UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Migració d'un programari ERP a tecnologies web Open-Source

Treball Fi de Grau

Grau en Enginyeria Informàtica

Autor: Sergio Belda Galbis

Tutor: Vicente Pelechano Ferragud

Cotutor: Antonio Tarín Gómez

2017-2018

Resum

Aquest treball il·lustra el procés de migració o reenginyeria d'un sistema software ERP produït per una empresa de software de gestió empresarial. El producte software ha estat desenvolupat seguint els processos que caracteritzen a l'enginyeria software i que es descriuen en detall i, a més, han estat integrades característiques i ferramentes software pròpies de les metodologies de desenvolupament àgil. El desenvolupament s'ha portat a terme utilitzant tecnologia web O*pen-Source*, la qual dota al producte de múltiples avantatges pròpies del *framework*, respecte al sistema de partida.

Paraules clau: ERP, migració, *web components*, Polymer, Node.JS, agilitat.

Resumen

Este trabajo ilustra el proceso de migración o reingeniería de un sistema software ERP producido por una empresa de software de gestión empresarial. El producto software ha sido desarrollado siguiendo los procesos que caracterizan a la ingeniería software y que se describen con detalle y, además, han sido integradas características y herramientas software propias de las metodologías de desarrollo ágil. El desarrollo ha sido realizado utilizando tecnología web *Open-Source*, la cual dota al producto de múltiples ventajas propias del *framework*, respecto al sistema de partida.

Palabras clave: ERP, migración, *web components*, Polymer, Node.JS, agilidad.

Abstract

This document illustrates the migration or reengineering process of an ERP system produced by a business management software enterprise. The software product has been developed following the processes that represents the software engineering and which have been described accurately. Moreover, some features and software tools of agile development methodologies have been integrated. The development has been accomplished using Open-Source web technology, which gives multiple benefits of the framework, compared to the previous system.

Keywords: ERP, system migration, web components, Polymer, Node.JS, agility.

Taula de continguts

Índex de imatges

Índex de taules

1. Introducció

Un ERP (*Enterprise Resource Planning*), que significa planificació de recursos empresarials, és una ferramenta que ofereix la possibilitat de integrar certes operacions d'una empresa, com ara la producció, la logística, l'inventari, o la comptabilitat.

Aquest document mostra la migració d'un software ERP utilitzant tecnologia web, més específicament, tecnologia *Open-Source* per a produir aplicacions web mitjançant *web components*. A més, exposa les diferents fases de la enginyeria de software per les qual es passa en el desenvolupament d'un producte software així com la metodologia de desenvolupament utilitzada.

1.1. Motivació

El projecte s'ha desenvolupat seguint les diferents etapes de l'enginyeria software. A més, s'han adoptat metodologies de desenvolupament àgil. Aquest tipus de desenvolupament és l'utilitzat en la producció de software actual i ha resultat interessant per aquest projecte per tal d'il·lustrar els coneixements adquirits així com obtindre nous.

La tecnologia utilitzada també resulta atractiva per a aquest projecte, ja que moltes empreses opten actualment per utilitzar tecnologies de desenvolupament web *Open-Source*.

Pel que fa a l'empresa, aquesta es troba interessada en portar a terme un procés de reenginyeria dels seus productes que ofereixen amb la qual cosa puguen adaptar-los a les noves tecnologies, i noves necessitats, ja que cada vegada més empreses es troben interessades per requisits com la portabilitat o disponibilitat de recursos.

En resum, aquest projecte ha servit per tal de conèixer noves tècniques, ferramentes software, i tecnologies de desenvolupament que s'utilitzen actualment i que resulten de gran interès. I per a l'empresa, ha resultat útil per poder portar a terme un nou desenvolupament d'un dels seus sistemes.

1.2. Antecedents

La tecnologia utilitzada en el producte ERP del qual es parteix és Visual Basic, un llenguatge de programació desenvolupat a 1991, l'última versió del qual va ser Visual Basic 6, que va deixar de tenir suport al 2008.

El producte ERP va ser desenvolupat a 1993, amb la qual cosa aquest llenguatge era un dels més utilitzats per tal d'elaborar programari d'escriptori, i fins l'actualitat no ha canviat la tecnologia del mateix.

1.3. Objectiu

Aquest treball s'ha desenvolupat en el context d'una empresa de producció de software empresarial. La empresa ha pres la decisió de migrar el software ERP degut a diverses raons.

Per una banda, podem destacar la tecnologia en la qual estava desenvolupat el sistema de partida. Com s'ha comentat, estava produït amb la tecnologia Visual Basic 6, que va deixar de tenir suport per part de Microsoft fa uns anys, i per tant, deixa al llenguatge sense actualitzacions i suport d'errors. Encara que actualment encara hi ha molts programes implementats amb aquesta plataforma, cada vegada són més les empreses que actualitzen els seus sistemes a tecnologies actuals.

Per altra banda, podem destacar la portabilitat que ofereixen les tecnologies web, ja que les aplicacions poden ser utilitzades en un gran nombre i diferents tipus de dispositius.

Per últim, cal destacar les propietats que aporta la nova tecnologia utilitzada en el nou software, aquesta és el *framework* de Google Polymer 2.0. Aquest ofereix certes avantatges respecte a altres, com ara el desenvolupament de components genèrics reutilitzables. És a dir, aquest entorn de treball permet desenvolupar components que poden ser reutilitzats tant al propi projecte com a altres projectes de l'empresa desenvolupats baix tecnologia web. Tot i això, també aporta una major cohesió i un baix acoblament dels components de l'aplicació.

1.4. Impacte esperat

Pel que fa a les avantatges o millores que suposa el producte resultant d'aquest projecte, en podem destacar que la migració d'un software d'escriptori a tecnologies web ofereix la possibilitat de ser adaptat a més tipus de dispositius. Permet que la interfície puga ser millorada amb la qual cosa augmente la eficiència i facilitat d'ús, així com que l'usuari puga sentir que la interfície resulta més atractiva per a ser utilitzada. Per altra banda, també es pot incrementar la experiència d'usuari gràcies a fer que les tasques d'usuari es realitzen en menys accions, ja que el fet d'adaptar el sistema a diferents dispositius exigeix que es redueixen els passos dels usuaris per realitzar un treball.

1.5. Metodologia

La metodologia de treball en el desenvolupament d'aquest projecte entra dintre de l'àmbit de la enginyeria del software. El tipus de metodologia de treball seguida ha estat la del desenvolupament àgil. Aquest mètode o forma de treballar es propi de l'actualitat on un dels elements més importants del desenvolupament software és l'entrega i desenvolupament ràpids. El software no es genera com una sola unitat sinó que es desenvolupa amb una sèrie d'increments que aporten funcionalitats al sistema i que són lliurades al client. L'aplicació d'aquesta metodologia junt amb els processos propis de la enginyeria del software seran il·lustrats al llarg d'aquest document.

1.6. Estructura

Aquest document té per objectiu mostrar les fases per les quals s'ha passat en el desenvolupament del sistema software. S'inicia amb una documentació de l'anàlisi del sistema i l'àmbit d'actuació, així com una anàlisi de la tecnologia de desenvolupament.

A continuació, es fa una especificació de la enginyeria de requisits portada a terme al projecte, amb una especificació de requisits de negoci i software.

En l'àmbit del disseny es mostrarà la definició del disseny arquitectònic adoptat al nostre sistema i el disseny de prototips de interfície del producte.

Es donarà a conèixer part de la implementació, amb mostres de l'estàndard i components de la tecnologia adoptada en el projecte, així com d'exemples de codi net seguit al desenvolupament.

Després, es mostrarà la metodologia de desenvolupament seguida, exposant aquelles tècniques i ferramentes software que han ajudat a que aquest siga de naturalesa àgil.

Seguidament, es parlarà de la etapa de manteniment del sistema amb exemples de proves, refactoritzacions i manteniment correctiu del sistema.

Per últim, es mostrarà el resultat del sistema desenvolupat, i es portarà a terme una redacció de limitacions en el desenvolupament així com una conclusió del mateix.

2. Estat de l'art

En aquest capítol es portarà a terme l'estudi estratègic o anàlisi de la situació actual de la tecnologia. Al tractar-se d'un treball desenvolupat en l'àmbit d'una pràctica en empresa, es descriurà el context d'aquesta empresa. Es realitzarà una documentació d'altres aplicacions que existeixen actualment al mercat i que posseeixen característiques funcionals semblants o iguals a les que van ser desenvolupades al projecte.

2.1. Context

Aquest projecte s'ha desenvolupat a una empresa de producció de software de gestió empresarial, que desenvolupa ferramentes com ara BPM[1], CRM[2], ERP, SGA[3], etc.

L'empresa en la que s'ha desenvolupat el producte és Ahora Freeware, una empresa valenciana amb 25 anys d'experiència que ha viscut diverses transformacions com ara el pas d'un sistema de negoci estàndard a un model de negoci *Freeware*, el qual ha fet que augmente tant el nombre de clients com l'àrea geogràfica de distribució. Aquest es basa en distribuir de forma gratuïta tot el software que es desenvolupa a l'empresa, i obtenir els ingressos amb la consultoria i la compra de llicències.

Aquesta empresa va desenvolupar a 1993 un software ERP baix tecnologia Visual Basic 6, la qual permetia adoptar aquest sistema en sistemes operatius Windows. Aquest producte es ofert a diferents empreses de diversos àmbits de producció gràcies al canal de distribució del que disposa la empresa i que fa que el software s'adapte al context de treball de cada negoci. És a dir, l'empresa compta amb diversos distribuïdors per a cada tipus d'àmbit empresarial per tal de poder centralitzar el producte a les necessitats específiques.

Aquest software està desenvolupat amb una tecnologia que deixà de tenir suport, i per tant, resulta obsoleta davant les noves característiques que ofereixen les tecnologies de desenvolupament actual, i que necessita d'una renovació per tal

[1] BPM (*Business Process Management*): Ferramenta software que permet a les empreses modelitzar, implementar i executar conjunts d'activitats per optimitzar els processos de negoci.
[2] CRM (*Customer Relationship Management*): Sistema software per a la gestió de les relacions amb els clients, amb la venta i el màrqueting.
[3] SGA (Sistema de Gestión de Almacenes): Sistema software per a la gestió dels recursos i operativa d'un magatzem.

d'adaptar-se a les necessitats i tecnologies actuals. A més, cal destacar com a dada interessant que el llenguatge en el que estava desenvolupat el sistema no disposa de gran popularitat a l'actualitat ja que segons estadístiques de 2018 de pàgines web de gran reputació dintre de la comunitat de desenvolupadors com StackOverflow, donen la valoració del llenguatge més temut o que menys interès suscita per continuar sent utilitzat a Visual Basic 6 amb un 89,9%.

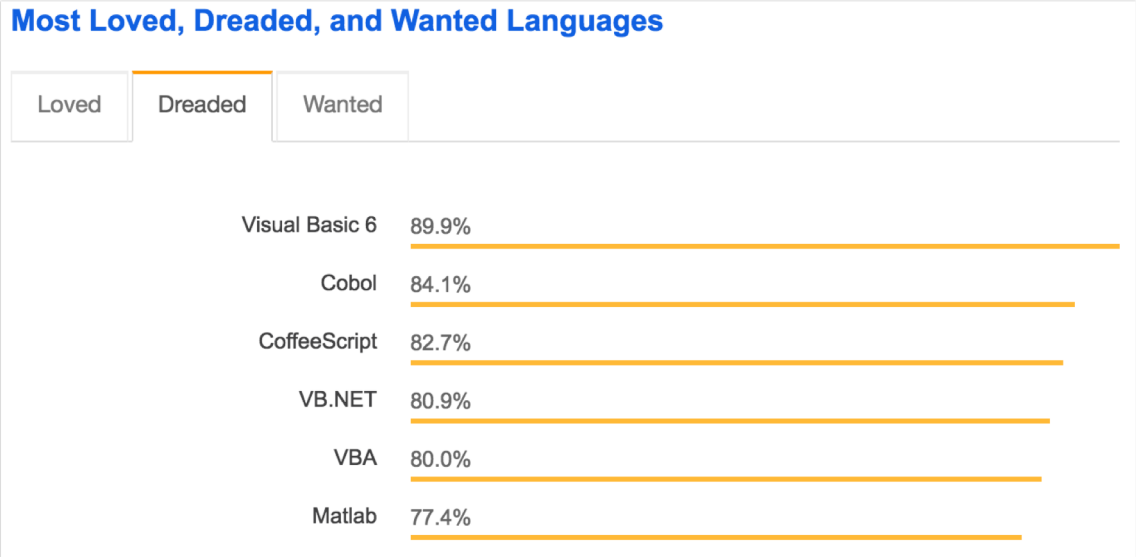

Figura 2.1 Estadística 2018 StackOverflow - Llenguatges més"temuts"

Amb aquest propòsit i necessitat naix aquesta migració i el treball desenvolupat a aquest projecte.

2.2. Anàlisi del mercat

En el moment de realitzar l'anàlisi del producte software que va a ser desenvolupat és adequat observar l'àmbit de utilització del nostre sistema. Per aquest motiu, resulta adequat portar a terme un estudi del mercat que ens proporcione una idea sobre les característiques funcionals que esperen els usuaris que utilitzen aquestes aplicacions al mateix temps ens permeta conèixer el tipus de usuari que va a utilitzar el nostre sistema.

Actualment, un 82% d'empreses a Espanya utilitzen un software de gestió empresarial ERP, segons un estudi de la companyia SoftDoit, un comparador de solucions software empresarials que té l'objectiu de mostrar a les empreses la millor opció. És a dir, un software de gestió empresarial és present en gran part d'empreses a Espanya per la seua importància de cara al control dels recursos empresarials.

Podem destacar diferents exemples de sistemes de recursos empresarials presents al mercat actual i que ens poden mostrar característiques funcionals que

resulten interessant al nostre sistema o que guarden relació amb aquest. Entre les múltiples empreses que es dediquen a la producció de software empresarial tenim Odoo, SAP i Microsoft, amb el seu software Microsoft Dynamics.

En primer lloc, Odoo és una empresa belga que desenvolupa software de gestió empresarial i que es presenta com un competidor de SAP i Microsoft Dynamics. L'objectiu d'aquesta empresa és el d'oferir un conjunt d'aplicacions per a empreses, fàcils d'utilitzar que conformen un conjunt de ferramentes per a qualsevol negoci que les necessite. Es basa en un sistema integrat per mòduls o aplicacions que coexisteixen depenent de l'àmbit d'actuació empresarial i gràcies a aquest fet permet adaptar-se a empreses de diferents mides, des d'una persona a milers d'empleats. Està integrat en més de 300.000 empreses. Una de les dades interessants d'aquest producte és que compten amb una comunitat de desenvolupadors de mòduls per al sistema multitudinària i que el proveeix per tant, de més funcionalitats.

El que podem destacar d'aquest software és la modularitat per tal de personalitzar l'entorn i poder adaptar-lo a les necessitats dels usuaris. Aquest fet proporciona al sistema escalabilitat horitzontal per tal d'adaptar-se a diferents tipus d'empreses.

Figura 2.2 Odoo

En segon lloc, SAP és el líder internacional en software de gestió empresarial. Està present en gran part de les empreses i posseeix instal·lacions a diferents països. SAP ofereix diferents ERP, depenent de la mida de la empresa. Integra als seus serveis *machine learning*, analítiques predictives, i processos optimitzats incorporats.

Presenta un servei en el núvol o centralitzat a nivell d'una empresa, així com un model híbrid. A més l'oferta del ERP al núvol compta amb un servei de seguretat que ofereix la empresa.

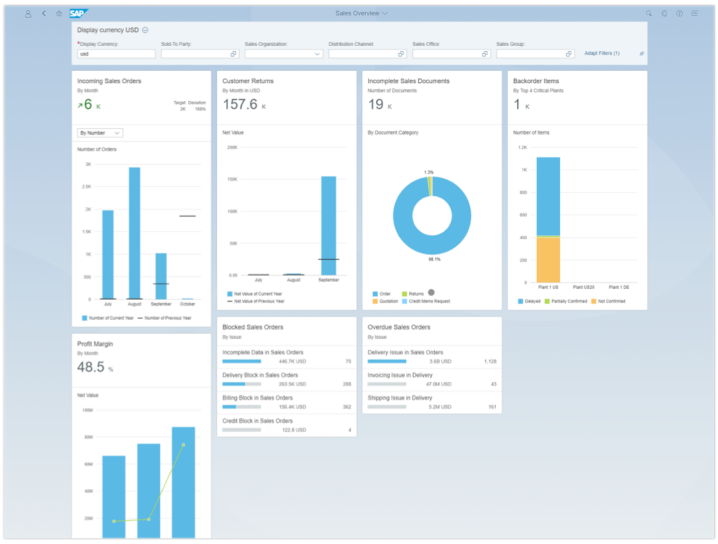

Figura 2.3 SAP

Per últim, Microsoft Dynamics és un sistema *cloud* de ERP i CRM desenvolupat per Microsoft. Al igual que SAP, manté el seu servei al núvol amb la qual cosa descentralitza l'accés a la informació. Aquest sistema es composa per diferents subsistemes, depenent de l'àmbit de treball podem trobar aplicacions per a venta, màrqueting, finances i operacions, etc.

Aquest últim fet resulta interessant per tal de poder ser utilitzat per diferents departaments dintre d'una empresa i com ocorre amb el software de Odoo, manté la idea de dividir el sistema en subsistemes o mòduls.

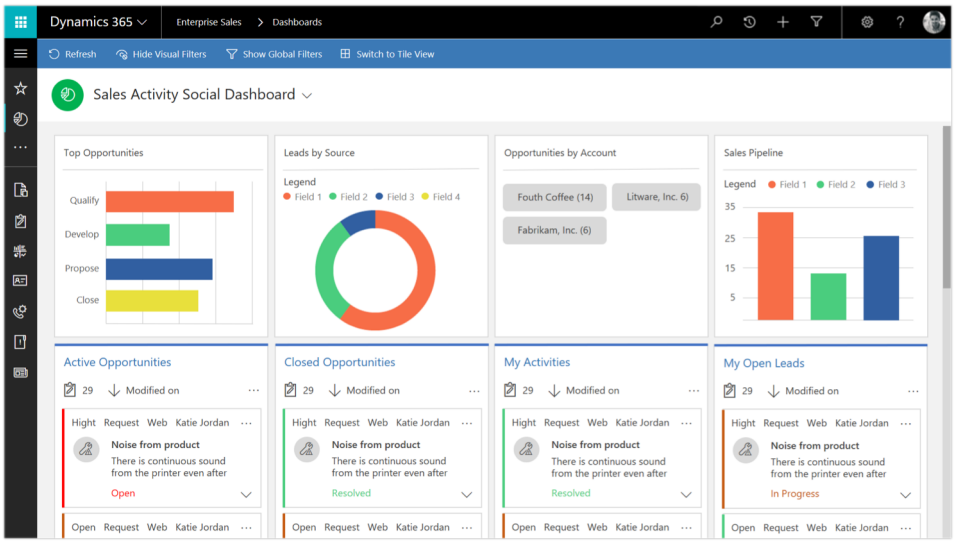

Figura 2.4 Microsoft Dynamics

El que podem extraure d'aquest anàlisi és que la majoria d'aplicacions ERP que podem trobar al mercat basen el seu funcionament en presentar la informació de forma gràfica, així com modularitzar diferents vistes o departaments per tal de tenir la informació de forma heterogènia.

Per últim, integrar tecnologies com *machine learning*, anàlisi predictiu, etc., i traslladar el servei al núvol resulta atractiu de cara a augmentar característiques com la portabilitat o la eficiència d'ús.

3. Anàlisi del problema

En aquest capítol es portarà a terme la descripció del procés d'anàlisi realitzat en aquest projecte. Aquesta acostuma a ser la primera etapa en la producció software i inclou entre altres un estudi de la factibilitat del producte a desenvolupar i de la tecnologia que s'utilitzarà a la nostra solució.

3.1. Tecnologia web

Pel que fa a la tecnologia utilitzada per al desenvolupament del software, es tracta de tecnologia web *Open-Source*. Quan s'ha elegit aquesta tecnologia s'ha fet pensant en les avantatges que té sobre la tecnologia prèvia del producte que es va a migrar. És a dir, s'ha triat una tecnologia que no sols permeta fer una transformació del producte inicial sinó, que a més, s'incremente la qualitat del software gràcies a les propietats d'aquesta. Aquest projecte conforma la migració de la capa de presentació de l'ERP, dotant a l'antic sistema d'una nova interfície desenvolupada amb una nova tecnologia i que quan eventualment es migre l'estructura de la base de dades i s'adapten tots els formularis del sistema antic de Visual Basic 6 s'obtindran totes les avantatges de la tecnologia web. Integrar una nova tecnologia amb una tecnologia antiga s'aconsegueix amb un *framework* que s'especificarà més endavant.

En primer lloc, cal destacar la importància d'elegir la programació web en l'actualitat i més concretament, el llenguatge de programació JavaScript. Una de les avantatges principals és la portabilitat que ofereix i que altres tecnologies no. La portabilitat és una de les característiques que es poden destacar en la qualitat d'un software, dintre del model de qualitat ISO/IEC 25010, i en aquest cas és referent a la capacitat per a poder ser instal·lat en la major heterogeneïtat de dispositius possibles. És a dir, la programació web és utilitzada en l'actualitat per tal de desenvolupar software per a qualsevol aparell: mòbils, ordinadors, televisions intel·ligents, etc. Aquesta característica ajuda al nostre sistema a poder ser adoptat en diferents tipus de dispositius, amb la qual cosa qualsevol client podria obtenir informació de l'ERP en qualsevol aparell. Cal destacar, que des del punt de vista comercial també resulta atractiva la idea de que no calga tenir un dispositiu específic per a reproduir el teu software.

Per altra banda, cal destacar que les tecnologies web redueixen el cost de desenvolupament ja que quan es porta a terme, es fa pensant en la diversitat de dispositius que podran utilitzar el nostre software, és a dir, no es deu adaptar el desenvolupament per cada sistema de forma nativa. Aquest fet, però, té el seus inconvenients, com ara que la experiència d'usuari pot veure's limitada degut a que el fet de desenvolupar per a diferents plataformes causa que no poden ser utilitzats tots el components, i avantatges de hardware i software que ofereixen aquestes plataformes específiques.

Pel que fa al llenguatge de programació, com s'ha citat abans, aquest és JavaScript. JavaScript és un llenguatge lleuger e interpretat, que és utilitzat com a llenguatge de *scripting* per a pàgines web, però també es empleat en molts entorns sense navegadors com ara Node.js. És un llenguatge de *script* multi-paradigma, ja que suporta programació funcional, orientada a objectes e imperativa. L'estàndard de JavaScript és ECMAScript, que és una especificació de llenguatge de *scripting* que forma la base de, entre altres, JavaScript. L'actual especificació d'ECMAScript és la versió 6. Aquesta versió és la que s'ha decidit adoptar en el desenvolupament d'aquest projecte i permet la utilització de iteradors, dades binaries, estructures de dades com *maps* i *sets*, *arrow functions*, etc.

JavaScript és un dels llenguatges més utilitzats en l'actualitat, per la seua versatilitat multiparadigma i per la possibilitat de ser utilitzat en qualsevol desenvolupament software per a distintes plataformes.

Pel que fa a la popularitat del llenguatge, JavaScript, és un dels llenguatges més utilitzats hui a dia a la programació web i segons estadístiques del portal web StackOverflow és un dels llenguatges més valorats per la comunitat amb un 61,9% de popularitat.

Per últim, cal destacar les avantatges que suposa fer ús d'una tecnologia de desenvolupament com JavaScript front a Visual Basic 6. JavaScript és un dels llenguatges de programació més utilitzats actualment, amb una corba d'aprenentatge molt inferior ja que posseeix una sintaxis semblant a llenguatges com C o Java, es pot portar a terme qualsevol aplicació web amb aquest llenguatge, i és utilitzat en la majoria d'entorns de treball web actuals.

3.2. Tecnologia utilitzada

A l'hora de desenvolupar una aplicació software per a plataforma web, hi ha diferents alternatives. A continuació, es presentaran les principals opcions que hi ha actualment que s'han plantejat ser escollides com a tecnologia de desenvolupament i les raons que han portar a elegir una d'aquestes.

Entre alguns dels *frameworks* o entorns de treballs utilitzats avui a dia en la programació web que poden ser empleats al projecte podem destacar: React, Angular JS, Vue JS i Polymer.

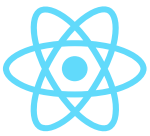

Figura 3.1 React

En primer lloc, React, és un *framework* desenvolupat per Facebook. Es descriu com una llibreria de JavaScript declarativa i flexible per desenvolupar interfícies d'usuari. És un entorn de treball enfocat al desenvolupament de SPA[1]. A més, ajuda a crear aplicacions en un entorn de dades canviant. Aquestes dues característiques podien ser apropiades per a un software de les nostres característiques que normalment és desenvolupa a una sola pàgina i per altra banda, conté centenars de dades que poden canviar en qualsevol moment.

Figura 3.2 Angular JS

En segon lloc, Angular JS, és un *framework* mantingut per Google que basa el seu desenvolupament en un patró de model vista controlador (MVC). Soporta el *data binding*, que és la actualització automàtica de dades sense tenir que tenir en compte la actualització conscient de la vista. Per altra banda, també té com a característica la injecció de dependència que permet reemplaçar components que ja no són utilitzats. Pel que fa al components de l'aplicació, aquests són reutilitzables. Per últim, millora la mantenibilitat ja que és un *framework* que suporta molt bé el *testing*, degut a que separa el comportament de les vistes. Aquesta tecnologia és adequada per a aquest projecte gràcies a la injecció de dependències que permetria la actualització automàtica de les dades, i per altra banda, desenvolupar una aplicació basada en un patró de disseny model vista controlador, proporciona a l'aplicació una millor arquitectura, i per tant, augmenta la mantenibilitat del producte.

[1] SPA (*Single-Page Application*): Aplicació de página única.

Tot i això, Angular JS presenta una corba de aprenentatge més elevada que altres *frameworks* i aquest fet pot dur a que no s'acompleixin alguns objectius a causa de tenir que aprendre certes competències tècniques.

Figura 3.3 Vue JS

En tercer lloc, Vue JS, és un *framework* progressiu que permet ser adoptat incrementalment, i que se centra en la capa de vista de l'aplicació que és el que té interès en aquest projecte. A més, és un *framework* que permet adoptar altres ferramentes i llibreries, amb la qual cosa seria adequat per al projecte, de cara a integrar altres llibreries de JavaScript o *frameworks* per integrar la aplicació en un entorn d'escriptori.

Figura 3.4 Polymer

Per últim, Polymer, proporciona un conjunt de característiques per desenvolupar elements personalitzats. Aquests elements s'anomenen "*Custom Elements*", i tenen la mateixa naturalesa que els elements del DOM[1] estàndard, és a dir, poden ser etiquetes de la mateixa espècie que un div, img, svg, etc. Aquests també poden tenir un estat que pot ser manipulat, tenir un estil predefinit o definir-lo de forma externa, i responen davant esdeveniments.

Aquest terme, "*Custom Elements*", són la base dels anomenats "*Web Components*", que són un conjunt de APIs que permeten la creació de noves etiquetes, reutilitzables i personalitzables. Aquests es creen, generant un sol fitxer que conté una part en HTML per als elements que componen el nostre component; una definició en JavaScript del comportament del mateix i una especificació CSS per a dotar-li un estil, que podrà ser utilitzat en tot el software i en altres important aquest.

Aquests, són importants al nostre projecte, ja que a una SPA, permeten generar etiquetes o components que poden ser utilitzats a tot el projecte, i en altres projectes del mateix àmbit; de forma que permeten estalviar codi, modularitzar aquest, augmentar la cohesió i dotar al producte de components que seguixen una guia d'estil comú.

[1] DOM (*Document Object Model*): És una interfície de programació per a documents HTML i XML. Representa l'estructura del document, l'estil i el contingut.

Polymer és una tecnologia que controla les dades de manera eficient, mitjançant observadors, que són *callbacks* o cridades quan les dades canvien; propietats que varien segons les dades d'entrada o *data binding* que com ja s'ha especificat, permet actualitzar les propietats i atributs d'un node quan canvien les dades.

Per últim, permet definir respostes davant d'esdeveniments dintre dels propis components, amb la qual els nodes o *web components* creats amb Polymer poden canviar davant esdeveniments.

A més, Polymer és utilitzat per part de Google en aplicacions com ara, Youtube Web, Google Earth, o Google Play Music. Aquestes aplicacions utilitzen components comuns per tal de seguir la seua pròpia guia d'estil, com ara Material Design, i per tant, dotar als seus sistemes de coherència i reutilitzar elements comuns. De la mateixa forma, al nostre projecte es poden definir els propis components.

Aquesta última tecnologia, Polymer, ha estat elegida finalment com a entorn de treball del nostre projecte, més específicament en la versió 2. Ha estat triada per la seua facilitat de creació de components personalitzats que es poden crear a tot el sistema així com la incorporació de *web components* ja creats, amb la qual cosa es pot importar la lògica funcional d'aquests i estalvia gran part de codi i temps de treball. A més, es tracta d'un entorn canviant i es poden crear processos que responen davant d'esdeveniments així com fer ús del *data binding*. Si es compara amb els altres *frameworks* anteriorment especificats, cal destacar que React és un *framework* que en el moment d'anàlisi de la tecnologia a utilitzar encara es trobava a una etapa inicial amb la qual cosa suposava un obstacle aprendre un nou *framework* que podria estar sotmès a actualitzacions; per altra banda, Polymer presenta una corba d'aprenentatge més baixa que Angular JS i respecte a Vue JS i els web components, Polymer mostra un major nombre de funcionalitats respecte a aquests.

Com s'ha comentat anteriorment, aquest projecte té com a objectiu la migració d'una aplicació que està basada en tecnologia d'escriptori, per a sistemes operatius Windows. Realitzar una migració a una plataforma web ajuda a que augmente la compatibilitat del software ja que s'incrementa el nombre de dispositius que utilitzaran l'aplicació. Tot i això, l'elecció d'aquest tipus de tecnologia s'ha fet tenint en compte que l'objectiu d'aquest projecte és migrar tota la capa de client o de presentació i que sols és la primera etapa d'una migració final de tota l'aplicació i que permetrà el seu ús en qualsevol dispositiu. És a dir, per a aquest projecte es limita l'ús de l'aplicació a dispositius d'escriptori degut a que s'utilitzen formularis del software previ desenvolupats amb tecnologia d'escriptori.

Com que l'adaptació de l'antic software a un nova tecnologia és una tasca duradera, i en aquest treball s'ha passat la quasi totalitat de la capa de presentació de l'antic sistema, exceptuant els formularis de Visual Basic 6 d'accions sobre el sistema;

es requereix d'una tecnologia que integre la nova tecnologia en l'antiga i mantinga la compatibilitat. Per a poder integrar el nou desenvolupament sobre l'antic sistema s'utilitza CEF.

El *framework* CEF (*Chromium Embedded Framework*). *Chromium Embedded Framework* és un marc de codi obert basat en el projecte Google Chromium. Integrar el projecte en CEF permet, a més de poder incrustar el software web en una finestra d'escriptori mantenint totes les característiques funcionals, controlar la càrrega de recursos, navegació, menús contextuals, i altres activitats utilitzant el mateix rendiment i tecnologies que s'utilitza en el navegador Google Chrome que té la mateixa naturalesa i està desenvolupat baix el projecte Chromium.

Aquest entorn de treball ha estat utilitzat per importants empreses tecnològiques per portar les seues aplicacions web a entorns d'escriptori com ara, Adobe, Amazon, o Spotify.

Figura 3.5 ***Chromium Embedded Framework***

4. Enginyeria de requisits

L'enginyeria de requisits és la primera etapa en l'enginyeria del software. Els requisits software són descripcions del que deu acomplir el sistema, el servei que ofereix i les restriccions d'operació. En aquesta etapa és realitza l'el·licitació de requisits, és a dir, identificar els requisits, saber d'on venen i com obtenir-los. A més, també s'inclou la definició i especificació de requisits, la seua validació i per últim, la gestió adequada dels mateixos.

L'èxit d'aquesta etapa només es pot obtenir amb la participació col·laborativa dels anomenats *stakeholders* o gent interessada i l'equip de desenvolupament. La enginyeria de requisits és una etapa particularment crítica ja que els problemes comesos en aquesta etapa repercutiran inevitablement en les etapes posteriors de disseny i desenvolupament.

Existeixen diferents tipus de requisits, que es troben a tres nivells, requisits de negoci, d'usuari i requisits software. Aquest capítol se centrarà en mostrar els requisits de negoci i software d'aquest desenvolupament.

4.1. Requisits de negoci

Els requisits de negoci descriuen el propòsit d'alt nivell i les necessitats del producte per a augmentar guanys, reduir despeses, etc. A continuació es descriuran els requisits de negoci del projecte.

En aquest procés és important definir a què tipus de companyies està enfocat el sistema, qui ho utilitzarà, qui tindrà accés al sistema i quines tasques cal fer amb el sistema. També cal obtenir informació sobre qui són els *stakeholders* i per quina raó el sistema pot millorar el seu desenvolupament.

En primer lloc, l'àmbit d'ús del sistema és el d'una empresa petita o gran que vol tenir accés als seus recursos empresarials, com ara factures, clients, etc. També necessita tenir accés a analítiques gràfiques com comparatives o gràfiques de dades. El sistema deu ser adaptable a diferents àmbits empresarials ja que es distribueix a diferents sectors. Aleshores, depenent del tipus d'empresa tindran accés a diferents tipus de recursos.

En segon lloc, el sistema deu ser accedit per diferents tipus d'usuaris dintre d'una empresa segons el seu lloc de treball, amb la qual cosa existeixen diferents tipus d'usuaris. Aquesta característica ve determinada per diferenciar els usuaris en grups de seguretat en funció de la responsabilitat i els permisos d'accés que tinguen.

Pel que fa als *stakeholders* o gent interessada amb el desenvolupament del projecte, podem destacar diversos casos:

- Empresa productora del sistema: En aquest cas, la empresa que desenvolupa el software està interessada en el desenvolupament del sistema per tal de vendre'l als seus clients. Al tractar-se d'una migració, es tracta de donar als clients actuals un millor servei, i d'intentar captar a més clients. L'empresa seguint una estratègia B2C (*Business-to-Consumer*), intenta arribar a clients directament amb el producte.
- Distribuïdors del producte: L'empresa també segueix una estratègia B2B, (*Business to Business*), en la que posa a disposició el sistema a distribuïdors per tal de que tinguen el paper de intermediaris. Un canvi en el sistema suposa un interès de cara als distribuïdors a l'oferir un producte més competitiu.
- Empresa que adopta el sistema: La empresa que adopte el sistema deu veure cobertes les seues necessitats. Per un costat, la empresa que posseïa el sistema previ a la migració deu sentir que s'ofereix un producte modern amb una millora des d'un punt de vista funcional i d'usabilitat, amb la qual cosa resulta atractiu el desenvolupament. Per altra banda, una empresa que utilitze el sistema per primera vegada també es deu sentir interessada ja que s'obri una alternativa a qualsevol producte que estigués utilitzant.
- Clients d'una empresa que adopte el sistema: En cas que una empresa adopte el sistema, si el sistema resulta en un augment del seu rendiment productiu pot portar a un augment en la satisfacció dels clients d'aquesta empresa.

4.2. Requisits software

Els requisits de software són descripcions de les necessitats funcionals i no funcionals que el software deu realitzar per a satisfer els requisits d'usuari i del negoci.

Seguidament, s'exposarà la el·licitació de requisits portada a terme, la qual determina la forma d'obtenir els requisits.

4.2.1. El·licitació de requisits

La el·licitació de requisits és la primera activitat en el desenvolupament software i consisteix en determinar d'on venen el requisits i com obtenir-los. De forma genèrica un procés d'el·licitació consta dels següents processos.

En primer lloc, es realitza una definició d'objectius. La definició d'objectius inclou definir els objectius organitzacionals i descriure el problema. Per una banda, l'objectiu organitzacional principal és el de desenvolupar un nou sistema ERP, basant-se per aquest objectiu en el sistema previ que distribuïa l'empresa. Es tracta d'una transició d'un sistema existent a una nova tecnologia. L'objectiu de negoci principal és el de obtenir un producte competitiu al mercat que permeta oferir una renovació de la plataforma prèvia amb la qual cosa es mantinguen les funcionalitats existents i s'oferisquen noves gràcies a les avantatges que ofereixen les tecnologies web *Open-Source*. És a dir, el problema es defineix com migrar un sistema ja existent i mantenir els requisits del sistema dotant-lo de les millores que ofereixen les noves tecnologies.

Existeixen algunes restriccions sobre el sistema, com ara que la implementació d'aquest projecte consisteix en dotar d'una nova capa de presentació al sistema de partida, amb la qual cosa alguns dels formularis o finestres de l'aplicació es llancen amb la tecnologia anterior amb la intenció d'adaptar tots aquests formularis a la nova tecnologia de la mateixa manera que s'ha fet amb la resta de requisits funcionals. Aquesta restricció causa que una vegada aquest desenvolupament estiga acabat, no es puga utilitzar temporalment en altra plataforma que no siga un dispositiu d'escriptori per tal de poder veure aquests formularis. Per tant, en el moment en que es migre completament aquesta part, es podria utilitzar en diferents plataformes.

En segon lloc, es porta a terme una adquisició coneixement base. Aquest procés pot dur-se a terme gràcies al fet d'analitzar el sistema previ. Com s'ha especificat anteriorment, el software del qual es partix està desenvolupat baix tecnologia Visual Basic 6. Aquesta tecnologia fa possible que el sistema funcione en dispositius d'escriptori. El producte ERP es integrat en una empresa i pot ser personalitzat en funció de l'àmbit de treball de l'empresa client.

Pel que fa a l'anàlisi funcional, es poden extraure captures del sistema per poder estudiar que requisits funcionals mínims deu acomplir el sistema així com que aspecte presenta el software en l'àmbit de la experiència d'usuari i interfície d'usuari per poder adaptar el software actual a una experiència d'usuari semblant per a mantenir la satisfacció del client.

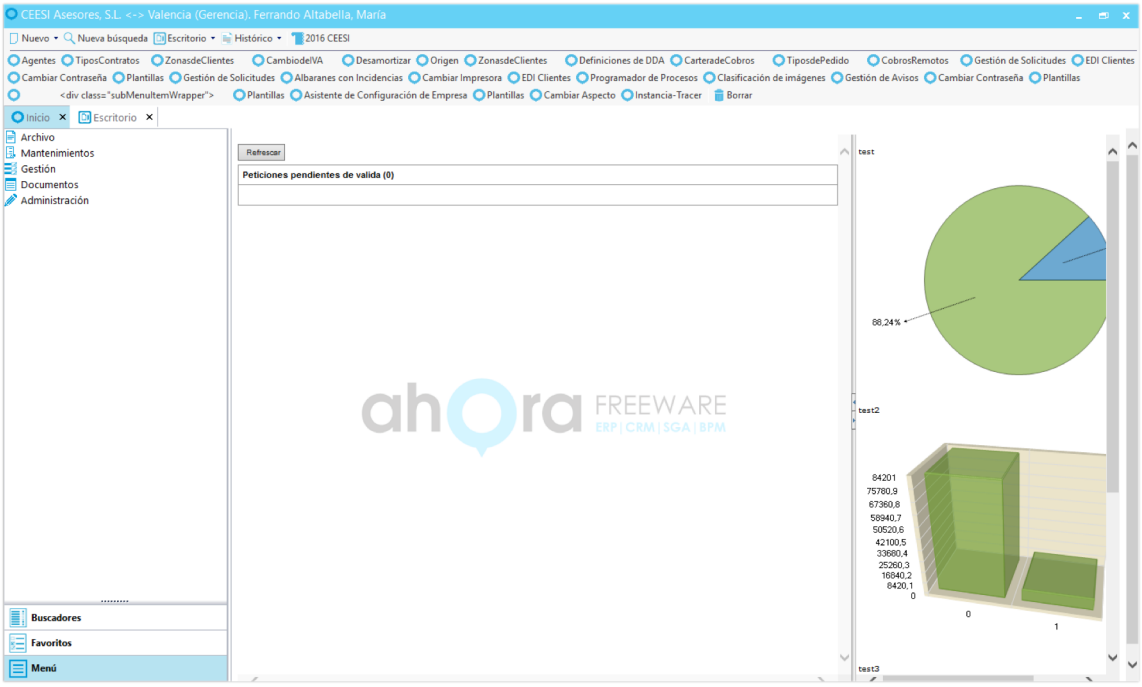

Figura 4.1 Menú i *dashboard* del sistema ERP original

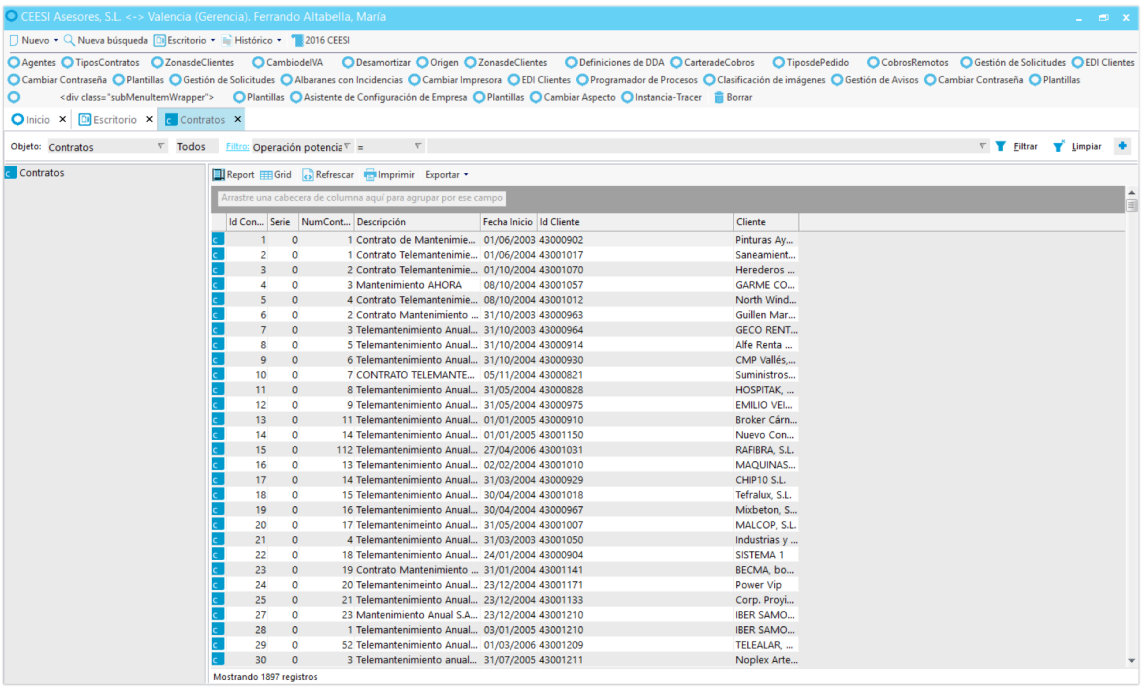

Figura 4.2 Cerca en sistema ERP original

Com podem observar a les dues captures, el sistema ERP presentava una barra amb accessos directes a formularis, recursos, etc., així com uns accessos a realitzar cerques al sistema, i mostrar últims formularis accedits, a la part superior.

En la part esquerra inferior podem accedir als elements que em posat a favorits, veure cerques predefinides i accedir a formularis del sistema.

A la part dreta se'ns exposen indicacions gràfiques sobre el rendiment d'objectes com ara factures, clients, etc. Quan realitzem una cerca se'ns mostra a una taula informació, dintre d'una nova pestanya. Aquesta informació pot ser visitada polsant sobre l'objecte. Sobre la taula es poden realitzar diferents accions com ara imprimir, filtrar segons valors, generar un arxiu Excel amb les dades, etc.

Pel que fa a cada objecte, es pot observar una estructura on es relaciona amb altres recursos. És a dir, per exemple, per a un client, es poden observar els distints contactes d'aquest, sent aquest contacte un objecte que també podria ser visualitzat de la mateixa forma.

En tercer lloc, la organització del coneixement. En aquest punt, es poden identificar *stakeholders*, subsistemes i límits. Es pot fer ús de tècniques d'el·licitació com ara perfils de stakeholders, models de actors, etc.

Nom	Rol	Usuari directe	Interessos
Empresa Ahora Freeware	Treballador de l'empresa	Sí	Augmentar ingressos al oferir un producte més competitiu, així com utilitzar un nou sistema propi
Distribuïdor del sistema	Distribueix el sistema dintre del sector empresarial	Sí	Oferir un producte més competitiu a les empreses a les que es distribueix el sistema. L'objectiu principal és augmentar el nombre de clients
Client del sistema	Usuari empleat a una empresa usuària del sistema	Sí	Incrementar la seua productivitat i satisfacció, així com una millora de rendiment
Client d'una empresa que adopta el sistema	Individu que adquireix un producte	No	Una major satisfacció pels productes oferts per la empresa usuària del sistema

Taula 4.1 Perfils de *stakeholders*

El sistema no consta de cap subsistema, ja que no està dividit en diferents sistemes en funció de l'usuari ni en funció de diferents requisits funcionals. Sí que pot presentar mòduls o configuracions per a determinats administradors del sistema que un usuari del sistema no té.

Per últim, la adquisició de requisits. Aquest procés consta d'algunes activitats com ara identificar alguns requisits de *stakeholders*, requisits del domini, etc. Per a aquest procés, s'estudia el domini del problema. El domini del problema o àmbit és l'empresarial, i per tant, cal tenir en compte requisits que esperen els usuaris com ara obtenir informació sobre recursos empresarials, gràfics de rendiment de la producció, i cerca de recursos específics. Tots aquests requisits s'especifiquen més endavant en l'anomenada especificació de requisits. Aquest procés, per tant, resulta recomanable per situar el projecte en el context en el que es desenvolupa.

4.2.2. Especificació de requisits

La especificació de requisits és una etapa de la enginyeria de requisits que consisteix en la descripció detallada dels requisits. Té com a entrada diferents objectius, requisits del sistema, propietats rellevants del domini, etc., i com a eixida una primera versió de la ERS (Especificació de Requisits Software) organitzats d'acord a una estructura coherent que la faça completa, fàcil de verificar i no ambigua.

Per a realitzar aquesta especificació de requisits se seguirà una estructura semblant a la del model de la ERS de l'estàndard IEEE 29148:2011, que reemplaça al model IEEE 830:1998, però consta de la mateixa estructura.

En primer lloc, el propòsit d'aquest desenvolupament és el d'obtenir una migració de la capa de client d'un sistema ja que va estar creat amb una tecnologia que passa a ser obsoleta i aleshores deixa de tenir suport i necessita ser adaptat a les noves tecnologies.

Pel que fa a l'abast, el producte es desenvolupa per a un el seu ús en un entorn empresarial, més concretament, al tractar-se d'un ERP, serà utilitzat per empreses amb la necessitat de gestionar els seus recursos i sistemes de logística, producció, etc.

En el que a la perspectiva del producte es refereix, al tractar-se de la primera etapa de la migració de tot el sistema en el seu conjunt, el producte final d'aquest projecte és un subsistema o formarà part d'un sistema major.

Les característiques del usuaris que van a utilitzar el producte són les d'una persona que pot tenir diferents càrrecs dintre de l'empresa que utilitza el sistema, des d'un administratiu, fins un gerent, un comptable, etc. Per aquest motiu, el producte software no està concebut per a un usuari expert amb amplis coneixement d'ús de sistemes informàtics. És a dir, el sistema deu ser fàcil d'utilitzar e intuïtiu.

Pel que fa a les restriccions que s'apliquen sobre el producte podem destacar que encara que la tecnologia del producte siga web, el resultat final d'aquest projecte

serà un software que s'executarà sobre dispositius d'escriptori a causa de la necessitat d'utilitzar alguns dels formularis del software previ que ha estat migrat, a l'espera de que es migre tot el sistema a la nova tecnologia escollida.

Hi ha suposicions i dependències com ara que els requisits a continuació descrits són estables, i s'assumeix que els usuaris d'aquest sistema estan familiaritzats amb l'àmbit empresarial, és a dir, que els conceptes tècnics que s'exposen al sistema són comprensibles pels usuaris. A més també se suposa que els usuaris estan familiaritzats amb l'antic sistema i per tant coneixeran les característiques del nou producte.

Com que es tracta de migrar un sistema ja existent, es tracta de replicar els requisits funcionals que aquest presenta. Es poden, però, definir requisits de disseny que deu acomplir el sistema. És a dir, requisits que deuen complir els diferents components gràfics que conformen el sistema i que estan relacionats amb els requisits funcionals del sistema antic.

Nom del requisit	**Informació en finestra principal**
Descripció del requisit	En la finestra principal del sistema existeix una pestanya fixa anomenada *dashboard* conformada per components en forma de targetes que contenen informació sobre els recursos de la empresa en forma de comparatives, gràfiques de línies, gràfiques de pastís, gràfiques de barres, taules amb informació de recursos, etc.
Prioritat del requist	Alta

Taula 4.2 Informació en finestra principal

Nom del requisit	**Guardar *dashboard* actual**
Descripció del requisit	El dashboard actual configurat per l'usuari deu poder ser guardat per l'usuari per tenir-lo disponible. Aquest *dashboard* pot tenir un nom que l'identifica i es carrega al iniciar l'aplicació l'últim guardat. Per altra banda, el *dashboard* actual pot ser modificat i guardat de nou
Prioritat del requist	Alta

Taula 4.3 Guardar *dashboard* actual

Nom del requisit	**Netejar *dashboard* actual**
Descripció del requisit	El *dashboard* actual pot ser buidat amb el desig de configurar una nova vista per a l'usuari. Abans de buidar el *dashboard* aquest pot ser guardat per començar de nou
Prioritat del requist	Alta

Taula 4.4 Netejar *dashboard* actual

Nom del requisit	**Afegir component al *dashboard* actual**
Descripció del requisit	Deu ser possible afegir una targeta al *dashboard* actual per tal de configurar la vista. Aquests components poden ser targetes de cerques predefinides de recursos, elements gràfics com taules, gràfiques, *widgets*
Prioritat del requist	Alta

Taula 4.5 Afegir component al *dashboard* actual

Nom del requisit	**Carregar *dashboard***
Descripció del requisit	L'usuari deu tenir la capacitat de carregar una estructura de *dashboard* predefinida per ell. Aquests *dashboards* han estat tots guardats per l'usuari
Prioritat del requist	Alta

Taula 4.6 Carregar dashboard

Nom del requisit	**Crear un nou objecte**
Descripció del requisit	L'usuari pot crear un nou recurs al sistema. Aquest recurs o objecte pot ser, un client, una factura, etc.
Prioritat del requist	Mitjana

Taula 4.7 Crear un nou objecte

Nom del requisit	**Consultar històric de accessos a formularis**
Descripció del requisit	L'usuari pot accedir a diferents formularis que produeixen canvis al sistema. Cada un d'aquests formularis apareixerà al desplegable d'històric on l'usuari pot tornar a tenir accés
Prioritat del requist	Alta

Taula 4.8 Consultar històric de accessos a formularis

Nom del requisit	**Buscar recurs**
Descripció del requisit	L'usuari pot cercar un recurs com un client, contacte, etc., al sistema. Aquests tipus d'objectes estan predefinits i la cerca de recursos es fa definit el nom del recurs a cercar
Prioritat del requist	Alta

Taula 4.9 Buscar recurs

Nom del requisit	**Cerca avançada de recurs**
Descripció del requisit	L'usuari pot realitzar una cerca avançada al sistema. Aquesta cerca avançada consisteix en definir el valor per a cada propietat d'un objecte o recurs que es desitge cercar
Prioritat del requist	Mitjana

Taula 4.10 Cerca avançada de recurs

Nom del requisit	**Filtre de cerca**
Descripció del requisit	L'usuari pot realitzar una cerca i definir un valor per als distints filtres que existeixen en funció del recurs
Prioritat del requist	Mitajana

Taula 4.11 Filtre de cerca

Nom del requisit	**Taula resultat de cerca**
Descripció del requisit	Quan l'usuari realitza una cerca, aquesta es motra a l'usuari en forma de taula. Cada entrada de taula correspon a un element que coincideix amb els valors o propietats cercades. Quan selecciona un element de la taula, es poden realitzar diferents accions amb aquest
Prioritat del requist	Alta

Taula 4.12 Taula resultat de cerca

Nom del requisit	**Vista detallada d'objecte**
Descripció del requisit	Quan un usuari selecciona un objecte de la taula, té l'opció de veure una vista dellada de l'objecte realacionant-lo amb altres elements de la base de dades com una mena de graf. Aquesta vista s'anomena arbre
Prioritat del requist	Alta

Taula 4.13 Vista detallada d'objecte

Nom del requisit	**Enviar element a favorits**
Descripció del requisit	Tant un element com un formulari o un recurs que haja estat cercat pot ser enviat als favorits de l'usuari amb la intenció de ser accedit en qualsevol moment. En la secció de favorits poden ser creades carpetes per agrupar elements
Prioritat del requist	Mitjana

Taula 4.14 Enviar element a favorits

Nom del requisit	**Enviar element a correu**
Descripció del requisit	L'usuari deu de poder enviar un element cercat de la taula
Prioritat del requist	Baixa

Taula 4.15 Enviar element a correu

Nom del requisit	**Menú de formularis del sistema**
Descripció del requisit	L'usuari deu de poder tenir accés a distints formularis per a modificar aspectes del sistema com canviar la contrasenya o fer canvis sobre recursos. Aquests diferents formularis deuen aparèixer sobre un menú
Prioritat del requist	Alta

Taula 4.16 Menú de formularis del sistema

A continuació, s'exposaran alguns dels requisits no funcionals que ha de complir el nostre sistema.

Nom del requisit	***Responsive design***
Descripció del requisit	El sistema deu adaptar els elements que componen la interfície a la resolució del dispositiu en el que s'executa. Les targetes que es mostren al *dashboard* han de ser *Responsive Design*, és a dir, s'han de ajustar automàticament i redimensionar a l'espai visible per l'usuari
Prioritat del requist	Alta

Taula 4.17 *Responsive design*

Nom del requisit	**Usabilitat**
Descripció del requisit	Capacitat que permet a l'usuari controlar el sistema i operar-lo amb facilitat. Així, com la capacitat de la interfície de agradar a l'usuari i satisfer la interacció amb l'usuari. Les característiques dels usuaris que van a utilitzar el sistema no són necessàriament les d'un usuari expert i, per tant, el sistema deu ser amigable, intuïtiu i fàcil d'utilitzar
Prioritat del requist	Alta

Taula 4.18 Usabilitat

Nom del requisit	**Modularitat**
Descripció del requisit	Capacitat que permet a un sistema que un canvi en un dels seus components tinga un impacte mínim en els demés. Una de les raons de adoptar els *web components* es que al realitzar un canvi en un d'aquests components no produeix que s'haja de realitzar més canvis en els que depenen d'aquest. Amb aquest fet s'aconsegueix que es puguen afegir nous mòduls o eliminar alguns dels presents, per tant, el sistema adopta també escalabilitat al poder incrementar les característiques funcionals
Prioritat del requist	Alta

Taula 4.19 Modularitat

5. Disseny de la solució

Abans de començar el desenvolupament del producte que correspon al procés de convertir la especificació del sistema en un producte executable, és necessari portar a terme una descripció del disseny i arquitectura de la nostra aplicació.

El disseny de software s'entén com una descripció de la estructura del software que va a implementar-se, els models i les estructures de dades utilitzades pel sistema, així com les interfícies entre components del sistema.

5.1. Disseny arquitectònic

El disseny arquitectònic correspon a la primera etapa del disseny de software, i es tracta del procés de identificar l'estructura global del sistema, els principals components, les seues relacions i com es distribueixen.

El disseny arquitectònic resulta important per diverses raons, com ara, el traçat de l'arquitectura requereix de cert anàlisi que porten a conèixer si el sistema pot o no cobrir requisits crítics com el rendiment, fiabilitat o mantenibilitat. També resulta interessant pel fet que un model arquitectònic pot ser semblant per a sistemes amb requisits similars i per tant, pot ser reutilitzat en aquests.

A continuació, es mostra el disseny arquitectònic del nostre software:

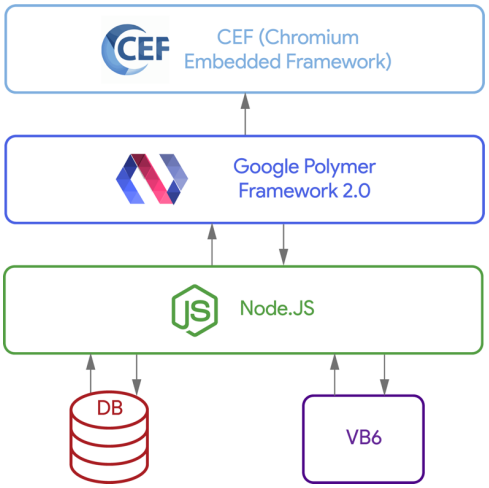

Figura 5.1 Diagrama del disseny arquitectònic per capes

En aquest diagrama es mostra una disseny arquitectònic per capes que està constituïda per:

- CEF (*Chromium Embedded Framework*): *Framework* de Google basat en el motor Chromium que compta amb una capa de comunicació entre la capa web que renderitza per tal d'interactuar amb components del sistema.

  Aquesta capa renderitza els components web per tal de mostrar el sistema al usuari i permet que el sistema desenvolupat baix tecnologia web siga funcional en dispositius d'escriptori.

- Google Framework Polymer 2.0: Entorn de treball de Google baix el qual està desenvolupat el projecte i la part frontend o capa de presentació d'aquest sistema. Aquesta capa està conformada per components desenvolupats amb tecnologia HTML5, CSS3, i Javascript en l'estàndard ECMAScript 6.

  Aquests components són els anomenats *web components* que conformen els diferents elements de la interfície gràfica amb els quals interactua l'usuari i que es comuniquen entre si.

  Aquesta capa es mostra a l'usuari en dispositius d'escriptori mitjançant la capa del CEF, i aquesta la incrusta per tal de mostrar-la a una finestra. Per tant, aquesta capa és el que constitueix el frontend del nostre sistema.

- Node.JS: Aquesta és la capa de backend del nostre sistema que comunica amb la base de dades SQL. S'encarrega d'atendre les peticions que es fan des de la part frontend, i fa les cridades a la base de dades i en torna les respostes.

- Visual Basic 6: Aquesta capa opera en el nostre sistema proporcionant alguns dels formularis presents en l'antic software. És a dir, es conserven alguns dels formularis del sistema inicial baix Visual Basic 6.

- Base de dades SQL: En la capa de persistència es troba la base de dades baix tecnologia SQL.

El procés de migració resulta llarg ja que el sistema inicial consta d'una magnitud de requisits significant. Amb la qual cosa, en aquest projecte, no es porta a terme una migració de la estructura de la base de dades, sinó que s'adapta el *backend* per poder utilitzar la estructura anterior. Per altra banda, l'aplicació inicial, desenvolupada baix tecnologia Visual Basic 6, conté un gran nombre de formularis de creació de recursos empresarials i consultes, amb la qual cosa es reutilitzen alguns d'aquests formularis. Integrar una tecnologia moderna com JavaScript amb una d'antiga com Visual Basic 6 és possible gràcies a l'entorn de CEF.

5.2. Prototipat

A continuació es mostren els prototips de la interfície d'usuari que han estat dissenyats per a complir les característiques funcionals del sistema software. Aquests han estat realitzats utilitzant la ferramenta software de disseny Sketch, que permet de forma precisa donar una visió del resultat final del nostre producte.

El prototipat de interfícies resulta adequat per tal de donar una visió als desenvolupadors dels elements que compondran el nostre sistema i amb els quals podrà interactuar l'usuari. Resulta adequat portar a terme un anàlisi de interfície d'usuari (UI) i d'experiència d'usuari (UX), per tal de conèixer el tipus d'usuari al que va dirigit el producte i la millor forma de que el sistema resulte intuïtiu i fàcil d'utilitzar.

Per a la gestió dels prototips de interfície i els canvis sobre aquests s'ha fet ús de la plataforma InVision App, la qual permet compartir els prototips amb tots els membres d'un equip de desenvolupament per tal de tenir-los sempre disponibles així com disposar de característiques que proporciona la pròpia ferramenta com l'anàlisi dels components que conformen el prototip i la seua conversió a CSS3. Per altra banda, aquest sistema facilita el fet de compartir els prototips amb tots els membres d'un equip i qualsevol canvi es reflexa i resulta una forma àgil de treballar.

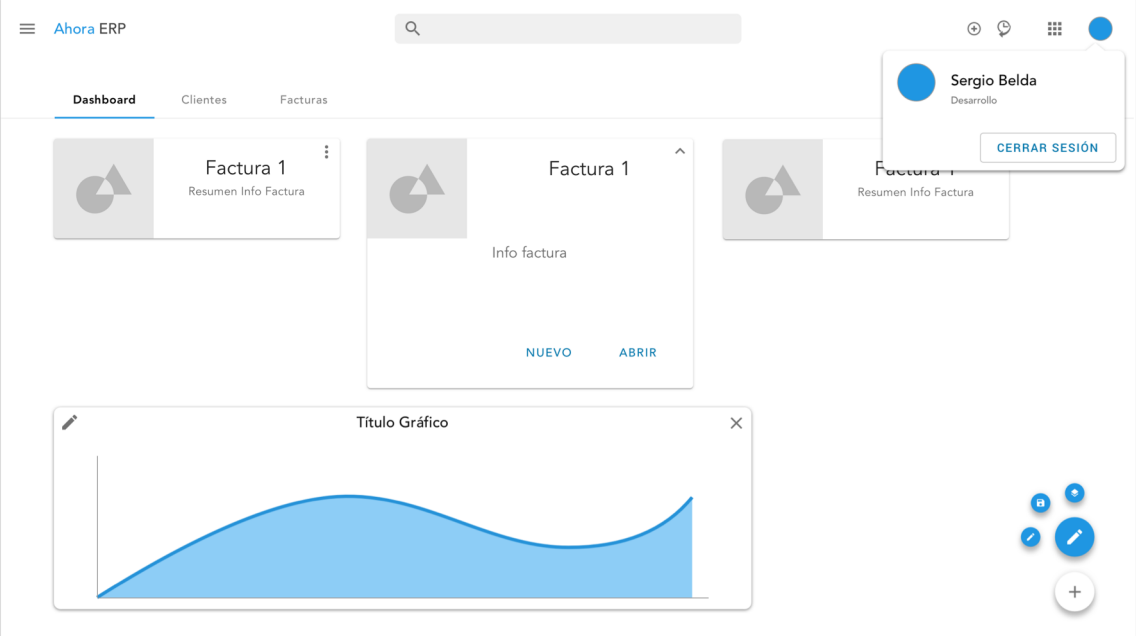

Figura 5.2 Finestra principal de *dashboard*

A la finestra principal deu figurar la pestanya principal en la qual apareixen els distints indicadors en forma de gràfiques i targetes amb informació. Aquests elements són personalitzables i deuen de poder ser moguts per la finestra. La disposició i estructura deu poder ser guardada gràcies a les opcions dels botons circulars de la banda inferior.

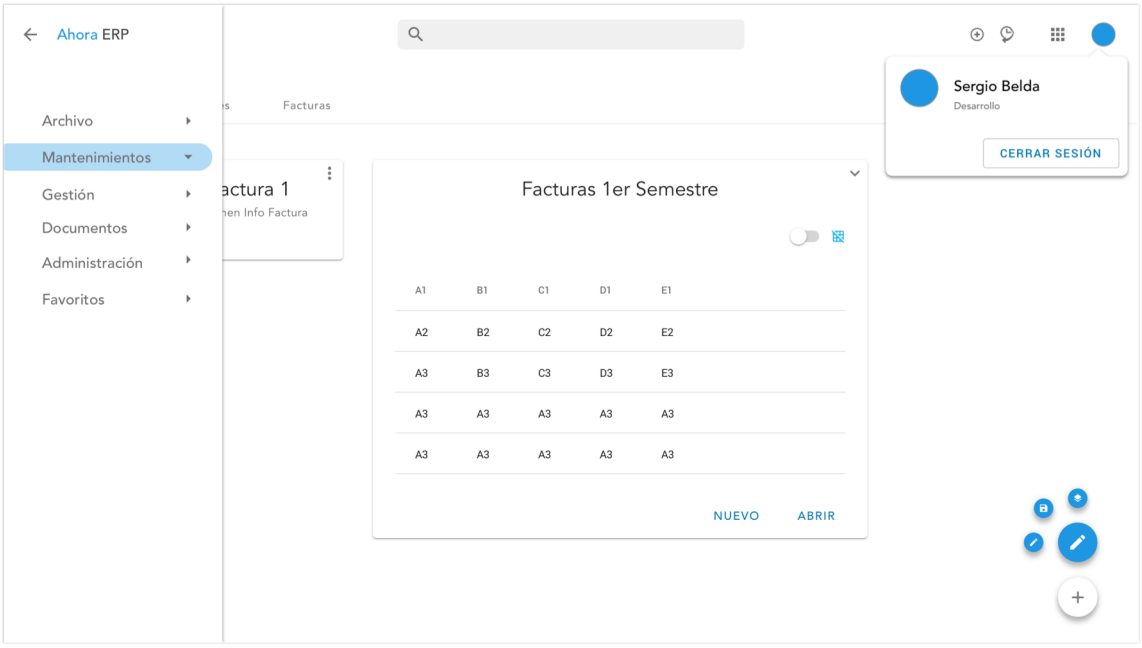

Figura 5.3 Menú lateral esquerre d'accés a formularis

A la banda esquerra del nostre sistema deguem disposar d'una llista d'elements desplegables que contenen els diferents formularis que poden ser llançats al nostre sistema. Alguns d'aquests formularis poden ser el de canviar la contrasenya o afegir un un nou usuari.

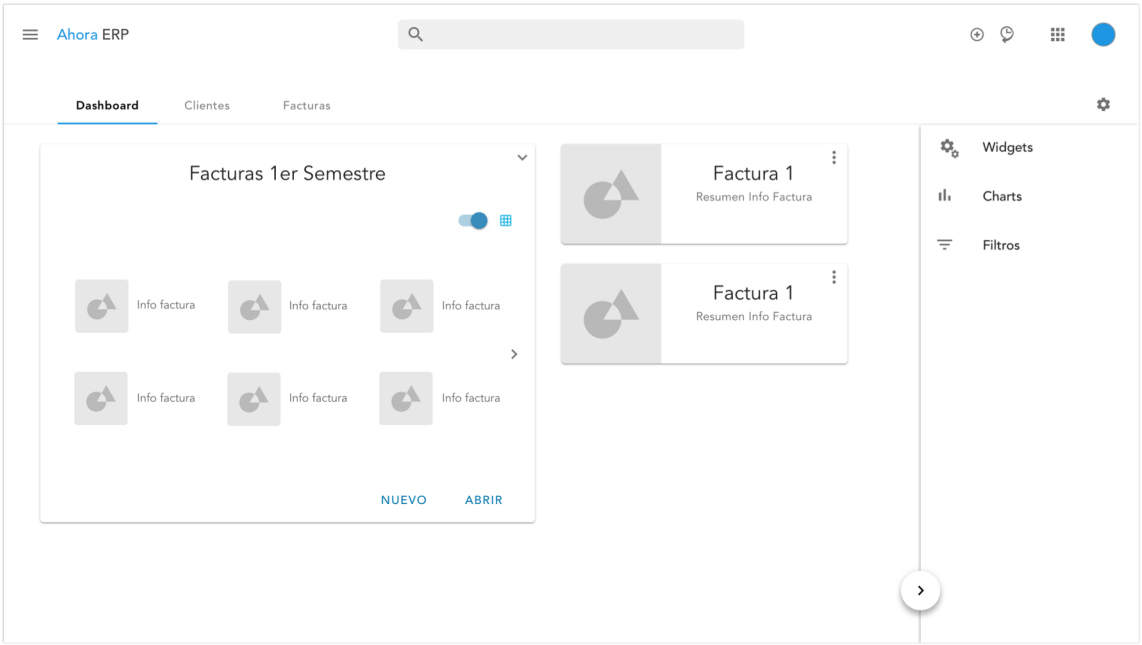

Figura 5.4 Menú lateral dret amb objectes que importar al *dashboard*

A la pestanya principal del sistema, deu aparèixer un botó circular per afegir elements al *dashboard*. Aquests elements poden ser gràfiques, comparatives de

recursos, i cerques predefinides. Aquests elements apareixen a aquest desplegable després de que l'usuari els afegeixi.

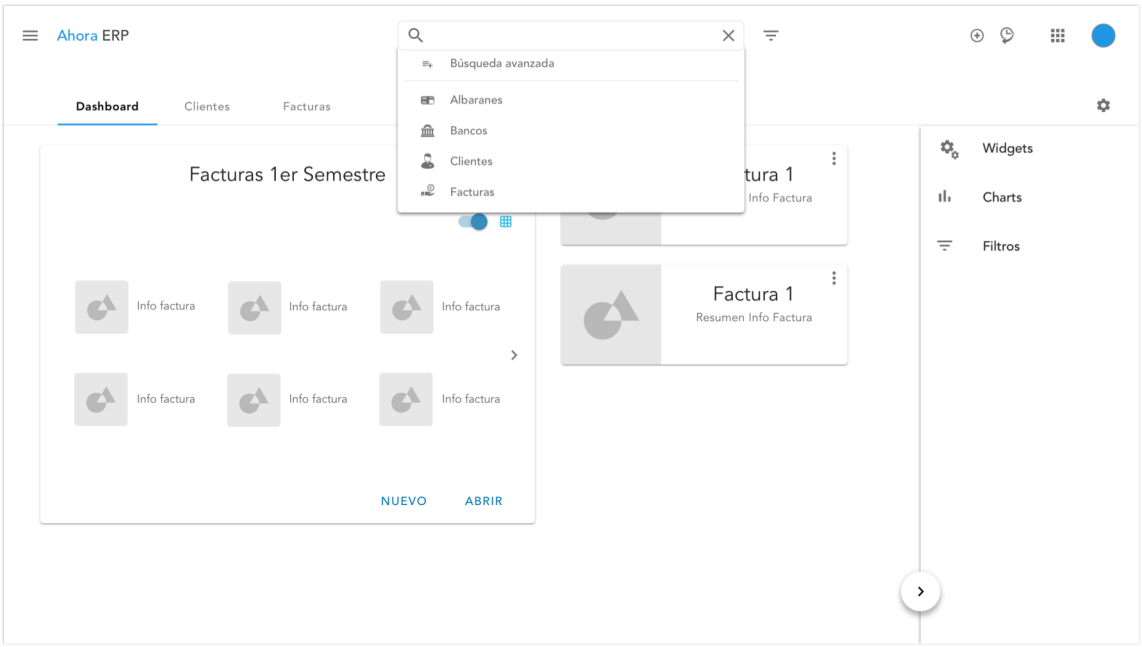

Figura 5.5 Desplegable de cerca

En la part superior deu aparèixer d'una barra de cerca que dispose d'un desplegable amb els diferents recursos que poden ser cercats.

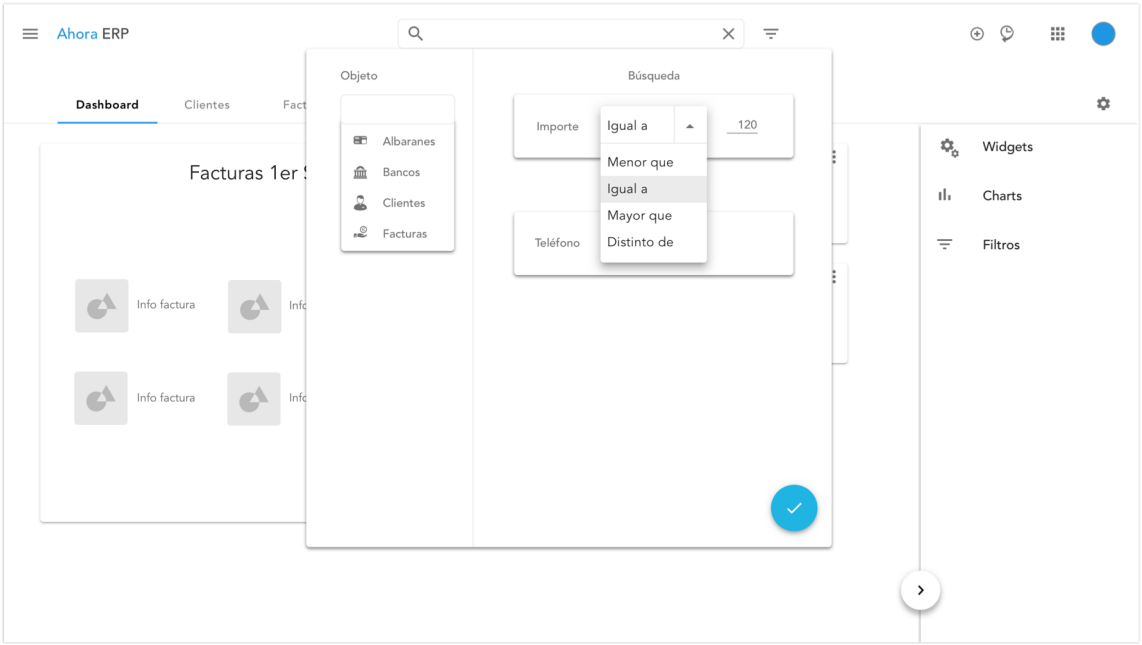

Figura 5.6 Desplegable de cerca avançada

El desplegable de cerca avançat és un element que deu permetre a l'usuari accedir a les diferents propietats de cada recurs de manera que es puga personalitzar i especificar molt més la cerca. El desplegable consta d'una columna amb els diferents

recursos que poden ser cercats i les propietats per a cadascun deuen ser arrastrades a una àrea en la que es configuren els valors que deuen prendre.

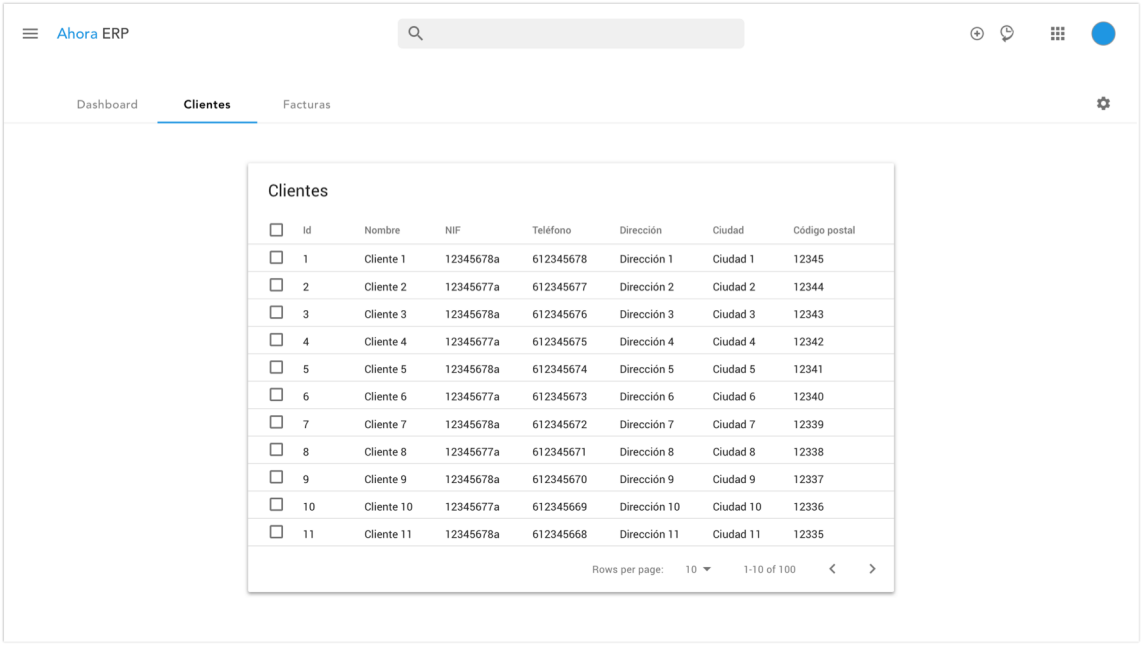

Figura 5.7 Pestanya de cerca

Cada cerca genera una nova pestanya en la que deu aparèixer una taula en la que cada entrada són els diferents elements de la base de dades que coincideixen amb la nostra cerca. Cada entrada de taula deu de poder ser seleccionada i per tant, sobre cada item es poden realitzar diferents accions. Aquestes accions apareixen a un menú contextual. A aquesta vista deu de poder crear-se un nou objecte al sistema per a la col·lecció cercada i per tant, deu aparèixer un botó per afegir un nou objecte.

Pel que fa a la taula, deu de poder visualitzar-se una vista en que cada element de la taula es mostre com una targeta. Sobre aquesta taula també es poden realitzar cerques per a un valor d'una propietat en particular.

Finalment, aquests dissenys s'han desenvolupat seguint principis propis de UX (*User Experience*). Alguns d'aquests principis són la familiaritat amb altres sistemes, al utilitzar components propis de Material Design; la facilitat d'aprenentatge, al constar de pocs components gràfics iestar basat en el sistema previ; claredat de distribució dels components a la pantalla; i reducció d'errors, a l'evitar que alguns components porten a confusió.

6. Implementació

Una vegada portada a terme les etapes d'enginyeria de requisits i disseny, i definides les unitats de treball que apareixeran al backlog, es pot portar a terme la implementació, la qual constitueix l'etapa de producció software del sistema.

Com s'ha especificat al disseny el sistema està definit per una arquitectura per capes, amb un frontend implementat amb tecnologia Polymer, un backend amb Node.JS, i la integració de l'antic sistema mitjançant la tecnologia CEF.

A aquest capítol es descriu internament el frontend i backend, descrivint les tecnologies i de quina forma s'implementen.

6.1. Frontend

La capa de frontend d'aquest projecte ha estat desenvolupada baix tecnologia web, com s'ha especificat anteriorment, mitjançant l'ús del *framework* Polymer, de Google, el qual es basa en definir els nostres propis components web. Aquest fet permet definir el comportament dels nostres components web que seran utilitzats al sistema i importar components amb la seua lògica implementada per tal de poder adoptar aquesta funcionalitat.

En la implementació d'aquest projecte s'han creat components específics o "custom elements". Aquests són la base dels anomenats web components que són utilitzats en *frameworks* com Vue JS. Es basa en estàndards per crear components reutilitzables utilitzant en la seua creació tecnologia HTML per a la creació estructural del component, CSS per a la personalització del mateix i JavaScript per tal de afegir-li comportament i lògica al component. En resum, amb la creació de nous elements es creen noves etiquetes HTML.

Un element "custom element" es defineix mitjançant JavaScript. Mitjançant una crida de JavaScript es crea el nou component i per tal de ser utilitzar simplement és importat i es col·loca l'etiqueta que el defineix.

```
class TestComponent extends HTMLElement {...}
window.customElements.define('test-component', TestComponent);
```

Mitjançant la crida ".define()" sobre l'element global customElements, es crea el nou element al navegador i es crida de la mateixa forma que s'invoca a una etiqueta div.

Un component es troba definit per una secció d'estil definit amb CSS, una secció HTML on es defineix l'estructura del component, així com un apartat JavaScript on es defineix el comportament del component.

A continuació, es mostra un dels fitxers HTML desenvolupat per al nostre sistema i que constitueix un component web que podrà ser utilitzat tant a aquest projecte com a altre en el que es faça ús de *web components*, sols caldrà importar-lo.

```
<link rel="import" href="../bower_components/polymer/polymer-element.html">
...
<dom-module id="ahora-button">
  <template>
    <style>
      :host {
        display: block;
      }
      .ahora-button {
        ...
        color: #00A7DF;
        border: 1px solid rgba(0,0,0,0.2);
      }
      .ahora-button-blue {
        ...
        color: white;
        border: 1px solid var(--ERPBlue);
      }
      .imgButton {
        margin-right: 8px;
      }
    </style>
    <paper-button id="button" btnclass={{btnclass}}>
      {{btntext}}
    </paper-button>
  </template>
  <script>
    /**
     * Clickable button element used to launch different processes.
     *
     * @class AhoraButton
     *
     */
    class AhoraButton extends Polymer.Element {
      static get is() {
        return 'ahora-button';
      }

      static get properties() {
        return {
          /**
           * ...
           */
          btntext: {
            type: String
          },



          /**
           * ...
           */
          btnclass: {
            type: String,
            observer:"_changeStyle"
          }
```

```
        };
      }

      connectedCallback() {
        super.connectedCallback();
        var ctx = this;
      }

      _changeStyle(newvalue, oldvalue){
        this.$.button.className = `${this.btnclass}`;
      }
    }

    window.customElements.define(AhoraButton.is, AhoraButton);
  </script>
</dom-module>
```

Com podem observar al fragment de codi anterior, entre la etiqueta “style” es defineixen les classes d’estil que s’afegixen als elements web que conformen el nostre component, mitjançant el llenguatge CSS3. A continuació, es defineix l’estructura del nostre component amb el llenguatge HTML5, que a més es tracta d’un component importat al projecte d’una font externa i que com s’ha especificat ajuda a importar tota la funcionalitat present en aquest component. Finalment, entre l’etiqueta <script> es defineixen els mètodes que constitueixen la lògica del nostre component. El primer mètode en ser executat és el “connectedCallback()”, el qual s’invoca al insertar l’element al DOM. Cadascun dels elements posseeix un Shadow DOM, que és un DOM propi de l’element per a que siga independent de la resta, li aplique estil i la represente. L’element “template” permet declarar fragments de DOM que s’analitzen, permaneixen inactius durant la càrrega de la pàgina i poden ser activats més endavant en temps d’execució. Aquests fragments de DOM constitueixen el Shadow DOM de cada component.

Al mètode de “get properties()” es defineixen les propietats del nostre component i podran ser consultades cridant al component i fent una crida al mètode get() passant per paràmetre la propietat. Per a cada propietat es pot definir un observador al qual se li passa el mètode que serà invocat quan la propietat de l’objecte canvie.

El *data binding* és una de les característiques interessants d’aquest entorn de treball i dels web components. El *data binding* connecta les dades d’un component amb les propietats o atributs del DOM d’un altre element. D’aquesta forma al “template” d’un component es poden definir les propietats i atributs sobre components que podran ser modificats dinàmicament.

A l’exemple de codi anterior es fa ús del *data binding* al nostre codi gràcies a introduir la partícula {{propietat}} a la secció del “template”. Depenent de si es fa ús de

{{}} o [[]], la propietat que encapsula serà visible únicament per al propi component i els objectes que es troben a aquest component o podrà serà accedida cap a altres.

```
    <paper-button id="button" btnclass={{btnclass}}>
      {{btntext}}
    </paper-button>
  </template>
  <script>
...
      static get properties() {
        return {
          /**
           * ...
           */
          btntext: {
            type: String
          },
          /**
           * ...
           */
          btnclass: {
            type: String,
            observer:"_changeStyle"
          }
        };
      }
```

A aquest codi podem observar l'ús del *data binding* amb la definició de dues propietats, anomenades "btnclass" i "btntext", que fan referència a la classe o estils que tindrà el nostre element i el text del que disposarà. Amb la qual cosa quan s'utilitze el nostre component es definiran els valors d'aquestes propietats en la creació d'aquest component. A continuació es mostra l'exemple d'ús del nostre component per tal de que el component adopte la classe i el text que ens interessa.

```
<ahora-button btnclass="ahora-button" btntext="Cerrar Sesión"></ahora-button>
```

A la propietat "btnclass", s'ha definit un observador, de manera que quan aquesta propietat siga inicialitzada o canviada en temps d'execució, farà una crida al mètode definit. En aquest cas, en el moment en que s'inicialitza la propietat "btnclass" es fa una crida al mètode "_changeStyle()", amb la qual cosa, es canvia la classe del component "paper-button" que forma part del component "ahora-button" i al seu torn, té les seues propietats i mètodes. El "btntext" indica el text que tindrà el nostre botó i es passat també al component "paper-button".

Aquest *framework* posseeix una altra característica interessant que és la possibilitat d'autogenerar estructures HTML a partir d'estructures de dades com *array* per a cada element del *array*. És a dir, quan es defineix l'estructura d'un component web es pot definir dintre d'una etiqueta "template" els elements que seran generats en funció dels elements que conformen una estructura de dades. Per a aquest propòsit, al "template" es dóna valor a l'atribut is, de forma que si es passa "dom-

repeat", genera l'estructura HTML que es situa dintre del "template" per a cada element del *array*. Per altra banda, quan es passa "dom-if", depenent de si es dóna una condició, es crearà l'element definit dintre del "template".

A continuació, s'exposa una mostra del nostre codi on es fa ús d'aquestes dues propietats de l'entorn de treball.

```
<template id="menu" is="dom-repeat" items="{{items}}" initialCount="3" count="{{id}}">
  <div id="item_{{index}}">
    <paper-icon-item  open="false" tipoobjeto="{{item.TipoObjeto}}" sql="{{item.Sql}}"
idlink="{{item.IdLink}}" descrip="{{item.Descrip}}" class="suggestion" on-click="toggle" on-
contextmenu="options">
      <iron-icon icon="{{item.Icono}}" slot="item-icon"></iron-icon>
      {{item.Descrip}}
    </paper-icon-item>
  </div>
</template>
```

A l'etiqueta "template" es passa a l'atribut is "dom-repeat", per indicar que el que es troba dintre del "template" es generarà per a cada *item* de l'estructura de dades definida a l'atribut *items*. És a dir, per a cada *item* es crearà l'estructura que apareix dintre del "template" i a la qual podran passar-se propietats d'objecte de cada element de l'estructura de dades, de forma que en aquest cas cada nou "paper-icon-item" tindrà als seus atributs les propietats d'objecte de cadascun dels elements del *array*. Aquesta característica també pot ser utilitzada per implementar recursivitat d'estructures de components web de manera que si es té un arbre de dades, per a cada node de l'arbre i cadascun dels seus fills es mostrarà gràcies al "dom-repeat", i alhora per a cada fill podrà seguir-se el mateix procediment.

En cas que es necessite definir l'estrucura del component web en funció d'una determinada condició es farà ús del valor "dom-if" per a l'atribut is. D'aquesta forma, depenent de certes condicions es pot personalitzar un component web. Un exemple d'ús del nostre sistema d'aquesta propietat és el component web de les finestres emergents amb diàlegs. En aquest component s'ha definit al "template" els diferents tipus d'elements que pot tenir, un títol, un camp de text, una botonera, un *view pager* d'elements, etc. D'aquesta forma en la generació d'aquest component es pot decidir que elements apareixeran, per tal d'estalviar codi i crear un component genèric per a les diferents funcionalitats.

```
<div id="textbody" class="textBody">{{descrip}}</div>
<template is="dom-if" if="{{_equals(hastextfield, 'true')}}">
  <paper-input id="input"></paper-input>
</template>
<template is="dom-if" if="{{_equals(hasviewpager, 'true')}}">
  <ahora-view-pager id="viewpager" style="display: flex; margin-bottom: 16px;"></ahora-
view-pager>
</template>
<template is="dom-if" if="{{_equals(hasbuttons, 'true')}}">
  <div id="buttons" class="profile-dialog-row-end" style="margin-top: 16px;">
    <ahora-button buttonclass="ahora-button-blue" on-click="multifunction"
```

```
btntext="Sí"></ahora-button>
      <ahora-button buttonclass="ahora-button" on-click="_closeDialog"
btntext="No"></ahora-button>
   </div>
</template>
```

Com podem observar, cada element es troba envoltat de la seua etiqueta "template" pertinent. A cada etiqueta "template" es defineix la condició per la qual seran instanciats els components que hi ha dintre. A l'atribut "if" es fixa el mètode que es cridat amb els paràmetres que són passats. És a dir, en l'exemple anterior es comprova que cada propietat del component siga igual a "true" amb la qual cosa el component de dintre del "template" serà inclòs. Per tant, sobre el component es fa una crida al mètode set per a canviar la propietat del component determinada per tal de fer que un element siga visible o no per a l'usuari.

Per acabar definint les característiques destacables de Polymer, als components web podem definir accions que són llançades quan es produeix un esdeveniment a altre component del sistema.

```
document.addEventListener("cerrarMenuEvt", function (e) {
   try {
      this.body.querySelector("#filterMenu").remove();
   } catch (e) {

   }
});
```

Aquest codi es defineix al mètode "connectedCallback()" del nostre component de manera que en llançar-se l'esdeveniment "cerrarMenuEvt", es porta a terme el codi definit al cos de la funció. Aquesta funcionalitat ajuda a que els components puguen respondre davant d'esdeveniments i que les accions que es produïxquen al llarg del sistema puguen influir al comportament de tots els components, amb la qual cosa s'aconsegueix que hi haja una major cohesió entre components.

En relació al codi implementat, aquest ha estat desenvolupat seguint normes pròpies de l'anomenat *clean code*, o codi net. La implementació d'un codi correcte és un dels factors principals d'èxit d'un producte. Mantenir un codi net implica que el sistema siga correcte des del punt de vista tècnic i que a més, estiga obert a modificacions i millores, ja que aquestes es podran introduir de forma adequada sense implicar introduir possibles errors al sistema. Un exemple del codi correcte el trobem en la elecció del nom de variables i funcions. Normalment, resulta adequat que els noms transmeten una informació essencial sobre el que es pretén fer, que aquests noms puguen ser buscats fàcilment, i que revelen informació sobre el que s'està fent. Per exemple, al següent codi podem observar com el nom de la funció transmet el que es vol fer, buidar la vista actual de targetes; com s'obté la llista de targetes gràcies al

context, “ctx”, que es passa al objecte JSON com a paràmetre de la funció, “params”, i com la funció “remove()” transmet que a cada iteració del bucle s’està esborrant una targeta de la vista.

```
var vaciarLayout = (params) => {
   var cards = params.ctx.getElementsByTagName("ahora-card");
   for (var i = cards.length - 1; i >= 0; i--) {
      cards[i].remove();
   }
}
```

Els noms transmeten informació sobre el que es fa, no s’utilitzen noms com ara “list” o “l”, y quan s’utilitzen abreviatures, s’entén a que es fa referència.

En quant a les funcions, es recomana que la mida no siga excessiva, que únicament realitzen una funció, els noms deuen ser descriptius i es deu posar de manifest que fa la funció, el nombre d’arguments no deu ser abundant i evitar la duplicitat de codi. Per exemple, com es mostra a la següent funció, s’afegeix un element a una targeta i es passa el títol i els items que conformen la targeta per argument. La mida no es llarga, fa una única funció i els noms de les variables són descriptives.

```
appendCardItem(titleText, items){
   var paperCard = document.createElement("paper-card");
   this.$.row.appendChild(paperCard);
   var title = document.createElement("div");
   title.innerHTML = titleText;
   title.className = "cardTitle";
   paperCard.appendChild(title);
   var body = document.createElement("div");
   body.className = "cardBody";
   for (var i = 0; i < items.length; i++) {
      var checkbox = document.createElement("paper-checkbox");
      body.appendChild(checkbox);
      checkbox.innerHTML = items[i].Nombre;
      checkbox.onclick = (e) => {
         //...
      }
   }
   paperCard.appendChild(body);
   return paperCard;
}
```

Per últim, el framework Polymer permet la reutilització de components a altres sistemes, de forma que es pot generar un estàndard de components gràfics com fa Google amb Material Design.

Per exemple, al nostre sistema hem definit un component “ahora-toast” per a les notificacions del sistema. Per a aquest propòsit s’ha creat un *web component* i, si es vol utilitzar tant sols caldria importar el component. D’aquesta manera podem crear el nostre propi estàndard de disseny pensat per a la companyia.

```
<dom-module id="ahora-toast">
  <template>
    <style>
    :host {
       display: block;
    }

    .toast {
       border - radius: 6px;
    }
    </style>
    <paper-toast id="toast" className="toast" text="{{text}}" duration="{{duration}}">
      <template is="dom-if" if="{{_equals(duration, '0')}}">
        <paper-button on-click="close" className="yellow-button">CERRAR</paper-button>
      </template>
    </paper-toast>
  </template>
<script>
  ...
  /**
   ...
   */
  open(){
    this.$.toast.open();
  }
...
```

Per tal de utilitzar el component a altre sistema tant sols caldrà importar-lo i fer ús del mètodes que es notifiquen a la documentació. La documentació per al component seria la següent.

setPersistent()	Fa el missatge persistent. Serà necessari tancar-lo. Si es fa persistent, automàticament apareix un botó "CERRAR" per tancar el missatge
open()	Obre el missatge
close()	Tanca el missatge
setText(text)	Inserta el text en el missatge
setStyle(type)	L'argument type deu ser: - ok - error - warning Si es passa qualsevol altra cadena, el missatge apareixerà amb el color neutre de la llibreria de paper-toast

Taula 6.1 Mètodes del component "ahora-toast"

6.2. Backend

Per a la implementació del *backend* del sistema ha estat utilitzat Node.JS, entorn de treball construït amb el motor de JavaScript que treballa en la banda servidor. Utilitza un model d'operacions E/S sense bloqueig orientat a objectes que el fa lleuger i eficient. També s'utilitza Express, un entorn de treball per a Node.JS que permet establir peticions. Mitjançant una instància de Express, i establint una connexió amb la base de dades gràcies a un mòdul de Node.js, podem utilitzar els mètodes de peticions HTML, GET, POST, PUT, DELETE, per poder executar una funció específica.

```
app.post('/saveCustomCard', function(req, res){
   var configArray = ahoraParser.generateConnectionString(req);
   let coleccion = configArray[1].query.Coleccion;
   let vista = configArray[1].query.Vista;
   let html = configArray[1].query.HTML;
   let query = `UPDATE Objetos_Propiedades_Empleados_Card SET Vista='${vista}',
HTML='${html}' WHERE Coleccion='${coleccion}'`

   new sql.ConnectionPool(configDB).connect().then(pool => {
      return pool.request().query(ahoraParser.queryWithPermission(query));
   }).then(result => {
      res.send({'msg':'ok'});
   }).catch(err=>{
      console.log(err);
   });
});
```

La funció realitza una actualització sobre la base de dades. Per a aquest objectiu, fa una cridada POST, i passa la *query* SQL que va a ser executada i els paràmetres específics per tal de portar-la a terme correctament.

Atén les peticions de la part frontend, comunica amb la base de dades i en retorna la informació. En la part frontend es realitzen cridades Ajax i s'especifica la funció de backend que deu invocar-se.

```
ajaxCall("/addfolder", {"query": {"dataname": value, "idnodo": idnodo}}, function (res) {
   var ahoraMessage = new AhoraMessage();
   ahoraMessage.showMessage("ok", "Carpeta añadida con éxito");
   var event = new CustomEvent('reloadEvt', {bubbles: true, composed: true, 'detail':
params});
   params.ctx.dispatchEvent(event);
});
```

7. Desenvolupament àgil

Encara que la enginyeria software no implica que el desenvolupament que se'n faça del producte siga àgil, és un concepte molt important en l'àmbit d'aquesta i de qualsevol desenvolupament de software en l'actualitat.

Els processos de desenvolupament de software basats en especificar per complet els requisits, i després dissenyar, implementar i provar el sistema, no acostumen a constituir un desenvolupament ràpid de software. A mesura que els requisits canvien, o es descobreixen problemes, els requisits, el disseny i la implementació del sistema tenen que ser reelaborats i provats de nou.

Les metodologies àgils són mètodes de desenvolupament incremental on els increments són mínims. A més, cada dos o tres setmanes es creen noves versions que s'alliberen als clients per tal de que hi haja retroalimentació per la seua part. És a dir, es basen en la participació activa dels clients per a poder actuar de forma ràpida al canvi de necessitats i d'especificació de requisits.

La filosofia darrere dels mètodes àgils està reflexat al manifest àgil, que varen acordar alguns dels líders d'aquestes metodologies com Kent Beck o Martin Fowler, i que exposa principis com ara: entregar software freqüentment que funcione, treballar conjuntament la gent del negoci amb els desenvolupadors, buscar la excel·lència técnica i de disseny, etc.

Són moltes les metodologies que han sorgit per a desenvolupar una aplicació de forma àgil. Entre les metodologies més utilitzades actualment podem destacar, Scrum, XP (*Extreme Programming*) o Kanban. Moltes ferramentes de software permeten que els desenvolupadors puguen fer ús d'aquestes metodologies de forma adequada i han estat utilitzades en aquest projecte per tal que el desenvolupament en fos àgil.

En aquest capítol es presenta les metodologies utilitzades en el desenvolupament d'aquest projecte, les característiques i principis empleats durant el procés i per últim, es mostrarà el flux de treball amb la exposició de captures del seguiment àgil.

7.1. Metodologia

Aquest projecte ha estat desenvolupat integrant principis de la metodologia de desenvolupament àgil Scum, Kanban i característiques de XP (Extreme Programming).

La metodologia àgil Scrum es caracteritza per crear una sèrie de iteracions de longitud fixa anomenades *sprints* que aporten als equips un marc per llançar software regularment. Cada final de iteració representa una fita que porta a una sensació de progrés i permet realitzar una bona estimació i una bona retroalimentació de les proves. Aquesta metodologia s'ha utilitzat en aquest desenvolupament definint una sèrie de quatre *sprints* que contenen les diferents unitats de treball que constitueixen el desenvolupament del nostre projecte. Al final de cada *sprint* s'allibera una versió que permet mostrar el treball realitzat, fer proves del sistema i reaccionar de forma ràpida front a canvis en els requisits del software.

El mètode Kanban es basa en representar visualment sobre un tauler kanban les unitats de treball, el qual permet a l'equip veure l'estat de cadascuna en qualsevol moment. Aquest mètode s'ha integrat al projecte fent ús d'un tauler kanban.

Extreme Programming és una metodologia que emfatitza la satisfacció del client i basa el desenvolupament en oferir el software en petites iteracions que fan que siga fàcil reaccionar a canvis en els requisits per part dels clients. És a dir, entrega el sistema als clients l'abans possible e implementen els canvis que se suggereixen. Aquesta metodologia compta amb característiques com ara la integració continua del sistema, programació en parella, o fer ús de proves d'acceptació.

Al projecte s'han posat de manifest principis del manifest àgil com ara el lliurament freqüent de software que funcione ja que els *sprints* estaven plantejats de manera que cadascun es centrés en requisits que tenien en comú una funcionalitat genèrica del sistema. És a dir, es creen nodes o mòduls de l'aplicació i en cada *sprint* s'ha desenvolupat un d'aquests mòduls en conjunt per tal que el client pogués veure aquest desenvolupament. A més, els clients també formen part en la producció del sistema ja que poden adoptar el sistema abans que estiga completament desenvolupat i sol·licitar canvis als requisits software.

També, és important el ritme de treball en el desenvolupament del projecte i per aquest motiu, s'ha adoptat la metodologia de desenvolupament Scrum, ja que la introducció de *sprints* en el desenvolupament software obliga a l'equip a complir uns termes de temps que porten a que el treball estiga sotmès a un control del ritme del temps.

Cal destacar que l'atenció a la excel·lència tècnica és un valor que es posa de manifest al desenvolupament àgil i aquesta s'aconsegueix gràcies a donar importància

al codi que es desenvolupa mitjançant normes de codi net i fer ús de refactoritzacions, així com plantejar el disseny i l'estructura de components del sistema.

A continuació, detallarem les característiques que han estat utilitzades de cadascuna de les metodologies que s'han descrit anteriorment.

Pel que a la metodologia Kanban es refereix, al nostre projecte s'ha dissenyat un tauler kanban que estava format pel backlog, les tasques a fer al *sprint* actual, les unitats de treball que s'estaven realitzant i les que estaven sent provades. Les unitats de treball, passen del backlog, on estan totes les implicades al projecte que seran passades al *sprint* corresponent, i en aquest *sprint* es passen per les columes del kanban de tasques a fer, tasques en desenvolupament i tasques en prova. En cas que una tasca no complica les proves d'acceptació passa a la columna de desenvolupament i sinó a tasques acabades. Aquesta metodologia permet veure de forma resumida el transcurs del nostre *sprint*. Per a implementar aquesta metodologia s'ha utilitzat la ferramenta software Trello.

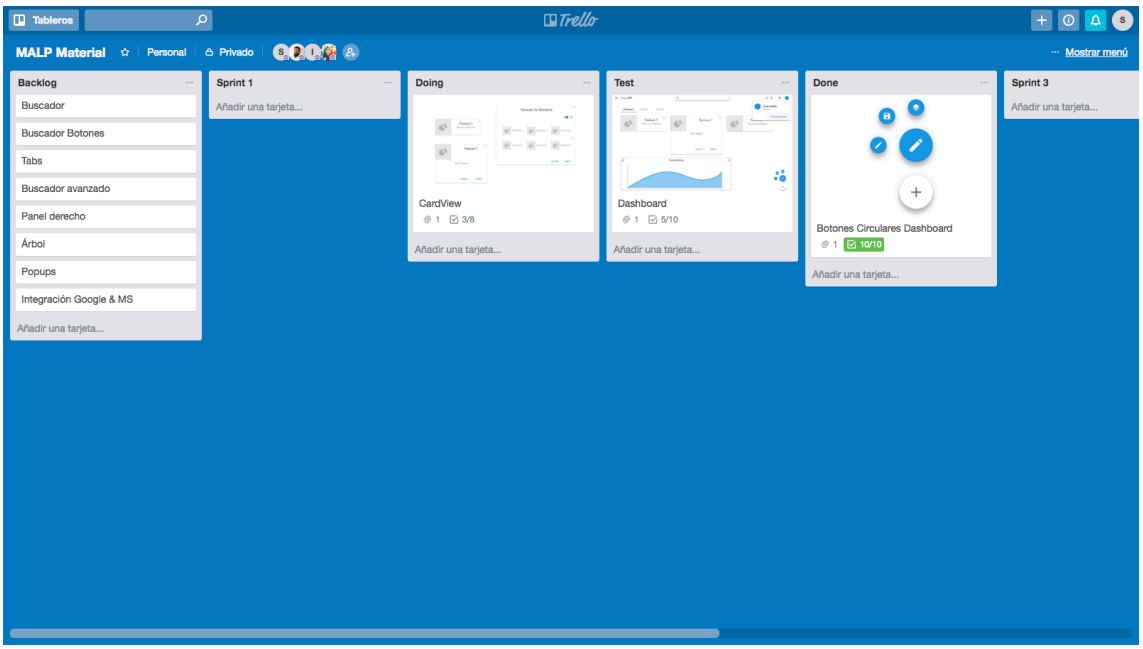

Figura 7.1 Captura del tauler *kanban* durant el primer *sprint*

Com podem observar a la imatge, les tasques que possiblement entren als *sprints* es col·loquen al *backlog*, aquelles que entren al *sprint* es passen a la columna pertinent del *sprint* actual i cada vegada que es comencen es passen a tasques que s'estan realitzant. Cada tasca posseeix una llista de *checks* que conformen les proves d'acceptació d'aquestes tasques. Una vegada es finalitza aquesta tasca i s'haja fet *check* a tots els elements de la llista, es passa a la columna de prova. En la columna de prova, altre desenvolupador s'encarrega de provar que tots els elements de la llista estan correctament implementats i aleshores la passa a feta. En cas contrari tornaria a passar-la a tasques que s'estan realitzant i desmarcaria allò que no és correcte.

La metodologia Scrum també ha estat aplicada en el desenvolupament, mitjançant la creació de quatre *sprints*. Aquests *sprints*, tenien una duració fixa, una setmana, en la que es desenvolupaven, es provaven i s'integraven les unitats de treball realitzades del projecte. En el desenvolupament del projecte s'ha utilitzat la ferramenta software TUNE-UP Process per tal de introduir les unitats de treball del projecte i realitzar un control de temps dels sprints. Les unitats de treball es troben al *backlog* i es passen al *sprint* corresponent, i durant el *sprint* cada desenvolupador s'encarrega de realitzar la unitat de treball que té assignada i comptabilitzar el temps que es troba realitzant la tasca de desenvolupament.

Aquesta ferramenta permet reflectir el treball realitzat per cada desenvolupador, el temps individual de cada membre, el temps empleat en cada *sprint*, i permet visualitzar gràfiques de *burndown* o treball pendent, i diagrama de flux acumulat pròpies d'aquest tipus de metodologia. Aquest apartat gràfic ajuda a analitzar el resultat en el context de l'esforç i per tant, realitzar canvis en els següents *sprints*.

A més per a cada unitat de treball, permet assignar a les diferents etapes per les que passa a diferents membres de l'equip. Amb aquesta característica es pot controlar que tasca deu fer cadascú.

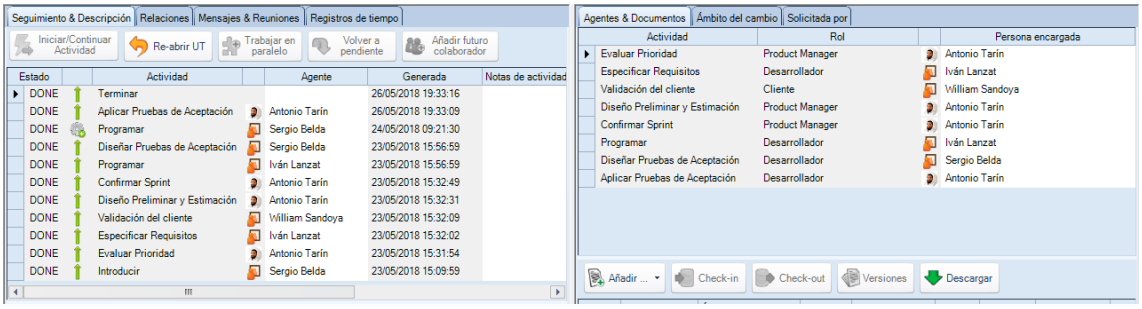

Figura 7.2 Captura d'una unitat de treball a TUNE-UP Process

Com podem observar, la unitat de treball passa per diferents etapes, introduir, especificar i definir requisits, validar la especificació i el disseny de prototips, i finalment confirmar el pas al *sprint* on es comptabilitza el temps de programació i es pot comparar amb el temps estimat.

Pel que fa a la metodologia XP ha estat present al projecte portant a terme algunes de les propietats que la caracteritzen. Durant el desenvolupament del producte s'han realitzat tècniques com ara la programació en parella, que es basa en escriure codi en parella compartint una única màquina i porta a millorar el codi escrit. També s'ha realitzat un procés de refactorització durant el desenvolupament que porta a evaluar constantment el codi, eliminar codi duplicat, i millorar el disseny del sistema. A més, s'ha integrat un estàndard de programació, el ECMAScript 6 i s'ha decidit de quina forma es documentava el codi i s'introduïen el *bugs* o *issues*.

Per últim, al principi de cada *sprint*, per a cada unitat de treball del *backlog* que anava al *sprint* s'han definit proves d'acceptació. Les proves d'acceptació són característiques funcionals que deu complir cada unitat de treball per considerar que aquesta ha estat desenvolupada correctament. En aquest cas s'utilitzava la ferramenta Trello per definir les proves d'acceptació per a cascuna de les unitats de treball i un dels membres que no havia participat a la implementació d'aquesta unitat s'encarregava de portar a terme l'avaluació de la correcció.

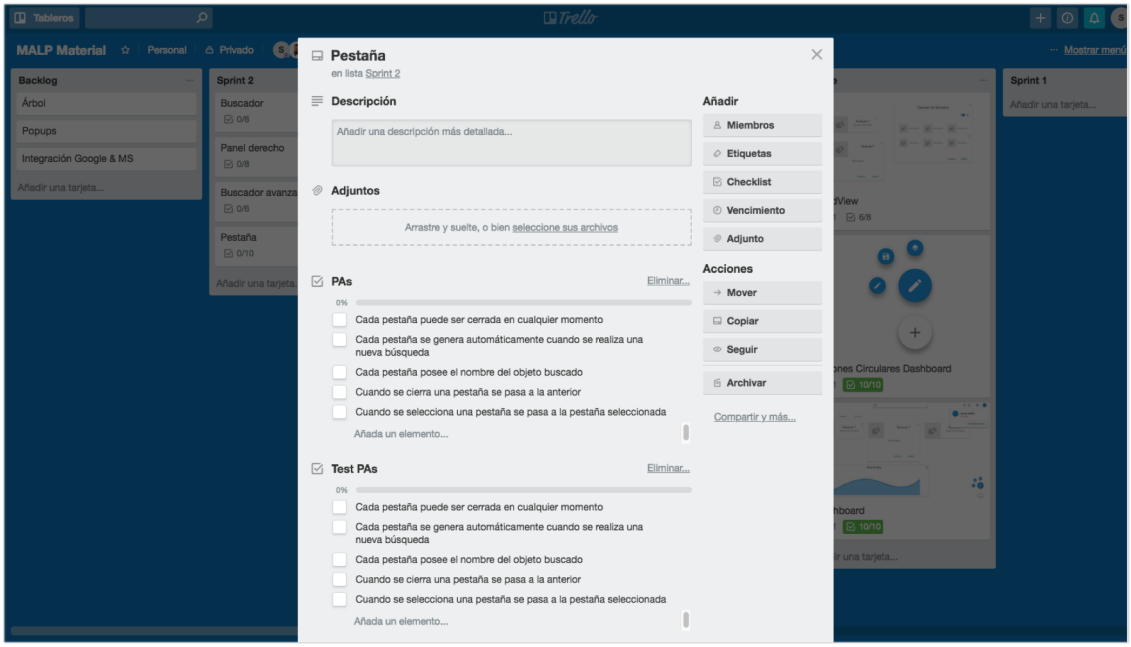

Figura 7.3 Captura de les proves d'acceptació d'una unitat de treball

7.2. Flux de treball

La implementació del sistema s'ha definit en una sèrie de *sprints*, una de les característiques distintives de Scrum. Per tal de portar a terme un seguiment àgil del procés de desenvolupament es pot optar per fer un seguiment del tauler *kanban*. El tauler *kanban* permet visualitzar en què activitat es troba cada unitat de treball i podem conèixer el WIP[1] de cada activitat. Si volem conèixer la tendència del WIP de les activitats del tauler necessitem col·leccionar captures amb les dades. Un diagrama de flux acumulat permet conèixer la tendència del WIP durant el *sprint*.

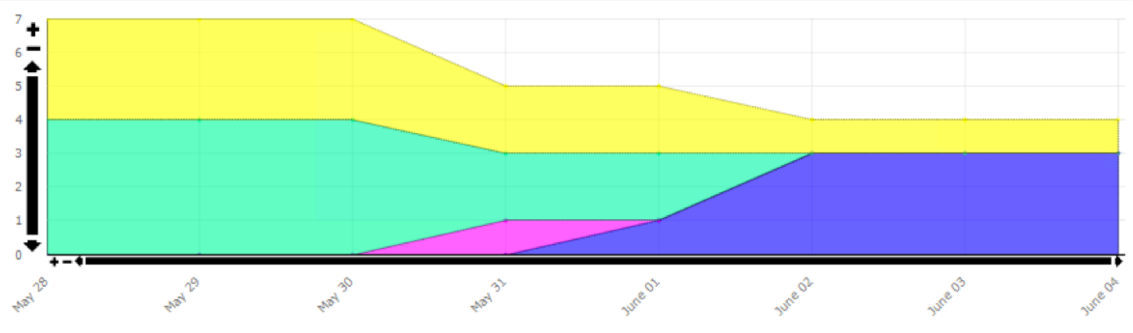

Figura 7.4 Diagrama de flux acumulat del *sprint* 2

A aquest diagrama de flux acumulat podem observar com al principi apareixen les unitats de treball que estan sent programades i de les quals estan dissenyant-se les proves d'acceptació, franja groga i verda respectivament. Passat un temps una unitat de treball passa a aplicar proves d'acceptació, passa un dia i es passada a acabada. Les dues unitats de treball restants acaben després i s'apliquen les proves d'acceptació el mateix dia que es passen a acabar. Aquest últim fet és recomanable ja que resulta preferible avaluar una unitat de treball el mateix moment en que es passa a aplicar proves. El WIP de les unitats de treball en cada activitat es pot extraure com el nombre d'unitats de treball, eix d'ordenades, que hi ha en cada moment del període del *sprint*, eix d'abscisses. Gràcies a aquest diagrama ens permet avaluar si es preferible limitar el WIP per a les següents iteracions o *sprints* i conèixer en que activitats es troben més temps les unitats de treball. La idea de limitar el WIP sembla natural per tal que una columna no acumule massa treball i perjudique el flux. Com que aquesta gràfica en particular està tenint en compte el període del *sprint*, no podem extraure el *Lead Time*, que és el temps que tarda una unitat de treball en ser introduïda al *backlog* fins que finalitza. El que podem calcular aproximadament és el *Cycle Time* que és el temps que passa des de que la unitat de treball es agafada del *backlog* i passa al flux de treball fins que finalitza. En aquest diagrama podem observar que és d'aproximadament 5

[1] WIP (*Work In Progress*): Nombre d'elements de treball o unitats de treball que en un moment es troben en una activitat el flux de treball representat a un tauler kanban.

dies, ja que en el moment en que es passa del *backlog* al *sprint* es quan es dona inici al *sprint*.

Aquests tipus de gràfiques permeten realitzar un seguiment àgil convencional, però, en cas que es requerisca d'un seguiment més precís, es pot optar per les gràfiques *burndown* o de treball pendent. La gràfica *burndown* il·lustra l'esforç restant comparat amb l'esforç abordable al *sprint*.

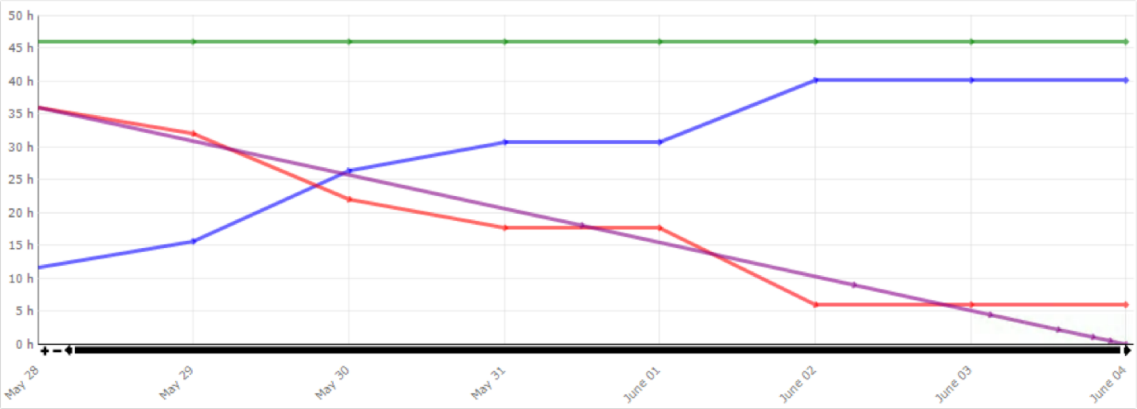

Figura 7.5 Gràfica *burndown* del *sprint* 2

En aquesta gràfica *burndown*, podem observar l'esforç restant o temps aproximat que queda per finalitzar el *sprint* amb la línia roja. Aquesta línia roja va disminuint en funció del temps que es comptabilitza per a cada unitat de treball que s'està portant a terme. La línia morada, indica un esforç restant de referència, considerant una velocitat de projecte constant. La línia roja és recomanable que estiga per davall de la línia morada, per tal que el temps o esforç restant vaja acord a la línia de referència i que per tant, el temps invertit siga el més constant fins el final.

La línia blava és un complement anomenat gràfica *burnup*, i permet conèixer l'esforç invertit en cada unitat de temps. Aquesta gràfica permet conèixer quin és l'esforç i temps que hi queda per a cada unitat de temps. Cal destacar que per a que aquesta gràfica s'ajuste correctament a la realitat cal que l'estimació que se'n faça de les unitats de treball siga acurada i per tant, es podrà tenir un control adequat del que queda aproximadament.

Al disposar a una mateixa gràfica els esforços invertits, estimats i restants, es facilita la interpretació de l'estat de la iteració i com s'ha anat desenvolupant.

A aquesta gràfica poden ocórrer variacions a causa de canvis durant el transcurs del *sprint*, que invaliden la interpretació de les dades. Per exemple, l'esforç restant pot veure's invalidat per una activitat no estimada o una unitat de treball que s'ha subestimat i per tant, s'ha sobrepassat el temps dedicat a aquesta. També poden haver variacions sobre l'esforç restant observat, degut a canvis en l'esforç invertit

durant una unitat de treball, un ajust a una estimació, o una introducció d'una estimació faltant. A més introduir una unitat de treball al *sprint* actual pot també provocar una variació sobre l'esforç restant.

Tot i això, aquesta gràfica suscita el debat de si realment entraria en el anomenat entorn àgil de desenvolupament. Si cal invertir esforç per tal de il·lustrar aquesta gràfica no resulta un treball àgil, ja que es deu realitzar un treball addicional. En cas que s'automatitze com es fa gràcies a la ferramenta TUNE-UP Process, pot resultar de gran ajuda degut al seguiment detallat que ofereix. Per aquest motiu, és l'equip qui deu decidir si és convenient portar a terme un seguiment detallat i registrar el procés de treball, tal com suggereix la metodologia Kanban, així com fer ús de *sprints*, i per a aquesta decisió deu de tenir-se en compte el context del producte o servei per tal de conèixer si ho requereix.

Al nostre sistema, pot resultar d'ajuda, per veure en que parts del producte es dedica més esforç i apareixen més dificultats de treball. La introducció d'aquest tipus de gràfiques ajuda a tenir un seguiment més detallat del treball que es porta a terme i a millorar a cada *sprint*.

8. Manteniment

L'etapa de manteniment és una de les etapes més importants dintre del cicle de vida del software ja que constitueix gran part del seu temps de vida. Bé siga perquè es realitza un manteniment adaptatiu o perfectiu, un producte software requereix constantment d'un manteniment per no transformar-se inevitablement en un sistema obsolet. És a dir, un producte necessita més o menys un creixement continu amb l'objectiu de mantenir la satisfacció per part del client. De fet, aquest projecte no deixa de ser un treball de reenginyeria que forma part del manteniment del sistema ERP original i que per tant es porta a terme amb la intenció de reconstruir un sistema en una nova forma i implementant un nou codi.

El manteniment s'integra durant el cicle de desenvolupament d'un sistema també i en aquest cas s'han portat a terme diferents activitats com el disseny de proves.

8.1. Proves

Les proves són un factor crític per a determinar la qualitat del software. La prova de software busca realitzar una avaluació de diferents aspectes del mateix. Existeixen diferents tipus de proves dintre del procés de prova com ara proves unitàries, proves funcionals o de integració, del sistema i d'acceptació.

A continuació, es mostren exemples de proves unitàries definides per al codi del nostre projecte, tant per a la part de *backend* implementada amb Node.JS com per a la part dels components web de Polymer.

Pel que fa al codi de Node.JS, les proves unitàries s'han definit utilitzant la llibreria Mocha i Chai. Mocha és un *framework* que permet realitzar de forma sencilla proves asíncrones. A l'executar els test, permet la presentació d'informes flexibles i precisos. Chai és un llibreria de assercions BDD[1]/TDD[2] que pot ser utilitzat junt a altre *framework* de testing de Javascript.

[1] BDD (*Behaviour Driven Development*): Procés que consisteix en dissenyar proves abans que el codi estiga escrit, que verificaran que el comportament del codi és correcte des del punt de vista de negoci. Prova comportament i no la implementació de codi.
[2] TDD (*Test Driven Development*): Procés consistent en definir proves que fallen, que una vegada estiga acabada la implementació del codi seran satisfactòries. D'aquesta manera es dissenya el mínim funcionament amb l'objectiu de que passen les proves.

```
it("Obtener propiedades de objeto", function(done) {
   chai.request(lh).post("/DamePropiedadesObjeto").type("json")
      .send({"query": {"objeto": "Clientes", "params": ""}})
      .end(function(error, response) {
         assert.equal(response.statusCode, 200);

         let res = JSON.parse(response.text);

         for (var i = 0; i < res.length; i++) {
            assert.notEqual(res[i].Propiedad, null);
            assert.notEqual(res[i].Descrip, null);
            assert.notEqual(res[i].Orden, null);
            assert.notEqual(res[i].IdTipo, null);
         }

         done();
      });
});
```

Per exemple, a la nostra base de dades tenim una taula on es defineixen les propietats de cada objecte o recurs del nostre sistema, per a un client seria un nom, telèfon, etc. Al mòdul de Node.JS de la part de *backend* es defineix una cridada que s'anomena "DamePropiedadesObjeto". A aquesta cridada es passa l'objecte que en aquest cas és un client. El que comprova aquesta prova és que per a totes les propietats d'un objecte, tenim diferents atributs com la descripció o el nom de la propietat, i que per tant, no són *null*.

El propi entorn de desenvolupament ens permet llançar les proves i observar el resultat.

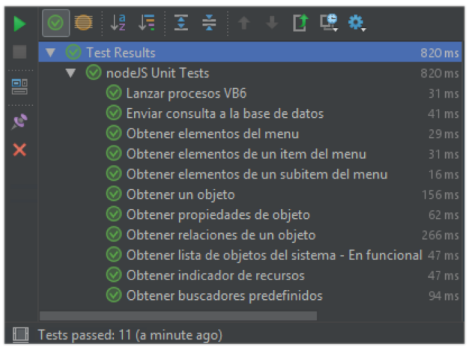

Figura 8.1 Resultat de l'execució de proves unitaries d'alguns mètodes del backend

Polymer també permet fer proves unitàries per als seus components, mitjançant la ferramenta de proves anomenada *Web Component Tester*. *Web Component Tester* utilitza llibreries com Mocha, Chai i Selenium.

Per exemple, al nostre sistema apareixen distints tipus de finestres de diàleg en funció de l'acció que s'està portant a terme. Per aquest fet s'ha definit un diàleg genèric per al nostre sistema que depenent del tipus de diàleg i gràcies a les propietats i observadors dels *web components*, represente el diàleg de diferents formes, ocultant

i mostrant els diferents elements que conformen cada tipus de diàleg com ara botons per a elegir entre diferents opcions, llistes d'elements, etc. Les proves unitàries permeten provar els mètodes de la lògica del component que fan possible aquest comportament.

```
<script>
  suite("ahora-dialog", function() {

  test("Input dialog", function() {
    var ahoraDialog = fixture("AhoraDialog");
    document.body.appendChild(ahoraDialog);
    ahoraDialog.open();
    ahoraDialog.setType("input");
    var listView = ahoraDialog.$.listview;
    var viewPager = ahoraDialog.$.viewpager;
    var input = ahoraDialog.$.input;
    ahoraDialog.setInputValue("Value of input textfield");
    ahoraDialog.setInputLabel("Label of input textfield");
    ahoraDialog.setTextBody("Dialog textbody");
    assert.equal(listView.style.display, 'none');
    assert.equal(viewPager.style.display, 'none');
    assert.equal(input.style.display, 'flex');
    assert.equal(input.value, "Value of input textfield");
    assert.equal(input.label, "Label of input textfield");
    assert.equal(document.body.querySelector("ahora-
dialog").shadowRoot.querySelector("#textbody").innerHTML, "Dialog textbody");
  });
```

El que es fa a aquesta prova és crear un component de tipus "ahora-dialog", que és afegit al cos de la nostra pàgina de proves, s'obre, i es defineix com a tipus de diàleg, diàleg d'entrada de dades (*input*). Aquest diàleg deu tenir un component que és el camp d'entrada de text, *input*, i no deu constar d'altres components propis d'altres tipus de diàlegs com ara el *listView* o el *viewPager*. Els mètodes de "setInputLabel()", "setInputValue()" i "setTextBody()" també són provats, donant valors per a aquests elements i provant que són correctes.

Figura 8.2 Proves per al component ahora-dialog.html

Quan accedim a la direcció URL del nostre test, s'executen els test i sobre la consola del nostre navegador apareixen els resultats.

9. Resultat

El desenvolupament d'un sistema acostuma a ser un procés iteratiu e incremental i el que constitueix el cicle de desenvolupament és el que s'ha descrit al llarg d'aquest document. El cicle de vida d'un producte software és un període extens que exigeix que es porte a terme un manteniment del sistema si no es vol que es convertisca finalment en un sistema obsolet, ja que és l'única manera de mantenir la satisfacció del client. Una vegada portades a terme les etapes que constitueixen l'enginyeria de software i el procés de desenvolupament d'un producte, s'obté un producte mínim viable (MVP), el qual pot ser utilitzat per clients per tal de provar les característiques funcionals i el sistema des del context tècnic per tal de detectar possibles errors i comunicar-ho als desenvolupadors.

A continuació es presenten les captures del sistema desenvolupat. Es destacaran els requisits funcionals desenvolupats i s'exposaran de forma il·lustrativa.

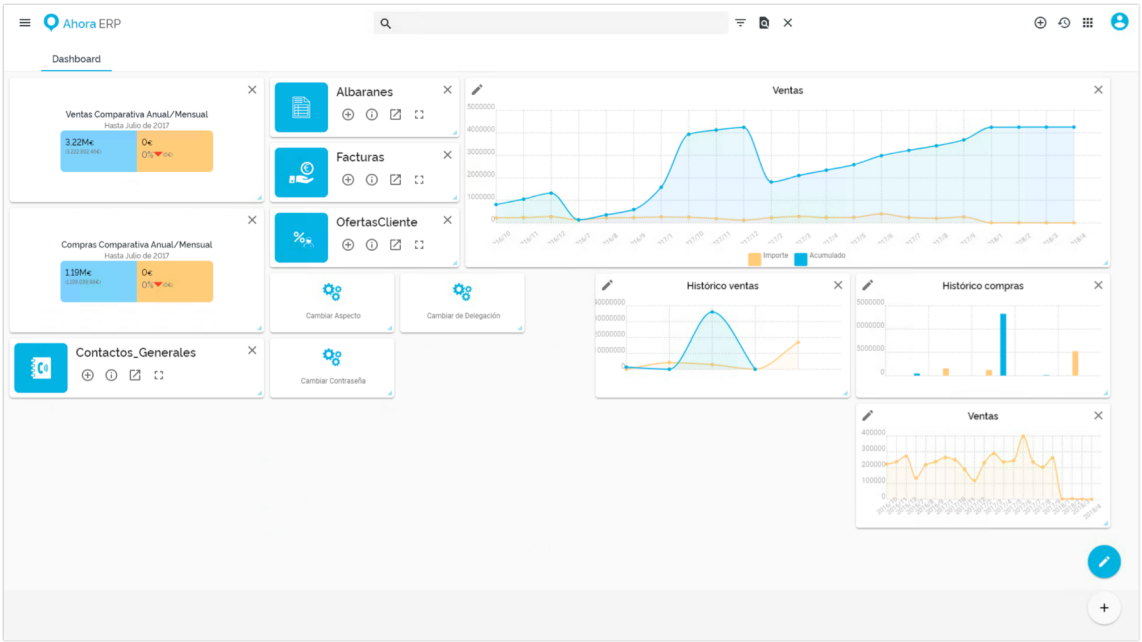

Figura 9.1 *Dashboard* o finestra principal

En aquesta imatge podem observar la finestra principal on es mostra un *dashboard* o panell de components on tenim visió sobre estadístiques e indicadors gràfics sobre recursos i rendiment. Aquestes gràfiques es generen en base a un conjunt de dades que provenen de la base de dades. També podem observar targetes sobre cerques que es poden realitzar al sistema. Les targetes poden ser redimensionades i poden ficar-se en qualsevol posició del panel principal.

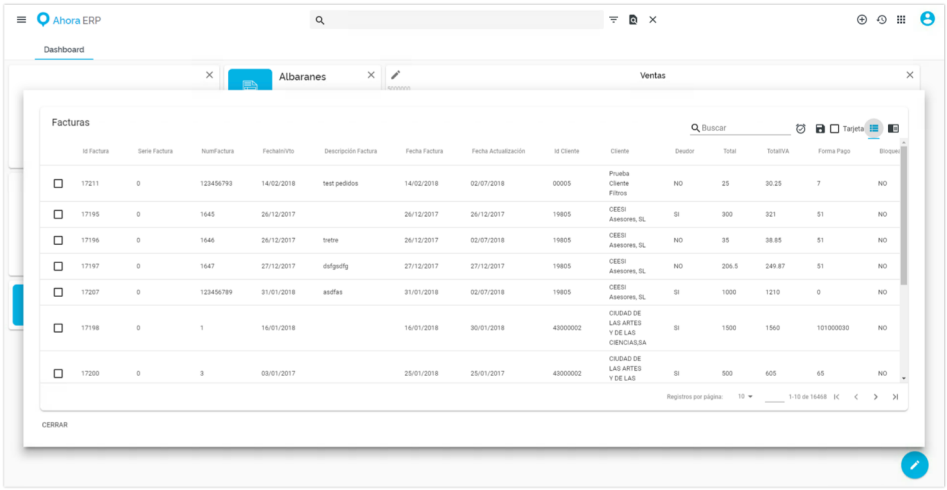

Figura 9.2 Visió detallada d'una targeta de cerca

El *dashboard* també pot ser guardat amb la qual cosa podem tenir diferents configuracions estructurals de components en funció de les necessitats en cada moment. Els *dashboard* guardats poden ser consultats en qualsevol moment. Així com si es vol afegir components al nostre *dashboard* existeix un panel a la part dreta on apareixen possibles indicadors que poden ser seleccionats. Totes aquestes accions poden ser llançades gràcies als *floating action button* que hi ha a la part inferior.

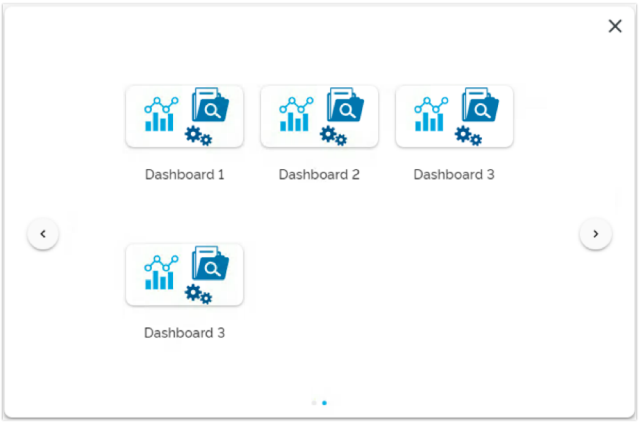

Figura 9.3 *Dashboard* guardats per l'usuari

A la part superior disposem de la barra de cerca en la que podem elegir el diferent recurs del sistema a ser cercat i especificar el nom de l'objecte cercat. A la part dreta disposem de filtres per a la cerca i d'un botó per a llançar la cerca avançada.

Figura 9.4 Desplegable de cerca

En el moment en que es realitza la cerca, es crea una nova pestanya. En aquesta pestanya figura una taula amb les coincidències de la cerca. Cada entrada de la taula pot ser seleccionada. Quan es selecciona un objecte s'obre un menú contextual on figuren les diferents opcions que es poden fer amb aquest objecte, com enviar a favorits, enviar la informació per correu o obtenir més informació de l'objecte.

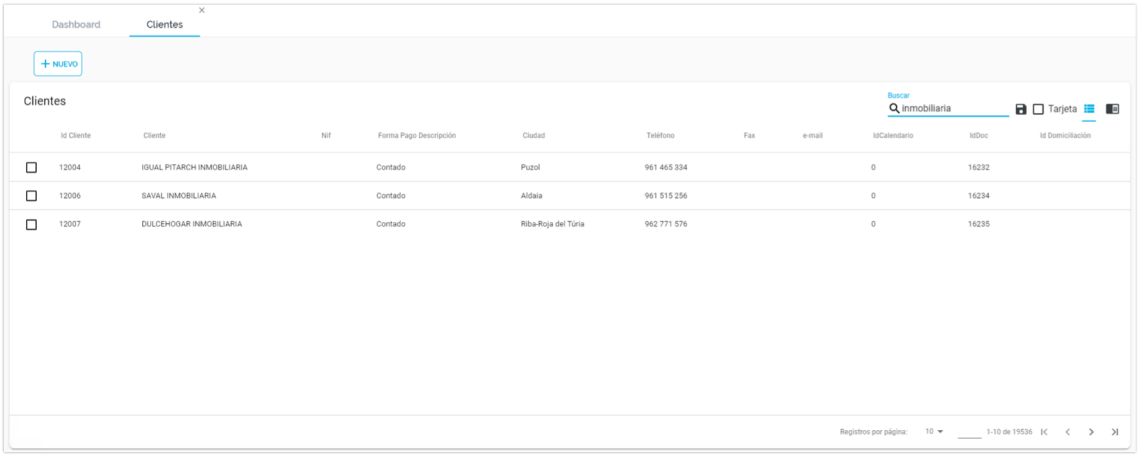

Figura 9.5 Pestanya de cerca de recursos

Com podem observar a la imatge, apareixen diferents objectes de tipus client a la taula després de cercar la paraula "inmobiliaria" als clients de la nostra taula. Una vegada se selecciona un objecte, apareixen una sèrie de icones dalt de la taula per poder realitzar accions com obrir l'objecte, crear un nou objecte d'aquest tipus, imprimir, etc. La icona de guardar de dalt de la taula serveix per guardar la cerca actual i enviar-la al menú desplegable de la part dreta del *dashboard* i que podrà ser afegit a aquest per poder tenir-la sempre accessible.

Quan un dels elements es arrastrat fins on està la icona superior esquerra, s'obri un menú que disposa de totes les relacions de l'objecte actual de la taula amb recursos empresarials de la nostra empresa. Es tracta d'un arbre de relacions o graf que permet veure les relacions d'un objecte amb altres. Aquest arbre pot ser posat al costat de la

taula per poder ser visualitzat constantment i continuar utilitzant el sistema sense que es tanque.

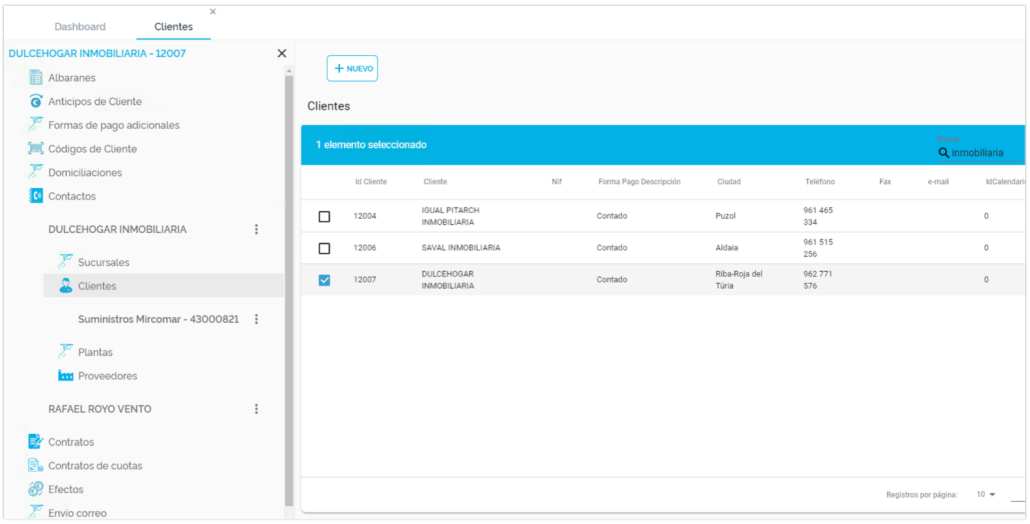

Figura 9.6 Arbre o graf d'un objecte

A la banda esquerre tenim un deslegable amb seccions d'accés a formularis de operacions que es poden realitzar sobre el sistema. Tots aquests formularis són els del anterior ERP, ja que a aquest desenvolupament s'ha centrat en desenvolupar tota la lògica de presentació del ERP i més endavant s'adaptarien tots els formularis a la nova tecnologia. Per tant, quan es fa click a un dels elements es llança una finestra de Visual Basic 6 del anterior ERP. Aquesta acció es fa gràcies a la tecnologia CEF integrada al sistema, i que ajuda a que l'antiga tecnologia puga ser incrustada en la nova tecnologia.

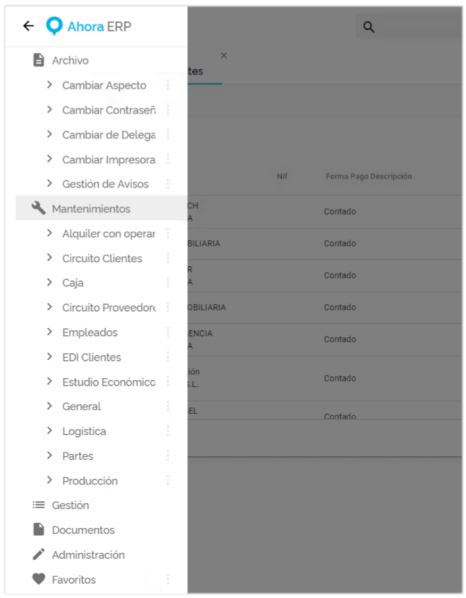

Figura 9.7 Menú lateral esquerre d'accés a formularis

A la banda dreta del *dashboard*, tenim un menú desplegable que es llança amb el botó circular d'afegir amb tots els components que poden ser afegits a la finestra principal. En el moment en que se seleccionen apareixen com a targetes sobre aquesta.

Figura 9.8 Menú lateral dret del *dashboard*

La cerca avançada es un desplegable on es pot escollir el objecte a cercar i les propietats que s'especifiquen a la cerca. Per a cada propietat es poden definir els valors que adoptaran.

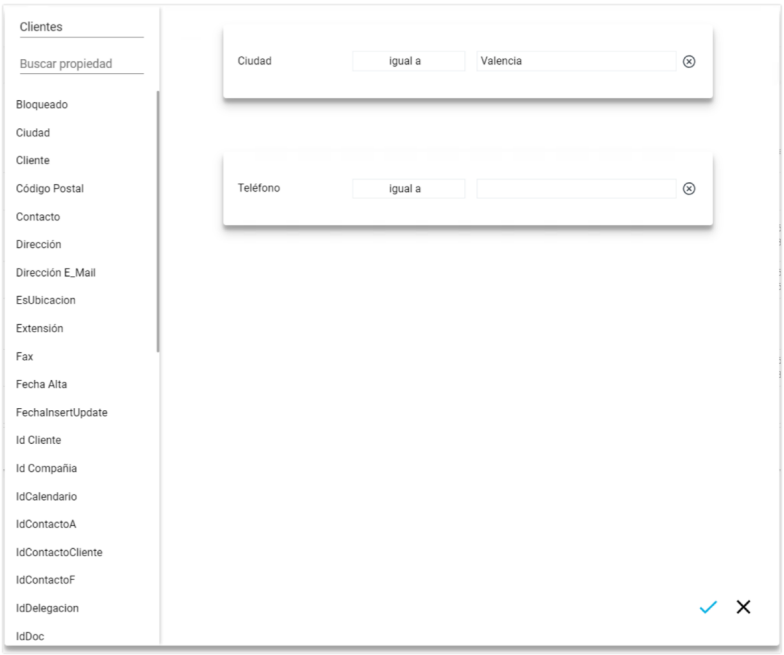

Figura 9.9 Cerca avançada

Per últim, a la banda superior dreta disposem de tres seccions, nou objecte, històric i barra d'accés. Històric mostra un desplegable amb els últims formularis consultats per l'usuari i la barra d'accés aquells que l'usuari ha enviat a aquesta per tenir-los sempre a l'abast.

Finalment, es mostra una captura del sistema integrat a CEF per ser executat a un escriptori del sistema operatiu Windows junt amb la crida de formularis de l'antic software que s'aprofiten a aquest sistema.

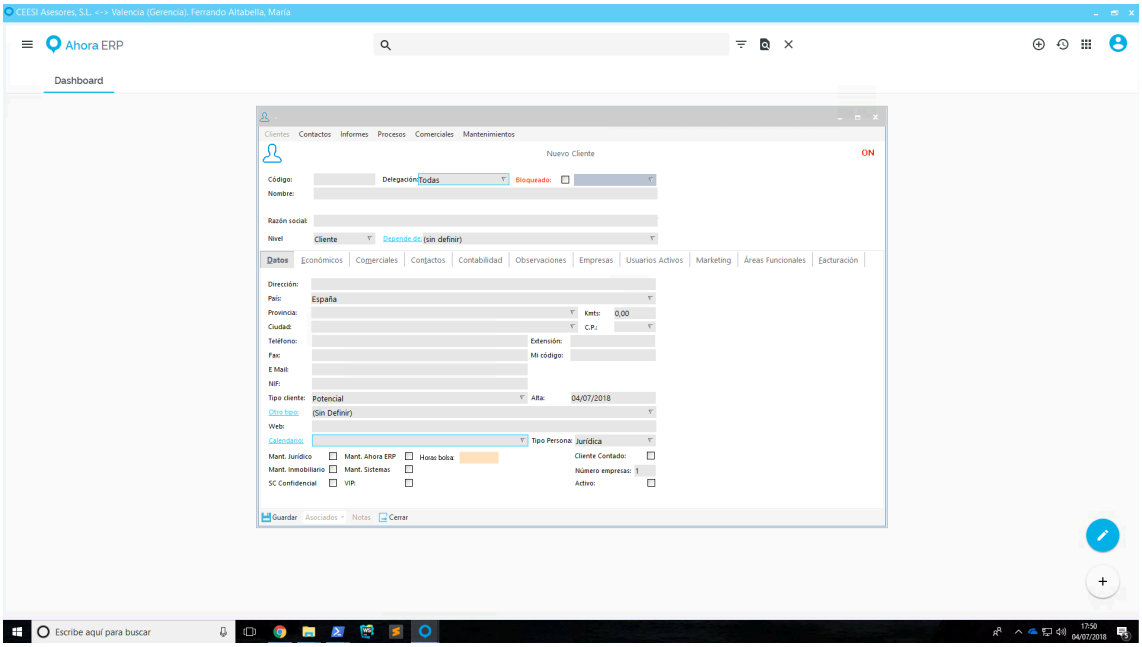

Figura 9.10 Integració de formularis Visual Basic 6 amb el nou desenvolupament

10. Limitacions

Aquest document ha mostrat un desenvolupament de migració o reenginyeria d'un sistema que estava desenvolupat amb una tecnologia en desús i que per tant, forma part d'una modernització de la plataforma. Tot i això, el desenvolupament de tot el sistema baix tecnologies modernes precisa de molt de temps. Aleshores, aquest desenvolupament ha format part de la primera etapa de migració de tot el sistema ERP. Per tant, hi ha elements que no han estat migrats a noves tecnologies com poden ser alguns dels formularis Visual Basic 6 que es llancen a l'aplicació. Per aquest motiu, actualment aquest sistema es limita a un ús a dispositius d'escriptori i no compleix amb el requisit no funcional de la portabilitat. No obstant, aquest procés es fa pensant en que finalment tots aquests elements seran adaptats, i per tant, es complirà finalment aquest requisit gràcies a la tecnologia escollida. Per altra banda, la tecnologia CEF utilitzada al sistema ha ajudat a poder integrar aquest desenvolupament a dispositius d'escriptori i a fer cridades a l'antiga tecnologia per tal de que ambdues convisquen.

Altra limitació o dificultat per al projecte ha estat la estructura de la base de dades. Aquesta, presenta una estructura arcaica i seguix encara una lògica que es va definir per a l'antic sistema. Per aquest motiu, les cridades de *backend* s'han realitzat tenint en compte aquestes limitacions i en perspectiva a una reformulació futura d'aquesta estructura.

11. Conclusions

Com a punt final a aquest document podem destacar una sèrie de conclusions que es poden extraure del procés de treball portat a terme.

Pel que fa a la tecnologia, resulta confortant utilitzar noves tecnologies a projectes de desenvolupament. Per una banda, podem destacar el coneixement que es deriva del seu ús, ja que comporta no sol un aprenentatge d'una tecnologia en particular sinó que va més enllà amb l'anàlisi de diverses tècniques i entorns de treball i la investigació que se'n fa. Per altra banda, podem destacar les virtuts que utilitzar noves tecnologies suposa. Migrar un sistema amb plataformes actuals proveeix d'avantatges que ofereixen les noves tècniques al sistema concret, millorant la satisfacció del client incorporant noves característiques pròpies de la tecnologia. Aquest fet produeix que els desenvolupadors augmenten els seus coneixements i els apliquen a futurs projectes.

Des del punt de vista empresarial, portar a terme un procés de reenginyeria ajuda a modernitzar o refrescar una plataforma que va quedant obsoleta amb el pas del temps degut a que el client percep que altres productes del mateix àmbit compleixen característiques que l'actual sistema no fa, i estan a un nivell tècnic superior. A més a més, començar un procés de migració, incita a que no sols un producte d'una empresa siga millorat sinó que obre les portes a que es modernitze tota una sèrie de productes que ofereix l'empresa, i tenint en compte l'aprenentatge de noves tecnologies, poden ser aplicades en aquests projectes i augmentar la qualitat tècnica dels productes.

Des de la perspectiva de l'enginyeria de software, resulta interessant la forma en la que els conceptes teòrics poden ser portats a la pràctica, de manera que, duent a terme les distintes etapes s'aplegue a la conclusió de que introduir un procés sistematitzat i organitzat ajuda al desenvolupament d'un sistema. És a dir, seguir una sèrie de processos establerts porta a produir un sistema de major qualitat tècnica i que es realitza de forma organitzativa. Introduir conceptes de la metodologia àgil porta a comprovar de que forma ajuden a que un projecte evolucione i es constituïsca de forma correcta front a metodologies tradicionals que no ofereixen aquestes garanties. D'aquesta manera es comprova com els processos i principis de la metodologia àgil ajuden a un equip a detectar les seues fortaleses i debilitats i de quina forma s'impliquen en el desenvolupament.

En resum, aquest projecte ha ajudat a donar una visió de com les noves tecnologies ajuden a millorar les característiques funcionals d'un producte, com impliquen una millora en els sistemes d'una organització i de quina manera l'aplicació de l'enginyeria software ajuda al desenvolupament segur d'un projecte software.

11.1. Relació del treball desenvolupat amb els estudis cursats

En aquest treball s'han posat de manifest coneixements que s'han adquirit amb els estudis cursats. A aquest document s'ha exposat el desenvolupament d'un producte software, al context del món laboral. S'han especificat les etapes per les que s'ha passat a aquest desenvolupament i el tipus de metodologia de treball adoptada, i que constitueixen part del coneixement rebut a l'àmbit de l'enginyeria software estudiada durant el grau.

Més concretament, pel que fa a les etapes de l'enginyeria software, podem destacar el coneixement adquirit a l'assignatura Enginyeria Software (ISW); els apartats de anàlisi i enginyeria de requisits podem relacionar-los amb l'aprenentatge a l'assignatura Anàlisi i Especificació de Requisits (AER); el disseny arquitectònic del projecte amb l'assignatura Disseny de Software (DDS); la utilització de característiques i ferramentes de les metodologies àgils amb Procés de Software (PSW); el disseny de prototips de interfície i anàlisi d'experiència d'usuari amb Interfícies Persona Computador (IPC); l'etapa de manteniment i els requisits no funcionals de qualitat amb Qualitat de Software (CSO). El desenvolupament d'un projecte software es pot relacionar amb l'aprenentatge a la matèria Projecte d'Enginyeria Software (PIN).

Per altra banda, també s'han mostrat tecnologies que s'aprenen als estudis o que estan relaciones per la familiaritat de la seua naturalesa. Tecnologies com SQL que es mostra a l'assignatura Bases de Dades (BDA), o Node.JS a l'assignatura Tecnologies de Sistemes de Informació en la Xarxa (TSR).

En conclusió, s'han aplicat coneixements relacionats amb l'àmbit de la programació i el software a aquest treball, així com tecnologies actuals relacionades amb el context del desenvolupament web.

12. Bibliografia

Sommerville, I., Olguín, V. C., & Velázquez, S. F. (2011). *Ingeniería de software*. Madrid: Pearson Educación de México.

Robert C. Martin. (2012). *Código Limpio*. Madrid: Ediciones Anaya Multimedia.

"Stack Overflow Developer Survey 2018". 2018. https://insights.stackoverflow.com/survey/2018/

"1ª edición del Estudio de SoftDoit sobre El uso de software ERP en España (2017) | SoftDOit". 2017. https://www.softwaredoit.es/estudio/primer-estudio-softdoit-uso-software-erp-espana-2017.html

"*Quiénes somos - Odoo*". https://www.odoo.com/es_ES/page/about-us

"ERP Cloud | Soluciones ERP basadas en la nube | SAP". https://www.sap.com/spain/products/erp/erp-cloud.html

"Polymer library - Polymer Project". 2016. https://www.polymer-project.org/2.0/docs/devguide/feature-overview

"¿Qué es un sistema ERP y para qué sirve?". https://www.ticportal.es/temas/enterprise-resource-planning/que-es-sistema-erp

"Introduction - webcomponents.org". https://www.webcomponents.org/introduction

"*JavaScript | MDN*". 2018. https://developer.mozilla.org/es/docs/Web/JavaScript

"*JavaScript language resources - JavaScript | MDN*". 2018. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources

"*Tutorial: Intro To React – React*". https://reactjs.org/tutorial/tutorial.html#what-is-react

"*AngularJS: Developer Guide: Developer Guide*". https://docs.angularjs.org/guide

"*Introduction — Vue.js*". https://vuejs.org/v2/guide

"*Custom Elements v1: Componentes web reutilizables*". https://developers.google.com/web/fundamentals/web-components/customelements

"*Chromiumembedded / cef — Bitbucket*". https://bitbucket.org/chromiumembedded/cef

"*Especificación de Requisitos según el estándar de IEEE 830*". 2008. *https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf*

Letelier Torres, Patricio. "Actividad: Tablero kanban + concepto de WIP + Diagramas de Flujo Acumulado". *Agility at work*. 1 de desembre de 2012. http://agilismoatwork.blogspot.com.es/2012/12/actividad-tablero-kanban-concepto-de.html

Letelier Torres, Patricio. "Gráficas Burndown: ¿cómo rentabilizar el esfuerzo asociado a su elaboración?". *Agility at work*. 3 de desembre de 2011. http://agilismoatwork.blogspot.com/2011/11/graficas-burndown-como-rentabilizar-el.html

"*Principis darrere el manifest àgil*". 2001. http://agilemanifesto.org/iso/ca/principles.html

"*The Home of Scrum*". https://www.scrum.org/

"*Extreme Programming: A Gentle Introduction*". 2013. http://www.extremeprogramming.org/

"*Scrum | Una breve introducción | Atlassian - El orientador ágil*". https://es.atlassian.com/agile/scrum

"*Kanban - El orientador ágil*". https://es.atlassian.com/agile/kanban

"Chai". http://www.chaijs.com/

"Mocha - the fun, simple, flexible JavaScript test framework". https://mochajs.org/