Títol:

Estudi potencialitat API FamilySearch

Autor: Sergi Porras Pagès

Data: 22 de setembre del 2016

Director: Enric Mayol Sarroca

Departament del director: ESSI

Titulació: Enginyeria en Informàtica (2003)

Centre: Facultat Informàtica de Barcelona (FIB)

Universitat: Universitat Politècnica de Catalunya (UPC)
BarcelonaTech

As we express our gratitude, we must never forget that the highest appreciation is not to utter words, but to live by them.

— *John F. Kennedy*

A totes les persones que m'han regalat el seu suport.

Continguts

1 Estudi tècnic de l'aplicació web **7**

1.1 Decisió del tipus d'aplicació a implementar 7

1.2 Tecnologies i patrons de l'aplicació web 8

1.3 Tecnologies pel desenvolupament de l'aplicació 22

Annexos **25**

A Comptes de prova FamilySearch **27**

B Acord d'ús limiat **29**

Secció 1

Estudi tècnic de l'aplicació web

En aquesta secció, s'introduirà quina aplicació pilot ha decidit desenvolupar-se, per mostrar els exemples d'interacció i potencialitat de l'API de FamilySearch, així com les diferents tecnologies que han estat estudiades i utilitzades, pel desenvolupament de l'aplicació.

1.1 Decisió del tipus d'aplicació a implementar

Per poder començar a estudiar el conjunt de tecnologies que el projecte requeriria, necessitàvem saber quina mena d'aplicació seria implementada.

Des del principi, teníem bastant clar, que de disposar de l'oportunitat, intentaríem implementar una pàgina web. Després de realitzar l'estudi inicial de l'API i de les diferents opcions de desenvolupament possibles, crear una pàgina web semblava l'opció més flexible i factible i per tant, vam decidir tirar endavant per aquest camí.

Una pàgina web representava l'opció més factible i flexible, ja que els protocols per integrar-se amb APIs es troben bastant desenvolupats i a més a més, oferia l'oportunitat d'utilitzar tant els SDK oficials, com les diverses eines de desenvolupament, que facilitarien les tasques de creació i interacció amb usuaris.

Per tots aquests motius, a part de la motivació personal d'assolir les habilitats necessàries per desenvolupar una aplicació web, aquesta opció semblava la més adequada.

Així doncs, un cop decidit que s'implementaria una pàgina web, calia fer un reconeixement de les diferents tecnologies disponibles en el mercat i escollir-ne, les més adequades, que poguessin treballar de forma conjunta.

Les tecnologies estudiades poden ser dividides en tres grans blocs: Les tecnologies per la creació d'aplicacions web, les tecnologies de desenvolupament i les tecnologies de desplegament al núvol.

Les tecnologies per la creació d'aplicacions web, representen aquell conjunt de llenguatges, arquitectures i frameworks, que serien utilitzats de cara a la construcció de la pàgina web. Amb altres paraules, el conjunt d'eines i llenguatges que s'utilitzaria per implementar el servidor, les comunicacions entre el servidor i l'API de FamilySearch i el frontal o interfície d'usuari de l'aplicació.

Les tecnologies de desenvolupament, representen el conjunt de tecnologies i eines específiques, que han estat utilitzades per assistir i facilitar, la creació de l'aplicació web.

Finalment, les tecnologies de desplegament, fan referència a l'allotjament web escollit i les tecnologies necessàries per poder completar el desplegament de l'aplicació al núvol.

1.2 Tecnologies i patrons de l'aplicació web

En aquest apartat de la memòria, es descriurà el conjunt de patrons de disseny web i tecnologies, que han estat utilitzades perquè l'aplicació web funcioni i compleixi, amb tots els requisits que ens havíem marcat per ella.

1.2.1 El model: Model Vista Controlador

A l'hora de crear l'aplicació web, volíem crear-la mitjançant una estructura comprensible i eficient, on cada tecnologia realitzes el seu rol principal i deixés aquelles tasques per les quals no havia estat concebuda, a altres tecnologies.

En el món del desenvolupament web, predomina una arquitectura de tres capes que emula el model vista controlador. Però, en què consisteix exactament aquest model? El model vista controlador, també conegut com a MVC[1], és un patró d'arquitectura pensat per la implementació d'aplicacions, que disposen d'una interfície d'usuari.

Com bé indica el nom, el model és compost principalment per tres elements. El Model, la Vista i el Controlador. A continuació, descrivim amb més profunditat el rol de cada un d'aquests components.

El Model és el principal encarregat de gestionar i manipular les dades amb les quals treballa el sistema. El Model, també s'encarrega de crear la lògica i regles, sobre les quals l'aplicació funciona. D'aquesta forma, aquest component serà l'encarregat de gestionar les connexions amb les bases de dades, en cas que aquestes existeixin i crearà els blocs de dades a retornar, de forma que puguin ser compresos pel Controlador.

La Vista o Vistes, representen les representacions visuals de la informació. En

[1]Wikipedia. *Model–view–controller — Wikipedia, The Free Encyclopedia.* [Online; accessed 25-August-2016]. 2016. URL: `https://en.wikipedia.org/w/index.php?title=Model%E2%80%93view%E2%80%93controller&oldid=735649380`.

altres paraules, la interfície que l'usuari veurà i amb la qual podrà interactuar. Mitjançant les interaccions amb aquesta interfície d'usuari, l'usuari és capaç d'indicar a l'aplicació, quines són les accions que vol que realitzi.

Finalment, el Controlador s'encerrega de recollir els diferents inputs enviats per l'usuari, validant-ne l'estat i comunicant-lo a la capa del Model, per obtenir les dades o recursos (entesos com a fitxers servits per la capa Model), demanats per l'usuari. En cas de necessitat, el Controlador també és l'encarregat de modificar la vista o l'estat d'una vista, per reflectir, en tot moment, l'estat actual de l'aplicació a la interfície d'usuari.

La gràcia d'aquest model és que l'usuari només disposa d'accés i permisos d'interacció a la capa de les vistes. Aquestes, comuniquen les accions realitzades per l'usuari al controlador, que a la vegada, s'encarrega de gestionar les comunicacions amb el model i quan aquest retorna dades, transmetre-les un altre cop cap a la vista.

D'aquesta forma, l'ús de la capa model és completament transparent per l'usuari. La figura 1.1, mostra un exemple de com podria funcionar el model MVC en un cas ideal. En la nostra aplicació web, s'ha intentat seguir aquest model en la mesura que ha estat possible.

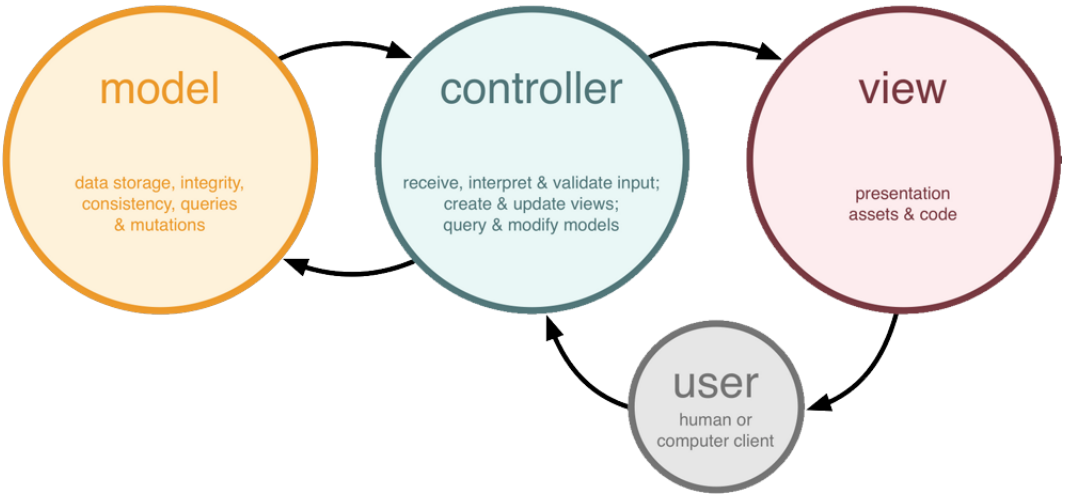


Figura 1.1: Exemple de funcionament del model MVC.

Un últim aspecte important de cara a la terminologia d'aquesta secció, és entendre com relacionem cada una de les capes del patrò MVC, amb els diferents elements que conformen una aplicació web.

Les pàgines web, s'acostumen a poder dividir en dos grans conceptes o elements principals: el front-end i el back-end.

El front-end, fa referència a la capa de presentació o amb altres paraules, el navegador de l'usuari i per tant, aquest element serà associat a la capa Vista del model MVC.

Per altra banda, el back-end és el component que acostuma a realitzar l'accés a

les bases de dades i prepara la informació que ha de ser servida al front-end. Per tant, en el nostre cas, el back-end jugarà el paper de la capa Model, en la nostra aplicació web.

Finalment, el paper del Controlador, sol ser representat en una aplicació web com la part del codi del client (navegador), que l'usuari no veu, ni interactua directament amb ella. En alguns sectors, aquesta part d'una pàgina web es coneix pel nom del 'back-end del front-end'.

La figura 1.2 mostra els tres elements descrits de les aplicacions web i el paper que jugaran, dins del model MVC, cada un d'ells. Durant els següents apartats d'aquesta secció de la memòria, anirem emplenant cada una de les caixes, amb les diferents tecnologies que seran utilitzades.

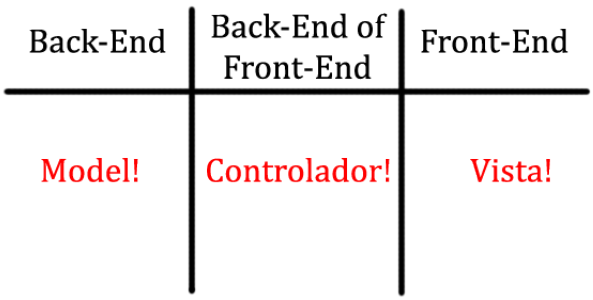


Figura 1.2: Elements de l'aplicació web i el patro MVC

1.2.2 Les tres capes del disseny web (Front-end)

Abans d'entrar en les tecnologies que utilitzarem per crear l'aplicació web, volem destacar una peculiaritat del front-end. En general, el front-end és dividit en tres capes, també conegudes com les capes Estructura, Estil i Comportament[2].

La capa d'estructures és generalment coneguda com la capa del contingut. Aquesta, representa l'estructura sobre la qual el contingut de la pàgina web serà pintat. Es podria entendre també, com la creació de diferents caixes, sobre les quals es pintarà el contingut.

La capa d'estil, defineix com les diferents estructures han de ser situades en el navegador de l'usuari, així com l'estil dels diferents elements que seran pintats en aquestes estructures. Per exemple, controla el color de la font o les imatges de fons.

Finalment, la capa del comportament, s'encarrega de respondre a les diferents accions realitzades per l'usuari i a modificar l'estat de les capes estructures i estil. Aquesta capa del front-end és el que hem anomenat en l'apartat anterior, 'el back end del front end' i com ja hem comentat, jugarà el paper de Controlador en el model

[2] Jennifer Kyrnin. *The three layers of web design.* [Online; accessed 25-August-2016]. 2016. URL: `http://webdesign.about.com/od/intermediatetutorials/a/aa010707.htm`.

MVC.

Ara bé, però per què és important diferenciar aquestes tres capes i perquè les hem reflectit en la memòria? Les raons són diverses:

- **Recursos compartits:** Moltes vegades, certs aspectes de les capes d'estil i comportament, podran ser reutilitzats en diverses pàgines de l'aplicació web i per tant, no té cap sentit haver de duplicar el contingut d'aquests a cada pàgina de l'aplicació, com s'hauria de fer en cas de no voler diferenciar les capes.
- **Descàrregues més ràpides:** Un cop un d'aquests recursos compartits ha estat descarregat, aquest s'emmagatzema en el navegador de l'usuari i ja no cal tornar-lo a descarregar. En cas no voler utilitzar aquesta arquitectura per capes, cada pàgina hauria de descarregar tot el codi necessari per funcionar.
- **Treball en equip:** Permet que diferents desenvolupadors treballin a la vegada, en apartats diferents de la mateixa pàgina de l'aplicació web, sense superposició.
- **User-friendly:** Permet crear aplicacions més fàcils d'interpretar pels motors d'optimització de cerca, així com proporcionar una accessibilitat i compatibilitat amb diferents navegadors, més elevada.

1.2.3 La tecnologia HTML 5

La primera tecnologia que calia estudiar si es volia implementar una pàgina web, era la tecnologia HTML i més concretament, el HTML 5[3].

Aquesta tecnologia és utilitzada per crear l'estructura de les aplicacions web i configurar quina informació s'ensenyarà a cada lloc del navegador. La tecnologia HTML és considerada un llenguatge d'etiquetatge.

El concepte llenguatge d'etiquetatge, significa que els diferents blocs de contingut són envoltats per etiquetes d'obertura i clausura, que atorguen un significat concret al contingut situat a l'interior.

Per exemple, per indicar la creació d'un bloc de contingut ( ), que en el seu interior conté tres paràgrafs (<p>), es podria utilitzar una estructura com la que segueix:

Codi 1.1: Bloc de contingunts HTML amb tres paràgrafs

```
<div>
    <p> ... </p>
    <p> ... </p>
    <p> ... </p>
</div>
```

[3]Wikipedia. *HTML5 — Wikipedia, The Free Encyclopedia.* [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=HTML5&oldid=735803895.

El llenguatge HTML disposa d'un ampli ventall d'etiquetes diferents, que permeten introduir imatges, crear enllaços web, ressaltar el text en negreta o cursiva, etcètera, etcètera.

A més a més, aquest llenguatge crea el que s'anomenen estructures jeràrquiques, on un element pot tenir el rol de pare o fill d'un altre. Per exemple, en l'exemple de codi mostrat, el contenidor és el pare de tots els paràgrafs. El llenguatge HTML permet crear tants nivells jeràrquics com es desitgin.

Així doncs, el llenguatge HTML és utilitzat per crear la capa d'estructures explicada a l'apartat anterior, 'Les tres capes del disseny web' i per tant, podem afirmar que aquesta tecnologia és utilitzada únicament a la capa del front-end de la nostra aplicació web.

1.2.4 La tecnologia CSS

La tecnologia CSS[4], també coneguda com a generació de fulls d'estil en cascada, és un llenguatge utilitzat per explicitar com ha de ser l'aspecte i forma dels diversos elements que apareixen en l'estructura d'una pàgina web o el que és el mateix, en el seu HTML.

El CSS va néixer per poder separar el contingut d'un document, de la presentació d'aquest. El llenguatge CSS permet, entre moltes altres coses, decidir la font i estil de cada element de la pàgina web, l'alineació del text, les separacions entre els diferents components del HTML, els colors o imatges de fons, l'estil dels enllaços web, les transicions entre els diferents estats d'un component HTML, etcètera, etcètera.

Una de les principals característiques del CSS és que al tractar-se d'una fulla d'estils en cascada, un element pot tenir definides diferents regles que marquen el valor d'un cert atribut i el llenguatge CSS és capaç de sabers quina ha d'aplicar, segons la posició de cada regla, dins de l'estructura HTML.

A menys que s'especifiqui de forma contrària, la regla que s'aplica per personalitzar l'atribut d'un element HTML, és la que aplica directament sobre l'element o la heretada del seu pare més proper, que té aquell atribut estilitzat.

D'aquesta forma, s'aconsegueix dotar d'un comportament genèric a certs components HTML i personalitzar només aquells que volem dotar d'atributs diferents.

En resum, la tecnologia CSS permet controlar tots els atributs que fan referència a l'aparença de les diferents estructures i components HTML. Això, ho aconsegueix mitjançant tres classes d'etiquetatge diferents:

- **Etiquetes HTML:** Aquestes etiquetes s'utilitzen per aplicar regles a tots els elements HTML que encaixin amb una etiqueta en concret. Per exemple, si s'utilitza l'etiqueta 'div', es podria decidir des d'aquesta regla la tipografia a

[4]Wikipedia. *Cascading Style Sheets — Wikipedia, The Free Encyclopedia.* [Online; accessed 25-August-2016]. 2016. URL: `https://en.wikipedia.org/w/index.php?title=Cascading_Style_Sheets&oldid=735986518`.

ser utilitzada per tots els contenidors del HTML.

- **Etiquetes de classe:** Aquestes etiquetes són creades pel desenvolupador mitjançant la concatenació del caràcter '.', amb un nom qualsevol (per exemple: '.color-blue'). Com el seu nom indica, les etiquetes de clase permeten assignar regles d'estil a classes concretes. Aquestes classes, poden ser incloses dins dels components HTML, proporcionant així l'estil desitjat a només aquell conjunt de components marcats per l'etiqueta de classe especificada.
- **Etiquetes identificadores:** Aquestes etiquetes són similars a les de classe, però en comptes d'utilitzar el caràcter '.', abans del nom que vol ser introduït, s'utilitza el caràcter '#'. La diferència principal entre les etiquetes de classe i les identificadores, és semàntica. En principi, en un fitxer HTML, hauria d'existir com a màxim una instància de cada etiqueta identificadora, mentre que les etiquetes de classe, poden ser utilitzades en múltiples elements HTML.

Per exemple, podríem incloure l'etiqueta identificadora *first* i l'etiqueta de classe *notícia* a un element HTML, de la següent forma:

 ...

El llenguatge CSS també ofereix la possibilitat de definir regles més complexes, com per exemple, definir un conjunt d'atributs per aquells elements que tinguin un pare específic a l'estructura HTML o que es trobin afectats per esdeveniments especials, com podria ser per exemple, el 'mouseover' (element HTML enfocat pel cursor del ratolí).

Tanmateix, les bases són les mateixes tant pels casos simples com pels casos més complexos i creiem que el text introduït en aquesta secció, hauria de ser suficient de cara a comprendre la funcionalitat d'aquesta tecnologia i fer-se una idea de com funciona.

La tecnologia CSS es veu utilitzada únicament a la capa del front-end de l'aplicació web.

1.2.5 La tecnologia de plantilles Mustache

Abans d'entrar en els detalls d'aquesta tecnologia, cal comprendre que és un llenguatge de plantilles. Els llenguatges de plantilles existeixen principalment per satisfer dos propòsits.

En primer lloc, reutilitzar i minimitzar la quantitat de codi HTML necessari. Els llenguatges de plantilles permeten la incorporació de documents HTML, dins d'altres documents HTML. D'aquesta forma, s'evita la creació de codi redundant o replicat.

Per exemple, si quasi totes les pàgines d'un domini web tenen la mateixa barra de navegació, seria poc eficient haver de replicar l'estructura HTML d'aquesta en totes i cada una de les pàgines. Per evitar-ho, s'utilitzen aquests llenguatges de

plantilles, que permeten definir només un cop l'estructura i utilitzar-la a totes les pàgines desitjades.

Aquests llenguatges també solen permetre la creació de bucles de codi HTML i iterar sobre ells. Imaginem que es volen crear 100 paràgrafs de text en un document HTML, en comptes de crear-los de forma manual, podem crear un bucle amb un llenguatge de plantilles, definir el contingut de només una de les iteracions i deixar que el codi s'encarregui de crear totes les línies de codi HTML necessàries.

El segon propòsit dels llenguatges de plantilla consisteix en dotar de contingut dinàmic als fitxers HTML, a través dels paràmetres servits pel servidor. D'aquesta forma, s'aconsegueix proporcionar contingut personalitzat als documents HTML, abans d'enviar la pàgina cap al client o navegador.

Per la creació de la pàgina web es van estudiar tres llenguatges de plantilles diferents: Mustache[5], Twig[6] i EJS[7].

Mustache era el més simple dels tres i el que dotava al frontal de menys lògica. En el costat oposat, estava Twig, que permetia realitzat tota mena d'operacions en el frontal, fins al punt de permetre la creació de variables dins del HTML. EJS, es trobava en un punt intermedi i mai va acabar de resultar una opció a considerar.

Una de les altres diferències principals entre Mustache i Twig és que Mustache pot ser utilitzat en quasi qualsevol llenguatge de programació, mentre que Twig és específic del llenguatge PHP. Al final, com que es va optar per la utilització d'un servidor Node.js (com s'explicarà més endavant en aquesta secció de la memòria), la possibilitat d'utilitzar el llenguatge Twig quedava descartada i feia de Mustache l'opció escollida.

El fet que Mustache sigui un llenguatge de plantilles sense gaire lògica, no significa que sigui menys complet que els altres. Segueix podent complir amb les dues funcionalitats principals descrites en aquest apartat i només implica que les dades que volen ser utilitzades en el HTML, han de venir estructurades des del servidor.

Les tres operacions principals de Mustache són:

- **Utilitzar un paràmetre del servidor en el HTML:** Per invocar en el HTML un paràmetre del servidor, només cal utilitzar el nom del paràmetre envoltat dels caràcters '{{' i '}}'. Per exemple, *{{parametreServer1}}*.
- **Invocar el HTML d'un altre fitxer:** Això s'aconsegueix mitjançant la inclusió de la següent etiqueta en el codi HTML del fitxer desitjat: *{{>navbar}}*. El codi anterior, importaria el contingut del fitxer *navbar.html*, en el document HTML actual.
- **Iteracions sobre blocs de codi:** Imaginem que el servidor retorna un vector

[5]Mustache. *Mustache, Logic-less template.* [Online; accessed 25-August-2016]. 2016. URL: `https://mustache.github.io/`.

[6]TWIG. *Twig, Template Engine for PHP.* [Online; accessed 25-August-2016]. 2016. URL: `http://twig.sensiolabs.org/`.

[7]EJS. *Embedded Javascript.* [Online; accessed 25-August-2016]. 2016. URL: `http://www.embeddedjs.com/`.

de països, si volguéssim pintar el nom de cada país un en un paràgraf diferent, podríem definir el bucle Mustache de la següent forma:

{{#countries}} {{name}} {{/countries}}.

On {{#countries}} .. *{{/countries}}* representa el bloc de codi HTML a replicar a cada iteració i *{{name}}*, el paràmetre a imprimir de cada país.

Resulta doncs bastant palpable, la utilitat que poden arribar a tenir aquests llenguatges de plantilles i perquè s'ha decidit utilitzar-ne un en el desenvolupament de l'aplicació web.

Aquesta tecnologia pot esdevenir confusa de cara a comprendre en quin lloc de l'aplicació web treballa. Es podria pensar que és una tecnologia del front-end, ja que manipula el HTML, però en realitat, aplica al back-end, modificant el contingut del HTML abans de servir-lo al client.

1.2.6 Les tecnologies Javascript i jQuery

Tot i que aquestes dues tecnologies no són exactament el mateix, volem presentar-les de forma conjunta, ja que són utilitzades d'aquesta forma en el món de les aplicacions web.

Javascript[8] és un llenguatge dinàmic d'alt nivell que s'ha convertit, conjuntament amb el HTML i el CSS, en un dels tres pilars de la programació web. Gairebé totes les pàgines l'utilitzen i és suportat per tots els navegadors moderns. Javascript és un llenguatge de programació molt flexible i permet diferents estils de programació, com pot ser la programació orientada a objectes o estils de programació imperatius i funcionals.

Cal comprendre que Javascript també és utilitzat fora del món web, però aquest representa el seu marcat principal. Tampoc s'ha de confondre aquest llenguatge amb el llenguatge de programació Java. A pesar de les similituds entre els dos, es tracta de dos llenguatges de programació diferents amb desenvolupaments separats.

Per altra banda, jQuery[9] és una llibreria de Javascript plena de funcionalitats, dedicada a la manipulació de documents HTML, CSS i gestió de les comunicacions entre aquests i el servidor de l'aplicació web. Aquesta llibreria, s'ha convertit en un estàndard de la programació web i resulta indispensable de cara a crear aplicacions web funcionals i interactives.

jQuery destaca sobretot per la possibilitat d'executar blocs de codi diferents quan l'usuari interactua de forma específica amb elements del HTML que disposen de certes etiquetes de classe o identificadores, de la mateixa forma que els fitxers

[8] Wikipedia. *JavaScript — Wikipedia, The Free Encyclopedia.* [Online; accessed 25-August-2016]. 2016. URL: `https://en.wikipedia.org/w/index.php?title=JavaScript&oldid=735960528`.

[9] jQuery. *jQuery, Write less do more.* [Online; accessed 25-August-2016]. 2016. URL: `https://jquery.com/`.

CSS utilitzen aquestes etiquetes per definir l'aparença dels components HTML de la web.

Javascript i jQuery són utilitzats de forma conjunta en la capa anomenada 'back-end del front-end' i seran, per tant, els encarregats de realitzar la funció de Controlador en l'arquitectura MVC de la nostra aplicació web.

No volem entrar gaire més en detall de com s'utilitzen aquestes tecnologies, ja que les possibilitats són realment il·limitades i intentar fer un petit manual d'ús, resultaria impossible. No obstant això, per tal que s'entengui una mica més la funcionalitat d'aquestes, imaginem-nos una pàgina web que disposa d'un botó amb identificador *submit*, que un cop pressionat, canvia el valor d'un camp de text del HTML de la pàgina web.

En aquest exemple, s'utilitzaria jQuery per detectar que el botó submit del HTML ha estat pressionat, Javascript per definir una nova variable de text i jQuery per imprimir el contingut d'aquesta nova variable al camp de text del HTML. El codi podria tenir un aspecte similar al que segueix:

Codi 1.2: Example of jQuery and Javascript

```
$('#submit').click(function () {
  var newText = 'next text to display';
  $('#textField').text(newText);
});
```

Tot i ser un exemple molt bàsic, creiem que pot ajudar a comprendre perquè s'utilitzen aquestes tecnologies i com interactuen amb els elements de la pàgina web.

1.2.7 La tecnologia Bootstrap

Bootstrap és el marc de treball o 'framework' més popular pels llenguatges HTML, CSS i Javascript, de cara a desenvolupar aplicacions web de disseny adaptatiu i aplicacions web orientades a dispositius mòbils.

Bootstrap proporciona un conjunt d'estils CSS i codis Javascript que faciliten el desenvolupament de certes funcionalitats per les aplicacions web.

La funcionalitat principal que va portant-se a decidir utilitzar aquesta tecnologia és la funcionalitat del grid o quadrícula. A l'hora de programar una pàgina web que vol ser pintada de forma apropiada tant en dispositius mòbils, com escriptoris, un dels punts més complicats és el d'assegurar que tots els components definits es comportaran com s'espera en tots els dispositius i que ocuparan posicions diferents segons la grandària del dispositiu en què es mostren.

El grid de Bootstrap parteix de la base que tot el contingut web se situarà dins de contenidors, que o bé poden ocupar tot l'ample disponible del dispositiu o fins a una amplada màxima. L'objectiu del grid és partir aquests contenidors en un seguit de files i columnes, on cada casella resultant, pot contenir un bloc de codi diferent.

La imatge 1.3 mostra un exemple de com un contenidor podria estar dividit en

quatre files, on cada fila esta formada per un nombre de columnes diferent.

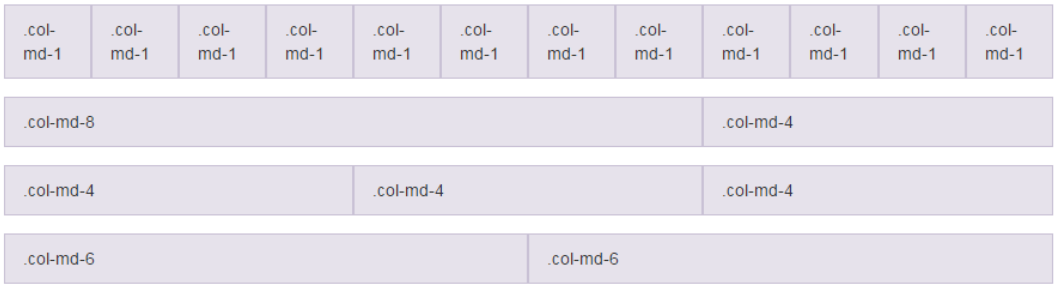


Figura 1.3: Exemble d'un contenidor bootstrap amb 4 files i diferent nombre de files

La gràcia de dividir cada fila en columnes és que en el moment que la grandària de la fila definida, supera la del dispositiu que les vol pintar, les columnes interiors de la fila s'apilen unes sobre les altres de forma automàtica, adaptant d'aquesta forma el contingut a dispositius amb pantalles més petites, sense necessitat de realitzar canvis en el codi o haver d'implementar un codi específic per aquests dispositius.

La imatge 1.4 mostra com una configuració web per escriptori, quedaria reorganitzada en un dispositiu mòbil.

Figura 1.4: Exemple del mateix grid en pantelles Escriptori i Mòbil

A part del grid, que representa la característica principal per la qual es va decidir utilitzar Bootstrap, aquesta entorn de treball també ofereix estils o classes CSS, per taules, botons, formularis, imatges, tipografies, icones, barres de navegació, alertes, barres de progrés, contingut expansible i més funcionalitats.

A pesar que la majoria de classes oferides per Bootstrap han de ser retocades mitjançant CSS propi, per tal d'encaixar i adaptar els diferents elements a la nostra aplicació web, representen un molt bon punt de partida que estalvia temps als desenvolupadors i a la vegada, n'assegura la correcta visualització a través dels diferents dispositius.

Resumint, el marc de treball Bootstrap proporciona principalment fitxers de codi CSS i Javascript que serveixen per complementar els nostres propis fitxers de codi. De forma anàloga a les tecnologies CSS i Javascript, anteriorment descrites, aquest entorn de treball actua tant en el front-end del nostre sistema, com en el 'back-end del front-end'. Per tant, exerceix a la vegada en les capes Vista i Controlador de la nostra aplicació web.

1.2.8 El SDK Javascript oficial de FamilySearch

Una de les tecnologies principals sobre les quals feia falta prendre una decisió al més aviat possible, era la que marcaria com s'haurien de realitzar les comunicacions entre l'aplicació web i l'API de FamilySearch.

La primera opció de la qual disposàvem era realitzar una implementació directa contra l'API. Aquesta aproximació tenia els números de ser la més flexible de totes, ja que permetria realitzar les peticions a l'API de forma completament personalitzada.

Per contrapartida, s'haurien de tractar les respostes XML o JSON de l'API de forma manual, el que acostuma a resultar una tasca repetitiva i tediosa. A part, la codificació de les diferents URI per tal d'accedir als recursos de l'API, també hauria de ser realitzada de forma manual, amb la incertesa afegida de què qualsevol lleuger canvi en l'estructura de l'API, implicaria haver de canviar el codi de la nostra aplicació.

Després d'estudiar les funcionalitats i documentació disponible en el portal de desenvolupadors de FamilySearch, la idea d'utilitzar un SDK per la implementació de l'aplicació web, va anar cobrant força.

Els avantatges principals d'utilitzar un SDK eren el processat automàtic de les respostes de l'API i la utilització de funcions de conveniència, que simplificaven certes tasques i permetien accedir a la informació de forma fàcil i transparent. El desavantatge principal, una reducció en la flexibilitat a l'hora de navegar a través dels recursos de FamilySearch i el fet d'haver d'estudiar el funcionament d'una nova tecnologia.

Després de valorar-ho bastant, es va decidir realitzar la implementació dels exemples d'interacció amb l'API, mitjançant un SDK.

El benefici de tenir un cert grau de seguretat sobre els petits canvis als quals l'API es pogués veure sotmesa, més el fet de no haver de realitzar un tractament manual de les dades, compensava l'esforç extra d'estudiar el funcionament d'un SDK. A part, per tal de demostrar el propòsit i potencial general de l'API, no feia falta ser capaç de controlar totes les interaccions al més mínim detall.

Pel simple fet d'haver decidit implementar una aplicació web, eren tres els diferents SDK que podien ser utilitzats. El SDK de Python va quedar descartat des del començament, ja que no es trobava acabat i tampoc és un SDK oficial. Recordar, que els SDK oficials són desenvolupats per l'organització FamilySearch i per tant, més robusts de cara a possibles canvis en l'API.

La disputa doncs, quedava entre la utilització del SDK de Javascript o el de PHP. Després d'un estudi 'superficial' dels dos, ens vam acabar decantant pel Javascript SDK.

A pesar que la preferència inicial era utilitzar el SDK de PHP, ja que en el passat havia tocat una mica el llenguatge i tenia una idea aproximada de com es podia crear una aplicació web amb PHP, les funcionalitats d'aquest estaven molt per darrere de les que oferia el SDK de Javascript i l'extensa documentació del segon, també garantia una corba d'aprenentatge més fàcil i ràpida.

Tot plegat, va fer que el SDK escollit per implementar els exemples fos el que va ser creat amb el llenguatge Javascript[10]. Els detalls més tècnics d'aquest, així com el seu funcionament bàsic, seran explicats en la següent secció de la memòria, just abans de presentar l'aplicació web i els exemples implementats.

Finalment, esmentar que el SDK de Javascript, ens obria les portes a escollir si el volíem implementar a la capa Model o a la capa Controlador de l'arquitectura MVC, ja que ambdues opcions eren viables.

La implementació més robusta i segura seria implementar-lo en la part del servidor, per tant, en el back-end. No obstant això, a causa del baix grau de dificultat de les operacions que es volien realitzar i la inexistent necessitat d'emmagatzemar informació en bases de dades, es va decidir implementar-lo a la capa del controlador.

D'aquesta forma, es reduïa al mínim la necessitat d'utilitzar la tecnologia Ajax, el que reduïa el nombre de tecnologies a estudiar i facilitava la impressió dels resultats provinents de les peticions a l'API al front-end de l'aplicació.

Un cop finalitzat el projecte i veient-ho amb la perspectiva del coneixement ja obtingut, probablement, l'opció d'implementar el SDK al back-end de l'aplicació, no hauria esdevingut gaire més complicada. Tanmateix, cal recordar que el coneixement inicial a l'hora d'implementar una aplicació web, era pràcticament nul.

1.2.9 La tecnologia Node.js

Una de les poques tecnologies que ens quedaven per escollir, era la que faria funcionar el back-end o servidor de l'aplicació.

Aquest component de l'aplicació web generalment s'encarrega de gestionar les peticions de navegació i servir el contingut demanat als navegadors o usuaris. També s'encarrega de gestionar l'accés a les bases de dades (en cas que existeixin) i processar-

[10]FamilySearch Organization. *FamilySearch Javascript SDK*. [Online; accessed 25-August-2016]. 2016. URL: `http://familysearch.github.io/familysearch-javascript-sdk/2.5/`.

ne la informació, per tal de servir-la d'una forma que la capa Controlador pugui comprendre i utilitzar.

Com que havíem escollit utilitzar el SDK de Javascript per realitzar les connexions a l'API de FamilySearch, es va decidir que era bona idea implementar el back-end amb una tecnologia que pogués fer funcionar el SDK, en cas que al final no volguéssim implementar la lògica de la connexió amb l'API a la capa del controlador.

Després de buscar, l'única tecnologia que complia amb les condicions que buscàvem, era Node.js[11]. Node.js ha estat implementat com un model basat en esdeveniments sense interrupcions, el que el converteix en un sistema en temps d'execució lleuger i eficient.

El paràgraf anterior, ve a significar que Node.js funciona mitjançant esdeveniments i de forma asíncrona, el que permet que moltes connexions siguin tractades de forma simultània. En el moment que una petició conclou, es dispara un esdeveniment de finalització, que s'encarrega d'activar una rutina que decidirà quina és la següent acció a realitzar. Aquest aspecte fa de Node.js, una tecnologia molt escalable.

A més a més, el fet de tractar-se d'un software de codi obert que ha rebut una gran acceptació, ha propiciat que molts desenvolupadors s'hagin dedicat a crear paquets de software que n'amplien les funcionalitats inicials. Totes aquestes extensions, poden ser trobades al repositori de paquets NPM, el més gran del món en quant a llibreries de codi obert.

Comentarem més sobre aquesta tecnologia i com funciona, en els apartats específics d'implementació que apareixeran més endavant en aquesta memòria.

Aquesta tecnologia realitza el paper del Model en l'arquitectura MVC de la nostra aplicació o el que és el mateix, en el back-end d'aquesta.

1.2.10 La tecnologia Express o Express js

De la mateixa forma que descrivíem a Bootstrap com un marc de treball per les capes vista i controlador, Express[12] és un marc de treball pel servidor de l'aplicació i en el nostre cas concret, per la tecnologia Node.js.

Així doncs, Express és un marc de treball minimalista i flexible pensat per crear aplicacions web i aplicacions mòbil simples, mitjançant Node.js. Express també està pensat per la creació d'APIs robustes.

Diem que Express és minimalista, perquè mitjançant l'aplicació d'una fina capa d'eines destinades al desenvolupament web i situada per sobre de la tecnologia Node.js, s'aconsegueix una gran potencialitat sense la necessitat d'emmascarar o alterar les funcionalitats principals i simples per les quals Node.js destaca.

[11] NodeJS. *NodeJS, Javascript Engine.* [Online; accessed 25-August-2016]. 2016. URL: `https://nodejs.org/en/`.

[12] ExpressJS. *ExpressJS, Node Framework.* [Online; accessed 25-August-2016]. 2016. URL: `https://expressjs.com/`.

Molts altres marcs de treball o frameworks populars que han desitjat ampliar les possibilitats de Node.js, han nascut a partir de la base d'Express.

Podem justificar la utilització d'aquest entorn de treball per la nostra aplicació, ja que els requisits que tenim pel back-end, després de decidir implementar la interacció amb l'API de FamilySearch a la capa del controlador, són relativament simples.

Les funcionalitats o eines principals que proporciona Express són:

- **Eines d'encaminament:** L'encaminament recull el conjunt d'accions i processos que tenen lloc des que el servidor web rep la petició d'accés a una URL, fins que respon a la petició amb els recursos adequats.
- **Eines middleware:** El concepte middleware serveix per referir-nos a aquelles peces de software que fan de pont entre dues parts o components d'un sistema. En el marc de treball en el qual ens estem referint, serveix per capturar les peticions al servidor per part de l'usuari i abans de servir-ne la resposta, realitzar algunes accions o comprovacions. Un exemple típic podria ser el de comprovar que un usuari té els permisos suficients per veure la pàgina demanada.
- **Eines per la utilització de llenguatges plantilla:** Aquest conjunt d'eines són les que ens permeten configurar el servidor Node.js per comprendre i poder utilitzar, per exemple, el llenguatge de plantilles Mustache que hem descrit en aquest mateix apartat de la memòria.
- **Eines per gestionar errors:** Eines específiques per gestionar els errors com a middleware.
- **Eines de depuració:** Eines destinades a ajudar als desenvolupadors durant els moments de desenvolupament a detectar errors mitjançant, en gran part, la consola.
- **Eines de connexió a bases de dades:** Conjunt d'eines per integrar de forma fàcil diferents bases de dades amb el servidor.

Donada la naturalesa de la nostra aplicació, les funcionalitats que ens interessen més són les tres primeres. Aquesta tecnologia, al tractar-se d'un framework de Node.js, s'utilitza, evidentment, en la part del back-end de la nostra aplicació.

1.2.11 Recapitulació de les tecnologies pel desenvolupament web

Els apartats anteriors d'aquesta secció de la memòria s'han utilitzat per explicar i justificar les diferents tecnologies que formen part de la nostra aplicació web.

En total, es tracta de nou tecnologies que han hagut de ser estudiades per separat i aprés a utilitzar de forma conjunta. L'única tecnologia sobre la qual es disposava en certa forma, d'un petit coneixement previ, era la tecnologia Javascript.

Es va dedicar una part important del temps destinat al projecte a aprendre a utilitzar tot aquest conjunt de tecnologies, però creiem que el resultat ha estat bastant satisfactori.

El mapa de tecnologies i el paper que juga cada una d'aquestes en l'arquitectura de capes proposada, es pot veure representat a la figura 1.5.

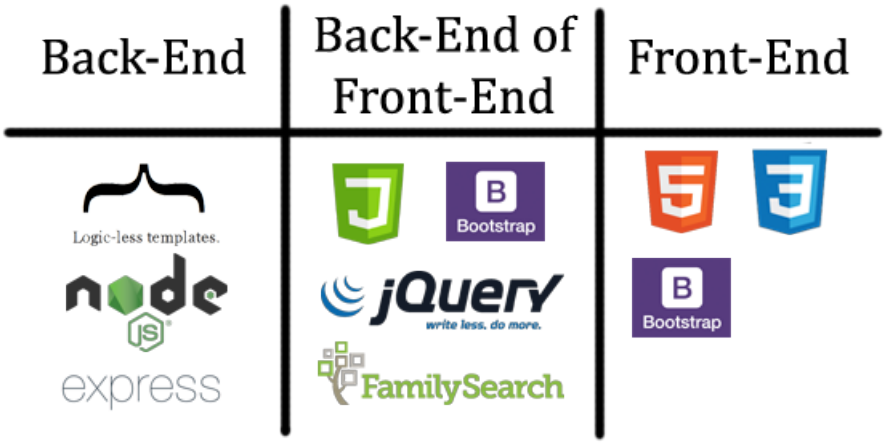


Figura 1.5: Mapa final de tecnologies en cada capa de l'aplicació

1.3 Tecnologies pel desenvolupament de l'aplicació

Apart de les tecnologies que permeten la creació i funcionament de l'aplicació web, hi ha un altre conjunt de tecnologies o eines que s'han utilitzat per facilitar el desenvolupament d'aquesta.

1.3.1 La tecnologia GitHub

A causa de les complicacions resultants de compaginar una jornada laboral a temps complet amb l'elaboració del projecte, el desenvolupament d'aquest ha estat realitzat des de diverses localitzacions diferents. En concret, han estat tres els principals ordinadors o localitzacions des de les que s'ha treballat: Casa, Portàtil i oficina de l'empresa.

Per tal de facilitar la sincronització d'arxius entre les diferents estacions de treball, es va decidir utilitzar la tecnologia GitHub[13].

GitHub és un repositori Git hostejat al núvol. Ofereix totes les funcionalitats pròpies de Git destinades al control de revisions distribuïdes i control del codi font, afegint-ne algunes de noves.

Un cop creat el repositori on tot el projecte queda emmagatzemat, els servidors de GitHub s'encarreguen de què aquest sigui disponible des de tot arreu. D'aquesta

[13] GitHub. *GitHub help documentation.* [Online; accessed 25-August-2016]. 2016. URL: `https://help.github.com/`.

forma, cada estació de treball diferent pot descarregar-se el codi per primera vegada, mitjançant la instrucció:

> git clone https://github.com/sinh15/pfc-family-search.git pfc-family-search

És important evitar treballar en dues estacions de treball diferents, a menys que entre sessions de treball s'actualitzin els canvis en el repositori emmagatzemat al núvol, per tal d'evitar superposicions i conflictes entre les versions dels arxius.

GitHub disposa d'eines per fer front a aquestes situacions, ja que en realitat és una eina pensada per treballar en equip sobre els mateixos arxius de codi. No obstant això, com que aquest projecte ha estat realitzat per una sola persona, s'ha prescindit de la utilització de branques i tots els canvis s'han aplicat sobre la branca mestra del projecte.

Un cop s'acaba una sessió de feina, independentment de l'estació de treball, es poden pujar els canvis al núvol mitjançant tres simples instruccions:

> git add .
> git commit -m "sessió de treball X finalitzada"
> git push origin master

De la mateixa forma, abans de començar a treballar des de qualsevol estació de treball, podem recuperar l'últim estat del projecte mitjançant la instrucció:

> git pull origin master

A part dels beneficis que aquesta eina ens proporciona de cara a treballar de forma distribuïda i en diferents entorns, serveix al mateix temps de 'backup' o reserva, en cas que alguna de les oficines de treball quedi malmesa o es vulgui recuperar una versió antiga d'algun fitxer del codi.

1.3.2 Node.js i Express en l'àmbit local

Durant el desenvolupament, per tal de facilitar les proves sobre l'aplicació i no haver de realitzar un desplegament al núvol per cada canvi que es volgués comprovar, aquesta s'ha programat en un entorn local.

Aquest fet implica que el nostre sistema operatiu feia de servidor per l'aplicació web i aquesta només resultava accessible a través de l'URL 'http://localhost:8080'.

Per tant, el nostre sistema local necessitava ser capaç d'emular les tecnologies que formarien part del servidor. A recordar, Node.js i Express.

No entrarem a descriure la tecnologia Node.js ni Express un altre cop, ja que ja ho hem fet en apartats anteriors. El que sí que volíem fer, era exposar que aquestes tecnologies havien estat instal·lades en l'àmbit local per tal de fer funcionar l'aplicació des dels ordinadors de desenvolupament.

1.3.3 Tecnologia NPM

Com s'ha indicat en un apartat anterior, NPM[14] és un contenidor de paquets orientats a la plataforma Node.js. Aquesta tecnologia pot ser instal·lada en l'àmbit local i d'aquesta forma, descarregar extensions per les aplicacions Node.js que volem provar.

Per instal·lar un conjunt de paquets, podem fer-ho introduint la instrucció: *npm install*, que s'encarrega d'instal·lar en l'àmbit local, totes aquelles dependències que hagin estat declarades en el servidor de l'aplicació web o introduir la instrucció: *npm install 'package name'*, per instal·lar un paquet en concret.

1.3.4 Paquet Nodemon

Un paquet que volem destacar, instal·lat a través del repositori NPM i que només ha estat utilitzat en l'entorn de desenvolupament local i no desplegat al núvol, s'anomena Nodemon.

Nodemon observa els canvis realitzats en els fitxers de codi de la nostra aplicació i en cas que algun canviï, aquest reinicia el servidor que fa córrer l'aplicació de forma automàtica, per tant, sense necessitat de reiniciar-lo de forma manual i poder veure així els canvis a l'entorn local de forma immediata.

Pot semblar un paquet que aporta poca funcionalitat, però quan et trobes implementat una pàgina web, generalment realitzes molts petits canvis en el codi dels quals vols observar-ne l'afecte abans de continuar programant. Per fer-ho, faria falta reiniciar manualment el servidor, però aquest paquet, ens estalvia aquesta tasca.

El fet de ser un paquet que ha estalviat bastant de temps a l'hora de realitzar petites proves i observar-ne els canvis, volíem que trobes la seva representació dins de la memòria.

[14] NPM. *NPM, Package Manager for Javascript.* [Online; accessed 25-August-2016]. 2016. URL: `https://www.npmjs.com/`.

Annexos

Annex A

Comptes de prova FamilySearch

Recordem que l'aplicació desenvolupada pot ser trobada sota el següent URL:

https://pfc-fs-api-potential.herokuapp.com/

I que el codi de l'aplicació pot ser trobat en la seva totalitat, en el repositori GitHub:

https://github.com/sinh15/pfc-family-search

Recordades les URL als principals recursos del projecte, s'han creat tres comptes de prova a la pàgina de FamilySearch, pels diferents membres del tribunal. D'aquesta forma, s'estalvia als membres la necessitat de passar a través del procés de registre i activació, però en cas de preferir-ho, sempre poden crear-se un compte a la pàgina oficial de FamilySearch i utilitzar-lo en l'aplicació web desenvolupada en aquest projecte.

Per la creació d'aquests comptes, no s'ha utilitzat cap informació personal dels membres del tribunal, les persones creades han estat marcades com a privades (no són accessibles) i només s'ha utilitzat una combinació de la primera lletra del nom i el cognom, per crear un nom d'usuari de fàcil utilització.

Per tal d'obtenir els comptes de prova creats pel president, vocal i secretari del tribunal, contacteu amb l'Enric Mayol, tutor del projecte o amb mi mateix a través del correu electrònic: *sporras89@gmail.com*.

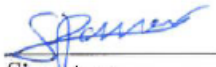
Annex B

Acord d'ús limiat

Limited App Key Production Use Agreement

I agree to not distribute the below product until I have received written or email notification that the feature(s) applied for has been certified. Until receiving that notification, I will only use the App Key on development and testing systems own and operated by myself, my company, and a limited number (100 or less) of selected beta testers identifiable by at least name, email, and phone number.

FamilySearch User Name:	Sergi Porras
Company Name:	Facultat Informàtica de Barcelona (FIB)
Product Name:	PFC-FS-API-Potential
Platform Type:	WEB Application
Version:	1.00
FamilySearch Sign-in Name:	sinH15



Signature

SERGI PORRAS PAGÈS
Print Name

Data Analyst
Position

29/08/2016
Date

Bibliografia

[1] Bootstrap. *Bootstrap, HTML CSS JS framework.* [Online; accessed 25-August-2016]. 2016. URL: https://jquery.com/.

[2] Population Reference Bureau. *2014 World Population Highlights.* [Online; accessed 25-August-2016]. 2014. URL: https://www.youtube.com/watch?v=h5zplKrc9-0.

[3] Population Reference Bureau. *Distilled Demographics: How Many People Have Ever Lived on Earth?* [Online; accessed 25-August-2016]. 2011. URL: https://www.youtube.com/watch?v=7WTctr5kviA.

[4] Cookie Session Collective. *Cookie Session Documentation.* [Online; accessed 25-August-2016]. 2016. URL: https://github.com/expressjs/cookie-session/.

[5] Chris Coyier. *Design Considerations: Text on images.* [Online; accessed 25-August-2016]. 2014. URL: https://css-tricks.com/design-considerations-text-images/.

[6] Chris Coyier. *Smooth Scrolling.* [Online; accessed 25-August-2016]. 2016. URL: https://css-tricks.com/snippets/jquery/smooth-scrolling/.

[7] Agència Espanyola de protecció de dades. *Reglament de la LOPD.* [Online; accessed 25-August-2016]. 2016. URL: https://www.agpd.es/portalwebAGPD/canaldocumentacion/informes_juridicos/reglamento_lopd/index-ides-idphp.php.

[8] EJS. *Embedded Javascript.* [Online; accessed 25-August-2016]. 2016. URL: http://www.embeddedjs.com/.

[9] ExpressJS. *ExpressJS, Node Framework.* [Online; accessed 25-August-2016]. 2016. URL: https://expressjs.com/.

[10] Roy Fielding. *What Is REST?* [Online; accessed 25-August-2016]. 2012. URL: http://www.restapitutorial.com/lessons/whatisrest.html.

[11] David Úbeda Fuster. *Com afecta la legislació actual a la feina del genealogista?* [Online; accessed 25-August-2016]. 2010. URL: http://upcommons.upc.edu/bitstream/handle/2099.1/9460/60418.pdf.

[12] GitHub. *GitHub help documentation.* [Online; accessed 25-August-2016]. 2016. URL: https://help.github.com/.

[13] Google. *Barchart Documentation.* [Online; accessed 25-August-2016]. 2016. URL: https://developers.google.com/chart/interactive/docs/gallery/barchart.

[14] Google. *Geomap Documentation.* [Online; accessed 25-August-2016]. 2016. URL: https://developers.google.com/chart/interactive/docs/gallery/geomap.

[15] Google. *Google Analytics Documentation.* [Online; accessed 25-August-2016]. 2016. URL: https://developers.google.com/analytics/.

[16] Google. *Linechart Documentation.* [Online; accessed 25-August-2016]. 2016. URL: https://developers.google.com/chart/interactive/docs/gallery/linechart.

[17] Heroku. *Heroku Node.js Documentation.* [Online; accessed 25-August-2016]. 2016. URL: https://devcenter.heroku.com/categories/nodejs.

[18] Church of Jesus Christ of Latter-Day Saints. *The Family.* [Online; accessed 25-August-2016]. 2016. URL: https://discover.mormon.org/en-us/topics/the-family/.

[19] jQuery. *jQuery, Write less do more.* [Online; accessed 25-August-2016]. 2016. URL: https://jquery.com/.

[20] Jennifer Kyrnin. *The three layers of web design.* [Online; accessed 25-August-2016]. 2016. URL: http://webdesign.about.com/od/intermediatetutorials/a/aa010707.htm.

[21] Ben Lin. *jQuery Preload.* [Online; accessed 25-August-2016]. 2011. URL: http://dreamerslab.com/blog/en/preload-images-with-jquery-preload-plugin/.

[22] Forbes Lindesay. *Promsies overview.* [Online; accessed 25-August-2016]. 2016. URL: https://www.promisejs.org/.

[23] Masylum. *Include Javascript Files to Node.* [Online; accessed 25-August-2016]. 2011. URL: http://stackoverflow.com/questions/5797852/in-node-js-how-do-i-include-functions-from-my-other-files.

[24] Rhonda R. McClure. *Understanding Sources.* [Online; accessed 25-August-2016]. 2002. URL: http://www.genealogy.com/articles/twigs/rhonda011702.html.

[25] Momondo. *The DNA Journey.* [Online; accessed 25-August-2016]. 216. URL: https://www.youtube.com/watch?v=tyaEQEmt5ls.

[26] Mustache. *Escaping HTML.* [Online; accessed 25-August-2016]. 2015. URL: https://github.com/janl/mustache.js/blob/master/mustache.js#L60.

[27] Mustache. *Mustache, Logic-less template.* [Online; accessed 25-August-2016]. 2016. URL: https://mustache.github.io/.

[28] NodeJS. *NodeJS, Javascript Engine.* [Online; accessed 25-August-2016]. 2016. URL: https://nodejs.org/en/.

[29] NPM. *NPM, Package Manager for Javascript*. [Online; accessed 25-August-2016]. 2016. URL: https://www.npmjs.com/.

[30] Randal S. Olson. *144 years of marriage and divorce in 1 chart*. [Online; accessed 25-August-2016]. 2015. URL: http://www.randalolson.com/2015/06/15/144-years-of-marriage-and-divorce-in-1-chart/.

[31] Optimizilla. *Optimizilla, Web optimizer*. [Online; accessed 25-August-2016]. 2016. URL: http://optimizilla.com/.

[32] FamilySearch Organization. *FamilySearch developers portal and documentation*. [Online; accessed 25-August-2016]. 2016. URL: https://familysearch.org/developers/.

[33] FamilySearch Organization. *FamilySearch Javascript SDK*. [Online; accessed 25-August-2016]. 2016. URL: http://familysearch.github.io/familysearch-javascript-sdk/2.5/.

[34] FamilySearch Organization. *FamilySearch webpage*. [Online; accessed 25-August-2016]. 2015. URL: https://familysearch.org.

[35] FamilySearch Organization. *Helper functions Sample App Javascript SDK*. [Online; accessed 25-August-2016]. 2015. URL: https://github.com/FamilySearch/javascript-sdk-sample-app/blob/master/assets/helpers.js#L241.

[36] FamilySearch organization. *API Change Log*. [Online; accessed 25-August-2016]. 2016. URL: https://familysearch.org/developers/docs/change-log.

[37] FamilySearch organization. *GEDCOMX*. [Online; accessed 25-August-2016]. 2012. URL: https://familysearch.org/developers/docs/guides/gedcomx.

[38] Antonio Alfaro de Prado. *El registro civil de España (1871—¿?)* [Online; accessed 25-August-2016]. 2015. URL: http://www.genealogiahispana.com/archivos/el-registro-civil-de-espana-1871/.

[39] Kim Preshoff. *Population pyramids: Powerful predictors of the future*. [Online; accessed 25-August-2016]. 2014. URL: https://www.youtube.com/watch?v=RLmKfXwWQtE.

[40] James Rothwell. *Florida police officer shoots black man caring for autistic patient*. [Online; accessed 25-August-2016]. 2016. URL: http://www.telegraph.co.uk/news/2016/07/21/florida-police-officer-shoots-at-man-caring-for-autistic-patient/.

[41] Chris Sevilleja. *Use ExpressJS to Get URL and POST Parameters*. [Online; accessed 25-August-2016]. 2015. URL: https://scotch.io/tutorials/use-expressjs-to-get-url-and-post-parameters.

[42] Yaron Steinbuch. *Woman records horrific scene after boyfriend is fatally shot by police*. [Online; accessed 25-August-2016]. 2016. URL: http://nypost.com/2016/07/07/woman-live-streams-bloody-aftermath-of-police-involved-shooting/.

[43] TWIG. *Twig, Template Engine for PHP*. [Online; accessed 25-August-2016]. 2016. URL: http://twig.sensiolabs.org/.

[44] w3schools. *HTML5 Local Storage*. [Online; accessed 25-August-2016]. 2016. URL: http://www.w3schools.com/html/html5_webstorage.asp.

[45] FamilySearch Wiki. *Genealogical Ethics (National Institute) — FamilySearch Wiki,* [Online; accessed 25-August-2016]. 2015. URL: https://familysearch.org/wiki/en/index.php?title=Genealogical_Ethics_(National_Institute)&oldid=2301878.

[46] FamilySearch Wiki. *Research Process — FamilySearch Wiki,* [Online; accessed 25-August-2016]. 2016. URL: https://familysearch.org/wiki/en/index.php?title=Research_Process&oldid=2567582.

[47] Wikipedia. *Application programming interface — Wikipedia, The Free Encyclopedia*. [Online; accessed 6-September-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=Application_programming_interface&oldid=737224491.

[48] Wikipedia. *Cascading Style Sheets — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=Cascading_Style_Sheets&oldid=735986518.

[49] Wikipedia. *GEDCOM — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=GEDCOM&oldid=731965739.

[50] Wikipedia. *Genealogy — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=Genealogy&oldid=735149060.

[51] Wikipedia. *HTML5 — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=HTML5&oldid=735803895.

[52] Wikipedia. *JavaScript — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=JavaScript&oldid=735960528.

[53] Wikipedia. *JSON — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=JSON&oldid=735120383.

[54] Wikipedia. *Model–view–controller — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=Model%E2%80%93view%E2%80%93controller&oldid=735649380.

[55] Wikipedia. *Representational state transfer — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=Representational_state_transfer&oldid=735975196.

[56] Wikipedia. *The Church of Jesus Christ of Latter-day Saints — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=The_Church_of_Jesus_Christ_of_Latter-day_Saints&oldid=735587626.

[57] Wikipedia. *XML — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-August-2016]. 2016. URL: https://en.wikipedia.org/w/index.php?title=XML&oldid=736122302.

[58] Wogan. *Sorting Javascript Objects*. [Online; accessed 25-August-2016]. 2016. URL: http://stackoverflow.com/questions/1129216/sort-array-of-objects-by-string-property-value-in-javascript.