a cada tipus d'Entities, i que aquí és on segurament s'implementarà algun patró d'accés a les dades (*Active Record, DAO, etc*).

7.2.1.5 Resum

Amb totes aquestes parts explicades fins ara, podríem dir que tenim l'arquitectura bàsica per a poder desenvolupar un sistema software.

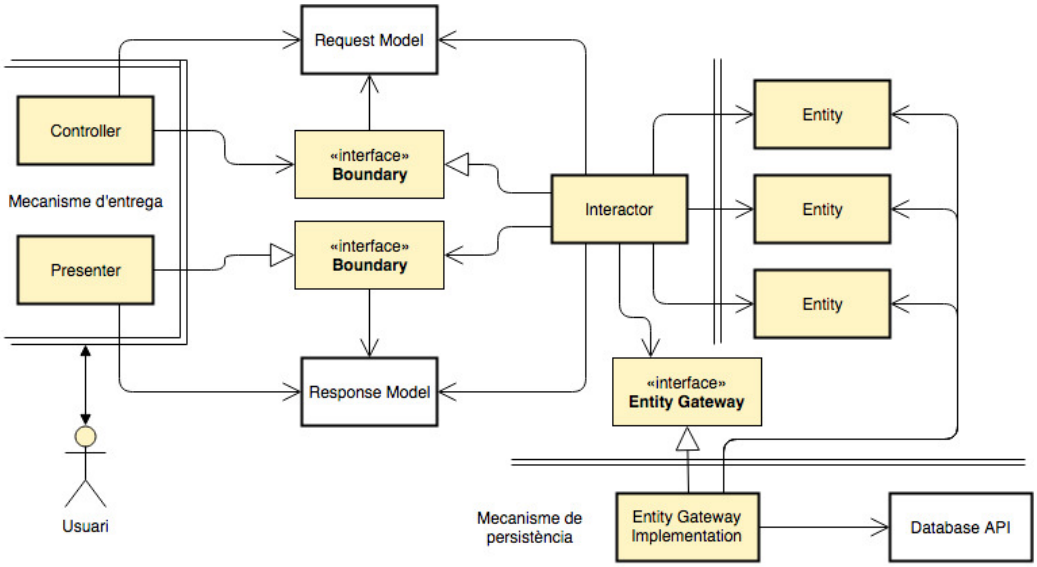


Figura 7.1: Diagrama de classes bàsic d'una arquitectura neta

Veient l'anterior diagrama és fàcil detectar 4 capes en l'arquitectura. Començant per la capa que podríem dir *Domini* la qual conté les Entities, seguidament tindríem els Interactors amb els Boundaries i després a un mateix nivell les capés de presentació i persistència.
Cal dir que no és obligatori que hi hagui aquestes capes, segons els requeriments del sistema poden haver-hi menys capes (per exemple no volem persistència), o tenir-ne més (en aplicacions mòbils podem tenir més capes per diferents serveis del dispositiu (GPS, giroscopi, etc)).

També és habitual trobar la següent imatge com a resum d'una arquitectura neta:

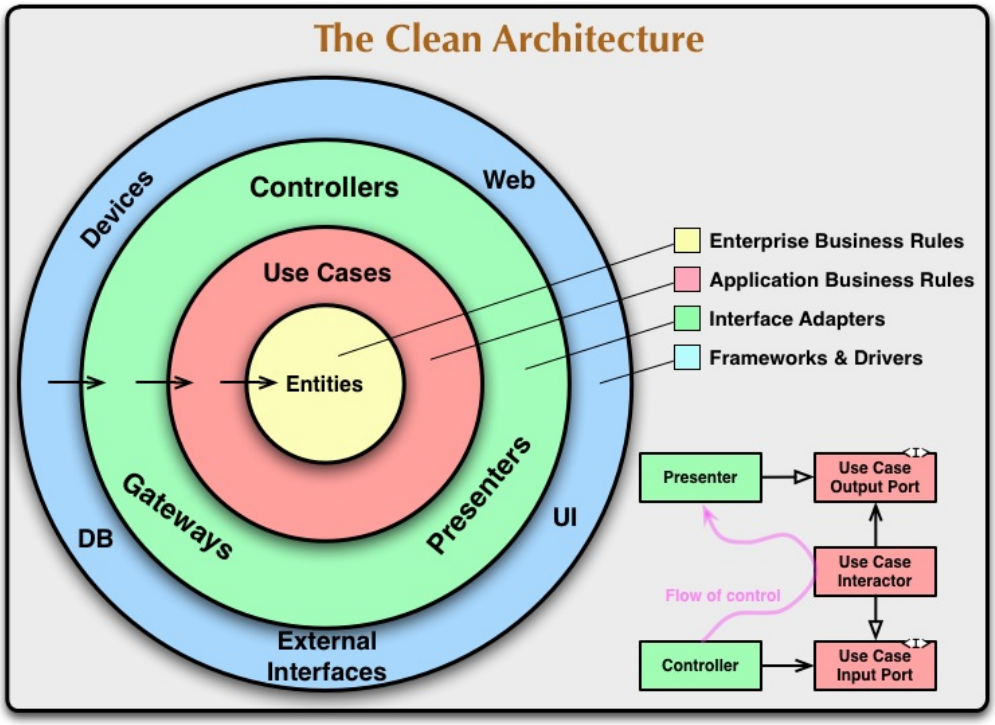


Figura 7.2: Diagrama resum de les parts més importants d'una arquitectura neta

Com es pot veure, tot i que l'anterior imatge explica la mateixa arquitectura i també es defineixen 4 capes, aquestes no coincideixen del tot amb les anteriors comentades.

En aquest cas es defineixen dues capes interiors que sí que coincideixen amb les anteriors comentades (Entities i Use Cases, que són els Interactors), però seguidament es mostra una capa on s'agrupen totes les implementacions dels diferents Boundaries que puguin haver-hi en el sistema, i per acabar una quarta capa on trobaríem totes les implementacions més concretes del sistema (DB, UI, etc).

També es pot veure que es mostra la *Llei de la dependència* i un exemple de com "travessar" fronteres entre capes sense trencar aquesta llei.

8. Implementació

En l'apartat de Disseny s'ha explicat com s'organitzen els diferents elements d'una arquitectura neta en l'àmbit teòric, però és obvi que en crear una implementació concreta, l'arquitectura final no serà igual a la teòrica. A més, a l'utilitzar metodologies àgils en aquest projecte, no es parteix d'un disseny final ja definit completament que "només" s'ha d'implementar, sinó que s'ha anat construint iteració a iteració i encara no es pot considerar del tot acabat, per això encara es pot diferenciar més l'arquitectura implementada actual de la teòrica.

En termes generals, l'arquitectura implementada fins al moment és la següent:

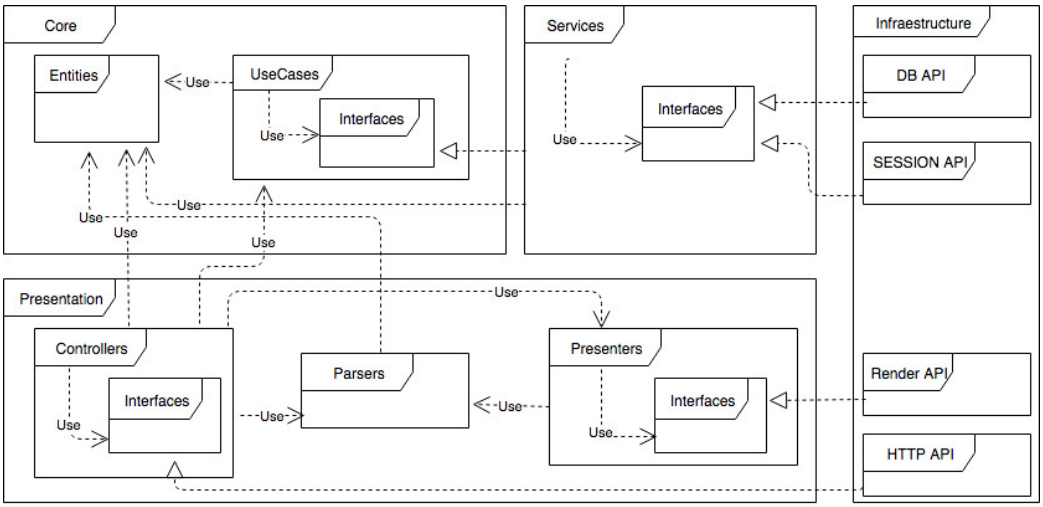


Figura 8.1: Diagrama d'implementació d'una arquitectura neta amb PHP

Observant la direcció de les fletxes de dependències es pot visualitzar quines són les capes concèntriques i quines les externes.

Per començar, es pot veure que des de la capa d'Entities no surt cap fletxa cap enfora i totes són cap endins. Per tant vol dir que aquesta capa, tal com hauria de ser, és la més cèntrica de totes. I seguint analitzant el diagrama d'aquesta manera, fàcilment es veu que la següent capa és la dels UseCases, seguida de les capes de Services i Presentation, que estan a un

mateix nivell. Per acabar, la capa més externa és la de Infraestucture.

Tot seguit s'explicaran les diferents capes que componen l'arquitectura i possibles millores a fer en un futur.

8.1 Capes

8.1.1 Entities i UsesCases

Aquestes dues capes tal com diu l'arquitectura neta teòricament contenen tots els objectes que codifiquen totes les lògiques de negoci. La capa de UseCases (Interactors en l'explicació teòrica) té una subcapa amb els Boundaries (Interfaces) necessaris pel flux de dades.
Com es pot veure, aquestes capes s'han agrupat en una més gran anomenada **Core**, simplement per fer més explícit que aquestes capes componen el nucli de la Intranet.

8.1.2 Services

Seguidament veiem que al costat del *Core* trobem la capa de **Services**, que bàsicament conté totes les implementacions necessàries per a les Interfaces de la capa de UseCases. A la vegada, aquesta capa també declara un seguit d'Interfaces per mirar "d'externalitzar" encara més detalls d'implementació com ara la connexió concreta a la base de dades, o l'accés al sistema de sessions del servidor.

8.1.3 Presentation

També al voltant de la capa de *Core* trobem la capa de **Presentation**. En aquesta capa podem veure que tenim objectes **Controllers, Parsers i Presenters**, que bàsicament estan implementant una espècie de patró MVPC (Model View Presenter & Controller). També es pot veure que en la capa de Presenters estan definides unes Interfaces, que bàsicament seran necessàries

perquè els Presenters utilitzin algun tipus de motor de render HTML sense dependre d'aquest.

8.1.4 Infraestructure

Per acabar, la capa més externa és la de **Infraestructure**. En aquesta capa trobem les implementacions concretes dels detalls més externs. És aquí on farem ús de frameworks i llibreries externes i internes de PHP.

En concret, hi ha una implementació per fer la connexió a la base de dades utilitzant la llibreria PDO pròpia de PHP. En aquest sentit també hi ha una implementació per llegir i escriure fitxers en disc que es va fer servir com a base de dades durant les primeres iteracions, i que posteriorment es va deixar de fer servir per començar a utilitzar la implementació que connectava a la base de dades sense necessitat de tocar res dins el Core.
A part també ens podem trobar una implementació d'un SessionManager que fa servir les mateixes variables de PHP de $SESSION.

També hi ha totes les implementacions necessàries per a la capa de Presenter, això és una implementació d'un motor de render de HTML utilitzant la llibreria de Twig i el component de Form del framework de Symfony.

Aquí també trobem el *punt d'entrada* del servei web, però queda aïllat en una espècia de "HTTP API" on fem ús del framework Silex per simplificar el codi. Aquest component bàsicament és l'encarregat de llegir les peticions HTTP del servidor, i redirigir-la al Controller corresponent. I quan rep una resposta del Controller, genera una resposta HTTP que retorna al client.

Dins d'aquesta capa, en concret juntament amb el motor de render, es troba el codi SASS i Javascript per generar els estils CSS de les pàgines i tot el codi Javascript necessari per donar dinamisme a la Intranet, incloent-hi els components fets amb el framework ReactJS.

8.2 Testing

Com s'ha comentat en la secció de Mètodes de validació, una part dels tests es fan de forma manual durant les demostracions en les reunions de seguiment

i en el dia a dia del desenvolupament. Però una altra part molt important són els tests automàtics que ajuden a donar robustesa al sistema i assegurar que les modificacions que es van fent no introdueixen cap error.

Per a realitzar aquests tests automàtics s'ha creat tota una *Suite*[1] de tests unitaris i d'integració. Gràcies a l'arquitectura implementada ha sigut molt fàcil poder crear aquests tests, ja que el nivell d'acoblament entre els diferents elements del sistema és molt vaig i per tant és molt fàcil crear tests on totes les dependències de l'element testejat estan reemplaçades per algun tipus de *TestDouble*[2]. A nivell més pràctic, en el projecte podem trobar una carpeta de tests, on està replicada tota l'estructura de classes però amb els tests (a part d'algunes classes extres d'ajuda per a realitzar els tests).

Com hem dit, s'han creat tests unitaris per a tot el sistema, exceptuant algunes capes més externes. A dia d'avui hi ha 468 tests unitaris creats que amb 844 *assertions* validen tota la integritat del sistema.

```
[12:41 ]-[vagrant@intranet]-[/var/www/intranet]
$ ./tests u
PHPUnit 6.0.13 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.1.3-2+deb.sury.org~xenial+1 with Xdebug 2.6.0-dev
Configuration: /var/www/intranet/phpunit-unit.xml

............................................................  63 / 468 ( 13%)
............................................................ 126 / 468 ( 26%)
............................................................ 189 / 468 ( 40%)
............................................................ 252 / 468 ( 53%)
............................................................ 315 / 468 ( 67%)
............................................................ 378 / 468 ( 80%)
............................................................ 441 / 468 ( 94%)
...........................                                  468 / 468 (100%)

Time: 2.23 seconds, Memory: 12.00MB

OK (468 tests, 844 assertions)
```

Figura 8.2: Execució de Tests unitaris

També s'han creat tests d'integració entre les capes de serveis i infraestructura referents a les classes d'storage i la base de dades. En total hi ha 99 tests amb 128 *assertions* que validen totes les transaccions amb la base de dades.

[1]Test Suite: `https://en.wikipedia.org/wiki/Test_suite`
[2]TestDouble: `https://martinfowler.com/bliki/TestDouble.html`

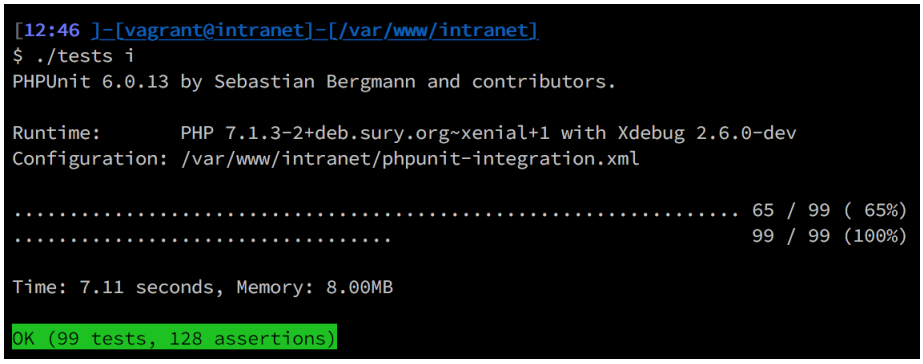


Figura 8.3: Execució de Tests d'integració

Com es veu, l'execució dels tests unitaris és molt més ràpida (2.23 segons) que la dels tests d'integració (7.11 segons), tot i que el nombre de tests unitaris quasi és 5 vegades més elevat que els d'integració. Com que hem estat utilitzant la tècnica de TDD per a tot el desenvolupament, és important tenir això en consideració, ja que si l'execució dels tests és massa llarga, fa inviable utilitzar correctament aquesta tècnica. Per això es van crear un script que permet executar uns tipus de tests o uns altres. Així, en el dia a dia, els tests unitaris s'executen molts cops, però els d'integració només en tocar alguna cosa relacionada amb la base de dades. També, per mirar d'incrementar la velocitat de desenvolupament, s'ha acabat configurant el IDE utilitzat per poder executar els tests i veure els resultats de forma més ràpida i sense passar per la consola.

Com es pot veure, la quantitat de tests que hi ha és força elevada, i aquests no pararan d'augmentar en paral·lel amb l'aplicació. Per això és molt important organitzar els tests de forma que sigui fàcil mantenir-los, i interpretar-los ràpidament quan algun falli. El fet de tenir una estructura de tests que és una rèplica de l'estructura de l'aplicació en sí facilita molt el trobar els arxius de tests quan es busquen.

També, en escriure els tests s'ha seguit sempre una nomenclatura per facilitar la seva interpretació. En general, en tots els sistemes, els tests segueixen un mateix patró; primer es configura l'element i el seu entorn per tenir controlat l'estat inicial, aleshores s'executa el que es vol testejar i finalment es valida que el resultat és el correcte. Aquest patró el podríem anomenar *Given, When, Then*. És per això que el nom de tots els tests indiquen el *Given, When, Then* del test en concret, i internament en els tests es pot

veure molt bé la separació en el codi de les tres àrees. Com es pot veure en la següent imatge, com a resultat d'aplicar aquesta nomenclatura fa que sigui molt fàcil saber que està provant cada test:

```php
public function test_editRequestedInvoice_saveCorrectUpdatedEntityToRepository() {
    //Expect
    $this->invoicesRepository->expects( matcher: $this->once())
                             ->method( constraint: "saveInvoice")
                             ->with( ...arguments: $this->editedInvoice);

    //When
    $this->sut->editRequestedInvoice( request: $this->editRequest);
}

public function test_editRequestedInvoice_returnCorrectUpdatedEntity() {
    //When
    $returnedEntity = $this->sut->editRequestedInvoice( request: $this->editRequest);

    //Then
    $this->assertEquals( expected: $this->editedInvoice,  actual: $returnedEntity);
}

public function testWrongInvoiceId_editRequestedInvoice_throwInvoiceNotFoundException() {
    //Expect
    $this->expectException(InvoiceNotFoundException::class);

    //Given
    $this->editRequest->invoiceId = 999;

    //When
    $this->sut->editRequestedInvoice( request: $this->editRequest);
}

public function testRequestWithEditedDateWithDifferentYear_editRequestedInvoice_updateInvoiceNumber() {
    //Given
    $editedDate              = new \DateTime( time: "2001-01-01");
    $this->editRequest->date = $editedDate;

    $this->invoicesRepository->method( constraint: "generateNextInvoiceNumberForDate")
                             ->with( ...arguments: $editedDate)
                             ->willReturn( value: 35);

    //When
    $editedInvoice = $this->sut->editRequestedInvoice( request: $this->editRequest);

    //Then
    $this->assertEquals( expected: 35,  actual: $editedInvoice->getNumber());
}
```

Figura 8.4: Exemple de nomeclatura dels tests

Per últim cal destacar que, com també es comentava en la secció de Mètodes de validació, s'ha estat desenvolupant utilitzant *Git* de forma que ha permès crear *Pull Requests* de cada funcionalitat implementada. I s'ha configurat el servidor del repositori central (GitHub) perquè utilitzi el servei de *TravisCI* per a executar tota la suite de tests en cada *Pull Request*. Això dóna encara més seguretat que a l'integrar les modificacions necessàries per a implementar una nova funcionalitat aquesta no té cap error, i a la vegada també podem estar segurs que cap de les funcionalitats ja existents es veu afectada.

8.3 Millores en l'arquitectura

La primera gran millora fa referència als objectes que depenent de les Entities. En el model teòric, els UseCases i els Entities Gateways eren els únics que depenien de les Entites. Més en concret, tota la capa de presentació (Controllers i Presenters) no tenien cap coneixement de les Entities. En el cas de la implementació mostrada, es pot veure com certs elements de la capa de presentació (Controllers i Parsers) sí que depenent de les Entities. En l'arquitectura teòrica, per evitar això es fan servir sempre uns *Request i Response Models* entre la capa de presentació i els UseCases. Al ser, inicialment, les Entities del sistema força simple, el fet d'utilitzar aquests models pel flux de dades entre presentació i UseCases comportava crear moltes classes idèntiques i per tant molta repetició de codi. Com que en cap moment estem trencant la *llei de dependències* vam valorar que fins que no fos necessari ho deixaríem tal com està per mantenir la simplicitat. Però en veritat, potser ja ha arribat el moment de fer aquest canvi, ja que les Entites estan començant a créixer força.

L'última possible millora important també és entre la capa de presentació i els UseCases. Bàsicament es pot veure que els Controllers depenen directament dels UseCases, en comptes d'utilitzat els Boundaries. Això també s'ha fet així per simplificar inicialment l'arquitectura, ja que els UseCases inicialment eren molt simples (com a entrada tenien pocs paràmetres i com a sortida retornaven Entities, o col·leccions d'aquestes, força simples). Igual que anteriorment, potser ja ha arribat el moment de fer aquest canvi.

8.4 Tecnologies usades

Avui en dia hi ha molts llenguatges de programació per a programar serveis web. Des de Java fins Swift, i passant per molts altres.

En aquest projecte però s'ha triat fer ús de PHP, ja que és un dels llenguatges més estesos en el món web i fa molts anys que existeix. Tot i que és un llenguatge de scripts (Scripting language[3]), i en els seus inicis no tenia funcionalitats per fer programació orientada a objectes, a poc a poc i sobretot

[3]Scripting language: `https://en.wikipedia.org/wiki/Scripting_language`

amb les últimes versions s'han anat afegint moltes funcionalitats que permeten la programació orientada a objectes. S'utilitza el framework Silex[4] per agilitzar algunes parts del codi de PHP. Silex és un micro-framework basat en un altre framework més gran que és Symfony[5], per tant també s'utilitzen altres components de Symfony en algunes parts del projecte, com ara el component de Twig[6] com a motor de templates per a PHP. Per tal de gestionar totes aquestes dependències que té el codi PHP del projecte, s'utilitza el gestor Composer[7], que facilita la importació de les llibreries/frameworks necessàries.

Per a la part de presentació, en ser una web, es fa servir HTML, CSS i Javascript. Per tal de millorar el codi CSS i fer-lo més llegible i mantenible es fa ús del framework SASS[8]. El framework ReactJS[9] es fa servir per a crear les interficies més complexes i que es vol que tinguin més dinamisme. Com en la part de PHP, per a gestionar totes les dependències d'aquesta part, es farà ús del gestor de dependències Yarn[10]. També s'empra Brunch[11] com a build tool, el qual bàsicament agafa tot el codi CSS i Javascript (propi i de dependències), el prepara, ajunta i minimitza per tal de reduir la mida final dels fitxers.

Per a poder desenvolupar tot això es fa servir Vagrant[12] per tal de poder crear i compartir una màquina virtual amb tot el necessari configurat per a fer-la servir d'entorn de desenvolupament.

Per a desplegar l'app, tant en un entorn de testeig per a usuaris (stage) com per a producció es fa servir un servidor on hi ha el servei de Dokku[13]. Bàsicament Dokku ens permet desplegar la versió que vulguem en l'entorn que vulguem fent un simple *push* de git al servidor.

[4]Silex: `https://silex.symfony.com/`
[5]Symfony: `https://symfony.com/`
[6]Twig: `https://twig.symfony.com/`
[7]Composer: `https://getcomposer.org/`
[8]SASS: `http://sass-lang.com/`
[9]ReactJS: `https://facebook.github.io/react/`
[10]Yarn: `https://yarnpkg.com`
[11]Brunch: `http://brunch.io/`
[12]Vagrant: `https://www.vagrantup.com/`
[13]Dokku: `http://dokku.viewdocs.io/dokku/`

9. Recursos

Podem dividir els recursos necessaris per a realitzar aquest projecte en tres categories; humans, materials i de software.

9.1 Recursos humans

En aquest projecte principalment i participaran 4 persones:

- **Desenvolupador principal:** aquest seré jo, amb una dedicació setmanal de 25 hores.
- **Desenvolupador de suport:** aquest serà l'Asier, amb una dedicació d'unes 4-5 hores setmanals. La seva dedicació podrà variar molt durant el projecte depenent de factors com ara altres projectes de l'empresa (que restaran hores a aquest projecte) o la necessitat de fer més hores per desviacions en la planificació.
- **Comissió Intranet:** formada per dos socis de l'Apòstrof (Gemma i Martí) que desenvoluparan el rol de *ProductOwner* del projecte. La seva dedicació serà d'entre 1 i 2 hores a la setmana per a fer les reunions necessàries de seguiment i planificació.

9.2 Recursos materials

- **Llocs de treball:** on es realitzarà el projecte, principalment a l'Apòstrof. Això inclou taules, cadires, sales de reunions, electricitat, etc.
- **Equips informàtics per al desenvolupament:** els ordinadors que farem servir l'Asier i jo per a desenvolupar el projecte, i el meu personal per a la realització de la memòria del projecte.
- **Servidor de producció:** on es penjarà la intranet un cop acabada per al seu ús.

9.3 Recursos de software

- **Entorn de desenvolupament:** editor o IDE que cada desenvolupador farà servir. Generalment seran els editors *Atom*, *Sublime Text* o *PHPStorm* .
- **Vagrant:** per a crear un entorn de treball virtual i distribuït i que tots els desenvolupadors treballin amb les mateixes condicions.
- **Composer:** gestor de dependències de PHP.
- **PHPUnit:** llibreria per a realitzar els tests unitaris per a la verificació del codi font.
- **Codeception:** llibreria per a realitzar els tests d'integració per a la verificació del sistema.
- **Git:** sistema de control de versions de codi, per a gestionar el codi font del projecte.
- **Github:** servidor online de repositoris de Git.
- **Trello:** per a la gestió de projecte i les iteracions.
- **Google Drive:** per emmagatzemar tota la documentació del projecte i poder compartir documents.

10. Planificació

S'ha de tenir en compte que en aquest projecte s'utilitzarà una metodologia molt semblant al *Scrum*. Per tant la planificació es farà creant un llistat d'històries d'usuari que seran el màxim d'independents entre elles. Això es fa per poder modificar-ne l'ordre en qualsevol moment si el *Productowner* ho decideix i també poder-les realitzar en paral·lel.

Per tant aquesta planificació temporal inicial pot quedar modificada si el *ProductOwner* decideix en algun moment modificar els ordres aquí descrits. Tot i així, el resultat final hauria de ser el mateix.

També s'ha de tenir en compte que en utilitzar la metodologia de treball de TDD, cada història d'usuari ja porta incorporat el temps necessari per a realitzar els tests associats. I que totes les tasques associades a GEP i a la creació de la memòria del projecte es realitzaran fora de les meves 25 hores setmanals de feina a l'Apòstrof.

10.1 Calendari

El projecte té una durada de 7 mesos, començant el dia 1 de febrer i amb data límit del 1 de Setembre. Tot i que la defensa del projecte serà cap a l'Octubre, s'ha decidit posar la data límit a inicis de Setembre per tenir un marge per a possibles desviacions i poder preparar la defensa del projecte.

10.2 Planificació inicial

En tot projecte inicialment s'ha de fer una primera planificació on cal analitzar els requisits d'aquests, els seus objectius, i fer unes estimacions de temps i costos. Gran part d'això es realitzarà a GEP.

A part, amb la metodologia del *Scrum* també s'ha de definir un *Backlog* inicial. Aquest *Backlog* bàsicament és un llistat, ordenada segons prioritat, amb totes les històries d'usuari necessàries per al projecte. També, a aquestes històries d'usuari, se'ls hi assigna una puntuació que serveix per estimar

l'esforç per implementar-les. Aquesta part sí que es realitzarà dins les 25 hores setmanals de feina a l'Apòstrof, ja que es farà conjuntament amb l'Asier, i es revisarà amb la Comissió Intranet.

10.3 Iteracions del projecte

Les iteracions seran de dues setmanes, i en iniciar-les es farà una reunió amb la Comissió Intranet per tal de decidir quines històries d'usuari s'implementaran. En finalitzar les iteracions també es farà una reunió amb la Comissió Intranet per mostrar la feina feta i rebre feedback.

10.3.1 Iteració 0

En aquest projecte s'ha de fer una implementació d'una arquitectura neta amb PHP, i aquesta no és la típica arquitectura que s'ha vist durant la carrera. Per això en aquest projecte farem una iteració 0 (més llarga del normal) on es desenvoluparà una història d'usuari bàsica (login d'un usuari) per poder estudiar com dur a terme la implementació de l'arquitectura. Evidentment en les següents iteracions s'haurà d'anar perfilant l'arquitectura, però en aquesta primera iteració i dedicaré més temps.

10.3.2 Iteració 1

En aquesta iteració es crearan totes les funcionalitats relacionades amb la gestió bàsica de les feines. Donar d'alta, editar, esborrar i llistar. Ens referim a gestió bàsica, ja que només es gestionaran les dades bàsiques d'una feina, tota la part d'hores, despeses, factures es farà més endavant.

10.3.3 Iteració 2

En aquesta segona iteració s'afegirà el concepte de partides dins les feines. Es modificarà la fitxa de feina per poder llistar, editar, crear i esborrar partides d'una feina.

10.3.4 Iteració 3

Tot seguit s'implementaran les funcionalitats per gestionar els còmputs d'hores dedicades a cada feina (llistat, inserció i esborrar). En concret es crearà un llistat d'hores dedicades per dia i treballador amb un formulari per poder introduir més hores al sistema.

10.3.5 Iteració 4

En la quarta iteració s'afegiran les despeses a la fitxa de les feines. En concret es mostrarà un llistat de les despeses que una feina pugui tenir i un formulari per crear-ne de noves. També es podran editar i esborrar les ja existents.

10.3.6 Iteració 5

Tot i que els usuaris ja existeixen en el sistema, en aquesta iteració es crearan les funcionalitats per a poder-los gestionar; llistat d'usuaris, formulari d'alta i edició, i funcionalitat per esborrar un usuari.

10.3.7 Iteració 6

De la mateixa manera que amb els usuaris, tot i ja existir el concepte de treballador en el sistema, es crearan totes les funcionalitats necessàries per gestionar-los; llistat de treballadors, formularis d'alta i edició i acció de esborrar.

10.3.8 Iteració 7

En aquesta setena iteració continuarem afegint funcionalitats per a gestionar "entitats" del sistema. Per tant es crearan totes les funcionalitats necessàries per a gestionar les àrees de l'empresa; llistat, formularis d'alta i edició i eliminació.

10.3.9 Iteració 8

En aquest cas es crearan totes les funcionalitats necessàries per a gestionar els tipus de despeses; llistat, formularis d'alta i edició i eliminació.

10.3.10 Iteració 9

En aquesta iteració es crearan les funcionalitats referents a la gestió dels clients; llistat, formularis d'alta i edició i eliminació.

10.3.11 Iteració 10

En la iteració número 10 es crearan les funcionalitats de gestió de l'última "entitat" que ja existia al sistema, però que encara no es podien gestionar. Aquesta seran els proveïdors. Per tant es crearan les funcionalitats referents a la gestió dels proveïdors; llistat, formularis d'alta i edició i eliminació.

10.3.12 Iteració 11

Un cop acabat amb les funcionalitats de gestió d'"entitats" en les iteracions anteriors, en aquesta iteració començarem a desenvolupar funcionalitats referents a la facturació. En concret s'afegirà un llistat de factures a la fitxa de les feines i es crearà un formulari per d'*alta ràpida* que crearà una nova factura en estat *demanada*. També es crearà la fitxa de factura des d'on es podran editar tots els paràmetres de les factures.

10.3.13 Iteració 12

En aquesta iteració realitzarem la resta de funcionalitats referents a la facturació. En concret es crearà una vista d'impressió de les factures, i s'implementaran les funcionalitats de rectificar i esborrar una factura. També es modificarà la fitxa d'usuari per poder gestionar quin treballador té associat cada usuari.

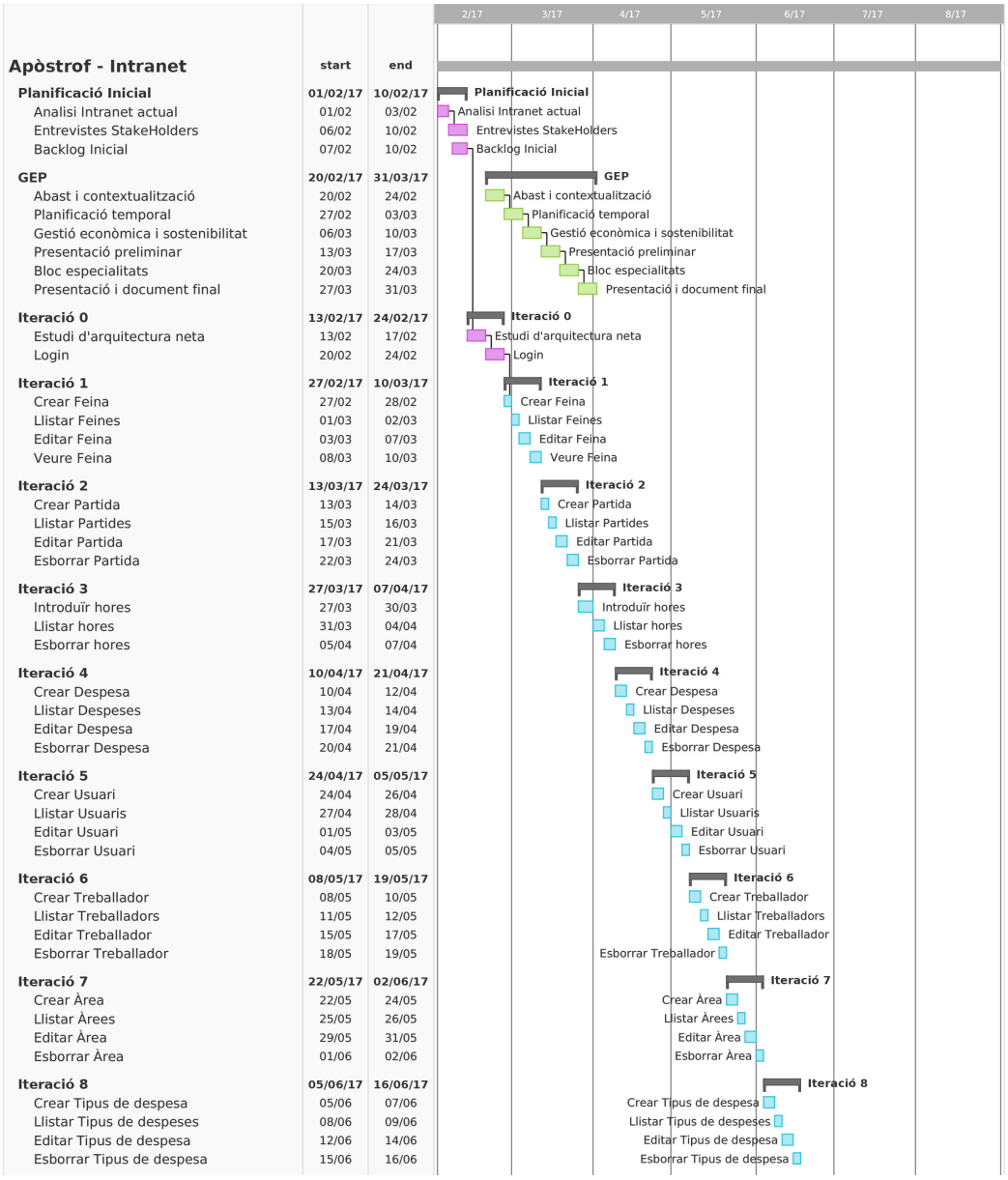
10.3.14 Iteració 13

Per acabar es crearà la funcionalitat per poder esborrar feines i també s'implementa el logout.

10.4 Finalització

En finalitzar totes les iteracions s'haurà de posar la intranet en marxa per al seu ús. També es realitzarà la memòria i la documentació necessària.

10.5 Gantt



Apòstrof - Intranet	start	end
Planificació Inicial	**01/02/17**	**10/02/17**
Analisi Intranet actual	01/02	03/02
Entrevistes StakeHolders	06/02	10/02
Backlog Inicial	07/02	10/02
GEP	**20/02/17**	**31/03/17**
Abast i contextualització	20/02	24/02
Planificació temporal	27/02	03/03
Gestió econòmica i sostenibilitat	06/03	10/03
Presentació preliminar	13/03	17/03
Bloc especialitats	20/03	24/03
Presentació i document final	27/03	31/03
Iteració 0	**13/02/17**	**24/02/17**
Estudi d'arquitectura neta	13/02	17/02
Login	20/02	24/02
Iteració 1	**27/02/17**	**10/03/17**
Crear Feina	27/02	28/02
Llistar Feines	01/03	02/03
Editar Feina	03/03	07/03
Veure Feina	08/03	10/03
Iteració 2	**13/03/17**	**24/03/17**
Crear Partida	13/03	14/03
Llistar Partides	15/03	16/03
Editar Partida	17/03	21/03
Esborrar Partida	22/03	24/03
Iteració 3	**27/03/17**	**07/04/17**
Introduïr hores	27/03	30/03
Llistar hores	31/03	04/04
Esborrar hores	05/04	07/04
Iteració 4	**10/04/17**	**21/04/17**
Crear Despesa	10/04	12/04
Llistar Despeses	13/04	14/04
Editar Despesa	17/04	19/04
Esborrar Despesa	20/04	21/04
Iteració 5	**24/04/17**	**05/05/17**
Crear Usuari	24/04	26/04
Llistar Usuaris	27/04	28/04
Editar Usuari	01/05	03/05
Esborrar Usuari	04/05	05/05
Iteració 6	**08/05/17**	**19/05/17**
Crear Treballador	08/05	10/05
Llistar Treballadors	11/05	12/05
Editar Treballador	15/05	17/05
Esborrar Treballador	18/05	19/05
Iteració 7	**22/05/17**	**02/06/17**
Crear Àrea	22/05	24/05
Llistar Àrees	25/05	26/05
Editar Àrea	29/05	31/05
Esborrar Àrea	01/06	02/06
Iteració 8	**05/06/17**	**16/06/17**
Crear Tipus de despesa	05/06	07/06
Llistar Tipus de despeses	08/06	09/06
Editar Tipus de despesa	12/06	14/06
Esborrar Tipus de despesa	15/06	16/06

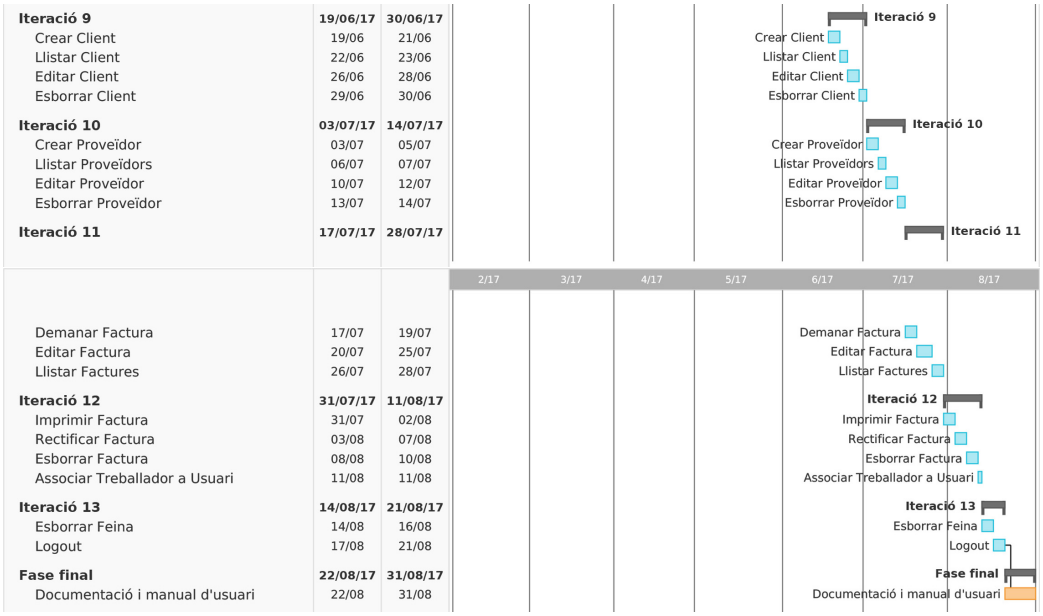


Figura 10.1: Diagrama de Gantt

11. Alternatives i pla d'acció

És molt probable que com a tot projecte apareguin factors que no permetin portar a terme la planificació prèviament explicada. Generalment aquests factors es deuen a una mala estimació dels esforços necessaris per a implementar les diferents funcionalitats del sistema. També poden aparèixer imprevistos que facin endarrerir el projecte per no poder dedicar les hores previstes.

11.1 Mala planificació

En el cas que es necessitin més hores de les planificades s'intentarà fer hores extres per a poder recuperar aquest temps. Però si igualment no és suficient, o no es poden fer aquestes hores extres, aleshores s'haurà de tornar a prioritzar el *Backlog* amb una reunió amb la *Comissió Intranet* per mirar de descartar tasques que es puguin dur a terme més endavant i quedar-se amb les realment importants i imprescindibles.

11.2 Imprevistos

També pot ser que a l'Apòstrof sorgeixi la necessitat que jo dediqui més hores de les previstes a altres projectes. En aquest cas s'intentarà fer el mateix que en l'apartat anterior. Primer s'intentarà suplir aquesta mancança d'hores fent-ne d'extres, i en el cas que no es pugui fer es reprioritzarà el *Backlog* per mirar de descartar funcionalitats que no siguin trivials i que es puguin fer més endavant.

12. Desviacions

Com s'ha dit, a l'inici del projecte es van pensar tantes històries d'usuari com es va poder, es van prioritzar i es va fer una estimació de quan es tardaria a implementar-les. Després es van repartir en iteracions per mirar de preveure quan s'acabaria el projecte. Al cap d'uns mesos de dur a terme el projecte hem tingut unes quantes desviacions respecta la planificació inicial.

Aquestes desviacions han estat principalment pels següents motius:

- **Mala planificació**
  No es va preveure bé el temps que ha estat necessari per a implementar l'arquitectura neta. A causa de la poca documentació que hi ha d'aquesta arquitectura, i sobretot en PHP, hem tardat més del previst a trobar una forma amb la qual ens sentíssim a gust per a programar totes les parts del sistema.

- **Modificacions de recursos**
  Durant la durada del projecte, no s'han pogut dedicar totes les hores que s'havien previst, ja que s'han hagut de dur a terme altres projectes de l'empresa. En concret l'Asier (un dels dos desenvolupadors) quasi no ha pogut fer res de la Intranet, ni tampoc revisar codi fet per mi(*Code Review*). Fins a arribar al punt en què l'Asier ha deixat l'empresa i per tant ara mateix només hi ha un desenvolupador per a finalitzar el projecte.

- **Canvis en les tecnologies**
  Inicialment no es va preveure que faríem servir el framework ReactJS, però al cap d'un mes de feina vam valorar de fer servir aquesta eina per a millorar les interfícies d'usuaris més complexes. Després de veure el que vam tardar a familiaritzar-nos amb ReactJS i poder implementar alguna çosa final", és evident que vam cometre un error en el moment de decidir introduir aquesta tecnologia, ja que ens ha fet endarrerir molt i potser podríem haver fet sevir alguna alternativa més senzilla.

12.1 Modificacions

La principal conseqüència d'aquestes desviacions ha estat la necessitat de fer hores extres per part meva per mirar d'implementar el màxim de funcionalitats possibles per a la finalització d'aquest projecte.

Per altra banda, i tal com s'havia previst, en cas de tenir problemes per qualsevol motiu per arribar a temps a implementar totes les funcionalitats de la Intranet, s'ha anat tornant a prioritzar el *Backlog* amb la *Comissió Intranet* per mirar de descartar tasques que es puguin dur a terme més endavant i quedar-se amb les realment importants i imprescindibles.

Per tant s'han deixat per més endavant funcionalitats poc importants com poden ser finalitzar sessió i l'esborrat d'algunes entitats que a la pràctica quasi mai s'esborren (clients, proveïdors, feines, etc).

12.1.1 Mètodes de validació

Com es va dir inicialment, s'ha fet servir la metodologia de TDD a l'hora de programar, per tal d'assegurar que tot està correctament testejat amb testos unitaris i d'integració. També es va dir que entre els dos desenvolupadors del projecte es farien *Code Reviews*, però com que un dels desenvolupadors quasi no ha pogut participar en el projecte, aquestes no s'han fet del tot. Com a alternativa, s'han continuat creant *Pull Request* per a integrar els canvis de cada història d'usuari implementada, però només s'ha revisat el codi per part del mateix desenvolupador que ho creava. També s'ha posat en marxa un servidor d'integració continua per tal d'assegurar una mica més que cap *Pull Request* acceptada integrarà errors en el sistema.

13. Pressupost

13.1 Identificació dels costos

Com en la majoria de projectes d'informàtica, aquest projecte compta amb 3 tipus de costos; recursos humans, hardware i software.

Tot i així, com que tot el software necessari pel desenvolupament d'aquest projecte és gratuït, no caldrà fer el càlcul de costos d'aquest apartat.

13.2 Estimació dels costos

13.2.1 Recursos humans

Com s'ha especificat anteriorment, aquest projecte serà desenvolupat principalment per una sola persona (jo, Ferran Martin), tot i que comptarà amb l'ajuda d'un segon desenvolupador de forma puntual (l'Asier Illarramendi). També i participaran les persones que composen la Comissió Intranet (*ProductOwner*). Aquests seran la Gemma Casamajó (que també és la directora del projecte) i en Martí Lázaro.

Degut a que aquest projecte es desenvolupa en una empresa, per poder fer els càlculs dels costos dels recursos humans, tot seguit farem un resum de les hores treballades i quin preu tenen.

Per tal de poder fer una bona estimació dels costos, s'ha dividit el projecte en 4 fases; planificació inicial, iteració 0, conjunt d'iteracions i fase final.

En la fase de **planificació inicial** hi participem l'Asier i jo, i té una durada de 8 dies. En aquesta fase jo hi treballo 5 hores al dia, i l'Asier col·labora en l'última part amb un total de 8 hores.

Pressupsot Planificació Inicial	**Hores**	**€/hora**	**Preu total**
Ferran	40	32	1.280€
Asier	8	32	256€
Total			**1.536€**

Taula 13.1: Pressupost de la Planificació Inicial

A partir d'aquesta fase i en endavant, la jornada que jo dedicaré a la intranet serà de 20h setmanals. De la meteixa maner, l'Asier dedicarà 5 hores setmanals.

La **iteració 0** s'ha comptabilitzat a part, ja que només hi treballaré jo, i no hi ha hores de coordinació.

Pressupsot Iteració 0	**Hores**	**€/hora**	**Preu total**
Ferran	40	32	1.280€

Taula 13.2: Pressupost de la Iteració 0

Un cop feta la interació 0 s'inicia la fase del projecte formada pel **conjunt d'iteracions**, amb un total de 13. A cada iteració es contemplen hores de coordinació i de desenvolupament.

Dins les hores de coordinació hi participen el Marti, la Gemma i jo. En total seran 2 hores per iteració.

A les hores de desenvolupament hi participem l'Asier i jo, i en total són 48 hores per iteració.

Pressupsot Conjunt d'iteracions	**Hores coord.**	**Hores dev.**	**€/hora**	**Preu total**
Ferran	2	38	32	1.280€
Asier	0	10	32	320€
Martí	2	0	32	64€
Gemma	2	0	32	64€
Total				**1.728€**

Taula 13.3: Pressupost del Conjunt d'iteracions

Per tant, el **cost total del conjunt d'iteracións** serà de 1.728€ * 13 iteracions = **22.464€**.

A la **fase final** només hi participaré jo, i hi dedicaré 4 dies. En aquesta última fase tornaré a dedicar 5 hores al dia.

Pressupsot Fase final	**Hores**	**€/hora**	**Preu total**
Ferran	20	32	640€

Taula 13.4: Pressupost de la Fase final

Com es pot veure, les hores totals s'han basat en les estimacions que s'han fet en l'apartat de planificació i s'han inclòs les hores de reunions i seguiment per part de la Comissió Intranet. Per tant el total de tot queda de a següent forma:

Pressupsot Recursos humans	**Preu**
Planificació inicial	1.536€
Iteració 0	1.280€
Conjunt d'iteracions	22.464€
Fase final	640€
Total	**25.920€**

Taula 13.5: Pressupost dels Recursos humans

13.2.2 Hardware

El hardware necessari per al desenvolupament del projecte constarà dels ordinadors de cada desenvolupador i del servidor de producció on es publicarà la intranet. Però, cal tenir en compte que, com que aquest servidor ja s'està utilitzant per altres tasques i projectes, no el tindrem en compte en els següents càlculs de costos.

Pressupost Hardware	**Euros**	**Temps útil**	**Amortització**
iMac de sobretaula 21'	1279€	5 anys	255,80€
Macbook pro 13'	1699€	5 anys	339,80€
Total			**595,60€**

Taula 13.6: Pressupost del Hardware

13.3 Control de gestió

Amb tot el calculat prèviament tenim un pressupost per un projecte el qual surt bé al 100%. Com que això és molt difícil que passi, tot seguit es farà un càlcul de possibles imprevistos i contingències per ajustar el pressupost.

El principal imprevist és el del temps; que es necessitin més hores de les estimades per a assolir els objectius del projecte. Com s'ha explicat anteriorment, en el cas que això pases, primer s'intentarà millorar la priorització de les tasques per tal de mirar de descartar les que no es considerin necessaries per que només quedin les realment necessaries i impresindibles. Però en el cas que tot i així es necessitesin més hores es farien hores extres per tal de compensar-ho. Es calcula que aquestes **hores extra** podrien ser un 10% respecte les hores dedicades a desenvolupament durant la fase de *conjunt d'iteracions*. Per tant, això faria augmentar el pressupost en (38 + 10)hores * 13 iteracions * 0.1% * 32€/h = **1.996,8€**

Per últim, per tal d'estar segurs que tot queda cobert en el pressupost, s'aplicarà un percentatge respecte el total en concepte de **contingències**, que serà del **5%**.

13.4 Pressupost total

Un cop calculats tots els apartats, el pressupost total queda de la següent forma:

Pressupost total	**Cost(€)**
Recursos humans	25.920
Hardware	595,60
Imprevistos	1.996,8
Contingències	1.425.62 *(5%)*
Total del projecte	**29.938,02€**

Taula 13.7: Pressupost Total

14. Sostenibilitat

En aquest apartat analitzarem la sostenibilitat d'aquest projecte, ho farem en les dimensions econòmica, ambiental i social. S'ha de tenir en compte que només analitzarem la part de PPP de la matriu de sostenibilitat, ja que, com que ja s'ha explicat, l'abast d'aquest projecte només arriba fins a aquest apartat.

14.1 Dimensió econòmica

Durant la planificació del projecte s'han tingut en compte els costos per als recursos humans, de hardware i de software. Com em vist anteriorment, els recursos humans seran els mínim per a poder realitzar el projecte, el mateix que per als recursos de hardware. En quan als recursos de software, ja que tot el que es fa servir es gratuit o open source, el cost és 0.

En ser un projecte intern de renovació de la intranet actual de l'empresa, aquest no generarà una rendibilitat econòmica directa per a l'empresa. Però el fet de millorar el software actual sí que aporta un valor a l'empresa de forma indirecta. També, com s'ha esmentat anteriorment, sí que es valora poder oferir aquesta intranet com a servei a altres cooperatives, però això queda fora de l'abast d'aquest projecte.

Per tant, la valoració per la part econòmica seria de 8 sobre 10, ja que s'han contemplat força bé els riscos i el projecte és viable.

14.2 Dimensió social

Aquest projecte té un impacte directe només sobre la pròpia empresa, ja que permetrà a l'empresa millorar els seus procesos i analitzar les dades per tal de millorar la rendibilitat, productivitat, etc. També es vol comentar que tot i que l'Apòstrof és l'únic que rep un benefici directe, els seus clients també en reben de forma indirecta, i al ser l'Apòstrof una cooperativa la qual participa

molt en projectes del tercer secor de caire social, creiem que aquest és un aspecte important.

Com s'ha dit abans, una idea de futur és la de oferir aquesta intranet a altres cooperatives, i tot i queaixò queda fora de l'abast d'aquest projecte, és important comentar-ho.

Amb tot això, la puntuació per aquesta part seria d'un 7 sobre 10, ja que tot i que al ser desenvolupat en una cooperativa, i que per tant a nivell social té potencial per tenir força impacte, hi ha funcionalitats importants que no entren dins l'abast d'aquest projecte.

14.3 Dimensió ambiental

Com en la majoria de projectes de software, la dimensió social es veu afectada majoritàriament per temes de consum energètic per als punts de treball i servidors (hardware, llum, calefacció, etc). A part, en aquest projecte la utilització de material d'oficina (papers, bolígrafs, etc) serà mínima. Per tant, la puntuació que té el projecte en aquesta dimensió serà de 7 sobre 10.

Amb tot això, la **taula de sostenibilitat** per aquest projecte queda de la següent manera:

Sostenibilitat	**PPP**
Econòmic	8/10
Ambiental	7/10
Social	7/10

Taula 14.1: Taula de sostenibilitat

15. Conclusions

L'objectiu principal d'aquest projecte era el de crear una primera versió d'un software de gestió de feines que permetés a l'empresa l'Apòstrof SCCL deixar de fer servir la intranet actual. Podem dir que aquest objectiu s'ha aconseguit satisfactòriament, tot i les desviacions i modificacions respecte al pla inicial. S'ha pogut arribar a crear una primera versió funcional, útil i amb valor per a l'Apòstrof, amb la qual es continurà treballant per ampliar-la i afegir funcionalitats no tan bàsiques que per ara s'han deixat de banda.

En l'aspecte més tècnic de l'arquitectura de software usada, aquest projecte també tenia una motivació personal per veure com funcionava aquesta arquitectura en un context d'una aplicació web en PHP. Un cop finalitzada aquesta primera versió del sistema puc dir que estic molt content amb els resultats obtinguts. L'arquitectura ha permés crear un sistema robust de forma sencilla i a la vegada fàcilment mantenible i extensible. També he de dir que l'ús d'aquesta arquitectura juntament amb practiques de *Clean Code* fan que constantment estiguis refactoritzant i reorganitzant parts del codi, per tant és indispensable disposar d'una eina que faciliti això. Per això al poc de començar el projecte vam canviar d'IDE i vam començara fer servir *PHPStorm* (amb una llicència gratuita) que aporta funcionalitats que faciliten aquestes tasques (sobretot l'autocompletar i el renombrar).

Aquest projecte també m'ha permés veure i provar el framework de Javascript ReactJs, que actualment té molta demanda en el sector. El fet d'adoptar l'ús d'aquest framework és un dels principals motius de les desviacions, i per tant possiblement no va ser la desició més encertada. No vaig saber valorar bé la corva d'aprenentatge necessaria per arribar a implementar components de ReactJs de la mida necessaria per la Intranet, i vist en retroespectiva inclús crec que aquest tema és prou gran per un altre TFG. Tot i això, estic content d'haber aprés a usar aquest framework, ja que, entre altres coses, també m'ha permés experimentar amb altres tipus d'arquitectures.

15.1 Treball futur

Com s'ha dit sempre, en aquest projecte hem desenvolupat una primera versió d'una intranet, per tant encara queda molta feina per arribar a tenir un producte final.

La planificació que fem d'aquest projecte d'ara en endavant passa principalment per tres etapes generals.

Una primera on s'acabaran d'implementar un seguit de funcionalitats força importants, com ara la gestió dels calendaris laborals dels treballadors i les seves bases horaries, i la generació automàtica d'informes que ajudaran molt a la presa de decision dins l'empresa.

La segona etapa servirà per a implementar una capa *multicoopertiva* que permetra donar accés a la Intranet com a servei a altres cooperatives. En aquests moments, el model de negoci respecte a això no està del tot clar, ja que es vol buscar un model basat en l'economia solidaria. Però en tot cas un cop s'arrivi a aquest punt, aconseguir un retorn de la inversió feta en aquest software serà més fàcil.

I per últim, durant la tercera etapa es destinaran els recursos a anar afegint funcionalitats menys importants, però que a poc a poc aniran afegint més valor al producte final. Aquests funcionalitats poden ser un sistema d'avisos interns, sistemes de comentaris per les feines, sistemes de gestió de projectes, etc. En veritat si es va mantenint el sistema, aquesta etapa mai s'hauria d'acabar, ja que sempre serà necessari adaptar el software a les noves necesitats.

Índex de figures

Índex de taules