**Estudi tècnic del projecte**

**Aplicació a implementar**

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

Des del principi volíem implementar una pàgina web. Després de realitzar un estudi inicial de la API i de les diferents opcions de desenvolupament, crear una pàgina web semblava una de las opcions més factibles i flexibles.

El fet d'haver d'interactuar amb una API, que funciona per URIs, ens va fer pensar que implementar una pàgina web tenia tot el sentit del món.

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

Les tecnologies o recursos estudiats es poden dividir en tres grans blocs: Les tecnologies web, tecnologies de desenvolupament i les tecnologies de desplegament.

Les tecnologies web representen aquell conjunt de llenguatges, frameworks i arquitectures que serien utilitzades de cara a la construcció de la pàgina web. És a dir, quin llenguatge s'utilitzaria per implementar el servidor, com es realitzaria la comunicació amb la API de FamilySearch, quins llenguatges per implementar la part visual de la web, etcètera, etcètera.

Les tecnologies de desenvolupament són el conjunt de tecnologies i eines específiques destinades a ajudar durant el desenvolupament.

Finalment, les tecnologies de desplegament fan referència al allotjament web escollit i com s'havia de realitzar el desplegament.

**Tecnologies web**

**El model vista controlador**

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

En el desenvolupament web s'utilitza molt una arquitectura de tres capes que emula bastant bé el model vista controlador. Però, en que consisteix exactament el model vista controlador? El model vista controlador, també conegut com MVC, és un patró d'arquitectura pensat per la implementació d'aplicacions amb interfície d'usuari per ordinadors.

Com el nom bé indica, el model es composa principalment de tres elements. El Model, la Vista i el Controlador. A continuació és descriu amb més profunditat el rol de cada un dels components.

El Model s'encarrega de gestionar les dades, lògica i regles sobre les que l'aplicació funciona. Aquest component del model MVC és el que gestionarà les connexions amb les bases de dades en cas de que sigui necessari, crearà els blocs de dades a retornar i els retornarà al controlador.

La Vista o Vistes representen el conjunt de representacions visuals de la informació. En altres paraules, la interfície. L'usuari interactua amb aquesta per tal de poder realitzar les diferents operacions que l'aplicació ofereix.

Finalment, el Controlador s'encerrega de recollir els diferents inputs enviats per l'usuari, validar-ne l'estat, comunicar-se amb el model en cas de necessitat i modificar les vistes com a conseqüència de les accions de l'usuari.

La gràcia d'aquest model és que l'usuari només disposa d'accés i permís d'interacció amb les vistes. Aquestes, comuniquen les accions de l'usuari a un controlador. El controlador com ja hem explicat s'encarrega de gestionar les comunicacions tant amb el model com les vistes, fent completament transparent l'ús del model a l'usuari.

La figura \_\_\_\_ mostra un exemple de com funciona el model MVC de forma ideal i que intentarem seguir en gran mesura durant la implementació de la pàgina web.

[]

Un últim aspecte important de terminologia que anirà sortint és a quin component d'una pàgina web s'assigna cada una de les tres capes del model MVC.

Les pàgines web s'acostumen a poder separar mitjançant dos conceptes el front end i el back end.

El front end fa referència a la capa de presentació, i per tant, serà associat en el nostre cas a la capa de Vistes del model MVC.

Per altre banda, el back end és el component que realitza l'accés a les dades i seria l'equivalent a la capa del Model en l'arquitectura MVC.

Finalment, el controlador del model MVC es veuria reflexat en una pàgina web com la part del codi del client (navegador) que l'usuari no veu ni interactua directament amb ella. A vegades aquesta part es coneix pel nom del "back end del front end".

La figura \_\_\_ mostra les tres capes que s'utilitzarà en el projecte per seguir fins a cert grau l'arquitectura MVC. Durant els següents apartats d'aquesta secció de la memòria, l'anirem emplenant amb les diferents tecnologies que seran utilitzades i al final d'aquesta, es mostrarà el seu estat final.

[]

**Les tres capes del disseny web**

Abans d'entrar en les tecnologies que utilitzarem per l'apartat visual de la web, també conegut com el front end, o FE d'ara en endavant, volem explicar les tres capes amb les que aquest s'acostuma a dividir.

Les tres capes són anomenades: Estructura, Estil i Comportament.

La capa d'estructures es generalment coneguda per ser el contingut. Representa l'estructura sobre la que el contingut de la pàgina web serà pintat. Es podria veure com el conjunt de llenços que es preparen per tal de pintar-ne contingut a sobre.

La capa d'estil defineix com les diferents estructures creades s'han de composar unes respecte les altres. També dictamina l'estil que tots els diferents elements de la web tindran, com per exemple, el color de la font, imatges de fons, etcètera.

Finalment la capa de comportament s'encarrega de respondre a les diferents accions realitzades per l'usuari. Aquest nivell de la pàgina és el que hem anomenat en l'apartat anterior "el back end del front end".

Ara bé, perquè és important diferenciar aquestes tres capes i perquè ho hem volgut explicar en la memòria? Les raons son diverses:

* Recursos compartits. Moltes vegades certs aspectes de les capes d'estil i comportaments podran ser utilitzats per diverses pàgines del nostre domini i per tant no te cap sentit replicar-les.
* Descàrregues més ràpides. Un cop un d'aquests recursos compartits ha estat descarregat, ja no cal tornar-lo a descarregar en futures pàgines.
* Permet el desenvolupament de diverses persones a la vegada sense solapament.
* Millora dels aspectes SEO (optimització dels motors de cerca), accessibilitat i compatibilitat.

**La tecnologia HTML 5**

La primera tecnologia que cal estudiar si es pretén implementar una pàgina web és el HTML, i en el nostre cas, el HTML 5. Aquesta tecnologia consisteix en un llenguatge d'etiquetatge utilitzat com estàndard per la presentació de contingut a través del núvol. Com el seu nom indica, es tracte de la cinquena versió d'aquest estàndard.

El concepte llenguatge d'etiquetatge significa que els blocs de contingut s'envolten per etiquetes d'obertura i clausura que donen un significat concret al contingut situat a l'interior.

Per exemple, per indicar un bloc de contingut que al seu interior conté tres paràgrafs, es podria utilitzar una estructura similar a la que segueix:

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

Aquesta tecnologia s'utilitza per crear la capa d'estructures explicada a l'apartat: 'Les tres capes del disseny web' i tal com s'indica, s'utilitza per crear l'estructura sobre la que el contingut es mostrarà. Per tant, el HTML 5 és utilitzat únicament per la capa del front end.

**La tecnologia CSS**

La tecnologia CSS, també coneguda com a full d'estils en cascada, és un llenguatge utilitzat per marcar com ha de ser l'aspecte i forma dels diversos elements marcats per l'estructura de la pàgina o el HTML.

El CSS va néixer per separar el contingut d'un document de la presentació d'aquest. Aquest llenguatge permet entre moltes altres coses decidir la font i estil de la font de cada element de la pàgina web, decidir 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 estats dels diferents components, etcètera, etcètera.

Una gràcia del CSS és que al ser una fulla d'estils en cascada, un element pot tenir més d'un valor diferent pel mateix atribut, per exemple, color de la font. La regla que decideix l'aspecte final d'un element és la més recent o interior. D'aquesta forma es pot donar un comportament genèric a certes etiquetes i personalitzar només aquelles que faci falta en situacions concretes.

En resumen, la tecnologia CSS permet controlar tots els elements que fan referència a l'aparença. Això ho realitza mitjançant tres classes d'etiquetatge diferents:

* **Etiquetes HTML:** Aquestes etiquetes s'utilitzen per aplicar regles a tots els components HTML que encaixin amb una etiqueta en concret. Per exemple, si s'utilitzen l'etiqueta 'body', es podria decidir des d'aquí la tipografia a ser utilitzada per tots els continguts del cos del HTML.
* **Etiquetes de classe:** Aquestes etiquetes, creades per l'usuari mitjançant la concatenació del caràcter '.' amb un nom qualsevol (per exemple, .color-blue), permeten assignar regles d'estil a classes concretes creades per l'usuari. Aquestes classes es poden incloure després dins de les etiquetes HTML proporcionant així l'estil desitjat només a aquell conjunt d'etiquetes marcat de forma específica.
* **Etiquetes identificadores:** Aquestes etiquetes són similars a les de classe, però en comptes d'utilitzar el caràcter '.' abans del nom que li volem donar, s'utilitza el caràcter '#'. La diferencia principal és el concepte d'identificar una etiqueta de forma única i per tant, l'estil s'aplicaria només al element HTML amb l'identificador que encaixes amb el nom de l'etiqueta.

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

No obstant, les bases són les mateixes pels cassos complicats i creiem que el text exposat ha de ser suficient per tal de fer-se una idea de com funciona aquesta tecnologia.

La tecnologia CSS s'utilitza de forma única en la capa del front end.

**La tecnologia de plantilles Moustache**

Abans d'entrar en els detalls d'aquesta tecnologia, cal comprendre que és un llenguatge de plantilles.

Els llenguatges de plantilles existeixen principalment per dos propòsits:

1. **Reutilització i minimització de codi HTML.** De forma general els llenguatges de plantilles permeten importar fitxers HTML a altres documents HTML. D'aquesta forma s'evita la creació de codi redundant o replicat. Per exemple, si casi totes les pàgines d'un domini web tenen la mateixa barra de navegació, seria poc eficient haver de crear l'estructura HTML d'aquesta en totes i cada una de les pàgines. També permeten crear bucles de codi HTML i iterar sobre ells. Imaginem que es volen crear 100 paràgrafs de text, 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 tots els paràgrafs necessaris.
2. **Recollir paràmetres del servidor:** La segona funcionalitat dels llenguatges de plantilla és el de recollir els paràmetres enviats pel servidor conjuntament a la pàgina HTML i poder utilitzar-los per pintar contingut dinàmic al document HTML en el moment d'impressió.

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

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

Una de les altres diferencies principals entre Mustache i Twig és que Mustache pot ser utilitzat en casi qualsevol idioma de programació web mentre que Twig es específic de PHP. Al final es va decidir utilitzar un servidor Node.js (com s'explicarà més endavant) fet que descartava la possibilitat d'utilitzar el llentuatge Twig.

El fet de que Mustache sigui un llenguatge de plantilles casi sense lògica no significa que sigui menys potent. 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:** Per invocar en el HTML un paràmetre del servidor només cal utilitzar el nom del paràmetre envoltant 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, per exemple, el contingut del fitxer navbar.html en el document HTML actual.
* **Iteracions sobre blocs de codi:** Imaginem que el servidor retorna un vector de països, si volguéssim pintar el nom de cada un en un paràgraf podríem definir el bucle HTML de la forma següent: {{#countries}} <p> {{name}} </p> {{/countries}}

Resulta bastant palpable comprendre lo útils que poden arribar a ser aquests llenguatges de plantilla i perquè s'ha decidit utilitzar-los en el desenvolupament de l'aplicació web.

Aquesta tecnologia pot esdevenir confusa a on aplica i a pesar que pugui semblar una tecnologia del frontal, o front end, en realitat aplica a la part del servidor o back end, modificant el HTML abans de servir-lo al client o navegadors.

**Les tecnologies Javascript i jQuery**

Tot i que aquestes dos tecnologies no són exactament el mateix, volem presentar-les de forma conjunta ja que és com s'utilitzen en el món de les aplicacions web.

Javascript és un llenguatge dinàmic i 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 poden ser els estils orientats a objectes, imperatius o funcionals.

Cal comprendre que Javascript també és utilitzat fora del món web, per aquest n'és el seu principal mercat. Tampoc s'ha de confondre aquesta llengua amb Java. A pesar de les similituds entre els dos, són dos llenguatges de programació diferents, amb desenvolupaments separats.

Per l'altre banda, jQuery és una llibreria de Javascript plena de funcionalitats dedicada a la manipulació de documents HTML, CSS, gestió de les interaccions amb aquests i comunicació amb el servidor. Aquesta llibreria s'ha convertit en un estàndard de la programació web i resulta indispensable de cara a la programació web.

Javascript i jQuery són utilitzats de forma conjunta en el "back end del front end" i compleixen, per tant, la funció de controlador en l'arquitectura MVC explicada al començament d'aquesta secció de la memòria.

No volem entrar molt més en detall en aquestes tecnologies, ja que les possibilitats són realment infinites, però perquè s'entengui una mica més la funcionalitat d'aquestes, imaginem una pàgina web que disposa d'un botó amb identificador 'submit', que un cop pres, canvia el valor d'un camp de text de la pàgina web.

En aquest exemple, s'utilitzaria jQuery per detectar que el botó 'submit' del HTML ha estat pressionat, Javascript per definir com a variable el nou camp de text i jQuery per imprimir el contingut de la nova variable en el camp de text específic del HTML. El codi podria ser per exemple el que segueix:

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

Tot i ser un exemple molt bàsic, creiem que pot ajudar a comprendre perquè s'utilitzen aquestes tecnologies en el món de la programació web.

**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 pàgines web.

La funcionalitat principal per la que es va decidir utilitzar aquesta tecnologia és la funcionalitat del grid o quadrícula. A l'hora de programar una pàgina web que sigui visible de forma acceptable tant en dispositius mòbils com escriptoris, un dels punts més complicats és al d'assegurar que tots els components definits es comportin com s'espera d'ells i ocupin posicions diferents segons la grandària del dispositiu en el que són pintats.

El grid parteix de la base que el contingut web es situarà dins de contenidors que poden ocupar tot l'ample possible o bé una amplada màxima. Per simplificar-ho, podríem dir que l'objectiu del grid es partir aquests contenidors en files i columnes, on cada casella pot contenir un bloc de codi diferent. La imatge [] mostra un exemple de com un contenidor podria estar dividit en quatre files, on cada fila esta formada per un nombre de columnes diferent.

La gracia de dividir les files 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, creant d'aquesta forma el contingut per un dispositiu mòbil sense necessitat de canvis en el codi. La imatge [] mostra com una configuració web per escriptori es trancaria i reorganitzaria per un dispositiu mòbil.

Apart del grid, que representa la característica principal per la que es va decidir utilitzar bootstrap, aquesta tecnologia també ofereix estils, o classes CSS, per taules, botons, formularis, imatges, tipografies, icones, barres de navegació, alertes, barres de progrés, contingut expansible, etcètera.

A pesar de que, evidentment, la majoria de classes han de ser retocades mitjançant CSS propi per tal d'encaixar els diferents elements a la nostre aplicació web, representen un molt bon punt de sortida que evita la creació de grans quantitats de codi CSS i a la vegada, n'assegura la correcta visualització a través dels diferents dispositius.

Com hem comentat, el marc de treball bootstrap crea bàsicament fitxers de codi CSS i Javascript que serveixen per complementar els nostres fitxers de codi. Per tant, aquest actua tant en el front end del nostre sistema com en el "back end del front end". Per tant, exerceix a la vegada, un efecte a les capes de vista i control del patró MVC.

**MVC**https://yalantis.com/media/content/ckeditor/2014/05/mvc\_role\_diagram.png  
https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller  
https://en.wikipedia.org/wiki/Front\_and\_back\_ends

**3 Layer Web Design**http://webdesign.about.com/od/intermediatetutorials/a/aa010707.htm

**NODE JS**http://javascript.tutorialhorizon.com/2015/08/11/what-does-npm-start-do-in-nodejs/  
https://www.npmjs.com/enterprise  
https://howtonode.org/introduction-to-npm  
https://nodejs.org/en/

**HTML 5**https://en.wikipedia.org/wiki/HTML5

**Plantilles**https://mustache.github.io/  
http://www.embeddedjs.com/  
http://twig.sensiolabs.org/

**Jquery i Javascript**https://jquery.com/  
https://en.wikipedia.org/wiki/JavaScript

**Bootstrap**http://getbootstrap.com/