TFC-Programació d'aplicacions per a mòbils usant HTML5

Victor Corominola Ocaña

Gener 2015

Document acumulatiu d'estudi, d'experiències i d'objectius d'aprenentatge per a la realització d'aplicacions mòbils amb HTML5

Contingut

Descripció general del projecte

Avui en dia, ja sigui pel ritme de vida que portem, pel context econòmic actual, o simplement per desconfiança, hem perdut la costum que tenien els nostres antecessors (avis o besavis) de relacionar-nos i conèixer als nostres veïns i la gent que ens envolta.

Les relacions socials s'estan estenent en el món tecnològic però són, habitualment, d'un àmbit global o extens.

Per altra banda, les empreses i fabricants de productes al consum, cada vegada estan més interessades en conèixer costums i hàbits de les persones per poder oferir vendes dirigides (conegut com *profiling*).

Aquesta aplicació pretén ser el punt de partida per generar una xarxa de coneixement de l'entorn immediat, circumscrit al veïnat o barri de residència dels usuaris.

Com a repte (a més a més del propi tecnològic), també sorgeix l'adaptació de la Llei Orgànica de Protecció de Dades, en quant aquesta informació (tot i ser proporcionada per tercers, segons l'arquitectura i persistència, podria ser necessària informar a l'Agència de Protecció de Dades i als propis usuaris).

Objectius d'aprenentatge

Objectiu general

L'objectiu general és aprendre a desenvolupar una aplicació, el màxim de transportable possible (entre diferents plataformes), tot i que s'orienta principalment a sistemes Android, emprant eines de tecnologia web estàndard (com HTML5, CSS3 i JavaScript).

El projecte consisteix en poder fer una aplicació mòbil híbrida, amb una càrrega de dades al servidor, i una mica de càrrega d'informació als dispositius (bàsicament la capa de presentació i un identificador de registre per anar sol·licitant informació al servidor). La transmissió de missatges s'efectuarà mitjançant l'especificació RESTful.

Amb aquesta metodologia, el que es vol obtenir és:

a) Fer una aplicació que pugui funcionar tot i no tenir connexió de dades
b) Permetre als usuaris desar de manera local informació "*en itinere*"
c) Tenir una aplicació amb un bon rendiment.

Objectius principals

Com a objectius principals del projectes es plantegen:

- Creació d'una aplicació *responsive* (adaptable a diferents mides i models de pantalla, que conservi bon aspecte)
- La creació d'una aplicació mitjançant REST per a realitzar els CRUD's necessaris (entitats detectades en un primer anàlisi: usuaris, persones, actes, i objectes). La intenció inicial és implementació mitjançant base de dades (probablement MongoDb)
- Desenvolupar l'aplicació per tal que la transmissió de missatges la faci encapsulant en JSON i emprant JSSS, com JQueryMobile.js i Node.js
- Intentar incrustar aquesta Smart Client App en una Hybrid App o bé en una SPA (Single Page Access)

Planificació

Temporització del projecte

La temporització està marcada per, bàsicament quatre fites:

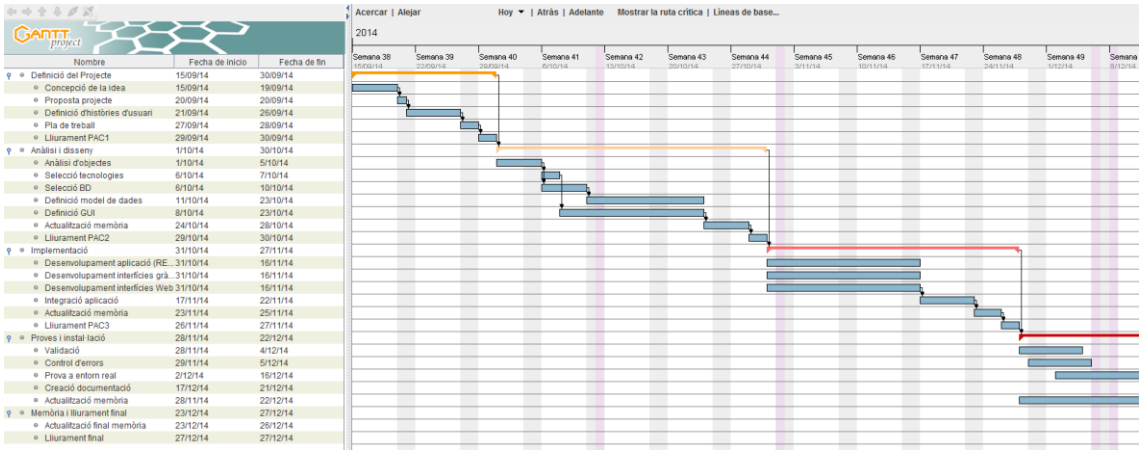
- Lliurament PAC1
- Lliurament PAC2
- Lliurament PAC3
- Lliurament final i Debat Virtual (si s'escau)

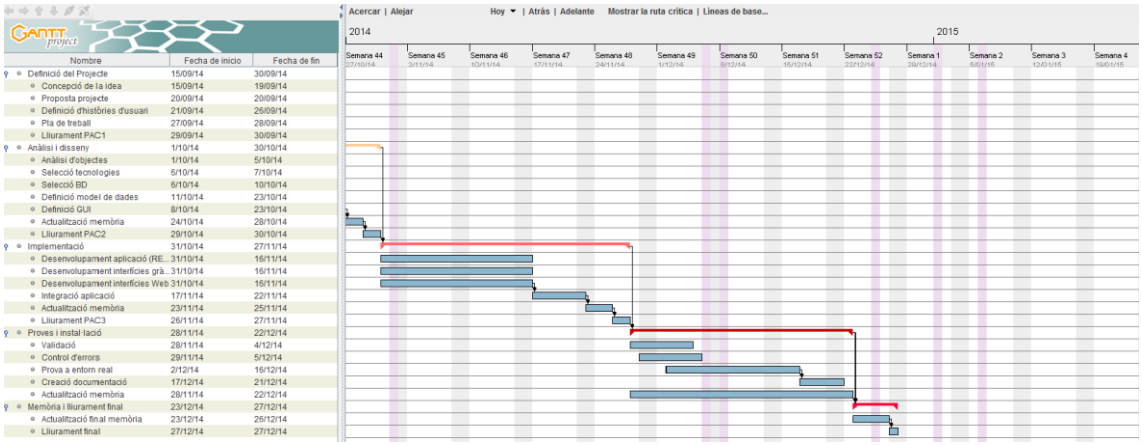
Calendari

El calendari previst és el següent:

	Data Prevista		Data Prevista
Definició del Projecte	**30/09/14**	**Implementació**	**27/11/14**
Concepció de la idea	19/09/14	Desenvolupament aplicació (RESTful)	16/11/14
Proposta projecte	20/09/14	Desenvolupament interfícies gràfiques	16/11/14
Definició d'històries d'usuari	26/09/14	Desenvolupament interfícies Web	16/11/14
Pla de treball	28/09/14	Integració aplicació	22/11/14
Lliurament PAC1	30/09/14	Actualització memòria	25/11/14
Anàlisi i disseny	**30/10/14**	Lliurament PAC3	24/11/14
Anàlisi d'objectes	5/10/14	**Proves i instal·lació**	**22/12/14**
Selecció tecnologies	7/10/14	Validació	4/12/14
Selecció BD	10/10/14	Control d'errors	5/12/14
Definició model de dades	23/10/14	Prova a entorn real	16/12/14
Definició GUI	23/10/14	Creació documentació	21/12/14
Actualització memòria	28/10/14	Actualització memòria	22/12/14
Lliurament PAC2	30/10/14	**Memòria i lliurament final**	**27/12/14**
		Actualització final memòria	26/12/14
		Lliurament final	27/12/14

Diagrama de Gantt





Històries d'usuari

Les històries d'usuari les usarem com a marc de desenvolupament de l'aplicació, per mostrar la seva funcionalitat i per conèixer l'estat aproximat de finalització de l'aplicació.

Es detalla per a cada història de treball un primer esbós de les pantalles a emprar i una aproximació de càrrega de feina (entre parèntesi la càrrega si es decideix afegir funcionalitats addicionals).

- **Registre:** El primer pas per a l'aplicació és registrar-se per poder tenir accés a veure, crear, modificar i eliminar entitats (persones, objectes i actes)
- **Autenticació:** Una vegada l'usuari està registrat al sistema, és necessari que s'autentiqui per poder accedir a l'aplicació, veure les entitats, i poder crear-ne de noves o editar les existents. Una vegada autenticat, té accés a l'aplicació.
- **Alta Persona:** Donat que es pretén que l'aplicació sigui auto-mantinguda pels usuaris, se'ls facilitat l'opció de crear, modificar i eliminar persones.
- **Alta Objecte:** Donat que es pretén que l'aplicació sigui auto-mantinguda pels usuaris, se'ls facilitat l'opció de crear, modificar i eliminar objectes.
- **Alta Acte:** Donat que es pretén que l'aplicació sigui auto-mantinguda pels usuaris, se'ls facilitat l'opció de crear, modificar i eliminar actes. En el cas dels actes, com

aquests estan fets per una persona o grup de persones en concret, serà necessari facilitar a l'usuari que seleccioni una persona existent o bé la faci de nou (redirigint-lo a la creació de persones).

També com apunt addicional tenir en compte la possibilitat d'activar la geolocalització per tal d'inserir la situació actual a l'acte que es dona d'alta

- **Llistat Persones**: El primer pas per a l'aplicació és registrar-se per poder tenir accés a veure, crear, modificar i eliminar entitats (persones, objectes i actes)

Requisits Inicials

L'objectiu consisteix en crear una aplicació per a dispositius mòbils que sigui el màxim de compatible amb els diferents dispositius disponibles avui en dia al mercat.

Tots els dispositius o si més no la seva gran majoria, convergeixen en emprar una tecnologia estesa inclús a dispositius d'escriptori: l'HTML5.

Aquesta tecnologia ens permet crear aplicacions funcionals (donat que el resultat no són aplicacions natives vinculades al maquinari) per a dispositius mòbils com SmartPhones, Phablets, Tablets, etc.

A més a més, la recent especificació d'HTML5, permet emprar les característiques pròpies i típiques d'aquests dispositius com són:

- Pantalla tàctil
- Càmera fotogràfica
- GPS

Tot això ens interessa sense perdre de vista de les possibilitat de sincronització d'informació entre els diferents usuaris

Requisits funcionals

El que interessa que ens proporcioni aquesta aplicació és informació de la gent que tenim pel nostre entorn.

També ens interessa que aquesta informació introduïda al sistema sigui retro alimentada pels mateixos usuaris. D'aquesta manera:

1. Tenim dades, en principi, reals (no fictícies)
2. Els usuaris són els que alimenten d'informació el sistema
3. Proporcionem una eina social als usuaris

Requisits no funcionals

Interfície (Visual)

Cal que la nostra aplicació tingui un disseny específic per poder aprofitar les possibilitats que ens faciliten les pantalles de dispositius mòbils.

Així doncs, caldrà que des d'una mateixa pantalla física, podem gestionar totes les operacions de l'aplicació de manera ràpida i el màxim de fàcil per a l'usuari.

Tecnologia a utilitzar

Un dels objectius més importants en el desenvolupament d'aquesta aplicació és la **mobilitat**. Per tant és necessari que l'usuari pugui fer servir l'aplicació en qualsevol moment i en qualsevol lloc.

Un altre dels objectius és la **universalitat**, és a dir, poder disposar d'una aplicació multi plataforma. Per aquesta raó, s'entén que és necessari poder proporcionar a l'usuari l'opció d'executar l'aplicació des de qualsevols dispositiu que faci servir (smartphone Android, Apple, Firefox OS, Windows Phone, o tablet, ordinador d'escriptori, etc.).

I per últim, també cal tenir en compte la velocitat d'execució i d'operació de la pròpia aplicació, juntament amb facilitat de manteniment (si s'escau).

HTML5

El llenguatge de marques HTML[1] és un llenguatge altament generalitzat: qualsevol dispositiu d'avui en dia n'incorpora.

És un llenguatge multi plataforma, lleuger (si més no per a mostrar informació indispensable) i que permet l'addició d'scripts[2].

En la seva cinquena revisió, el HTML5 ha incorporat una especificació més estricta (minúscules, obrir i tancar les etiquetes, etc.), noves etiquetes (tags), i un detall molt important: persistència de dades a través de l'objecte LocalStorage.

Aquesta nova funcionalitat permet, emmagatzemar de manera local (ja sigui persistent o només en una sola sessió), informació amb una quantitat superior als 4kb (limitació per les antigues "cookies").

Per tots aquests motius es considera idònia aquesta tecnologia.

JavaScript

El JavaScript és un llenguatge script, basat en el concepte de prototipus, implementat en base a l'estàndard ECMAScript.

La facilitat d'ús i la seva potència permet realitzar implementacions d'accions juntament amb un prototipatge basat en objectes (ideal per a modelitzar gairebé qualsevol dada), així com manipular i establir sistemes de connexió tipus client-servidor de manera asíncrona, permetent a l'usuari tenir una sensació de continuïtat en la navegació i l'execució de webs.

---

[1] Hyper Text Markup Language

[2] Fragments de codi sense compilar incrustat conjuntament amb el codi principal

CSS

El llenguatge CSS[3], és un llenguatge que ha proporcionat una millora visual a les pàgines web que es realitzaven inicialment.

En la seva tercera revisió (a l'aplicació s'ha emprat CSS3), permet la modificació visual i de comportament d'elements de les pàgines web, aconseguint una adaptació agradable i sorprenent en alguns casos per a l'usuari.

És ideal en quant a usabilitat (per exemple, a l'entrar en un camp que la caixa que el conté canviï de color) i a l'igual que el JavaScript o el HTML està altament estès i estandarditzat.

A més a més permet la redistribució dels elements en funció de la pantalla on s'estigui mostrant l'aplicació (tècnica coneguda com a *Responsive Design*).

S'ha realitzats molts esforços per facilitar l'ús avançat de CSS, i usuaris o empreses de manera altruista han facilitat aquest apropament amb *frameworks* basats en aquesta tecnologia.

En el cas de l'aplicació, s'ha emprat el framework BootStrap.

PhoneGap

Es un *framework* per el desenvolupament de aplicacions mòbils, ideat inicialment per una empresa privada als Estats Units i que, vist el potencial que proporciona, va ser adquirit per una gran companyia de programari: Adobe Systems Inc.

Actualment és gratuïta (en la seva versió de consola) i de pagament juntament amb eines per a desenvolupament de webs com el Adobe Dreamweaver CS6.

Permet als programadors utilitzar eines generals com JavaScript, HTML5 y CSS3 y aplicar-les als dispositius mòbils, produint una sensació visual i d'ús d'aplicació nativa al dispositiu a on s'executa.

A més a més, aquest procés de "transició" entre una aplicació HTML cap a una aplicació nativa és ràpid i senzill: només cal introduir el paquet HTML juntament amb les llibreries necessàries per a la correcta execució (com arxius js, css, etc.), i triar la plataforma destí (Android, Apple, Windows Phone, BlackBerry).

L'inconvenient, però, és que cal realitzar el procés de "compilació" per a cada tipus de plataforma, sense poder efectuar-se (a dia de la redacció d'aquesta memòria) per a dispositius amb Firefox OS.

Framework MEAN

Un cop definides les tecnologies a emprar, s'ha realitzat un estudi i/o comparativa de les eines que poden ser més adients per a la generació d'una aplicació d'aquestes característiques.

---

[3] Cascade Style Sheet

Donat que l'entorn de programació de dispositius mòbils és un entorn atractiu, que genera controvèrsia i que, per les seves possibilitats, té una alta demanda empresarial i social, així com un alt potencial, s'han generat diversos entorns de treball (altruistament) per tal de facilitar als novells les tasques de creació d'aplicacions.

S'han avaluat frameworks de programació com:

- **Titanium**

  Appcelerator Titanium és un framework lliure i open source per al desenvolupament d'aplicacions natives per a dispositius mòbil i aplicacions d'escriptori basades en tecnologia web. Proporciona a l'usuari més de 100 controls personalitzables com taules, botons, llistes, suport per a la geolocalització, xarxes socials, i multimèdia.
- **SenchaTouch**

  Sencha Touch és un framework per a desenvolupar aplicacions per a dispositius mòbils usant HTML5, que permet la creació d'aplicacions com si fossin natives per a sistemes Android o Apple iOS. Suporta HTML 5, CSS 3 i JavaScript.
- **Iwebkit**

  iWebkit 5 és una versió de framework per a la creació ràpida i senzilla d'aplicacions tàctils **només per a iPhone i iPad.** La versió actual té moltes característiques noves i es fàcil d'entendre per, en pocs minuts, poder realitzar alguna petita aplicació web.
- **JqueryMobile**

  jQuery Mobile és el framework jQuery orientat a dispositius mòbils. Suporta iOS, Android, Windows Phone, BlackBerry, Symbian, Palm webOS i més dispositius. En alguns casos (com el Adobe Dreamweaver CS6) ja ve incorporat amb el programari i permet la realització ràpida (mitjançant plantilles) d'aplicacions d'una sola pàgina[4].
- **Backbone**

  Backbone és una eina de desenvolupament per al llenguatge de programació JavaScript amb una interfície RESTful per JSON, basada en el paradigma de disseny d'aplicacions MVC[5]. Està dissenyada per a desenvolupar aplicacions SPA i per mantenir diferents parts d'una aplicació web (backend i frontend) de manera sincronitzada.

Aquestes eines i frameworks s'han anat descartant, o bé per la manca d'informació o per la sobre informació o per no trobar una solució adaptada al que es pretenia en l'aplicació objecte d'aquesta memòria.

Finalment es va optar per a un nou framework que sembla que està agafant embranzida i que, per motius com la fiabilitat, continuïtat del projecte framework o l'èxit, van proporcionar més confiança: el framework **MEAN.**

---

[4] SPA (Single Page Application) són aplicacions multipàgina que s'executen contenides en una única página principal

[5] Model – Vista - Controlador

Aquest framework, realment està basat en quatre tecnologies, i permet una programació adaptada a la nostra metodologia MVC.

MEAN: What it MEANs?

MEAN són les sigles de quatre tecnologies que, associades entre sí, proporcionen la possibilitat de crear aplicacions escalables, segures i distribuïdes:

- **M**ongoDB
- **E**xpress
- **A**ngular
- **N**ode

A grans trets aquest framework es treballa de la següent forma:

a) Es programa i personalitza el servidor Express, amb tots els mòduls i ampliacions que es necessitin. En aquest context és on es programen les accions típiques d'una aplicació RESTful, amb l'especificació del que farà cada verb http (GET, POST, PUT, DELETE).
b) A continuació, s'executa el servidor programat i personalitzat a l'aplicació dins d'un servidor amb imatge Node.js. Aquest és l'encarregat de canalitzar els diferents *threads* que es rebin al servidor (recordem que és asíncron).
c) La persistència de dades la tenim assegurada i accessible des del servidor Node al propi servidor o bé a un servidor extern (Modulus o MongoLabs són exemples gratuïts en desenvolupament)
d) Mitjançant Angular es programa i dissenya l'aplicació client (com una aplicació SPA o bé de múltiples pàgines), combinada amb els estàndards ja coneguts: HTML5, CSS3 i JavaScript. Angular ens permetrà implementar, tractar i transformar les dades rebudes en format JSON per part del servidor a dins del navegador del client.

El pas de missatges entre el client i el servidor és asíncron i es realiza mitjançant capçaleres http.

MongoDB

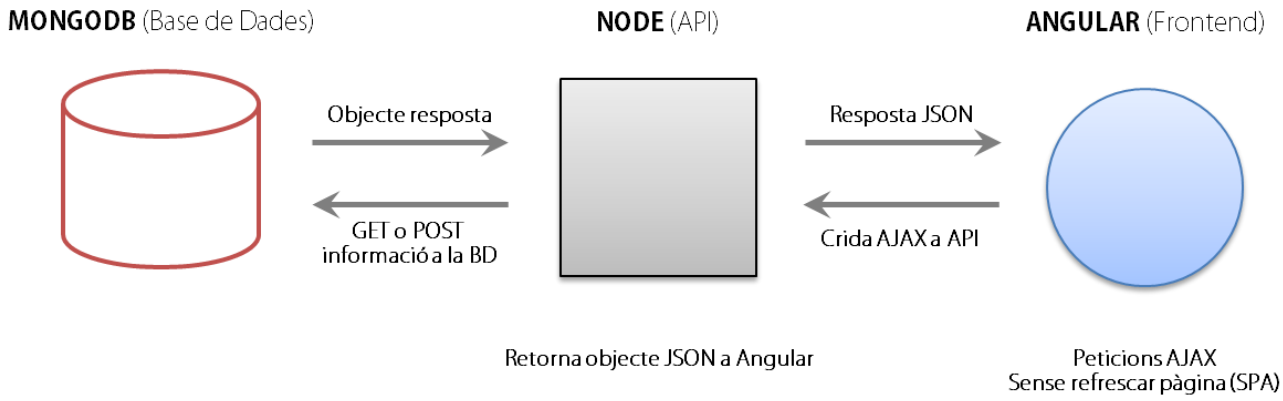
Tot i no ser un dels objectius principals, una de les claus de l'èxit de les aplicacions és la ràpida propagació, així com la sincronització entre diferents usuaris de la mateixa.

MongoDB és la tecnologia que ens permetrà tenir persistència de dades quan estiguem en línia.

Es tracta d'una base de dades basada a document (no és l'habitual base de dades relacional), distribuïda i sincronitzable entre diferents usuaris. Permet la connexió simultània d'usuaris i permet la introducció d'informació sense rigurositat (és a dir, sempre es pot afegir informació en format document sense necessitat d'estar prèviament definida amb un esquema ... tot i ser recomanable!).

És instal·lable localment (per a desenvolupament) o també es pot contractar en servidors distribuïts (el cost dependrà de la quantitat de servidors distribuïts necessaris per a fer les operacions requerides).

L'accés i control a l'esquema de la base de dades es realitza mitjançant, o bé una interfície web (proporcionada pel mateix fabricant en la seva versió de desenvolupament a http://www.mongolabs.com) o bé mitjançant l'intèrpret de comandes.



Express

Express és un framework emprat, principalment, a la cap de vista del nostre model MVC.

Permet formatejar i crear plantilles (vistes) per mostrar a l'usuari les dades que s'extreuen mitjançant els controladors.

L'èxit d'Express rau en que és senzill de fer servir i que engloba un conjunt d'aspectes desconeguts però necessaris (com el Session Handler, el bodyParser, cookieParser, etc.).

Angular

Angular és un framework JavaScript de codi obert mantingut per Google, que permet la realització d'aplicacions tipus SPA.

El seu objectiu és augmentar les aplicacions basades en navegadors amb capacitat MVC, intentant acostar de manera més senzilla el desenvolupament d'aplicacions i que les proves a efectuar siguin més fàcils.

Permet aquesta programació, a través de punts essencials com la dissociació del codi respecte dels elements DOM, o de la generació de comportaments mitjançant directives (marques definibles pel programador similars al les marques HTML).

Un altre punt fort d'aquesta tecnologia és la seva facilitat de lectura, de manteniment i de reutilització de components.

Node.js

Node és un framework escrit en JavaScript altament potent, que permet controlar i crear aplicacions RESTful.

Permet també crear el servidor per a controlar i enrutar les diferents peticions de la nostra aplicació, així com crear un entorn per a la correcta execució de la mateixa aplicació.

Eines utilitzades

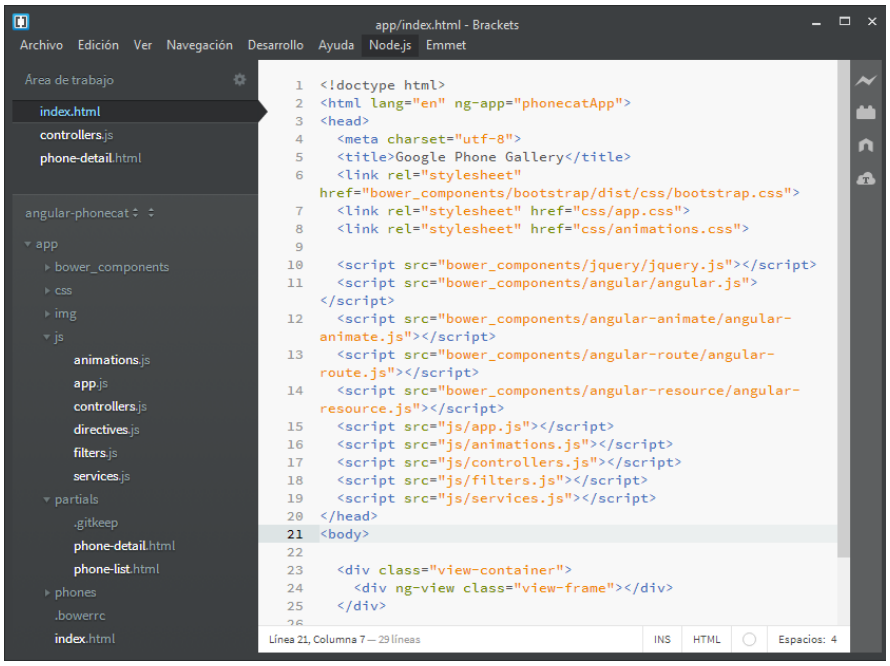
Aptana Studio

Aptana Studio és un IDE de desenvolupament similar a Eclipse, compost per eines de programació de codi obert per a desenvolupar aplicacions.

Permet la creació de projectes (per tenir agrupats tots els fitxers de l'aplicació), així com executar i depurar aplicacions web.

Brackets

És un programa que permet l'edició de fitxers orientat per a programadors de tecnologies web. A més a més incorpora una extensió per a l'execució d'aplicacions i entorns de desenvolupament basat en Node.js.



És molt útil per a poder escriure codi d'alguns fitxers de manera ràpida i controlada.

Git

Repositori de versions per documentar i distribuir les aplicacions realitzades.

L'avantatge principal (a més a més de la possible distribució i documentació dins de la comunitat), és que permet fer *rollbacks* o retrocessos de codi ja implementat.

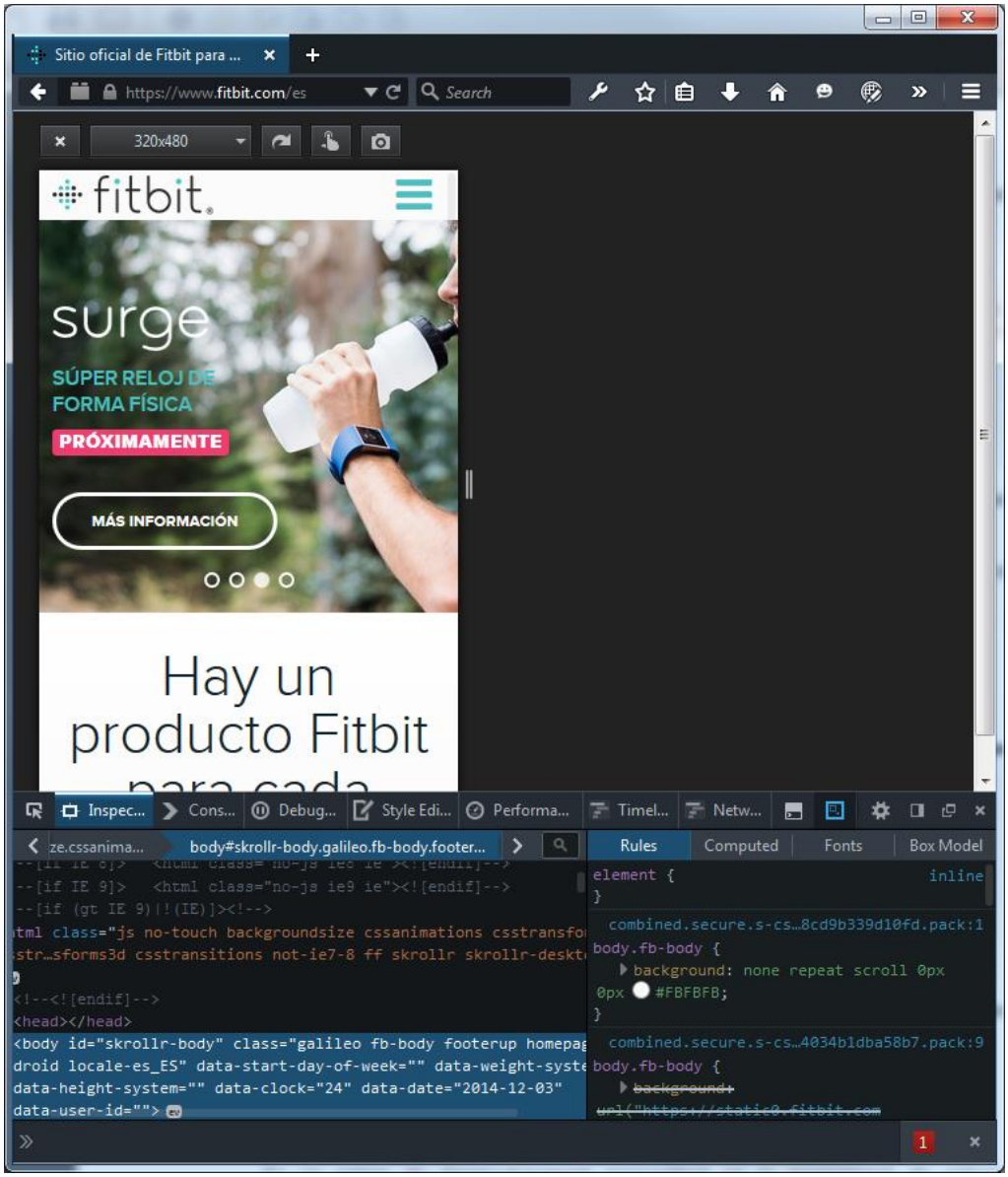
D'aquesta manera, es realitza com una "còpia de seguretat" del codi que s'està desenvolupant, permetent guardar-lo de manera definitiva, o tornar enrere en cas de trobar alguna errada o voler depurar el codi millor.

Mozilla Firefox (for developers)

Navegador web orientat específicament a desenvolupadors d'aplicacions.

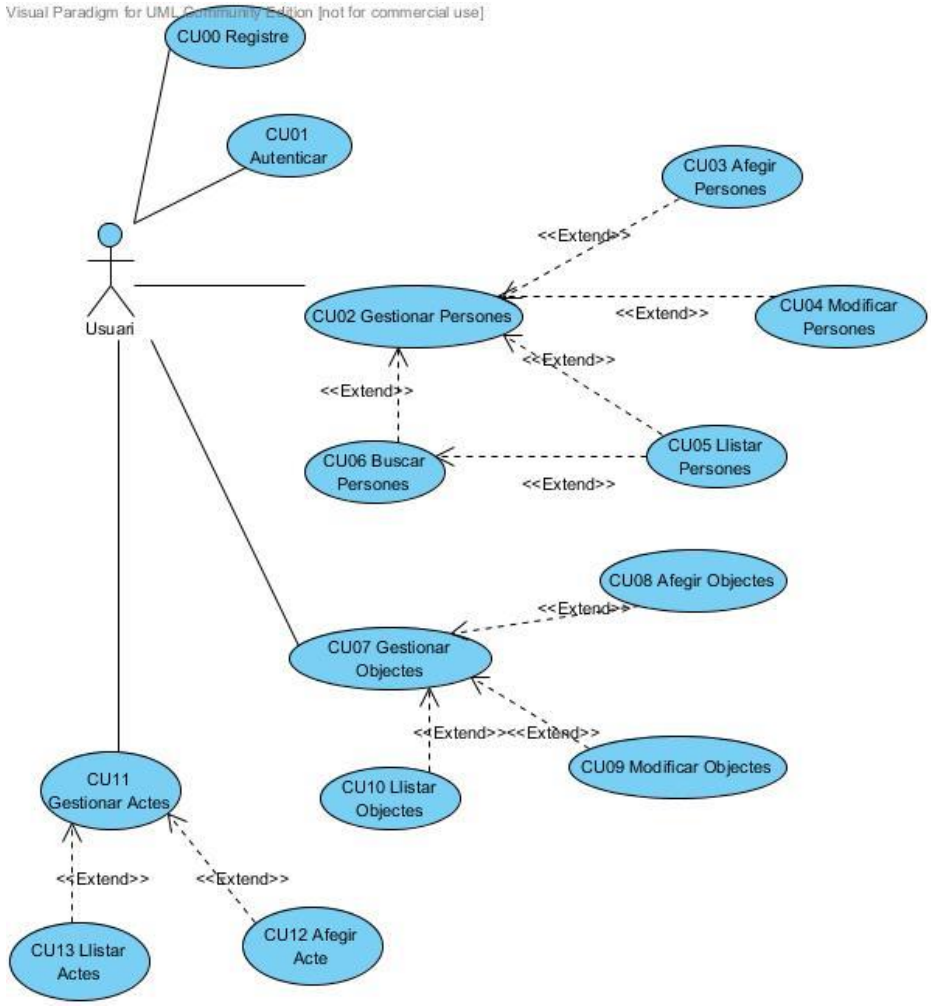
Com a extra fonamental (a més a més de la separació de navegadors per navegar respecte dels emprats per al desenvolupament), conté un conjunt d'eines que ajuden al programador a depurar aplicacions.

També incorpora la possibilitat d'executar emuladors de mòbils sense necessitat d'instal·lar-los per separat, a més a més de poder examinar objectes locals (HTML generat, JavaScript generat, LocalStorage, promitjos i càrrega de xarxa, etc.).



Anàlisis del sistema

Diagrama de casos d'us



Taula resum dels casos d'us

Id	Nom	Actor
CU00	Registre	Usuari
CU01	Autenticar	Usuari
CU02	Gestionar Persones	Usuari
CU03	Afegir Persones	
CU04	Modificar Persones	
CU05	Llistar Persones	
CU06	Buscar Persones	
CU07	Gestionar Objectes	Usuari
CU08	Afegir Objectes	
CU09	Modificar Objectes	

CU10	Llistar Objectes	
CU11	Gestionar Actes	Usuari
CU12	Afegir Acte	
CU13	Llistar Actes	

Descripció textual de casos d'us

CU00 Executar el REGISTRE

Identificador	CU00
Nom	**Registre**
Autor	Victor Corominola Ocaña
Resum	S'usa per a registrar a usuaris dins del sistema (per posteriors autenticacions).
Actor	Usuari
Precondicions	Usuari no ha d'estar donat d'alta al sistema
Postcondicions	Usuari donat d'alta al sistema correctament
Flux normal	1. L'usuari clica per donar-se d'alta. 2. Complimenta un formulari 3. Es finalitza l'enregistrament i es ratifica correu electrònic

CU01 Executar l'AUTENTICACIÓ dins del sistema

Identificador	CU01
Nom	**Autenticar**
Autor	Victor Corominola Ocaña
Resum	S'empra per tal d'autenticar i donar ple accés al sistema als usuaris.
Actor	Usuari
Precondicions	Estar donat enregistrat al sistema
Postcondicions	Donar accés al sistema
Flux normal	1. L'usuari clica per entrar al sistema 2. Complimenta un formulari (usuari i contrasenya) 3. Cas que sigui correcte dona accés al sistema. Cas contrari, torna pas 2.

CU02 Executar la gestió de PERSONES

Identificador	CU02
Nom	**Gestionar Persones**
Autor	Victor Corominola Ocaña
Resum	Punt d'inici per a la gestió de les persones

Actor	Usuari
Precondicions	Que l'usuari s'hagi autenticat correctament
Postcondicions	Cap
Flux normal	L'usuari clica aquesta opció per entrar o no a la gestió de persones

CU03 Afegir PERSONES

Identificador	CU03
Nom	**Afegir Persones**
Autor	Victor Corominola Ocaña
Resum	Emprada per funcionalitats relatives a les persones
Actor	Usuari
Precondicions	Que l'usuari s'hagi autenticat correctament
Postcondicions	S'ha afegit una nova persona o bé s'ha cancel·lat el procés
Flux normal	1. L'usuari clica per crear una nova persona 2. El sistema mostrarà un formulari per introduir les dades 3. L'usuari introdueix les dades i clica a desar/guardar 4. El sistema emmagatzema les dades

CU04 Modificar PERSONES

Identificador	CU04
Nom	**Modificar Persones**
Autor	Victor Corominola Ocaña
Resum	Emprada per funcionalitats relatives a les persones
Actor	Usuari
Precondicions	Usuari autenticat i ha d'haver una persona seleccionada
Postcondicions	Les dades de la persona han estat modificades o bé s'ha cancel·lat el procés
Flux normal	1. S'inicia en el moment que l'usuari té una persona seleccionada i clica a modificar. 2. El sistema mostrarà un formulari per a modificar les dades 3. L'usuari modifica les dades i clica a desar/guardar o bé cancel·la 4. El sistema emmagatzema les dades o cancel·la el procés

CU05 Llistar PERSONES

Identificador	CU05
Nom	**Llistar Persones**
Autor	Victor Corominola Ocaña
Resum	Emprada per funcionalitats relatives a les persones

Actor	Usuari
Precondicions	Que l'usuari s'hagi autenticat correctament
Postcondicions	Es mostren totes les persones
Flux normal	1. Una vegada l'usuari s'ha autenticat es mostren totes les persones de la població actual o de l'usuari.

CU06 Buscar PERSONES

Identificador	CU06
Nom	**Buscar Persones**
Autor	Victor Corominola Ocaña
Resum	Emprada per funcionalitats relatives a les persones (buscar)
Actor	Usuari
Precondicions	Ha d'existir un criteri de cerca
Postcondicions	Filtrat de persones o bé el procés ha estat cancel·lat.
Flux normal	1. El procés de cerca s'inicia quan s'introdueixen les dades del filtre a aplicar. 2. El sistema mostrarà les persones que compleixen els requisits de la cerca.

CU07 Gestionar OBJECTES

Identificador	CU07
Nom	**Gestionar Objectes**
Autor	Victor Corominola Ocaña
Resum	Punt d'inici per a la gestió dels objectes
Actor	Usuari
Precondicions	Que l'usuari s'hagi autenticat correctament
Postcondicions	Cap
Flux normal	L'usuari clica aquesta opció per entrar o no a la gestió dels objectes

CU08 Afegir OBJECTES

Identificador	CU08
Nom	**Afegir Objectes**
Autor	Victor Corominola Ocaña
Resum	Emprada per funcionalitats relatives als objectes
Actor	Usuari
Precondicions	Que l'usuari s'hagi autenticat correctament Que l'usuari té seleccionada una persona
Postcondicions	S'ha afegit un nou objectes o bé s'ha cancel·lat el procés

Flux normal	1. L'usuari clica per crear un nou objecte 2. El sistema mostrarà un formulari per introduir les dades 3. L'usuari introdueix les dades i clica a desar/guardar 4. El sistema emmagatzema les dades

CU09 Modificar OBJECTES

Identificador	CU09
Nom	**Modificar Objectes**
Autor	Victor Corominola Ocaña
Resum	Emprada per funcionalitats relatives als objectes
Actor	Usuari
Precondicions	Usuari autenticat i ha d'haver un objecte seleccionat
Postcondicions	Les dades de l'objecte han estat modificades o bé s'ha cancel·lat el procés
Flux normal	1. S'inicia en el moment que l'usuari té un objecte seleccionat i clica a modificar. 2. El sistema mostrarà un formulari per a modificar les dades 3. L'usuari modifica les dades i clica a desar/guardar o bé cancel·la 4. El sistema emmagatzema les dades o cancel·la el procés

CU10 Llistar OBJECTES

Identificador	CU10
Nom	**Llistar Objectes**
Autor	Victor Corominola Ocaña
Resum	Emprada per funcionalitats relatives als objectes
Actor	Usuari
Precondicions	Que l'usuari s'hagi autenticat correctament
Postcondicions	Es mostren tots els objectes
Flux normal	1. L'usuari selecciona una persona 2. Es mostra un llistat de tots els objectes associats a aquesta persona

CU11 Gestionar ACTES

Identificador	CU11
Nom	**Gestionar Actes**
Autor	Victor Corominola Ocaña
Resum	Punt d'inici per a la gestió dels actes
Actor	Usuari

Precondicions	Que l'usuari s'hagi autenticat correctament
Postcondicions	Cap
Flux normal	L'usuari clica aquesta opció per entrar o no a la gestió dels actes

CU12 Afegir ACTES

Identificador	CU12
Nom	**Afegir Actes**
Autor	Victor Corominola Ocaña
Resum	Emprada per funcionalitats relatives als actes
Actor	Usuari
Precondicions	Que l'usuari s'hagi autenticat correctament Que l'usuari té seleccionada una persona
Postcondicions	S'ha afegit un nou acte o bé s'ha cancel·lat el procés
Flux normal	1. L'usuari clica per crear un nou acte 2. El sistema mostrarà un formulari per introduir les dades 3. L'usuari introdueix les dades i clica a desar/guardar 4. El sistema emmagatzema les dades

CU13 Llistar ACTES

Identificador	CU13
Nom	**Llistar Actes**
Autor	Victor Corominola Ocaña
Resum	Emprada per funcionalitats relatives als actes
Actor	Usuari
Precondicions	Que l'usuari s'hagi autenticat correctament Que l'usuari tingui una persona seleccionada
Postcondicions	Es mostren tots els actes
Flux normal	1. L'usuari selecciona una persona 2. Es mostra un llistat de tots els actes associats a aquesta persona

Arquitectura pel desenvolupament

Una bona arquitectura per a implementar solucions una mica avançades, i que, a més a més poden sofrir variacions ràpidament (i en conseqüència necessiten un desenvolupament àgil[6]), és l'arquitectura MVC.

[6] http://en.wikipedia.org/wiki/Agile_software_development

Aquesta arquitectura ens permetrà separar una mica més, tant en anàlisi com en desenvolupament, el **model** (dades), la **vista** (interfícies d'usuari) i el **controlador** (el codi per a desenvolupar la lògica de negoci).

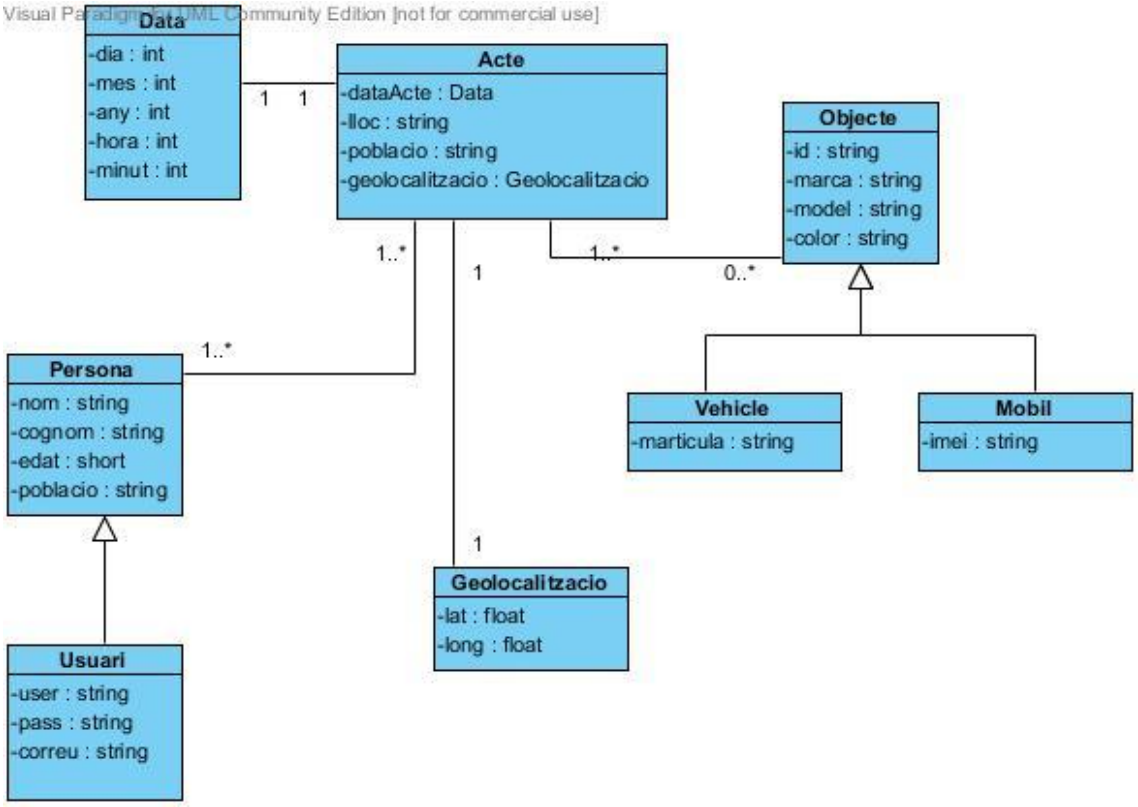
Com a exemple recent d'aquesta manera de fer, tenim la separació entre HTML i CSS (un s'ocupa de les dades –HTML– i l'altre de formatar-les –CSS–).

Model de Domini

El model de domini (també conegut com a model estàtic) representa les principals classes identificades en el domini del problema a solucionar.

Aquest s'emmarca en la capa del Model de la nostra metodologia MVC.

S'ha realitzat una especificació en notació UML:



Com es pot observar, les principals entitats són:

- **Persona:** és el nucli de l'aplicació i representa a les persones (ja siguin els usuaris com les que aquests mateixos inclouen al sistema)
- **Usuari:** hereta de la classe persona, i per les seves característiques, s'afegeixen atributs específics
- **Acte:** representa a tots els actes que les persones hi intervinguin
- **Objecte:** fa referència als objectes que les persones posseeixen, fent especial incidència en
    - **Vehicles**

- **Mòbils**

Com a entitats secundàries trobem:

- **Data:** Especificació formal del tipus de dades
- **Geolocalització:** Especificació formal del tipus de dades

Prototipatge

El prototitpatge s'emmarca en la capa de Vista dins la metodologia MVC. Ens permet veure una primer aproximació (sense codi addicional) de com quedarà l'aplicació.

Es podria dir que, el prototipatge té forces similituds amb el diagrama de casos d'ús.

Es pot veure online, el prototipatge a:

https://www.fluidui.com/editor/live/preview/p_3VVGLuUFSWpRBxsJCbuUCTPEIkIESLhm.1397161840233

També es pot veure online una explicació del prototipatge a:

http://prezi.com/zxww3dumlqju/?utm_campaign=share&utm_medium=copy

Login

Login (inici de l'aplicació)

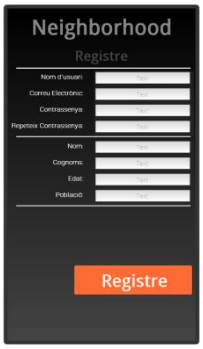
L'aplicació s'inicia amb aquesta pantalla.

L'usuari haurà de:

- Autenticar-se al sistema, o
- Enregistrar-se

Així doncs, haurà de clicar o bé a "**Entrar**" o bé "**Registre**"

Registre d'usuari



Registre d'usuari

Pantalla habilitada per tal que l'usuari s'enregistri.

Una vegada enregistrat, passarà a la pantalla de llistat d'usuaris.

Per a posteriors ocasions no serà necessari que es torni a enregistrar, només s'haurà d'autenticar a la pantalla d'inici.

Els camps (segons el model) són:

- Nom d'usuari
- Correu electrònic
- Contrasenya
- Nom
- Cognoms
- Edat
- Població

Llistat de Persones

Llistat de Persones

Llistat de persones on es mostrarà un llistat de totes aquelles persones que, algú hagi introduït al sistema i que siguin de la mateixa població que l'usuari actual.

A la part inferior hi ha dues opcions:

- Persones
- Actes

Aquestes opcions estan habilitades per tal si l'usuari vol veure altres usuaris o bé "events" o "actes" que hagin ocorregut a la població seleccionada a la parta superior.

Afegir Persones

Afegir Persones

Des d'aquesta pantalla, podrem afegir les persones que vulguem a dins de l'aplicació.

Es deixa a mans de l'usuari la veracitat de les dades introduïdes, tals com:

- Nom
- Cognom
- Edat
- Població

Aquesta última serà determinant per ser mostrada en la població de l'usuari.

Veure persona

Veure una persona en detall

Una vegada es seleccioni una persona, podrem veure amb més detall la seva informació associada, com:

- Dades personals (nom, edat i població)
- Objectes que té (vehicles i mòbils)
- Actes en els que ha participat

Ara, si volem veure un acte, clicarem a sobre

Modificar Persones

Modificar Persones

Una altra opció, si mantenim el botó clicat una estona, és la possibilitat de modificar una persona.

Per tal de fer-ho, caldrà que existeixi al sistema.

La modificació consistirà en introduir les noves dades o les modificacions pertinents als camps de text facilitats.

Denoteu que la pantalla és molt similar a la de "Afegir una persona" (només canvia la funcionalitat i la càrrega de les dades)

Afegir Objectes

Afegir Objectes (telèfon/vehicle)

Des d'aquesta pantalla, l'usuari podrà introduir l'objecte associat a la persona que estava consultant.

Per tal de distingir entre telèfon o vehicle, s'incorpora un botó alternatiu.

Els camps (comuns) proporcionats segons el model establert:

- ID (per posar la matrícula o el IMEI)
- Marca
- Model
- Color

Afegir Actes

Afegir Actes

Des d'aquesta pantalla també es poden afegir els actes relatius a la persona que s'està consultant en aquests moments.

La informació a afegir (segons el model establert) és:

- Data
- Hora
- Acte (breu descripció de l'acte)
- Ubicació (amb la possibilitat d'accedir al GPS)

Implementació

Donat que les eines emprades per a la realització de l'aplicació són eines basades en HTML i JavaScript (el framework MEAN tal i com s'ha detallat anteriorment), les capes de l'aplicació estan implementades dins d'aquest *framework*.

Estructuració de l'aplicació

Per tal de poder implementar l'aplicació, s'ha optat per usar l'estructura arbòria "**estil específic**"[7], donat que aquesta estructura permet escalabilitat de la pròpia aplicació de manera ordenada (és a dir, es genera la base de l'aplicació per a possibles posteriors millores o noves característiques).

S'ha intentat també repartir i separar els arxius corresponent al servidor Node respecte dels arxius que són públics (és a dir separar el frontend del backend).

Així doncs el codi quedaria estructurat de la següent:

```
server.js                -> arxiu d'inici del servidor
package.json             -> arxiu amb la descripció de l'app i dependències
README.md                -> arxiu descriptiu de l'app, instal·lació i ús.
    public/              -> arxius client de l'aplicació
        controllers/     -> directori de controllers (angular.js)
        directives/      -> directori de directives (angular.js)
        filters/         -> directori de filtres (angular.js)
        services/        -> directori de serveis (angular.js)
        stylesheets/     -> directori amb estils (bootstrap v.3) i propis
        vendor/          -> directori amb el ANGULAR.JS (scripts en js)
        views/           -> directori de les vistes parcials
            login.html   -> login view
            ...... .html -> ...... view
            ...... .html -> ......  view
        app.js           -> script d'aplicació principal (main)
        index.html       -> arxiu html principal (main)
        favicon.png      -> arxiu d'icona personalitzat
    server/              -> arxius servidor de l'aplicació
        controllers      -> arxius amb la lògica de servidor (CRUDs)
        models           -> arxius amb el modelatge de les dades
```

Es possible que hi hagi algun altre arxiu (donat que les aplicacions de manteniment de repositori com GitHub inclouen alguns arxius i carpetes ocults per tal de mantenir el codi sincronitzat amb les diferents versions o etapes – *branch* – de desenvolupament).

[7] Segons publicació "**Angular JS Essentials**" de Rodrigo Branas

Model

Per a la creació de la persistència s'han definit les dades per poder ser recuperades ja sigui d'una base de dades basada a documents (MongoDB), com de l'emmagatzematge local (localStorage) que proporciona el HTML5.

Actes

```
// app/models/acte.js
var mongoose = require('mongoose');
var geolocalitzacio = require('usuari.js');
var data = require('data.js');

var acteSchema = mongoose.Schema({
    dataActe            : data,
    lloc                : String,
    poblacio            : String,
    geolocalitzacio     : geolocalitzacio
});

// fem que el model sigui accessible per la nostra app
module.exports = mongoose.model('Acte', acteSchema);
```

Data

```
// app/models/data.js
var mongoose = require('mongoose');


var dataSchema = mongoose.Schema({
    dia         : Number,
    mes         : Number,
    any         : Number,
    hora        : Number,
    minut       : Number
});

// fem que el model sigui accessible per la nostra app
module.exports = mongoose.model('Data', dataSchema);
```

Geolocalització

```
// app/models/geolocalitzacio.js
var mongoose = require('mongoose');


var geolocalitzacioSchema = mongoose.Schema({
    lat         : Number,
    long        : Number
});

// fem que el model sigui accessible per la nostra app
module.exports = mongoose.model('Geolocalitzacio', geolocalitzacioSchema);
```

Persona

```
// app/models/persona.js
var mongoose = require('mongoose');


var personaSchema = mongoose.Schema({
    nom         : String,
    cognom      : String,
    edat        : Number,
    poblacio    : String
});
```

```
// fem que el model sigui accessible per la nostra app
module.exports = mongoose.model('Persona', personaSchema);
```

Objectes

```
// app/models/objecte.js
var mongoose = require('mongoose');


var objecteSchema = mongoose.Schema({
    id          : String,
    marca       : String,
    model       : String,
    color        : String
});

// fem que el model sigui accessible per la nostra app
module.exports = mongoose.model('Objecte', objecteSchema);
```

Mòbil

```
// app/models/mobil.js
var mongoose = require('mongoose');


var mobilSchema = mongoose.Schema({
    imei        : String
});

// fem que el model sigui accessible per la nostra app
module.exports = mongoose.model('Mobil', mobilSchema);
```

Vehicle

```
// app/models/vehicle.js
var mongoose = require('mongoose');


var vehicleSchema = mongoose.Schema({
    matricula        : String
});

// fem que el model sigui accessible per la nostra app
module.exports = mongoose.model('Vehicle', vehicleSchema);
```

Actes

```
// app/models/acte.js
var mongoose = require('mongoose');
var geolocalitzacio = require('usuari.js');
var data = require('data.js');

var acteSchema = mongoose.Schema({
    dataActe            : data,
    lloc                : String,
    poblacio            : String,
    geolocalitzacio     : geolocalitzacio
});

// fem que el model sigui accessible per la nostra app
module.exports = mongoose.model('Acte', acteSchema);
```

Vista

Per a la creació de la vista s'ha tingut en compte que l'aplicació final que es volia era una aplicació autocontinguda en una única pàgina (SPA).

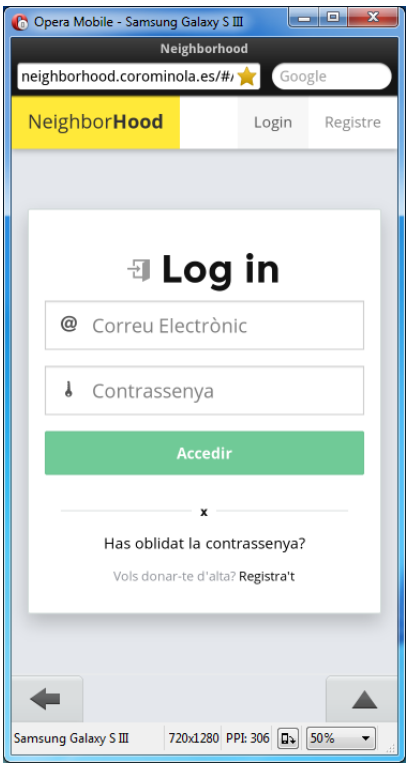
En el moment de la generació, per una qüestió d'usabilitat, s'han emprat altres colors més llegibles (lletra fosca sobre fons clar), enlloc dels que s'havien definit inicialment (lletra clara sobre fons fosc). Aquests correspondrien a la versió v.0 de la vista.

En la última versió de l'aplicació i per tal d'aconseguir que aquesta fos **Responsive**, es va optar per fer una lleugera adaptació amb el CSS de Bootstrap[8] (es presenta l'evolució a la segona columna).

A tenir en compte que aquest CSS permet, si així ens interessa, canviar únicament les plantilles pels colors, tipografies, fons, botons, etc., que vulguem, per tal d'acostar-la al disseny original.

Login

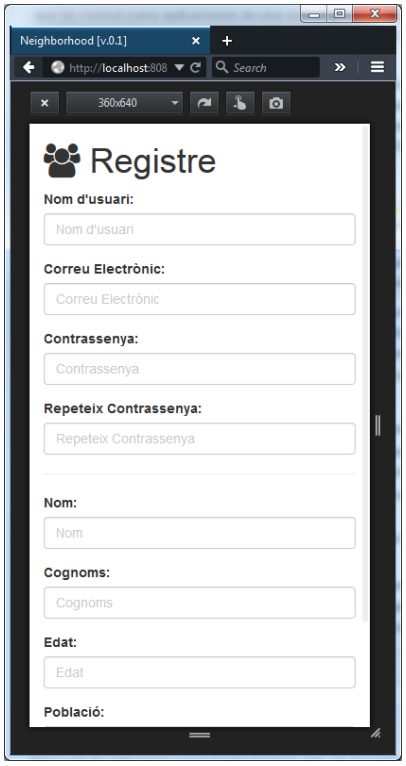
v.0



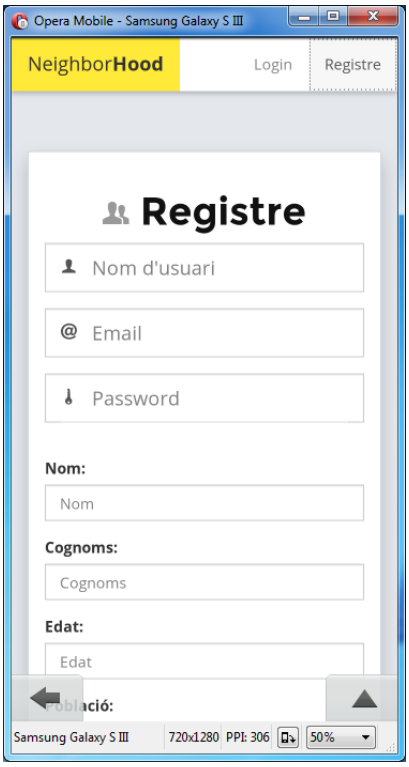
v.1

[8] http://getbootstrap.com/

Registre

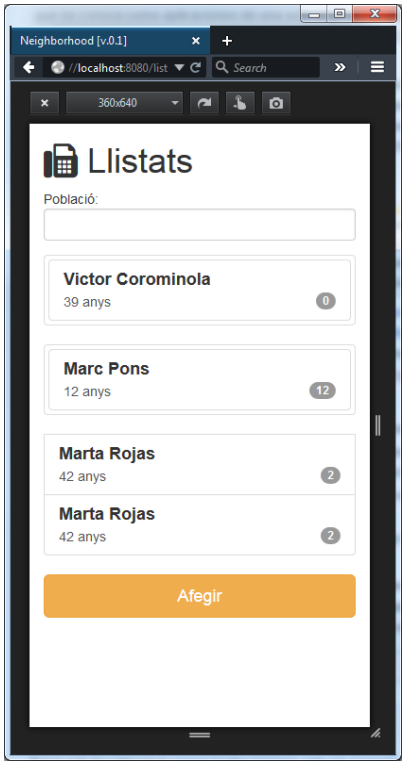


v.0

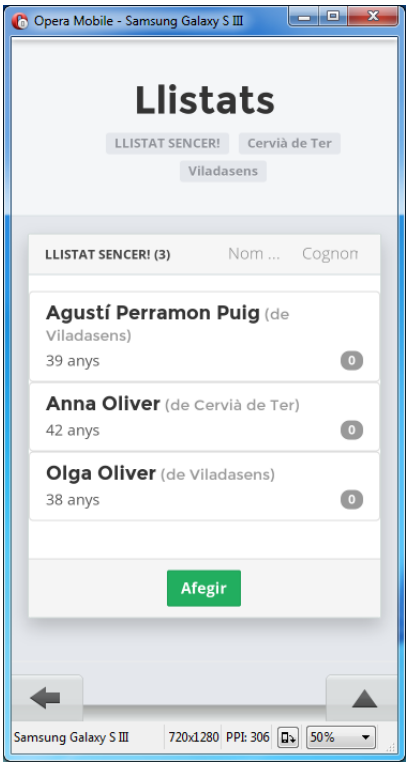


v.1

Llistat de Persones

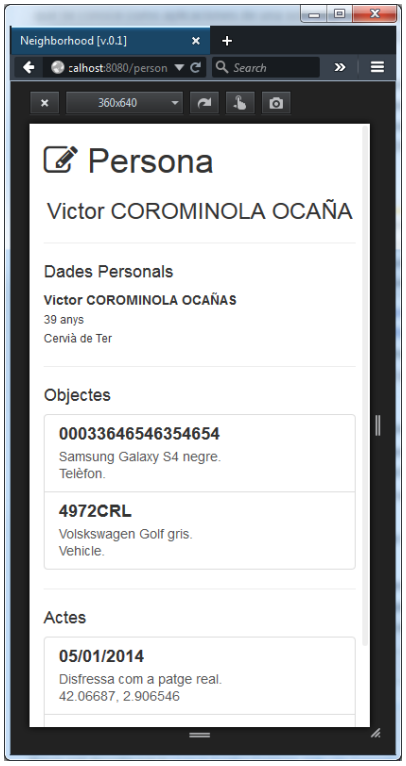


v.0

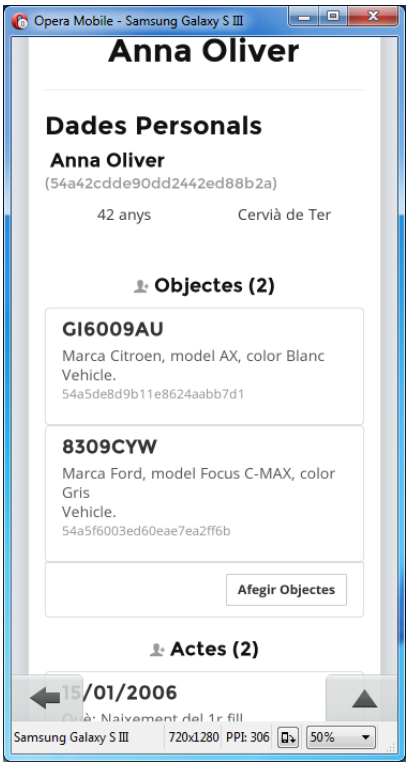


v.1

Detall Persona

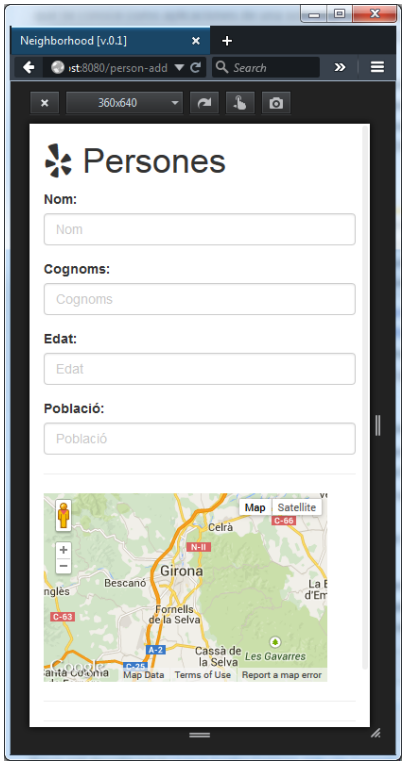


v.0



v.1

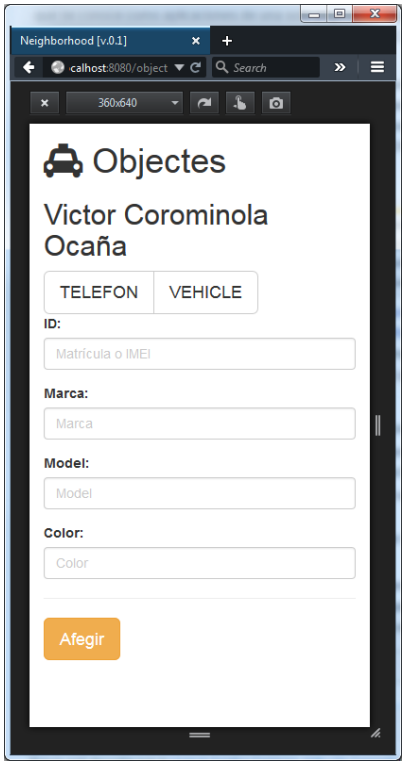
Afegir persones



v.0

v.1

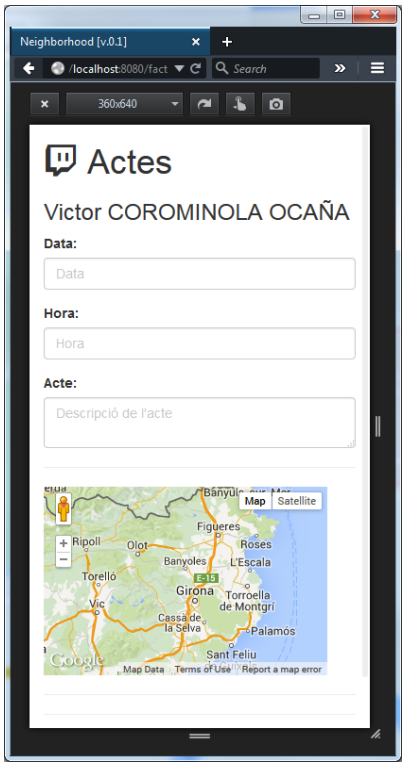
Objectes d'una persona



v.0

v.1

Actes



v.0

v.1

Controlador

Donat que s'ha desenvolupat l'aplicació en la seva totalitat (servidor i client) ha estat necessari separar els diferents controladors (el que gestionen la lògica de l'aplicació al servidor i els que la gestionen al client).

Servidor

En el costat del servidor és necessari implementar la lògica que s'ha d'executar en rebre un verb http (GET, POST, PUT o DELETE).

Donat que es tenien tres entitats (persones, objectes i actes), s'han separat en fitxers diferents (tot i que la funcionalitat de tots tres és molt semblant).

sControllerPerson.js

```
var mongoose = require('mongoose');
var Persona  = mongoose.model('Persona');

exports.findAllPersons = function(req, res) {
    if ((req.query.poblacio) && (req.query.poblacio!='LLISTAT SENCER!')){
        Persona.find( {poblacio: req.query.poblacio}, function(err, persons) {
        if(err) res.send(500, err.message);
            console.log('* GET /persons - població: ' + req.query.poblacio);
            res.status(200).jsonp(persons);
        });
    } else {
        Persona.find(function(err, persons) {
        if(err) res.send(500, err.message);
            console.log('* GET /persons -----');
            res.status(200).jsonp(persons);
        });
    }
};

exports.findById = function(req, res) {
      Persona.findById(req, function(err, person) {
                if (err) return res.sendStatus(500).send(err.message);
                res.status(200).jsonp(person);
      });
};

exports.findPersonaById = function(req, res) {
      Persona.findById(req.params.id, function(err, person) {
                if (err) return console.error(err);
                res.status(200).jsonp(person);
      });
};

exports.addPersona = function(req, res) {
      var persona = new Persona({
                nom:      req.body.nom,
                cognom:   req.body.cognom,
                edat:     req.body.edat,
                poblacio: req.body.poblacio
      });

      persona.save(function(err, persona) {
                if(err) return res.send(500, err.message);
        res.status(200).jsonp(persona);
      });
};
```

```
exports.updatePersona = function(req, res) {
	Persona.findById(req.params.id, function(err, persona) {
		persona.nom		= req.body.nom;
		persona.cognom	= req.body.cognom;
		persona.edat		= req.body.edat;
		persona.poblacio	= req.body.poblacio;

		persona.save(function(err) {
			if(err) return res.send(500, err.message);
	    res.status(200).jsonp(persona);
		});
	});
};

exports.deletePersona = function(req, res) {
	Persona.findById(req.params.id, function(err, persona) {
		persona.remove(function(err) {
			if(err) return res.send(500, err.message);
				res.redirect(200, '/persons');
		})
	});
};

exports.findPoblacions = function(req, res) {
	res.jsonp(Persona.distinct('poblacio'));
};
```

sControllerObject.js

```
var mongoose = require('mongoose');
var Objecte = mongoose.model('Objecte');

exports.getObjecte = function (req, res){
	Objecte.find(
		function(err, objecte) {
			if (err) res.send(err)
				res.json(objecte); });
}

exports.getObjecteById = function (req, res){
	Objecte.findById( req.params.id,
		function(err, objecte) {
			if (err) res.send(err)
				res.status(200).json(objecte);});
}

exports.getObjecteIdPersona = function (req, res){
	Objecte.find({idPerson : req.params.idPersona},
		function(err, objecte) {
			if (err) res.send(err)
				res.json(objecte); });
}

exports.setObjecte = function(req, res) {
	Objecte.create(
	    {idObjecte  : req.body.idObjecte,
	     marca      : req.body.marca,
	     model      : req.body.model,
	     color      : req.body.color,
	     idPerson   : req.body.idPerson
	    },
		function(err, objecte) {
		if (err) res.send(err);
			Objecte.find(function(err, objecte) {
				if (err) res.send(err)
					res.json(objecte); });});
	}
```

```
exports.updateObjecte = function(req, res){
      Objecte.update( {_id : req.params.id},
                {$set:{
                idObjecte  : req.body.idObjecte,
                marca      : req.body.marca,
                model      : req.body.model,
                color      : req.body.color
                }},
                function(err, objecte) {
                        if (err) res.send(err);

       Objecte.find(function(err, objecte) {
                        if (err) res.send(err)
                                res.json(objecte); });});
      }

exports.removeObjecte = function(req, res) {
      Objecte.remove({_id : req.params.id}, function(err, objecte) {
                if (err) res.send(err);
                        Objecte.find(function(err, objecte) {
                                if (err) res.send(err)
                                res.json(objecte); });});
}
```

sControllerActe.js

```
var mongoose = require('mongoose');
var Acte = mongoose.model('Acte');

exports.getActe = function (req, res){
    Acte.find(
        function(err, acte) {
            if (err)
                res.send(err)
            res.json(acte); });
}

exports.getActeById = function (req, res){
    Acte.findById( req.params.id,
        function(err, acte) {
            if (err) res.send(err)
                  res.status(200).json(acte); });
}

exports.getActeIdPersona = function (req, res){
    Acte.find({idPerson : req.params.idPersona},
        function(err, acte) {
            if (err) res.send(err)
                  res.json(acte); });
}


exports.setActe = function(req, res) {
    Acte.create(
        {   data       : Date.parse(req.body.data),
            lloc       : req.body.lloc,
            poblacio   : req.body.poblacio,
            fet        : req.body.fet,
            gLat       : req.body.gLat,
            gLong      : req.body.gLong,
            idPerson   : req.body.idPerson
        },
        function(err, acte) {
            if (err)
                res.send(err);
                  Acte.find(function(err, acte) {
                      if (err) res.send(err)
```

```
                        res.json(acte); });});});
}

exports.updateActe = function(req, res) {
    Acte.findById(req.params.id, function(err, acte) {
        acte.data       = req.body.data;
        acte.lloc       = req.body.lloc;
        acte.poblacio   = req.body.poblacio;
        acte.fet        = req.body.fet;
        acte.gLat       = req.body.gLat;
        acte.gLong      = req.body.gLong;

        acte.save(function(err) {
            if(err) return res.send(500, err.message);
                res.status(200).jsonp(acte); });});
};

exports.removeActe = function(req, res) {
    Acte.remove({_id : req.params.id}, function(err, acte) {
        if (err) res.send(err);
    Acte.find(function(err, acte) {
            if (err) res.send(err)
            res.json(acte); });});
}
```

Client

Del costat del client ha estat necessari crear les peticions dels verbs http que després el servidor Node executarà.

També en el costat del client s'han emprat (mitjançant Angular) àmbits de variables per definir variables i funcions que permetin manipular correctament el contingut dels formularis i parts de pàgina web que mostraven dades (com per exemple el filtre per nom i cognoms de les persones, o l'agrupació de persones segons la seva població).

Aquesta lògica del costat del client s'ha separat per funcionalitat i no per entitats com en el cas del servidor.

main.js

```
angular.module('NeighborHood')
    .controller('MainCtrl', function($scope, $http, $routeParams, Persona) {

        $scope.headingTitle = "Llistat sencer!";
        $scope.persons = Persona.query();

        $scope.poblacions = ['LLISTAT SENCER!'];
        $scope.filtraPerPoblacio = function(poblacio) {
            $scope.persons = Persona.query({ poblacio: poblacio });
            $scope.headingTitle = poblacio;
        };

    });
```

detail.js

```
angular.module('NeighborHood')
```

```
  .controller('DetailPersonCtrl', function($scope, $alert, $http, $routeParams,
$location, $modal, Persona) {

        Persona.get({_id: $routeParams.id}, function(person) {
            $scope.person = person;
        });

        $scope.esborrarPersona = function(person) {
            console.log("detail.js -> esborrarPersona: " + $scope.person._id);
            $http.delete('/api/persons/' + $scope.person._id)
                .success(function(data) {

                    $scope.person = {};
                    $alert({
                        content: 'La persona s\'ha esborrat correctament.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    });
                    $location.url("/persons");
                })
                .error(function(data) {
                    console.log('Error en DETAIL.JS -> esborrarPersona: ' +
data);
                })

        }

        // Codigo para Modal Popup BootStrap
        $scope.formVisibility = false;

        $scope.ShowForm=function() {
            $scope.formVisibility = !$scope.formVisibility;
            console.log("Visibilidad: "+!$scope.formVisibility);
            if($scope.formVisibility){
                // se muestra form
                $scope.nomOrg = $scope.person.nom;
                $scope.cognomOrg = $scope.person.cognom;
                $scope.edatOrg = $scope.person.edat;
                $scope.poblacioOrg = $scope.person.poblacio;
            } else {
                // se esconde form
                $scope.person.nom = $scope.nomOrg;
                $scope.person.cognom = $scope.cognomOrg;
                $scope.person.edat = $scope.edatOrg;
                $scope.person.poblacio = $scope.poblacioOrg;
            }

        }

        $scope.modificarPesona=function(){
            $scope.formVisibility = false;
            $http.put('/api/persons/' + $scope.person._id, $scope.person)
                .success(function(data) {

                    $alert({
                        content: 'La persona s\'ha modificat correctament.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    });
                })
                .error(function(data) {
                    console.log('Error en DETAIL.JS -> modificarPersona: ' +
data);
                })
        }
```

```
        $scope.llistarObjectes = function(){
            $http.get('/api/objectes/' + $routeParams.id )
                .success(function(data) { $scope.objectes = data; })
                .error(function(data) { console.log('Error: ' + data); });
        }

        $scope.llistarActes = function(){
            $http.get('/api/actes/' + $routeParams.id )
                .success(function(data) { $scope.actes = data; })
                .error(function(data) { console.log('Error: ' + data); });
        }

    })

  .controller('DetailObjectCtrl', function($scope, $alert, $http, $routeParams,
$location, $modal, Objecte) {
        console.log('controller - DetailObjectCtrl');
        console.log($routeParams);
        Objecte.get({_id: $routeParams.id}, function(objecte) {
            $scope.newObjecte = objecte;
        });
        $scope.modificarObjecte = function(newObjecte) {
            $http.put('/api/objecte/' + $scope.newObjecte._id,
$scope.newObjecte)
                .success(function(data) {
                    $alert({
                        content: 'L\'objecte s\'ha modificat correctament.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    });
                    $location.url("/objecte/"+$routeParams.id);
                })
                .error(function(data) {
                    console.log('Error: ' + data);
                    $alert({
                        content: 'ERROR: Ha ocorregut un error al modificar.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    });
                });
        };

        $scope.borrarObjecte = function(newObjecte) {
            $http.delete('/api/objecte/' + $scope.newObjecte._id)
                .success(function(data) {
                    $scope.newObjecte = {};
                    $alert({
                        content: 'L\'objecte s\'ha esborrat correctament.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    });

                })
                .error(function(data) {
                    console.log('Error: ' + data);
                    $alert({
                        content: 'ERROR: Ha ocorregut un error a l\'esborrar',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    });});
        };
    })
```

```
  .controller('DetailActeCtrl', function($scope, $alert, $http, $routeParams,
$location, $modal, Acte) {
        Acte.get({_id: $routeParams.id}, function(acte) {
            $scope.newActe = acte;
            $scope.newActe.data = new Date(acte.data);
        });

        $scope.modificarActe = function(newActe) {
            $http.put('/api/acte/' + $scope.newActe._id, $scope.newActe)
                .success(function(data) {
                    $alert({
                        content: 'L\'acte s\'ha modificat correctament.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    });
                    $location.url("/acte/"+$routeParams.id);
                })
                .error(function(data) {
                    console.log('Error: ' + data);
                    $alert({
                        content: 'ERROR: Ha ocorregut un error al modificar.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    });
                });
        };

        $scope.borrarActe = function(newActe) {
            $http.delete('/api/acte/' + $scope.newActe._id)
                .success(function(data) {
                    $scope.newActe = {};
                    $alert({
                        content: 'L\'acte s\'ha esborrat correctament.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    });

                })
                .error(function(data) {
                    console.log('Error: ' + data);
                    $alert({
                        content: 'ERROR: Ha ocorregut un error a l\'esborrar',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    });
                });
        };

  })
;
```

add.js

```
angular.module('NeighborHood')
  .controller('AddCtrl', function($scope, $alert, $http, Persona) {
        $scope.newPersona = {};
        $scope.personas = {};

        $scope.registrarPersona = function() {
            $http.post('/persons', $scope.newPersona)
            .success(function(data) {
                    $scope.newPersona = {};
```

```
                    $scope.personas = data;
                    $alert({
                        content: 'La persona s\'ha afegit correctament.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    })
                })
            .then(function() {
                    $scope.nom = '';
                    $alert({
                        content: 'La persona s\'ha afegit correctament.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    })
                })
            .catch(function(response) {
                $scope.nom = '';
                $alert({
                        content: response.data.message,
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    })
                })
            .error(function(data) {
                console.log('Error: ' + data);
                    $alert({
                        content: response.data.message,
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    })
            })
        };
})

  .controller('AddCtrlObject', function($scope, $alert, $http, $routeParams,
$route, $location, Objecte){
        $scope.newObjecte = {};
        $scope.objectes = {};
        $scope.varIdPersona = $routeParams.id;
        $scope.newObjecte.idPerson = $routeParams.id;
        $scope.registrarObjecte = function() {
            $http.post('/api/objecte/'+$scope.varIdPersona, $scope.newObjecte)
                .success(function(data) {
                    $scope.newObjecte = {}; // Borramos los datos del formulario
                    $scope.objectes = data;
                    $alert({
                        content: 'L\'objecte s\'ha afegit correctament.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    })
                    $location.url("/persons/"+$scope.varIdPersona);
                })
                .error(function(data) {
                    console.log('Error: ' + data);
                    $alert({
                        // content: response.data.message,
                        content: data,
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    })
                });
        };
```

```
    })

  .controller('AddCtrlActe', function($scope, $alert, $http, $routeParams,
$route, $location, Acte){
        $scope.newActe = {};
        $scope.actes = {};
        $scope.date = new Date();

        $scope.varIdPersona = $routeParams.id;
        $scope.newActe.idPerson = $routeParams.id;
        $scope.registrarActe = function() {
            $http.post('/api/acte/'+$scope.varIdPersona, $scope.newActe)
                .success(function(data) {
                    $scope.newActe = {};
                    $scope.actes = data;
                    $alert({
                        content: 'L\'acte s\'ha afegit correctament.',
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    })
                    $location.url("/persons/"+$scope.varIdPersona);
                })
                .error(function(data) {
                    console.log('Error: ' + data);
                    $alert({
                        content: data,
                        animation: 'fadeZoomFadeDown',
                        type: 'material',
                        duration: 3
                    })
                });
        };

        $scope.getGPS = function(){
            navigator.geolocation.getCurrentPosition(function(position) {
                $scope.newActe.gLat = position.coords.latitude;
                $scope.newActe.gLong = position.coords.longitude;
            });}});
```

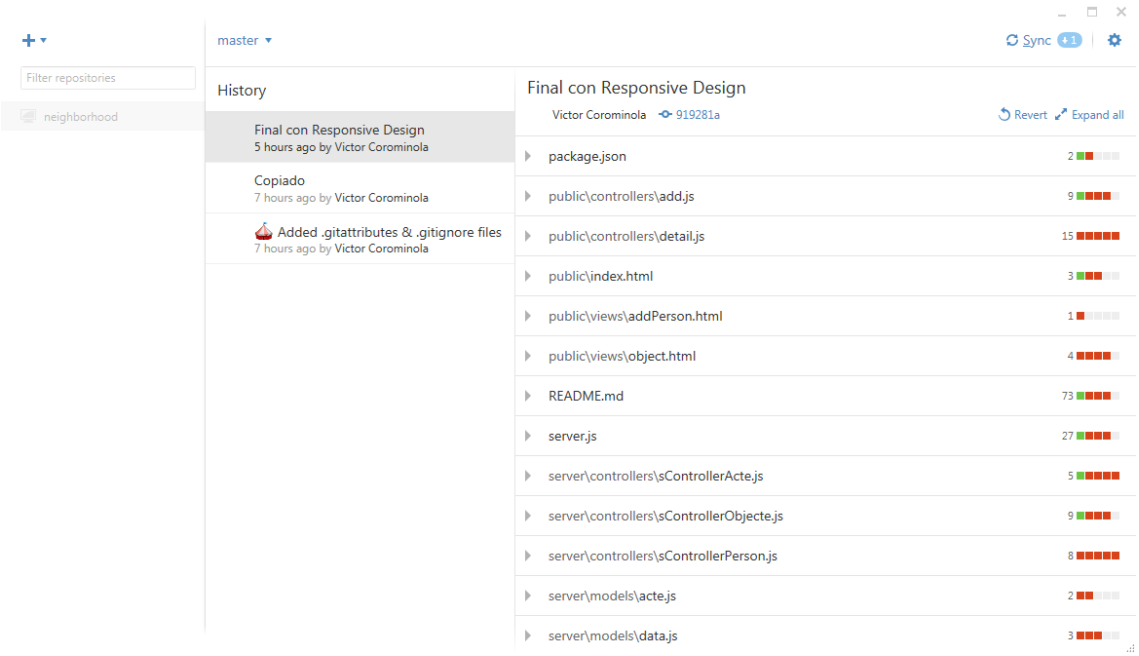
Distribució de l'aplicació

L'aplicació, actualment es troba distribuïda en dos formats: en format de codi i en format de producció.

Repositori

Com s'ha comentat al llarg de la memòria, s'han emprat eines col·laboratives de control de versions, concretament el GitHub.

Aquest repositori, a diferència d'altres, a més a més de la versió de repositori editable online, incorpora una versió mitjançant intèrpret de comandes i una versió en format GUI per Windows:



No s'entrarà a detallar com s'ha realitzat el seguiment del codi i de les seves modificacions, donat que no és l'objectiu d'aquesta memòria.

Tot i així, el repositori on es pot trobar tot el codi d'aquesta aplicació és a:

- https://github.com/viccor/NeighborHood

En el mateix repositori s'explica com preparar el servidor Node, i executar l'aplicació en mode local, si així es desitja[9].

[9] Aquestes instruccions es troben a la descripció del repositori, i a l'arxiu **READ.me** del mateix repositori

Producció

S'ha creat una versió en producció de l'aplicació sota una llicència de desenvolupament. Això significa que no té cap cost per al productor, però que té una caducitat[10].

A dia d'avui (gener 2015), està activa i es pot consultar a través d'alguna d'aquestes adreces:

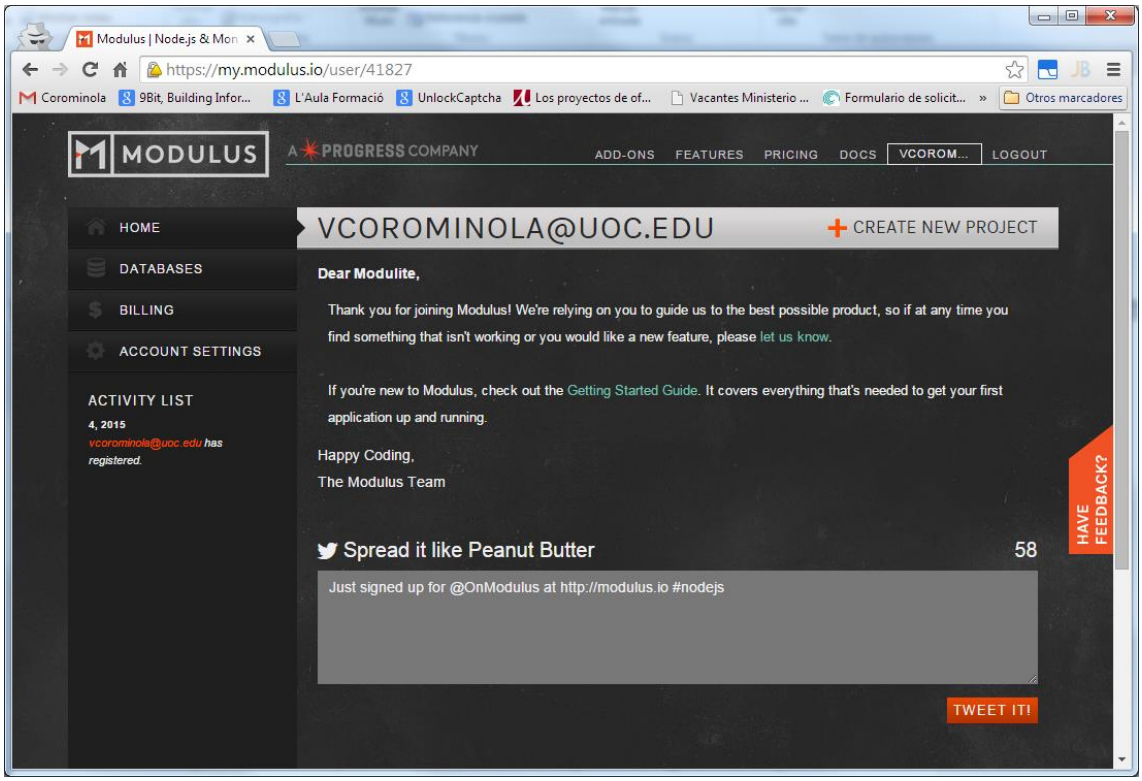
- http://neighborhood.corominola.es
- http://neighborhood.corominola.es:8800

MongoDB

Per a la persistència de dades, s'havia triat un SGDB no SQL i basat en documents com és el MongoDB.

A l'igual que en els anteriors apartats, s'ha triat un servidor gratuït per a programadors, però amb limitacions de mida i de caducitat del maquinari emprat.

El servidor es diu MODULUS, i es pot trobar a https://modulus.io/.

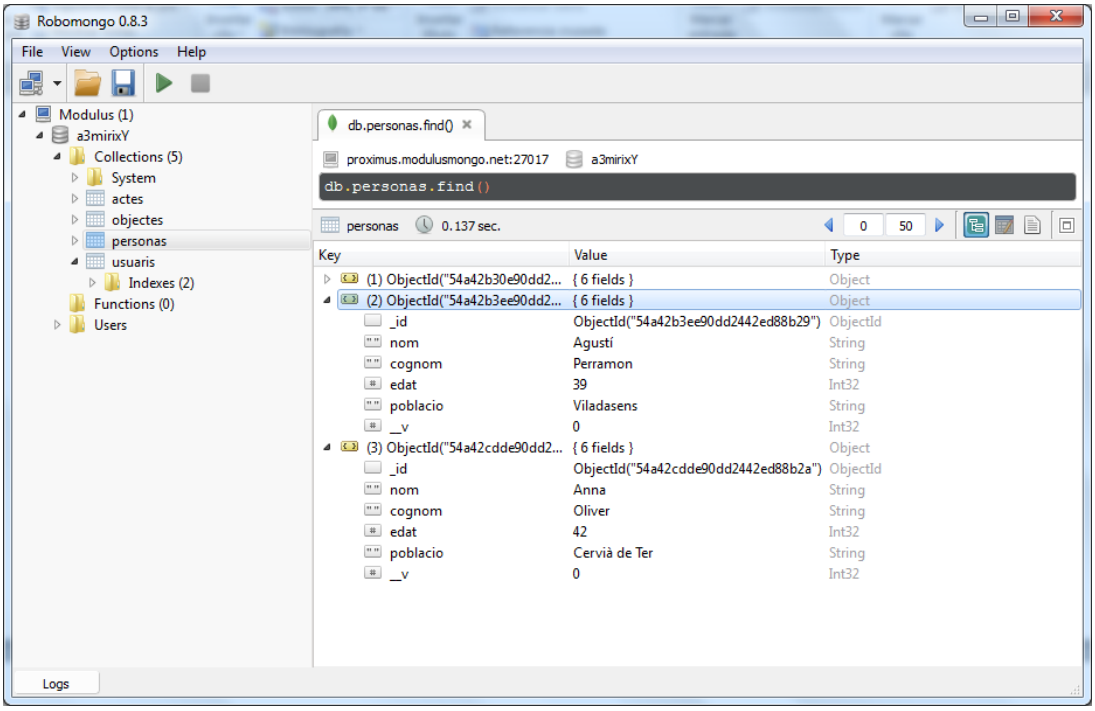


L'interacció amb aquesta base de dades NoSQL, es pot fer mitjançant codi (com és obvi), mitjançant intèrpret de comandes del propi SGBD o bé emprant alguna aplicació amb interfície GUI més amigable.

[10] Depèn de la data que ho mireu no es pot garantir que estigui totalment funcional

Robomongo

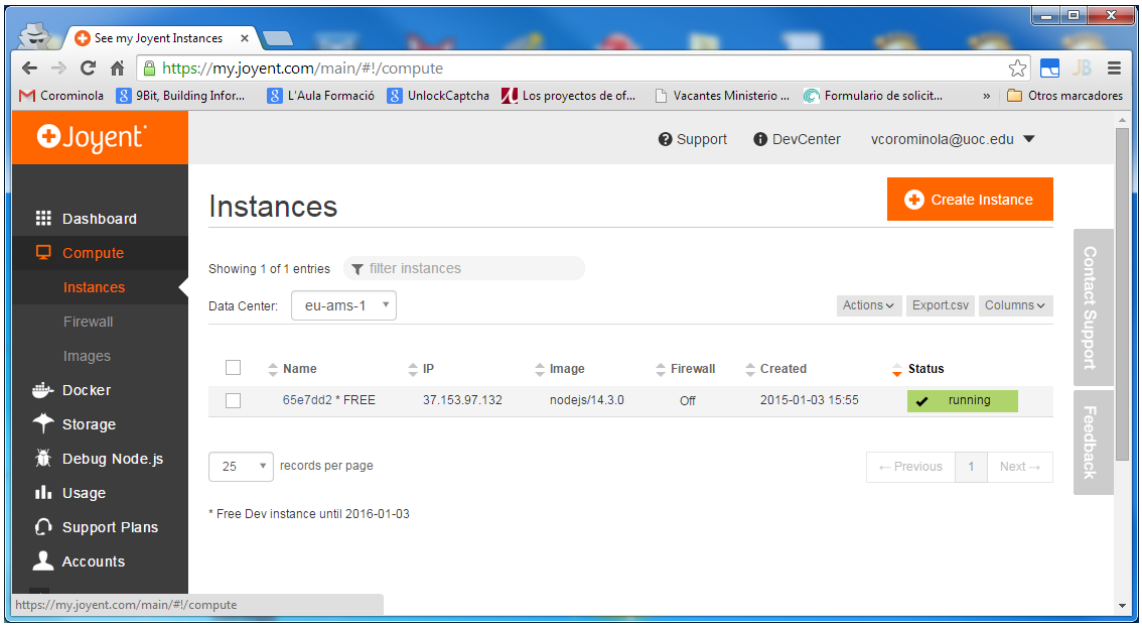
Aquesta aplicació és la que s'ha emprat per accedir i comprovar els CRUDs a la base de dades. És senzilla de configurar (només cal adreça, base de dades i parell usuari/contrasenya) i ràpida en termes d'execució:



Node.JS

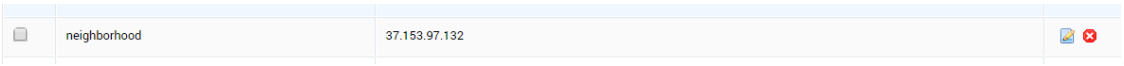
Per a poder executar el servidor Express que s'ha programat en entorn Node, s'ha triat un servidor d'una empresa impulsora d'aquestes noves tecnologies de nom **Joyent.**

Es pot trobar a https://www.joyent.com/, i després d'enregistrar-se, és possible crear un servidor virtual gratuït com a desenvolupador, però amb les mateixes limitacions en el temps que els anteriors.



DNS

Per a la implementació del redireccionament via DNS de l'adreça IP (facilitada per Joyent) cap a la URL http://neighborhood.corominola.es, s'ha emprat el servidor català CDMON, que permet gestionar de manera gratuïta servidors DNS.



Conclusions

Per a un programador autodidacta, acostumat als llenguatges procedimentals, i a l'estructura client-servidor, resulta complicat comprendre el funcionament d'una aplicació RESTful (sobretot per l'asincronia de missatges), o el concepte de les funcions *callbacks* o el pas de funcions com a arguments d'altres funcions.

Però un cop superada aquesta limitació, s'obre un univers amb fronteres indefinides que proporciona una quantitat de possibilitats utilitzant els estàndards (com són HTML, CSS o JS).

Aquesta nova manera de programar i entendre el web (principalment però no exclusivament per a aplicacions mòbils) ha proliferat en una quantitat de frameworks que desorienten, en primera instància a tot aquell que estigui interessat en avançar cap al futur.

Queden subrogades, doncs, les arquitectures síncrones que no permeten creativitat i productivitat a l'usuari.

Personalment crec que els objectius que m'havia plantejat per a aquest treball de final de carrera han estat assolits, tot i que ha quedat pendent l'encapsulament d'aquestes aplicacions en un entorn híbrid a través de PhoneGap o Cordova ...

Millores i passos següents

Per tal de millorar l'aplicació, caldria, tal i com està actualment distribuïda:

1. Actualitzar i posar al dia les condicions d'ús
2. Establir un sistema de recuperació de contrasenya en cas de no recordar-la (actualment no implementat)
3. Establir un tauler de control per a l'usuari (com canvi de contrasenya, de nom, etc.). Segons l'especificació de l'aplicació en aquest TFC, aquest punt no estava inclòs.
4. Incorporar geolocalització visual: actualment es realitza la geolocalització mitjançant la consulta al GPS del sistema, però seria interessant poder-ho fer visualment amb un mapa del Google Maps[11].
5. Promoure l'aplicació per tal que la població la faci servir. Un dels punts forts de l'aplicació és que és auto mantinguda (els propis usuaris l'alimenten d'informació). Per això caldria monetitzar l'aplicació i/o generar accions de màrqueting/difusió.
6. Un altre punt important per tal de millorar l'aplicació seria la fiabilitat de la informació continguda. Caldria buscar un sistema per proporcionar aquesta fiabilitat i veracitat d'informació.

[11] Existeix API anomenada **gmaps.js** per tal d'implementar-ho.

Referències d'Internet

- Local/Web Storage: http://www.html5rocks.com/en/tutorials/offline/storage/?redirect_from_locale=es
- Profiling: http://en.wikipedia.org/wiki/Profiling_%28information_science%29
- RESTful: http://es.wikipedia.org/wiki/Representational_State_Transfer
- CRUD: http://en.wikipedia.org/wiki/Create,_read,_update_and_delete
- http://www.slideshare.net/carlostoxtli/html5-para-moviles-las-reglas-no-escritas
- Compatibilitats: http://mobilehtml5.org/
- Responsive: http://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable
- MEAN: http://meanjs.org/docs.html
- Node: https://github.com/joyent/node/wiki/Node-Hosting

Persistència de dades

- https://www.google.es/?gfe_rd=cr&ei=SfEgU970GIuJ8Qf5y4HICg#q=persistencia+datos+android&start=10
- http://namathis.com/tema.php?tema=programacion-android-persistencia-datos-bases-datos-sqlite
- http://danielggarcia.wordpress.com/2013/11/01/persistencia-en-android/
- http://javiergarbedo.es/index.php/desarrollo-android/88-persistencia-de-los-datos/313-bases-de-datos-en-android
- http://www.quirksmode.org/blog/archives/2009/06/html5_storage_t.html
- http://zbutton.wordpress.com/2010/10/16/html5-y-bases-de-datos-locales/
- http://www.unir.net/postgrado-aplicaciones-moviles.aspx
- http://basesdedatosavanzadas.wikispaces.com/Moviles

Altra documentació

- http://en.wikipedia.org/wiki/Backend_as_a_service
- https://developers.google.com/mobile/build
- http://es.search.yahoo.com/search?ei=UTF-8&fr=chrc-comodo&p=RESTful
- http://www.apuntesdejava.com/2010/11/restful-la-forma-mas-ligera-de-hacer.html
- http://www.dosideas.com/cursos/course/view.php?id=14
- http://www.restapitutorial.com/httpstatuscodes.html
- http://www.html5rocks.com/en/tutorials/offline/storage/?redirect_from_locale=es
- http://www.slideshare.net/carlostoxtli/html5-para-moviles-las-reglas-no-escritas
- http://mobilehtml5.org/
- http://www.mobilebusinessschool.es/#

- http://www.slideshare.net/search/slideshow?searchfrom=header&q=MOBILE+html5
- http://www.androidcurso.com/index.php/tutoriales-android-fundamentos/32-unidad-2-diseno-de-la-interfaz-de-usuario-vistas-y-layouts/112-creacion-de-una-interfaz-de-usuario-usando-xml
- http://donflopez.tumblr.com/post/30737985045/introduccion-a-socket-io

Referències Bibliogràfiques

- **Desarrollo de aplicaciones en la nube para dispositivos móviles**
  AUTOR: Rodger, Richard
  EDITORIAL: ANAYA

- **Javascript y jQuery**
  AUTOR: McFarland, David Sawyer
  EDITORIAL: ANAYA

- **HTML5, CSS3 y JavaScript**
  AUTOR: Rubiales Gómez, Mario
  EDITORIAL: ANAYA

- **Building Node Applications with MongoDB and Backbone**
  AUTOR: Wilson, Mike
  EDITORIAL: O'REILLY

- **AngularJS Essentials**
  AUTOR: Rodrigo Branas
  EDITORIAL: Packt Publishing, 2014

- **Recipes with AngularJS**
  AUTOR: Frederik Dietz
  EDITORIAL: Lean Publishers, 2014

- **MEAN Machine (a beginner's practical guide to the JavaScript stack)**
  AUTOR: Chris Sevilleja i Holly Lloyd
  EDITORIAL: Lean Publishers, 2014