Universitat Autònoma de Barcelona

escola d'enginyeria

5735-1: Anàlisi i Disseny eCommerce

Productes De Casa, venta de productes de la Agricultura Catalana

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Efraín Felipe Motta Ávila
i dirigit per
Marc Talló
Bellaterra,15 de Setembre de 2014

El sotasignat, **Marc Talló**

professor/a de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en/na **Efraín Felipe Motta Ávila**

I per a que consti firma la present.

Signat: .....

Bellaterra, 15 de Setembre de 2014

FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Títol del projecte: *5735-1: Anàlisis i Desenvolupament eCommerce, Productes De Casa, venta de productes de la Agricultura Catalana*	
Autor[a]: *Efraín Felipe Motta Ávila*	**Data:** *9/2014*
Tutor[a]/s[es]: *Marc Talló*	
Titulació: *Llicenciatura en Enginyeria Informàtica*	
Paraules clau (mínim 3) • ***Català:*** Comerç electrònic, Aplicació web, PHP, MySQL, Anàlisi Requeriments, Estructura de Servidors, cakePHP, Framework, MVC. • ***Castellà:*** Comercio Electrónico, Aplicación Web, PHP, MySQL, Análisis de requerimientos, Estructura de servidores, cakePHP, Framework, MVC. • ***Anglès:*** Electronic commerce, Web Application, PHP, MySQL, Requirements Analysis, Server Structure, cakePHP, Framework, MVC.	
Resum del projecte *(extensió màxima 100 paraules)* • ***Català:*** *El projecte que és trobarà a continuació consisteix en l'anàlisi i desenvolupament d'una aplicació web orientada al comerç electrònic per un client que distribueix productes relacionats amb l'agricultura catalana. Aquest client ens planteja una idea respecte al model de negoci, i ens demana la realització tant de la part de Programació, com la part de documentació necessària per que pugui ser fàcil de mantenir en futures versions del software i tindre un bon punt de partida per a futures versions.* • ***Castellà:*** *El proyecto que se encontrará a continuación consiste en el análisis y desarrollo de una Aplicación web orientada para el comercio electrónico para un cliente que distribuye productos relacionados con la agricultura catalana. Este cliente nos plantea una idea respecto al modelo de negocio, i nos pide la realización tanto de la parte de programación como de la parte de documentación necesaria para que pueda ser fácil de mantener y partir de una buena base para futuras versiones del software.* • ***Anglès*****:** *In The following project you will find the analysis and development of a web Application eCommerce Oriented. This application has been made following the specifications of a client that present us, an idea about, how he thinks the business model should be. The client requests us to produce the programming part as documentation part necessary, in order to having a good starting point for future versions of software.*	

Aquest projecte està dedicat a la meva família i sobre tot a la meva parella, Adriana, sense tu, res d'això no hagués estat possible.

Aquesta pàgina s'ha deixat intencionadament en blanc

ÍNDEX

Capítol 1. INTRODUCCIÓ

Amb el naixement d'Internet i les noves tecnologies ha aparegut un nou horitzó immens, ple d'oportunitats, que configura un nou i remodelat entorn de mercat caracteritzat per la creixent competitivitat, cicles accelerats de vida dels productes i els ràpids –i dràstics- canvis tecnològics. Podríem assegurar amb plena certesa que es tracta de tota una revolució. Qui no coneix algú que no hagi viscut l'experiència de vendre o comprar productes/serveis via Internet?.

Tota una nova industria activa i en ple creixement amb un públic exigent, canviant i sempre buscant el que vol de la manera més ràpida, econòmica i amb la millor qualitat possible. Un públic que a la vegada, amb la seva demanda, genera una gran competència i que per tant, incrementa la dificultat de l'assumpte. És per això que a les empreses, neix la necessitat de professionals que siguin capaços d'analitzar, desenvolupar, implantar i gestionar aquests tipus de plataformes electròniques de venta electrònica, amb l'objectiu d'utilitzar-lo com unitats estratègiques de negoci.

Com podem veure, el mon del "eCommerce" ens planteja tot un repte no pas amb fàcil solució. Repte que aquest projecte vol assumir, estudiant les diferents plataformes i eines de desenvolupament. Valorant els "pros" i "contres" que ens aporten, per tal d'implantar i gestionar una aplicació web amb les característiques bàsiques, però ben estructurades, per una tenda de venta de productes online.

En aquest primer capítol vostè podrà fer-se una primera visió global del projecte, on és parlaran dels objectius i motivacions principals pel desenvolupament del mateix. També és farà una petita introducció sobre del què s'està fent en referència al comerç electrònic, amb les necessitats més comuns i demandades pel mercat actual, arribant al final a una solució que s'anirà desenvolupant al llarg de la durada d'aquest projecte.

1.1. PRESENTACIÓ

Es té un client que ens presenta una idea per la venda dels productes amb els quals comercia. El nostre client es dedica a comercialitzar productes relacionats amb la agricultura catalana amb marquès molt especifiques per un públic exclusiu i selecte. Vol una plataforma web que li permeti publicar els seus productes a internet. Plataforma de la qual, sobretot ens remarca la idea, de que no es vol convertir en el típic "eCommerce" convencional. Definint un factor diferenciador, que consisteix en la generació d'un tipus de contingut que a la seva vegada serveixi per promocionar els productes. Aquest contingut consisteix en, per exemple, per aquells productes de tipus alimentació, publicació de receptes les quals, dins dels seus ingredients, facin referència a productes que es pugin comprar dintre d'aquesta plataforma web. Tot i així, els productes han de poder-se emmagatzemar en una web amb les funcionalitats que la tenda virtual més bàsica pugui tenir.

El client també expressa, que no té cap pressa amb lo que respecta a la sortida del producte, i vol com a primera versió les funcionalitats més bàsiques i amb la millor qualitat possible. Ens comenta que per l'experiència prefereix partir d'un producte de fàcil manteniment, ja que per futures millores sigui més àgil la seva implantació.

Per tant la primera versió, la demana amb les funcionalitats convencionals i molt bàsiques de tendes online (p.e Categorització de productes, una fitxa descriptiva per cada producte i que sigui accessible des de la web, productes relacionats, etc..) i a més un lloc en la web per tal d'emmagatzemar receptes que van relacionades a les fitxes del productes. Demana que se li doni la prioritat al desenvolupament en plataformes mòbil primer. No demana cap aplicació mòbil, però que la web sigui adaptable a plataformes amb pantalles més petites amb elevades limitacions de rendiment.

Com podem veure, aquesta idea té un objectiu molt clar, però a la vegada, per tal de escollir la manera en como es desenvoluparà, serà necessari un estudi de les possibilitats que es tenim valorant totes les implicacions que ens pugui plantejar aquest tipus de projecte. Aquest apartat pretén donar una primera visió genèrica que la idea que es demana desenvolupar per tal de contextualitzar els següents apartats i entendré el perquè d'algunes decisions es puguin prendre durant la llargada del projecte.

1.2. MOTIVACIONS

Durant aquests anys de formació, he après diferents maneres de trobar solucions a problemes i reptes, que s'han anant plantejant en diverses assignatures de la carrera. Paral·lelament, he tingut l'oportunitat de treballar en diferents empreses relacionades amb el desenvolupament de software, prenent diferents rols en equips de treball, amb necessitats específiques segons la naturalesa del projecte on es treballava. Situació que sempre m'ha portat a qüestionar-me: on està el nexe entre el que après i el món real, o en altres paraules, com en puc treure profit del coneixement que tinc per tal de ser capaç desenvolupar-me com a Informàtic en el món laboral.

Per tant, la meva motivació principal és la de extreure profit dels coneixements obtinguts a la carrera, posant-los en pràctica en un cas pràctic, amb necessitats reals com las que pot plantejar una aplicació web de tipus "eCommerce", creant una solució senzilla però robusta i que al final, complementi la meva formació posant en pràctica les diferents habilitats que he pogut desenvolupar durant aquests últims anys d'estudi.

1.3. OBJECTIUS

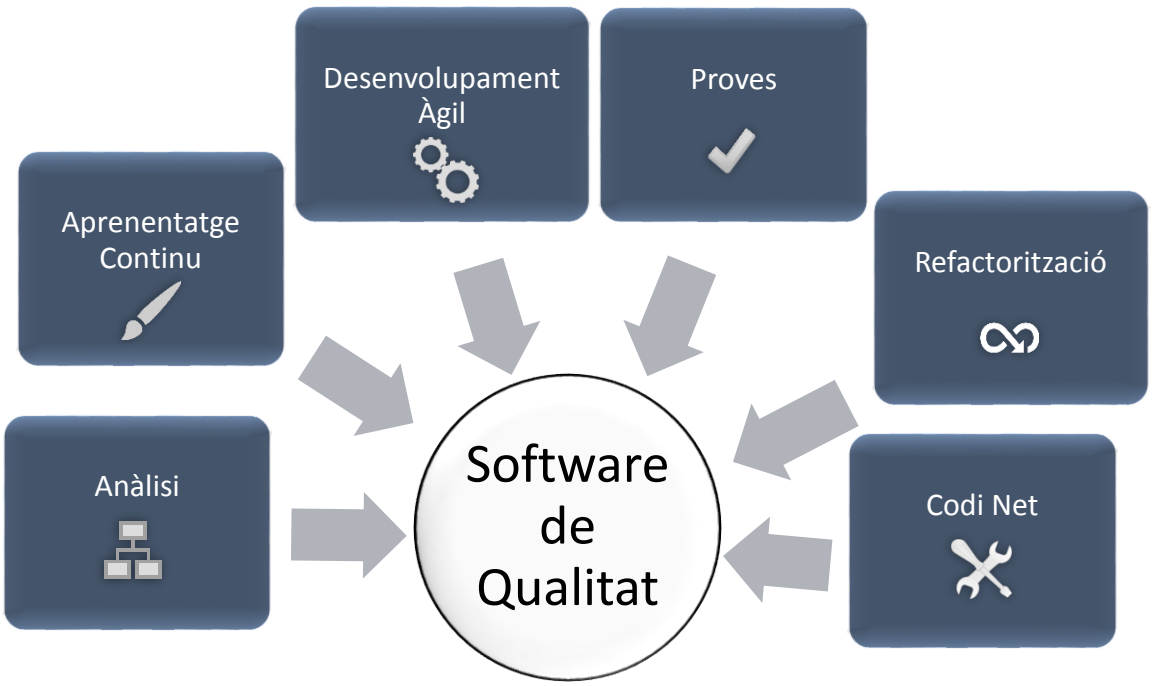
1.3.1. Objectius Generals

Aquest projecte té com a principal objectiu el anàlisis i disseny d'una aplicació web seguint les especificacions que el client ens ha donat. Fent un seguiment amb tota la documentació, per a que aquestes siguin útils pel manteniment i millora el producte en futures versions. En primer lloc, es vol analitzar solucions ja desenvolupades i ofertes al mercat actual, tals com Magento, Prestashop, Shopify, ... Etc. Veient quines avantatges i inconvenients ens aportaria la utilització de les mateixes.

En segon lloc, s'analitzarà també la possibilitat de fer una aplicació web des de zero, per que, com veurem al capítol II, tot i que moltes de les plataformes existents porten funcionalitats força complertes, moltes vegades en funció de les necessitats del client pot convindré o no seleccionar aquesta possibilitat.

Finalment, sigui quina sigui l'opció escollida, és vol entregar una aplicació de qualitat. Com sabem, la industria del desenvolupament del software està experimentant un gran canvi. Que per l'experiència, s'ha posat en evidència en el desenvolupament del software, que no tot és radicà en acabar el treball de qualsevol manera, sinó que és imprescindible que el producte final entregat tingui la màxima qualitat possible. D'aquesta manera aconseguim major satisfacció per part dels clients. Entregant a la vegada, productes que siguin molt més fàcils de mantenir, el que finalment es tradueix, en una millor rendibilitat a llarg termini.

Per això aquest projecte s'aferrarà a l'ús de l'enginyeria del software, amb les bones pràctiques del costat que són les que al final, com a Enginyers Informàtics, haurem de ser capaços d'utilitzar en el en el nostre favor i no en contra. Per què el bon ús d'aquestes pràctiques, serà el que marcarà la diferència en la feina que realitzem a la nostra carrera professional.



Il·lustració 1.1 Pilars del Software de Qualitat

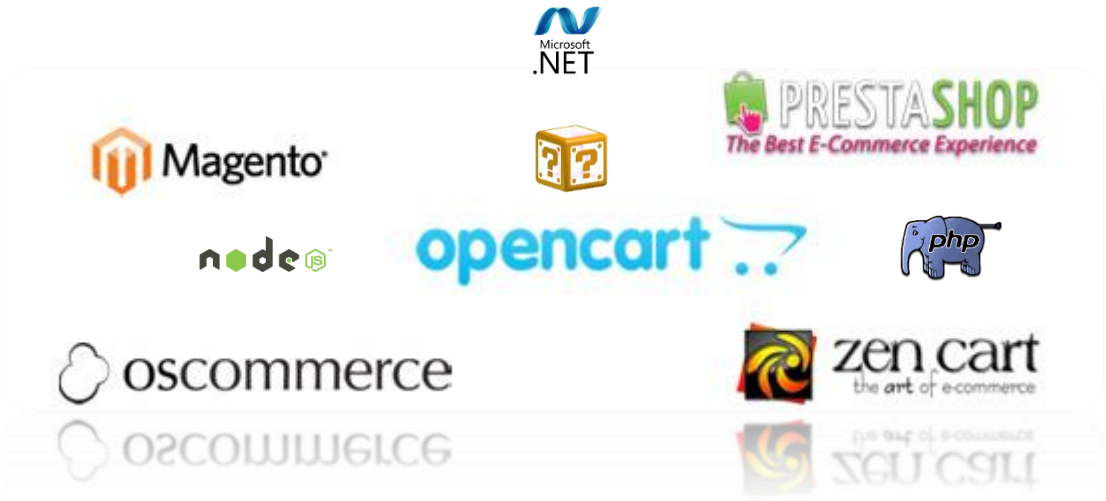
1.3.2. Objectius Detallats

- ✓ Recol·lecció Informació de les necessitats del client.
- ✓ Definició i Anàlisi Requeriments (RQ).
- ✓ Fer un estudi de viabilitat del projecte per tal seleccionar la manera de fer en funció de les possibilitats que tenim disponibles, analitzant per tant:
  - Riscos i fases del projecte.
  - Estudi de viabilitat econòmica.
- ✓ Definició Entorn desenvolupament de l'Aplicació en funció de les especificacions.
- ✓ Definició Tasques a realitzar, en funció de les especificacions, tenint en compte el temps de durada del projecte → Planificació del projecte.
- ✓ Estimació Tasques a realitzar durant el projecte en funció de tres Paràmetres:
  - Complexitat
  - Esforç
  - Incertesa
- ✓ Disseny d'una prova de concepte de l'Aplicació amb el seguiment de bones practiques, en la mida del possible, i com és evident en la mida que ens aporti valor al producte final.

1.4. ESTAT DE L'ART

Escollir una plataforma eCommerce és una de les decisions més importants que s'ha de prendre quan es vol implantar una botiga online. La plataforma correcta ha de potenciar el nostre negoci i no condicionar-lo. Per tant, ha de proveir les eines necessàries per executar el nostre negoci de la millor manera possible sent capaç de fer-se càrrec de tot el material tècnic desordenat per si mateix.

Arribats a aquest punt veiem necessari fer un estudi complert, sobre les plataformes més utilitzades i demandades per la necessitat del mercat actual. Aquesta part serà explicada amb més detall en el capítol 2, on debatrem els pros i contres d'algunes d'aquestes plataformes.



Il·lustració 1.2 Plataformes eCommerce i tecnologies software

1.5. Estructura de la Memòria

L'estructura d'aquesta memòria s'ha dissenyat en funció de l'estàndard establert per la Universitat Autònoma de Barcelona i amb el seguiment el tutor del projecte. A continuació podrà veure el desglossament de capítols de la memòria d'aquest Projecte.

CAPÍTOL 1 INTRODUCCIÓ. Conté la visió general del projecte i quines han estat les motivacions, així com els objectius generals i detallats del mateix.

CAPÍTOL 2 ESTUDI DE VIABILITAT. S'avaluen les diferents maneres que tenim disponibles al nostre abast seleccionat la que més ens convé en funció de les necessites i especificacions del projecte, deixant planificat la feina a desenvolupar estimada aproximadament tenint en compte el temps del qual es disposa.

CAPÍTOL 3 FONAMENTS TEÒRICS. És fa una descripció teòrica dels coneixements necessaris per l'estructuració i desenvolupament d'aquest projecte.

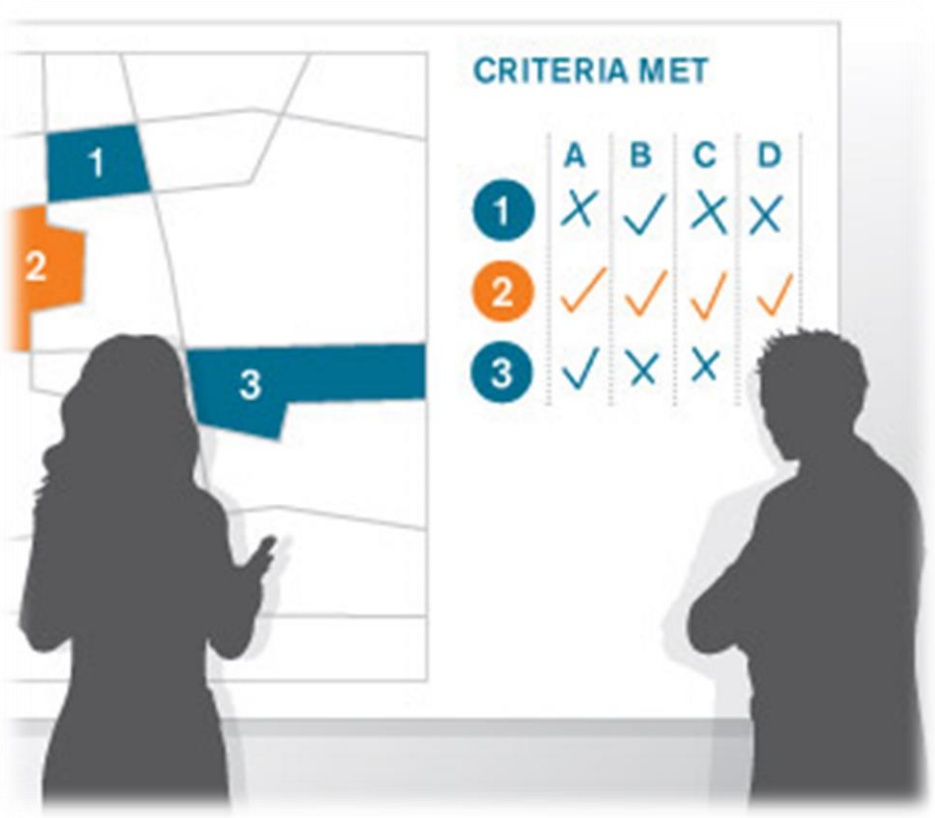
CAPÍTOL 4 ANÀLISI. En aquest capítol passarem a l'acció. Es detallarà i documentarà com s'implementarà pas a pas l'aplicació.

CAPÍTOL 5 IMPLEMENTACIÓ. S'explicarà el resultat final del projecte, parlant de les implicacions i problemes que hagin pogut sortir mentre s'ha dur a terme la implementació.

CAPÍTOL 6 PROVES. Es definiran uns escenaris on l'aplicació demostrarà el que és capaç de fer seguint les especificacions definides al capítol 2.

CAPÍTOL 7 CONCLUSIONS. En aquest capítol s'avaluaran els resultats obtinguts del projecte, tals com: objectius assolits, intencions de futur, valoració personal. També es revisarà la planificació inicial del projecte afegint les possibles modificacions que hagin pogut aparèixer durant el projecte.

CAPÍTOL 8 BIBLIOGRAFIA. Si es vol consultar les fonts d'informació que aquest projecte ha utilitzant, en aquest capítol es farà una llista detallada.



Capítol 2. ESTUDI DE VIABILITAT

En aquest capítol s'estudia la viabilitat del projecte per tal de determinar si es possible el seu desenvolupament dins d'una sèrie de limitacions com les que poden ser, per exemple, econòmiques, temporals i/o tecnològiques.

Començarem per la definició dels objectius d'aquest capítol, passant per l'estudi de l'art detallat. Arribant finalment a la selecció de la plataforma que s'utilitzarà per la realització del projecte. Finalment es definiran les Especificacions de l'aplicació, tan funcionals com no funcionals i tècniques.

Aquest capítol també deixarà definida una primera estimació aproximada, amb la seva planificació associada, de les tasques a realitzar durant la durada tinguda, fent una valoració i Anàlisi de riscos que en planteja aquest projecte.

2.1. OBJECTIUS

Com hem pogut veure, el comès d'aquest capítol no és més que el de definir una primera línia del projecte. Analitzarem quines necessitats i limitacions ens planteja, en funció de les especificacions definides pel "client", escollint finalment l'opció que més s'acosti a les característiques i funcionalitats necessàries per aquest projecte.

Per tant al final d'aquest capítol veurem definit les següents directives:

- ☐ Especificacions
- ☐ Planificació
- ☐ Valoració i Anàlisi de Riscos
- ☐ Conclusions

2.2. ESPECIFICACIONS

2.2.1. Introducció

Tenim un client, comerciant de productes de la agricultura catalana. Aquest client planteja la necessitat d'una plataforma de comerç online on pugi posar a la venta els productes amb que treballa, ja que considera que d'aquesta manera podrà arribar a un numero de compradors més grans dels que té. Aquest apartat estructurarà les necessitats plantejades pel client i definiran una guia de treball per la part de desenvolupament. En cas de dubte de quines funcionalitats s'han d'implementar o com s'han d'implementar, s'haurà de consultar aquest apartat, ja que ha estat pactat detalladament amb el client.

2.2.1.1. Propòsit

1. El propòsit d'aquest capítol és el de definir una línies de guia per el desenvolupament d'aquest projecte, creant un document pactat amb el client, on es plasmi les necessitats del client, amb l'objectiu d'aclarir qualsevol mal entès d'aquesta manera s'intenta minimitzar els problemes d'interpretació que es puguin ocasionar.
2. Aquest apartat va dirigit al equip de desenvolupament del projecte, coordinador del projecte i client.

2.2.1.2. Personal Involucrat

Nombre	Efraín Felipe Motta Ávila
Rol	Desenvolupador del projecte
Categoria professional	Estudiant Enginyeria Informàtica
Responsabilitats	Documentació del productes Implementació del producte Proves del producte Entrega del producte
informació de contacto	efrainfelipe.motta@e-campus.uab.cat

Nombre	Marc Talló
Rol	Coordinador de Projecte
Categoria professional	Professor Universitat Autònoma de Barcelona
Responsabilitats	Coordinació segons normes de la universitat
informació de contacto	marc.tallo@uab.cat

Nombre	Joan Torrents Pareja
Rol	Client
Categoria professional	Comercial
Responsabilitats	N/A
informació de contacto	jtorrents@champinet.com

2.2.1.3. Resum

Aquest capítol es fragmentarà en les següents parts:

1. Descripció General.
   1.1. Es definirà la perspectiva del producte, funcionalitat principals sense entrar en massa detall, característiques dels usuaris, Restriccions, Suposicions, i Evolució previsible del sistema.
2. Requeriments funcionals.
   2.1. Definició d'accions fonamentals que ha de realitzar el software al rebre informació, processar-la i produir resultats. En aquelles se inclou:
      2.1.1. Comprovació de valides de las entrades.
      2.1.2. Seqüencia exacta d'operacions.
      2.1.3. Resposta a situacions anormals (desbordaments, comunicacions, recuperació de errors).
      2.1.4. Paràmetres.
      2.1.5. Generació de sortides.
      2.1.6. Relaciones entre entrades i sortides (seqüencies d'entrada i sortides, formules per a la conversió de la informació).
      2.1.7. Especificació dels requisits lògics per a la informació que serà emmagatzemada en la base de dades (tipus d'informació, requerida).
3. Requeriments no funcional.
   3.1. Especificacions dels requisits relacionats amb la carrega que s'espera que tingui que suportar el sistema. Per exemple, el número de terminals, el numero esperat d'usuaris simultàniament connectats,, numero de transaccions per segon que deurà suportar el sistema. Etc.
4. Requeriments tècnics.
   4.1. Qualsevol altre requeriment que no encaixi o no apliqui amb cap de les seccions anteriors. Per exemple, Requisits Legals, culturals, polítics. Etc.

2.2.2. Descripció General

Aquest capítol detallarà el software des d'una perspectiva de producte, llistant les normes i limitacions generals.

2.2.2.1. Perspectiva de Producte

Es té un client que vol canvia la manera de gestionar la venda dels productes que distribueix. Amb aquesta plataforma web, necessita emmagatzemar la informació del catàleg de productes que ven, fent que aquesta informació sigui accessible des de internet. I que qualsevol persona, des del seu dispositiu preferint de connexió a internet, sigui capaç d'accedir a una plataforma, que li permeti visualitzar i comprar els productes i contingut que el nostre client vol.

Es tracta d'un producte independent. Es desenvoluparà la base de software necessària d'una plataforma de comerç electrònic.

2.2.2.2. Funcionalitat i abast del Producte

La plataforma de comerç electrònic Productes de Casa suporta les següents funcionalitats de producte:

- Registre d'usuaris.
- Modificació informació del usuaris.
- Inici i Tancament de sessió.
- Navegació definida per:
  - "Home" (Vista benvinguda).
  - Categories de productes (Vista de categoria).
  - Informació de productes (Fitxa Producte).
  - Receptes (Vista totes les receptes).
  - Receptes de cuina (Fitxa de Receptes).
  - Informació Contacte.
- Cercador de productes.
- Carret de la Compra.
- Compra productes del Carret de la Compra via una passarel·la de pagament.
  - Pagament via transferència bancaria.

2.2.2.3. Restriccions Generals

El software de Productes de Casa utilitzarà *nGinx 1.6 o superiors versions* com a servidor web/proxy.

El software de Productes de Casa utilitzarà *MySQL 5.5 o superiors versions*, com a gestor de base de dades.

El software de Productes de casa serà escrit en *PHP5 o versions superiors* basat en el *Framework CakePHP 2.5.3 o versions superiors.*

Quan s'utilitzi qualsevol llibreria de software, el codi font d'aquesta llibreria serà proporcionat pel codi font del sistema.

El software de Productes de casa es deu executar des d'un servidor amb versions *Debian 7 o versions superiors.*

Tot el codi del software estarà versionat en un repositori *Git fast version control*.

2.2.2.4. Evolució Previsible del sistema

Productes de casa Versió 1.0, que es la descrita per aquest document, serà una base amb funcionalitats molt bàsiques. Per tant les possibles evolucions dependran de si el projecte te continuïtat real, despès de la entrega d'aquesta versió, es a dir, un desplegament públic a internet. En aquest apartat es definiran les parts del software que es preveu hauran de complementar-se i son les següents:

- TPV, Més d'una forma de pagament possible.
- Protocols SSL per la passarel·la de pagament.
- Enllaç/vinculació de les comptes d'usuaris a les xarxes socials com Facebook, Twitter o Instagram.
- Gestió automàtica de comandes.

- Importació i actualització automàtica de la informació dels productes / receptes de la tenda.
- Posicionament Cerca (SEO).
- Publicitat via adreça electrònica (Mailing).

2.2.3. Requeriments Funcionals

2.2.3.1. Registre d'usuaris

RQ10
Donat un usuari no registrat, quan un usuari entra a la pàgina web de Productes de Casa, llavors la aplicació web ha de proveir a l'usuari la possibilitat de registrar-se amb un botó "Soc un nou client" .

RQ11
Donat un usuari no registrat, quan pressiona el botó "Soc un nou client", llavors la aplicació web ha de presentar un formulari de registre titulat "Crear Compte" amb els següents camps obligatoris:

- Nom Usuari.
- Correu electrònic.
- Correu electrònic repetit.
- Contrasenya.
- Contrasenya repetida.

Després d'aquests camps la aplicació a de proveir un botó de confirmació de creació de compte "Crear Compte".
Finalment just sota del botó "Crear Compte" ha de aparèixer el següent missatge:

"Creant aquest compte, Estàs d'acord amb les condicions de us i avis de privacitat Productes de Casa"

RQ12
Donat un usuari que està registrant-se a la web, quan es fa clic al botó "Crear Compte", llavors la aplicació web ha de validar que els camps "Correu electrònic" i "Correu electrònic repetit" siguin iguals.

RQ13
Donat un usuari que està registrant-se a la web, quan es fa clic al botó "Crear Compte", llavors la aplicació web ha de validar que els camps "Contrasenya" i "Contrasenya Repetida" siguin iguals en cas contrari no es podrà crear el compte.

RQ14
Donat un usuari que està registrant-se a la web, quan es fa clic al botó "Crear Compte", llavors la aplicació web ha de validar que els camps "Contrasenya" i "Contrasenya Repetida" siguin iguals en cas contrari no es podrà crear el compte.

RQ14
Donat un usuari que està registrant-se a la web, quan introdueix les dades correctament i fa clic al botó "Crear Compte", llavors la aplicació web ha de emmagatzemar les dades del compte i adreçar a la pàgina "Home" amb la sessió iniciada.

2.2.3.2. Edició Dades d'Usuari

RQ21
Donat un usuari registrat a la web, Quan l'usuari te la seva sessió iniciada, la aplicació web haurà de proveir la possibilitat de editar del compte associat a la secció iniciada amb un botó anomenat "Editar el meu compte".

RQ22
Donat un usuari registrat i amb seva sessió iniciada a la web, Quan l'usuari fa clic en el botó "Editar el meu compte", la aplicació ha de adreçar al usuari a un formulari que contindrà tots el camps modificables de l'usuari amb la possibilitat de guardar els canvis o descartar-los.

RQ23
Donat que el formulari d'edició de dades del compte esta mostrat en pantalla, quan l'usuari fa clic en la opció de guardar els canvis, el programa ha de validar que tota la informació omplerta es correcta. Guardarà els canvis, si i solo si, la informació entrada segueix la política de validació de camps.

2.2.3.3. Inici Sessió

RQ31
Donat un usuari no registrat a la web, quan l'usuari vol iniciar sessió, la aplicació ha de proveir la possibilitat d'iniciar sessió amb un botó "Inicia Sessió".

RQ32
Donat un usuari no registrat a la web, quan l'usuari fa clic al botó iniciar sessió, la aplicació presentarà un formulari d'inici de sessió amb els següents camps:

- Correu Electrònic.
- Contrasenya.

RQ33
Donat un usuari no registrat a la web, quan introdueix les seves dades correctes de Inici de sessió i fa clic al boto inicia sessió, llavors la aplicació adreçarà a la home amb la sessions de l'usuari iniciada. En cas contrari avisarà que les dades introduïdes no son dades correctes i no ha de permetre inicia sessió.

2.2.3.4. Tancament de sessió

RQ41
Donat un usuari que té iniciada una sessió, la aplicació web haurà de permetre una opció per tancar sessió amb un botó de tanca sessió.

2.2.3.5. Vistes Navegació web

RQ51

la aplicació web ha de proveir *"breadcrumbs"* associats al lloc que es troba per les següents pàgines:

- Pàgina de categoria
- Fitxa de producte
- Pàgina de receptes
- Fitxa de receptes

2.2.3.5.1. Home

RQ522

Donat que un usuari accedeix a la pàgina web de productes de casa, la primera pàgina que es mostra és la pàgina anomenada home que seguirà el següent format:

- Capçalera amb logotip tenda i menú *("Header").*
- Cos de la pagina *("Container").*
- Mapa web, condicions d'us, Notificació de privacitat i Informació Copyright *("Footer").*

RQ523

Quan el cos de la pagina es presentat a la pantalla, la aplicació web haurà de presentar barres de desplaçament *("sliders")* dels productes de la web per categoria que hagin estan seleccionats per un administrador.

RQ524

Donat un usuari que no ha iniciat sessió, quan la pagina home es presentada, la aplicació web ha de presentar la possibilitat de accedir des del menú a les següents seccions:

- Categories.
- Receptes.
- Inici sessió.
- Registre de nous clients.
- Carret de la compra.
- Informació de contacte.

RQ525

Donat un usuari que ha iniciat sessió, quan la pagina home es presentada, la aplicació web ha de presentar la possibilitat de accedir des del menú a les següents seccions:

- Categories.
- Receptes.
- El meu compte.
- Carret de la compra.
- Informació de contacte.

2.2.3.5.2. Carret Compra

RQ531

La aplicació web haurà de presentar una versió resumida del carret de la compra en el menú.

RQ531
La aplicació web haurà de presentar una versió detallada del carret de la compra accessible des de la versió resumida en el menú.

RQ532
Quan la versió resumida es consultada, la aplicació haurà de presentar el llistat de productes que es tenen emmagatzemats a la tenda amb la següent informació:

- Nom
- Quantitat
- Preu

I finalment presentarà un botó anomenat “Veure Carret”.

RQ533
La aplicació web ha de permetre que en la versió detallada del carret de la compra es puguin veure i editar la informació dels productes que es volen comprar.

RQ534
La aplicació web ha de permetre que en la versió resumida del carret de la compra es puguin veure però no es puguin editar la informació dels productes que es volen comprar.

RQ535
Quan la versió detallada del carret de la compra es consultada, la aplicació web ha de presentar un boto anomenat “Passar per la caixa” que quan sigui accionat, adreçarà a l’usuari als passos de pagament. *(El comportament relacionat amb els passos de compra i pagament està descrit a l’apartat 2.2.3.7)*

2.2.3.5.3. El meu compte

RQ541
Donat que un usuari que no ha iniciat sessió, quan està navegant per la web, la aplicació web de presentar sempre en el menú la opció de veure la informació del meu compte.

2.2.3.5.4. Pàgina de categoria

RQ551
Donat un usuari està navegant per la web, quan accedeix a una categoria per veure el seu contingut, la aplicació web de presentar una pagina amb tots el productes relacionats amb aquesta categoria.

RQ552
Donat un usuari està navegant per la web, quan accedeix a una categoria per veure el seu contingut, la aplicació web de presentar una pagina que contingui una imatge capçalera característica de la categoria que es consulta.

2.2.3.5.5. Fitxa de producte

RQ561
Donat un usuari està navegant per la web, quan accedeix a la informació de un producte, la aplicació ha de proveir una vista amb les següents parts per cada producte visualitzat:

- Foto Producte.
- Titulo del producte.

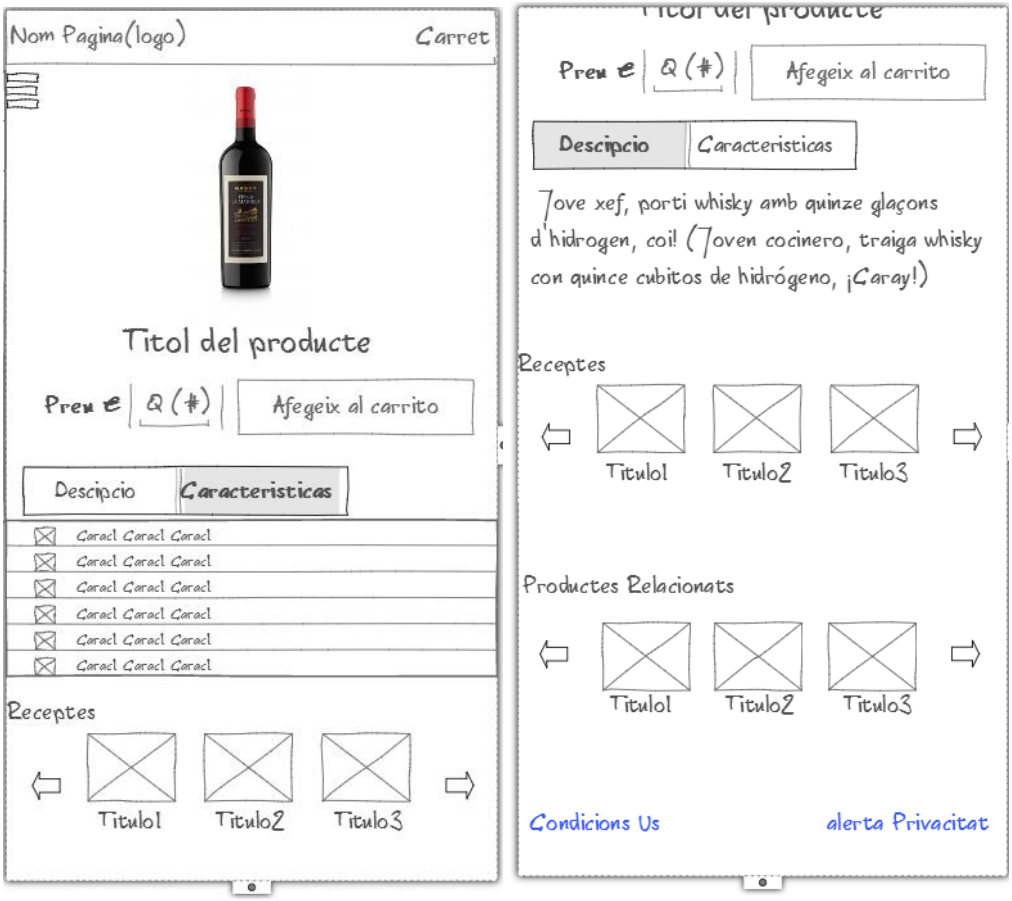
- Secció afegit producte carret compra.
- Descripció del producte.
- Característiques.
- Receptes de cuida associades a aquest producte.
- Productes relacionats.

RQ562

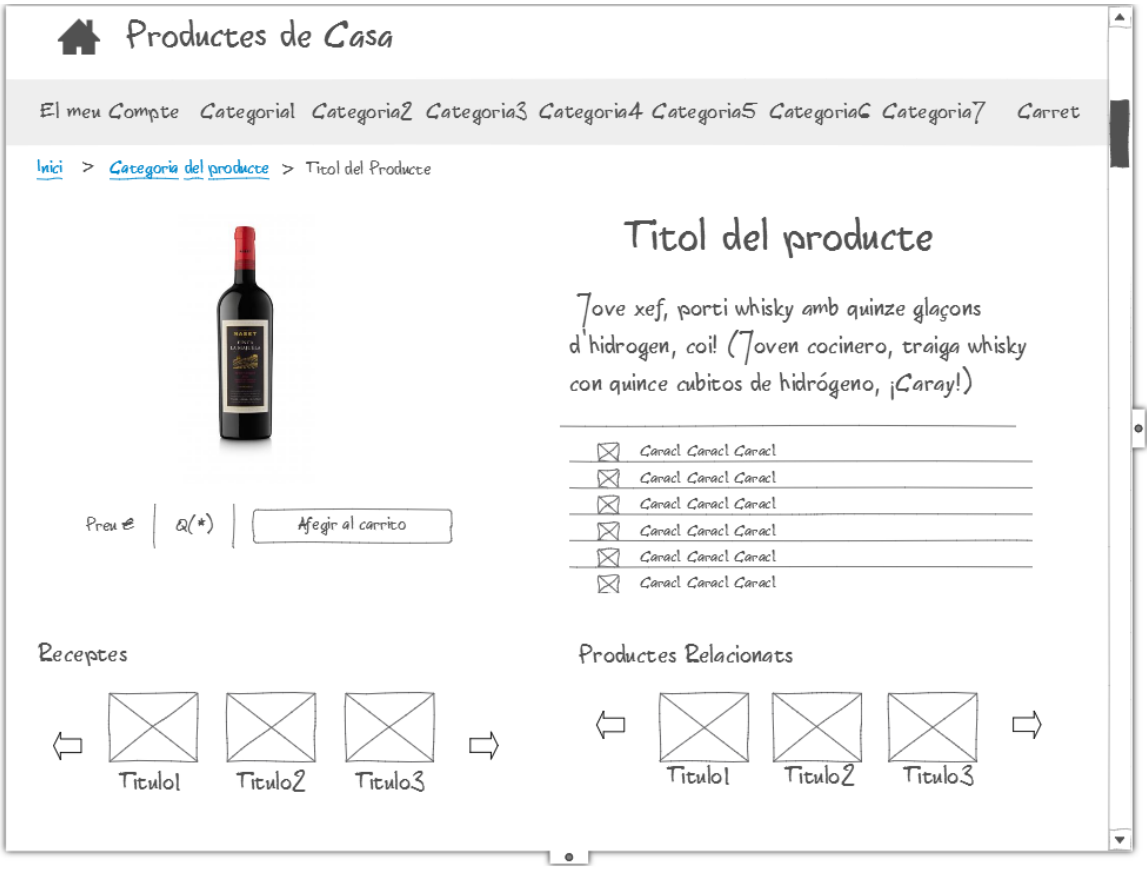
La secció afegir producte al carret de la compra estarà composat per:

- Preu en € (Read Only)
- Quantitat (Read and Write)
- Botó afe geix carret.

Esbós Mòbil


Nom Pagina(logo)
Carret
Titol del producte
Preu €
Q (#)
Afegeix al carrito
Descripcio
Caracteristicas
Caracl Caracl Caracl
Receptes
Titulo1
Titulo2
Titulo3
Productes Relacionats
Condicions Us
alerta Privacitat

Esbós PC


Productes de Casa
El meu Compte
Categoria1
Categoria2
Categoria3
Categoria4
Categoria5
Categoria6
Categoria7
Carret
Inici > Categoria del producte > Titol del Producte
Titol del producte
Preu €
Q(*)
Afegir al carrito
Caracl Caracl Caracl
Receptes
Titulo1
Titulo2
Titulo3
Productes Relacionats

2.2.3.5.6. Pàgina de Receptes

RQ571

Donat que un usuari accedeix a una pagina de receptes, la aplicació web ha de presentar un llistat de totes les receptes disponibles a la base de dades, mostrades en barres de desplaçament *("sliders")* per cada categoria.

2.2.3.5.7. Fitxa de Receptes

RQ581

Donat que un usuari accedeix a una pagina de recepta, la aplicació web ha de presentar la següent informació:

- Productes Relacionats
- Foto de la recepta
- Passos de la recepta

2.2.3.5.8. Informació de contacte.

RQ591

Donat que un usuari accedeix a una pagina de contacte, haurà de tindre disponible tota la informació de contacte de la empresa.

2.2.3.6. Cerca de productes

RQ61

La aplicació web ha de proveir a l'usuari en qualsevol vista, un cercador que permeti la entrada de text.

RQ62

Quan el client fa una cerca mitjançant el cercador, la aplicació web ha de mostrar el llistat de productes que coincideixin amb la cerca que l'usuari ha fet.

2.2.3.7. Compra de productes

RQ71

Donat un client es troba al carret de la compra, quan el client fa clic al botó "Passa per la caixa" la aplicació web ha de presentar els següents passos:

1. Pas Benvinguda.
2. Pas Direcció.
3. Pas Productes.
4. Pas Pagament.
5. Pas Confirmació.

RQ72

El pas de benvinguda serveix para comprovar que l'usuari que vol fer la compra està enregistrat a la web, per tant haurà de proveir la possibilitat de Iniciar sessió o enregistrar-se i en que l'usuari estigui enregistrat aquest pas serà omès.

RQ73

El pas de direcció serveix per definir la direcció d'enviament que tindrà la comanda.

RQ74
El pas de Productes serveix per veure i editar la informació dels productes del carret de la compra.

RQ75
Donat que un usuari està editant la informació dels productes en el pas de Productes, quan queda buit, la aplicació ha de adreçar a la vista del carret de la compra vuit i sortir de la passarel·la de pagament.

RQ76
El pas de confirmació serveix per veure les dades de la comanda i haurà de proveir el adreçament necessari per tal de poder editar i veure els següents passos:

- Direcció.
- Productes.
- Pagament.
- Finalitzar la compra.

2.2.3.8. Categories

RQ81
Donat la aplicació web de Productes de Casa, la aplicació haurà de presentar les següents set categories a la web:

1. Vins Negres
2. Vins Blancs
3. Vins Rosats
4. Caves
5. Cerveses
6. Licors
7. Olis i Vinagres

2.2.4. Requeriments No Funcionals

2.2.4.1. Bases de Dades

Productes de casa distribueix productes relacionats amb la gastronomia i agricultura catalana.

Es vol emmagatzemar la informació del productes que es distribueixen dintre de la base dades. Aquesta informació que es vol emmagatzemar per cada producte ha de ser la següent:

- Nom del producte
- Característiques
- Imatges
  - Miniatura
  - Mitja
- Preu
- Numero d'existències (Stock)
- Data de actualització
- Categoria

Un producte ha de estar només en una categoria però una categoria pot tindre més d'un producte.

El que respecta a la informació que es vol emmagatzemar de les categories haurà de ser la següent:

- Nom Plural
- Nom singular
- URL Imatge capçalera categoria

Un producte pot tindre més d'una característica i una característica por estar associada a un o més productes.

El que respecte a la informació que es vol emmagatzemar relacionada amb les característiques, haurà de ser la següent:

- Nom
- URL Imatge miniatura

Es necessita un historial de les comandes que es facin a la tenda, per tant es voldrà emmagatzemar la següent informació:

- Import total de la comanda
- Adreça on s'ha d'enviar la comanda
- Data de la comanda
- Taxes
- Numero de seguiment

Un producte podrà pot tenir una o més comandes associades, i una comanda por tindre un o més productes associats. Es per aquesta raó que es necessita saber, per cada producte associat a una comanda, la quantitat que es compra.

Les compres dels usuaris es guardarem com a forma de comanda, un usuari pot tindre una o més comandes associades, una comanda només pot pertànyer a un usuari.

Els usuaris es podran registrar i fer compres usant la web es precisa guardar la següent informació per tal d'identificar-los:

- Contrasenya
- Nombre
- Cognom
- Adreça electrònica
- Data de registre
- Data de naixement
- Telèfon

A més cada usuari podrà tindre una o més direccions d'enviament associades de les quals es voldrà emmagatzemar la següent informació:

- Nom destinatari
- Línia adreça un
- Línia adreça dos
- Província / Regió

- Codi Postal
- País
- Numero de telèfon
- Adreça Electrònica de Contacte

La base de dades ha d'estar preparada per emmagatzemar la informació relacionada amb receptes de cuina, que tinguin associats productes de la tenda.

D'una recepta es vol guardar la següent informació:

- Títol de la recepta
- Contingut de la recepta

Una recepta ha de tenir associats un o més productes i un producte pot estar associat a un o més productes.

2.2.4.2. Seguretat

Contrasenyes

Les contrasenyes no hauran d'estar emmagatzemades a la base de dades. La aplicació haurà de emmagatzemar el resultat d'aplicar una funció hash "SHA1".

Traçabilitat d'accions – "Logging"

La aplicació web ha d'estar preparada per enregistrar les accions dels usuaris en fitxers tipus *"log"* per tal de poder identificar qualsevol tipus d'error o anomalies a la aplicació.

2.2.4.3. Usabilitat

Es dona gran importància a que la aplicació sigui visible possible a dispositius amb capacitat de rendiment més limitat com podem ser els Smartphones o Tables. El concepte que es demana es el de Diseño Adaptable, o més conegut com "Responsive Design".

Il·lustració 2.1 Responsive Design

2.3. ESTAT DE L'ART

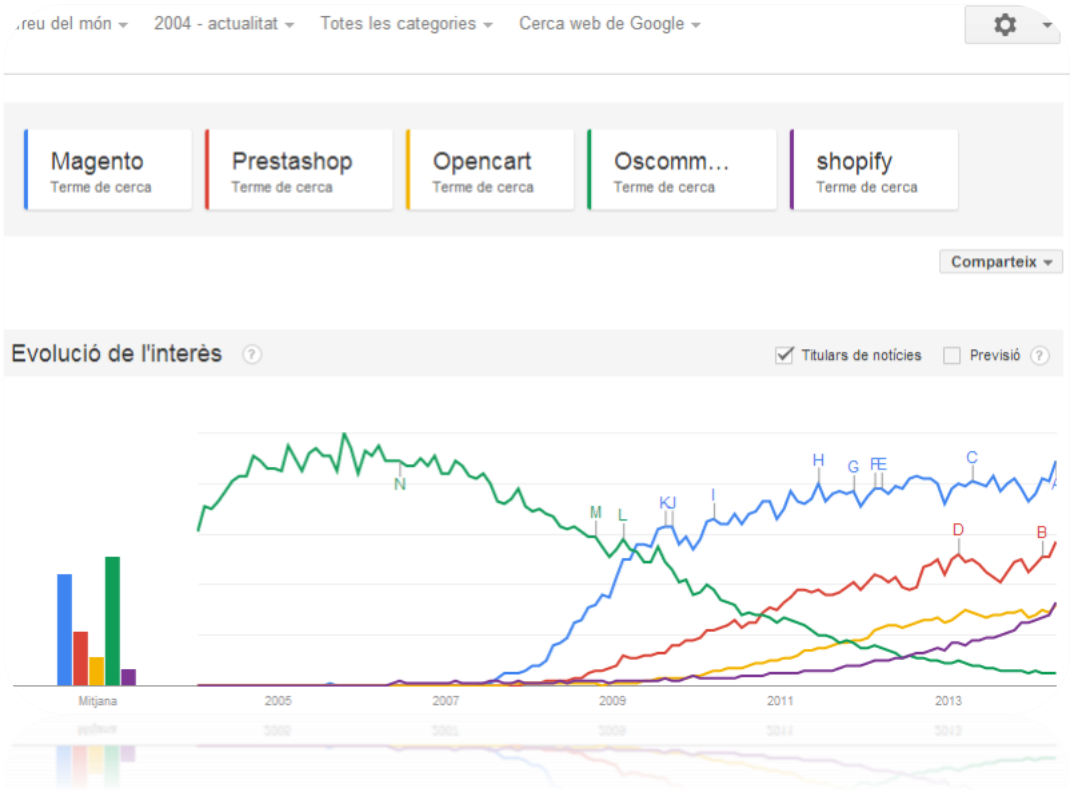
Actualment trobem una diversitat molt àmplia en el que representa plataformes per construcció de negocis online. En aquest apartat estudiarem les característiques dels més exitosos, destacant les seves característiques amb l'objectiu de veure si es possible treure profit d'alguna d'elles.

També analitzarem l'opció alternativa. La creació d'una aplicació web des de zero. On analitzarem un parell de tecnologies de la mateixa manera que ho farem amb les plataformes *eCommerce*.

2.3.1. Plataformes eCommerce

M'agradaria començar aquest apartat, preguntant sobre la visió que tenen els nostres grans amics "Google" respecte a les plataformes de *eCommerce* que hi ha al mercat actual.

Com podem observar en la figura 2.1 ens podem fer una idea, a grans trets, de quines són les més famoses a dia d'avui. Es per això que les utilitzaré com cas d'estudi, ja que es troben en el punt que es troben per alguna raó i considero interesant tindré present les seves característiques més destacades.



Il·lustració 2.1 Informació Google Trends Plataformes eCommerce

Magento es ara mateix la plataforma líder en el mercat per la seva funcionalitat i versatilitat en el control de maquetació, contingut i disseny dels negocis desenvolupats en aquest sector.

Característiques

- Bona gestió atributs del productes.
- Optimització en els camps de cerca.
- Segmentació dels clients.
- Eines de suggeriments de productes
- Suport de múltiples passarel·les de pagament
- Possibilitat creació múltiples tendes o diferents departament dins de la mateixa tenda

Per altra banda tenim a **Prestashop** amb més de 165. 000 tendes obertes arrel del món, és una eina que, no obstant la bona competència que té, ha estat capaç de mantindré un volum gran de clients utilitzant la seva solució.

Característiques

- Fàcil gestió Aplicació web.
- Facilitat d'integració formes de pagament
- Facilitat d'administració del catàleg de productes
- Proporciona eines d'Anàlisi y prestació d'informes
- Proporciona eines de Marketing tals com fidelització de clients

Com hem vist, les dues eines tenen punts característics bastant contundents per tal d'implantar i executar un negoci online fent ús d'una d'aquestes plataformes.

	Magento Open Source eCommerce	PRESTASHOP The Best E-Commerce Experience
Facilitat Instal·lació	★★★★☆	★★★★★
Nombre de Característiques	★★★★★	★★★☆☆
Requeriments de sistema	★★★☆☆	★★★★☆
Compatibilitat	★★★★★	★★★★☆
Facilitat de personalització	★★☆☆☆	★★★★☆
Suport Tècnic	★★★★★	★★★★☆
Funció de Cerca	★★★★☆	★★★★★
Temes disponibles	★★★★★	★★★★☆
SEO	★★★★★	★★★☆☆

Taula 2.1 Comparativa Magento Prestashop

Podem concloure que **Magento** és una bona opció per aquells negocis que busquen un major control sobre la personalització de la seva botiga. També orientat per negocis més grans o que s'estiguin plantejant créixer. En contraposició tenim a **Prestashop** una opció que aquells negocis més petits o amb aspiració de no créixer tan despresa trobarà en aquesta plataforma una eina amistosa, ja que és un sistema que es fa fàcil de mantenir i no requereix una inversió de temps tan elevada.

2.3.2. Creació web Sencera

Aquesta opció requereix un nivell més elevat per tal de dur a terme, ja que implica certs coneixements per tal de seleccionar la tecnologia oportuna i necessària per la implementació de la nostra aplicació web.

Actualment **PHP+MySQL** usant frameworks que segueixen el patró MVC tenen una comunitat molt amplia, usat per una gran quantitat d'empreses relacionades amb el comerç electrònic. Tot i així, PHP és un llenguatge interpretat en execució, per certs usos pot resultar un inconvenient que el codi font no pugui ser ofuscat. Aquesta característica te implicacions importants en el rendiment d'una aplicació PHP, ja que, si no és implementada correctament, sol funcionar amb baix rendiment en la seva versió de codi a més baix nivell. Aquest tipus de problemes solen ser solucionats amb tècniques de cache de codi i amb tècniques de balanceig de carrega.

Alternativament, han anat apareixent tecnologies innovadores que intenten donar solucions en quan el que respecte al rendiment, tals com **Django (Bassada en Python), NodeJS (Basada en JavaScript i orientat a servidor).** Tot i que són opcions molt atractives, aquestes plataformes, encara estan en creixement i la comunitat, tot i que és molt activa, encara es molt jove.

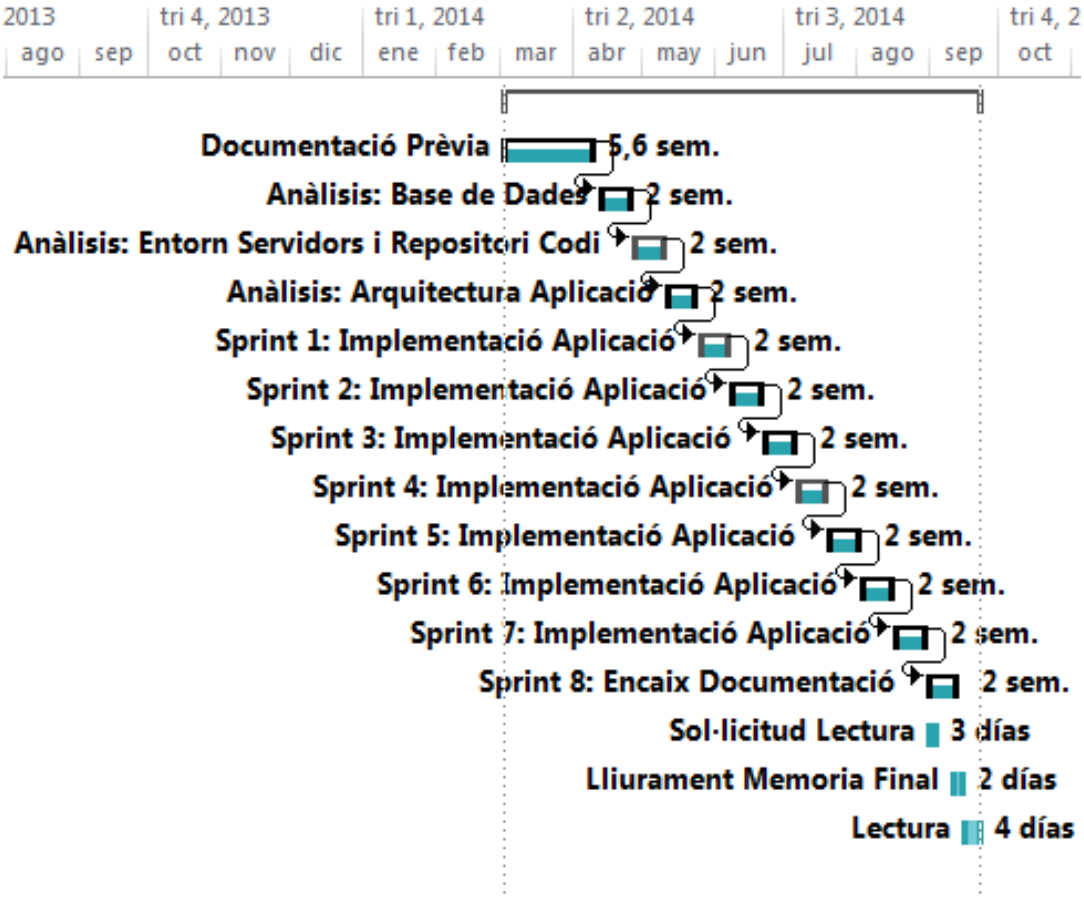
2.3.3. Conclusions

Donada les necessitats d'aquest projecte, escollir opcions com Magento o Prestashop, ens implicaria modificació de bases de dades i això podria representar un gran problema a l'hora de mantenir la nostra aplicació. També la configuració a nivell "Responsive", tot i que les ultimes versions de Prestashop per exemple ja estan implementant funcionalitats relacionades amb aquest tòpic, encara no acaben de incloure una funcionalitat concreta, i ens donaria una feina extra per tal d'incloure aquest tipus de funcionalitat.

Donat això, aquest projecte aposta per escollir una solució creada des de cero i personalitzada segons les necessitats del client, i per tant es prendrà aquest camí per tal de desenvolupar aquest projecte.

2.4. PLANIFICACIÓ

2.4.1. Diagrama de Gantt



Il·lustració 2.2 Taula planificació taula de Gantt

2.4.2. Taula detallada Tasques

Nombre de Tasca	Duració	Principi	Final
Projecte Fi Carrera	146 d.	03/03/2014	22/09/2014
Documentació Prèvia	5,6 sem.	03/03/2014	09/04/2014
Definició Objectius	1 sem	03/03/2014	07/03/2014
Definició Requeriments	2 sem.	10/03/2014	21/03/2014
Formació per Estudi de l'art	2 sem.	24/03/2014	04/04/2014
Encaix Document per Lliurament	3 d.	07/04/2014	09/04/2014
Anàlisi: Base de Dades	2 sem.	14/04/2014	25/04/2014
Anàlisi Requeriments	1 d.	14/04/2014	14/04/2014
Disseny Entitat Relació	3 d.	15/04/2014	17/04/2014
Disseny Lògic	3 d.	18/04/2014	22/04/2014
Disseny Físic	2 d.	23/04/2014	24/04/2014
Documentació	1 d.	25/04/2014	25/04/2014
Anàlisi: Entorn Servidors i Repositori Codi	2 sem.	28/04/2014	09/05/2014

Anàlisi de Requeriments	1 d.	28/04/2014	28/04/2014
Disseny Lògic	4 d.	29/04/2014	02/05/2014
Disseny Físic	4 d.	05/05/2014	08/05/2014
Documentació	1 d.	09/05/2014	09/05/2014
Anàlisi: Arquitectura Aplicació	2 sem.	12/05/2014	23/05/2014
Anàlisi de Requeriments	1 d.	12/05/2014	12/05/2014
Disseny Lògic	4 d.	13/05/2014	16/05/2014
Disseny Físic	4 d.	19/05/2014	22/05/2014
Documentació	1 d.	23/05/2014	23/05/2014
Sprint 1: Implementació Aplicació	2 sem.	26/05/2014	06/06/2014
Tasques d'Sprint	8 d.	26/05/2014	04/06/2014
Proves	0,5 d.	05/06/2014	05/06/2014
Documentació	1 d.	05/06/2014	06/06/2014
Deployment Release	0,5 d.	06/06/2014	06/06/2014
Sprint 2: Implementació Aplicació	2 sem.	09/06/2014	20/06/2014
Tasques d'Sprint	8 d.	09/06/2014	18/06/2014
Proves	0,5 d.	19/06/2014	19/06/2014
Documentació	1 d.	19/06/2014	20/06/2014
Deployment Release	0,5 d.	20/06/2014	20/06/2014
Sprint 3: Implementació Aplicació	2 sem.	23/06/2014	04/07/2014
Tasques d'Sprint	8 d.	23/06/2014	02/07/2014
Proves	0,5 d.	03/07/2014	03/07/2014
Documentació	1 d.	03/07/2014	04/07/2014
Deployment Release	0,5 d.	04/07/2014	04/07/2014
Sprint 4: Implementació Aplicació	2 sem.	07/07/2014	18/07/2014
Tasques d'Sprint	8 d.	07/07/2014	16/07/2014
Proves	0,5 d.	17/07/2014	17/07/2014
Documentació	1 d.	17/07/2014	18/07/2014
Deployment Release	0,5 d.	18/07/2014	18/07/2014
Sprint 5: Implementació Aplicació	2 sem.	21/07/2014	01/08/2014
Tasques d'Sprint	8 d.	21/07/2014	30/07/2014
Proves	0,5 d.	31/07/2014	31/07/2014
Documentació	1 d.	31/07/2014	01/08/2014
Deployment Release	0,5 d.	01/08/2014	01/08/2014
Sprint 6: Implementació Aplicació	2 sem.	04/08/2014	15/08/2014
Tasques d'Sprint	8 d.	04/08/2014	13/08/2014
Proves	0,5 d.	14/08/2014	14/08/2014
Documentació	1 d.	14/08/2014	15/08/2014
Deployment Release	0,5 d.	15/08/2014	15/08/2014
Sprint 7: Implementació Aplicació	2 sem.	18/08/2014	29/08/2014
Tasques d'Sprint	8 d.	18/08/2014	27/08/2014
Proves	0,5 d.	28/08/2014	28/08/2014
Documentació	1 d.	28/08/2014	29/08/2014
Deployment Release	0,5 d.	29/08/2014	29/08/2014
Sprint 8: Encaix Documentació	2 sem.	01/09/2014	12/09/2014

Tasques d'Sprint	8 d.	01/09/2014	10/09/2014
Proves	0,5 d.	11/09/2014	11/09/2014
Documentació	1 d.	11/09/2014	12/09/2014
Deployment Release	0,5 d.	12/09/2014	12/09/2014
Sol·licitud Lectura	3 d.	01/09/2014	03/09/2014
Lliurament Memòria Final	2 d.	12/09/2014	15/09/2014
Lectura	4 d.	17/09/2014	22/09/2014

Taula 2.2 Tasques detallades

2.5. VALORACIÓ

2.5.1. Valoració Temps

Dias Treballats	146 Dies
Hores Treballades per dia	2h
Preu per Hora	20€
Total Preu Temps	**5840.00€**

2.5.2. Valoració Material

Maquines Virtuals	0€
Servidor Producció	10€ (lloguer 2 mesos)
Llicencia Sublime Text Code Editor	65€
Total	**75.00€**

2.5.3. Valoració Final

Servidor Producció	**5915€**

2.6. ANÀLISI DE RISCOS

Riscos tecnològics: La aplicació que vol desenvolupar aquest projecte, sigui estructurada com sigui, al final serà emmagatzemat en un sistema de servidors. Això implica un risc ja que en cas de que el servidor es faci malbé, es podria perdre el codi font de la nostra aplicació. És per això que és necessari tindre en compte un control de versions del codi font, afegint una redundància controlada del servidors on resideix el codi font de la nostra aplicació.

Classificació: Risc Mitjà.

Riscos de Recursos Humans: Aquí tenim un risc prou elevat, ens trobem que és un projecte amb mida de treball de grandària elevada. Al ser només un únic membre , la responsabilitat de que la feina sigui acabada recau només en una persona, i en cas que hi falti el projecte podria veure totalment compromès.

Classificació: Risc Alt.

Riscos Organitzacionals: En aquest cas, encara no es prou gran el projecte com per dependre de decisions organitzacionals. Tot i que s'ha de tindre en compte per futures incorporacions del projecte, per aquesta primera versió, la comunicació es prou flexible degut a la proximitat amb el client.

Classificació: Risc Baix.

Riscos d'Eines: Tot i que hi haurà una dependència de diferents eines, sigui com sigui el rumb d'aquest projecte, degut a la seva naturalesa. Podríem estar tranquils ja que si es fa una bon Anàlisi les eines al final només et dona una manera d'aplicar aquest Anàlisi i fer-lo palpable.

Classificació: Risc Baix.

Riscos de Requeriments: Aquí tenim un punt crític, però dependrà de l'enfocament del projecte. La idea d'aquest projecte és la de tindre una continua retroalimentació de la informació entre el client i el desenvolupador. D'aquesta manera aconseguim que els requeriments siguin remodelats en funció de lo que el client veu que s'està implementant[1]

Classificació: Risc Mitja.

Riscos d'Estimació i Planificació: Com sempre aquest és un rics que s'ha d'assumir en tot projecte, i encertar sempre no és una tasca fàcil, i només ens el dona, la experiència. Tot i així aquest projecte està desenvolupant enmig d'un curs acadèmic i pot fer que les estimacions i planificacions variïn en funció d'imprevistos, com exàmens de recuperació, lliuraments,... etc.

Classificació: Risc Alt.

2.7. CONCLUSIONS

Ens quedem principalment amb la idea que aquest projecte farà una aplicació des de zero amb els beneficis i inconvenients que això comporta i hem estudiat a l'apartat 2.3 d'aquest capítol.

Aclarir també que tot i que s'ha fet una valoració de la feina feta, aquest treball com que és amb objectiu acadèmic, serà una donació i no es cobrarà res per l'aplicació que es desenvolupi durant el trajecte d'aquest projecte.

[1] Aquest plantejament està basat en les idees del desenvolupament Àgil del software http://www.agilemanifesto.org/iso/ca/

Capítol 3. FONAMENTS TEÒRICS

Passem a la part de recol·lecció de tècniques necessàries per tal de desenvolupar el projecte plantejat en els capítols anteriors. En aquest capítol es trobarà un Anàlisi teòric detallat, el que jo, com a responsable desenvolupador d'aquest projecte, considero necessari realitzar, per tal de marcar una guia de recerca clara i concisa, per la implementació i entrega de les diferents exigències i funcionalitats que em planteja aquest projecte.

Amb l'objectiu de organitzar les idees en funció de l'àmbit de treball, he dividit aquest capítol en tres parts:

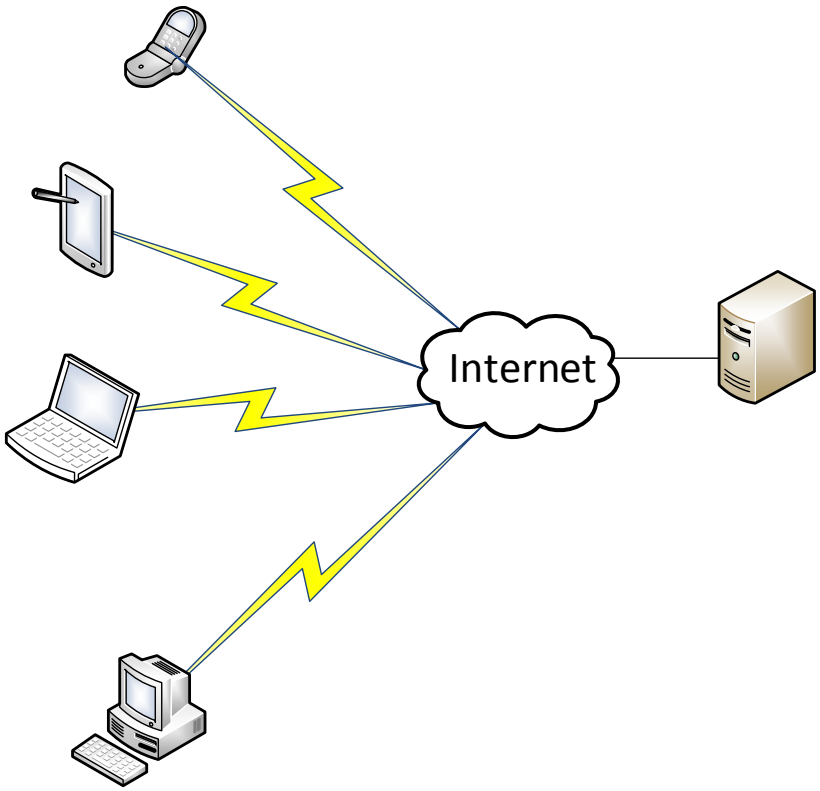
1. **Desplegament del software**
2. **Estructuració i emmagatzematge de dades**
3. **Disseny de software**

3.1. DESPLEGAMENT DEL SOFTWARE

Abans de començar, es necessari definir un concepte clau. Es tracta de la **Lògica de negoci d'un sistema**, el que s'entén per un conjunt de processos, que implementen les regles de funcionament del sistema. I es que hi ha un factor rellevant que diferència a una aplicació web si el comparem amb un simple lloc web. Consisteix en la possibilitat que s'ofereix a l'usuari d'interactuar sobre la lògica de negoci que s'ha dissenyat en un servidor. Per exemple, en el nostre cas, una tenda web, la lògica de negoci són els processos que implementen el procediment de compra, selecció de productes, carret de la compra, confirmació de la compra, pagament, seguiment de la entrega,...etc.

Es per això que es imprescindible una definició i modelatge a nivell d'arquitectura concret, per tal de desplegar i fer accessible el nostre software de manera que es pugui assegurar un mínims de rendiment del nostre sistema. Anem a veure a continuació estructures clàssiques que es podem servir d'ajuda per tal de definir el nostre escenari de desplegament del software

3.1.1. Modelatge Client Servidor



Il·lustració 3.1 Estructura bàsica client Servidor

En computació, es tracta d'un model o estructura distribuït d'aplicació, el qual, divideix tasques o carreges de treball entre proveïdors d'un recurs o servei, anomenats, **Servidors**, i sol·licitants de serveis, anomenats **Clients**. Es comú, que els clients i els servidors es comuniquen a través d'una xarxa informàtica de maquines allunyades un de l'altre, tot i que es possible que el client i el servidor poden residir en el mateix sistema. Un host servidor executa un o més programes de servidor que comparteixen els seus recursos amb els clients, però, en canvi, un client no comparteix cap dels seus recursos i demana la funció de contingut o servei que resideix al servidor.

3.1.1.1. Característiques

Client → Emissor

- És el responsable de començar les sol·licituds o peticions, tenen per tant un paper actiu en la comunicació (Dispositiu Mestre Esclau).
- Rep i espera les respostes del servidor.
- Interactua amb el sistema (usuaris finals) mitjançant una Interface Gràfica.
- Té limitacions de velocitat de connexió en funció del servei de xarxa que utilitzi i s'ha de tindre en compte, si es vol maximitzar el performance en qualsevol plataforma[2].

Servidor → Receptor

- Programat per iniciar-se i esperar, per les possibles peticions que puguin arribar dels clients, exerceixen llavors, un paper passiu en la comunicació[3].
- Preparat pel processament de les peticions, de les quals, generen una resposta que es enviada als clients.
- Tenen limitacions a nivell màxim de peticions, però en general estan pensats per acceptar les connexions des d'un gran número de clients.
- No és comú que els servidors interactuïn amb els usuaris finals.

3.1.1.2. Avantatges

Tenim diferents aspectes que afavoreixen la utilització d'aquest tipus d'arquitectura. Per una banda tenim el concepte de **Centralització del Control**. Quan parlem de control diem totes aquells processos tals com, els accessos, recursos i integritat de les dades, que són controlats pel servidor de forma que un programa client defectuós o no autoritzat no pugui fer mal al sistema. Aquest últim concepte afavoreix i fa fàcil el manteniment, ja que al estar distribuïdes les funcions i responsabilitats entre diferents estacions independents, es possible reemplaçar, reparar, actualitzar, o fins i tot, traslladar un servidor, mentre que els seus clients no es veuran afectats per aquest canvi. Donat això, ens porta un nou concepte, anomenat **Encapsulació.** Finalment ens trobem tecnologies suficientment desenvolupades, que implementen el paradigma del client servidor, assegurant la seguretat en les transaccions, la afabilitat de la Interface i facilitant la usabilitat.

3.1.1.3. Inconvenients

Congestió de tràfic ha estat sempre un problema en el paradigma client servidor. Suposant l'escenari en el que una quantitat de grandària elevada de peticions de clients simultània, arriben a un mateix servidor, aquest mateix es veu afectat en termes de rendiment.

Per un altre banda, es possible trobar-se amb problemes de robustesa. Això vol dir en definitives, que quan un servidor està caigut, les peticions dels clients no podran ser processades i satisfetes correctament[4].

[2] És evident que no és possible tindre el millor performance en totes les plataformes. Però si es poden tindre en compte implicacions capacitat processament dispositius, amplada de banda,... etc.
[3] Conegut també com a **Dispositiu Esclau** en un model de comunicació tecnològic. http://en.wikipedia.org/wiki/Master/slave_(technology)
[4] Un exemple d'arquitectura amb molt bora robustesa son les de tipus d'Igual a Igual (de l'anglès *Peer to Peer*) http://ca.wikipedia.org/wiki/D%27igual_a_igual

3.2. ESTRUCTURACIÓ I EMMAGATZEMATGE DE DADES

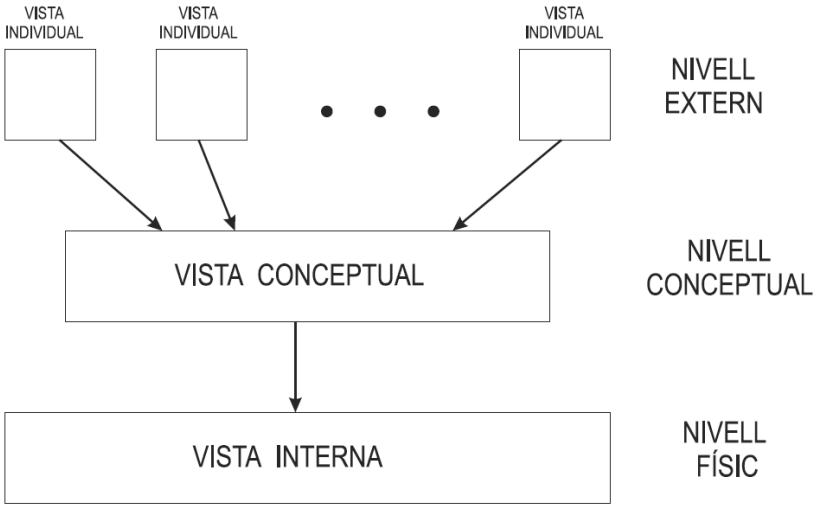
Una vegada vist el que respecta a estructuració de servidors i clients, hi ha un segon punt molt important, i es el relacionat a la arquitectura d'emmagatzematge de les dades d'una aplicació software. Ben es cert que una bona elecció de modelatge de servidors es necessària, sense una bona estructuració de les dades es probable que el nostre esforç es vegi mal aprofitat.

És per això que considero important la recol·lecció de pràctiques relacionades amb l'estructuració de les dades per tal de tindre bones referències a l'hora de modelar l'arquitectura que portarà la aplicació web que es desenvoluparà en aquest projecte. Anem a veure a continuació que tipus d'arquitectures clàssiques com a cas d'estudi d'aquest projecte.

3.2.1. Arquitectures Base de dades

Arquitectura ANSI/SPARC

Es tracta d'un estàndard de disseny abstracte per l'administrador de sistemes de Bases de Dades proposat a l'any 1975. La majoria dels més moderns DBMS estan basats en aquests tipus de sistemes. Tot i així, el model ANSI-SPARC mai s'ha proposat com un Standard formal.



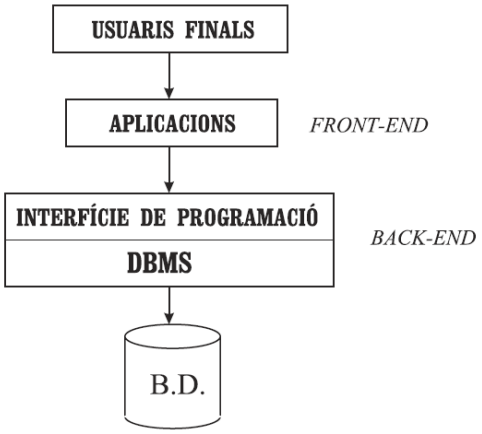
Il·lustració 3.2 American National Standards Institute, Standards Planning And Requirements Committee (Study Group on Database Managment System)

Anem a veure les parts de les que es composa:

1. ***Vista Interna:*** Més proper al emmagatzemament físic i és l'encarregat d'organitzar i emmagatzemar les dades a un molt baix nivell o millor dit, **Nivell Físic.**
   - Nivell no Relacional → Baix nivell
   - Elements: Registres, apuntadors, índexs, ... etc.
2. ***Vista Conceptual:*** Es tracta d'un nivell intermig que fa de Interface entre els dos nivells restants. Aquests tipus de vistes tenen el nom de **Nivell Conceptuals**.
   - Nivell Relacional.
   - Elements: Taules, operadors i relacions.
3. ***Vista Individual:*** És la part més propera al usuaris. Té com a missió principal el tractament de la forma com els usuaris individuals perceben les dades. Aquest tipus de vistes prenen el nom de **Nivell Extern**.
   - Nivell Relacional.

Arquitectura Back-end / Front-end

En aquest cas ens trobem amb una visió diferent de l'arquitectura d'una base de dades. Visió des del punt de vista del desenvolupament i execució d'aplicacions software, sobre les dades.



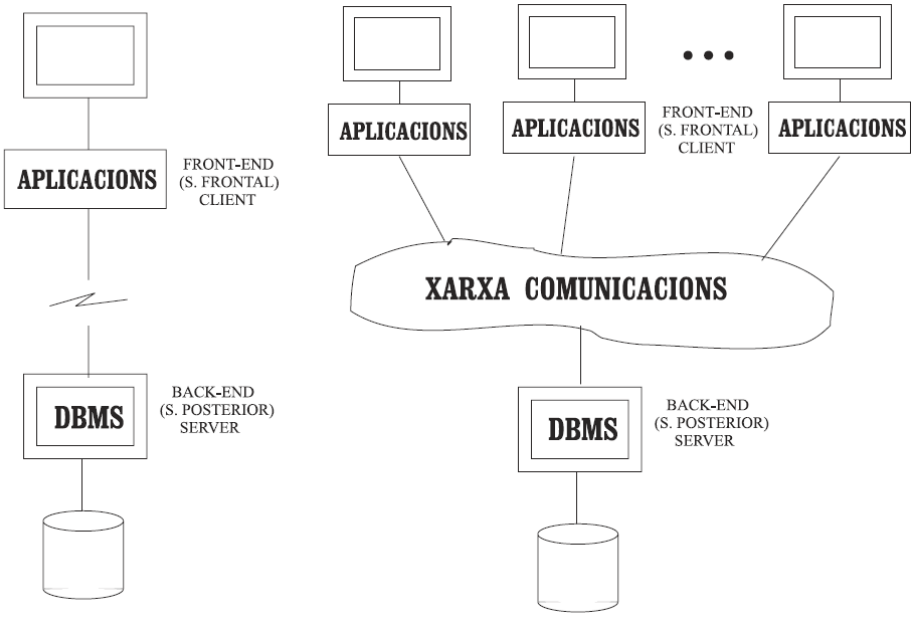
Il·lustració 3.3 Sistema típic Back-end / Front-end

Anem a veure les parts de les que es composa:

1. ***Back-end:*** En aquest cas està format per un sistema d'administració de bases de dades (DBMS). Realitza totes les funcions descrites a en el DBMS.
2. ***Front-end:*** Aplicació que gestiona peticions i les tradueix en sentencies que seran interpretades i respostes per la part Back-end.

Aquest tipus d'arquitectura està molt a prop del paradigma Client Servidor, si fem la següent analogia:

- **Secció Posterior:** Servidor, Despatxador. Execució de DBMS en una màquina.
- **Secció Frontal:** Client. Execució de l'aplicació en un altre màquina.

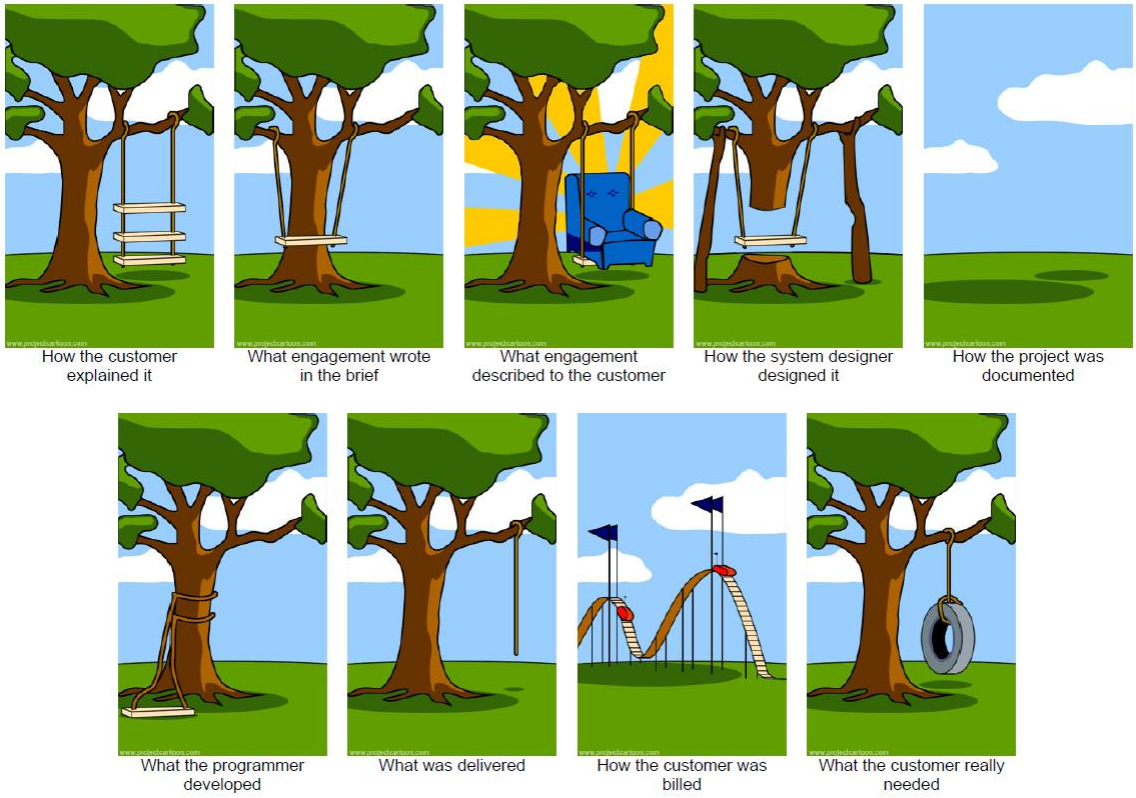


Il·lustració 3.4 Paradigma C/S i Arquitectura Back-end / Front-end

3.2.2. Disseny una base de dades

Quan parlem de disseny de bases de dades, es important tindre clar, quines són les fases necessàries per a que es tingui la informació estructurada de una manera que afavoreixi la implementació de la nostra lògica de negoci. Per tant, a continuació veurem quines són aquestes fases i la seva respectiva explicació del treball que s'ha d'elaborar i per a que serveix en cada una d'elles.

Captació i Anàlisi de requeriments

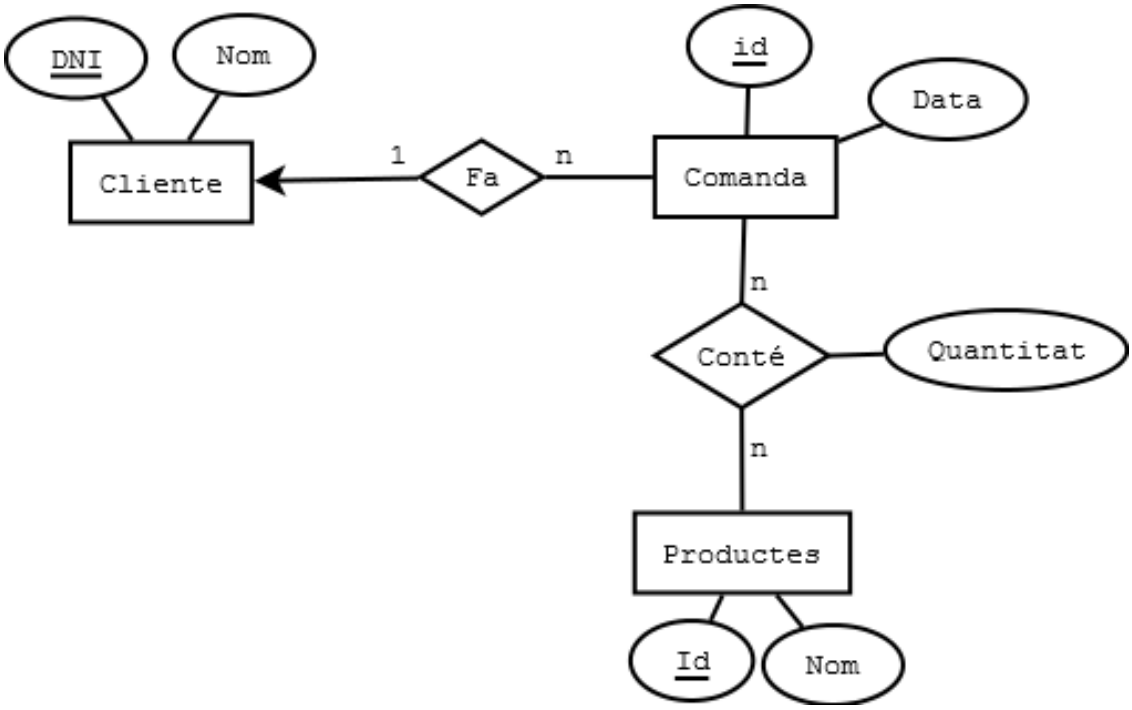


Il·lustració 3.5 Requeriment entès per diferents perspectives.

Aquesta fase es tracta de la realització d'una sèrie d'entrevistes amb l'usuari final que utilitzarà la base de dades, amb l'objectiu d'extreure la informació necessària per tal de fer un disseny que cobreixi les necessitats del seu negoci.

En aquesta fase és de gran importància, i ha de quedar molt clara ja que qualsevol error d'implementació degut mal entès durant aquest procés es veurà reflexa un costos elevats i manteniments innecessaris.

Disseny conceptual de la base de dades

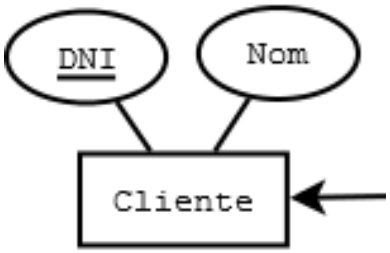


Il·lustració 3.6 Exemple Diagrama Entitat Relació

Arribat aquest punt es precisa d'un sistema de modelatge de dades, amb l'objectiu de descriure les dades or aspectes de la informació de un domini de negoci o els seus requisits de procés, d'una manera abstracta que es presti, per tal de ser finalment implementat en una base de dades com pot ser aquelles de tipus relacional. Estem parlant de un tipus de modelatge anomenat, Model Entitat Relació.

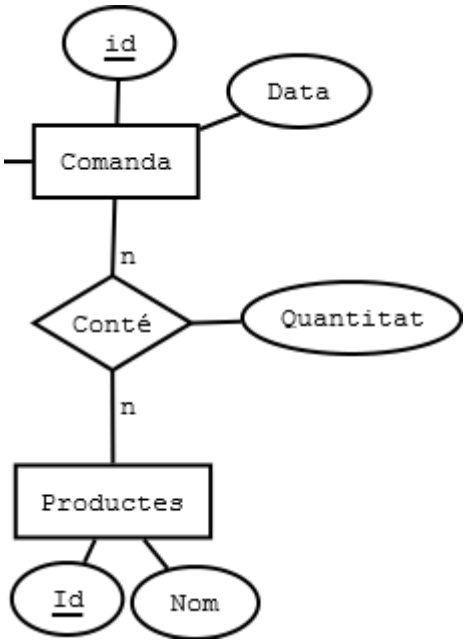
Anem a continuació a analitzar els principals elements, que un model entitat relació està conformat. Per això partirem del diagrama exemple descrit a la il·lustració 3.6.

1. **Entitats:** Quan ens referim és alguna que existeix en el món real, distingible de la resta de coses, i de la qual ens interessen emmagatzemar les seves propietats o característica.



Il·lustració 3.7 Entitat Exemple

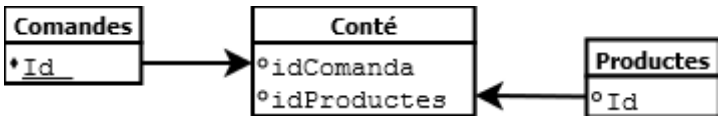
2. **Relacions:** Quan parlem de relacions en referirem a totes aquelles associacions o dependències conceptuals que es poden definir entre dues o més entitats. Tot i així el numero màxim recomanable de relacions es de tres entitats, relació la qual s'anomena ternària, sent aquesta poc comú per les característiques tan concretes que té.



Il·lustració 3.8 Relació Exemple

3. **Atributs:** Quan ens referim al terme atributs, parlem d'aquelles propietats o característiques ens interessa guardar de les entitats. També ens las podrem trobar en relacions ja que pot interessar definir segons quines característiques per a una relació, com és el cas que ens trobem en la il·lustració 3.8.

Disseny lògic



Il·lustració 3.9 Exemple Traducció Relacional

Una vegada es té definit la fase conceptual superada, ara hem de trobar la manera de realitzar una traducció o conversió a un sistema relacional, per tal de utilitzar-lo un sistema real d'aquest tipus, que ens pugui gestionar i emmagatzemar les nostres dades ja estructurades. Es llavors, quan parlem Estructuració en Esquemes Relacionals. Al final, son conceptes molt similars, la qual cosa, permet passar fàcilment d'un model a l'altre.

Disseny Físic

Arribem al pas final, el qual consisteix en la definició dels fitxers i estructures d'emmagatzematge corresponent al conjunt de relacions. En aquesta fase s'ha de definir les estructures d'emmagatzematge necessària sobre els fitxers, ja que d'aquesta manera, podrem millorar el temps d'accés a la informació.

3.3. DISSENY DE SOFTWARE

Una vegada hem fet l'estudi de com definir els pilars necessaris per una aplicació software com la que vol implementar aquest projecte, ara es moment de parlar-ne del disseny d'arquitectura en termes de més alt. A l'apartat 3.1 parlàvem del concepte de lògica de negoci, serà en aquesta fase on aquesta lògica es implementada per tal de que l'usuari final sigui capaç de fer-la servir. A continuació estudiarem les diferents possibilitats que tenim a nivell d'arquitectura de software relacionat amb les necessitats que aquest projecte necessita.

No és tracta d'una fase trivial, en la programació es considerava gaire bé un art. Per sort, amb el temps s'ha fer una gran feina en aquest camp, i s'han anant descobrint i desenvolupant formes i guies generals, amb base a les quals se puguin resoldre els problemes plantejats per diferents projectes. És això lo que coneixem com Arquitectura del Software actualment, i es de lo parlarem i estudiarem a continuació.

3.3.1. Arquitectures Software de aplicacions Web

Tres Nivells

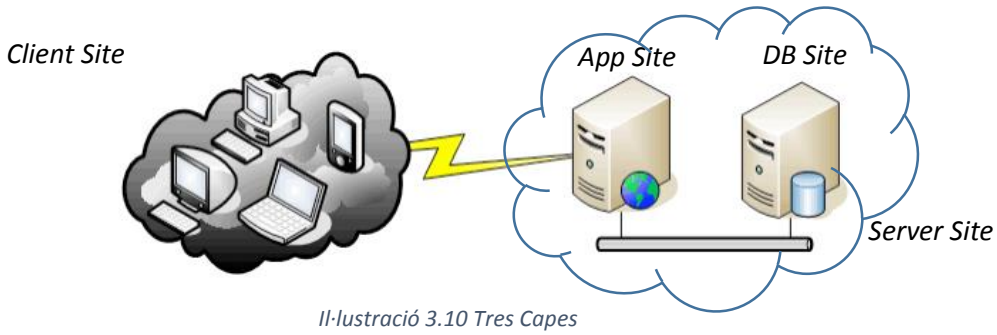
Quan parlem d'aplicacions distribuïdes, partint del model client servidor, quan augmentem la complexitat dels processos s'acaba amb el denominat "Client Pesat". Al posar major part del codi necessari per portar els processos al client, aquest ha de descarregar del servidor les dades necessàries per dur-los a terme. Per tant veiem clar dues raons per lo que això és ineficient:

- La xarxa pateix una gran carrega degut a les múltiples descarregues de cadascuna de las estacions dels clients.
- La gran dependència en el rendiment del hardware al costat del client.

Il·lustració 3.9 Clients Heterogenis

Les arquitectures en tres nivells divideix la funcionalitat per optimitzar l'ús de recursos. S'aconsegueixen solucions molt més flexibles i escalables. Els tres nivells són:

1. Client: conté els components d'usuari que són únics per a cada un d'ells. Això és la lògica de la aplicació especifica del usuari i la Interface.
2. Aplicació: Construeix un entorn multiusuari, i manté les parts compatides de la aplicació. Las operacions amb us intensiu de dades s'han d'executar en aquest nivell. També és el punt on es pot dur a terme la coordinació de transaccions.
3. Emmagatzematge: És el nivell de base de dades. S'especialitza en donar servei de persistència a les dades de la aplicació i permet manipular grans volums d'ells.



Il·lustració 3.10 Tres Capes

Al mantindré bona part del codi en el nivell intermedi, la aplicació es manté lluny de la Interface d'usuari i de la base de dades. Per tant, es pot fer canvis en el codi en el nivell d'aplicació sense que això afecti a la resta. A més, les dades es troben centralitzades i fàcilment accessibles sense la necessitat de moure'ls completament fins el client.

Aquest model s'utilitza àmpliament en l'entorn web. S'incorpora la figura del servidor d'aplicacions que permet que el sistema gestioni la lògica de negoci i l'estat. Com que es tracta d'una aplicació web, la Interface amb el client segueix sent bàsicament HTTP i per tant el servidor d'aplicacions incorpora un servidor web o utilitza un extern.

La divisió de l'aplicació en nivells suposa la necessitat d'establir interfases entre elles. Si a més, es requereix de la construcció d'una arquitectura independent dels fabricants, llavors una de les millors opcions seria l'ús d'elements que implementin Interfases estàndard.

Client web lleuger

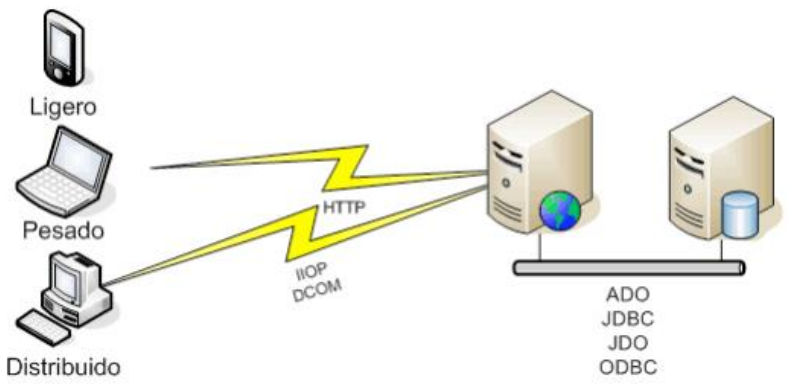
Tota la lògica de negoci es manté al servidor. Això permet tindre clients que només requereixin l'ús d'un navegador web senzill. Bàsicament HTML, Formularis per la entrada de dades i llenguatge Script (p.e JavaScript) para el panell previ de les dades introduïdes.

Client web pesat

Parteix de la lògica de negoci que realitza el client. Applets, ActiveX, ... pel encapsulament del codi que permet la seva execució en el navegador web del client. Per a la comunicació amb el servidor, es segueix utilitzant HTTP.

Client web distribuït

L'Aplicació es desenvolupa en base a l'ús d'objectes distribuïts. Per a la comunicació entre els objectes s'utilitza CORBA//IIOP, DCOM o Java RMI. Aquesta arquitectura suposa la major carrega del procés i comunicacions.



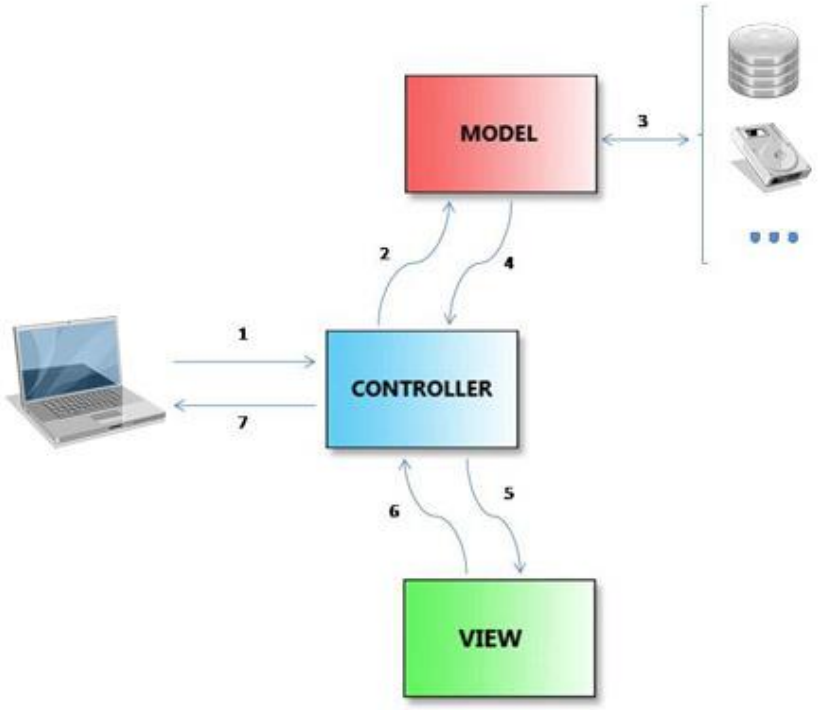
Il·lustració 3.11 Interfases a la arquitectura de tres nivells segons arquitectura client web

La elecció de l'arquitectura del client web es deu fer tenint present els aspectes de risc pel rendiment comentats en el problema del "Client Pesat". La capacitat de la xarxa de agafar la transferència de dades cap al client i la configuració hardware del mateix.

Quatre Nivells

Partint del model en tres nivells, s'han definit arquitectures derivades d'aquest patró. Es tracta de l'arquitectura de quatre nivells o més coneguda com a patró d'arquitectura Model Vista Controlador[5].

Aquest patró és probablement un dels més mencionats en el mon de la programació web aquests últims anys. Qualsevol que hi hagi treballat en algun projecte relacionat amb les aplicacions web haurà escoltat d'ell. Anem a analitzar les idees enrere del patró i els seus principals components amb l'objectiu de tenir las següents responsabilitats clarament separades:



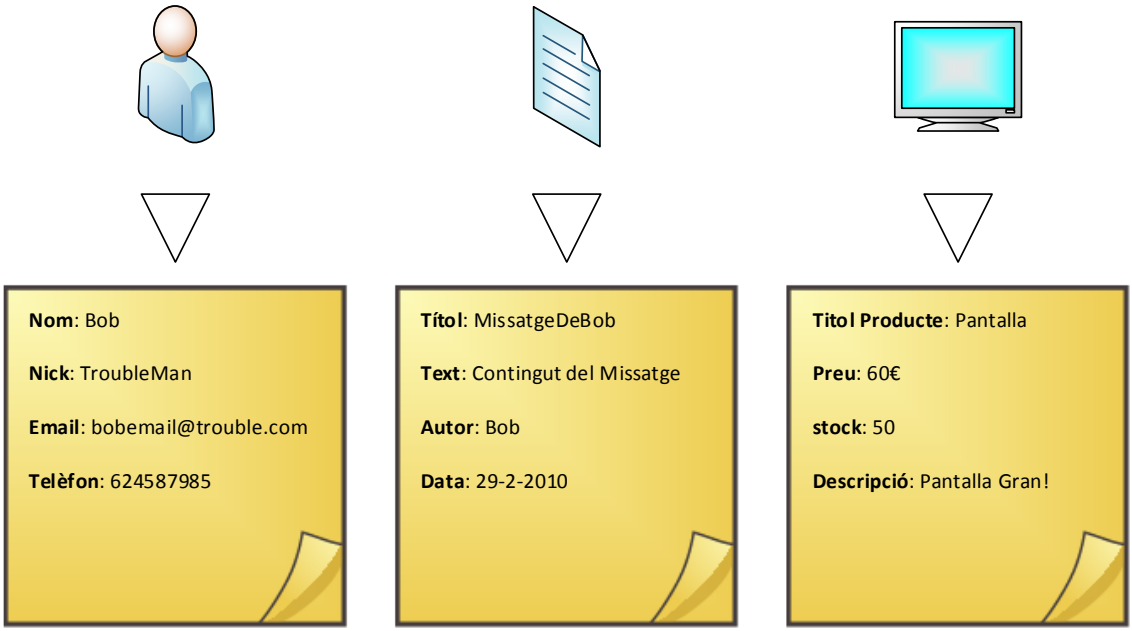
Il·lustració 3.12 MVC Vista general

Model > Gestió de les dades i lògica de negoci.

El Model són les dades i les regles que són aplicades a les dades, les quals, representen els conceptes que la aplicació manega. En qualsevol sistema software, tot es modelat com dades que són usades d'una certa manera. Com s'ha de representar un usuari, un missatge o un llibre per a una aplicació? Doncs en ultima instància es tracta de només dades que han de ser tractades d'acord a les regles especifiques, per entendre-ho millor, si per exemple, les dates de

[5] MVC Pattern, articulo Wikipedia
http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

aniversari són introduïdes al sistema no podem ser emmagatzemades en el sistema, no podran ser dates del futur, o un e-mail ha de portar un format específic 'nomusuari'@'DominiGestorEmail', limitacions en caràcters pels noms dels usuaris del sistema,...etc.



Il·lustració 3.13 Món Real v.s. Món de l'Aplicació.

El model dona al controlador la representació de les dades o lo que sigui que l'usuari està demanant (Productes, Comandes, Categories, ...etc.). Aquest model de dades serà el mateix, sense importar com nosaltres les presentem a l'usuari, es per això que nosaltres podem escollir qualsevol cista disponible per a "Renderitzar".

Aquest model conté la part més important de la lògica de la nostra aplicació, Una lògica que aplica a el problema que nosaltres estem tractant, ja sigui un bloc, fòrum, tenda de productes, un banc,.. etc. Per tant, el controlador contindrà una organització de lògica interna per a la aplicació com a tal, alguna cosa més com gestions internes.

Vistes > Presentar les dades del usuaris en qualsevol format i layout.

Les vistes proveeixen diferents maneres de presentar les dades rebudes del model. Poden ser plantilles allà on les dades són emplenades. Haurien de existir diferents vistes i l'encarregat d'escollir la vista adequada serà el controlador.

Una aplicació web està usualment composat d'un conjunt de controladors, models i vistes. El controlador ha de ser estructurat com un controlador principal, el qual rep totes les peticions i crides especifiques de controladores que controlen accions per cada cas.

Controladors > Rep les peticions dels usuaris i crida als recursos necessaris per dur-les a terme.

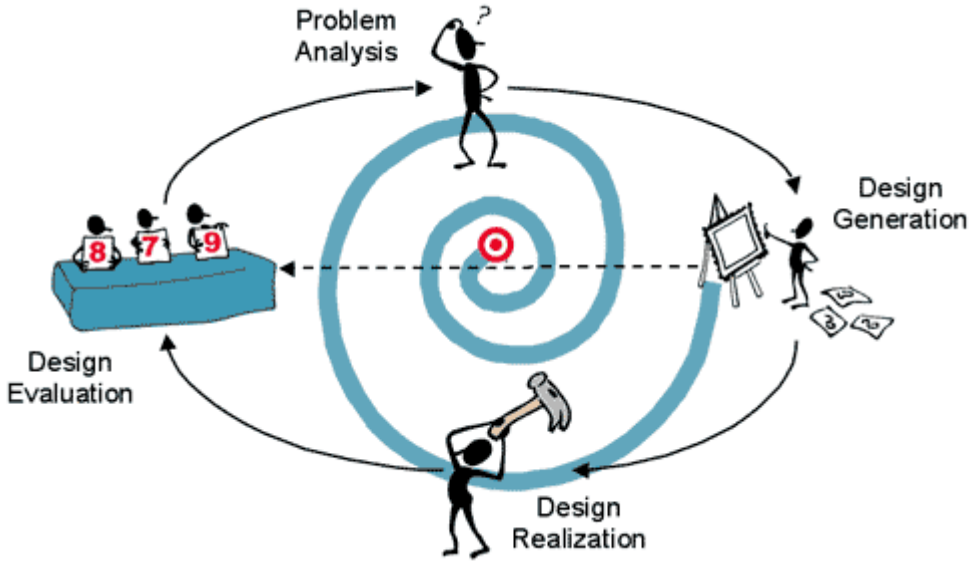
Aquest rol és l'encarregat de servir les peticions dels usuaris, en entorns web, en general són rebudes com peticions HTTP GET o POST quan l'usuari interactua amb els elements de la Interface gràfica d'usuari amb l'objectiu de executar accions permeses segons la lògica de negoci que s'hagi programat. Usualment el controlador serà el que cridarà al Model adequat seleccionant la vista adequada.

Conclusions

Observem que l'arquitectura tipus Quatre nivells, ens aporta molt bones avantatges, em quedaria amb l'avantatge més obvia, i es que utilitzant una arquitectura com MVC, guanyen una clara separació de la presentació i la lògica de l'aplicació.

A més, és suportada per diferent tipus d'usuari usant, diferents tipus de dispositius, la qual cosa, és un problema comú plantejat avui en dia. La Interface presentada ha de ser diferent si les peticions venen des d'un ordenador de sobretaula o si venen des d'un dispositiu mòbil. El model sempre contestarà exactament amb la mateixes dades, la única diferència és que el controlador escollirà una vista diferent per a ensenyar.

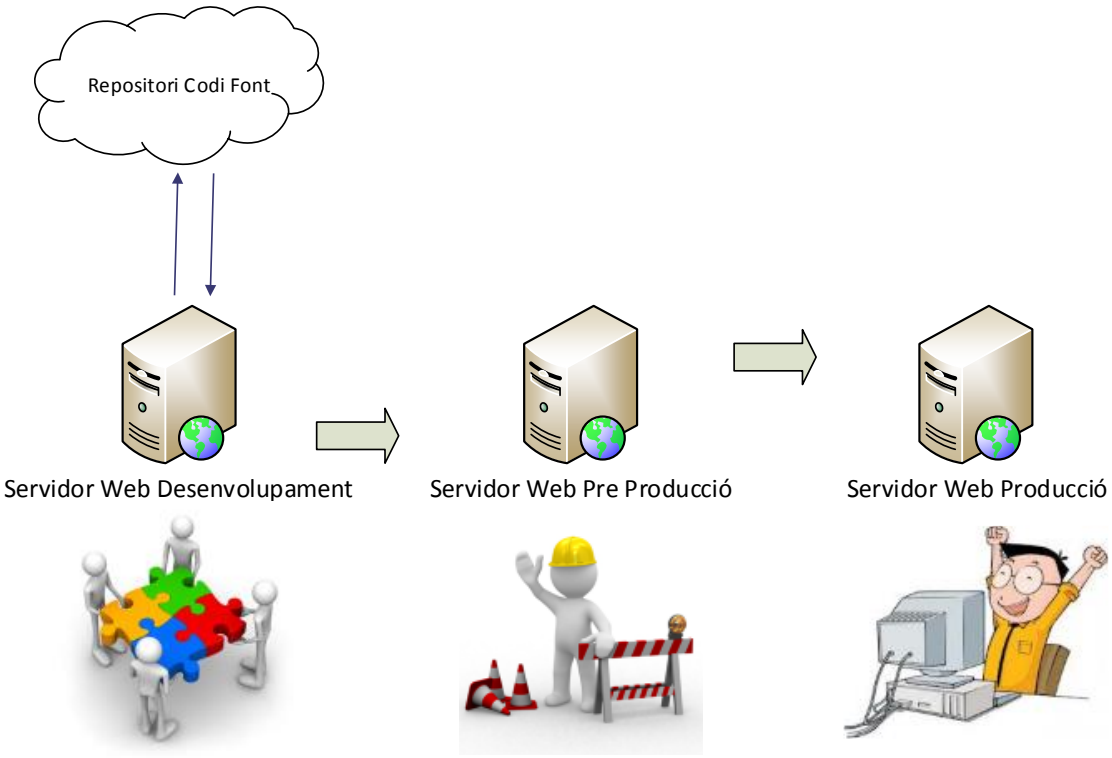
A més d'aïllar a la vista de la lògica de negoci, la separació MVC redueix la complexitat en el disseny d'aplicacions de grans dimensions. El codi és molt més estructurat i, per tant, més fàcil de mantenir, provar i reutilització. Si mirem capítol enrere, veurem que aquesta última cita, s'assembla molt i encaixa bastant bé amb els objectius que aquest projecte vol aconseguir. Per tant serà una de les decisions més clares del projecte a la hora de dissenyar i implementar el nostre software, el "com ho farem" el veurem al següent capítol.



Capítol 4. ANÀLISI

Arribat a aquest punt ja podríem dir que tenim l'estat mental idoni, en el qual tenim els conceptes adequats al cap per tal de desenvolupar una aplicació de las característiques que ens planteja aquest projecte. En aquest capítol podrem trobar una imatge conceptual del que serà la nostra aplicació en totes les seves facetes. Començant per la part d'arquitectura física, on s'ha decidit que residirà el codi i la seva administració, arribant fins la part d'arquitectura de software a un nivell més alt (estructuració del nostre sistema). Tot, basat en les necessitats de la nostra aplicació reflectida a les especificacions que podrà trobar a l'apartat 2.2 especificacions. Farem referència als apartats estudiats del Capítol 3, on mostrarem com aquest projecte interpreta aquest coneixements i els utilitza al seu favor per aconseguir les metes marcades.

4.1. DESPLEGAMENT DEL SOFTWARE



Il·lustració 4.1 Nivells Estructura Servidores

Per definir una estructura conceptual del nostre sistema de desplegament del software, aquest projecte ha decidit dividir-lo en tres mòduls o parts diferents, que estudiarem una per una amb les seves característiques associades, amb l'objectiu de marcar una directriu clara a l'hora d'escollir una implementació que s'adapti a les nostres necessitats.

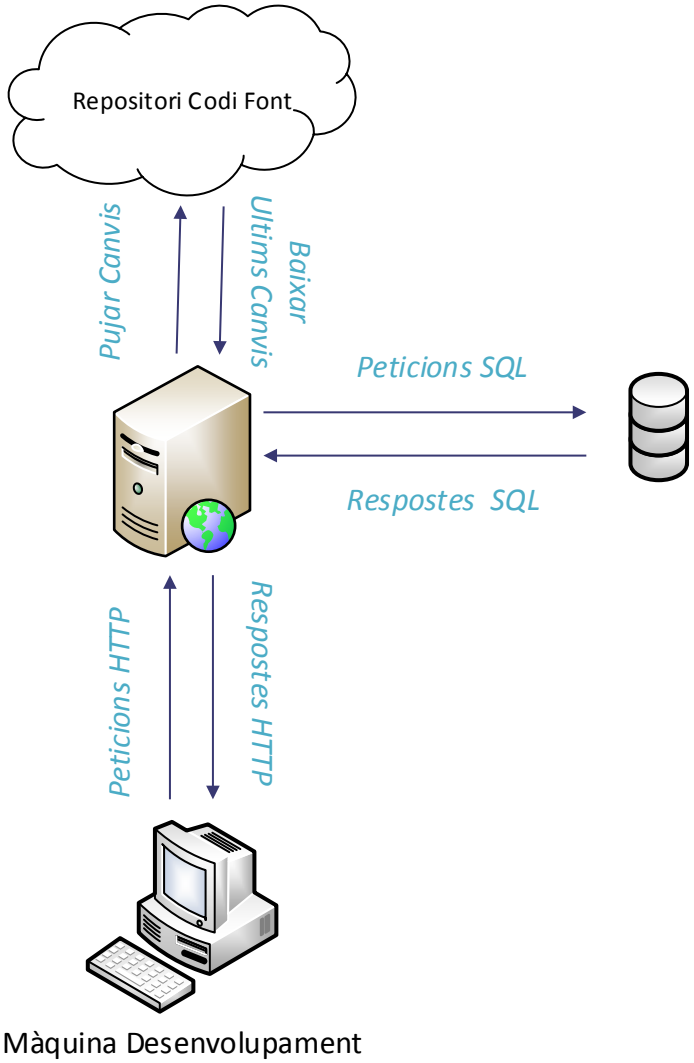
4.1.1. Servidor Web Desenvolupament i Repositori Codi Font

Aquest component és l'encarregat de simular un entorn real, però es diferent i amb menys restriccions que els altres dos. Al final del que es tracta és de definir un entorn de desenvolupament on podem veure i visionar els canvis, del codi font de la nostra aplicació, sense molta complicació. Per entendre més en detall que és el que es vol definint aquest component, anem a analitzar les característiques que té:

Característiques

- *Capacitat d'execució codi font:* Necessitem totes les llibreries que ens permetin l'execució de les instruccions escrites en el llenguatge de programació escollit per tal de la implementació.
- *Execució motor base de dades*: Com hem vist en els capítols anteriors, la nostra aplicació necessitarà accés a les dades emmagatzemades en una base de dades per tal de presentar-les i donar la possibilitat d'interactuar entre elles, per tant, es necessari llibreries per tal de administrar un sistema relacional de bases de dades.
- *Motor de Tractament de peticions HTTP*: Aquesta no és una característica particular d'aquest component, de fet, per naturalesa, és una característica que necessitarem a tots els servidors, ja que, com és evident al ser una aplicació web, que serà accessible des d'Internet.

- *Facilitat de configuració:* és una decisió que aquest projecte a pres, amb l'objectiu de simplificar la feina a realitzar.
- *Local:* L'accés haurà de ser restringit, per tant, no hauria de ser visible remotament. Al final del que es tracta, és de proveir al desenvolupador d'un entorn on pugui fer proves a nivell de programació mentre desenvolupa les diferents funcionalitats de l'aplicació.



Il·lustració 4.2 Descripció gràfica funcionament escenari desenvolupament

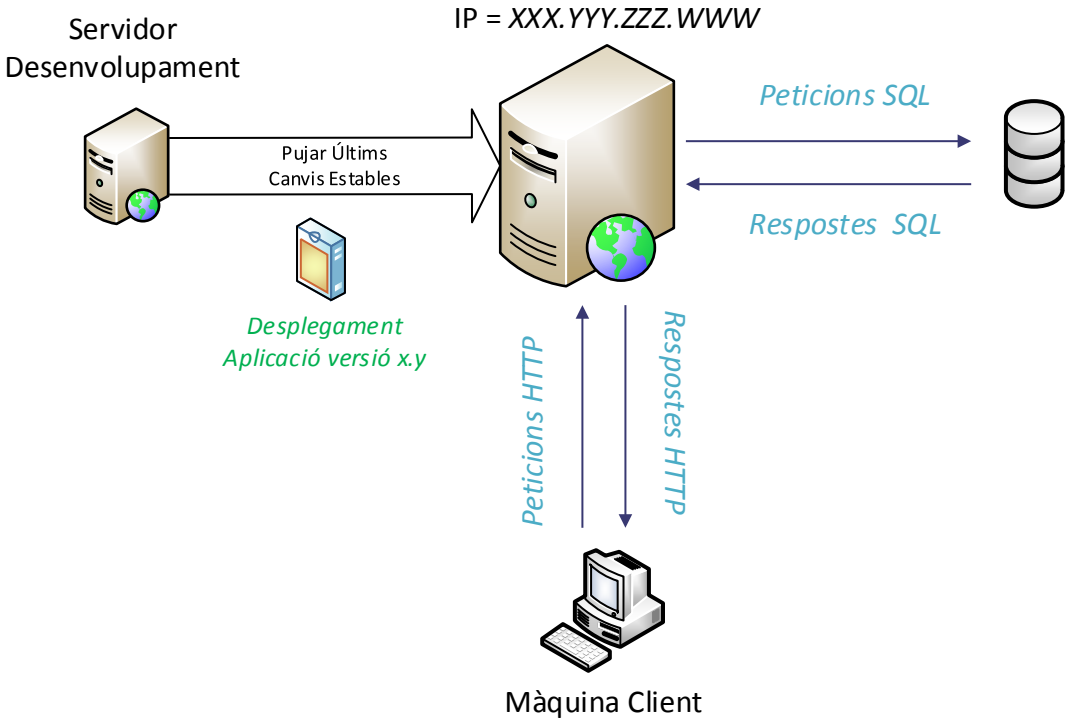
4.1.2. Servidor Web Preproducció

Aquest component és l'encarregat de proveir un entorn amb les mateixes característiques del servidor a on arribaran les peticions dels usuaris finals. L'objectiu principal és el de tindre un escenari de proves fiable, per tal de determinar si una versió concreta, desenvolupada del software és apta o no per ser desplegada en el servidor de producció.

Característiques

- *Capacitat d'execució codi font.*
- *Execució motor base de dades*.
- *Motor de Tractament de peticions HTTP.*

– *Accessible Remotament amb restriccions:* Al final del que es tracta es de donar accés a les funcionalitats del software ja assemblades. D'aquesta manera sabrem que és un entorn que es comportarà igual que quan estigui en el servidor de producció. Això pot ser útil per fases de Proves de l'aplicació o demostracions i/o revisions de la lògica de negoci amb el client.



Il·lustració 4.3 Descripció gràfica funcionament escenari Preproducció

4.1.3. Servidor Web Producció

Finalment ens queda la etapa final, aquest component serà l'encarregat, com ja hem dit, de rebre, tractar i respondre les peticions dels usuaris finals. La única diferència entre el servidor anterior és que l'accés haurà de ser remot i sense restriccions. Això implicarà tindre un domini associat a la adreça física d'aquest servidor.

4.2. ESTRUCTURACIÓ I EMMAGATZEMATGE DE DADES

Una vegada tenim clar l'estructuració que necessitem per emmagatzemar la nostra aplicació software i les seves dades associades, ara es necessari definir l'estructuració de les dades que necessita la nostra aplicació, és a dir, com les guardarem per a que siguin accessibles des de la nostra aplicació. Com hem vist al Capítol 3, hi ha explicada, una manera sistemàtica de definir aquest procés i es que el que aquest projecte usarà per definir la base de dades.

Abans de continuar es important tindre present l'apartat 2.2.4.1 de Bases de Dades definit les especificacions del Capítol 2, que es d'on traurem la informació per definir les entitats i relacions de la base de dades de la aplicació que vol implementar aquest projecte.

4.2.1. Disseny base de Dades

4.2.1.1. Anàlisi dels requeriments

Aquesta és la fase inicial, que consisteix en una recol·lecció i Anàlisi de la informació provinent del client, per tal de poder estructurar la nostra base de dades. En el nostra cas, la recol·lecció d'informació, ja ha estat feta com hem dit abans. Ara el que ens falta és la part en que nosaltres, com a dissenyadors de bases de dades, fem un Anàlisi tota aquesta informació detalladament, per tal de traduir-la a un sistema relacional, que finalment, serà traduir en sentencies SQL i executades i administrades des d'un sistema d'administració de base de dades (DBMS).

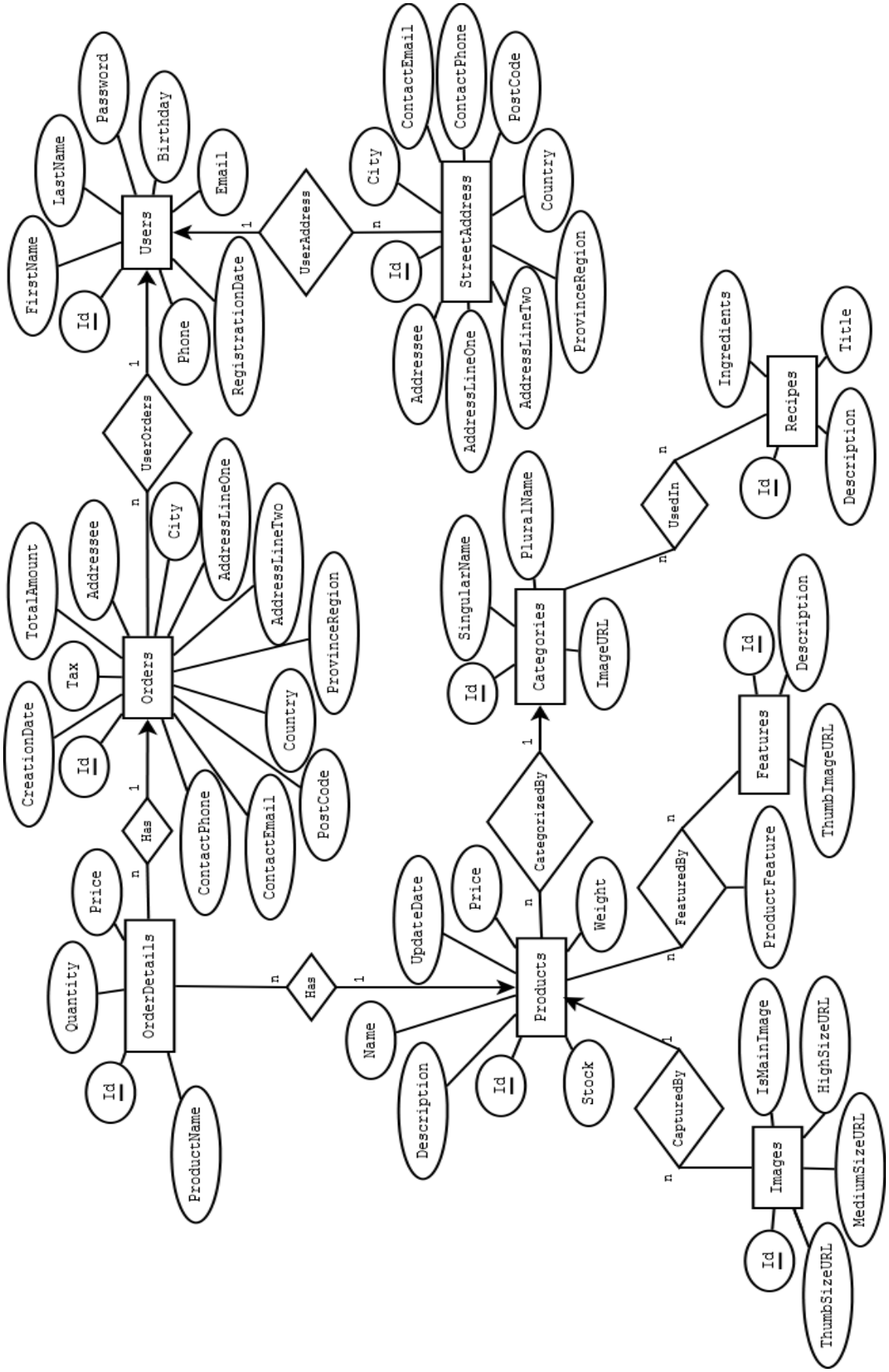
S'han detectat un total de 8 Entitats i són les següents:

1. Productes.
2. Categories de producte.
3. Característiques de Producte.
4. Imatges de producte.
5. Comandes.
6. Detall de les comandes.
7. Usuaris registrats.
8. Direccions .
9. Receptes de cuina.

Seguidament s'han detectat un total de 8 Relacions

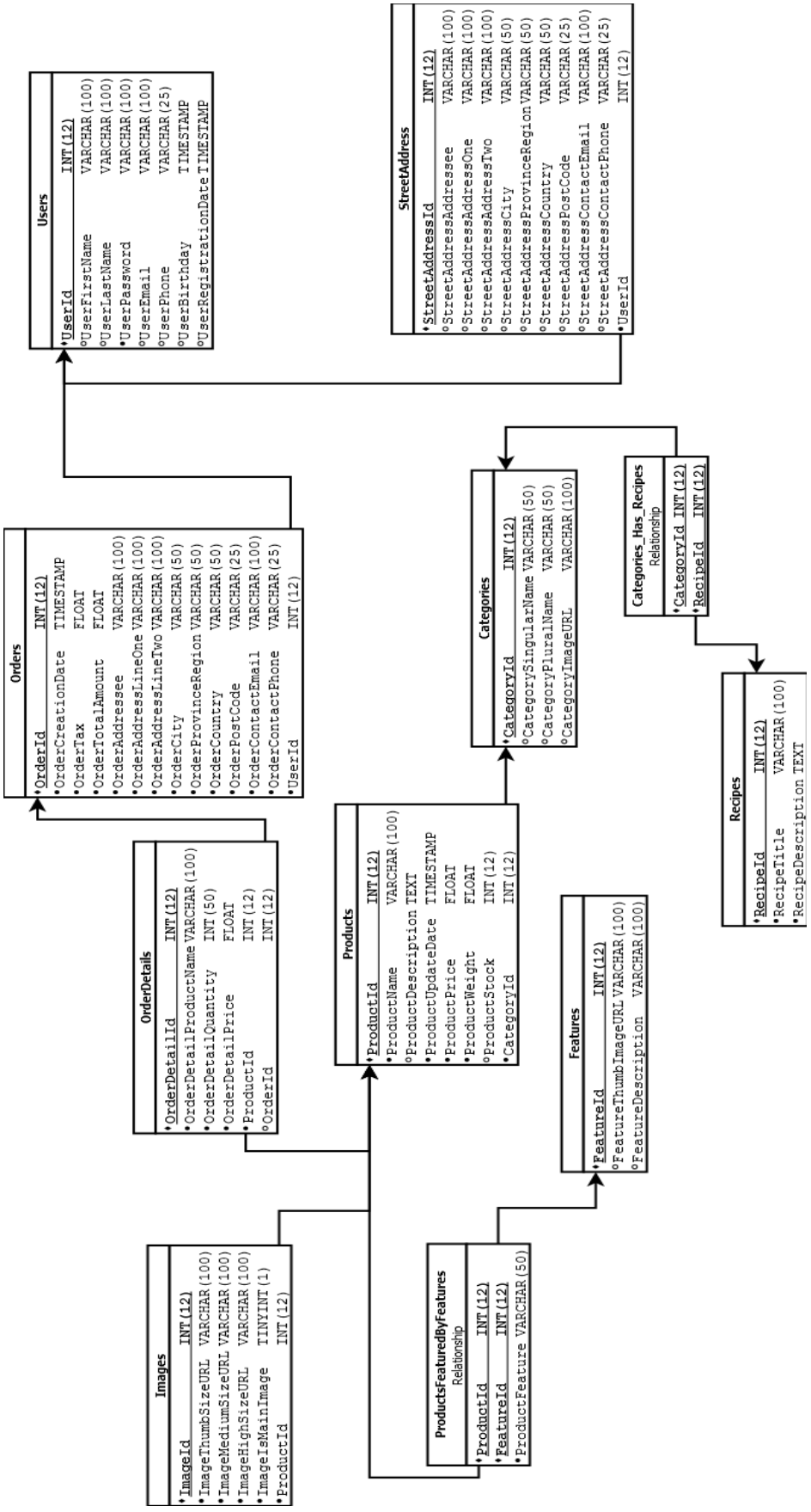
1. Un Producte pot estar categoritzat en una única categoria, però en canvi les categories poden tindre un o més productes relacionats. *Cardinalitat **(N:1)***.
2. Una característica de producte pot tindre relacionat un o més productes i un producte pot tindre un o més característiques associades a ell. *Cardinalitat **(N:M)***.
3. Una imatge pot estar associat a un únic producte, en canvi un producte pot tindre associat un o més imatges. *Cardinalitat **(N:1)***.
4. Una categoria pot tindre una o més receptes i una recepta pot ser usada en una o més categories. *Cardinalitat **(N:M)***.
5. Els detalls d'una comanda poden pot tindre un únic producte associat, en canvi, un producte pot estar en un o més detalls de comandes. *Cardinalitat **(N:1)***.
6. Una comanda pot estar dividit en un o més detalls de comanda, en canvi un detall de comanda només pot estar associat a una única comanda. *Cardinalitat **(1:N)***.
7. Un usuari pot fer un o més comanda, però una comanda només pot pertànyer a un únic usuari. *Cardinalitat **(1:N)***.
8. Un usuari pot tindre una o més direccions d'enviament associades, en canvi, les direccions associades, només poden pertànyer a un usuari. *Cardinalitat **(1:n)***.

4.2.1.2. Disseny Conceptual



Il·lustració 4.4Entitat Relació

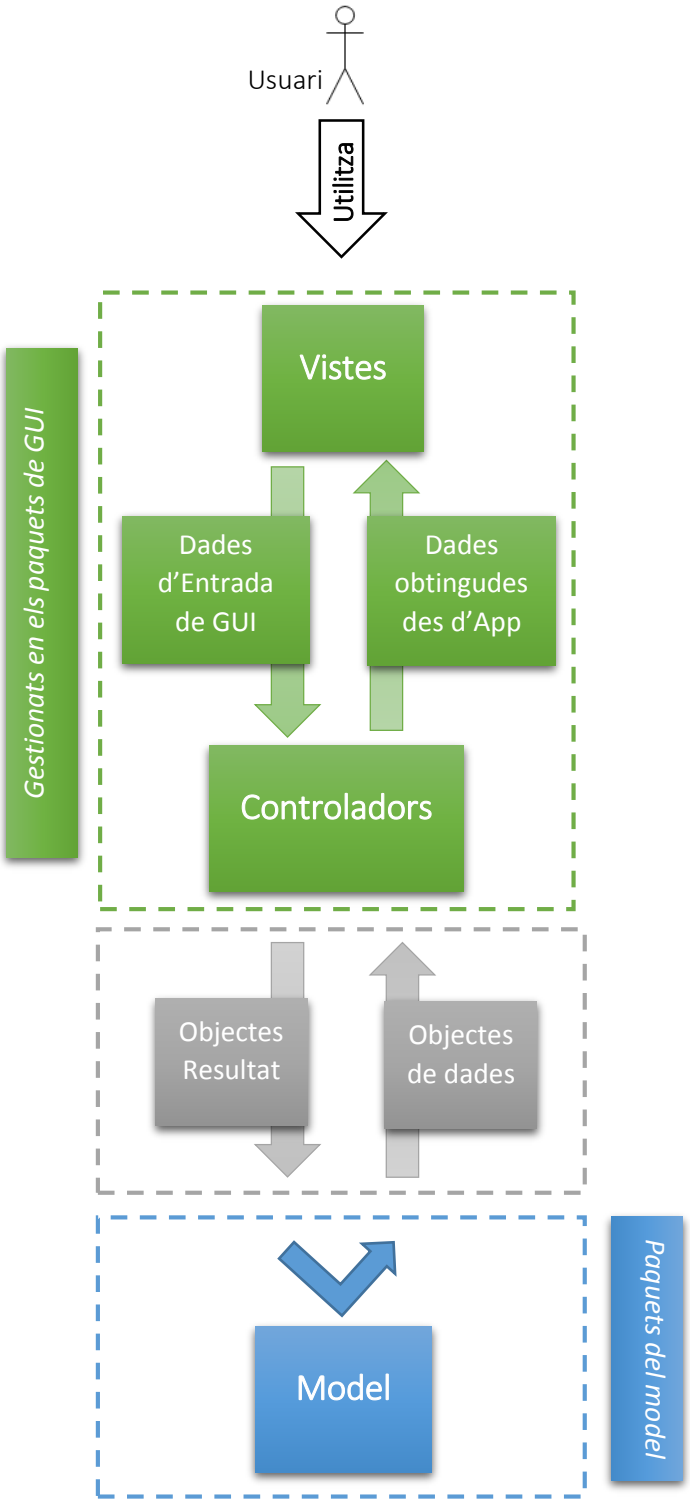
4.2.1.3. Disseny Relacional



Il·lustració 4.5 Esquema Relacional

4.3. DISSENY DE SOFTWARE

4.3.1. Anàlisi Model Vista Controlador



Il·lustració 4.6 Esquema Funcions i components del patró MVC

Com observem a la figura anterior, el patró és senzill. Es basa en el principi de "Divideix i conqueriràs!". Ja hem estudiat en el capítol anterior les seves característiques, que són les que necessitem per desenvolupar la nostra aplicació. Per tal d'entendre com funciona internament anem a fer un Anàlisi de robustesa, on veurem més detalladament el funcionament.

Anàlisi de robustesa

Un anàlisi d'aquest tipus implica analitzar el text narratiu de casos d'us, identificació d'una primera aproximació del conjunt d'objectes que participen en aquest casos d'ús i la classificació d'aquest objectes en base de les funcionalitats que exerceixen. Això ajuda a la creació de particions en els objectes dins d'un paradigma Model Vista Controlador.

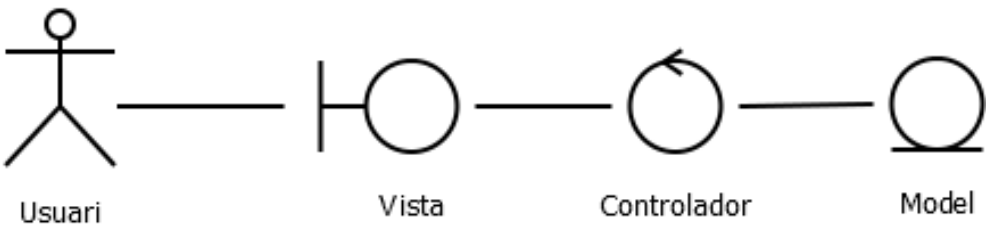
L'anàlisi de robustesa permet el descobriment continu d'objectes i ens ajuda a assegurar que estem indicant la majoria de les principals classes de domini abans d'iniciar qualsevol disseny o desenvolupament addicional. Aquestes classes poden ser classificades com:

1. *Entitat:* no són més que la informació o de les dades dels objectes de frontera. Aquests podrien ser taules de les basses de dades, arxius excel,... etc. O fins i tot podrien ser sessió de transitoris o les dades de emmagatzemades a la memòria cache.
2. *Frontera:* Són els objectes amb els quals els actors es comuniquen en nostre sistema de software. Aquest poden ser qualsevol tipus de finestres, pantalles, quadres de diàleg i menús o altres interfícies d'usuari en el sistema.
3. *Objectes de control:* o controladors, són els objectes de negoci. Aquí és on es poden captures les regles de negoci que s'utilitzen per filtrar les dades que es van a presentar a sol·licitud de l'usuari.

Com observem la relació amb el paradigma MVC amb els objectes derivats d'aquest anàlisi es de **un a un**:

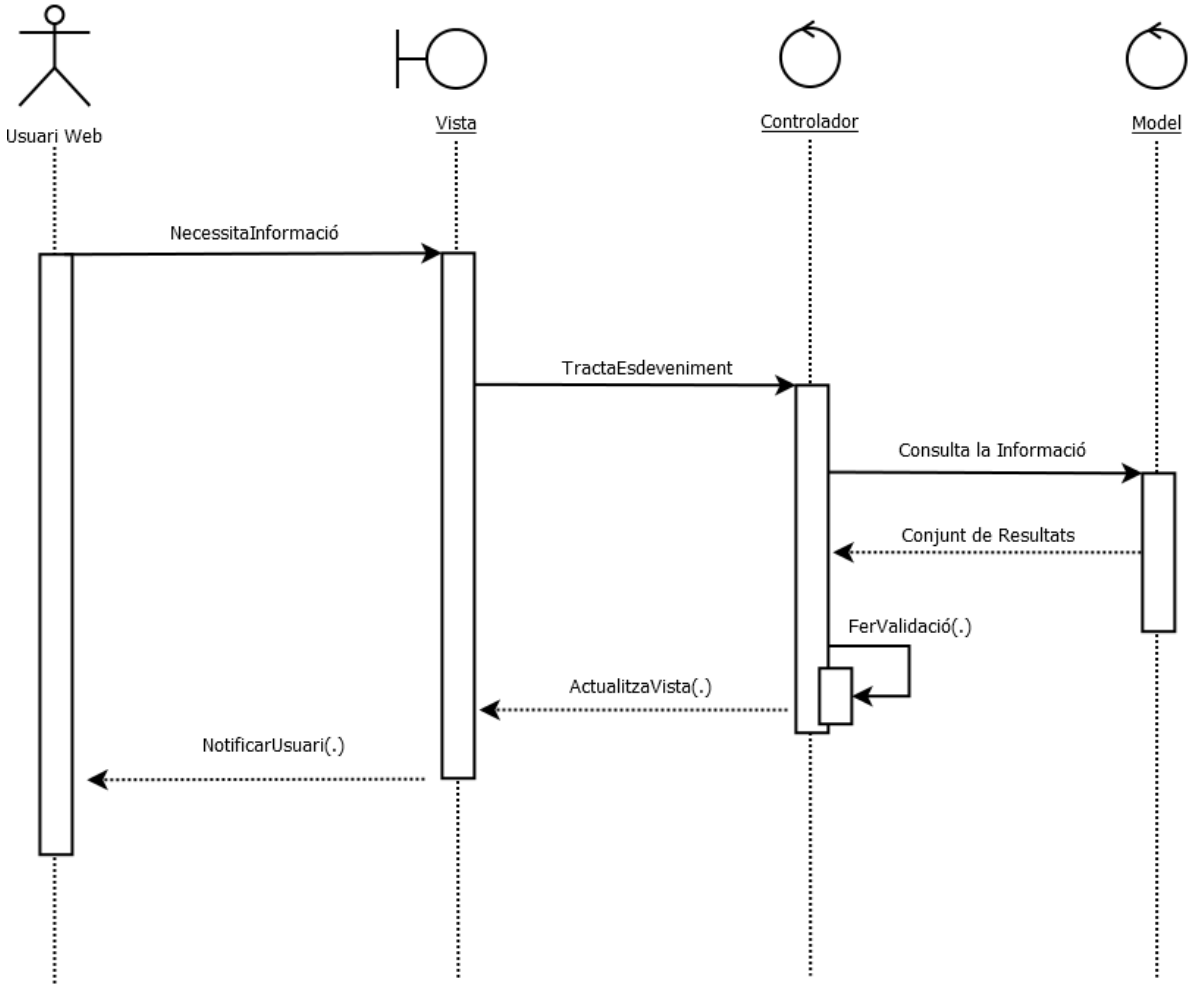
- Objecte entitat ↔ Model.
- Objecte Frontera ↔ Vista.
- Objecte de Control ↔ Controlador.

Per tant apliquen les mateixes regles i és més fàcil el seu anàlisi. Anem a veure quin és el seu diagrama de regles:



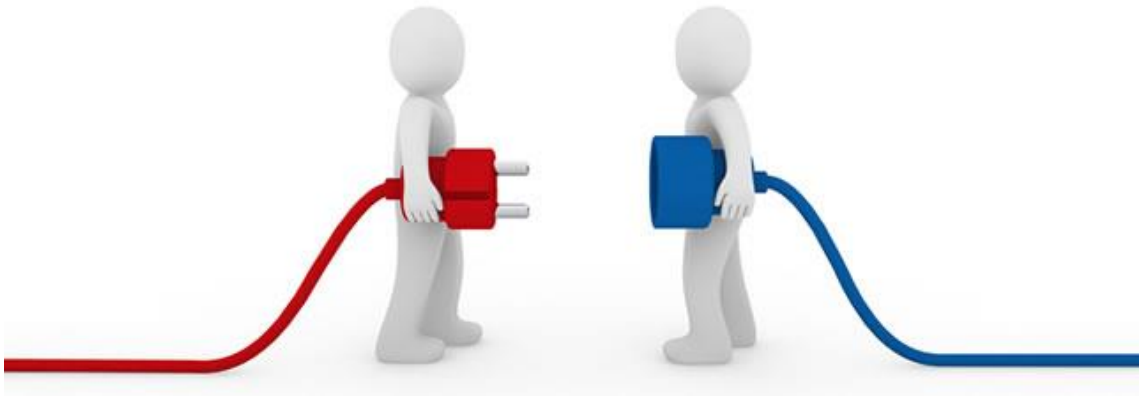
Il·lustració 4.7 Regles Casos d'ús MVC

Per tal de simplificar la explicació del diagrama de seqüència del MVC. Anem plantejar un cas genèric i hipotètic, però que en el nostre entorn web serà molt comú i tot que no serà el mateix, totes les implementacions seguiran, si no el mateix, un patró bastant similar. Imaginem que un usuari web inicia una consulta. Aquesta consulta genera un esdeveniment que és manejat pel controlador, que obté la informació necessària a partir de la instancia d'un model. Aquest valida la informació i retorna un conjunt de resultats a la vista.



Il·lustració 4.8 Diagrama Seqüencia MVC Hipòtesi

Com ha havíem comentat MVC, és un patró d'arquitectura per la segregació de les diferents capes, de tal manera que pot ser fàcil de mantenir per una aplicació, ja sigui moderada o molt complexa.



Capítol 5. IMPLEMENTACIÓ

En aquest capítol vostè, el lector, es podrà trobar amb el resultat d'aplicar els fonaments teòrics i l'anàlisi en el nostre projecte, remarcant les diferents tecnologies que aquest projecte ha utilitzat per tal de desenvolupar el comerç electrònic que té com objectiu.

5.1. DESPLEGAMENT DEL SOFTWARE

5.1.1. Servidor Web Desenvolupament i Repositori Codi Font

Servidor web Desenvolupament

Il·lustració 5.1 Logotips XAMPP i els seus components

Per tal de definir l'entorn de desenvolupament, aquest projecte ha utilitzat una solució multi-plataforma de servidor web anomenada XAMPP[6]. El nom és un acrònim dels components que conté:

- **X**, ve de l'anglès "cross" que es refereix a al terme "cross"-plataforma, o lo que és el mateix en català, multi-plataforma.
- **A**pache HTTP Server, que és el seu motor de tractament de peticions HTTP.
- **M**ySQL, que és el motor d'administració basses de dades que utilitza.
- **P**HP, Aquest servidor conté els intèrprets necessaris per tal de executar Scripts d'aquest llenguatge.
- **P**earl, i també els necessaris per Perl.

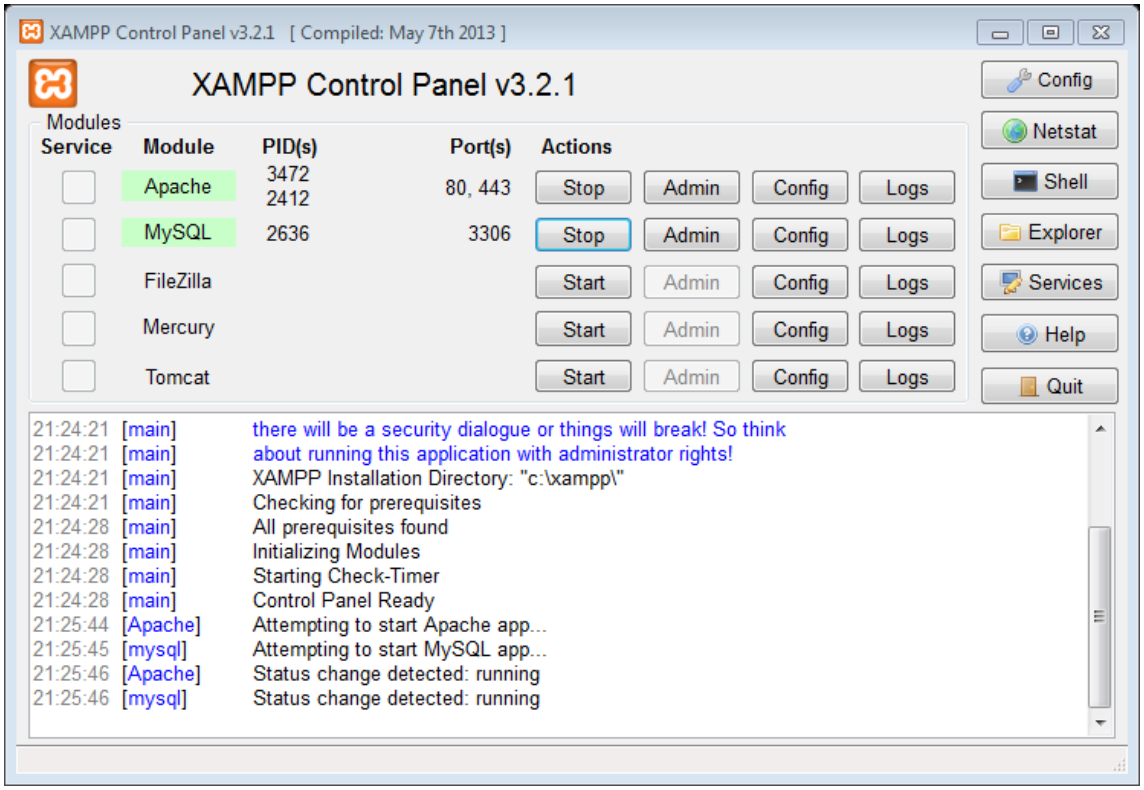
Característiques i Requeriments:

Per tal de fer us d'aquest servidor web els únics requeriments de la possibilitat de descomprimir fitxers *zip, tar, 7z o Executables de Windows*. El que respecta a la configuració, per defecte, no és necessari fer cap. Però et dona la possibilitat d'accedir als fitxers de configuració tant d'*Apache*, com *MYSQL*, *PHP* i *Pearl*. Inclou mòduls tals com *OpenSSL*[7] i *PHPMyAdmin*[8].

[6] XAMPP Apache + MySQL + PHP + Perl
https://www.apachefriends.org/index.html
[7] OpenSSL Pàgina Oficial
https://www.openssl.org/
[8] PHPMYadmin Pàgina oficial
http://www.phpmyadmin.net/home_page/index.php

Funcionament:

XAMPP està orientat en aquest projecte com a eina de desenvolupament, proveint un escenari per fer proves del codi de la aplicació sense la necessitat d'accés a internet, amb l'objectiu de fer la programació el més còmode possible. Aquesta comoditat ve donada, per la poca configuració que s'ha de fer a aquest servidor. Ara bé això es fa d'aquesta manera degut al seu objectiu que és el de només veure els canvis de manera instantània.



Il·lustració 5.2 XAMPP, Panell de control

Com observem a la figura anterior, la posta en marxa es prou intuïtiva. El panell de control, per defecte ens presenta tots els mòduls que té instal·lats, amb el seu panell d'accions associats que ens permetrà la seva administració. Per la nostra aplicació només serà necessari els mòduls Apache (motor de peticions HTTP) i Mysql (Motor d'administració de base de dades relacional).

Les peticions es fan *localhost* (127.0.0.1), el servidor ja està preparat per redirigir les peticions en funció del directori que s'especifiqui a la URL. Aquesta aplicació, és necessari que estigui en el directori d'instal·lació de XAMPP, en una carpeta anomenada *htdocs.* Per exemple, si accedim a la següent direcció:

- *http://localhost/productesdecasaapp*

Si el directori "Productesdecasaapp" esta creat en el directori htdocs, El servidor està configurat per cercar el fitxer *index (Pot ser un Script php o html)* per defecte que trobi i l'executarà, enviant al client la pàgina resultat.

Control de versions: GIT

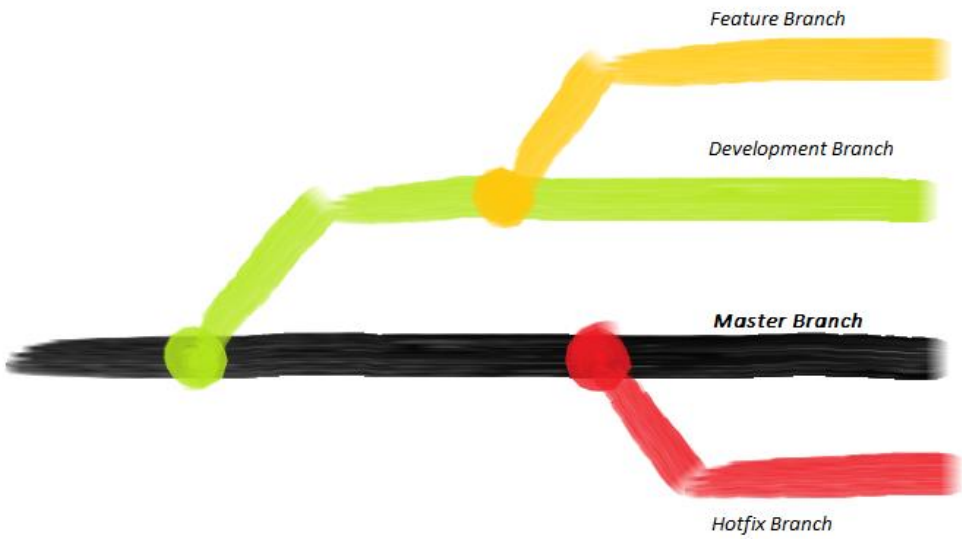
Il·lustració 5.3 Logotip Git[9]

Git és un software de control de versions orientada al manteniment de codi font d'una aplicació, amb un numero elevat de fitxers. Les característiques principals que ens aporta són les següents:

Característiques i Requeriments:

- Gestió àgil de branques: inclou eines especifiques per navegar i visualitzar un historial de desenvolupament no lineal. En el meu projecte és de gran ajuda sobretot per la consulta de versions del codi anteriors, amb l'objectiu de recuperar canvis perduts.
- Gestió Distribuïda: podria semblar que aquesta característica no té molt sentit en un projecte desenvolupat per una sola persona, però és molt útil sobretot per que em dona la independència en quan a la ubicació de desenvolupament. Basta tindre una maquina per tal de descarregar els últims canvis i localment generar uns de nous, per tal de pujar-los al repositori d'una branca en concret.
- Gestió eficient per projectes grans: donada la rapidesa de gestió del procés de diferenciació entre els fitxers, entre d'altres millores d'optimització de fitxers. Aquest projecte en una primera versió, la diferencia de velocitat respecte altres sistemes de controls de versions no serà prou considerable, no obstant, una aplicació d'aquest tipus dona peu a un creixement ràpid, i és quan aquest sistema ens aportarà més.

Funcionament



Il·lustració 5.4 Branques Productes casa App

[9] Informació àmplia Git
http://git-scm.com/book

Aquest projecte ha definit, seguint les bones pràctiques, la possibilitat de definit quatre tipus de branques que anem a descriure a continuació

1. Master Branch: Només es podrà publicar codi a producció hagi passat estable per aquesta branca. Per tant és la nostra rama principal
2. Development Branch: com veiem a la figura anterior, es treta de la rama *Master*, es també coneguda com una rama d'integració, cada funcionalitat nova ha de se integrat en aquesta rama, la idea es que quan es tingui una versió que puguem dir, és "estable", es fa un *merge* de *Development* contra *Master*
3. Features Branch: Es arribar a molt baix nivell, però quan es tenen projectes molt grans son bastant útils. Consisteix en per cada nova funcionalitat molt concreta, es fa una rama des de development, i quan es té el canvi finalitzar es fa un merge contra el master.
4. Hotfix Branch: Aquesta última, serveix per a errors o "bugs" que surten de manera esporàdica a producció i és necessari arreglar i publicar de forma urgent. Una vegada es té el canvi fet, s'haurà de fer un merge des de Hotfix contra el master.

Cal aclarir, que aquestes dues ultimes (Features i Hotfix branch), no han estat utilitzades per aquest projecte. Per si finalment és fa una publicació del codi en futures versions, s'han de considerar ja que aporten molta informació respecte el nostre codi, a més de ser prou útils, per al nostre desenvolupament.

5.1.2. Servidor Web Preproducció i Producció

Per la implementació del servidor de preproducció és necessari tindre un sistema de característica iguals sinó bastant similars a les del servidor de producció. Això és necessari per tal de tindre un escenari de proves prou fiable com per a saber, si en un entorn de preproducció funciona correctament, el codi funcionarà igual al servidor de producció. Tot i així, amb l'objectiu de reduir despeses, el servidor de preproducció serà simular de manera virtual.

Llavors primer anem a veure quines son les característiques amb les que s'han implementat aquest servidors i finalment, explicarem com farem la simulació virtual del entorn de preproducció.

Requeriments:

Com sabem hi ha tres mòduls necessaris per fer funcionar el nostre codi, Motor de peticions HTTP, Motor d'administració de base de dades i capacitat d'execució scripts del llenguatge de programació que s'està utilitzant en el nostre porjecte, a continuació veurem quins han estat els escollits per aquest projecte, descrivint les seves característiques principals, però la quals han estat seleccionats:

- Nginx, com a motor de peticions HTTP.
- MySQL, Com a motor d'administració de basses de dades.
- PHP com a intèrpret de llenguatge de programació.

Com podem veure la única diferencia respecte el nostre sistema de desenvolupament, és el **motor de peticions HTTP**, això té una raó i la veurem a continuació.

Nginx

Il·lustració 5.5 Logotip NGINX[10]

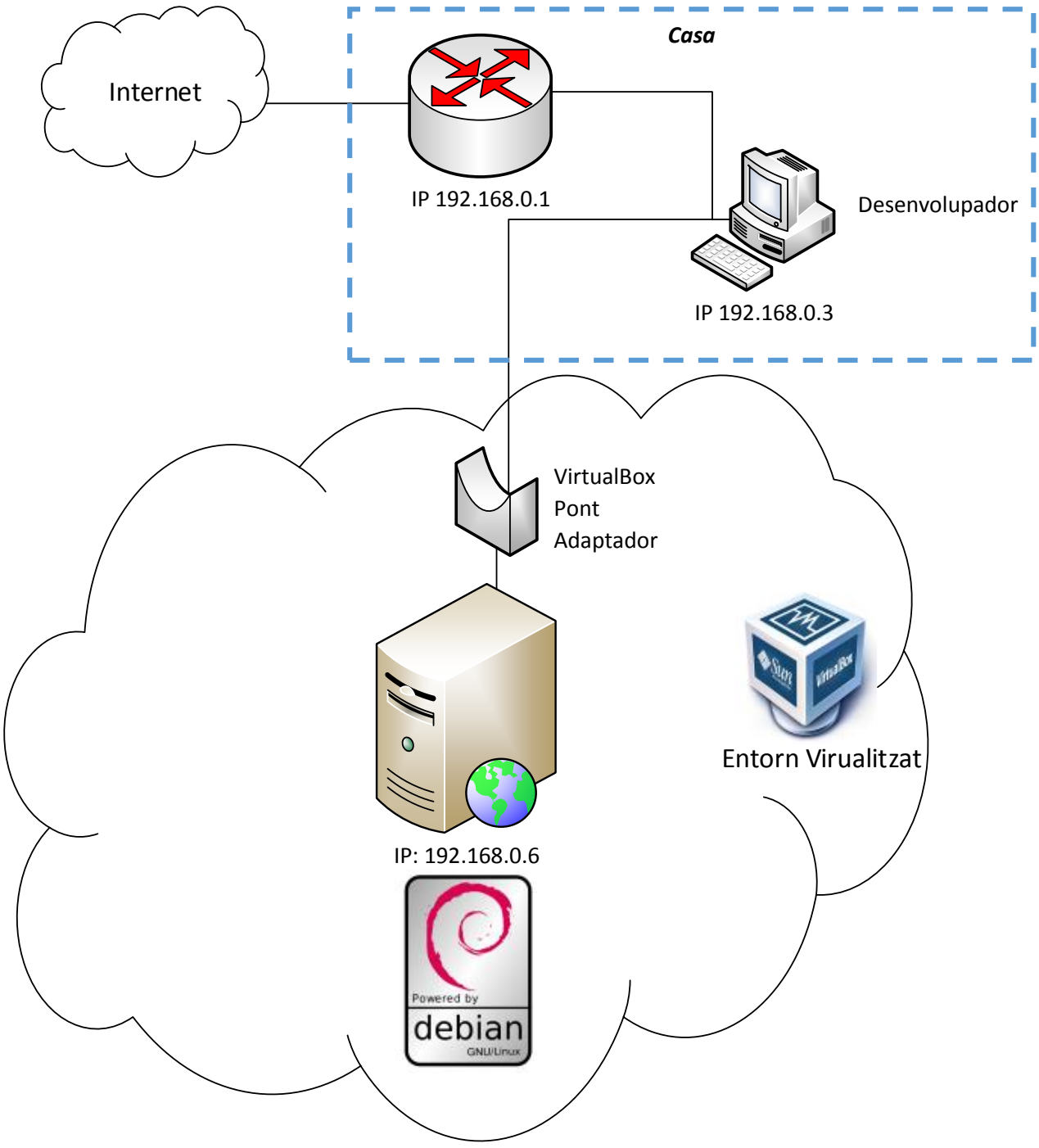
Es tracta d'un servidor web/Proxy invers lleuger d'alt rendiment. Sent software lliure i de codi obert, és una plataforma que es pot corre en sistemes tipus Unix, tals com poden ser GNU/Linux, BSD, MAC OS X,...etc.). Tot sigui dit, és una eina que necessita una configuració molt detallada, però que ens doni el millor rendiment possible respecte una arribada de peticions massives al nostre sistema. No obstant aquest projecte atractiu totes les funcionalitats que ens aporta sobretot per a futures versions que estigui a internet. Anem a veure les característiques mes destacades:

- És un servidor d'arxius estàtics, índexs i auto-indexat.
- Proxy invers amb opcions de memòria cau (Memòria principal / Cache).
- Balanceig de carrega.
- Tolerància a fallades.
- Suport HTTP en SSL.
- Suport per l'autenticació.
- Habilitat per suportar més de 10.000 connexions simultànies.

[10] per més información relacionada amb ngix consultar http://nginx.org/

Funcionament:

En aquest cas, s'ha utilitzat com a eina de virtualització Oracle VM VirtualBox, donada una imatge Debian, es crea una maquina bàsica, a la qual s'instal·len les llibreries necessàries, mencionades anteriorment. L'accés és remot gràcies a que li podem assignar una IP, mitjançant l'adaptador tipus pont que ens proveeix Virtual box.



Il·lustració 5.6 Esquema de xarxa Desenvolupament

5.1.3. Servidor Web Producció

Hem vist a l'apartat anterior l'estructura del servidor de preproducció, i com hem dit, és una estructura que a nivell de sistema ha de ser el màxim similar possible, amb l'objectiu de tindre una fiabilitat bastant alta, quan nous canvis són integrats en servidor de producció. Tot i així, hi ha una diferència, es tracta de del hosting de l'aplicació. En aquest cas es vol un accés des d'Internet, per tant és necessari una DNS que pugui traduir la adreça IP del nostre servidor de producció.

Per aquest projecte s'ha aprofitat el contracte personal que el client té per la seva pàgina personal, amb una empresa de hosting de servidors. Com que només es té un domini llogat és necessari trobar una solució. És per això que arribem al concepte de Hosts Virtuals[11].

Hosting Virtual

Un host virtual és com s'anomena a un "Servidor" virtual dins d'un servidor real. Per exemple, si nosaltres tenim un servidor per allotjar les nostres pàgines web, primer tenim configurat el servidor per a que respongui a les peticions que se li fan des de fora, i després dins del servidor hem de crear un "servidor virtual" que respongui a un determinat domini o subgrup de dominis. Aquest servidor virtual, que no es un servidor real com a tal, el que està fent es la creació d'un entorn sobre aquest domini, on es pot definir el directori font de la aplicació en qüestió, permisos d'accés,.. etc.

La seva utilització té diferents maneres de treure profit, per exemple, en servidors amb carrega elevada de peticions, és molt importat implantar un escenari de balanceig de carrega, això el que significa al final, es que hem de donar un mateix nom de domini a diferents maquines, i que, quan arribi una petició serà atesa per la primera màquina que estigui disponible. De la mateixa manera, a vegades es té la necessitat de donarle diferents noms a la mateixa màquina, aquest cas es pot donar, per exemple, quan tenim diferents noms de domini que, cada vegada que es faci peticions, arribaran a la mateixa aplicació.

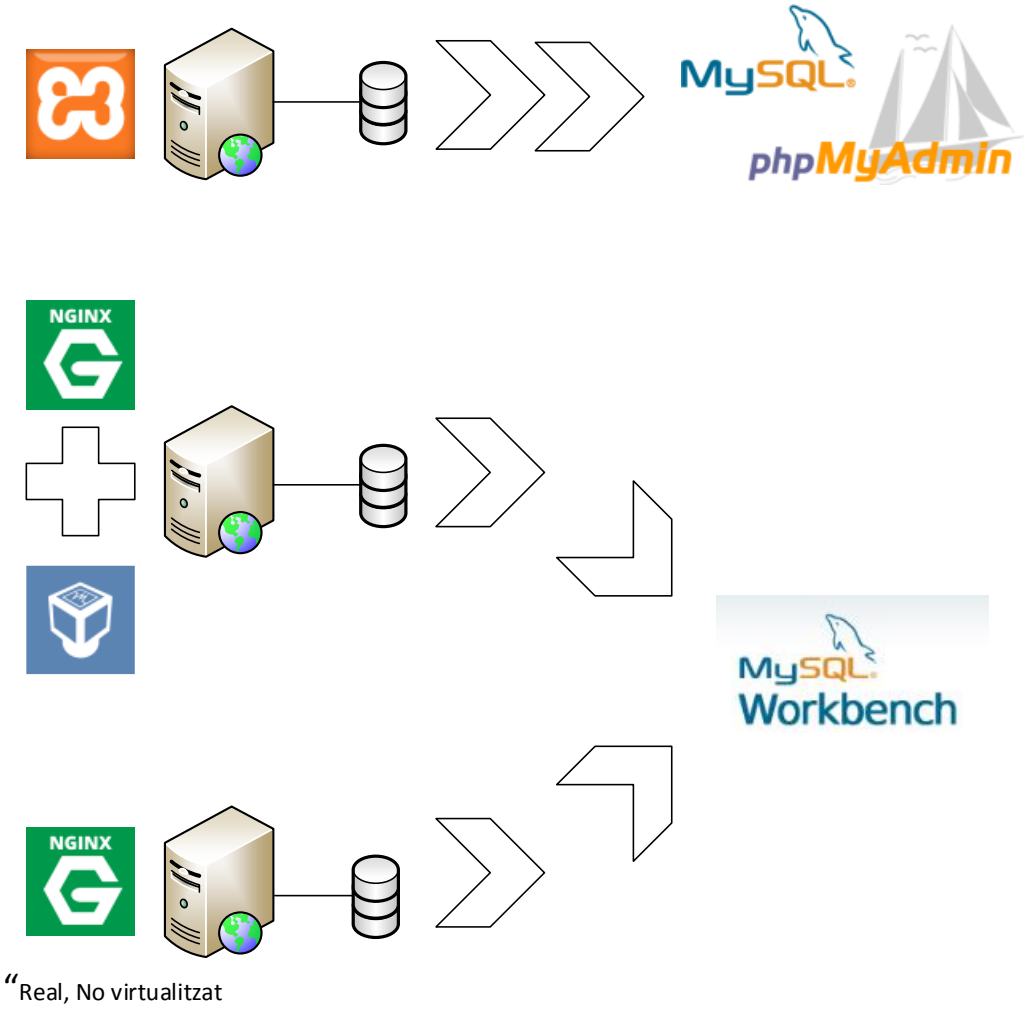
Il·lustració 5.7 Exemple de subdominis web

En la il·lustració anterior, veiem un exemple gràfic d'una de les avantatges que ens dona la virtualització de host per un domini web. I Aquest serà cas que aquest projecte en traurà profit, ja que ens proveeix la possibilitat d'aprofitat un domini ja existent, per definir un

[11]Hosting Virtual Informació complementària.
http://en.wikipedia.org/wiki/Virtual_hosting

subgrup de dominis. El nom del domini que es té és: www.champinet.com. La nostra aplicació serà atesa pel subdomini app.champinet.com . Aquest subdomini, tindrà una regla especifica dintre de la configuració del notre servidor, que cridarà els fitxers indeix de la aplicació i exectarà el seu codi, retornant una resposta cap el client.

5.2. ESTRUCTURACIÓ I EMMAGATZEMATGE DE DADES



Il·lustració 5.8 Tecnologies Basses de dades

Si observem la figura anterior, aquest seria el nostre entorn de base de dades, d'aquesta manera gestionarem el que respecta a l'estructura de base de dades. Al final tot té el mateix motor de base de dades, la única diferència és que accedim amb eines diferents en funció de les característiques de cada escenari. Com és evident, en el entorn de desenvolupament, ja ens proporciona una Interface gràfica prou útil gracies a la eina que s'utilitza. En canvi, pels servidors de producció i preproducció, amb l'objectiu de no sobre carregar la màquina amb per exemple un phpMyAdmin, tot es gestiona amb comandes Linux. I per fer més fàcil l'administració, aquest projecte ha utilitzat MySQL Workbrench[12].

[12] Per més informació de la eina i les seves característiques.
http://www.mysql.com/products/workbench/

Una vegada tenint clar quin serà el nostre entorn d'administració, podem agafar el modelatge dissenyat al *4.2 Estructuració i emmagatzematge de dades*. Com a resultat tenim un esquema de base de dades configurat, el qual és capaç de ser exportat com a script SQL i el qual serà el nostre Script de configuració d'instal·lació de base de dades.

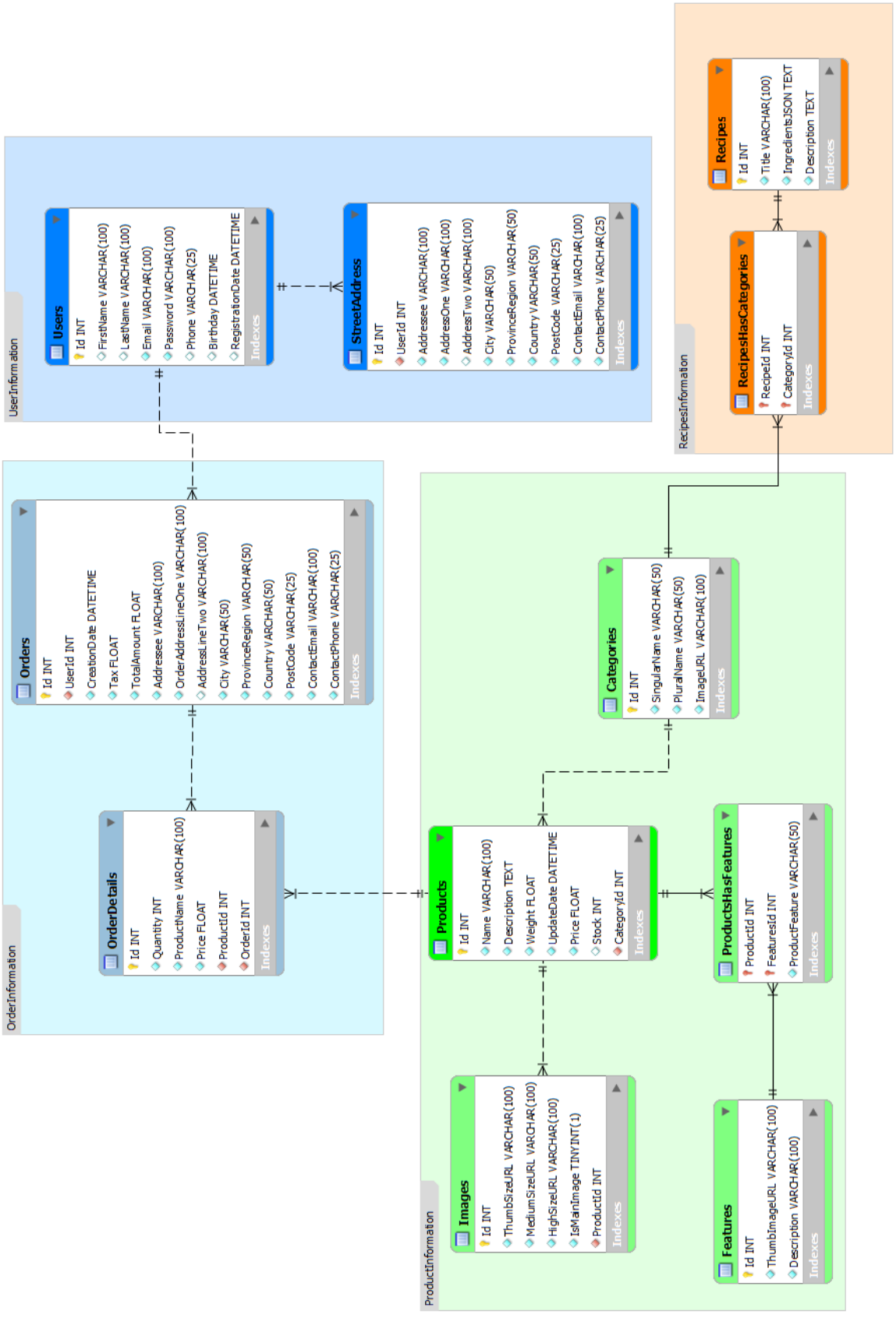
Script Base De Dades

```
-- -----------------------------------------------------
-- Schema productesdecasa_schema
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `productesdecasa_schema` DEFAULT CHARACTER
SET utf8 COLLATE utf8_general_ci ;
USE `productesdecasa_schema` ;
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`Categories`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `productesdecasa_schema`.`Categories` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `SingularName` VARCHAR(50) NOT NULL,
  `PluralName` VARCHAR(50) NOT NULL,
  `ImageURL` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`Products`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `productesdecasa_schema`.`Products` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(100) NOT NULL,
  `Description` TEXT NOT NULL,
  `Weight` FLOAT NOT NULL,
  `UpdateDate` DATETIME NOT NULL,
  `Price` FLOAT NOT NULL,
  `Stock` INT NULL,
  `CategoryId` INT NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`Images`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `productesdecasa_schema`.`Images` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `ThumbSizeURL` VARCHAR(100) NOT NULL,
  `MediumSizeURL` VARCHAR(100) NOT NULL,
  `HighSizeURL` VARCHAR(100) NOT NULL,
  `IsMainImage` TINYINT(1) NOT NULL,
  `ProductId` INT NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`Users`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `productesdecasa_schema`.`Users` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `FirstName` VARCHAR(100) NULL,
  `LastName` VARCHAR(100) NULL,
  `Email` VARCHAR(100) NOT NULL,
  `Password` VARCHAR(100) NOT NULL,
  `Phone` VARCHAR(25) NULL,
  `Birthday` DATETIME NULL,
  `RegistrationDate` DATETIME NULL,
```

```
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`Orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `productesdecasa_schema`.`Orders` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `UserId` INT NOT NULL,
  `CreationDate` DATETIME NOT NULL,
  `Tax` FLOAT NOT NULL,
  `TotalAmount` FLOAT NOT NULL,
  `Addressee` VARCHAR(100) NOT NULL,
  `OrderAddressLineOne` VARCHAR(100) NOT NULL,
  `AddressLineTwo` VARCHAR(100) NULL,
  `City` VARCHAR(50) NOT NULL,
  `ProvinceRegion` VARCHAR(50) NOT NULL,
  `Country` VARCHAR(50) NOT NULL,
  `PostCode` VARCHAR(25) NOT NULL,
  `ContactEmail` VARCHAR(100) NOT NULL,
  `ContactPhone` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`OrderDetails`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `productesdecasa_schema`.`OrderDetails` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `Quantity` INT NOT NULL,
  `ProductName` VARCHAR(100) NOT NULL,
  `Price` FLOAT NOT NULL,
  `ProductId` INT NOT NULL,
  `OrderId` INT NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`StreetAddress`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `productesdecasa_schema`.`StreetAddress` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `UserId` INT NOT NULL,
  `Addressee` VARCHAR(100) NOT NULL,
  `AddressOne` VARCHAR(100) NOT NULL,
  `AddressTwo` VARCHAR(100) NULL,
  `City` VARCHAR(50) NOT NULL,
  `ProvinceRegion` VARCHAR(50) NOT NULL,
  `Country` VARCHAR(50) NOT NULL,
  `PostCode` VARCHAR(25) NOT NULL,
  `ContactEmail` VARCHAR(100) NOT NULL,
  `ContactPhone` VARCHAR(25) NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`Features`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `productesdecasa_schema`.`Features` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `ThumbImageURL` VARCHAR(100) NOT NULL,
  `Description` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`Recipes`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `productesdecasa_schema`.`Recipes` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `Title` VARCHAR(100) NOT NULL,
  `IngredientsJSON` TEXT NOT NULL,
  `Description` TEXT NOT NULL,
  PRIMARY KEY (`Id`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`ProductsHasFeatures`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS
`productesdecasa_schema`.`ProductsHasFeatures` (
  `ProductId` INT NOT NULL,
  `FeaturesId` INT NOT NULL,
  `ProductFeature` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`FeaturesId`, `ProductId`))
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `productesdecasa_schema`.`RecipesHasCategories`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS
`productesdecasa_schema`.`RecipesHasCategories` (
  `RecipeId` INT NOT NULL,
  `CategoryId` INT NOT NULL,
  PRIMARY KEY (`CategoryId`, `RecipeId`))
ENGINE = InnoDB;
```

Esquema Relacional



5.3. DISSENY DE SOFTWARE

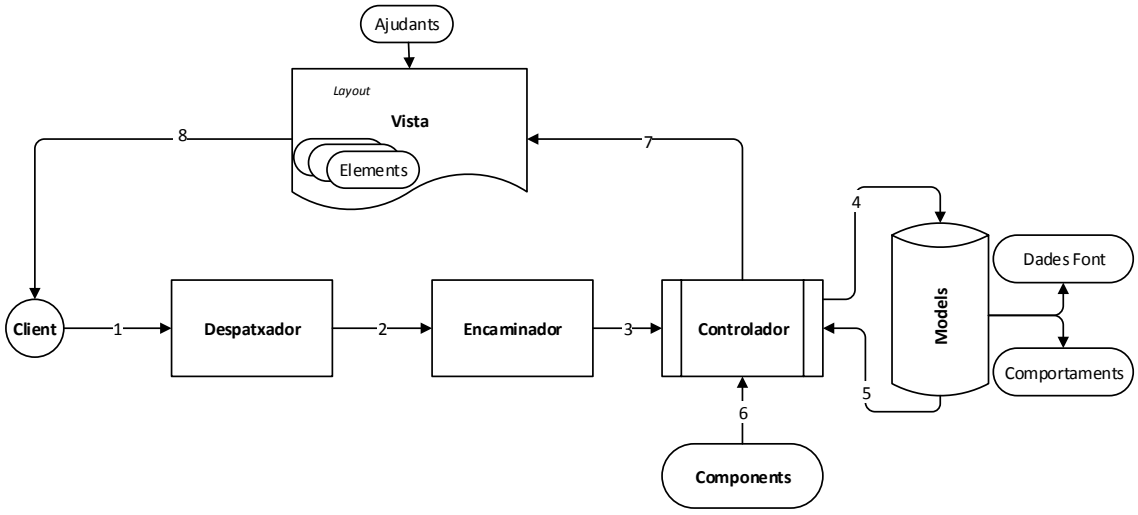
Com hem vist a l'apartat 4.3.1 Anàlisi Model Vista Controlador, necessitem una plataforma que segueixi les característiques del patró d'arquitectura MVC. Per això tenim dues possibilitats, per una part Implementació del patró des de zero, on s'utilitzen les llibreries nucli de PHP i és defineixen les restriccions necessàries per a que una plataforma d'aquestes característiques funcioni correctament. Per altre banda tenim la possibilitat d'utilitzar un *Framework*[13] que implementi el patró, i que la única preocupació sigui la de implementar la lògica que es vol i la configuració que això conforma.

La resposta al dilema plantejat anterior no és gens trivial, ja que, és molt difícil prendre la decisió degut a que no existeix cap mètrica que ens pugui assegurar que el nostre projecte serà implementat de millor o pitjor manera utilitzant un *Framework* o no. No obstant aquest projecte considera que la decisió va molt més enllà de la productivitat i el temps que es pugui trigar en fer les tasques proposades. Normalment els *frameworks* són treball revisat moltes vegades, solucionant problemes típics que apareixen quan és desenvolupen problemes d'una tipus en concret, revisat per una gran comunitat de desenvolupament, per tant això els fa menys propensos a sofrir errors. És per això que aquest projecte, aposta per la utilització d'una plataforma com els *frameworks*.

5.3.1. Marc de Treball: Cake PHP

La elecció pel desenvolupament serà la d'un framework php anomenat CakePHP[14]. És tracta d'una plataforma que ens defineix un marc de treball que ens facilitat el desenvolupament de la nostra aplicació web. Implementa el patró MVC per tant, és idoni, per la feina que planteja el projecte.

Tractament de les Peticions



Il·lustració 5.9 Petició Cake

[13] Per a més informació relacionada amb lo que és un framework
http://en.wikipedia.org/wiki/Software_framework
[14] Per més informació del framework CakePHP
http://en.wikipedia.org/wiki/CakePHP

1. La petició és llançada des d'un client web mitjançant una URL. Aquest client web, fa una petició al nestre servidor web. Per exemple URL = http://myapp.com/users/login.
2. El component Encaminador, s'encarrega de tractar la URL, amb l'objectiu de treure els paràmetres que afecten la lògica de negoci durant la petició.
3. Donats els paràmetres que ens dona el nostre encaminador, la petició que es fa amb la URL, és mapejada a una acció del controlador (Mètode especifico de la classe del controlador). En el nostre exemple, seria el mètode *Login()* del controlador de Usuaris. Remarcar, que hi ha una funció especial anomenada *beforeFilter(),* que és executada abans que qualsevol lògica de controlador sigui executada.
4. Una vegada seleccionat el controlador, el controlador s'encarregarà de utilitzar els models amb l'objectiu de donar accés a les dades de la aplicació. El model usat, pot fer us dels "Comportaments" i "Dades Font" durant les operacions que estigui fent. Si bé no es requereix l'ús del model, tots els controladors inicialment requereixen almenys d'un model per funcionar.
5. Una vegada el model ha recuperat les dades, el model és retornat al controlador.
6. El controlador una vegada te les dades que el model li proporciona, ha de utilitzar el components amb l'objectiu de refinar les dades o realitzar les operacions. Per exemple, manipulació, autentificació, enviar correus electrònics,... etc.
7. Una vegada els controladors han tractat les dades, aquestes són enviades a la visat usant el mètode *set()* del controlador. Això farà que la lògica de la vista sigui executada. La qual pot incloure l'ús dels elements i ajudants. Per defecte la vista és presentada dintre d'una "Presentació" o lo que és lo mateix un Layout.
8. Finalment, el controlador necessita fer algunes crides per a que les vistes puguin ser presentades i tot el codi enviat al client web que ha fet la petició.

A continuació veurem que l'ús del framework implica unes certes convencions, característica la qual, és una de les raons principals per les que aquest projecte aposta per aquesta tecnologia, ja que ens ajuda a estandarditzar el nostre projecte, traduint-se en millor qualitat de codi i fàcil manteniment a llarg termini. Recordem que és un dels objectius per aquest projecte. *(veure Il·lustració 1.1 Pilars del Software de Qualitat)*

Convencions Controladors

Han de tindre el nombre en plural, con el format *"CamelCase"*[15] i que finalitza amb la paraula en anglès *"Controller".* El primer mètode a sobreescriure és l'anomenat *index()*. Per exemple si cridem a la URL http://myapp.com/controller/ per defecte cridarà a la vista index, en canvi se el que es vol, es cridar a una en concret del controlador, s'haura d'implementar el mètode *view()* i podrem cridar a la següent URL http://myapp.com/controller/view/. Un detall important es que els mètodes vista només és podran cridar si són públics, en altre cas no seria possible.

[15] Per veure més informació sobre com funciona el format Camel Case http://en.wikipedia.org/wiki/CamelCase

Capítol 6. PROVES

Una vegada hem vist com hem preparat la nostra aplicació per respondre davant de les peticions dels usuaris, ara és necessari aturar-nos i definir un entorn de validació de les diferent funcionalitats que el nostre software pot tenir. Es podran trobar diferents cassos de prova en funció de les especificacions aparegudes en el l'apartat 2.2 Especificacions del segon capítol d'aquest projecte.

6.1. SINTAXI DONAT-QUAN-LLAVORS

Quan entrem en el món de la qualitat, en el que respecte Enginyeria del Software és molt importat parar i preocupar-nos a definir processos útils i àgils que ens ajudin a assegurar i validar la qualitat del nostre codi. Ha de ser també un procés fàcil de mantenir, ja que d'aquesta manera assegurem que el nostre pla de proves està actualitzat i ens esta donat una traçabilitat correcta de l'estat del nostre codi en funció de les especificacions que s'han definit. D'aquesta manera aconseguim un mètode que ens aporta valor al nostre producte ja que ens ajuda a reaccionar ràpidament davant dels possibles defectes que puguin aparèixer a la nostra aplicació.

És per això que aquest projecte aposta per un procés d'aquest tipus, que ens pugui ser útil per la definició d'escenaris de prova fiables davant de la nostra aplicació. Aquest apartat estarà basat en una metodologia coneguda com *"Desenvolupament orientat a Comportaments"*[16]. Si consultem sobre aquesta metodologia veurem que és molt àmplia, i no serà usada en la seva totalitat en aquest projecte. Només ens quedarem amb el concepte de **Especificacions de Comportament**, la qual consisteix en la utilització d'una sintaxi anomenada *llenguatge de Gherkin*, per tal de definir uns escenaris de prova o també coneguts com **Criteris d'Acceptació**[17] d'una funcionalitat a implementar. Per entendre millor el concepte anem a veure un exemple.

Tenim una especificació que ens diu que com a usuari d'una aplicació d'un banc, vull ser capaç de transferir diners entre comptes corrents actives, d'aquesta manera no sobregiro el meu compte i no haig d'anar al banc per ferla transferència. El criteri d'acceptació quedaria de la següent manera.

Escenari: Transferència quantitat positiva

Donat un compte corrent "A" que té 100$

I un compte de xecs "B" que consta de 50$

Quan Faig una transferència de 10$ des del banc "A" al compte "B"

Llavors la transferència acabarà correctament

Llavors el compte "A" ha de tenir un total de 90$

I el compte "B" ha de tenir $60

Il·lustració 6.1 Exemple Gherkin Sintaxi

Com hem vist un dels punts forts són que són cassos de prova definits pel client que vol el software, i donat que són escenaris propensos a automatitzar aporten molt valor al nostre producte.

[16] de l'anglès *behavior-driven development* (abreviat BDD) consultar per més informació. http://en.wikipedia.org/wiki/Behavior-driven_development

[17] Per més informació sobre les proves d'acceptació. http://en.wikipedia.org/wiki/Acceptance_testing

6.2. PLA DE PROVES

6.2.1. Criteri D'acceptació #1: Registre d'usuaris

Escenari: Registre dades d'un usuari

Donat que un usuari de productes de casa Obre i omple el formulari de registre

Quan l'usuari confirma el formulari amb les dades entrades correctament

Llavors L'usuari és adreçat a la pàgina home web amb les dades correctament.

Escenari: correu electrònic ben entrat

Donat que un usuari de productes de casa Obre i omple el formulari de registre amb correu electrònic i el repetit entrats amb valors diferents

Quan l'usuari confirma el formulari

Llavors L'usuari no podrà continuar amb la creació del compte fins que les dades siguin entrades correctament.

Escenari: Contrasenya ben entrada

Donat que un usuari de productes de casa Obre i omple el formulari de registre amb contrasenya i la repetida entrats amb valors diferents

Quan l'usuari confirma el formulari

Llavors L'usuari no podrà continuar amb la creació del compte fins que les dades siguin entrades correctament.

6.2.2. Criteri D'acceptació #2: Edició dades usuari

Escenari: Botó d'edició de compte

Donat Que soc un usuari de productes de casa

Quan Desplego la informació relacionada amb el meu compte

Llavors L'aplicació m'ha de presentar un botó "Editar el Meu Compte"

Escenari: Formulari d'edició

Donat Que soc un usuari de productes de casa

Quan faig clic al botó "Editar el Meu Compte"

Llavors L'aplicació m'ha de presentar un formulari amb les dades relacionades amb el meu compte

6.2.3. Criteri D'acceptació #3: Inici Sessió

Escenari: Iniciant Sessió

Donat que soc un usuari no registrat de productes de casa

Quan faig clic en el botó d'inici sessió

Llavors l'aplicació ha de presentar un formulari amb els camps, "Correu Electronic" i "Contrasenya".

6.2.4. Criteri D'acceptació #4: Tancament Sessió

Escenari: Tancant Sessió

Donat que soc un usuari registrat de productes de casa

Quan faig clic en el botó de tancar sessió

Llavors l'aplicació adreça a la pagina home, desvinculant qualsevol sessió oberta amb el meu compte.

6.2.5. Criteri D'acceptació #5: Vistes

Escenari: Home

Donat Que un usuari ha entrat productes de casa

Quan la pàgina és carregada correctament

Llavors la primera vista que veurà serà la home.

Escenari: Categories

Donat Que un usuari ha entrat productes de casa

Quan fa clic en el menú de categories en una categoria en concret

Llavors l'aplicació presentarà la pagina d'aquesta categoria amb un llistat de productes associats a aquesta categoria.

Escenari: Tancant Sessió

Donat Que un usuari ha entrat productes de casa

Quan fa clic en algun producte concret

Llavors l'aplicació presentarà la pàgina corresponent a la fitxa del producte on podrà trobar la informació relacionada amb aquest producte

6.2.6. Criteri D'acceptació #7: Compra de productes

Donat Que un usuari ha entrat productes de casa

Quan la fitxa d'un producte és oberta

Llavors l'aplicació ha de presentar un botó que permeti comprar el producte.

Capítol 7. CONCLUSIONS

Arribem al final d'aquest projecte, i que millor que reflexionar sobre el treball fet, per tal de millorar en futurs projectes i reptes d'aquesta tipus. En aquest capítol es podrà trobar una reflexió dels diferents obstacles i coneixements adquirits durant el desenvolupament d'aquest projecte. Si vostè, el lector, està consultat la memòria per obtindré una guia per la realització d'un altre projecte similar, aquest capítol li donarà uns quants consells que segons el meu punt de vista personal, són almenys, meritoris d'estudi.

Finalment es comentaran les previsions de futur que aquest projecte té, passant per les millores previstes, acabant amb una valoració personal.

7.1. CREACIÓ, DESENVOLUPAMENT I FINALITZACIÓ DELS OBJECTIUS

Hi ha tres conceptes els quals, gràcies a aquest projecte he après a controlar i considero important descartar-los ja que poden servir d'ajuda per a futurs projectes:

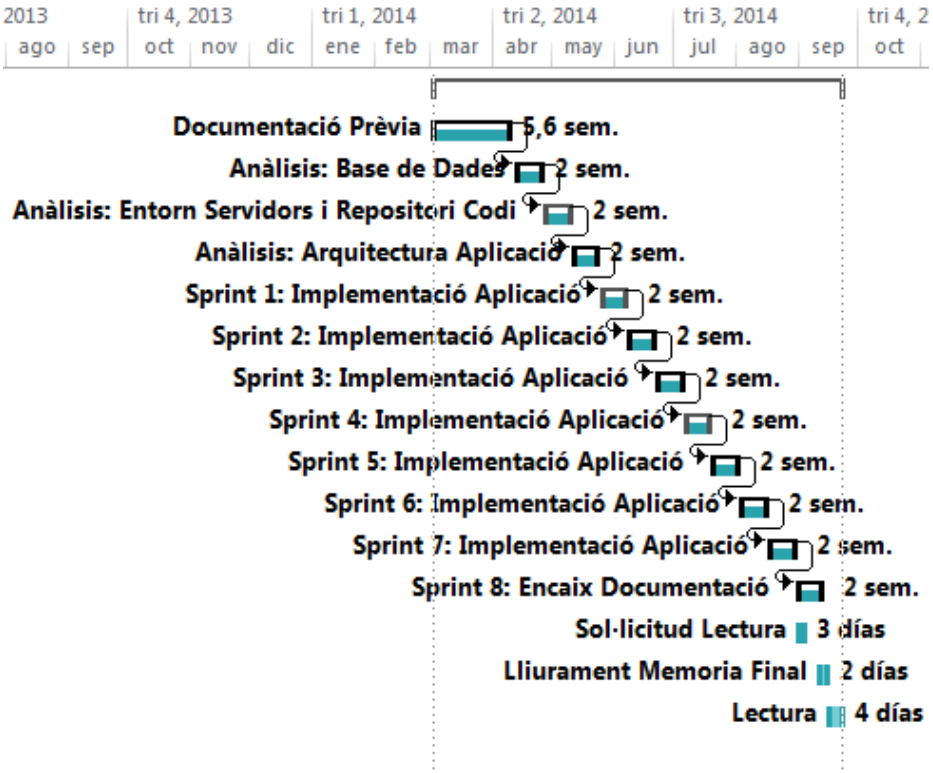
- Control de l'optimisme: a vegades quan penses que tens les idees molt clares, s'abusa de l'optimisme, la qual cosa, provoca estimacions poc encertades.
- Definició Objectius Concrets i no genèrics: Un concepte que és una pena haver-me trobat gairebé al final del projecte, i és el anomenat en anglès SMART Criteria[18]. Ve de l'acrònim Específic ,Mesurable, Assignable, Realista, Temps de durada.

7.2. PREVISIONS FUTUR

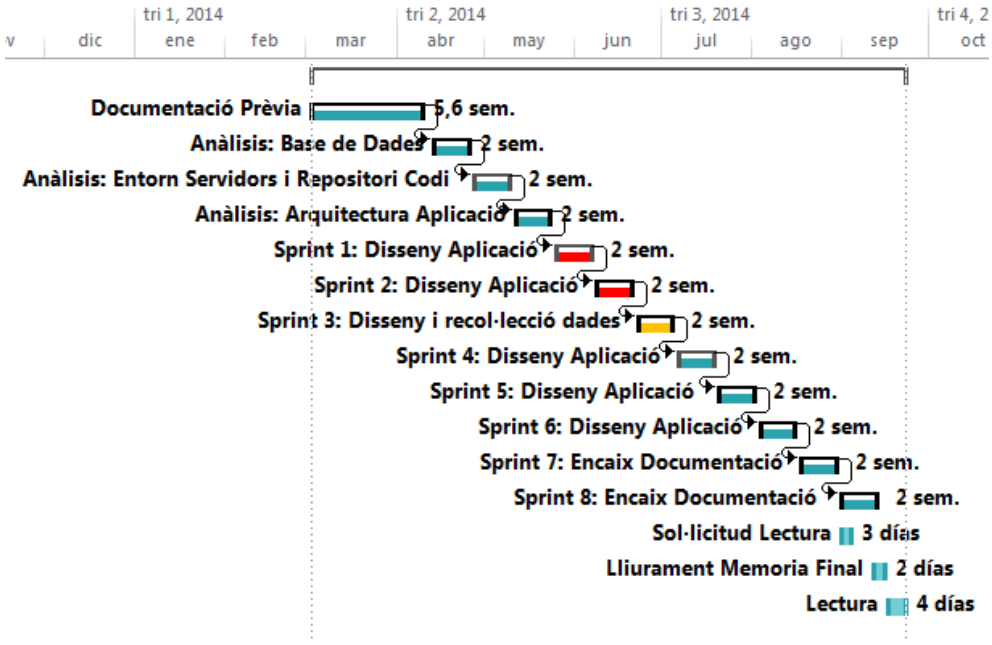
Actualment la idea ha estat proposta a un Majorista de vins reconegut, que ens a donat a veure el seu interès en l'aplicació. Una vegada entregat aquest projecte la idea que es té, és la de preparar una sessió de "Demostració" amb diferents funcionalitats que els hi agradaria veure que no han estat possibles d'assolir en la llargada d'aquest projecte.

[18] Per més informació sobre SMART criteria
http://en.wikipedia.org/wiki/SMART_criteria

7.3. DESVIACIÓ PLANIFICACIÓ



Il·lustració 7.1 Planificació Inicial



Il·lustració 7.2 Planificació Final

Com hem dit abans, la planificació d'aquest projecte a estat molt optimista i les fases marcades en vermell i groc, van ser les parts afectades per la setmana d'exàmens del curs 2014/15. Això v provocar una desviació bastant critica en el desenvolupament del projecte, i que va ser el responsable de no poder realitzar 100% de les funcionalitats posades a les especificacions d'aquest projecte.

7.4. OBJECTIUS ACONSEGUITS

- ✓ Recol·lecció Informació de les necessitats del client.
- ✓ Definició i Anàlisi Requeriments (RQ).
- ✓ Fer un estudi de viabilitat del projecte per tal seleccionar la manera de fer en funció de les possibilitats que tenim disponibles, analitzant per tant:
  - Riscos i fases del projecte.
  - Estudi de viabilitat econòmica.
- ✓ Definició Entorn desenvolupament de l'Aplicació en funció de les especificacions.
- ✓ Definició Tasques a realitzar, en funció de les especificacions, tenint en compte el temps de durada del projecte → Planificació del projecte.
- ✓ Estimació Tasques a realitzar durant el projecte en funció de tres Paràmetres:
  - Complexitat
  - Esforç
  - Incertesa

~ Disseny d'una prova de concepte de l'Aplicació amb el seguiment de bones practiques, en la mida del possible, i com és evident en la mida que ens aporti valor al producte final.

Aquest últim objectiu no ha estat assolit al 100%, només s'han arribat a les funcionalitats més critiques.

7.5. POSSIBLES MILLORES PREVISTES PEL FUTUR

7.5.1. Desplegament Software

Automatització Procés "Build"

Tot i que el programari relacionat amb aquest projecte és relativament senzill de tractar degut a que encara no és prou gran com per ser sensible a erros quan es fa una nova versió, durant la llargada del projecte si es van veure diferents processos sensibles d'automatització i que per futures versions del software seria oportú tindre-ho en compte.

7.5.2. Base de dades

Bases de Dades no relacional (NoSQL)

Les basses de dades relacionals no tenen res de dolent, precisament gracies al transcurs dels anys hem aconseguit aprendre tècniques bastant comuns per a normalitzar-les en la mida del possible, escalar-les segons creix la demanda i utilitzar-les com un sistema de persistència per a emmagatzemar la informació des de el nostre llenguatge processal o orientat a objectes favorit. La quota d'us de software tals com *SQLite, MySQL, PostgreSQL ...* etc, és molt alta, i es troben entre la major part dels desenvolupaments moderns.

Però actualment han arribat a la web, el software orientat a serveis, tals com la nube o diferents *startups* amb milions d'usuaris. Tot això porta problemes anomenats d'alta escalabilitat. Si bé els models relacionals poden ser adaptats per fer-los escalar inclús en els entorns més difícils, si que és cert, que sovint, cada vegada és fan menys intuïtius a medida que augmenta la complexitat. Tripletes y quadruplets JOIN en consultes SQL que no fan bona pinta, ja que donen lloc, interpretacions equivoques que portin poc eficiència.

Per tant els sistemes No SQL intentes solucionar aquest problema. Tot i que en la seva primera versió (i pot ser en la segona o fins i tot la tercera) no serà pas necessari un sistema d'aquest tipus. Si que és cert que aquest tipus d'aplicacions son propenses a problemes d'aquest tipus i és per això que aquest projecte vol tindre-ho com una nota mental per futures versions del software.

7.5.3. Funcionalitats Software

Protocols SSL

Com ben sabem i hem vist al llarg d'aquest projecte, Internet és el supermercat del futur. Gràcies a la seva comoditat, un numero de grandària amplia ja està fent les seves compres pel seu comerç electrònic preferit, ja sigui, llibres, roba, viatges... etc.

Però això implica un riscos de seguretat grans ja que al final estem tractant amb dades bancaries sensibles de ser robades. I es per això que per futures i pròximes versions és necessari la implementació de certificats SSL per tota la pàgina, sobre tot per les passarel·les de pagament.

7.5.4. Termes legals

Llei de protecció de dades[19]

Actualment, com a propietari d'un comerç electrònic ens hem de ajustar al que estipula la llei 34/2002, de serveis de la societat de la infirmació i Comerç electrònic *(LSSICE)* i també la llei orgànica de protecció de dades *LO15/1999* i l'últim decret que la desenvolupa *(RD 1720/2007),* el qual estableix les mides de seguretat per a fitxers automatitzats o no, que continguin dades de caràcter personal. És per això que per futures versions no molt llunyanes, és de vital importància que aquest temes siguin tractats, amb les mides que siguin necessàries a tots els nivells de la nostra aplicació web.

[19] Per mer informació sobre la llei de protecció de dades consultar la web de l'agència espanyola de protecció de dades http://www.agpd.es/portalwebAGPD/index-ides-idphp.php

Capítol 8. BIBLIOGRAFIA I WEBGRAFIA

8.1. LLIBRES DE CONSULTA

[Data darrera Consulta] *Informació llibre*

[14Juliol] Apunts Basses de dades I - Universitat Autònoma de Barcelona

Enric Martí – Catonte

http://caronte.uab.cat/course/category.php?id=6

[15Juliol] Jim Conallen: "Building Web Applications with UML".

Addison Wesley, 1999. ISBN: 0-201-61577-0

http://safari.awprofessional.com/0201615770

[16 Juliol] Object-Oriented Software Engineering: A Use Case Driven Approach

http://books.google.co.uk/books/about/Object_oriented_software_engineering.html?id=A6lQAAAAMAAJ

[2Agost] Clean Code: A Handbook of Agile Software Craftsmanship

http://www.tud.ttu.ee/im/Kaarel.Allik/JOOP/Clean_Code_-_A_Handbook_of_Agile_Software_Craftsmanship.pdf

8.2. WEBS DE CONSULTA

[Data darrera Consulta]	*Informació web*
[12Abril]	The ANSI-SPARC Architecture http://en.wikipedia.org/wiki/ANSI-SPARC_Architecture
[19Maig]	Distributed Application Architecture http://www.oracle.com/technetwork/java/index.html
[27Maig]	An introduction to Software Architecture http://dl.acm.org/citation.cfm?id=865128
[1Juny]	Applying the Model-View-Controller Paradigm to Adaptive Test http://www.utdallas.edu/~yiorgos.Makris/papers/dt12.pdf
[14Juliol]	Alistair Cockburn: "HexagonalArchitecture". http://c2.com/cgi/wiki?HexagonalArchitecture
[25Juliol]	The Elements of Typographic Style Applied to the Web http://webtypography.net/
[20Agost]	Publicacions GenbetaDev sobre basses de dades http://www.genbetadev.com/categoria/bases-de-datos
[1Setembre]	PHP: The Right Way http://www.phptherightway.com/

IL·LUSTRACIONS