**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 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 per aquest camí.

La pàgina web representava l’opció més flexible i factible, ja que els protocols per integrar-se amb APIs es troben bastant desenvolupats i a més a més, oferia l’oportunitat d’utilitzar els SDK oficials, així com 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 tal de desenvolupar una aplicació web, escollir aquesta opció tenia tot el sentit del món.

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, tecnologies de desenvolupament i les tecnologies de desplegament.

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 visual 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 al allotjament web escollit i les tecnologies necessàries per tal de poder completar el desplegament de l’aplicació al núvol.

**Tecnologies web**

**El 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 que no havia estat concebuda, a altres tecnologies.

En el món del desenvolupament web, sembla ser que predomina molt una arquitectura de tres capes, que emula bastant bé el model vista controlador. Però, en que consisteix exactament aquest model? El model vista controlador, també conegut com MVC, és un patró d'arquitectura pensat per la implementació d'aplicacions que disposen d’una interfície d'usuari.

Com el nom bé indica, el model es composat 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 que treballa el sistema. El Model, també s’encarrega de crear la lògica i regles, sobre les que l'aplicació funciona. D’aquesta forma, aquest component serà l’encarregat de gestionar les connexions amb les bases de dades, en cas que es requereixi l’utilització d’aquestes, 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 altres paraules, la interfície que l’usuari veurà i amb la que podrà interactuar. Mitjançant les diferents interaccions possibles amb aquesta, 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, validar-ne l'estat i comunicar-se amb la capa del Model per obtenir les dades o recursos (entesos com a fitxers servits per la capa Model), demandes 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 de l’aplicació a la interfície d’usuari.

La gràcia d'aquest model és que l'usuari només disposa d'accés i permís d'interacció, a la capa de les vistes. Aquestes, comuniquen les accions de l'usuari al 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 es mostren.

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.

**El SDK de Javascript oficial de FamilySearch**

Una de les tecnologies sobre les que calia escollir era la que marcaria com s'haurien de realitzar les comunicacions amb la API de FamilySearch.

Una de les opcions més evidents era la d'utilitzar una implementació directa contra la API. Aquesta aproximació segur que seria la més flexible de totes, ja que ens permetria realitzar exactament les peticions de la forma desitjada contra la API. En contrapartida, s'haurien de tractar les respostes XML o JSON de forma manual, el que acostuma a ser una tasca repetitiva i innecessària, la realització de peticions i codificacions de les URL també s'hauria de realitzar des de zero i qualsevol lleuger canvi en la API implicaria haver de canviar el codi per la nostre part.

Desprès d'estudiar les funcionalitats i documentació inclosa en el portal de desenvolupadors de FamilySearch, l'idea d'utilitzar un SDK per la implementació web va anar cobrant força. Els avantatges principals, el processat de les dades i altres funcionalitats extres empaquetades per tal de ser utilitzades de forma transparent. El desavantatge, menys flexibilitat a l'hora de demanar i navegar a través dels recursos i haver d'estudiar el funcionament d'una tecnologia extra.

Desprès de valorar-ho bastant es va decidir realitzar la implementació dels exemples d'interacció amb la API mitjançant un SDK. Es creia que el benefici de tenir un cert grau de robustesa als petits canvis que la API es pogués veure sotmesa, més el fet de no haver de realitzar el tractament de dades de forma manual, compensava l'esforç d'estudiar el funcionament d'un SDK. Apart, es creia que per tal de demostrar el propòsit general de la API, no feia falta ser capaç de controlar les micro interaccions amb la API.

Com que l'aplicació que es volia implementar era una pàgina web, eren tres els diferents SDK que podien ser utilitzats. El de Python va quedar descartat des del començament ja que no es trobava acabat i tampoc representava un SDK oficial. Recordar que els SDK oficials són desenvolupats per la mateixa organització de FamilySearch i per tant més robusts de cara a possibles canvis en la API.

La disputa entra la utilització del SDK de Javascript o el de PHP es va decantar a favor del SDK de Javascript després d'un estudi superficial dels dos.

A pesar de que la preferència inicial era utilitzar el SDK de PHP, ja que en el passat havia tractat una mica amb el llenguatge i tenia una idea aproximada de com aquest interactuava amb els elements del frontal d'una pàgina web, les funcionalitats d'aquest estaven molt per radera de les que oferia el SDK de Javascript i la documentació extensa del segon també garantia una corba d'aprenentatge més fàcil.

Tot plegat, va fer que el SDK escollit per implementar els exemples, fos el que va ser creat amb el llenguatge Javascript. Els detalls més tècnics d'aquest, així com el seu funcionament bàsic seran explicats en més detall en les següent seccions de la memòria i per tant, no hi entrarem en més detall en aquesta secció.

Finalment, esmentar que el SDK de Javascript podia ser implementat tant en la capa del Model com en la del Controlador de l'arquitectura MVC.

La implementació més robusta i segura seria la de implementar-lo en la part del servidor, per tant, en el back end. No obstant, degut al grau de dificultat de les operacions que es volien realitzar i la poca necessitat d'emmagatzematge de dades per part dels exemples, es va decidir implementar-lo a la capa del controlador.

D'aquesta forma, es reduïa al mínim la necessitat d'utilització de la tecnologia Ajax i feia molt més fàcil la impressió dels resultats provinents de les peticions a la API al front end de l'aplicació. Es creia que ja eren suficients les tecnologies que s'havien d'estudiar com per afegir-ne una altre que no era imprescindible.

Al final del projecte, i amb perspectiva del coneixement obtingut, segurament la implementació del SDK al back end no sembla tant complicada, però cal recordar que el coneixement en com una web havia de ser implementada abans del projecte, era pràcticament nul.

**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 las peticions de navegació i servir el contingut que toca als navegadors o usuaris. També s'encarrega de gestionar l'accés a les bases de dades (en cas de que existeixin) i processar la informació per tal de servir-la d'una forma que la capa del controlador pugui comprendre i utilitzar.

Com que havíem escollit utilitzar el SDK de Javascript per realitzar les connexions a la 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 de que al final no volguéssim implementar la lògica en la capa del controlador.

La única tecnologia que complia amb les condicions que es buscàvem era Node.js. Node.js s'ha implementat sobre un model basat en esdeveniments i no bloquejant, 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 es tractin 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 que s'ha de realitzar a continuació. Aquest aspecte converteix a Node.js com una tecnologia molt escalable.

A més a més, com tot software de codi obert que ocupa una gran part del mercat, molts desenvolupadors s'han dedicat a crear paquets de software que amplien les funcionalitats inicials de Node.js. Totes aquestes poder ser trobades al repositori de paquets del ecosistema npm, el més gran del món en quan a llibreries de codi obert.

Comentarem més sobre aquesta tecnologia i com funciona en els apartats específics d'implementació.

Aquesta tecnologia realitza el paper del Model en l'arquitectura MVC o en la analogia que hem creat per la nostre aplicació web, aquesta es utilitzada en el back end.

**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 és un marc de treball per el 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 que Node.js destaca.

Molts altres marcs de treball o frameworks populars que desitgen expandir en més profunditat 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 nostre aplicació degut a que les necessitats que tenim pel back end són relativament simples un cop s'ha decidit implementar la interacció amb la API de Familisearch en la capa del controlador.

Les funcionalitats o eines principals que proporciona Express són:

* **Eines d'enrutament:** L'enrutament recull el conjunt d'accions i processos que tenen lloc des de que el servidor web rep la petició d'accés a una URI i com aquest respon a la petició.
* **Eines middleware:** El concepte middleware serveix per referir-nos a aquelles peces de software que fan de pont entre dos parts o components d'un sistema. En el marc de treball en el que 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 molt típic és per exemple el de comprovar que un usuari te els permisos necessaris per veure la pàgina que ha demanat.
* **Eines per la utilització de llenguatges de 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 aquesta mateixa secció.
* **Eines per gestionar els 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.

Degut al a naturalesa de la nostra aplicació les funcionalitats que ens interessen més son les tres primeres.

Aquesta tecnologia, evidentment, al ser un framework de Node.js, s'utilitza en la part del back-end de la nostre aplicació.

**Conclusió sobre les tecnologies de 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 nostre aplicació web.

En total, es tracte de nou tecnologies que han hagut de ser estudiades per separat i aprés a utilitzar de forma conjunta. L'única tecnologia sobre la que es disposava en certa forma de 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 tres capes que s'havia definit al inici d'aquesta secció, es pot veure representada en al figura \_\_\_\_.

**Tecnologies específiques pel desenvolupament de l'aplicació**

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

**La tecnologia Github**

Degut a les complicacions de compaginar una jornada completa de treball amb l'elaboració del projecte, aquest ha estat desenvolupament des de diverses localitzacions. En concret, tres: Casa, Portàtil i l'oficina de l'empresa.

Per tal de facilitar la sincronització d'arxius entre les diferents estacions de treball es va decidir emprar Github.

Github es un respositori 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 està emmagatzemat, cada estació de treball diferent pot descarregar-se el codi 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 sobre els mateixos arxius per evitar en la mesura que sigui possible solapaments i conflictes. Github disposa d'eines per fer front a aquestes situacions, ja que en realitat és una eina pensada per treballar en equip sobre el mateix bloc de codi. No obstant, com que aquest projecte ha estat realitzat s'ha prescindit de la utilització de branques i tots els canvis s'han aplicat sobre el branca mestre 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'*

Apart 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 de que alguna de les oficines de treball quedi malmesa o es vulgui recuperar una versió antiga d'algun fitxer del codi.

**Tecnologia Node.js** **i Express a nivell Local**

Durant el desenvolupament, l'aplicació web no ha estat penjada el núvol. Per facilitar-ne les probes i no dependre tant de la connexió, 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 la url 'http://localhost:8080'.

Per tant, el nostre sistema necessitava ser capaç d'emular les tecnologies que formarien part del servidor. No entrarem a descriure la tecnologia Node.js ni Express perquè ja ho hem fet en apartats anteriors.

El que si que volíem fer era exposa que aquestes tecnologies havien estat instal·lades a nivell local.

**Tecnologia NPM**

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

**Paquet Nodemon**

Un paquet que volem destacar en aquesta secció ja que s'ha utilitzat únicament per el desenvolupament de la plataforma, i no per la versió de producció, doncs no tindria sentit, s'anomena Nodemon.

Aquest paquet observa els canvis realitzats en els fitxers que es troben en el directori en Nodemon ha estat instal·lat i en cas de que algun canvií, aquest reinicia l'aplicació de forma automàtica.

Pot semblar un paquet que aporta funcionalitat, però quan et trobes implementat una pàgina web generalment realitzes molts petits canvis dels que vols observar-ne l'afecte sense haver de tancar i rellançar l'aplicació.

El fet de que sigui un paquet que ha estalviat bastant de temps a l'hora de realitzar petites proves i observar-ne els canvis, volíem que quedes esmentat en la memòria.

**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/

**Familysearch Javascript SDK**http://familysearch.github.io/familysearch-javascript-sdk/2.4/#/api  
https://familysearch.org/developers/libraries

**Node.js**https://nodejs.org/en/  
https://expressjs.com/

**Github**https://github.com/

**Nodemon**https://github.com/remy/nodemon