Estudi de cas

Jesús Mejuto
Marc Verdera

PID_00198050

UOC
Universitat Oberta
de Catalunya

Els textos i imatges publicats en aquesta obra estan subjectes –llevat que s'indiqui el contrari– a una llicència de Reconeixement-NoComercial-SenseObraDerivada (BY-NC-ND) v.3.0 Espanya de Creative Commons. Podeu copiar-los, distribuir-los i transmetre'ls públicament sempre que en citeu l'autor i la font (FUOC. Fundació per a la Universitat Oberta de Catalunya), no en feu un ús comercial i no en feu obra derivada. La llicència completa es pot consultar a http://creativecommons.org/licenses/by-nc-nd/3.0/es/legalcode.ca

Índex

1. Disseny d'una intranet 7
1.1. Introducció 7
1.2. Objectius 7
1.3. Equip 7
1.4. Brífing 8
1.4.1. Resumint aquest brífing 9
1.5. Iteracions 9
1.5.1. Iteració 1: anàlisi de projecte 9
1.5.2. Iteració 2: arquitectura de la informació general 10
1.5.3. Iteració 3: arquitectura de la informació detall 11
1.5.4. Iteració 4: *wireframing home* 12
1.5.5. Iteració 5: *wireframing* distribuïdores i detall (primera fase) 14
1.5.6. Iteració 5: *wireframing* distribuïdores i detall (segona fase) 14
1.5.7. Iteració 6: *look and feel home page* 16
1.5.8. Iteració 7: *look and feel* distribuïdores i detall (primera fase) 18
1.5.9. Iteració 8: *look and feel* distribuïdores i detall (segona fase) 19
1.5.10. Iteració 9: maquetació HTML/CSS base 19
1.5.11. Iteració 9: maquetació HTML/CSS pantalles 21
1.5.12. Iteració 10: fi del projecte 22
1.6. Enllaços d'interès 22

2. Disseny d'una aplicació iPhone 24
2.1. Introducció 24
2.2. Objectius 24
2.3. Equip 24
2.4. Brífing 24
2.5. Iteracions 27
2.5.1. Iteració 1: anàlisi de projecte 27
2.5.2. Iteració 2: arquitectura de la informació 27
2.5.3. Iteració 3: *wireframing* general 28
2.5.4. Iteració 4: *wireframing app* 29
2.5.5. Iteració 5: *look and feel* 30
2.5.6. Iteració 6: disseny aplicació 32
2.5.7. Iteració 7: maquetació *mobile* 33
2.5.8. Iteració 8: lliurament final del projecte 35
2.6. Enllaços d'interès 35

3. Disseny d'una interfície per a *smart TV* 37

3.1. Introducció 37
3.2. Objectius 37
3.3. Equip 38
3.4. Brífing 38
3.5. Iteracions 39
3.5.1. Iteració 1: anàlisi del projecte 39
3.5.2. Iteració 2: arquitectura de la informació 39
3.5.3. Iteració 3: *wireframing*..... 40
3.5.4. Iteració 4: comandament a distància 43
3.5.5. Iteració 5: *look and feel*..... 44
3.5.6. Iteració 6: iconografia 45
3.5.7. Iteració 7: *mock-up*..... 47
3.5.8. Iteració 8: lliurament final del projecte 48
3.6. Enllaços d'interès 48

4. Desenvolupament d'una aplicació mòbil per a sistemes Android..... 49
4.1. Introducció 49
4.2. Què és Android? 49
4.3. Per què Android? 49
4.3.1. Avantatges 49
4.3.2. Inconvenients 51
4.4. Opcions per a desenvolupar en Android 52
4.5. Consideracions per al desenvolupament 53
4.5.1. Llenguatge 53
4.5.2. Instal·lació de l'entorn 53
4.5.3. Verificació 55
4.6. Visió general 56
4.7. Especificacions 56
4.8. Disseny 57
4.8.1. Esbós general 57
4.8.2. Disseny final 57
4.9. Iteracions 57
4.9.1. Iteració 1: definició del flux de l'aplicació 57
4.9.2. Iteració 2: disseny - esbós 59
4.9.3. Iteració 3: disseny - disseny natiu general 60
4.9.4. Iteració 4: disseny - disseny natiu de la pàgina distribuïdora 60
4.9.5. Iteració 5: disseny - disseny natiu contacte 61
4.9.6. Iteració 6: contingut - contingut estàtic 62
4.9.7. Iteració 7: crear la nostra aplicació en Android 62
4.9.8. Iteració 8: contingut - creació de *web services*..... 63
4.9.9. Iteració 9: contingut - connexió a *web services*..... 64
4.9.10. Iteració 10: Disseny - disseny final 64
4.9.11. Iteració 11: publicació en l'Android Market 65
4.9.12. Iteració 12: llançament i posada en societat 65
4.9.13. Iteració *n*: ...? 65

5. Desenvolupament d'una web amb continguts administrables en Drupal 67
5.1. Introducció 67
5.2. Per què Drupal? 67
5.2.1. Avantatges 68
5.2.2. Inconvenients 68
5.3. Consideracions generals de desenvolupament 68
5.3.1. Control de versions 68
5.3.2. *Deployment* 69
5.3.3. Seguretat 69
5.4. Estructura d'un projecte Drupal 71
5.5. Model vista controlador i Drupal 73
5.6. Visió general 74
5.6.1. Instal·lació de Drupal a *server* 74
5.6.2. El panell d'administració 74
5.6.3. Funcionalitats 74
5.7. Especificacions 75
5.8. Iteracions 75
5.8.1. Iteració 1: la primera pàgina 75
5.8.2. Iteració 2: tipus de contingut - notícies 76
5.8.3. Iteració 3: tipus de contingut - bàners 78
5.8.4. Iteració 4: tipus de contingut - formulari de contacte ... 79
5.8.5. Iteració 5: pàgines - notícies 79
5.8.6. Iteració 6: pàgines - bàners 80
5.8.7. Iteració 7: pàgines - formulari de contacte 80
5.8.8. Iteració 8: disseny 80
5.8.9. Iteració 9: maquetació - estils comuns 81
5.8.10. Iteració 10: maquetació - *templates* específics 81
5.8.11. Iteració 11: menú i navegació 82
5.8.12. Iteració 12: cercador 83
5.8.13. Iteració 13: internacionalització 84
5.8.14. Iteració *n*: ...? 85

6. Desenvolupament d'una aplicació per a Facebook 86
6.1. Introducció 86
6.2. Per què Facebook? 86
6.2.1. Avantatges 86
6.2.2. Inconvenients 86
6.3. Opcions per a desenvolupar en Facebook 87
6.4. Consideracions per a desenvolupar en Facebook 87
6.4.1. Certificats segurs 87
6.4.2. Seguretat 88
6.4.3. Limitacions del contingut 88
6.5. Visió general 88
6.6. Especificacions 89
6.7. Disseny 89

6.7.1. Esbós general 90
6.7.2. Disseny final 91
6.8. Iteracions 91
6.8.1. Iteració 1: definició del flux de l'aplicació i implementació mínima 91
6.8.2. Iteració 2: crear la nostra aplicació en Facebook 93
6.8.3. Iteració 3: *wireframes* - pantalla per a no socis 94
6.8.4. Iteració 4: *wireframes* - pantalla per a socis 96
6.8.5. Iteració 5: *wireframes* - formulari de recollida de dades 97
6.8.6. Iteració 6: còpia de seguretat en base de dades 97
6.8.7. Iteració 7: *wireframes* - pantalla final 99
6.8.8. Iteració 8: bases del concurs 100
6.8.9. Iteració 9: disseny final - pantalla per a no socis 100
6.8.10. Iteració 10: disseny final - pantalla per a socis 101
6.8.11. Iteració 11: disseny final - formulari de recollida de dades 101
6.8.12. Iteració 12: disseny final - pantalla final 102
6.8.13. Iteració 13: protecció de dades 102
6.8.14. Iteració 14: compartir en mur o amb amics 102
6.8.15. Iteració 15: llançament i posada en societat 103
6.8.16. Iteració *n*: ...? 103

1. Disseny d'una intranet

1.1. Introducció

Es tracta d'un projecte real que vam desenvolupar dins d'una agència per a un gran client. El projecte va durar uns tres mesos (treballant-hi un equip de cinc persones) i va ser un cas d'èxit en què el client va quedar satisfet. En gran part, aquest èxit es va aconseguir gràcies a les metodologies àgils, que ens van ajudar a entendre'ns amb el client i a aconseguir la seva confiança i implicació.

Era un projecte de gran envergadura, així que la col·laboració entre els membres de l'equip va ser vital per a poder-lo desenvolupar amb l'excel·lència que buscàvem.

1.2. Objectius

Tot seguit us mostrarem recursos, metodologies, petites problemàtiques i coneixements necessaris per a poder desenvolupar un projecte d'aquest tipus.

L'objectiu principal d'aquest document és que tingueu una visió global de com s'ha d'enfocar un projecte d'aquesta envergadura. Posarem èmfasi també en la metodologia de treball que hem tractat en aquesta assignatura i veurem com s'aplica a un cas real.

Tanmateix, entrarem en prou detall i us donarem recursos i eines perquè vosaltres mateixos ho pogueu dur a terme d'una manera professional.

1.3. Equip

A continuació definirem l'equip encarregat per al projecte i les característiques de cadascun dels membres:

- ***Product owner***. És el membre de l'equip que té relació amb el projecte i el client a la vegada. S'encarrega de gestionar i motivar l'equip i de captar les necessitats del client per a poder optimitzar els *sprints* i les prioritats en cada moment del projecte.

- **Scrum màster**. És el membre que coneix la metodologia àgil i ajuda a aplicar-la per a optimitzar els resultats del projecte. S'encarrega que l'equip assoleixi els objectius establerts pel *product owner*.

Web recomanat

Per a més informació, podeu consultar l'adreça d'Internet següent:

http://en.wikipedia.org/wiki/Product_owner - Core_Scrum_roles

Web recomanat

Per a més informació, podeu consultar l'adreça d'Internet següent:

http://en.wikipedia.org/wiki/ScrumMaster - Core_Scrum_roles

- **UX/UI *designer***. En realitat, diferenciem entre un dissenyador d'UX (*user experience*) i un d'UI (*user interface*), però actualment una figura polivalent ens aporta molt més valor per a aquest projecte. S'encarrega d'analitzar la intranet actual, fer una anàlisi de la competència i les metodologies de creació de continguts per a intranets i crear un arbre de continguts i *wireframing* per al projecte.

Webs recomanats

Per a més informació, podeu consultar les adreces d'Internet següents:

http://en.wikipedia.org/wiki/User_experience_design

http://en.wikipedia.org/wiki/UI_designer

- **IxD/ *visual designer***. En realitat, diferenciem entre un dissenyador d'IxD (*interaction designer*) i un de visual (*visual designer*), però actualment una figura polivalent ens aporta molt més valor per a aquest projecte. S'encarrega d'interpretar els *wireframes*, amb la col·laboració de l'UX/UI *designer* i del *web developer*, i de realitzar els dissenys.

Webs recomanats

Per a més informació, podeu consultar les adreces d'Internet següents:

http://en.wikipedia.org/wiki/Interaction_design

http://en.wikipedia.org/wiki/Visual_designer

- ***Web developer***. S'encarrega de programar tant el *front-end* com el *back-end* d'una pàgina web. Agafa els dissenys i els transforma en HTML/CSS seguint els estàndards web i les normatives d'accessibilitat W3C.

Web recomanat

Per a més informació, podeu consultar l'adreça d'Internet següent:

http://en.wikipedia.org/wiki/Frontend

En aquest cas, tenim un equip en què cadascun dels membres té una especialitat, però coneix el treball dels altres components de l'equip. D'aquesta manera aconseguim una molt bona comunicació entre ells i tindrem present l'estat del projecte global en les etapes de *planning*.

1.4. Brífing

En aquest projecte, vam rebre un brífing en què se'ns especificaven les necessitats del client. Aquestes necessitats implicaven una remodelació de la intranet de què disposaven en aquell moment, ja que tenia uns quants anys. Havien anat modificant funcionalitats i afegint-n'hi de noves i els funcionava molt lentament.

A partir d'aquí, nosaltres havíem d'analitzar les funcionalitats de la intranet actual, afegir-hi les noves funcionalitats que especificaven en el brífing i crear el primer esbós per a poder fer el nou arbre de continguts.

Web recomanat

Per a més informació, podeu consultar l'adreça d'Internet següent:

http://es.wikipedia.org/wiki/Briefing

Brífing

És la part de la preparació d'una acció publicitària en què es fa l'elecció ordenada, estratègica i creativa de les dades que ens permetran definir els objectius publicitaris d'una manera correcta i mesurable. Consisteix en un document escrit en què el departament de màrqueting ha de fer-hi constar tota la informació necessària per a deixar clares les dife-

rències comercials i definir el que es vol aconseguir amb la publicitat. El crea l'empresa client amb la informació del mercat de què disposa i amb les línies bàsiques del pla de màrqueting de la marca que vol publicitar. És un document resum, molt sintètic, que facilita la tasca de l'agència.

Extret de la Viquipèdia

1.4.1. Resumint aquest brífing

El nostre client és una gran marca que dóna llicències a productors de qualsevol tipus de producte (samarretes, cascos, sabates, tasses, etc.) perquè utilitzin la seva marca.

El llicenciatari (productor que vol crear el producte) s'ha de registrar a la intranet perquè el nostre client faci la validació del producte. El productor es registra i puja la proposta de disseny del producte a la intranet, on el client el validarà o hi farà certes observacions. Si no s'aprova el producte, aquest procediment s'anirà repetint de manera cíclica fins que el client l'aprovi.

A part d'això, la intranet és un espai on hi haurà tots els documents corporatius de la marca, com ara notes de premsa, notícies, etc.

També haurem de plantejar l'opció que hi hagi un usuari "visitant", que tindrà accés a uns documents determinats i a la informació on s'explica el procediment descrit anteriorment. Aquesta figura es preveu perquè és una manera de captar nous llicenciataris, una font d'ingressos de la marca.

Des del punt de vista formal, es vol respectar la línia gràfica i el to de comunicació de la marca, així que a l'hora de fer el disseny visual també haurem de revisar els documents corporatius.

1.5. Iteracions

1.5.1. Iteració 1: anàlisi de projecte

Abans de realitzar la primera reunió de ***kick-off***, ens repartim la feina entre els membres de l'equip per poder tenir una visió global de l'estat del projecte en cada una de les fases. Un cop ens reunim amb el client, presentem l'equip i exposem la metodologia que seguirem. D'aquesta manera, el client coneixerà l'equip i el plantejament del projecte.

Kick-off

És la primera reunió amb l'equip i el client del projecte. En aquesta reunió es defineixen els elements de base per al projecte i altres activitats de planificació, es presenten els membres de l'equip de projecte al client i s'ofereix l'oportunitat de parlar de la funció de cada membre de l'equip. També es poden comentar altres elements de base, com ara l'horari, l'estat que informa, etc.

En aquest punt és quan ens posem a treballar. Per començar, mirarem la intranet que tenen actualment, on veurem, juntament amb les noves necessitats que ens ha aportat el client en el brífing, les premisses en les quals ens hem de basar.

En primer lloc, el *product owner* estableix els objectius del primer *sprint*. En aquest cas, la presentació de la primera fase de l'arbre de continguts. En aquesta fase, l'UX/UI *designer* té la responsabilitat de liderar el projecte i de desenvolupar la primera proposta.

Per a fer l'anàlisi, l'equip ha de consultar recursos a Internet, com ara altres intranets –o, en aquest cas, aplicacions de gestió de projectes– ja que aquesta intranet al cap i a la fi tractarà d'això.

Aplicacions de gestió de projectes

Nosaltres ens hem fixat en aplicacions com les següents:

- BaseCamp
- HiTask
- Omniplan

Aquestes aplicacions són líders actualment en el mercat i moltes empreses les utilitzen per a la seva gestió.

Un cop feta aquesta anàlisi, hem de decidir quines pràctiques ens interessen o podem reaprofitar per a crear la nostra intranet i, a partir d'aquí, crearem l'arbre de continguts.

1.5.2. Iteració 2: arquitectura de la informació general

L'UX/UI *designer* farà l'arbre de continguts amb paper i llapis per plantejar una estructura jeràrquica de la informació, començant des de la *home page* i baixant categòricament en cada una de les diferents pantalles de la intranet.

En aquest cas tindrem quatre grans àrees de la intranet, que expliquem a continuació.

1) Usuaris. N'hi ha de tres tipus:

a) Administradors. Tenen tots els permisos per a manipular usuaris i avaluar els projectes que pujaran els llicenciataris. Poden aprovar o denegar projectes, donar d'alta o de baixa usuaris, pujar documents i notícies, entre altres funcions que definirem tot seguit.

b) Llicenciataris. Poden crear un projecte nou, pujar imatges i fer comentaris. A banda d'això, com els visitants, també poden consultar tota la documentació de la intranet, com ara vídeos, documents PDF o Excel.

c) Visitants. Poden consultar tota la documentació de la intranet, com ara vídeos, documents PDF o Excel amb un cert límit, delimitat per l'administrador quan dóna d'alta un visitant.

2) Documentació. En aquesta secció tenim documentació diferent, com ara:

- Guies d'estil.
- Vídeos corporatius.

- Documents corporatius de tipus divers (inspiració, publicitat, etc.).

Hem de ser capaços de crear un nou document com a administradors i mostrar el detall de cadascun.

3) Projectes. En aquesta secció hem de tenir en compte que hi ha una gran quantitat de projectes i que el client els vol classificar primer de tot per "llicenciataris" i després com a:

- Aprovats.
- En procés.
- Denegats.

De totes maneres, també voldrà veure una llista amb la fotografia de tots els productes i que es pugui reconèixer quins estan aprovats, en procés o denegats.

4) Premsa. Aquesta àrea s'ha de classificar per notes de premsa i notícies. Hem de ser capaços de crear un nou document com a administradors i mostrar el detall de cadascun.

Un cop definides les necessitats d'estructura de la intranet, l'exercici consisteix a fer un arbre de continguts que inclogui totes aquestes funcions. El primer lliurament serà aquest, i haurem de validar amb el client l'estructura base abans d'avançar amb l'arbre de continguts.

Webs recomanats

Per crear aquest arbre, us recomanem que consulteu una sèrie de recursos que podeu trobar a Internet. Per a aquest tipus de projectes hi ha programes especialitzats a fer arbres de continguts com els següents:

Omnigraffle

Axure

Irise

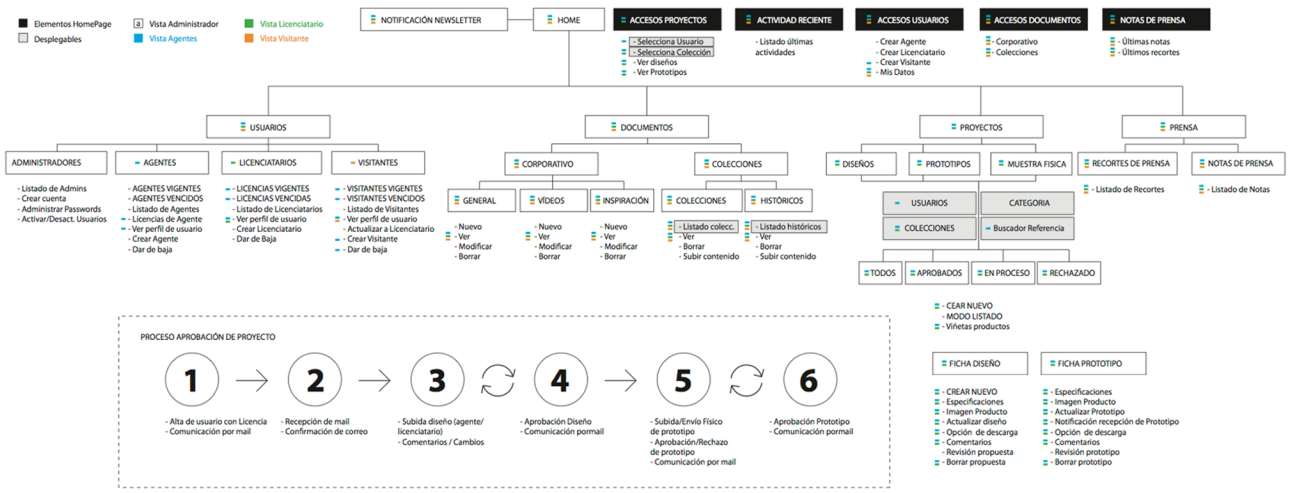
smartdraw

conceptdraw

Però també podem fer-ho amb programes vectorials com Flash, Illustrator o Fireworks.

1.5.3. Iteració 3: arquitectura de la informació detall

A partir de l'estructura base ja aprovada pel client, en aquesta iteració desenvoluparem el detall de les àrees i de continguts per a cada una de les seccions.



1.5.4. Iteració 4: *wireframing home*

Per a fer els *wireframes*, hem de tenir en compte una sèrie de normes d'usabilitat.

Usabilitat

De l'anglès *usability*, és la facilitat amb què la gent pot usar una eina o un giny (*widget*), per a aconseguir un objectiu concret. El terme també pot fer referència als mètodes per a mesurar la usabilitat i a l'estudi dels principis que doten un objecte d'usabilitat.

La usabilitat, tradicionalment relacionada amb el disseny de panells de control o interfícies per part dels enginyers, s'ha popularitzat amb l'arribada d'una gran quantitat de dispositius electrònics (mòbils, caixers automàtics, ordinadors, automòbils...) i l'ús de la informàtica (sistemes operatius, programari, pàgines web...). Per a fer-los intuïtius i fàcils d'utilitzar, poden requerir disciplines tan diverses com l'enginyeria, la psicologia, el disseny i la comunicació visual, la informàtica, l'etnografia, els estudis de mercat, etc.

La usabilitat permet:

- Més eficiència; amb menys temps es pot acabar una tasca concreta.
- Facilitat d'aprenentatge; el funcionament d'un objecte es pot deduir a partir de l'observació.
- Més satisfacció de l'usuari.

Jakob Nielsen i Steve Krug són dos dels experts destacats en usabilitat.

Extret de la Viquipèdia

Hem de tenir presents totes aquestes normes i recomanacions per a crear els *wireframes* de la intranet, però tot això portat a la pràctica es resumeix en una sola cosa: **sentit comú**.

Els *wireframes* consisteixen a dibuixar a manera d'esquema tots els elements que tindrà la intranet. Normalment, podem utilitzar diferents elements esquemàtics per a dibuixar cada una de les pantalles, com, per exemple:

- Textos descriptius.
- Quadrats.
- Triangles.
- Rodones.

- Fletxes.
- Text fals.
- Icones.

Però sempre acostumen a ser en blanc i negre i no tenen en compte proporcions, mides o alineacions exactes.

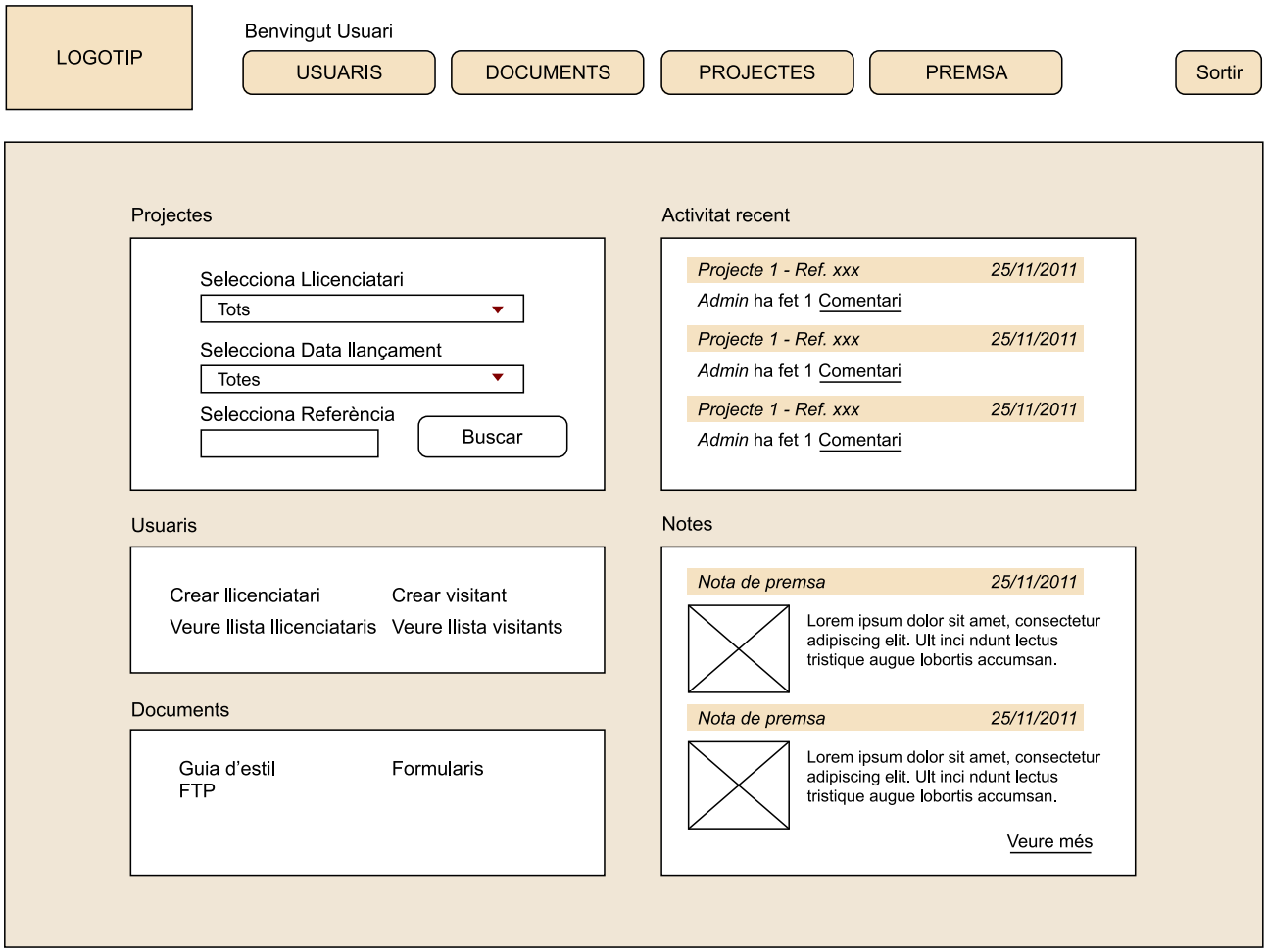
Crear *wireframes*

Per a crear aquests *wireframes* podeu fer servir qualsevol programa vectorial, però, igual que amb l'arquitectura d'informació, us poden servir altres eines:

- Omnigraffle
- Axure
- Irise
- smartdraw
- conceptdraw

En la xarxa podeu trobar recursos diversos per a Illustrator, Fireworks i Omnigraffle.

En aquesta fase, a partir de l'arbre de continguts ja aprovat pel client, plantejarem els *wireframes* de la *home*. El *wireframe* és important per si mateix perquè és on plantegem el menú de navegació i l'estructura formal del lloc (*site*).



1.5.5. Iteració 5: *wireframing* distribuïdores i detall (primera fase)

Un cop aprovats el *wireframe* de la *home page*, la navegació principal i l'estructura del *site*, treballarem les pantalles distribuïdores.

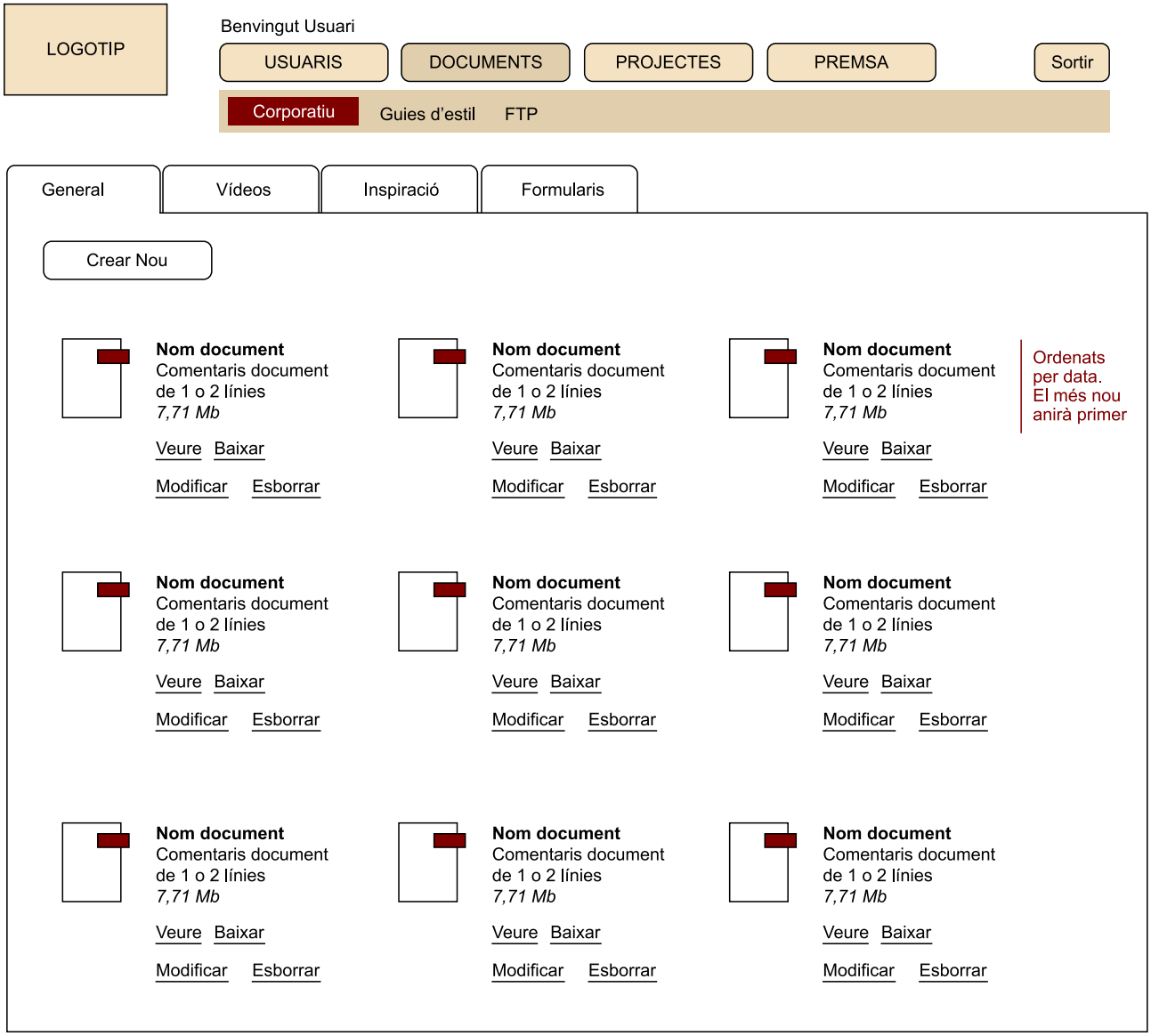
Aquests *wireframes* també són primordials en un projecte d'aquesta envergadura, perquè estem pensant la distribució en detall del contingut i els menús de segon nivell.

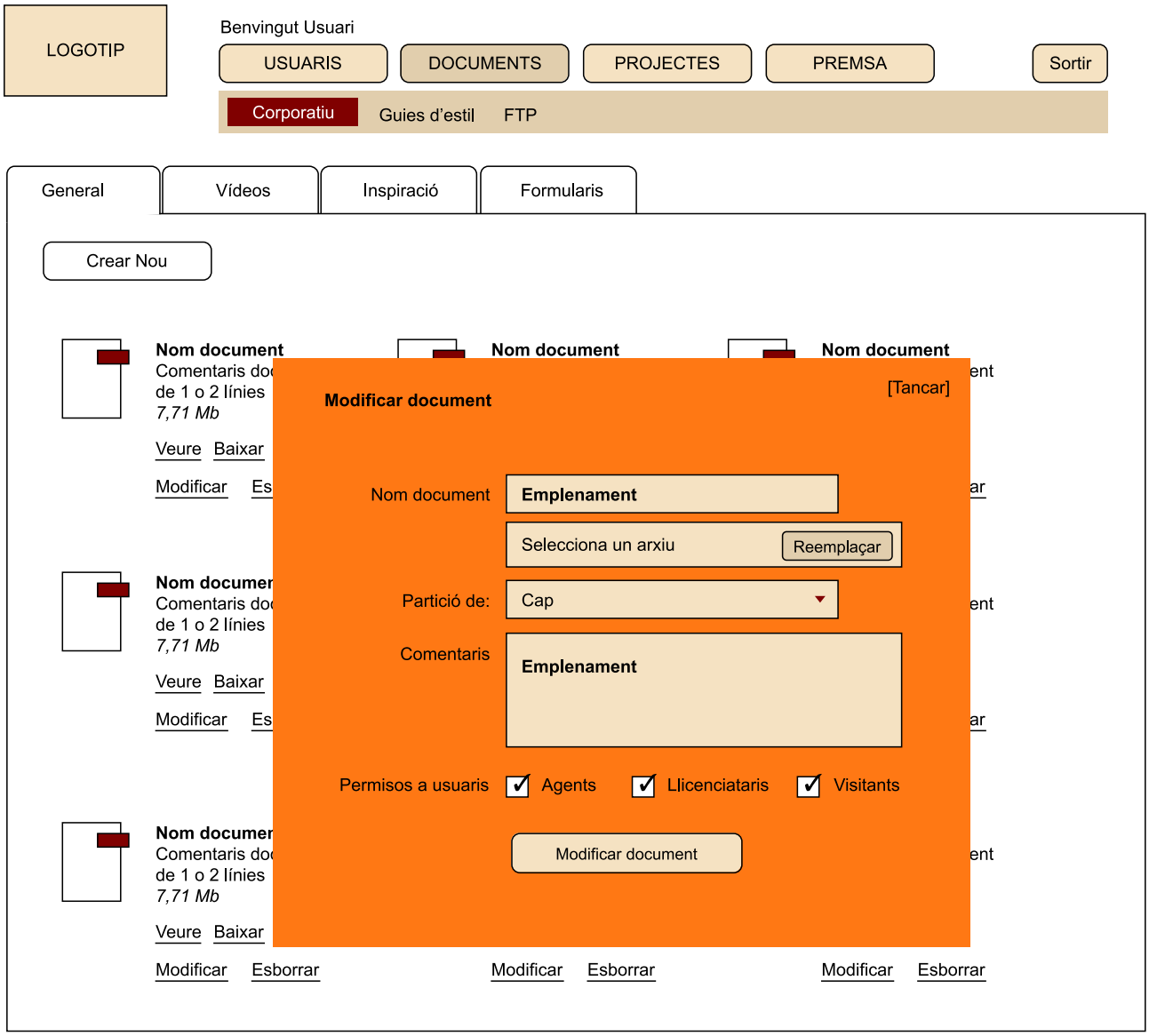
> L'exercici aquí serà presentar els *wireframes* de les pantalles d'"usuaris", "projectes", "documentació" i "premsa". Aquests *wireframes* no han d'estar polits al detall, però d'una manera general han de mostrar com estan distribuïdes la informació i les funcionalitats que aplicarem a la intranet.

1.5.6. Iteració 5: *wireframing* distribuïdores i detall (segona fase)

En aquesta fase, una vegada ens hagin aprovat la iteració anterior, desenvoluparem els *wireframes* ara ja sí amb tot detall de funcionalitats i continguts.

L'exercici aquí serà presentar els *wireframes* de les pantalles d'"usuaris", "projectes", "documentació" i "premsa" complets.





1.5.7. Iteració 6: *look and feel home page*

Paral·lelament a la creació de *wireframes*, podem anar pensant en el ***look and feel*** de la intranet. Quan tinguem la *home* plantejada a nivell de *wireframes* ja hi podrem començar a treballar.

> ***Look and feel***
>
> En el món interactiu, ***look and feel*** és un terme utilitzat respecte a una interfície gràfica i comprèn aspectes del seu disseny, incloent-hi elements com colors, formes, traçat i tipus de lletra, a més del comportament d'elements dinàmics com botons, caixes i cartes.
>
> Extret de la Viquipèdia

En el cas de crear el ***look and feel***, el *visual designer* ha de tenir en compte totes les coses vistes fins ara:

- **L'anàlisi de la competència** (també des del punt de vista gràfic), en què el dissenyador ha d'aportar idees, recursos i posar en pràctica les **millors pràctiques**. Tenir una idea sobre a qui s'adreça el projecte, quines neces-

sitats es plantegen, etc. és molt important a l'hora de plantejar un disseny visual.

- **L'arquitectura de la informació**, en què el dissenyador tindrà una idea global de l'abast del projecte. És important que el dissenyador tingui consciència d'aquest abast per a poder plantejar de manera exitosa un disseny.

- **Els *wireframes***, en què es veurà detallat pantalla per pantalla com estarà organitzada la informació i què hi haurà en cadascuna de les plantilles. A partir d'aquí, és feina del dissenyador definir la **jerarquia visual** i la importància que tindran cadascun dels elements en el disseny.

Una millor pràctica

És un mètode o tècnica que ha demostrat d'una manera consistent resultats millors que els aconseguits amb altres mitjans, i que s'utilitza com a punt de referència. A més, una "millor" pràctica pot evolucionar per a ser millors com les millores que es descobreixen. La millor pràctica és considerada per alguns un concepte de moda dels negocis, que s'utilitza per a descriure el procés de desenvolupament i seguiment d'una manera estàndard de fer les coses que diverses empreses poden emprar.

Les millors pràctiques s'utilitzen per a mantenir la qualitat com una alternativa a les normes obligatòries o legislades i poden estar basades en l'autoavaluació o l'avaluació comparativa.

Extret de la Viquipèdia

Jerarquia visual

En tota composició gràfica s'ha de crear una jerarquia visual adequada, per tal que els elements més importants es mostrin destacats com cal.

Mitjançant un disseny adequat es pot establir un camí visual que condueixi l'ull de l'espectador i li vagi mostrant la informació continguda en la composició d'una manera organitzada, lògica i fiable, que dirigeixi la seva percepció per la ruta més idònia.

Extret de DesarrolloWeb

Un cop fet aquest recordatori i abans de posar-nos a dissenyar, hem de fer un *benchmarking* de manera visual per a veure quins recursos, gràfics o estils seran els més idonis per a aplicar al disseny visual de la nostra intranet.

Webs recomanats

Hi ha molts recursos pel que fa a gràfics, icones, menús, textures, tipografies, etc. a Internet. A continuació mostrem alguns portals de referència que ens ajudaran a trobar els recursos que necessitem.

http://www.smashingmagazine.com/

http://www.alvit.de/handbook/

http://skout.co.za/

http://www.bluevertigo.com.ar/

Un cop ens haurem fet una idea de l'estil que volem donar al nostre disseny, passarem a l'acció. Per a dissenyar podem fer servir el Photoshop o el Fireworks, depenent del programa amb què ens sentim més a gust. Molts dissenyadors fan servir el Photoshop, però el fet és que Fireworks és el programa que Adobe prepara específicament per a disseny web.

Abans de posar-nos a treballar amb el programa, és molt important tenir en compte dues coses principals:

1) La resolució de pantalla: la resolució estàndard és 1.024 × 768, en què l'àrea visual real serà 960 × 600, aproximadament. La del nostre disseny, a diferència d'un disseny destinat a impremta, ha de tenir 72 pp (píxels per polzada).

2) La quadrícula: una eina que ens ajudarà molt a l'hora de dissenyar és un bon sistema de quadrícules, o *grid system*. Normalment, aquests sistemes tenen possibilitats i es mesuren en nombre de columnes.

Webs recomanats

Per a escollir la millor quadrícula que s'adeqüi al disseny, podeu buscar en alguns dels enllaços que mostrem a continuació:

http://www.thegridsystem.org/

http://960.gs/

http://modulargrid.org/#app

Resolució de pantalla

És el nombre de píxels que es pot mostrar en la pantalla. És determinada pel producte de l'amplària per l'alçària, mesurat en píxels, i s'obté una relació anomenada *relació d'aspecte*. En aquesta relació, es pot trobar una variació d'acord amb la forma del monitor i de la targeta gràfica. Es poden diferenciar dues mides de pantalla:

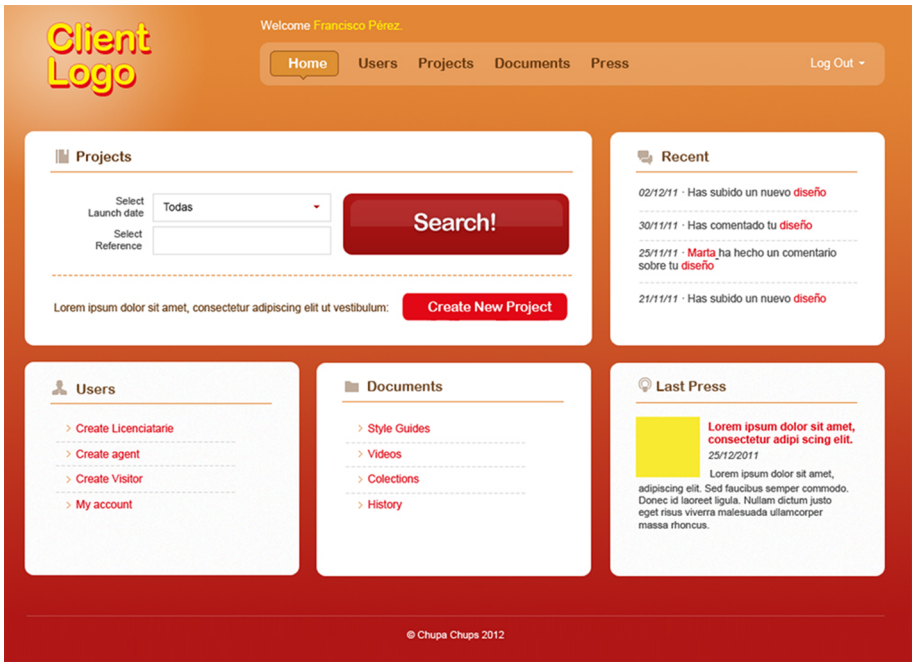
1) **Mida absoluta** : són les dimensions d'amplària i alçària de la finestra del monitor, mesurat generalment en polzades. Depèn del monitor.

2) **Resolució o mida relativa** : és determinada pel nombre de píxels que es mostren en la finestra del monitor, en què el píxel és la unitat mínima d'informació que es pot presentar en pantalla, de forma generalment rectangular. Depèn de la targeta gràfica.

Extret de la Viquipèdia

En aquesta iteració farem només el disseny de la *home page*. Generalment, aquest disseny és el més complicat perquè ha d'entrar pels ulls al client. Des dels punts de vista estructural i de continguts ja ho tenim tot definit, però ara hem de convèncer el client en el terreny gràfic.

L'exercici aquí serà fer un disseny de la *home page*.



1.5.8. Iteració 7: *look and feel* distribuïdores i detall (primera fase)

Un cop el client hagi donat la seva aprovació al disseny de la *home page*, farem el disseny de les pantalles interiors.

Aquest projecte té molta envergadura, així que haurem d'escollir estratègicament quines pantalles dissenyarem. Triarem les plantilles bàsiques per tal de dotar els programadors de tots els elements gràfics per a desenvolupar tota la intranet. El més normal és escollir la *home* i quinze pantalles que, en global, reuneixin tots els elements que surtin en tota la intranet.

En aquesta primera fase farem el disseny de la pantalla de projectes.

1.5.9. Iteració 8: *look and feel* distribuïdores i detall (segona fase)

En la iteració anterior hem dissenyat la pantalla de projectes, així que haurem definit el submenú i molts dels elements principals de la intranet.

En aquesta iteració ja farem el disseny de la resta de pantalles, en què utilitzarem molts dels elements emprats en la *home page* i la pantalla de projectes.

1.5.10. Iteració 9: maquetació HTML/CSS base

Com hem anat comentant al llarg del cas pràctic, aquest projecte té fites amb el client i es basa en les metodologies àgils, per la qual cosa hem tenir el client al corrent de com avança el projecte.

La part de la maquetació no n'és una excepció, i també n'hem de mostrar l'evolució en els diferents *sprints*.

La part de maquetació web es pot avançar un cop tinguem aprovat el disseny principal, ja que ens podrem basar en els estils generals i guiar-nos amb els *wireframes* desenvolupats prèviament.

Per a fer la maquetació, hem de tenir en compte unes quantes coses que són imprescindibles actualment. La maquetació s'ha de fer SEO-friendly i cal respectar els estàndards W3C. Ara ja parlarem d'HTML5 i CSS3, ja que comença a esdevenir l'estàndard.

W3C

El World Wide Web Consortium (W3C) és un consorci internacional que treballa per a desenvolupar i promocionar estàndards per al World Wide Web. El dirigeix Tim Berners-Lee, creador del WWW i autor de les especificacions de l'*uniformresource locator* , l'HTTP i l'HTML, que són les principals tecnologies d'aquesta xarxa.

Pastilla del W3C de recomanacions d'accessibilitat

El consorci s'administra de manera conjunta entre el MIT (als Estats Units), l'ERCIM (a Europa) i la Universitat de Keio (al Japó). Tanmateix, el W3C també té distribuïdes oficines arreu del món.

Aquest fou creat inicialment per a garantir una major compatibilitat i acord entre els diferents membres de la indústria a l'hora d'adoptar noves tecnologies web. Abans de la

seva creació, per exemple, ja hi havia problemes de compatibilitat a causa de versions diferents d'HTML específiques de cada venedor. Cal dir que aquests problemes encara perduren.

Un document del W3C segueix diferents **etapes de maduració** abans d'esdevenir una recomanació pròpiament dita:

- *Working draft*, WD, esborrany de treball.
- *Candidate recommendation*, CR, recomanació candidata.
- *Proposed recommendation*, PR, proposta de recomanació.
- *W3C recommendation*, REC, recomanació.

Les recomanacions es poden actualitzar de manera separada en documents d'errata, fins que hi hagi prou canvis significatius per a produir una nova edició de la recomanació. Aquest és el cas de l'XML, que es troba actualment en la tercera edició. Així mateix, el W3C també publica diversos tipus de notes informatives, que en principi no caldria considerar estàndards.

Extret de la Viquipèdia

HTML 5

HTML 5 estableix una sèrie de nous elements i atributs que reflecteixen l'ús típic dels llocs web moderns. Alguns són tècnicament similars a les etiquetes `<div>` i `<span>`, però tenen un significat semàntic, com ara `<nav>` (bloc de navegació del lloc web) i `<footer>`. Altres elements proporcionen noves funcionalitats per mitjà d'una interfície estandarditzada, com els elements `<audio>` i `<video>`. També presenta millores en l'element `<canvas>`.

Alguns elements d'HTML 4.01 han quedat obsolets, incloent-hi elements purament de presentació, com `<font>` i `<center>`, ja que els efectes són manejats pel CSS. També es posa èmfasi en la importància de l'*scripting* DOM per al comportament de la web.

Diferències amb HTML 4 i XHTML

Com a nous elements: *article* i *dialog*.

Com a nous atributs: *media*, *ping*, *autofocus*, *inputmode*, *min*, *max* i *pattern*.

Elements eliminats: *acronym*, *applet*, *basefont*, *big*, *center*, *dir*, *font*, *frame*, *frameset*, *isindex*, *noframes*, *s*, *strike*, *tt* i *u*.

Atributs eliminats:

- *rev* i *charset* a `<link>` i `<a>`
- *target* a `<link>`
- *nohref* a `<area>`
- *profile* a `<head>`
- *version* a `<html>`
- *name* a `<map>`
- *scheme* a `<meta>`
- *archive*, *classid*, *codetype*, *declare* i *standby* a `<object>`
- *valuetype* a `<param>`
- *charset* a `<script>`
- *summary* a `<table>`
- *header*, *axis* i *abbr* a `<td>` i `<th>`

Extret de la Viquipèdia

Ens ocuparem del *front-end* de la intranet i, per ajudar-nos en aquest desenvolupament, podem aconseguir diversos recursos. També cal estar informats de totes les novetats del sector, per la qual cosa és important estar ben connectats a la xarxa si no ens volem quedar enrere en un món tant canviant com aquest.

Web recomanat

Per a més informació sobre SEO-friendly, podeu consultar l'adreça d'Internet següent:

http://webdesign.about.com/od/seo/a/seo_content.htm

Webs recomanats

Pel que fa al *framework* HTML/CSS, podeu consultar:

http://www.blueprintcss.org/

Per a mantenir-vos informats, podeu consultar els portals de referència següents:

http://www.html5rocks.com/es/

http://www.css3.info/

http://www.css3.com/

També podeu buscar diversos recursos en portals com els que hem comentat abans:

http://www.smashingmagazine.com/

http://www.alvit.de/handbook/

http://skout.co.za/

http://www.bluevertigo.com.ar/

Els *frameworks* són força utilitzats per molts *front-ends developers*, però nosaltres som partidaris de començar una maquetació de zero o fer-nos la nostra pròpia estructura personalitzada. Després ens serà més fàcil adaptar el codi.

L'exercici aquí serà fer la maquetació de les pantalles bàsiques (*home page* i projectes) i crear els estils suficients perquè els desenvolupadors del *front-end* del projecte puguin començar a treballar amb la programació de la intranet.

1.5.11. Iteració 9: maquetació HTML/CSS pantalles

Una vegada els programadors ens hauran donat el vistiplau a l'estructura de la nostra maquetació, maquetarem la resta de pantalles per lliurar el paquet complet en HTML i CSS.

L'exercici aquí serà fer la maquetació de les pantalles restants ("documents", "usuaris" i "premsa").

1.5.12. Iteració 10: fi del projecte

El nostre projecte s'acabarà quan s'hagi lliurat al client l'arbre de continguts, els *wireframes*, el disseny i la maquetació *front-end* de la intranet. Un cop tinguem tot això, entregarem al client i als seus desenvolupadors de *back-end* tots els arxius.

Encara que sembli que aquí s'acaba el projecte, no és del tot cert, ja que en un projecte sempre hi sol haver petits ajustos en totes les fases que haurem realitzat.

Gràcies a les metodologies àgils, segurament no ens trobarem amb gaires d'aquests ajustos, i els que ens trobem seran fàcils de solucionar.

Això serà així per l'alta implicació del client en cada fase del projecte i perquè el client tindrà la sensació (i, de fet, serà veritat) que el projecte és una mica seu i estarà convençut del resultat.

1.6. Enllaços d'interès

http://es.wikipedia.org/wiki/Wikipedia:Portada

http://www.desarrolloweb.com

http://basecamp.com/

http://hitask.com/

http://www.omnigroup.com/

http://www.conceptdraw.com/

http://www.axure.com/

http://www.irise.com/

http://www.smartdraw.com/

http://www.smashingmagazine.com/

http://www.alvit.de/handbook/

http://skout.co.za/

http://www.bluevertigo.com.ar/

http://www.thegridsystem.org/

http://960.gs/

http://modulargrid.org

http://webdesign.about.com

http://www.blueprintcss.org/

http://www.html5rocks.com/es/

http://www.css3.info/

http://www.css3.com/

2. Disseny d'una aplicació iPhone

2.1. Introducció

És un clar exemple de projecte de disseny d'una aplicació mòbil a partir d'un encàrrec real d'una empresa de telefonia mòbil. Normalment, els encàrrecs per a fer aplicacions mòbils vénen de clients que no tenen cap idea sobre l'entorn.

En aquest cas, però, l'encàrrec prové d'un client que ja té coneixements de l'entorn pel que fa a programació, però necessita assessorament en la conceptualització de l'aplicació i en la realització dels *wireframes* i el disseny.

2.2. Objectius

Els objectius per a aquest projecte són dur a terme un treball ràpid i eficaç, utilitzant els recursos mínims per a obtenir un rendiment màxim. En un món tan canviant com el nostre, és important saber gestionar més d'un projecte a la vegada, i per a això cal saber distribuir molt bé els recursos de què es disposa.

2.3. Equip

A continuació definim l'equip encarregat per al projecte:

- *Product owner*.
- Scrum màster.
- UX/UI *designer*.
- IxD/*visual designer*.
- *Mobile developer*.

En aquest cas, cadascun dels membres de l'equip té una especialitat, però coneix la feina del altres membres de l'equip. D'aquesta manera aconseguim una molt bona comunicació entre ells i tindrem present l'estat del projecte global en les etapes de *planning*.

2.4. Brífing

En aquest projecte vam rebre un brífing en què se'ns especificaven, bàsicament, les necessitats tècniques a l'hora de desenvolupar el disseny del projecte.

El brífing obtingut

El sistema ha de proveir una interfície adaptada a les capacitats i limitacions del canal, i del terminal pel qual s'estigui accedint. Per al canal **Mòbil**, s'haurà de detectar el tipus de terminal que utilitza l'usuari, per a seleccionar entre tres tipus de plantilles:

1) ***Mobile Web Smartphone***: destinat a terminals que tinguin navegadors amb suport a HTML 5.0, CSS 3.0 i javascript 1.6 o superior, i que la majoria estan basats en la plataforma Webkit.

2) ***Mobile Web Featurephone***: destinat a terminals que tinguin suport a navegació 3G, però que no brinden un suport complet a HTML, ja que es limiten a HTML 4.0, CSS 2.0, i una baixa compatibilitat a Javascript 1.2.

3) ***Wap Simplephone***: destinat a terminals amb una capacitat de navegació limitada (2G) i amb pantalles reduïdes, cal evitar l'ús d'imatges en tant que sigui possible i l'ús de tecnologies de *client-side script* i té una compatibilitat limitada de CSS i HTML MP.

Per al canal **Web** s'utilitzarà una sola interfície, amb compatibilitat *cross-browsers* als navegadors més populars del mercat.

Qualsevol altra especificació o tecnologia no expressada en aquest document podrà ser assumida i considerada pel *factory*, sempre que no irrompi amb les exposades.

1) ***Mobile Web Smartphone***

Aquesta interfície haurà d'implementar-se mitjançant una plantilla Drupal que compleixi els requisits tècnics següents:

- Ha d'estar dissenyada i optimitzada per a pantalles de 320 px d'ample, però alhora haurà de tenir l'habilitat d'adaptar-se de manera automàtica a pantalles de 640 px d'ample. Mitjançant *CSS float width*:100%

- El pes de les imatges no ha de superar els 10 Kbytes.

- El pes d'una pàgina mitjana (sense el contingut pla) no ha de superar els 30 K (inclou js, css, html i imatges).

- No ha d'utilitzar imatges en format SVG.

- Es podrà basar en el Tema Drupal jQuery Mobile Theme.

- Haurà d'incloure la metaetiqueta (*metatag*) *viewport* amb *width=device-width* per garantir que el disseny i maquetació pugui adaptar-se a l'ample de la pantalla del dispositiu.

- Ha d'estar desenvolupada sota HTML 5.0.

- No ha d'utilitzar els elements HTML 5.0: *websocket*, *canvas*, *media*, *audio*, *video*, ni *object*.

- No haurà d'utilitzar *iframes*, ni *popup windows*.

- Podrà utilitzar CSS 3.0.

- Haurà d'utilitzar CSS Media Queries, *max/min-device-width*, i *orientation*, per a adaptar el disseny del portal quan es gira el mòbil de manera horitzontal o vertical (*Landscape*, *Portail*).

- Es permet l'ús de Javascript 1.6 o superior, biblioteques jquery, anomenades *ajax* o qualsevol altra tecnologia *client-side script* que en garanteixi el funcionament a través de les diferents plataformes i *browsers* especificats en aquest document.

- Podrà utilitzar els *frameworks* i les biblioteques jQuery Mobile, jQTouch, Sencha Touch per al desenvolupament d'interfícies mòbils.

- No ha d'utilitzar Adobe Flash, Flash Lite, Shockwave, Microsoft Silverlight, o qualsevol altra eina que necessiti la preinstal·lació d'un *browser plugin*.

- Ha de ser *cross-platform* compatible per als sistemes operatius i *browsers* mòbils següents:
  - Android 1.6 o superior
  - Apple Safary iOS 3.x o superior
  - Backberry Browser 5.x o superior
  - Nokia Symbian^3 i Serie 60 3th
  - Browsers 7.x o superior
  - Nokia MeeGo Browser
  - Opera Mobile (no Opera Mini)

  - Firefox Mobile 1.x o superior
  - Windows Phone 6 o superior
  - Altres navegadors mòbils basats en Webkit & Webkit2

2) ***Mobile Web FeaturePhone***

Aquesta interfície haurà d'implementar-se mitjançant una plantilla Drupal que compleixi els requisits tècnics següents:

- El disseny ha d'estar optimitzat per a pantalles de 220 px d'ample.
- El pes de les imatges no ha de superar els 8 Kbytes.
- El pes d'una pàgina mitjana (sense el contingut pla) no ha de superar els 15 K.
- No ha d'utilitzar imatges en format SVG.
- No ha d'utilitzar trucades *Ajax*.
- El desenvolupament haurà d'estar basat en HTML 4.0 i CSS 2.0.
- Ha de fer un ús limitat de Javascript 1.4.
- S'hauran de fer proves de compatibilitat i *cross-platform* als dispositius següents:
  - Blackberry OS 4.x i superior
  - Telèfons Symbian Serie 40, sèrie N o 60 en la primera, segona i tercera edició
  - Windows Mobile i/o Windows CE

3) ***Wap Simplephone***

Aquesta interfície haurà d'implementar-se mitjançant una plantilla Drupal que compleixi els requisits tècnics següents:

- El disseny ha d'estar optimitzat per a pantalles de 128 px d'ample.
- El pes de les imatges no haurà de superar els 5 Kbytes.
- El pes d'una pàgina mitjana (sense el seu contingut pla) no haurà de superar els 8 K.
- Solament ha d'estar basat en xHTML MP 1.0.
- Solament ha d'utilitzar Wireless CSS 1.1.
- No ha d'utilitzar cap codi javascript o codi *script client-side*.
- No podrà utilitzar imatges en format SVG o PNG, solament JPG i GIF.
- El disseny haurà de ser *crowss-browser* per a qualsevol terminal compatible amb WAP 2.0.

D'altra banda, vam tenir una reunió amb el client en què ens va explicar el tipus d'aplicació que volien desenvolupar i els objectius que calia assolir.

Es tractava d'una aplicació mòbil per a un portal de telefonia que oferia jocs, tons, salvapantalles, etc. i era un portal competència de Movistar Emoción.

2.5. Iteracions

2.5.1. Iteració 1: anàlisi de projecte

A partir del brífing obtingut, hem de portar a terme una anàlisi general de la competència i de les *apps* semblants que puguem trobar en el mercat.

Normalment, el client sempre ens dirà quina és la seva competència i quines preferències o idees té per a desenvolupar el projecte. La nostra feina, però, serà anar més enllà i buscar recursos, altres pàgines similars i mètodes interessants que puguem aplicar al projecte.

És bo buscar en mercats més avançats al nostre per a inspirar-nos. Normalment, en pàgines americanes, franceses, angleses o japoneses és on trobarem els mètodes més interessants.

Webs recomanats

Alguns portals per a veure tendències i recursos mòbils són els següents:

http://cssiphone.com/

http://www.mobileawesomeness.com/

http://www.refinedmobile.com/

http://tapfancy.com/

http://www.tappgala.com/

http://www.iosinspires.me/category/appinterfaces/

Un cop feta aquesta anàlisi, decidirem quines pràctiques ens interessen o podem reaprofitar per crear la nostra *app* i, a partir d'aquí, crearem l'arbre de continguts.

Web recomanat

Per a més informació, podeu consultar l'adreça d'Internet següent:

http://en.wikipedia.org/wiki/Mobile_apps

2.5.2. Iteració 2: arquitectura de la informació

L'UX/UI *designer* farà l'arbre de continguts amb paper i llapis per plantejar una estructura jeràrquica de la informació, començant des de la *home page* i baixant categòricament en cada una de les diferents pantalles de l'aplicació.

Com hem dit en l'apartat del brífing, es tracta de fer un portal per a una companyia de telefonia en què el sistema de negoci és molt clar: la **descàrrega de continguts** (tons, música, jocs, etc.).

Per a accedir a aquest portal caldrà un registre i, un cop descarregada l'aplicació, el portal mostrarà els continguts organitzats per:

- "Destacats": a la *home* es mostraran els continguts més descarregats o més nous.

- "Notícies": notícies relacionades amb els continguts o amb el portal mòbil.

- "Arxius per descarregar": on es podrà valorar, compartir amb les xarxes socials i descarregar.

- "Informació corporativa del client".

- "Secció de contactes": contactes que tinguin la mateixa aplicació i l'opció de convidar altres contactes de l'usuari a l'aplicació.

- "Perfil d'usuari": es veuran les dades personals i les opcions de configuració de xarxes socials.

Un cop definides les necessitats d'estructura de l'*app*, l'exercici consistirà a fer un arbre de continguts que inclogui totes aquestes funcions.

2.5.3. Iteració 3: *wireframing* general

Seguint el mètode àgil, haurem tingut validacions del client abans d'haver acabat tot l'arbre i així podrem començar amb el prototipatge (o *wireframing*) de la nova intranet de manera paral·lela.

Per a fer els *wireframes*, hem de tenir en compte una sèrie de normes d'usabilitat, igual que per a qualsevol aplicació web.

Per a plantejar uns *wireframes* per a mòbil, hem de considerar les diferències però, ja que aquí hem de tenir en compte que l'usuari interacciona amb l'aparell de manera tàctil, i no amb un ratolí i un teclat com quan l'usuari és davant d'un ordinador.

També hem de tenir en compte que l'usuari navegarà per mitjà d'un aparell amb una pantalla considerablement més petita que la d'un ordinador, així que pot ser que hàgim de dissenyar els elements amb proporcions diferents de les que normalment veiem quan treballem per mitjà del web.

Web recomanat

Un bon recurs per a informar-nos de tots aquests detalls d'usabilitat mòbil és la pàgina de Jakob Nielsen:

http://www.useit.com/alertbox/mobile-usability.html

Jakob Nielsen

Nielsen exposa que l'habitual és que un usuari no llegeixi amb detall ni tan sols una mínima part dels textos d'una pàgina web. En comptes d'això, i per economia de temps, l'usuari es limita a llambregar la pàgina. És a dir, realitza un ràpid escombratge visual de cada pàgina buscant elements que li cridin l'atenció. Per tant, és fonamental la utilització d'elements com els següents:

- Paraules ressaltades mitjançant negreta i canvis de color o de grandària. En aquest sentit, els enllaços actuen com a elements d'atracció visual, ja que es destaquen de la resta del text.

- Llistes d'elements amb vinyetes o numerades.
- Títols de secció i titulars breus intercalats (també anomenats *titolets*).

A causa d'aquesta economia de lectura, segons Nielsen, el contingut d'un text s'ha d'organitzar correctament per a guanyar l'atenció del lector. Per exemple, les idees més importants han d'aparèixer al principi i, després, l'argumentació corresponent. D'aquesta manera, ens assegurem que el possible lector recordi més bé la informació. Nielsen recomana utilitzar menys del 50% del text que s'usa habitualment en una publicació escrita. Els usuaris s'avorreixen amb els textos llargs. Els paràgrafs han de ser curts, de dues o tres frases únicament i d'un estil molt directe.

D'altra banda, assegura que els usuaris aprenen aviat a ignorar els missatges publicitaris exagerats, fins i tot quan intenten aparèixer com a informació objectiva camuflats en el text.

Extret de la Viquipèdia

En la primera fase de realització de *wireframes*, hem de fer una proposta general de l'estructura. Primer de tot plantejarem la pàgina de registre i la *home page*.

En aquesta presentació farem dues versions per a la *home page*, per tal que el client es decideixi per una d'elles o una combinació de totes dues.

2.5.4. Iteració 4: *wireframing app*

Un cop aprovada la proposta d'estructuració dels continguts que haurem presentat amb la *home page*, durem a terme la resta de *wireframes* per a la iteració següent.

Fem un incís per comentar que en cada iteració podem aprofitar per comentar al client sobre la iteració següent. En aquest cas, hauríem comentat la nostra idea de *wireframing* en la propera presentació, de manera que és impossible que ens desviem molt del que s'imagina el client.

En aquesta fase desenvoluparem, doncs, la resta de *wireframes*. Com que és una aplicació relativament petita, no caldrà que separem la resta de *wireframes* en diferents iteracions i així accelerarem el desenvolupament del projecte.

2.5.5. Iteració 5: *look and feel*

Un cop aprovats els *wireframes*, la tasca següent consistirà a realitzar una proposta de ***look and feel*** per a l'*app*.

Així com en els *wireframes* hem comentat de fer dues versions (això és una decisió més personal i dependrà del projecte i del client), aquí nosaltres som partidaris de presentar només una opció de disseny. Bàsicament, perquè presentar més d'una opció de disseny sempre dóna lloc a confondre el client.

D'aquesta manera, nosaltres som els que valorarem la millor versió en els terrenys visual i funcional.

Tanmateix, en el brífing obtingut se'ns demanava de realitzar una aplicació que funcionés en tres tipus de dispositius: iPhone, Featured Phone i Basc Phone. A nivell de *wireframing* no és rellevant, ja que variarà molt poc l'estructura dels continguts, i només plantejarem la versió iPhone; però a partir d'aquí serà important visualitzar com quedarà el disseny en les tres plataformes.

Webs recomanats

Hi ha molts recursos pel que fa a gràfics, icones, menús, textures, tipografies, etc. a Internet. A continuació oferim uns quants portals de referència que ens ajudaran a l'hora de trobar els recursos que necessitem.

http://cssiphone.com/

http://www.mobileawesomeness.com/

http://www.refinedmobile.com/

http://tapfancy.com/

http://www.tappgala.com/

http://www.iosinspires.me/category/appinterfaces/

http://www.smashingmagazine.com/

http://www.alvit.de/handbook/

http://skout.co.za/

http://www.bluevertigo.com.ar/

Abans de posar-nos a treballar amb el programa, és molt important tenir en compte dues coses principals:

1) La resolució de pantalla: en aquest cas venia donada en l'apartat de brífing. Tanmateix, hem de tenir en compte que la retina *display* per a l'Iphone fa que la resolució canviï a 640 px d'amplada.

2) La quadrícula: una eina que ens ajudarà molt a l'hora de dissenyar és un bon sistema de quadrícules, o *grid system*. Normalment, aquests sistemes tenen possibilitats i es mesuren en nombre de columnes.

Retina

És una denominació de la pantalla *in-plane switching* (IPS) de 3,5 polzades (en diagonal) usada en l'iPhone 4, l'iPhone 4S, l'iPod touch de quarta generació i ara també en la pantalla del nou iPad de tercera generació, malgrat que la pantalla d'aquest últim és de només 264 ppp.

Rep aquest nom a causa de l'alta densitat de píxels que té, ja que amb 960 × 640 píxels conté quatre vegades més píxels que la pantalla usada en l'iPhone anterior (iPhone 3GS de 480 × 320 píxels), i manté les 3,5 polzades de pantalla, la qual cosa es tradueix en 326 ppp. Segons Steve Jobs, durant la seva presentació en la WWDC 2010, aquesta resolució es troba per sobre dels 300 ppp que encara són perceptibles per l'ull humà; per tant, els píxels d'aquesta pantalla són tan petits que no es poden distingir a la vista humana i no hi ha cap diferència amb el material imprès. L'únic dispositiu mòbil que ha superat la densitat de l'iPhone 4 fins al moment és el Sony Xperia S amb una pantalla de 4,3 polzades i 1.280 × 720 píxels, la qual cosa proporciona un ppp de 342.

Extret de la Viquipèdia

L'exercici aquí serà fer un disseny de la pàgina de registre i de la *home page* per a les tres plataformes. En aquesta fase no solament és important un bon disseny, atractiu i funcional, sinó també la presentació.

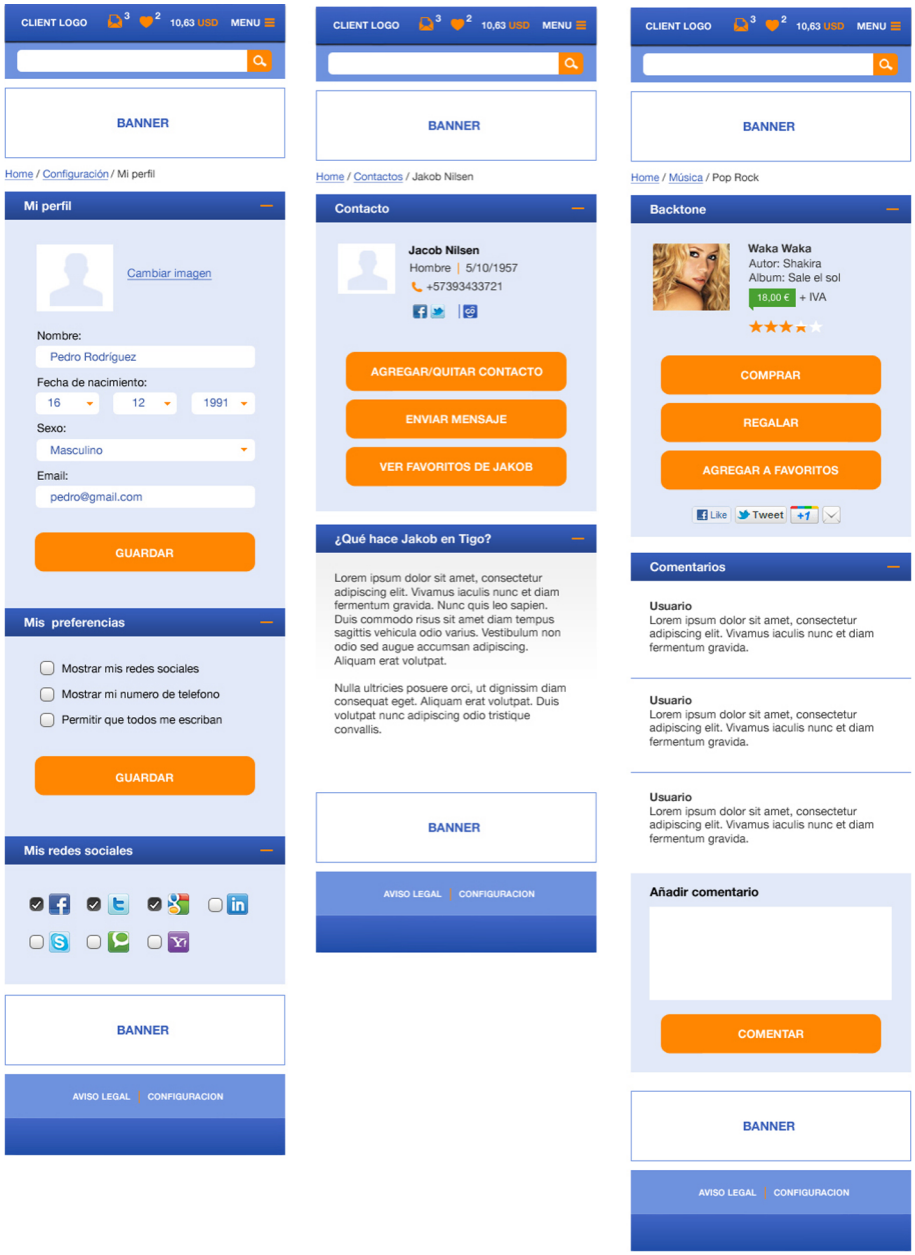
2.5.6. Iteració 6: disseny aplicació

Quan ens ve un disseny donat i hem de realitzar la resta de pantalles d'una aplicació, hem de ser molt curosos a respectar la línia gràfica i funcional.

Moltes vegades, el disseny de la resta de pantalles d'una aplicació i el del *look and feel* no són fets per la mateixa persona, i, si no és així, és important posar èmfasi en un bon traspàs de la informació.

A partir d'aquí, el repte més important és que el dissenyador que ha adquirit la responsabilitat de desenvolupar les pantalles a partir d'un disseny fet hi posi tot l'esforç i dedicació. Creiem que amb metodologies àgils és possible aconseguir més cohesió entre els membres de l'equip i, també, aquestes petites coses que poden succeir en un projecte.

En aquesta fase realitzarem totes les pantalles necessàries perquè els maquetistes puguin tenir tots els elements per a muntar l'aplicació en les tres plataformes. Realitzarem les pantalles distribuïdores i "especials", com la de contactes, la de detall o la de perfil.



2.5.7. Iteració 7: maquetació *mobile*

Com hem anat comentant al llarg de tot el cas pràctic, aquest projecte té fites amb el client i es basa en les metodologies àgils, per la qual cosa hem de tenir el client al corrent de com avança el projecte.

La part de maquetació no n'és una excepció, i també n'hem de mostrar l'evolució en els diferents *sprints*.

Podem avançar la part de maquetació un cop tinguem aprovat el disseny principal, ja que ens podrem basar en els estils generals i guiar-nos amb els *wireframes* desenvolupats prèviament.

En aquest cas només ens encarregarem de la producció parcial de l'aplicació en iPhone.

Podem produir l'aplicació de diverses maneres:

1) Primera opció

Desenvolupar-lo directament amb el llenguatge natiu d'Apple, l'Xcode.

Lectura complementària

Per a més informació sobre XCode, podeu consultar l'adreça d'Internet següent:
http://developer.apple.com/xcode/

Xcode

És l'entorn de desenvolupament integrat (IDE, en les sigles en anglès) d'Apple Inc. i se subministra gratuïtament juntament amb Mac OS X. Xcode treballa conjuntament amb Interface Builder, una herència de NeXT, que és una eina gràfica per a la creació d'interfícies d'usuari.

Xcode inclou la col·lecció de compiladors del projecte GNU (GCC), i pot compilar codi C, C++, Objective-C, Objective-C++, Java i AppleScript mitjançant una àmplia gamma de models de programació, incloent-hi, però no limitat a Cocoa, Carbó i Java. Altres companyies han afegit suport per a GNU Pascal, 1 Free Pascal, 2 Ada i Perl.3.

Extret de la Viquipèdia

En aquest cas, nosaltres no entraríem en el desenvolupament de l'aplicació, ja que els programadors adaptarien directament els dissenys en el llenguatge natiu d'Apple.

2) Segona opció

Actualment, hi ha molts desenvolupadors que fan servir alguns *frameworks* javascript per a desenvolupar l'aplicació en format web (*web app*) i, després, hi ha programes que transformen aquesta *web app* a llenguatge natiu per a mòbil.

Quan s'utilitza un *framework* per a crear una *web app* es fa una maquetació amb HTML5 i CSS i després s'usa el *framework* per a emular les funcionalitats natives de mòbil.

Framework

La paraula anglesa *framework* defineix, en termes generals, un conjunt estandarditzat de conceptes, pràctiques i criteris per a enfocar un tipus de problemàtica particular, que serveix de referència per a afrontar i resoldre nous problemes de caràcter similar.

En el desenvolupament de programari, un *framework* o infraestructura digital, és una estructura conceptual i tecnològica de suport definit, normalment amb artefactes o mòduls de programari concrets, amb base a la qual un altre projecte de programari pot ser més fàcilment organitzat i desenvolupat. Típicament, pot incloure suport de programes, biblioteques i un llenguatge interpretat, entre altres eines, per tal d'ajudar a desenvolupar i unir els diferents components d'un projecte.

Extret de la Viquipèdia

Els *frameworks* més comuns són els següents:

- JqueryMobile
- SensaTouch

I les aplicacions més comunes per a transformar de *web app* a llenguatge natiu per a qualsevol dispositiu mòbil són les següents:

- PhoneGap
- Titanium Appcelerator

Aquestes aplicacions van bé per a projectes senzills de poca complexitat. Per a desenvolupar una aplicació més complexa caldria fer-ho directament amb llenguatge natiu per a cada una de les plataformes.

En aquest cas, el client ens demanava els HTML i CSS per a poder desenvolupar l'aplicació amb SensaTouch i PhoneGap, així que nosaltres ens vam haver d'encarregar de maquetar les plantilles per a mòbil.

L'exercici aquí serà fer la maquetació de les pantalles que tenim dissenyades per a passar-les als programadors, perquè implementin el *framework* i ho transformin a llenguatge natiu amb les aplicacions que hem mostrat abans.

2.5.8. Iteració 8: lliurament final del projecte

Aquesta fase sempre és la més senzilla en un projecte si s'utilitzen metodologies àgils, ja que el client sap exactament el que se li entregarà. Sempre hi ha un marge d'error, però amb aquest tipus de metodologia serà molt petit.

En aquesta fase corregirem els petits errors que puguin sorgir, i donarem per finalitzat el projecte.

2.6. Enllaços d'interès

http://es.wikipedia.org/wiki/Wikipedia:Portada

http://cssiphone.com/

http://www.mobileawesomeness.com/

http://www.refinedmobile.com/

http://tapfancy.com/

http://www.tappgala.com/

http://www.iosinspires.me/

http://www.useit.com/

http://www.smashingmagazine.com/

http://www.alvit.de/handbook/

http://skout.co.za/

http://www.bluevertigo.com.ar/

http://developer.apple.com/xcode/

http://jquerymobile.com/

http://www.sencha.com/products/touch

http://phonegap.com/

http://www.appcelerator.com/

3. Disseny d'una interfície per a *smart TV*

3.1. Introducció

Amb Apple i Google treballant amb les *smart TV*, creiem que aquest cas serà força útil per als estudiants que s'estiguin plantejant d'especialitzar-se en disseny i desenvolupament d'aplicacions per a televisió interactiva.

Quan es tracta de dissenyar una aplicació per a televisió interactiva, hem de tenir en compte tots els principis d'interacció, igual que quan treballem en web o en aplicacions mòbils. Tanmateix, hem de pensar que l'experiència de l'usuari amb la interfície serà més limitada, ja que l'usuari interaccionarà amb un comandament de televisió.

Per això hem de plantejar una interfície en què la navegació serà més arcaica i l'usuari només podrà anar amunt, avall, a la dreta i a l'esquerra.

Televisió intel·ligent

La televisió intel·ligent (*smart TV* o també traduït com a *televisió híbrida*) descriu la integració d'Internet i de les característiques Web 2.0 a la televisió digital i al *set-top box*, i també la convergència tecnològica entre els ordinadors i aquests televisors i el STB. Aquests dispositius se centren en els mitjans interactius en línia, en la televisió per Internet, i en altres serveis com el vídeo a la carta.

La tecnologia dels *smart TV* no solament s'incorpora en els aparells de televisió, sinó també en altres dispositius com la *set-top box*, l'enregistrador de vídeo digital, els reproductors Blu-ray, les consoles de jocs i els *home cinemas*, entre d'altres. Aquests dispositius permeten als espectadors buscar i trobar vídeos, pel·lícules, fotografies i altres continguts en línia, en un canal de televisió per cable, en un canal de televisió per satèl·lit o emmagatzemat en un disc dur local.

Extret de la Viquipèdia

3.2. Objectius

Com en els altres casos, l'objectiu principal d'aquest exemple serà desenvolupar un projecte des del plantejament inicial fins al *mock-up* de la interfície. Per tant, després de llegir el cas, sereu capaços de plantejar un projecte d'aquest tipus, fent èmfasi en les metodologies àgils.

Així doncs, a partir d'un brífing veurem com es defineix l'arquitectura d'informació, els *wireframes*, trobarem un *look and feel* adequat i aprendrem a realitzar un *mock-up*. En aquest cas específic, posarem èmfasi també en la interacció amb el comandament i la iconografia, que adoptarà un paper molt important en aquest tipus d'interfície.

Lectures complementàries

Trobareu més informació sobre *mocks-up* en els enllaços següents:

http://en.wikipedia.org/wiki/Mockup

http://www.interaction-design.org/encyclopedia/mock-ups.html

3.3. Equip

Com en la majoria de projectes que apareixen avui dia (mòbil, televisió, Internet...), en què la tecnologia ens permet evolucionar més de pressa del que podem, és impossible trobar professionals amb vint anys d'experiència en desenvolupament de projectes d'aquest tipus. El que sí que trobem són professionals autodidactes amb coneixements flexibles.

A continuació definim l'equip de producció:

- *Product owner*.
- Scrum màster.
- UX/UI *designer*.
- IxD/*visual designer*.
- *Flash developer*.

Aquest projecte s'acabarà amb un *mock-up* de la interfície, així que necessitarem un *flash developer* per a simular l'aplicació.

3.4. Brífing

Aquest cas és real i es va desenvolupar dins d'una empresa que ja tenia un producte de televisió per cable, que oferia un servei d'EPG i videoclub.

Es necessitava una evolució del producte i migrar cap a una *smart TV*, així que el brífing es va plantejar internament per veure quines eren les possibilitats i quines funcionalitats es volien aplicar.

Així doncs, es va voler migrar el que hi havia cap a una interfície en què l'usuari fos capaç de:

- Crear un compte d'usuari.
- Crear un recomanador de sèries i pel·lícules.
- Seleccionar els programes preferits.
- Disposar d'una cartellera de pel·lícules i sèries, amb les funcionalitats de comprar o llogar.
- Enregistrar programes en directe, sèries o pel·lícules preferides.
- Millorar la navegació i l'experiència de l'usuari.
- Obtenir un *look and feel* més actual utilitzant iconografia.
- Accedir a Internet.

- Incorporar publicitat.

Guia electrònica de programes

La *guia electrònica de programes* (*electronic programming guide*, EPG) és un servei televisiu que informa d'una manera ràpida i senzilla de tota la programació d'un canal de televisió o de ràdio. El servei funciona només en emissions digitals com la TDT i la ràdio digital DAB. Representa l'evolució a l'era digital del tradicional servei de programació que ens ofereix el teletext. En el cas de la ràdio, representa una novetat, ja que en la tecnologia més antiga, com ho era l'RDS, no hi havia un servei que fes la mateixa funció. Alguns camps típics són, pel que fa al contingut, títol, data, dades PDC, gènere, edat mínima, sinopsi, directe o diferit, *copyright*, subtítols de teletext, banda sonora, idioma original, etc. Pel que fa a les característiques de l'emissió, es pot informar sobre canal, hores de començament i acabament, PALplus, sistema d'encriptació, programa repetit, format del so, format televisiu i idioma.

Extret de la Viquipèdia

3.5. Iteracions

3.5.1. Iteració 1: anàlisi del projecte

Per a fer una anàlisi d'un projecte d'aquest tipus, hem de mirar altres *smart TV* que hi hagi en el mercat. Les primeres que hauríem de mirar són les d'Apple i Google, perquè és conegut de tothom que són les empreses pioneres en aquest mercat.

Actualment, aquestes dues encara no han sortit al mercat, però sí que podem mirar televisions interactives com les següents:

- Samsung Smart TV platform
- Sony Bravia Internet Video smart TV platform
- Panasonic Viera Connect / Smart Viera platform
- LG Smart TV platform
- Toshiba Places online TV platform
- Philips NetTV platform
- Sharp AQUOS Net smart TV platform

Un cop analitzades les característiques d'aquestes aplicacions, podrem centrar-nos en el desenvolupament de la nostra, i agafar les *best practices* que s'ajustin als nostres objectius.

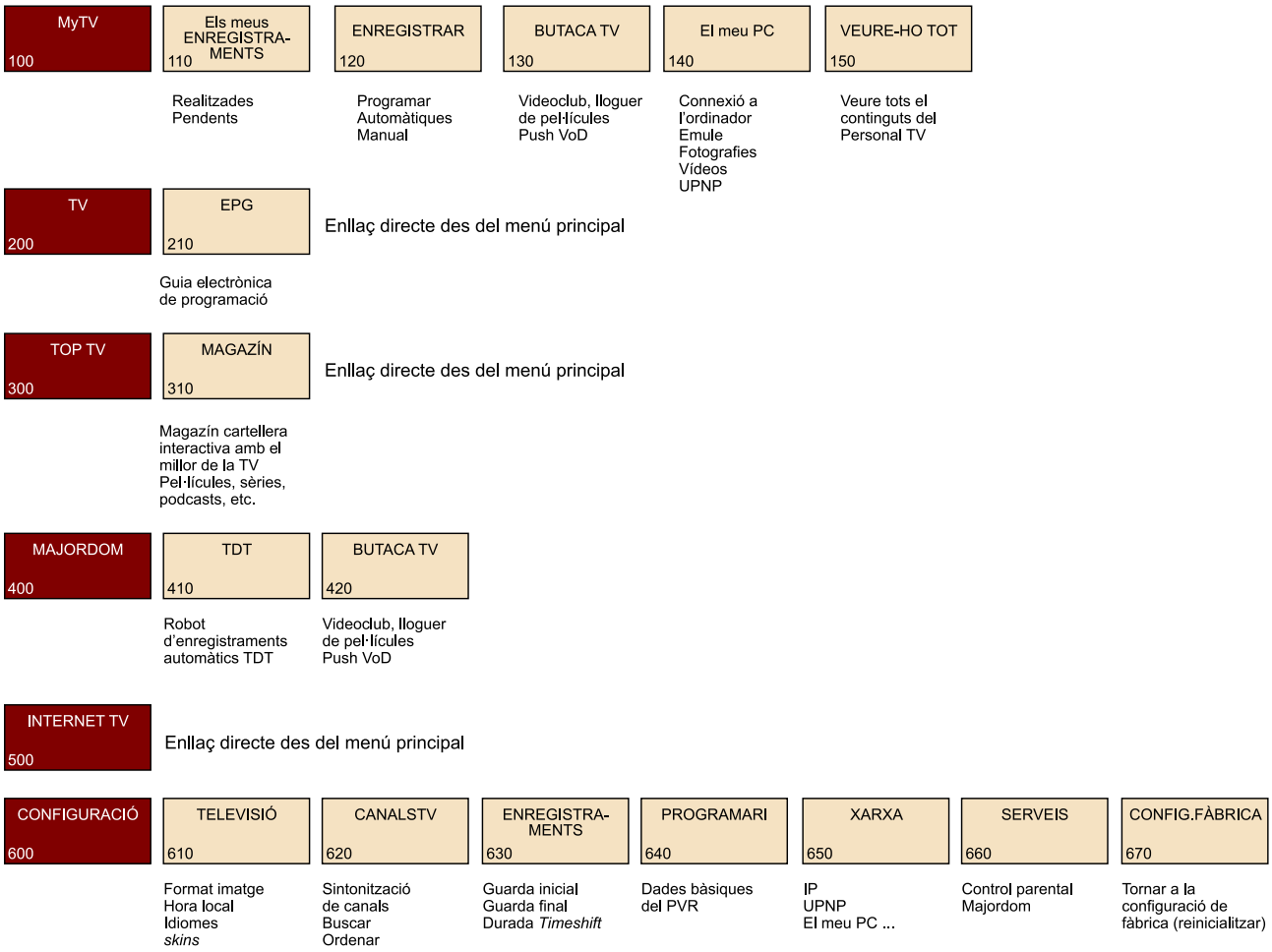
3.5.2. Iteració 2: arquitectura de la informació

En aquest punt, ens hem de reunir amb els directors de producte (els nostres clients) per a definir exactament quines volem que siguin les nostres funcionalitats i així poder jerarquitzar i estructurar la informació.

Tal com hem dit en la introducció, hem de tenir en compte la limitació de l'usuari per a accedir a la informació utilitzant un comandament a distància. Així doncs, un cop definits tots els continguts que apareixeran en la interfície, l'UX/UI *designer* s'encarregarà d'estructurar la informació.

L'exercici aquí serà plantejar una arquitectura seguint el brífing obtingut.

A continuació, us mostrem un exemple de com podria ser aquesta arquitectura.



3.5.3. Iteració 3: *wireframing*

Com hem comentat en aquest cas, el client són els directors d'àrea, però el procediment de metodologies àgils que apliquem serà exactament el mateix que en els altres casos. Farem iteracions igualment.

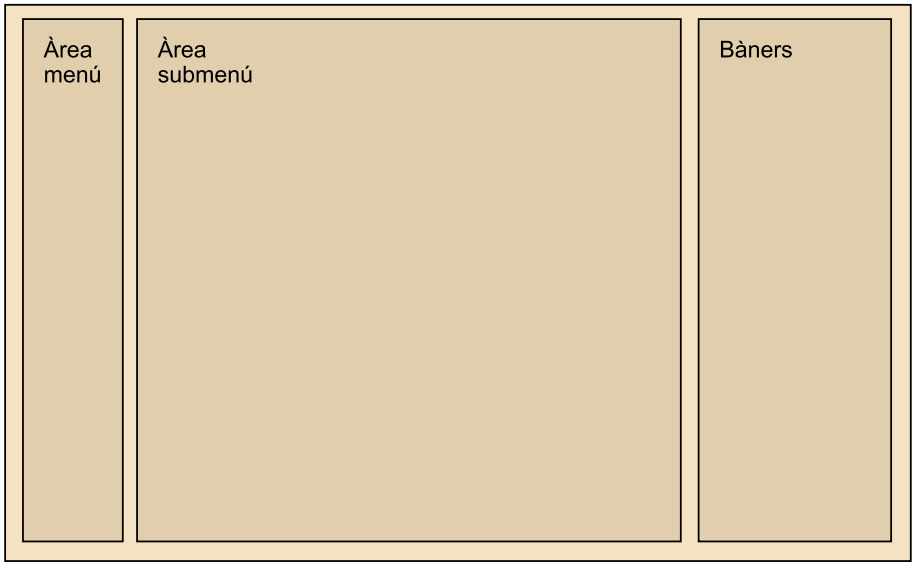
Un cop arribats en aquest punt, haurem aprovat l'arquitectura d'informació de la nostra nova *smart TV*, així que durem a terme els *wireframes* o esquelet per a visualitzar com aniran estructurades cadascuna de les pantalles.

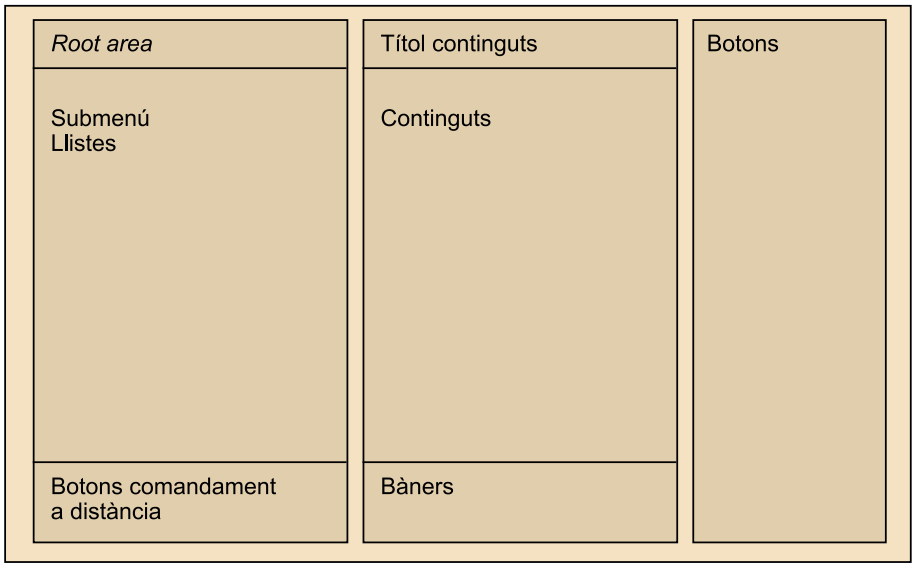
Hi ha un factor molt important que cal tenir en compte quan es desenvolupen *wireframes* de qualsevol tipus. Ser econòmics. És a dir, hauríem d'intentar que amb el nombre mínim de *wireframes* possible es pogués encabir tota la informació que volem.

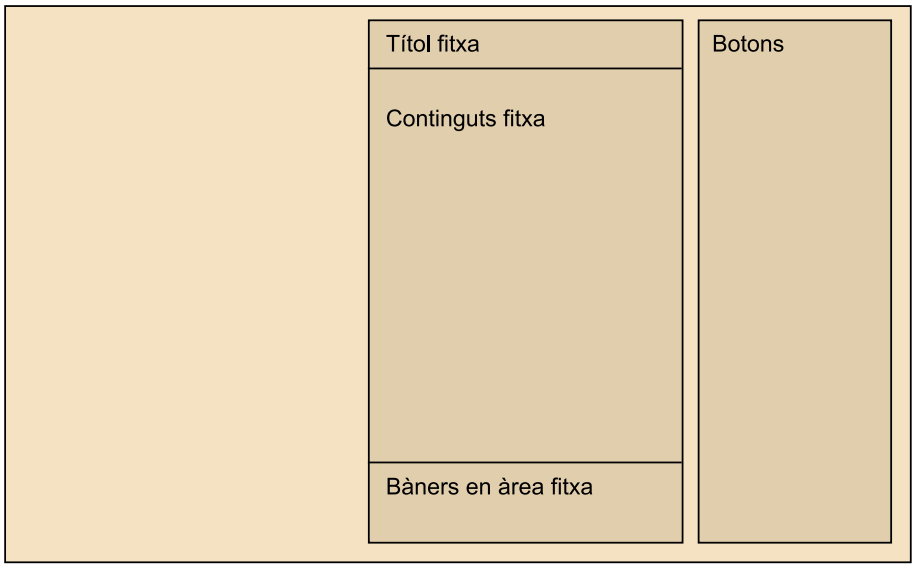
Aquest és un punt importantíssim per a la rendibilització d'un projecte, ja que estalviarem recursos tant en el disseny com en el desenvolupament posterior.

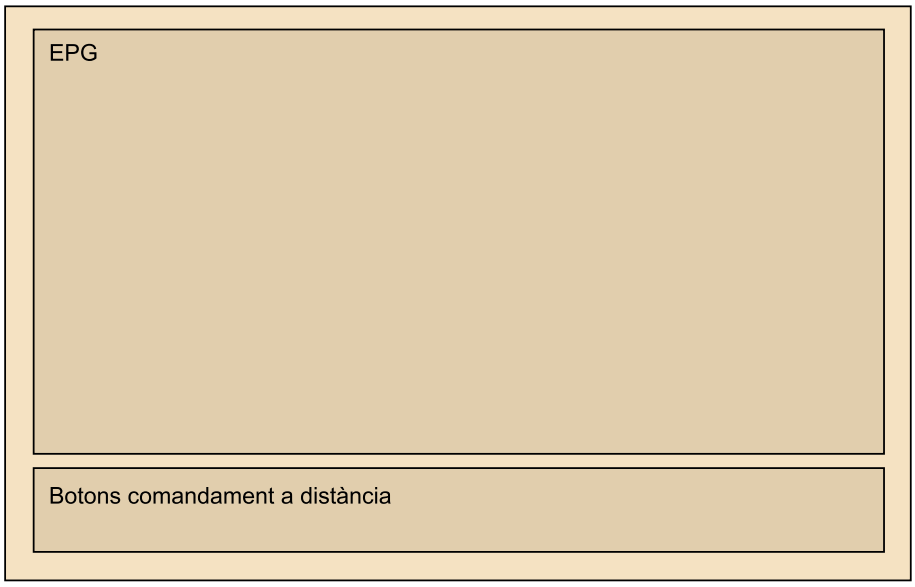
L'exercici aquí serà, amb els mínims *wireframes* possibles, intentar plantejar tota la nostra nova interfície.

A continuació us mostrem un exemple de com podrien ser aquests *wireframes*.




Root area
Submenú
Llistes
Botons comandament
a distància
Títol continguts
Continguts
Bàners
Botons


Títol fitxa
Continguts fitxa
Bàners en àrea fitxa
Botons


EPG
Botons comandament a distància

3.5.4. Iteració 4: comandament a distància

Un punt bàsic a l'hora també de crear l'experiència d'usuari és el disseny del comandament a distància, en què hem de dotar el comandament de les funcionalitats que volem que l'usuari apliqui en la nostra interfície.

Comandament a distància

Un **control remot** o **comandament a distància** és un dispositiu electrònic usat per a realitzar una operació remota (o telecomandament) sobre una màquina.

El terme s'empra generalment per a referir-se al comandament a distància (en general anomenat simplement *comandament* o, a l'Amèrica Llatina, *control*) per al televisor o un altre tipus d'aparell electrònic propietari, com ara DVD, Hi-Fi, ordinadors, i per a encendre i apagar un interruptor o l'alarma, o obrir la porta del garatge. Els comandaments a distància per a aquests aparells són normalment objectes petits (fàcilment manipulables amb una mà) amb una matriu de botons per a ajustar els diferents valors, com, per exemple, el canal de televisió, el número de cançó i el volum. De fet, en la majoria de dispositius moderns el comandament conté totes les funcions de control, mentre que el mateix aparell controlat només disposa dels controls més primaris. La majoria d'aquests controls remots es comuniquen amb els aparells respectius per senyals d'infraroig (ANAR) i només uns quants utilitzen senyals de ràdio. En els vehicles moderns, les claus tradicionals incorporen ara comandaments a distància amb diverses funcions. Com a font d'energia normalment disposen de petites piles de tipus AA, AAA o de botó.

Extret de la Viquipèdia

Actualment, els comandaments a distància tenen moltes funcionalitats que no utilitza gairebé ningú.

Moltes vegades realitzar un comandament a distància no estàndard és un cost que una empresa no es pot permetre, i s'opta per agafar un comandament del mercat que s'ajusti a les funcionalitats que plantegem en la nostra nova interfície.

L'exemple que us mostrem a continuació és un comandament estàndard, que no respecta les característiques que busquem per a la nostra aplicació, ja que tantes funcionalitats distrauran l'usuari.

L'exercici aquí serà fer un comandament que incorpori les necessitats que realment tenim.

3.5.5. Iteració 5: *look and feel*

Aquesta part és la més delicada, en què definirem la gràfica i l'estil de l'aplicació. Quan treballem amb televisió interactiva, adoptarem una mica els principis de la interacció en *tablets* o en dispositius mòbils.

Per bé que, a diferència de les *tablets* i les aplicacions mòbils, l'usuari se situa lluny de la pantalla, hem de crear igualment iconografia, utilitzar botons grans, lletra gran i diferenciar molt bé la jerarquia de la informació.

A part d'això, en televisió interactiva hi ha un altre factor que adquireix importància, i és el fet de saber diferenciar molt bé l'àrea de navegació i d'interacció de l'usuari amb la reproducció de vídeo que hi hagi en aquell moment. Per això hem de delimitar bé les àrees, per a la qual cosa podem utilitzar transparències, animacions, colors i imatges.

L'exercici aquí, doncs, serà realitzar un disseny atractiu i funcional per a la nova interfície, un cop definits l'arquitectura d'informació i els *wireframes*.

3.5.6. Iteració 6: iconografia

Per a qualsevol interfície, actualment és molt important utilitzar la iconografia, i en aquest cas considerem que és vital.

I diem *vital* perquè, tal com hem comentat abans, l'usuari se situa lluny de la pantalla, amb un comandament que el limita força, i qualsevol cosa que no sigui motivar la seva interacció farà que ràpidament la rebutgi.

Per tant, la iconografia serà una ajuda més i caldrà que sigui molt intuïtiva.

Com a apunt, cal destacar que, quant a la usabilitat, la iconografia si no està normalitzada sempre ha d'anar acompanyada de text descriptiu.

L'exercici aquí serà realitzar una iconografia d'acord amb el *look and feel* que hem definit anteriorment.

A continuació us mostrem un exemple de com podria ser aquesta iconografia.

La meva TV

TV

Top TV

Majordom

Internet TV

Configuració

Els meus enregistraments

Enregistrar

Butaca TV

El meu PC

Veure-ho tot

EPG

Majordom TDT

Ok, acceptar

Ajuda contextual

Televisió

Canals

Enregistraments

Programari

Xarxa

Serveis

Configuració fàbrica

3.5.7. Iteració 7: *mock-up*

Com hem dit abans, un *mock-up* és un exemple força real de l'aplicació. En aquest cas, creiem que el millor programari per a desenvolupar-la és Flash, ja que ens permet manipular la interacció i reproduir vídeo.

Amb el Flash no hi haurà la interacció amb el comandament a distància, però podrem veure davant d'un ordinador com funciona la interfície.

El *mock-up* principalment s'utilitzarà per a poder realitzar tests d'usuari, en què agafarem voluntaris o persones de diferents rangs socials, edats, etc., per a poder millorar la nostra *smartTV* .

Què és el mètode de test amb usuaris?

És una prova d'usabilitat que es basa en l'observació i l'anàlisi de la utilització que un grup d'usuaris reals fa del lloc web, en què s'anoten els problemes d'ús amb els quals es troben per a poder solucionar-los posteriorment. Es tracta d'una prova duta a terme al **laboratori**, és a dir, no s'ha de confondre amb un estudi de camp.

Quan s'ha de dur a terme

"Com més tard, pitjor."

Com tota avaluació d'usabilitat, com més esperem a realitzar-la, més costosa serà la reparació dels errors de disseny descoberts.

Això vol dir que no solament hem d'efectuar aquest tipus de prova sobre el lloc web una vegada implementat, sinó també sobre els prototips del lloc.

"Sempre, després d'una avaluació heurística."

Si duem a terme un test amb usuaris sense haver realitzat prèviament una avaluació heurística, probablement prestarem massa atenció a problemes d'ús que trobin els participants que podrien haver estat descoberts amb una simple avaluació heurística.

Un test amb usuaris és més costós que una avaluació heurística, per la qual cosa seria malgastar temps i diners utilitzar-lo per a descobrir errors de disseny motivats pel no-compliment en el disseny de principis generals d'usabilitat (heurístics).

Extret de la pàgina d'Internet http://www.nosolousabilidad.com/articulos/test_usuarios.htm

Lectura complementària

Trobareu més informació sobre l'avaluació heurística en l'enllaç següent:

http://www.nosolousabilidad.com/articulos/heuristica.htm

3.5.8. Iteració 8: lliurament final del projecte

En la part del *mock-up*, en què realitzarem els tests d'usuari, segurament haurem de fer unes quantes repeticions, fins que el producte estigui optimitzat.

Un cop ho haurem aconseguit, traslladarem tot el material als desenvolupadors que s'encarregaran de programar l'aplicació.

3.6. Enllaços d'interès

http://es.wikipedia.org/wiki/Wikipedia:Portada

http://www.interaction-design.org/

http://www.nosolousabilidad.com/

4. Desenvolupament d'una aplicació mòbil per a sistemes Android

4.1. Introducció

El projecte consistirà a desenvolupar una aplicació mòbil per al sistema Android. L'aplicació presentarà el mateix contingut que una web ja existent. Això inclourà el contingut de les mateixes pàgines, un sistema de cerca textual i un sistema de votació de continguts.

Mitjançant l'aplicació es podrà accedir als continguts des de mòbils i *tablets* amb sistema operatiu Android.

4.2. Què és Android?

Android és el sistema operatiu de Google per a mòbils. És un sistema basat en Linux que s'utilitza per a sistemes mòbils de tipus *smartphone* i tauletes tàctils.

Android forma part de l'Open Handset Alliance, una agrupació d'empreses fabricants de maquinari per a dispositius mòbils.

4.3. Per què Android?

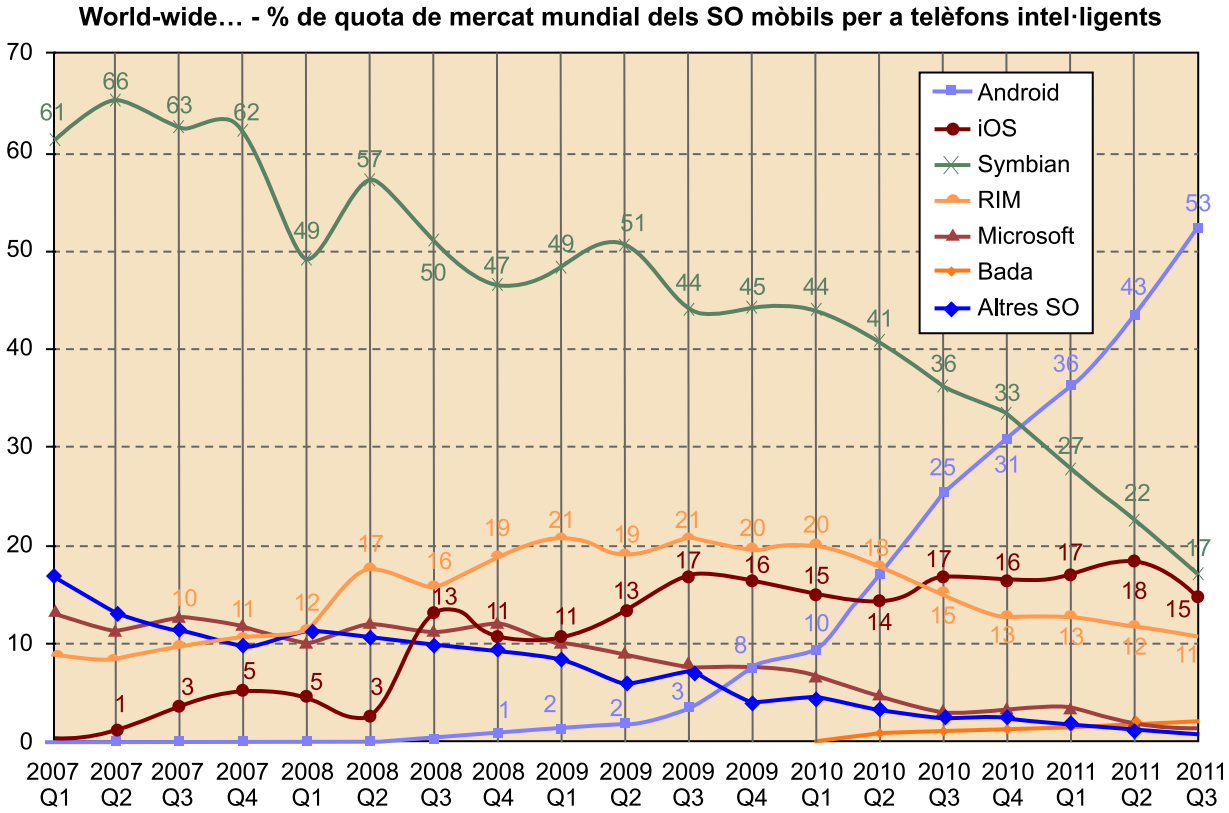
4.3.1. Avantatges

Actualment, el sistema Android és el que té més quota de mercat en l'àmbit dels *smartphones*. Hi ha diverses empreses que produeixen dispositius Android i la seva evolució no està determinada per un d'ells en exclusiva.

Apple

En el cas d'Apple, només Apple fabrica i desenvolupa l'iPhone i tot el seu entorn.

La gràfica següent mostra l'evolució d'aquesta progressió per quadrimestres (dades a escala mundial):



Font: Gartner. *Dades quadrimestrals* [en línia]. http://www.gartner.com/it/page.jsp?aneu=1924314

Lectura complementària

Trobareu més informació sobre els sistemes operatius per a mòbils en la pàgina d'Internet següent:

http://en.wikipedia.org/wiki/mobile_operating_system

Quadrimestre	An-droid	iOS	Sym-bian	RIM	Micro-soft	Bada	Un altre
2011 Q4	50,9%	23,9%	11,7%	8,8%	1,9%	2,1%	0,8%
2011 Q3	52,5%	15,0%	16,9%	11,0%	1,5%	2,2%	0,9%
2001 Q2	43,4%	18,2%	22,1%	11,7%	1,6%	1,9%	1,0%
2001 Q1	36,0%	16,8%	27,4%	12,9%	3,6%	1,7%	1,6%
2010 Q4	31,1%	16,1%	32,9%	13,1%	3,4%	1,3%	2,2%
2010 Q3	25,3%	16,6%	36,3%	15,4%	2,8%	1,1%	2,5%
2010 Q2	17,2%	14,2%	41,2%	18,2%	5,0%	0,9%	3,3%
2010 Q1	9,6%	15,3%	44,2%	19,7%	6,8%		4,4%
2009 Q4	7,6%	16,2%	44,7%	19,7%	7,9%		4,0%
2009 Q3	3,4%	17,0%	44,2%	20,5%	7,9%		7,0%
2009 Q2	1,8%	13,0%	51,0%	19,0%	9,3%		5,9%
2009 Q1	1,6%	10,5%	48,8%	20,6%	10,2%		8,2%

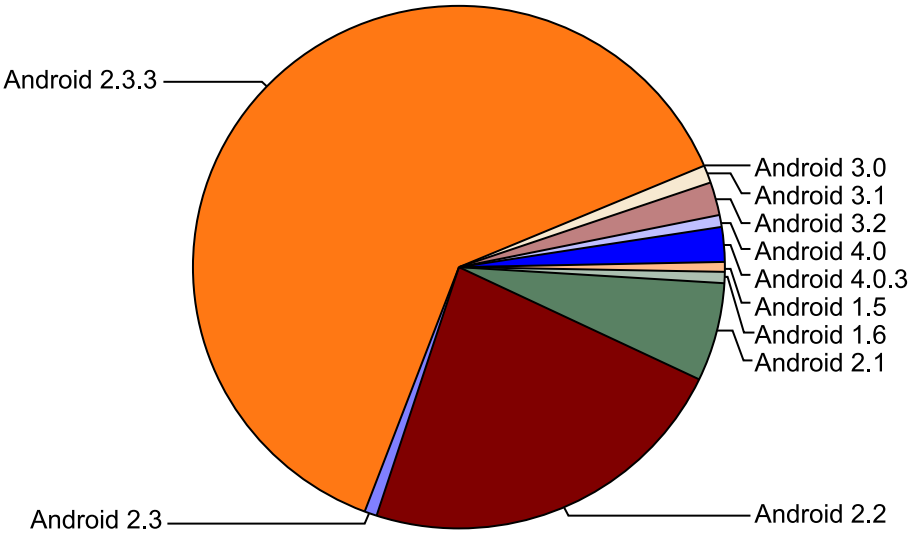
Font: Gartner. *Dades quadrimestrals* [en línia]. http://www.gartner.com/it/page.jsp?id=1924314

Quadrimestre	Android	iOS	Symbian	RIM	Microsoft	Bada	Un altre
2008 Q4	1,1%	10,6%	46,5%	19,3%	12,2%		9,1%
2008 Q3	0,6%	13,1%	50,3%	16,1%	11,2%		9,8%
2008 Q2		2,8%	57,5%	17,5%	12,1%		10,8%
2008 Q1		4,6%	49,5%	11,6%	10,4%		11,6%
2007 Q4		5,2%	62,3%	10,9%	11,9%		9,6%
2007 Q3		3,4%	63,1%	9,7%	12,8%		11,5%
2007 Q2		1,0%	65,6%	8,9%	11,5%		13,0%
2007 Q1			61,2%	8,7%	13,4%		16,8%

Font: Gartner. *Dades quadrimestrals* [en línia]. http://www.gartner.com/it/page.jsp?id=1924314

4.3.2. Inconvenients

A causa de la gran quantitat de fabricants i de dispositius que utilitzen Android, hi ha més fragmentació d'aquest tipus de dispositiu. Cadascun d'aquests dispositius pot tenir resolucions diferents, tenir o no tenir càmera i moltes altres diferències de funcionalitat.



Fragmentació entre versions. Font: Android developers [en línia]. http://developer.android.com/resources/dashboard/platform-versions.html

Com que hi ha aquesta diversitat de funcionalitats, serà responsabilitat del desenvolupador distingir i verificar si estan disponibles i en quina mesura.

En una plataforma més tancada i amb un únic fabricant, podem estar més segurs de les capacitats del dispositiu final que executarà l'aplicació.

Videojocs

Trobem una analogia del que succeeix en el mercat de *smartphones* en el tradicional mercat de videojocs entre PC i consoles de joc. En gran mesura, això ja ha canviat, però històricament el desenvolupador per al mercat d'ordinador personal ha de gestionar moltes

possibles combinacions de targetes gràfiques i configuracions, mentre que si desenvolupa per a una consola de joc específica es pot aprofitar al màxim el maquinari, que és el mateix per a tots els usuaris. Aquest ha estat sempre un dels atractius de les consoles de joc per a usuaris que simplement volen jugar, sense preocupar-se de configurar els seus sistemes o d'enfrontar-se a possibles incompatibilitats.

Un altre possible inconvenient que cal tenir en compte és que tots els estudis apunten al fet que els usuaris d'iPhone compren més aplicacions que els d'Android, encara que això es pot veure compensat per la base d'usuaris més gran d'Android.

4.4. Opcions per a desenvolupar en Android

En darrera instància, les aplicacions instal·lables des de l'Android Market estan desenvolupades en codi Java. No obstant això, podem generar aquesta aplicació desenvolupant directament el codi font en Java o utilitzant alguna altra plataforma que generi per nosaltres aquest codi.

A continuació presentem dues de les opcions més populars:

Més eines

Altres eines i entorns, en aquest cas sobretot per al desenvolupament de videojocs, són els següents: Unity o Moai.

1) Natives. Desenvolupar l'aplicació en Java permetrà el màxim control i rendiment sobre la plataforma Android. D'altra banda, hi ha un gran nombre de desenvolupadors en Java, per la qual cosa, una vegada coneguda l'API d'Android i el seu entorn, és més fàcil per a ells familiaritzar-se amb l'entorn de desenvolupament en aquesta plataforma.

2) Phonegap. Permet desenvolupar una aplicació en html i javascript. Aquesta plataforma ens permet aprofitar els coneixements en el desenvolupament web per a dur a terme aplicacions independents. Aquesta eina permet, a més, exportar a múltiples sistemes operatius, a més d'Android, amb una sola base de codi. Perquè això funcioni caldrà tenir-ho en compte des de l'inici del desenvolupament del projecte.

Algunes de les plataformes suportades per Phonegap són les següents:

- iOS
- Android
- Blackberry
- Windows Phone
- WebOS
- Symbian

4.5. Consideracions per al desenvolupament

4.5.1. Llenguatge

Les aplicacions en Android utilitzen com a llenguatge de desenvolupament el llenguatge de programació Java.

Llenguatge Java

Tècnicament, la màquina virtual desenvolupada per Google s'anomena *Dalvik* i és diferent de la màquina virtual estàndard de Java (JVM), encara que a l'efecte del desenvolupament del projecte podem ignorar això d'una manera segura i dir, simplement, que utilitza el llenguatge Java.

D'aquesta manera, el model de desenvolupament per a Android és familiar per a molts desenvolupadors.

4.5.2. Instal·lació de l'entorn

Per a desenvolupar per a un sistema Android, necessitem instal·lar el ***system development kit*** corresponent. El podem trobar a http://developer.android.com/sdk/index.html.

L'SDK de Google està desenvolupat en Java i requereix, per tant, el JDK[1] per a executar-lo. Podem descarregar l'última versió del JDK a http://www.oracle.com/technetwork/java/javase/downloads/index.html.

(1)Acrònim de *Java development kit*.

Per al desenvolupament d'aplicacions Android, utilitzarem una IDE[2] (permet editar codi, gestionar les versions, compilar, realitzar canvis d'estructura i altres funcionalitats des d'un sol programa) molt utilitzada en el desenvolupament en Java. Es tracta d'**Eclipse** i podem descarregar-la a www.eclipse.org/downloads/. La versió recomanada és la Classic.

(2)Acrònim d'*integrated development environment.*

Altres editors

Podem utilitzar altres editors o eines per a gestionar un projecte per a Android. Dos editors molt coneguts per a programar, però amb una dificultat d'entrada molt alta, són per exemple:

1) Emacs:

- http://www.gnu.org/software/emacs/
- http://emacswiki.org/

2) VIM:

- http://en.wikipedia.org/wiki/vim_%28text_editor%29
- http://www.vim.org/

L'ús d'editors com aquests és diferent del d'un IDE en diversos aspectes, però és possible incorporar-los en el flux de desenvolupament d'una aplicació Android.

Aquí ens centrem en l'opció basada en Eclipse perquè és una de les més senzilles com a introducció i la recomanada per Google. Si es volen utilitzar eines més independents i

senzilles, i controlar-ne cadascun dels components, caldrà utilitzar directament les utilitats que podem trobar en l'SDK de Google.

Google ens proporciona utilitats de línia d'ordres per a gestionar tots els aspectes del desenvolupament de manera pròpia.

Trobem algunes bones referències sobre aquestes ordres en les adreces d'Internet següents:

- http://developer.android.com/guide/developing/building/building-cmdline.html
- http://developer.android.com/guide/developing/tools/android.html
- http://developer.android.com/guide/developing/devices/emulator.html
- http://developer.android.com/guide/developing/tools/adb.html

L'avantatge d'aprendre a utilitzar aquestes utilitats és que permeten la integració d'aquestes funcionalitats en el nostre flux de treball. Es poden incorporar en *scripts* propis i en les nostres pròpies eines, per la qual cosa permeten treballar més còmodament. En qualsevol cas, això implica una inversió inicial o continuada per a polir el flux de desenvolupament que no és apte per a tots els equips de desenvolupament.

En aquest punt tenim ja la màquina de Java, Eclipse i l'SDK de Google. Necessitem també el *plug-in* ADT[3] per a Android que ens farà accessibles des d'Eclipse les funcionalitats de l'SDK.

(3)Acrònim d'*Android development tools.*

ADT

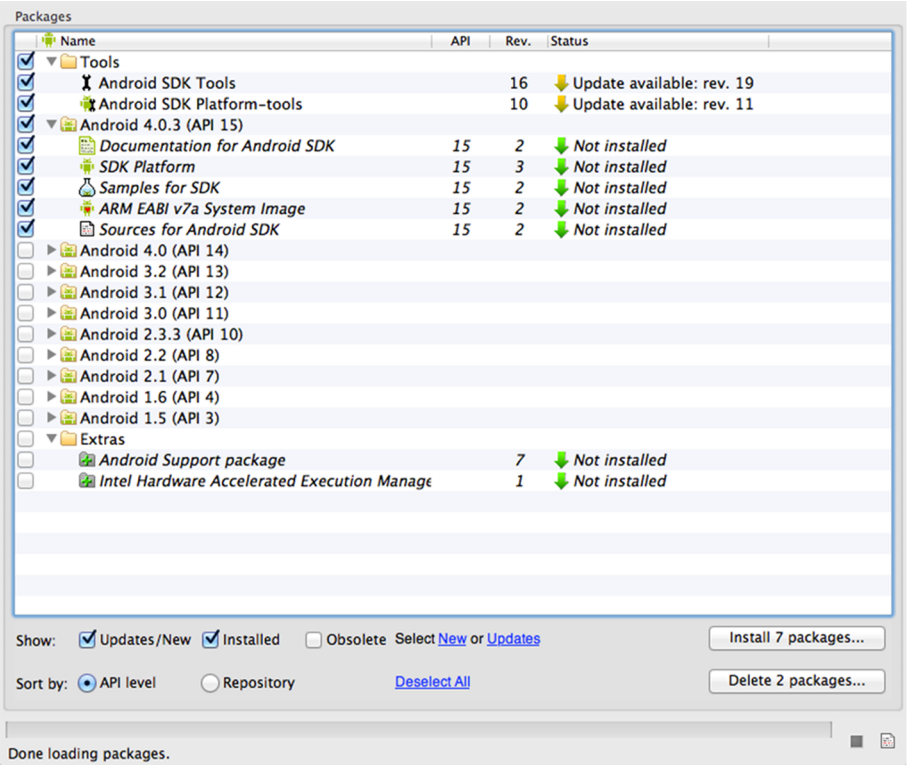
El procés d'instal·lació d'aquest *plug-in* és el mateix que tots els *plug-in* d'Eclipse. Podem trobar-ne una guia pas per pas en l'adreça http://developer.android.com/sdk/eclipse-adt.html#installing.

Amb el *plug-in* instal·lat, només ens falta instal·lar les plataformes sobre les quals treballarem i altres extres. Podem instal·lar diferents versions del sistema operatiu Android per verificar l'aplicació en múltiples versions o executar-les en l'emulador.

Per a això, des d'Eclipse accedim a **Windows** > **Android SDK manager**.

Si estem emprant les utilitats proporcionades per l'SDK sense fer servir Eclipse, simplement executem l'ordre **android**, accessible dins de la carpeta **tools**.

En qualsevol dels dos casos, se'ns presentarà un diàleg similar al següent, on podrem seleccionar els paquets que s'han d'instal·lar.



Podem instal·lar les opcions més habituals. Normalment no són les últimes, sinó una o dues versions anteriors, ja que des del desenvolupament de la versió fins que surten al mercat (o s'actualitzen) diversos dispositius amb aquesta versió sol passar un cert temps.

Amb aquest pas, ja tenim les eines i l'entorn preparats per a crear projectes i publicar-los des d'Eclipse.

4.5.3. Verificació

Entre les eines que ens proporciona Google, hi ha un emulador, amb moltes opcions per a diferents dispositius i configuracions. Malgrat totes aquestes facilitats, és extremament lent.

Anècdota

Com a referència purament anecdòtica, hem vist, en un sistema amb 8 Gb de RAM, un emulador amb una imatge d'un *smartphone* senzill (ni tan sols *tablet*) amb un temps d'arrencada d'uns cinc minuts.

Recomanem provar les aplicacions en dispositius reals. Idealment, les hem de verificar en tots els telèfons on es pugui instal·lar. A causa de la dificultat d'aquest procés si més no ho podem fer en els models principals.

Per a instal·lar l'aplicació en un dispositiu, hem de seguir els passos següents:

1) Definim l'aplicació com **debuggable** en l'arxiu **manifest**. L'arxiu es diu **AndroidManifest.xml**. Hem d'afegir-hi l'atribut **android:debuggable="true"** al *tag* `<application>`.

2) En el dispositiu on vulguem provar l'aplicació: a **Settings** > **Applications** > **Development** cal marcar el *check* **USB debugging**.

3) Seleccionar el dispositiu des d'Eclipse en el diàleg que saltarà en connectar el dispositiu.

Si no usem Eclipse, hem de fer servir l'ordre ADB per a connectar-nos al dispositiu.

En qualsevol cas, si es vol provar l'opció de l'emulador, podem trobar guies que ens indiquen els passos necessaris i les opcions disponibles.

Web recomanat

http://developer.android.com/guide/developing/tools/adb.html

El comandament **adb devices** mostrarà una llista dels dispositius connectats.

4.6. Visió general

El flux d'una aplicació Android:

1) Es desenvolupa l'aplicació.

2) Es verifica en l'emulador.

3) Es verifica en un dispositiu real.

4) Es publica l'aplicació en l'Android Market.

Webs recomanats

Podeu trobar unes guies altament fiables amb informació de primera mà que proporciona Google en les adreces d'Internet següents:

http://developer.android.com/guide/developing/devices/emulator.html

http://developer.android.com/guide/developing/tools/emulator.html

4.7. Especificacions

Què vol el client?

El client vol que el seu contingut sigui accessible al màxim nombre possible d'usuaris. Ja té una pàgina web amb els productes que ofereix i informació. Vol realitzar una aplicació per a dispositius mòbils.

De manera ideal, el desenvolupament d'aquesta aplicació mòbil no hauria de comportar canvis estructurals a la pàgina web, ni un canvi organitzatiu gaire gran quant a la generació del contingut de la pàgina.

Què se li proposa?

Es planteja al client la possibilitat de realitzar una aplicació per a iPhone o Android. S'opta per començar per Android en una primera fase i, més endavant, desenvolupar l'aplicació per a iPhone si la d'Android té l'èxit esperat.

L'aplicació serà de descàrrega gratuïta, ja que té com a objectiu donar visibilitat a la pàgina web ja existent i oferir accés al contingut que presenta d'una manera més convenient.

4.8. Disseny

4.8.1. Esbós general

Podem crear els components per codi o mitjançant xml. També existeixen moltes solucions útils que ens ajuden a abstreure aquests passos i ens permeten crear la UI[4].

(4)Acrònim de *user interface*.

Una d'aquestes utilitats és *droid draw*. Aquesta aplicació, disponible per a diversos sistemes, ens permet generar la interfície d'un projecte Android mitjançant una **interfície *drag 'n drop*** (arrossegant i deixant anar elements gràficament).

Podem realitzar uns esbossos de la tipologia de pàgines, a manera de *wireframe* d'una manera molt simple. Quan el grau de detall augmenta, és molt aconsellable utilitzar eines com aquestes per a generar àgilment les pantalles necessàries.

4.8.2. Disseny final

El disseny d'una aplicació nativa és, en molts casos, més delicat que el d'una pàgina web. Els usuaris d'una plataforma específica, com ho és Android, esperen que els controls tinguin un cert aspecte i es comportin d'una manera determinada. Això és perquè totes les aplicacions es comporten d'una forma similar i ens és útil per a garantir la usabilitat de la nostra aplicació.

Hi ha diversos recursos que ens ajudaran a donar coherència a la nostra aplicació. En conjunt, aquestes recomanacions donaran una sensació de familiaritat i de producte ben acabat a l'aplicació:

- http://developer.android.com/guide/practices/ui_guidelines/index.html
- http://developer.android.com/design/index.html

4.9. Iteracions

4.9.1. Iteració 1: definició del flux de l'aplicació

Tenim una sèrie de continguts disponibles en una pàgina web i els volem fer accessibles per mitjà d'una aplicació nativa d'Android.

La web ja existent consta de les urls següents:

```
es.weboriginal.com
/
/arbol_de_navegación
/quienes_somos
```

```
/servicios/
   -ilustración
   -diseño_web
   -seo
/clientes/
   -casos_de_éxito
   -showcase
      -demo1
      -demo2
      -demo3
      -demo4
/contacto/
   -donde_estamos
   -formulario_de_contacto

en.weboriginal.com
/en
/sitemap
/about_us
/services/
   -illustration
   -web_design
   -seo
/clients/
   -success_stories
   -showcase
      -demo1
      -demo2
      -demo3
      -demo4
/contact/
   -where_are_we
   -contact_form
```

Com podem observar, hi ha contingut en diversos idiomes i segueix un arbre de navegació habitual, amb pàgina de primer nivell i d'altres agrupades en seccions mitjançant pàgines de navegació. El contingut està ben organitzat mitjançant *pretty urls*.

Pretty urls

Les *pretty urls* o *search engine friendly urls* són urls ben dissenyades i estructurades d'una manera correcta.

http://en.wikipedia.org/wiki/rewrite_engine

Podem distingir també les urls segons l'idioma al qual pertanyen. La distinció entre idiomes es fa utilitzant el subdomini de cada idioma per a tenir flexibilitat total a l'hora de definir els noms de les pàgines.

Nota

Si no fos així, no hi podria haver dues pàgines amb el mateix nom accessibles d'una manera directa per a tots dos idiomes, com en el cas de la pàgina "seo".

El lloc té a més funcionalitat de cerca i un sistema de votació del contingut. Volem mantenir també aquestes funcionalitats.

Definirem un disseny que respecti aquesta estructura amb components bàsics.

4.9.2. Iteració 2: disseny - esbós

Amb la informació que ha de ser accessible i clara en la iteració anterior, podem realitzar els primers esbossos del disseny.

Podem utilitzar qualsevol eina de *wireframes*. En aquest cas, farem servir esbossos tradicionals (amb llapis o digitalment), ja que només ens interessa tenir clara la distribució bàsica dels elements en l'aplicació. Més endavant emprarem eines que ens ajudaran a generar la interfície gràfica d'una manera intuïtiva.

Utilitzarem una navegació per *tabs* en la part superior per a implementar el canvi entre idiomes. D'aquesta manera, l'usuari veurà fàcilment els idiomes disponibles i podrà passar de l'un a l'altre en qualsevol moment.

En la part superior, o bé de manera oculta depenent de l'estat de l'aplicació, utilitzarem un *widget* de recerca per a permetre la funcionalitat de buscar textualment en el contingut del lloc.

Per a fer una llista de les pàgines disponibles usarem una *listview*.

En l'enllaç podem veure un exemple, si bé està fet directament des d'Eclipse sense utilitzar cap eina d'ajuda, de com s'implementa una vista de tipus llista:

http://developer.android.com/resources/tutorials/views/hello-listview.html

Si seleccionem una de les pàgines de la llista succeirà el següent:

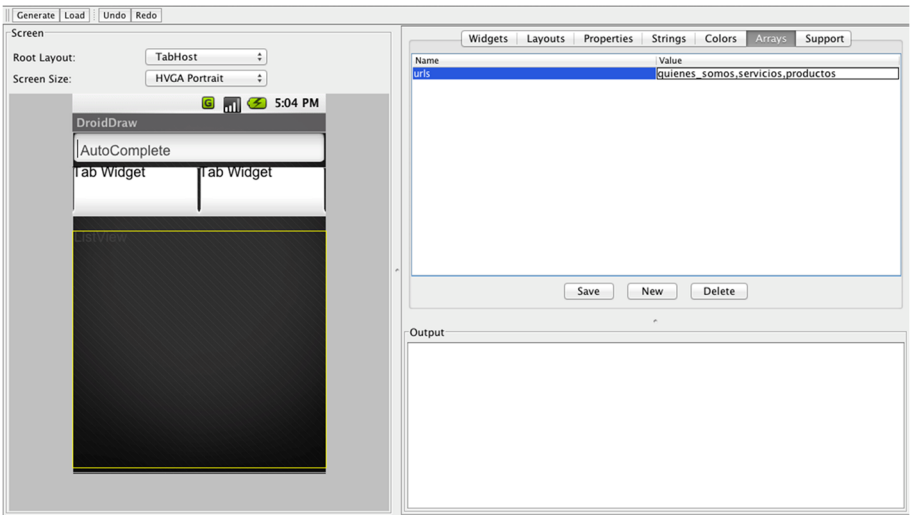
1) La pàgina és una pàgina amb contingut. Veiem un botó per a tornar a la llista anterior. Sota aquest botó se'ns mostra el contingut i, al final del contingut, un sistema d'estrelles per a votar-ne la qualitat.

2) La pàgina és una pàgina distribuïdora. Apareix una altra llista que ens mostra el contingut d'aquesta pàgina i un botó per a tornar al pas anterior.

4.9.3. Iteració 3: disseny - disseny natiu general

Amb l'estructura definida en la iteració anterior, ens disposem a realitzar el primer disseny amb elements natius. Utilitzem l'aplicació Droiddraw.

Podem veure a continuació la disposició d'elements de la pàgina principal:



Per a distribuir el contingut de la manera que més ens convingui, utilitzem *layouts*. Podem disposar els *layouts* de diverses maneres per a aconseguir diferents distribucions del contingut. Aquí, per exemple, ferm servir *layouts* horitzontals per a disposar els *tabs* i els *layouts* verticals per a situar, entre si, el quadre de cerca, el *layout* on apareixen els *tabs* i la llista d'urls.

Lectures complementàries

Podeu trobar alguns programes d'aprenentatge que expliquen amb exemples senzills els *layouts* en la pàgina d'Internet següent:

http://www.programacion.com/articulo/introducion_a_els_layouts_per_android_400

http://mobiforge.com/designing/story/understanding-user-interface-android-part-1-layouts

Per a més informació, tenim la documentació oficial, amb llistes de totes les opcions disponibles:

http://developer.android.com/guide/topics/ui/layout-objects.html

http://developer.android.com/guide/topics/resources/layout-resource.html

4.9.4. Iteració 4: disseny - disseny natiu de la pàgina distribuïdora

La pàgina distribuïdora serà pràcticament igual que la pàgina principal. L'única diferència és que, si som en una pàgina distribuïdora, hi hem arribat per mitjà d'una altra pantalla. Per tant, necessitem una manera de tornar a la pantalla anterior, a l'opció superior.

Per a això utilitzarem un simple botó de tornar.

La funcionalitat de cerca està oculta en la resta de l'aplicació. Podem implementar-ho de manera que aparegui a partir del menú de l'aplicació.

D'aquesta manera, al marge dels *tabs* de selecció d'idioma, no tenim elements permanentment a la part superior de la pantalla. D'altra banda, la majoria d'aplicacions disposen d'aquesta funcionalitat accessible des del menú d'Android una vegada som dins de l'aplicació, per la qual cosa satisfà el principi POLS[5].

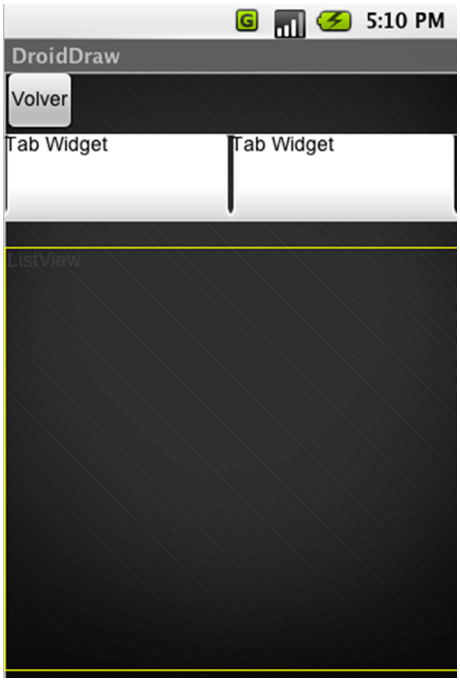
(5)Acrònim de *principle of least surprise*.

POLS

Consisteix que les coses funcionin de la manera que l'usuari espera, és a dir, de forma semblant a com ho fa la resta de funcions del sistema.

Podeu trobar-ne més informació en l'adreça d'Internet següent:

http://en.wikipedia.org/wiki/principle_of_least_astonishment

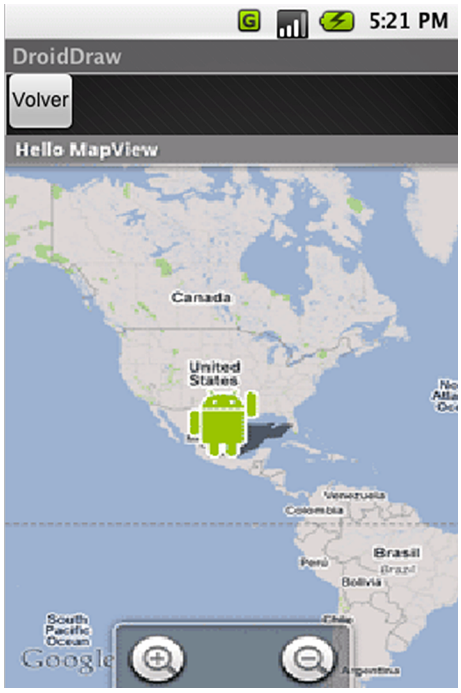


4.9.5. Iteració 5: disseny - disseny natiu contacte

En les iteracions anteriors, hem realitzat el disseny de les pàgines de contingut i les de distribució. En aquesta dissenyem la pàgina de contacte.

Per a això utilitzem un *widget* molt senzill, que dotarà de més funcionalitat la nostra aplicació. Amb el *widget* de Google Maps tenim tota la funcionalitat d'aquesta aplicació per a més facilitat de l'usuari.

No ens oblidem tampoc del botó de tornar, com en la resta de pàgines finals (no distribuïdores).



4.9.6. Iteració 6: contingut - contingut estàtic

En darrera instància volem que la informació del lloc d'on llegeix l'aplicació provingui directament de la pàgina web original.

De moment, introduïm informació estàtica en aquesta iteració. Temporalment, copiem la informació de la pàgina web en el mateix disseny.

Per a introduir aquesta informació podem utilitzar la facilitat que ens dóna Droiddraw per a crear *strings* o *arrays*.

> ***Strings* i *arrays***
>
> Els *strings* es corresponen amb cadenes de text; els *arrays* són col·leccions d'elements. En el primer tipus de dada podem guardar informació com el nom d'una secció o la descripció d'un element. En el segon tipus de dada podem, per exemple, guardar les opcions disponibles en un apartat o els idiomes.

Podem relacionar aquest tipus de dades amb els diversos components perquè mostrin el contingut de manera automàtica.

D'aquesta manera, la informació estarà disponible sense necessitat de connectar-se a Internet i podrem centrar-nos en el fet que aquesta informació es representi bé abans de canviar-ne l'origen.

Lectura complementària

Per a una guia dels passos que cal seguir, existeix el programa d'aprenentatge següent:

http://www.droiddraw.org/tutorial3.html

4.9.7. Iteració 7: crear la nostra aplicació en Android

Amb la interfície creada en la iteració anterior, podem generar ara l'aplicació en el dispositiu. El programa ens generarà el *layout*, que podem importar en el nostre projecte d'Android.

Webs recomanats

Si tenim dificultat per a exportar el disseny realitzat en Droiddraw a una aplicació, podem consultar les referències següents:

Exportar des de Droiddraw:

http://www.droiddraw.org/tutorial1.html

Crear una aplicació mínima amb un *layout*:

http://developer.android.com/resources/tutorials/hello-world.html

Un altre exemple d'aplicació mínima amb *layout*:

http://apcmag.com/building-a-simple-android-app.htm

Referències per al concepte d'*activities*:

http://developer.android.com/guide/topics/fundamentals/activities.html

4.9.8. Iteració 8: contingut - creació de *web services*

Fins ara hem estat generant el contingut de manera estàtica. La idea inicial ha estat que el contingut de l'aplicació sigui el mateix que el de la pàgina web original. Per a això, caldrà poder accedir a aquesta informació remotament, des de la nostra aplicació mòbil.

La solució habitual en aquests casos és l'ús de ***web services***. Els *web services* permeten la interacció entre diferents sistemes i faciliten l'intercanvi d'informació d'una manera controlada entre serveis molt dispars. Existeixen molts tipus de *web services*.

A continuació us fem la llista d'uns quants:

- REST
- SOAP
- WSDL[6]
- JSON-RPC
- WPS

(6)Acrònim de *web services description language.*

Els *web services* es poden desenvolupar en qualsevol llenguatge de tipus servidor. Perquè l'aplicació Android o altres serveis puguin accedir a la informació de la pàgina web caldrà crear una capa en l'aplicació web que s'encarregui de retornar només la informació sol·licitada.

Una solució habitual consisteix a oferir una API[7] per mitjà de la qual es pugui cridar els diferents recursos. S'haurà de crear aquest accés perquè la nostra aplicació connecti amb la informació de la pàgina web.

(7)Acrònim d'*application programming interface.*

API

Capa d'abstracció d'un programari utilitzada com a biblioteca per un altre programari.

Podeu trobar-ne més informació en la pàgina d'Internet següent:

http://es.wikipedia.org/wiki/interfaz_de_programaci%C3%B3n_de_aplicaciones

Trobem alguns exemples d'API en serveis d'empreses conegudes de negoci en Internet, com ara l'API de *Yahoo* i la del servei de *Flickr*.

D'aquesta manera, quan la informació de la pàgina web s'actualitzi, l'accés des de l'aplicació mòbil també tindrà accés a la mateixa informació.

Si això resultés massa enutjós sempre hi ha la possibilitat de distribuir el contingut en la mateixa aplicació, i que s'actualitzi únicament de manera estàtica. Així caldria actualitzar l'aplicació per a distribuir el nou contingut als usuaris.

4.9.9. Iteració 9: contingut - connexió a *web services*

Una vegada desenvolupat l'accés al *web service* en l'aplicació original podem connectar-nos a aquest *web service* des de la nostra aplicació.

Els recursos d'Android utilitzen memòria cau quan és possible, depenent dels valors en l'arxiu **AndroidManifest.xml**.

Hi ha diverses aproximacions per a accedir a un *web service* des d'Android, a causa dels molts tipus de *web services* que hi ha disponibles. En triarem una segons el tipus implementat per a accedir al contingut de la web original.

Webs recomanats

Alguns recursos que expliquen amb prou detall i d'una manera senzilla aquest procés són els següents:

http://stackoverflow.com/questions/1048310/how-to-call-a-net-web-service-from-android

http://programa-con-google.blogspot.com.es/2010/12/android-como-consumir-un-servicio-web.html

http://sarangasl.blogspot.com.es/2011/10/android-web-service-access-tutorial.html

4.9.10. Iteració 10: Disseny - disseny final

En aquesta iteració perfilem qualsevol problema que ens hàgim trobat amb el disseny en iteracions anteriors.

En molts casos, els *layouts* no es poden realitzar correctament en el primer intent. Atès que es poden imbricar de maneres complexes és bo replantejar-ne l'organització i subdivisió després d'un ús continuat de l'aplicació per a implementar possibles millores. La verificació en diversos dispositius també pot treure a la llum certs inconvenients d'alguns tipus de *layouts* respecte d'uns altres.

4.9.11. Iteració 11: publicació en l'Android Market

Publiquem la nostra aplicació en l'Android Market. Això implica pagar una única quantitat inicial[8].

(8)Actualment, aquesta quantitat és de 25 dòlars.

Per a pagar aquesta quantitat, cal crear un compte a ***Google checkout***, el sistema de pagament de Google. Aquest pagament només s'ha de dur a terme una vegada (no és recurrent).

Quan pugem l'aplicació a https://play.google.com/store, se'ns demanaran les icones i captures de l'aplicació, descripcions per als diferents idiomes, el preu, els rangs de recomanació per edats, etc. En resum, totes les dades que apareixeran més endavant en la botiga de Google. També haurem d'acceptar les condicions de Google i facilitar una web i dades de contacte.

Una vegada publicada, l'aplicació apareixerà en les categories corresponents del Market i els usuaris podran descarregar-la, fer valoracions i enviar-nos *feedback* sobre el seu ús.

4.9.12. Iteració 12: llançament i posada en societat

Al marge de la publicació en la botiga de Google, és interessant plantejar-se de realitzar algun tipus de promoció. Podem utilitzar les xarxes socials o campanyes de diferents tipus per a incentivar l'ús de la nostra aplicació. Un bàner a la pàgina web original pot ser suficient per a informar els nostres usuaris habituals de la disponibilitat d'aquesta nova manera d'accedir a la mateixa informació.

Cal tenir en compte que en l'Android Market hi ha una gran quantitat d'aplicacions i, tret que l'usuari busqui activament la nostra aplicació, és difícil que la trobi de manera accidental.

4.9.13. Iteració *n*: ...?

Seguint aquest model de desenvolupament, podem incorporar qualsevol canvi necessari a l'aplicació. Podem revisar el *layout* en qualsevol moment o afegir funcionalitat extra accedint al contingut necessari per mitjà del *web service*. Així, el lloc web original i l'aplicació Android poden compartir part del desenvolupament i evolucionar de manera conjunta. Per a funcionalitats més avançades, sempre tenim com a referència la documentació oficial d'Android, amb les recomanacions i bones pràctiques per als nostres projectes Android.

Recomanem descarregar exemples amb les funcionalitats desitjades per a investigar amb profunditat com estan desenvolupats.

Totes les marques són propietat dels amos respectius. L'autor no està afiliat amb Google, amb Android ni amb cap de les marques que s'esmenten en aquest document.

5. Desenvolupament d'una web amb continguts administrables en Drupal

5.1. Introducció

El projecte consistirà a desenvolupar un portal web fictici amb les característiques següents:

- Contingut administrable.
- *Back office* d'ús senzill per a administrar els continguts.
- Multiidioma.
- Amb menús i navegació de diversos nivells.
- Amb elements i notícies de diferents tipus.
- Llista de notícies i detall d'aquestes notícies.
- Amb funcionalitat de cerca.

5.2. Per què Drupal?

Drupal és un gestor de continguts o CMS[9] molt utilitzat. Pot ajudar-nos a accelerar el desenvolupament d'un lloc web perquè una gran part de la funcionalitat ja està escrita i verificada per la comunitat.

(9)Acrònim de *content management system*.

Codi aprofitat

"El millor codi és el que no s'ha d'escriure". Utilitzant programari d'uns altres podem retallar considerablement el temps de desenvolupament.

A continuació presentem una senzilla taula comparativa amb alguns dels CMS més habituals, segons la nostra experiència:

Taula comparativa de les principals CMS

Nom	Plataforma	Bases de dades	Llicència	Última data
Drupal	PHP	MySQL, Oracle, PostgreSQL, SQLite, Microsoft SQL Server	GPL	01/02/12
Joomla!	PHP	MySQL	GPL	15/03/12
Wordpress	PHP	MySQL	GPL	03/01/12
TYPO3	PHP	MySQL, Oracle, PostgreSQL	GPL	24/01/12
Django-cms	Python/Django	PostgreSQL, MySQL, SQLite 3, Oracle	BSD	27/01/12
MoinMoin	Python	Arxiu pla	BSD	21/02/12
Prestashop	PHP	MySQL	GPL	08/02/12

Webs recomanats

Podeu veure altres taules comparatives en les adreces d'Internet següents:

http://socialcompare.com/en/comparison/popular-content-management-system-cms-comparison-table

http://en.wikipedia.org/wiki/list_of_content_management_systems

5.2.1. Avantatges

Els avantatges que presenta Drupal són els següents:

1) Tota la part de gestió de continguts està desenvolupada des del començament.

2) El codi està revisat per un nombre més gran de gent perquè es tracta de codi lliure.

3) Hi ha un gran nombre de mòduls per a tot tipus de tasques i solen ser fàcils d'utilitzar.

5.2.2. Inconvenients

Els inconvenients que presenta Drupal són els següents:

1) Els estàndards de qualitat de Drupal no són tan alts com els d'altres projectes *open source*. De vegades, la implementació d'alguns mòduls[10] deixa bastant a desitjar.

(10)No n'esmentarem cap en concret.

2) Algunes funcionalitats molt específiques poden ser més difícils d'implementar que en un desenvolupament a mida.

5.3. Consideracions generals de desenvolupament

Drupal, com molts altres sistemes, consta d'una part d'arxius (codi i recursos gràfics) i una base de dades de seguretat. Durant el desenvolupament, totes dues parts es van transformant. Hi ha dues maneres de gestionar aquests canvis especialment útils per a la gestió d'un projecte:

1) Control de versions.

2) *Deployment*.

5.3.1. Control de versions

L'ús de **sistemes de control de versions** durant el desenvolupament permet:

- Tornar a versions anteriors d'arxius.
- Gestionar els canvis d'un gran nombre de desenvolupadors d'una manera àgil.
- Fer-ne ús com a documentació del mateix projecte.

Control de versions

Alguns exemples de sistemes de control de versions són Git, Mercurial o Subversion.

5.3.2. *Deployment*

El ***deployment*** consisteix a automatitzar la pujada a producció. Això ens ajuda a:

- Evitar errors manuals quan són més crítics (pujada a producció d'un nou desenvolupament).
- Poder desenvolupar amb iteracions ràpides.
- Solucionar imprevistos d'una manera ràpida.
- Fer-ne ús com a documentació posterior dels paràmetres de configuració.

Normalment existeixen entorns de desenvolupament, de verificació i de producció:

- **Desenvolupament**: es realitzen els canvis.
- **Verificació**: es comprova que tot funciona correctament. És recomanable provar l'aplicació en un test amb dades ja presents en la producció, per tal de verificar el *site* amb dades reals.
- **Producció**: és on hi ha l'aplicació real. Aquesta web és l'única amb la qual interaccionaran els usuaris finals de l'aplicació.

Aquests tres entorns han de ser tan semblants com sigui possible l'un amb l'altre (especialment, la verificació i la producció).

El sistema de *deployment* acostuma a ser molt personalitzat, per la qual cosa podeu utilitzar *scripts* que automatitzin les ordres bàsiques (copiar a una carpeta, pujar a una adreça FTP, copiar la base de dades, etc.).

Per a casos més complexos (diversos servidors o bases de dades o entorns replicats, per exemple) existeixen sistemes de *deploy* més complexos.

Webs complementaris

Són *deploy* més complexos Xef i Puppet.

5.3.3. Seguretat

La seguretat és molt important en qualsevol tipus de desenvolupament web.

No és extremament estrany que una persona o procés automatitzat aconsegueixi introduir-se en un servidor aprofitant-se d'alguna vulnerabilitat.

1) Bàsics

El programari té *bugs* (errors en un programa informàtic).

Es tracta de minimitzar aquests *bugs* seguint unes pràctiques adequades. Programes que són correctes avui poden ser insegurs simplement pel pas del temps. Es descobreixen vulnerabilitats que no se solucionen i poden ser aprofitades per un atacant.

Lectura complementària

Per conèixer l'origen del terme *bug*, podeu consultar l'adreça d'Internet següent:

http://en.wikipedia.org/wiki/software_bug#Etymology

2) Seguretat i SEO[11]

(11)Acrònim de *search engine optimization*.

Cal recordar que una de les conseqüències, encara que potser de les menys importants, d'una intrusió de seguretat és que els principals cercadors faran perdre, a la nostra pàgina, molts llocs en els resultats. Fins i tot poden arribar a eliminar-la dels resultats. Aquest procés pot alertar els usuaris de la pàgina, que poden perdre la confiança en el lloc web.

Mantenir la confiança

Tant la confiança dels usuaris com la dels motors de cerca és molt fàcil de perdre i extremament difícil de recuperar, per la qual cosa convé invertir en seguretat abans que es produeixi una intrusió.

3) Mesures de seguretat

Convé seguir les bones pràctiques del desenvolupament en el llenguatge i *framework* que utilitzem.

Webs recomanats

Algunes bones referències sobre mesures de seguretat a l'hora de desenvolupar una aplicació són les següents:

https://www.owasp.org/index.php/main_Page

http://stackoverflow.com/questions/2621706/good-guide-for-web-app-security

http://stackoverflow.com/questions/6166833/hacking-how-do-i-find-security-holes-in-my-own-web-application-did-i-do-a-good

http://stackoverflow.com/questions/3891349/web-applications-security

http://stackoverflow.com/questions/47323/top-tips-for-secure-web-applications

Per recuperar-nos més de pressa en cas d'una intrusió, ens poden ajudar els consells següents:

- Mantenir *back-ups* regulars i provar el procés de recuperació d'aquests *back-ups*.

- Automatitzar el *deploy* i la recuperació de *back-ups*. Pràcticament tots els serveis de *hosting* tenen un servei de *back-ups* que, per pocs diners més, ens proporcionarà la restauració de *back-ups* d'una data determinada. Convé tenir en compte això abans de triar un *hosting*.

- Actualitzar regularment la versió dels CMS, *frameworks* i biblioteques que utilitzem.

Molts atacants volen millorar l'eficiència dels atacs. Per això trien productes de programari que utilitzi molta gent i hi busquen vulnerabilitats. D'aquesta manera, una mateixa vulnerabilitat és usada per a introduir-se en molts servidors.

Convé fer servir l'última versió estable i actualitzar-la constantment. La versió més recent sol estar en desenvolupament i presenta problemes de seguretat que encara no s'han trobat.

La versió massa antiga té problemes de seguretat que ningú no soluciona, ja que és obsoleta.

El compromís entre totes dues sol ser la solució òptima per a la majoria de casos d'ús.

5.4. Estructura d'un projecte Drupal

Drupal utilitza una estructura de carpetes convencional que organitza les diverses parts de l'aplicació. Podem veure els dos primers nivells de directoris en el gràfic següent:

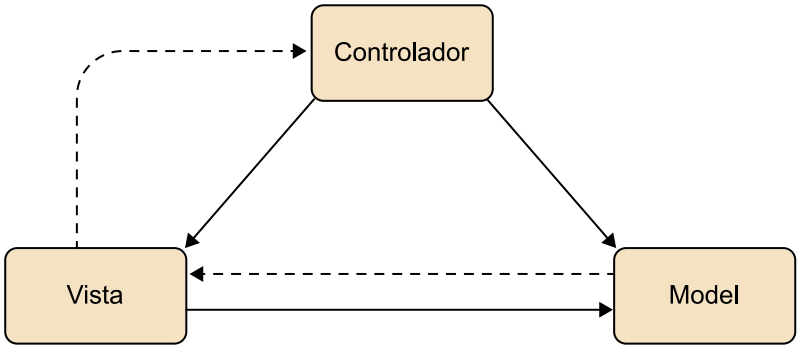
```
drupal-7.12/
├── includes
│   ├── database
│   └── filetransfer
├── misc
│   ├── farbtastic
│   └── ui
├── modules
│   ├── aggregator
│   ├── block
│   ├── blog
│   ├── book
│   ├── color
│   ├── comment
│   ├── contact
│   ├── contextual
│   ├── dashboard
│   ├── dblog
│   ├── field
│   ├── field_ui
│   ├── file
│   ├── filter
│   ├── forum
│   ├── help
│   ├── image
│   ├── locale
│   ├── menu
│   ├── node
│   ├── openid
│   ├── overlay
│   ├── path
│   ├── php
│   ├── poll
│   ├── profile
│   ├── rdf
│   ├── search
│   ├── shortcut
│   ├── simpletest
│   ├── statistics
│   ├── syslog
│   ├── system
│   ├── taxonomy
│   ├── toolbar
│   ├── tracker
│   ├── translation
│   ├── trigger
│   ├── update
│   └── user
├── profiles
│   ├── minimal
│   ├── standard
│   └── testing
├── scripts
├── sites
│   ├── all
│   └── default
└── themes
    ├── bartik
    ├── engines
    ├── garland
    ├── engines
    └── garland
```

Podem veure com guarda cada mòdul base en la carpeta **modules**.

A **sites/all** hi ha les carpetes **modules/** i **themes/**, on se situen els mòduls i temes personalitzats.

5.5. Model vista controlador i Drupal

El **model vista controlador** és un patró d'arquitectura de programari útil per a mantenir diferents seccions de l'aplicació separades i desenvolupar-se d'una manera més clara. Fomenta la **separació de responsabilitats**, un principi del desenvolupament de programari.



Font: http://craftyman.net/mvc-en-php/

Drupal no fa servir exactament aquest model, però pot ser útil veure quines abstraccions usa Drupal i quins equivalents tindrien en el model vista controlador per a entendre millor la funció que té.

1) Model. El model especifica les regles de negoci i l'accés directe a les dades. És el "cervell" de l'aplicació.

2) Vista. Drupal té diversos sistemes equivalents a la funcionalitat de vistes. D'una banda té els temes (*themes*), que serien conjunts de vistes amb una coherència visual.

Cadascuna d'aquestes vistes és estilable per codi de manera independent. Aquests arxius segueixen un sistema de convenció per saber de quin element seran la vista segons el nom que té l'arxiu. Cadascun d'aquests arxius seria el més semblat a una vista clàssica. Són *templates* que criden les funcionalitats necessàries.

Finalment tenim els **nodes**: elements individuals que podem estilar com vulguem.

3) Controlador. No existeix un controlador com a tal, encara que podríem entendre que la definició per mitjà dels panells d'administració actua realment de controlador: relaciona una certa pàgina amb unes vistes determinades.

5.6. Visió general

5.6.1. Instal·lació de Drupal a *server*

Per instal·lar simplement anem al lloc web de Drupal i descarreguem l'última versió estable disponible.

Descomprimim el resultat i el situem en un directori que pugui servir un servidor web (per exemple, Apache) i visitem la pàgina on puguem veure aquest directori. Normalment serà **localhost:número_de_puerto** si l'estem instal·lant en la màquina utilitzada per al desenvolupament. A partir d'aquí, podem seguir els passos per a configurar la base de dades, l'usuari i la contrasenya per defecte i altres característiques bàsiques del projecte.

Haurem de crear una base de dades amb les mateixes credencials especificades en aquests passos perquè l'aplicació pugui funcionar correctament.

Bases de dades i servidor

Podem utilitzar una gran varietat de programes per a tenir una base de dades i servidor en local i per a gestionar-la. Per a fer-la funcionar alguns dels programes més senzills són *WAMP* per a Windows i *MAMP* per a Mac OS. Per a sistemes UNIX/Linux, el més habitual seria instal·lar el servidor (per exemple, Apache) i la base de dades (per exemple, MySQL) des del sistema de gestió de paquets de la distribució (YUM, apt-get, pacman, ports).

Des de la versió 5.4 PHP incorpora també el seu propi servidor de verificació.

Per connectar-se a la base de dades recomanem *MySQL Workbench*, d'Oracle. Està disponible per a diversos sistemes.

Després d'aquest procés, ja tenim una aplicació funcional i podem començar el desenvolupament. Podem veure l'aplicació a **localhost:número_de_puerto** amb el text "welcome to localhost".

5.6.2. El panell d'administració

Totes les tasques d'administració de Drupal es desenvolupen per mitjà del panell d'administració de la mateixa web, accessible en la secció privada del *site*. Excepte en casos en què caldrà l'edició del codi o la manipulació d'arxius, la major part de canvis es realitzen per mitjà d'aquesta interfície web.

5.6.3. Funcionalitats

Ens interessa desenvolupar l'aplicació d'una manera àgil. Per tant, intentarem desenvolupar-la incrementalment i en petites iteracions. Per a això ens centrarem en les funcionalitats que volem obtenir.

En Drupal, les diverses funcionalitats bàsiques d'un *site* són accessibles per mitjà de mòduls. Instal·larem mòduls a mesura que necessitem les funcionalitats corresponents.

5.7. Especificacions

Què vol el client?

El client vol realitzar un lloc web de continguts administrables. No té un departament propi de desenvolupament web i vol que els seus empleats siguin capaços d'administrar els continguts necessaris d'una manera fàcil i intuïtiva. Té una idea clara sobre la navegació i els tipus de contingut que espera trobar. També demana consell sobre altres continguts que no sap exactament com ha de distribuir en el portal.

Què se li proposa?

El client no demana unes funcionalitats gaire avançades des del punt de vista tecnològic. Bàsicament el lloc web, si bé té un disseny i unes funcionalitats que no són trivials, no va al límit del que es pot fer amb la tecnologia i utilitza moltes funcionalitats presents en una gran quantitat d'aplicacions web.

A causa d'això i del cost descartem realitzar una aplicació a mida. En aquest cas hauríem de desenvolupar també la part de gestió de l'aplicació. El client no requereix una administració a mida, i està disposat a trobar algunes limitacions en aquest aspecte, a canvi d'agilitar el procés de desenvolupament.

5.8. Iteracions

5.8.1. Iteració 1: la primera pàgina

Si entrem en l'adreça del *server*[12] podem veure una pàgina per defecte amb el contingut corresponent. A continuació crearem una pàgina pròpia amb un nom que escollirem nosaltres.

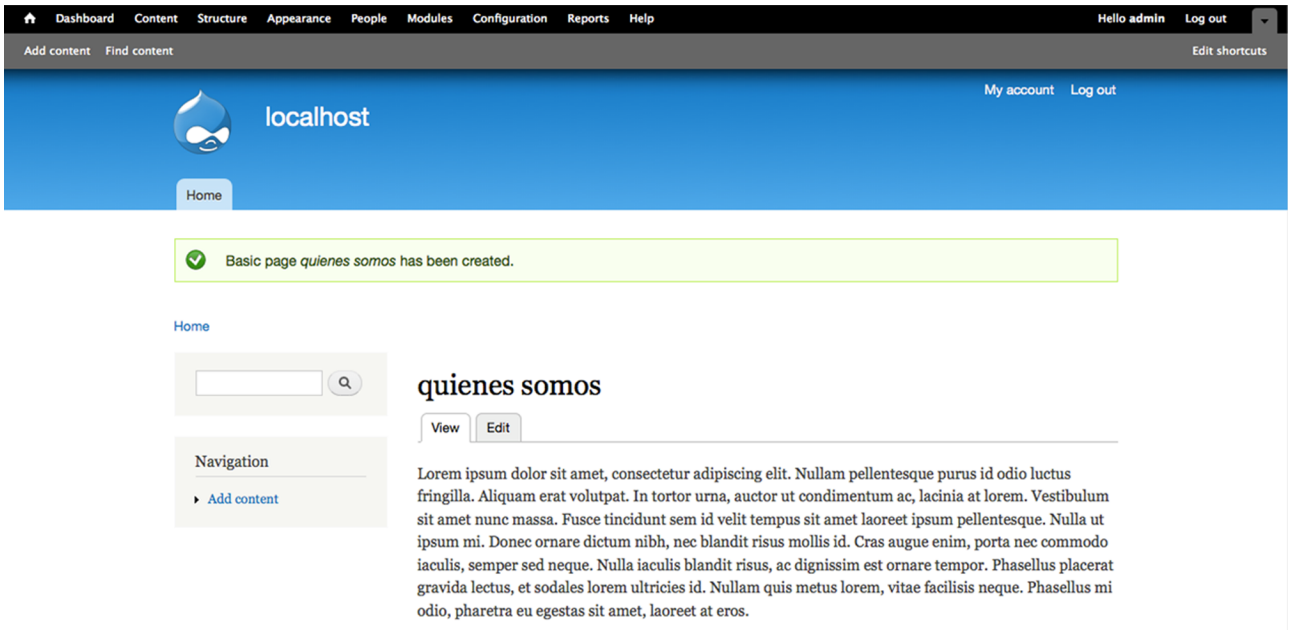
(12) D'ara endavant, *localhost*.

A **Content**, **add content** creem una nova pàgina de tipus **basic page**. Li posem un títol i un text fictici.

Text fictici

Per al text, se sol utilitzar el conegut *Lorem ipsum*. N'hi ha diversos generadors, un dels quals és: http://www.lipsum.com.

Ens indicarà que la pàgina s'ha creat correctament.



Podem veure, a més de la pàgina amb el contingut introduït, les diferents seccions de la pàgina. Aquestes seccions seran les mateixes per a totes les pàgines. Veiem el menú d'administració a la part superior i, a sota, la pàgina que veuran els visitants sense registrar de la pàgina. A més d'un menú de navegació, que de moment només mostra un element, veiem unes opcions laterals i el contingut en si mateix de la pàgina.

5.8.2. Iteració 2: tipus de contingut - notícies

Fins ara tenim un lloc bàsic en funcionament. També hem creat una pàgina de contingut senzill. Per a la creació de pàgines més complexes necessitarem crear diferents tipus de contingut.

A continuació definim els tipus de contingut per a la pàgina que mostrarà les notícies.

La *home* consistirà en una sèrie de notícies amb imatge petita, imatge gran, títol i un contingut textual de la notícia.

Per a això necessitarem instal·lar diversos mòduls:

- El mòdul *Views* ens permetrà crear llistes d'elements amb un criteri i, en general, organitzar el contingut. Al seu torn, aquest mòdul requereix el mòdul *Ctools* per a poder funcionar.

- També necessitarem el mòdul *Entityreference* per a enllaçar diferents tipus de continguts en estructures més complexes. Aquest mòdul, al seu torn, depèn del mòdul *Entity*.

En general veurem que les pàgines dels mòduls tenen una descripció dels mòduls dels quals depenen, les versions de Drupal amb les quals funcionen correctament, referències i altres recomanacions.

Per instal·lar mòduls de Drupal descarreguem els corresponents a la nostra versió de Drupal i els situem en la carpeta **modules** de l'estructura d'arxius del projecte.

Amb els arxius en el lloc indicat, activem els mòduls des del panell d'administració: en l'opció del menú **Mòduls** activem els mòduls i les dependències corresponents.

Existeixen diverses aproximacions per a treballar amb diferents tipus de contingut. Una opció interessant i molt flexible és crear tipus de contingut bàsic (imatge, notícia, bàner, etc.) i enllaçar-los entre si en tipus més complexos que representin una pàgina o secció complexa. Per exemple, un tipus "pàgina notícia" pot estar format per *n* notícies i dos bàners. Aquesta aproximació és la que usem aquí.

Creem una nova definició de tipus de contingut des de **Structure** > **Content types** > **add content type**. Ho anomenem *notícia*. Hi ha un gran nombre d'opcions extra disponibles en aquest punt. A continuació definim els camps que tindrà aquest tipus de contingut:

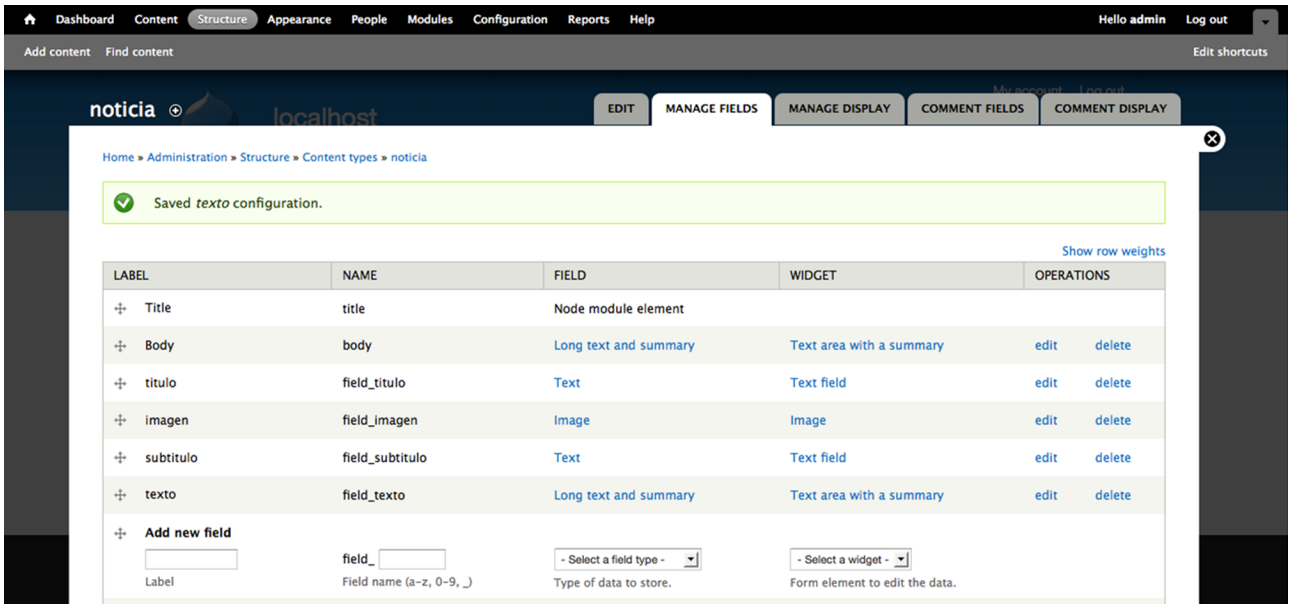
1) Títol: el títol de la notícia. És de tipus text.

2) Imatge: la fotografia que pot incloure la notícia. Podem posar-hi imatges per defecte, dimensions mínimes i màximes i un gran nombre d'opcions. És de tipus imatge.

3) Subtítol: el subtítol (si en té) de la notícia. És de tipus text.

4) Text: el contingut de la notícia. Triem l'opció **Long text with summary**, perquè ens generi el resum de la notícia.

Podem veure una imatge de la pantalla de definició de camps del tipus de contingut notícia:



En aquest punt, tenim definit el tipus de contingut notícia. Necessitem un tipus de contingut pàgina notícia, que contindrà, mitjançant referències del mòdul de referència d'entitats, continguts de tipus notícies. També podria incloure altres tipus de continguts: bàners, enquestes o qualsevol altre tipus que definim.

Al final d'aquesta iteració acabem amb la definició completa de tipus referents a la pàgina que mostra les notícies.

5.8.3. Iteració 3: tipus de contingut - bàners

Un tipus interessant per a un portal web és el de diferents mides de bàners. S'han de poder situar en les altres pàgines.

Tal com hem fet anteriorment, crearem un tipus de contingut específic per a cada tipus de bàner que tinguem. En aquest cas, implementem un tipus que anomenarem *bàner senzill*.

La funcionalitat d'incloure un camp amb un enllaç no existeix per defecte en Drupal. Per a utilitzar-la, necessitarem instal·lar el mòdul *Link*.

La definició d'un bàner senzill seria la següent:

1) Imatge: imatge que es mostrarà com a contingut del bàner. De tipus imatge.

2) Text: text opcional per a mostrar per sobre de la imatge. De tipus text.

3) Enllaç: Enllaç on apunta el bàner. És de tipus enllaç.

5.8.4. Iteració 4: tipus de contingut - formulari de contacte

Per a realitzar els formularis de contacte necessitem instal·lar el mòdul *Webform*.

Una vegada instal·lat i activat, l'utilitzem des de **Content** > **Add content** > **Webform**. L'anomenem *contacte*. A continuació en definim els camps, de la mateixa manera que definim els del tipus de contingut.

A més dels camps, podem definir a quina adreça de correu electrònic s'enviaran els resultats. Cada vegada que algú empleni aquest formulari s'enviarà a l'adreça de correu electrònic configurada. També quedaran els registres accessibles des de l'administrador, des de la pestanya **results**. Així podem comprovar fàcilment l'*input* dels usuaris de manera molt convenient.

5.8.5. Iteració 5: pàgines - notícies

Fins ara hem creat totes les definicions de tipus que necessitàvem per a les pàgines.

A continuació crearem pàgines amb els tipus de contingut que hem comentat. Utilitzarem les vistes en comptes de la creació de pàgines bàsiques per a tenir més flexibilitat. Les vistes permeten posar restriccions en els tipus de contingut, paginar els resultats i un gran nombre de funcionalitats molt útils.

Anem **a Estructure** > **Views** en el menú d'administració i creem una nova vista. Podem triar aquí el nom d'aquesta vista, que serà el nom de la pàgina que contingui la vista. Creem una vista de nom *pagina-noticies* i podrem veure aquesta vista en la pàgina **localhost/pagina-noticies**.

Nom de la pàgina

Si el nom de la pàgina conté caràcters prohibits, Drupal substituirà els caràcters per altres de permesos o els eliminarà de la url. Això sol ser motiu de confusió a l'hora de buscar els *templates* corresponents a aquesta vista o de buscar la pàgina que s'ha creat.

En la vista de pagina-noticies, accessible des d'**Estructure** > **Views**, filtrarem perquè només mostri el contingut de tipus pagina-noticia. D'aquesta manera, la pàgina tindrà accés a tota la informació definida amb aquest tipus, incloent-hi les dades relacionades amb aquest tipus per mitjà de referències.

Amb aquesta vista definida ja podem veure el resultat visitant la pàgina. Podem observar que la pàgina està buida.

Com és que la pàgina està buida?

En realitat, només hem creat la definició de relacions entre tipus i el que ha de mostrar la vista, però no hem creat contingut com a tal.

Amb la finalitat que hi hagi contingut real a la pàgina, l'afegirem des del panell d'administració. Aquí veurem un dels principals beneficis de Drupal.

Des de **Content** > **Add content** seleccionem el tipus de dada corresponent i anem afegint els valors per als diferents camps.

Panell d'administració

Automàticament, Drupal té aquest panell d'administració, amb restriccions per als diferents tipus i facilitats com la pujada d'imatges i el reescalat corresponent. Tot això, simplement a partir de la definició de tipus de contingut.

5.8.6. Iteració 6: pàgines - bàners

En aquesta iteració crearem el contingut dels bàners. Simplement hem d'agregar el contingut des de **Content** > **Add content**.

Per afegir els bàners a les diferents pàgines, podem incorporar-los en la definició de tipus de pàgines, com en l'anterior pagina-noticies. Qualsevol pàgina que mostri bàners tindrà en la definició de camps del seu tipus una referència al tipus bàner.

L'avantatge d'aquesta aproximació al problema és que tenim un tipus únic de dades per a cada pàgina i, al mateix temps, podem reaprofitar tipus de contingut més específics com ara notícia, bàner o imatge.

5.8.7. Iteració 7: pàgines - formulari de contacte

Crearem una pàgina per poder rebre *inputs* dels usuaris. Així podran enviar-nos les seves opinions o suggeriments. En el futur es podria ampliar la funcionalitat per a sol·licitar informació o adquirir serveis que oferim.

El tipus *web form* és un tipus més de contingut, com qualsevol altre. Podem incorporar-lo a la definició de les pàgines que l'utilitzin, fent servir una referència d'entitat. Després, des del *template* de la vista corresponent, l'estilem com ens convingui.

5.8.8. Iteració 8: disseny

En aquest punt ja tenim el contingut i les pàgines funcionals. Hem optat per desenvolupar una web funcional sobre la qual després aplicarem el disseny. Ha arribat el moment de definir el disseny d'aquestes pàgines, tenint en compte la manera com les pinta Drupal, per a aprofitar al màxim les funcionalitats que ens ofereix sense refer-les completament.

En alguns casos caldrà modificar la manera com pinta Drupal alguns elements. També pot ser necessari haver d'afegir CSS o Javascript per a modificar l'aspecte o el comportament del costat client d'aquests comportaments.

Tenint en compte aquests recursos, realitzem un disseny aproximat de les pàgines. Podrem iterar sobre aquest disseny a mesura que l'anem implementant.

Per saber les opcions que tenim per a implementar-lo, hem de veure la iteració següent.

5.8.9. Iteració 9: maquetació - estils comuns

En aquest punt comptem ja pràcticament amb la funcionalitat de tota la pàgina web. Tenim les diferents pàgines i els elements d'aquestes que es mostren.

En aquesta iteració començarem a personalitzar la manera com mostra Drupal aquests elements, principalment l'aspecte que afecta totes les pàgines.

Escollim un tema similar al que ens interessa o un de mínim per a tenir una base sobre la qual puguem començar. El tema per defecte (*Bartik*) és bastant senzill i és una bona opció.

Tema

Per canviar el tema actiu, simplement anem a l'opció de menú **Appearance**. Hi podem seleccionar un dels diversos temes.

Si accedim als arxius de *template* podem veure els *templates* genèrics. Els podem trobar a **/themes/bartik/templates/**.

També hi ha un arxiu *template.php*, que defineix funcions genèriques per al procés dels *templates*. Aquest arxiu permet modificar molt a fons certes funcionalitats, però també és més complex.

Webs recomanats

Per aprofundir més en les opcions que permet *template.php*, podeu consultar l'adreça d'Internet següent:

http://api.drupal.org/api/drupal/includes!theme.inc/function/theme/7

http://api.drupal.org/api/drupal/themes!garland!template.php/7

Per a aquest pas, triem el *template* comú a totes les pàgines: **page.tpl**. Hi incloem les rutes de css i javascript necessàries per al desenvolupament en el mateix arxiu.

Si hi afegim aquests arxius podem maquetar amb CSS i desenvolupar la programació client en Javascript, de la mateixa manera que en qualsevol altre desenvolupament web.

Nota

Inclourem els arxius css i javascript de la forma estàndard:

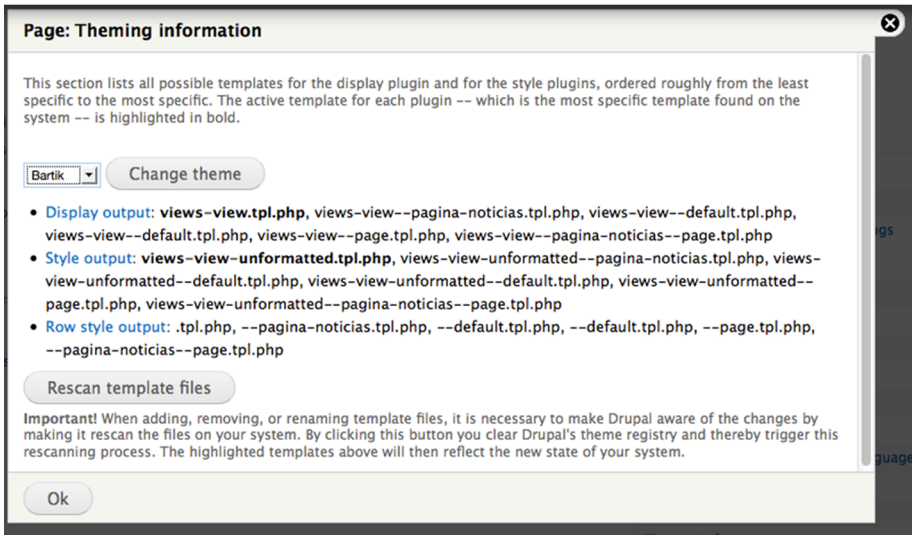
http://www.w3schools.com/tags/tag_link.asp

5.8.10. Iteració 10: maquetació - *templates* específics

Drupal crea diferents *templates* per a les vistes, pàgines i blocs. Pràcticament, qualsevol element present en el gestor té un *template* equivalent. Si modifiquem aquests elements, podem canviar totalment com es mostren tots els elements d'un lloc web desenvolupat en Drupal.

Un dels recursos més flexibles per a gestionar el contingut, i sobre el qual basem una gran part de l'organització dels elements, són les vistes. De vegades pot resultar complicat trobar el nom del *template* adequat per a una vista. Una bona manera de trobar els *templates* disponibles per a una vista és anar a la pàgina amb tota la informació que conté.

Des de **Structure** > **Views** editem la vista de la qual volem obtenir la informació. En l'edició podem veure diferents apartats. En l'apartat d'**Opcions avançades** seleccionem l'opció **Themes**, on es mostra la ubicació de tots els arxius de *templates* disponibles.



Si modifiquem o creem aquests arxius, podem personalitzar l'html que Drupal pintarà per a l'element (pot ser una vista o qualsevol altre tipus d'element).

5.8.11. Iteració 11: menú i navegació

Ara que ja tenim pàgines amb contingut podem establir un mecanisme per a navegar entre elles. Situarem les diferents rutes de les pàgines en un menú.

A **Structure** > **Menu** podem editar els menús i, des de l'opció **List links**, és possible veure els enllaços dels quals consta un menú i afegir-ne o treure'n d'altres.

Amb aquests canvis podrem modificar els tipus de menú que Drupal ens ofereix per defecte:

- **Main menu**: normalment, és un menú horitzontal a la part superior de la pantalla. Mostra les seccions principals del lloc.

- **Management**: aquest tipus de menú conté opcions per a les funcionalitats administratives del *site*.

- **Navigation**: aquesta tipologia de menús conté enllaços adreçats als visitants del lloc. Alguns mòduls afegeixen nous menús a aquest tipus quan s'utilitzen.

- **User menu**: menú vinculat a l'usuari. Conté els ítems de *Log out* i els propis del compte de l'usuari actiu.

Qualsevol d'aquests menús pot ser estilat amb CSS des de fulls d'estil inclosos en Drupal a partir de *templates* genèrics. El mateix succeeix amb el Javascript que utilitzin.

En alguns casos, especialment si volem un control total sobre com es pinta el menú en HTML, caldrà buscar els *templates* del menú i modificar-los. Pot passar que, si el comportament que volem modificar és molt específic, necessitem canviar el comportament del *core* de Drupal. Per a aquests casos, podem utilitzar les funcions de **template.php**.

Webs recomanats

A més d'altres pàgines de referència amb les opcions disponibles, alguns bons recursos de casos pràctics per a aquest tipus de modificacions són els següents:

http://drupal.org/node/11811

http://answers.oreilly.com/topic/1512-using-drupals-templatephp-for-overrides/

http://www.nerdliness.com/article/2008/01/01/power-template-php

http://stackoverflow.com/questions/5347154/drupal-template-php

http://drupal.org/node/223430

Hi ha un gran nombre de mòduls que ens ajuden amb la gestió, la personalització i l'aspecte dels menús. És aconsellable investigar quins ens poden anar més bé.

Mòduls per a la gestió i la personalització de menús

Entre els més útils trobem els següents:

- *Jump*. Crea una opció de navegació ràpida per a qualsevol vocabulari definit.
- *Menu block*. Crea llistes de blocs personalitzables i molt adaptables en els menús.
- *Menu minipanels*. Permet afegir contingut ric (*rich content*) en les opcions de menú.
- *Nice Menus*. Crea menús desplegables atractius que funcionen correctament en tots els navegadors.
- *Nice Primary Menus*. Divideix els menús en una llista de menús primaris i secundaris i facilita la navegació entre ells.
- *Site Menu*. Crea un arbre de navegació del *site* a partir de les categories que s'hi defineixen.
- *Submenu Tree*. Permet estructurar el contingut de manera jeràrquica i navegar segons aquesta jerarquia.
- *Superfish*. Permet utilitzar la llibreria *Superfish* automàticament en un menú de Drupal. Aquesta llibreria genera menús atractius d'una manera molt senzilla.

5.8.12. Iteració 12: cercador

En aquesta iteració investiguem la funcionalitat de cercador de l'aplicació.

Drupal incorpora aquesta funcionalitat per defecte. Gràcies a això no caldrà desenvolupar-la. Podem personalitzar les opcions de cerca des de **Configuration** > **Search settings**. Des d'aquest panell d'opcions podem realitzar, entre d'altres, les accions següents:

- Forçar la reindexació de tot el lloc web.

- Seleccionar en quins mòduls s'efectua la cerca i en quins no.

- Definir l'ordre d'importància de cada tipus de resultats, de manera que se'n destaquin uns per sobre dels altres.

5.8.13. Iteració 13: internacionalització

En aquesta iteració incorporarem funcionalitat multiidioma a tots els continguts que apareguin en el lloc. Hi ha diversos mòduls de Drupal per a aquesta funcionalitat.

1) Mòduls útils

Instal·larem els mòduls següents:

- *Internationalization*
- *Pathauto*
- *Token*
- *Transliteration*
- *Variable*
- *Chaos Tools*
- *Views*
- *Internationalization Views*

Per defecte, Drupal ve només amb l'idioma anglès. Afegirem el nostre a **localhost/admin/config/regional/language** a **Add language**.

En l'opció **Configure** també podem determinar quin dels idiomes és l'idioma per defecte i altres paràmetres.

2) Selector d'idioma

Activem l'opció de canviar d'idioma a **Structure** > **Block** per poder veure el selector d'idiomes a la pàgina.

3) Traduint contingut

Perquè aparegui el contingut traduït hem d'activar a **Publishing options** > **Multilingual support** l'opció **Enabled, with translation**.

Una vegada activada, observarem que apareixen pestanyes per a traduir quan es creen els continguts.

Amb aquestes opcions podrem traduir fàcilment el contingut del lloc.

Webs recomanats

Altres mòduls de traducció que ens poden ser útils són els següents:

Administration Language

Administration Menu

Demonstration site (Sandbox / Snapshot)

Moduli Filter

5.8.14. Iteració *n*: ...?

Qualsevol canvi que es produeixi en el desenvolupament del *site* a Drupal es pot fer seguint aquest procés incremental. La majoria de noves funcionalitats són facilitades trobant el mòdul de Drupal adequat. Els canvis de maquetació o de client segueixen el desenvolupament normal de CSS o Javascript. En els casos en què s'ha de modificar l'html generat, podem accedir al *template* corresponent i canviar la manera com es pinten els elements.

Totes les marques són propietat dels amos respectius. L'autor no està afiliat amb Drupal ni amb cap de les marques que s'esmenten en aquest document.

6. Desenvolupament d'una aplicació per a Facebook

6.1. Introducció

El projecte consistirà a desenvolupar una aplicació per a Facebook.

L'aplicació consisteix en un concurs fictici amb les característiques següents:

- Amb diferent contingut per a usuaris que han fet "M'agrada".
- Acceptació de bases legals.
- Disseny de les diferents pantalles que l'usuari utilitzarà.
- Recollida de dades de l'usuari i emmagatzematge en una base de dades.
- Opcions de compartir amb amics.

6.2. Per què Facebook?

6.2.1. Avantatges

Els avantatges que presenta Facebook són els següents:

1) Disposa d'una gran quantitat de base d'usuaris.

2) Facilita la connexió amb usuaris reals.

3) Una gran part de persones passen molt de temps a Facebook i ens dóna la possibilitat d'apropar la nostra marca o producte a aquest públic.

6.2.2. Inconvenients

Cal tenir en compte que quan desenvolupem una aplicació per a Facebook depenem d'aquesta empresa. L'API pot canviar en qualsevol moment (i, de fet, canvia), de la mateixa manera que els requisits de les aplicacions. Aquests canvis poden fer que certes aplicacions deixin de funcionar o funcionin de manera incorrecta quan abans ho feien normalment.

Reflexionar abans de començar

Abans de començar el desenvolupament d'una aplicació en Facebook, convé plantejar-se realment si és el més convenient per als objectius que el client pretén aconseguir. En alguns casos, una aplicació web estàndard ofereix més possibilitats per al client.

6.3. Opcions per a desenvolupar en Facebook

Facebook ens permet desenvolupar amb un sistema de *markup* propi anomenat *FBML* i amb un model de desenvolupament igual que una pàgina web normal. Com que no hi ha cap funcionalitat de FBML que no estigui ja disponible des d'un desenvolupament clàssic i que hi hauria una corba d'aprenentatge, optarem sempre per desenvolupar prescindint de FBML.

Per a aquest desenvolupament, podem usar qualsevol llenguatge del costat del servidor. En aquest cas utilitzarem PHP amb el *framework Codeigniter*.

Lectura complementària

Referència de documentació del *framework* **Codeigniter**:

http://codeigniter.com/user_guide//

6.4. Consideracions per a desenvolupar en Facebook

Temes de política de privadesa. En el cas d'una aplicació com la d'aquest cas, un concurs, convé mostrar unes bases legals que l'usuari ha d'acceptar abans d'entrar a l'aplicació.

6.4.1. Certificats segurs

Des de fa un cert temps, Facebook exigeix que les aplicacions tinguin el protocol segur. Caldrà tenir en compte això abans del desenvolupament per a contractar un certificat segur.

Lectura complementària

Per a més informació sobre el protocol segur, podeu consultar l'adreça d'Internet següent:

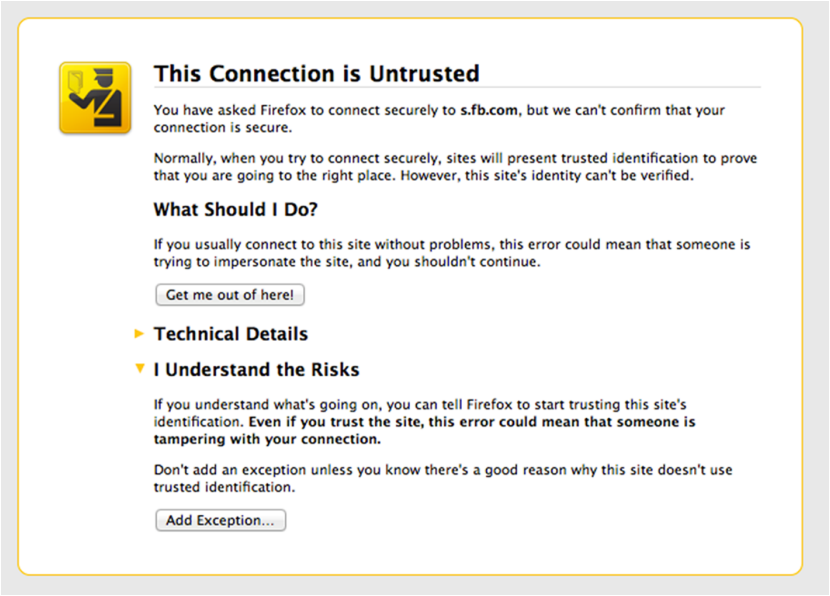
http://es.wikipedia.org/wiki/hypertext_Transfer_Protocol_Secure

Certificats segurs

La gran majoria d'empreses de *hosting* ofereixen serveis de contractació de certificats segurs. Ens interessa un certificat segur emès per una entitat important, ja que els navegadors mostren missatges d'advertiment quan els certificats no són d'entitats gaire importants, mentre que no els mostren per a les principals entitats certificadores. Aquests missatges, encara que són purament informatius, poden fer que l'usuari abandoni la pàgina.

Certificat d'entitat

La imatge que es mostra a continuació és la que es trobaran els usuaris si el certificat no és d'una entitat certificadora (CA) important. Malgrat que el missatge és acceptable, produeix més alarma de la que molts usuaris poden tolerar (deixaran la pàgina).



6.4.2. Seguretat

En les aplicacions de Facebook hem de seguir les mateixes mesures de seguretat que en el desenvolupament d'una pàgina web. Requereixen una atenció especial les aplicacions que tracten informació de caràcter personal o de concursos.

Webs recomanats

Alguns bons recursos sobre la seguretat en el desenvolupament de pàgines web són els següents:

http://programmers.stackexchange.com/questions/46716/what-should-every-programmer-know-about-web-development

https://www.owasp.org/index.php/main_Page

6.4.3. Limitacions del contingut

A l'hora de dissenyar i maquetar un aplicació per a Facebook hem de tenir en compte que no disposem de tot l'espai de pantalla disponible en una pàgina web. La integració amb Facebook implica una sèrie d'elements d'interfície de Facebook que inclouen el nostre contingut.

En aquests moments, l'espai que Facebook permet a la nostra aplicació és de 760 px d'amplària. En vertical no hi ha cap limitació concreta, ja que podem fer ús de l'*scroll*. Caldrà considerar que una part de l'espai vertical també està ocupat per interfície de Facebook, de manera que s'ha de ser especialment curós per a controlar què veu l'usuari en una primera visita abans de fer *scroll*.

6.5. Visió general

Per a maquetar una aplicació de Facebook s'utilitzen les mateixes eines i llenguatges que per al desenvolupament d'una web estàndard. L'aspecte es defineix mitjançant html i CSS, mentre que el comportament es desenvolupa en Javascript (per a programació del costat del client) i en un llenguatge servidor (en aquest cas, PHP) per al costat del servidor. Aquest llenguatge servidor sol connectar típicament amb una base de dades (en aquest cas, MySQL) per a persistir les dades.

El desenvolupament en PHP se sol fer a partir d'algun *framework*. Entre els *frameworks* més utilitzats en PHP trobem els següents:

- *Codeigniter*. És un *framework* lleuger amb un alt rendiment i les funcionalitats més habituals, sense forçar d'una manera excessiva a un desenvolupament concret per part dels desenvolupadors.

- *Kohana*. Està basat inicialment en **Codeginiter**. Incorpora moltes millores i utilitza funcionalitats de versions més recents de PHP.

- *Cakephp*. Aquest *framework* s'inspira en *Ruby onRails* per a tenir una funcionalitat similar, amb la filosofia de *convention over configuration*.

En aquest cas utilitzarem el *framework* **Codeigniter**.

Lectura complementària

Podeu trobar més informació sobre *convention over configuration* en l'adreça d'Internet següent:

http://en.wikipedia.org/wiki/convention_over _configuration

En l'enllaç següent es pot veure una comparativa entre diferents *frameworks*:

http://en.wikipedia.org/wiki/comparison_of_web _application_frameworks #PHP_2

6.6. Especificacions

Què vol el client?

El client ens sol·licita una aplicació en Facebook. Pensa que això pot dinamitzar la imatge del seu producte. Li agradaria tenir més informació sobre els potencials clients del seu producte, però no sap exactament com pot relacionar això amb Facebook.

Què se li proposa?

Li proposem crear un concurs en Facebook. La gent es farà soci per participar. S'obtindran dades sobre els usuaris i es dinamitzarà la marca.

6.7. Disseny

La fase de disseny ens ajudarà a comunicar-nos amb el client. És important que el client no percebi el disseny com el producte.

El disseny

El disseny és un artefacte del procés de desenvolupament. Serveix per a comunicar-se amb el client. **No és el producte final**. El producte final és l'aplicació en funcionament. Succeeix el mateix amb els documents d'especificacions, presentacions i similars.

Si això passa una vegada finalitzat el disseny, el client percebrà que l'aplicació ja deu ser funcional. És responsabilitat nostra comunicar-nos amb el client de manera adequada.

6.7.1. Esbós general

Ens interessa concretar al més aviat possible la funcionalitat amb el client i garantir que anem en la mateixa direcció. Podem esbossar el disseny dels elements necessaris i la seva interacció mitjançant *wireframes*.

L'avantatge dels *wireframes* és que ens permeten comunicar el disseny al client sense perdre la perspectiva pels detalls.

Decisions senzilles

Les decisions aparentment senzilles, a causa de la senzillesa aparent, poden ser molt més difícils de pactar (tothom es veu amb capacitat de decidir). Podeu trobar-ne més informació en la pàgina d'Internet següent:

http://en.wikipedia.org/wiki/parkinson%27s_Law_of_Triviality

Si hi ha incoherències o discrepàncies en el funcionament, convé afrontar-les al més aviat possible en el procés de desenvolupament.

Encara que hi pot haver canvis en el disseny, els *wireframes* ens serviran per a tenir un marc a partir del qual podem desenvolupar aquests canvis. D'aquesta manera no es replantejarà el redisseny de tots els elements possibles, sinó que la funcionalitat i navegació serà bàsicament la mateixa.

Això ens permet començar la implementació abans que el disseny final estigui tancat. El que sí hem d'assegurar és que el client comprèn correctament el paper dels *wireframes* i que es compromet amb la funcionalitat bàsica. Això no obre la porta a canvis irracionals i il·limitats.

Hi ha una gran quantitat d'aplicacions i serveis web per al desenvolupament de *wireframes*. En aquest projecte utilitzarem *Lumzy*.

Algunes altres opcions són les següents:

1) Com a servei web

- *Lovely Charts*
- *Cacoo*
- *Mocking Bird*
- *Mockflow*

2) Com a aplicació nativa

- *Balsamiq*
- *Omnigraffe*
- *Lovely Charts*

També podem utilitzar un programa habitual de disseny o il·lustració (Fireworks, Adobe Photoshop, Adobe Illustrator...), si bé recomanem fer servir un programa específic de producció de *wireframes*.

Una última opció és usar la maquetació html amb CSS, amb uns estils simplificats per a realitzar els *wireframes*. Això és factible quan els encarregats de la maquetació són el mateix equip encarregat del disseny.

6.7.2. Disseny final

Dins del marc dels *wireframes* podem variar en cert grau el disseny. És important matisar que això no dóna llibertat total per a canviar qualsevol aspecte del projecte. Hi ha parts del disseny d'una aplicació o pàgina web que estan estretament relacionades amb la funcionalitat que tenen. Fer canvis de funcionalitat greus a última hora és una fórmula per al desastre.

És important que es mantingui una coherència global en el projecte, i els canvis massa locals en el disseny sense tenir en compte l'aplicació com un tot solen trencar aquesta coherència.

6.8. Iteracions

D'acord amb la metodologia àgil, el projecte s'estructurarà al voltant d'una sèrie d'iteracions de durada curta que incrementaran gradualment les funcionalitats de l'aplicació. Al final de cada operació tindrem alguna cosa tangible.

6.8.1. Iteració 1: definició del flux de l'aplicació i implementació mínima

Una de les primeres funcionalitats que necessitem és la navegació per les diferents pàgines del projecte segons el funcional.

En aquest cas, el flux de navegació serà el següent:

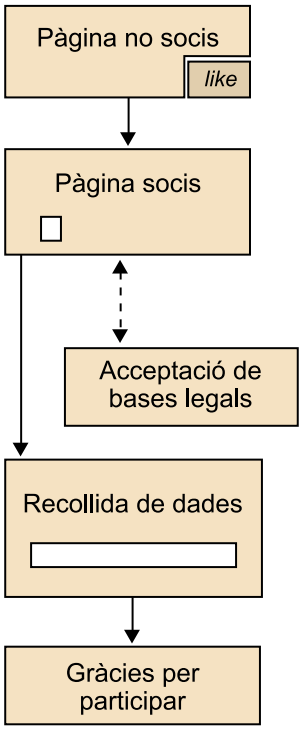
1) Accedim a la pàgina de presentació de l'aplicació. Per a accedir-hi necessitem fer "M'agrada" i donar els permisos adequats.

2) Veiem una pàgina especial només per a socis. Hi consten les bases legals i cal acceptar-les abans de poder seguir endavant en l'aplicació.

3) Després d'acceptar les bases legals procedim a emplenar les dades que ens sol·licita l'aplicació.

4) Una vegada emplenades les dades, es dóna les gràcies a l'usuari i se'ns informa dels passos següents, si n'hi ha.

Instal·lem *Codeigniter* i a continuació creem els controladors i els mètodes necessaris.



Definim les urls que acceptarà l'aplicació creant els mètodes corresponents en el controlador principal de l'aplicació.

Lectura complementària

En l'enllaç següent trobareu una guia d'usuari sobre el controlador:

http://codeigniter.com/user_guide/general/controllers.html

Utilitzem *pretty urls* per als diferents passos:

- `/ /`. La ruta base de l'aplicació. Quan entrem ens adreça aquí.
- `/soci`. La ruta per als usuaris que ja han fet "M'agrada".
- `/formulari`. Aquesta pàgina ens presenta les dades que cal emplenar.
- `/gràcies`. Pantalla final de l'aplicació. Se'ns dóna les gràcies per participar i se'ns informa de qualsevol dada rellevant.

En cadascuna de les urls definides en el controlador carreguem una vista corresponent, que simplement tindrà una pàgina buida amb el nom de la url. Carreguem cada vista des del mètode corresponent del controlador.

Lectura complementària

En l'enllaç següent trobareu una guia d'usuari sobre la vista:

http://codeigniter.com/user_guide/general/views.html

Al final d'aquesta iteració tenim el flux real de tota l'aplicació funcional, si bé el contingut de cadascun dels passos no és el real (ni tan sols hi són els elements més bàsics). En les iteracions següents anirem desenvolupant la funcionalitat corresponent a cadascun d'aquests passos.

Podem navegar per les diferents pàgines accedint a **www.nuestroservidor.com/el_nom_de_la_pàgina** (/, **soci**, **formulari**, **gràcies**).

6.8.2. Iteració 2: crear la nostra aplicació en Facebook

Fins ara hem accedit a la nostra aplicació, que és exactament igual a una aplicació web. En aquesta iteració integrarem la nostra pàgina per poder accedir des de dins de Facebook i començar a veure-la com realment la veuran els usuaris finals.

Ens interessa fer aquest pas al més aviat possible, per detectar inconsistències o limitacions de Facebook que hi pugui haver i poder solucionar-les tan aviat com sigui possible.

Creem un compte de Facebook de prova, amb el qual crearem l'aplicació. És interessant que els amics en aquest compte siguin pocs i de prova, ja que veuran notificacions sobre l'ús que fem de l'aplicació a mesura que l'anem desenvolupant.

Ens registrem amb el nostre compte de prova a www.facebook.com.

Creem l'aplicació a https://developers.facebook.com/apps i creem a **Crear nova app**.

Configurem l'aplicació. Necessitem un nom per a l'aplicació i un *namespace*, que determinaran la url final de l'aplicació.

En aquest procés hem de solucionar algun CAPTCHA[13] i proporcionar un número de mòbil o de targeta de crèdit perquè Facebook comprovi que les dades són reals.

(13)Acrònim de *completely automated public turing test to tell computers and humans apart.*

CAPTCHA

És un sistema automatitzat per a reconèixer que qui accedeix a una pàgina són persones i no programes. És molt útil per a minimitzar l'*spam* en comentaris o per a optimitzar recursos per als usuaris reals davant de *crawlers* o un altre tipus d'agents de programari.

El procés pot trigar bastant temps i aquí estem a la mercè de Facebook. Segons la nostra experiència, pot trigar des de poques hores fins a uns quants dies. Recomanem validar el compte de totes les maneres que tinguem a disposició si amb el pas dels dies no rebem el missatge al mòbil.

Configurem l'aplicació perquè apunti al lloc on s'allotgi la nostra pàgina.

Recordem que recentment Facebook només accepta aplicacions amb el protocol https://.

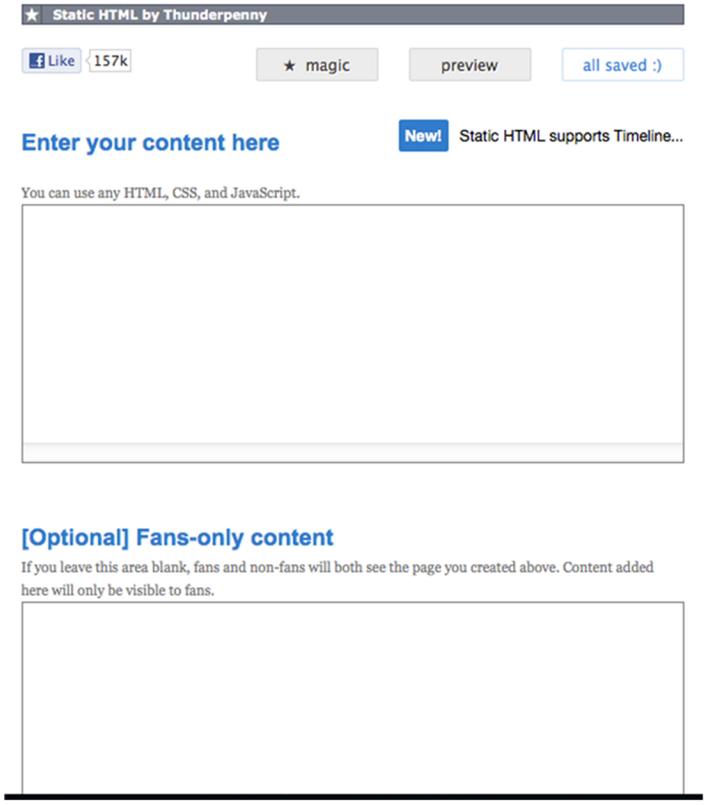
Webs recomanats

Algunes bones referències per a dubtes en el desenvolupament d'una aplicació en Facebook són les següents:

https://developers.facebook.com/docs/

http://facebook.stackoverflow.com/

Una vegada l'aplicació ha estat creada podem instal·lar alguna aplicació més que ens ajudarà a gestionar-la. Una d'aquestes aplicacions és *Static HTML by Thunderpenny*. Aquesta aplicació ens permet configurar dues pantalles de codi html, una per als usuaris que no han fet "M'agrada" i una altra per a ser vista quan ja han fet "M'agrada".



En la pàgina de la nostra *App* veurem un missatge al lateral, "Welcome". Si hi cliquem podrem començar a utilitzar aquesta aplicació.

Amb aquesta iteració ja tenim les rutes bàsiques funcionant amb contingut no real dins d'una aplicació de Facebook visible.

En les properes iteracions utilitzarem aquesta aplicació per a posar el codi corresponent per a no socis i per a socis.

6.8.3. Iteració 3: *wireframes* - pantalla per a no socis

En aquesta iteració i en les següents anirem definint a grans trets els elements presents en cadascuna de les pantalles de l'aplicació, basant-nos en el flux d'ús de l'aplicació.

En aquesta iteració volem mostrar la primera pantalla que veuran els usuaris de l'aplicació. Ha de ser atractiva i que cridi a l'acció clara, ja que és el que ha de convèncer els visitants perquè facin "M'agrada".

Posem un text que indiqui a l'usuari que ha de fer clic a "M'agrada" per entrar a l'aplicació.

En el *wireframe* següent podem veure com de simple és aquesta pàgina. La columna de la dreta representa la UI[14] de Facebook.

(14)Acrònim de *user interface*.

Amb aquest *wireframe* definit podem fer una implementació molt bàsica. En aquest cas, en tindrem prou només amb una imatge. Implementem en html els elements mínims que imitin el *wireframe*. Editem la vista per incloure en l'html una imatge i el codi html del botó "M'agrada".

A manera d'exemple, l'html podria tenir un aspecte semblant al següent:

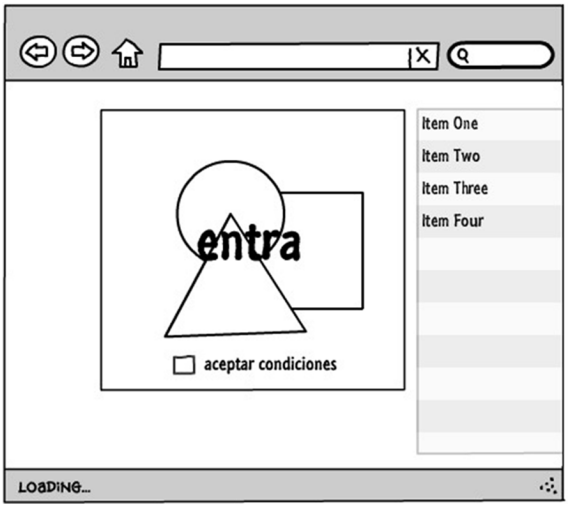
```
<! DOCTYPE html>
<html>
<head>
</head>
<body>
<div>
<img src="https://nuestrservidor.co/imgs/nosocio.jpg" />
</div>
</body>
</html>
```

Una vegada tinguem la pàgina implementada, podem fer **Anomena i desa** des del navegador per tenir la pàgina com a html estàtic.

Fem **Copiar** i **Pegar** d'aquest html estàtic i, en la pàgina de **static_html_plus**, ho enganxem en la casella corresponent als usuaris no fans (la primera).

Quan entrem en l'aplicació, veurem aquest codi estàtic fins que premem a "M'agrada".

6.8.4. Iteració 4: *wireframes* - pantalla per a socis



La pantalla per a usuaris que ja han fet "M'agrada" pot contenir més informació. Inclourem l'acceptació de les bases legals com a condició per a passar a l'aplicació pròpiament dita.

Per a realitzar una implementació bàsica d'aquest *wireframe* necessitarem:

1) Un enllaç o botó que vagi a la pàgina següent (formulari) en el flux definit.

2) Un codi javascript que validi que el *checkbox* d'acceptació de condicions està marcat.

3) Un enllaç al costat del *checkbox* d'acceptació de condicions que ens portarà (malgrat que encara no existeix) a les bases del concurs o aplicació.

4) Una imatge de fons o la maquetació corresponent en CSS per a una implementació mínima de la part gràfica d'aquest *wireframe*.

Una vegada realitzada aquesta implementació, guardem la pàgina com a codi estàtic i l'enganxem en la casella corresponent de **static_html_plus** al contingut de només fans (el segon apartat).

Si ja som fans de l'aplicació, veurem aquest contingut quan hi entrem. Amb això acaba el nostre ús de l'aplicació **static_html_plus**. Si més endavant volem retocar alguna d'aquestes due pàgines, haurem de tornar a copiar i enganxar l'html estàtic en la seva configuració.

6.8.5. Iteració 5: *wireframes* - formulari de recollida de dades

Definim ara les dades que volem recollir i com seran mostrades a l'usuari de manera simplificada amb l'ajuda d'un *wireframe*.

Per a fer una implementació mínima de la funcionalitat del formulari necessitarem:

1) Programar els formularis des de PHP per a incloure'n la validació, l'ordre en què el mostrarem i quins són obligatoris o no.

Formularis

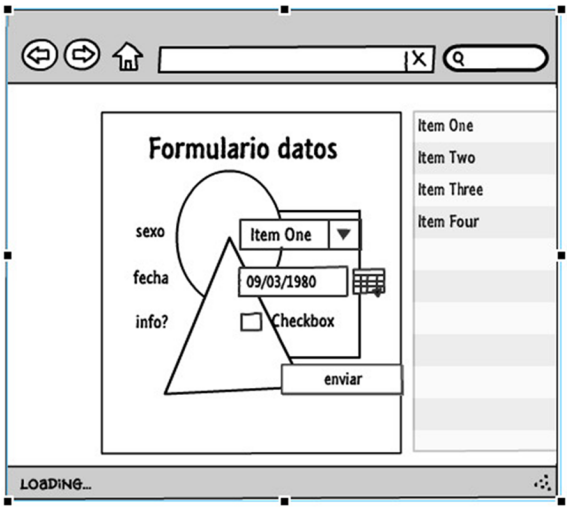
En el cas de **Codeigniter**, disposem de facilitats per a realitzar formularis i validacions complexes d'aquests.

En els enllaços següents trobareu guies de l'usuari referents a formularis i validació:

http://codeigniter.com/user_guide/helpers/form_helper.html

http://codeigniter.com/user_guide/libraries/form_validation.html

Altres *frameworks* tenen utilitats semblants. Si no en disposem de cap, haurem d'implementar aquesta funcionalitat nosaltres mateixos.



2) Validar des de javascript que els formularis compleixen les mateixes validacions, per no haver-los d'enviar per a fer la comprovació.

6.8.6. Iteració 6: còpia de seguretat en base de dades

Fins ara les pàgines no guardaven cap tipus d'informació de l'usuari; simplement eren navegables. En aquesta pàgina recollim dades de l'usuari.

Per a això haurem de recollir les dades del formulari i guardar-les en la base de dades de l'aplicació.

En instal·lar **Codeigniter** ja hem introduït les dades d'una base de dades. Ara necessitarem el següent: crear una taula per a guardar la informació i accedir a aquesta taula per a guardar les dades des del mètode de la nostra pàgina.

Crear estructura de la base de dades

Accedim amb una aplicació client de gestió base de dades a la base de dades.

Clients de base de dades

Existeixen molts clients de base de dades. Entre els d'ús habitual trobem els següents:

- http://dev.mysql.com/downloads/gui-tools/5.0.html: dels creadors de Mysql; per tant, és molt recomanable per a gestionar i treballar amb aquesta base de dades. Són una sèrie d'utilitats disponibles per a múltiples sistemes operatius que s'instal·len com a aplicacions natives.

- http://www.phpmyadmin.net/home_page/index.php: una aplicació web que s'instal·la en el servidor i permet gestionar-hi, amb un inici de sessió i una contrasenya, les bases de dades presents en el servidor.

Recomanem sempre que sigui possible usar aplicacions natives per a gestionar la base de dades. Solen ser més eficients, la qual cosa es traduirà en més eficiència a l'hora de desenvolupar i realitzar tasques de manteniment.

Hi creem una taula de nom *registres* amb els camps necessaris i els tipus corresponents. En aquest cas, segons el disseny bàsic del qual disposem, els camps són:

- **Sexe** (home/dona)
- **Data** (YYYY-mm-dd)
- **Rebre més info** (sí/no)

L'aplicació generarà per nosaltres el codi SQL[15] necessari per a crear les taules. Un desenvolupador pot crear la taula mitjançant *scripts* o algun altre sistema d'abstracció que acabi generant aquest codi de la mateixa manera.

(15)Acrònim de *structured query language*.

SQL

És un llenguatge de programació emprat per a tractar amb bases de dades relacionals.

En general, sol ser més pràctic i intuïtiu generar l'estructura inicial de taules mitjançant eines gràfiques, però no és l'única opció.

Guardar els registres en la base de dades

Les aplicacions desenvolupades en **Codeigniter** utilitzen, com ja hem vist abans, el model vista controlador. Fins ara hem fet servir els controladors per a distribuir el flux de l'aplicació.

Recordem que vam fer un sol controlador amb un mètode per a cadascuna de les pàgines disponibles en l'aplicació.

També hem utilitzat les vistes per a maquetar cadascuna de les pàgines de les quals consta l'aplicació. Al seu torn, aquestes vistes eren carregades en els controladors perquè apareguessin a les pàgines corresponents.

En aquest cas emprarem l'últim element d'aquesta manera d'estructurar codi: **els models** .

Els models abstreuen la manera com accedim als recursos en una aplicació. Una manera pràctica de pensar en els models és pensar que són actors que prenen part en l'acció. Així, no pensarem en un model base de dades, sinó en un model usuari, que disposarà dels mètodes "Inserir" o "guardar_dades".

Lectures complementàries

En els enllaços següents podeu trobar una guia de l'usuari sobre el model vista controlador i models:

http://codeigniter.com/user_guide/overview/mvc.html

http://codeigniter.com/user_guide/general/models.html

Creem un model "usuari" i un mètode "guardar_dades", que acceptarà les dades recollides en el formulari i les guardarà en la base de dades.

Lectura complementària

En l'enllaç següent trobareu una guia de l'usuari sobre base de dades:

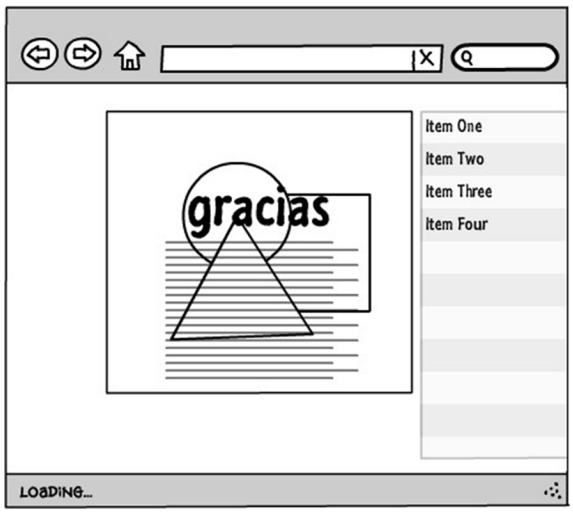
http://codeigniter.com/user_guide/database/index.html

Per accedir a la base de dades des d'un model **Codeigniter** ens proporciona algunes utilitats que ens faciliten la tasca. Són totalment opcionals, però poden agilitar el procés de desenvolupament si hi estem familiaritzats.

6.8.7. Iteració 7: *wireframes* - pantalla final

En aquesta pantalla simplement informem l'usuari que les seves dades s'han enviat correctament. És també el lloc indicat per a avisar-lo de futures accions o esdeveniments que es poden esperar. Li donem les gràcies per haver participat en el concurs o haver utilitzat la nostra aplicació.

El *wireframe* corresponent seria el següent:



6.8.8. Iteració 8: bases del concurs

Afegirem una pàgina més a les pàgines definides fins ara per incloure un enllaç a les bases legals de l'aplicació. Una manera senzilla d'implementar-lo és enllaçar directament a un arxiu PDF[16].

(16)Acrònim de *portable document format.*

PDF

És un format d'Adobe, considerat un dels estàndards *de facto* per a l'intercanvi d'arxius. És un bon format, per tant, ja que es pot obrir en qualsevol sistema i entorn i els usuaris hi estan familiaritzats.

D'aquesta manera no caldrà modificar l'accés a la pàgina. Simplement posarem l'enllaç en la vista corresponent.

Aquest arxiu pot ser proporcionat pel mateix client. Depenent de la dimensió del client i de l'àmbit del seu negoci podrà disposar d'un departament específic encarregat de tractar qüestions legals. En aquest cas és molt recomanable que aquest departament sigui l'encarregat de realitzar aquest document. Nosaltres simplement l'enllaçarem perquè l'usuari el pugui llegir abans d'acceptar les condicions.

Disseny final

Fins aquí tenim una implementació funcional completa. Seguim el flux de l'aplicació que vam definir i hem fet una implementació bàsica, però totalment funcional, a partir dels *wireframes* en comunicació contínua amb el client. En aquest moment, el disseny final hauria d'estar tancat per a poder maquetar les pàgines mitjançant CSS i polir-les perquè siguin visibles tal com les experimentarà l'usuari final.

6.8.9. Iteració 9: disseny final - pantalla per a no socis

A partir dels *wireframes*, que ja són funcionals, només ens falta maquetar de manera específica el disseny final de la pantalla prèvia per als no socis.

Segons el *wireframe* anterior, en aquest cas es tracta simplement d'una imatge, per la qual cosa posem una atenció especial a les dimensions de la imatge.

Una vegada exportada la imatge, guardem l'html i l'actualitzem entrant a la pestanya de Facebook de **Static HTML**. En aquest cas, anirà en el codi que no és exclusiu per a fans (el primer requadre).

6.8.10. Iteració 10: disseny final - pantalla per a socis

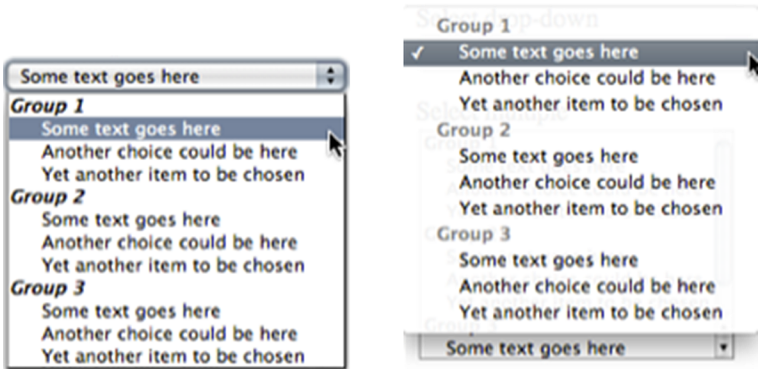
La pantalla per a persones que ja ha fet "M'agrada" consta d'una imatge més gran que la de no socis. Al marge del *check* obligatori i de l'enllaç a les bases legals, s'ha pogut maquetar pràcticament amb una sola imatge. També consta d'un text que convida a entrar en l'aplicació que, en aquesta iteració, serà substituït pel text definitiu.

Com que bàsicament és una imatge serà molt senzill de maquetar. Simplement, exportem la imatge en les mides adequades.

6.8.11. Iteració 11: disseny final - formulari de recollida de dades

En aquesta iteració actualitzem el disseny del formulari de recollida de dades de l'aplicació. Aquesta pantalla és una de les que probablement consta de més detalls. És important que els camps siguin atractius i fàcils d'emplenar i que quedi clar el format que s'espera en cadascun.

Hem de tenir en compte també que els controls de formularis (*inputs*, *selects*, *radiobuttons*, etc.) utilitzen controls natius en cada sistema operatiu. Això pot fer que un mateix formulari tingui aspectes lleugerament diferents, per exemple, en un entorn Mac Os o en un Windows. Els elements poden ser diferents fins i tot entre navegadors.



Mateix component *select* natiu en diferents plataformes

Depenent de l'aplicació, això pot no ser un problema. Al cap i a la fi, la funcionalitat és exactament la mateixa, però és important comunicar aquestes opcions al client.

Si es vol exactament el mateix disseny per motius de *look and feel* de l'aplicació, podem usar alguna llibreria per a ajudar-nos a tractar les diferències entre els formularis dels diferents entorns.

Lectura complementària

Podeu trobar més informació sobre *look and feel* en la pàgina d'Internet següent:

http://en.wikipedia.org/wiki/look_and_feel

Un disseny exacte d'aquests components de formulari pot no ser necessari i comporta un temps extra de desenvolupament, segons la familiaritat dels desenvolupadors amb les biblioteques emprades.

Jnice

Una bona opció per a maquetistes i programadors client és Jnice.

Permet maquetar components a mida. D'aquesta manera aconseguim que es vegi i se simuli el comportament en el component propi, mentre que per sota s'utilitza el component natiu de cada plataforma.

6.8.12. Iteració 12: disseny final - pantalla final

La pantalla final és l'últim pas per a acabar la maquetació. És una pantalla simple, amb pocs elements. Donem les gràcies a l'usuari i incloem un text en què l'avisem de la data del resultat del concurs i futurs esdeveniments.

6.8.13. Iteració 13: protecció de dades

És important que guardem únicament les dades necessàries per al concurs i que seguim la Llei oficicial de protecció de dades (LOPD). Les multes per incomplir aquesta llei són extraordinàriament altes.

També és important que la seguretat amb què es troba l'aplicació i la resta de dades siguin les adequades.

Abans de llançar l'aplicació convé fer-hi un últim repàs per a detectar si hi ha algun error de seguretat que hagi passat inadvertit durant el desenvolupament de l'aplicació.

Des de les primeres iteracions funcionals ja hem pogut provar el funcionament de l'aplicació. En aquest punt, podem realitzar les últimes proves o executar els tests unitaris una darrera vegada abans de llançar l'aplicació.

6.8.14. Iteració 14: compartir en mur o amb amics

En l'última pantalla de l'aplicació, pot ser útil proposar a l'usuari compartir aquestes dades amb algun amic o en el seu propi perfil de Facebook. Si el concurs és atractiu per a ell, probablement també ho serà per als amics.

Per a incloure aquesta opció utilitzem un dels *widgets* de Facebook.

Web recomanat

Aquí podem veure les diferents opcions disponibles per a situar el botó a la nostra pàgina:

http://developers.facebook.com/docs/share/

Exemple de botó de compartir de Facebook

Copiem el codi que Facebook ens proporciona i l'afegim en l'html de l'última pantalla. Probablement caldrà fer alguns petits retocs en la maquetació per a situar el botó justament en el lloc on volem.

6.8.15. Iteració 15: llançament i posada en societat

Fem pública l'aplicació i la viralitzem. Incloem l'adreça en material promocional de l'empresa o *mailing* o contactem amb una empresa de mitjans que viralitzi la nostra campanya.

A partir d'aquí haurem de recollir diferents mètriques per anar veient l'èxit de la nostra aplicació.

6.8.16. Iteració *n*: ...?

Si calgués modificar o ampliar funcionalitat per a un millor funcionament de l'aplicació podríem seguir aquest mateix model de desenvolupament, amb iteracions ràpides que millorin de manera incremental l'aplicació.

Totes les marques són propietat dels amos respectius. L'autor no està afiliat amb Facebook ni amb cap de les marques que s'esmenten en aquest document.