**Introducció a l'aplicació web**

En aquesta secció de la memòria s’introduiran tots aquells aspectes generals referents a l’aplicació web. En concret, es tractaran els següents punts:

* Accés i codi de l’aplicació web.
* Requisits funcionals i no funcionals de l’aplicació web.
* Estructura de l’aplicació web.
* Els fitxers de l’aplicació i les seves funcions.
* Funcionament general de l’aplicació web.
* Detalls específics de la implementació.
* Certificació de l’aplicació.
* Google Analytics.
* Optimització d’imatges.
* Hosting de l’aplicació web.

**Accés a l’aplicació web i codi de l’aplicació**

L’aplicació web es troba desplegada al núvol sota l'URL:

[*https://pfc-family-search.herokuapp.com/*](https://pfc-family-search.herokuapp.com/)

Per accedir a la zona específica d’exemples, la que s’encarrega de mostrar els diferents exemples d’interacció amb l’API, fa falta utilitzar el següent usuari i contrasenya:

* Usuari: tum000145207
* Contrasenya: 1234pass

Per altra banda, el codi de les aplicacions web sol ser extens en nombre de línies i mostrar-lo en aquesta memòria resulta impossible. El codi realitzat ocupa un total de [nombre de línies] repartides un [x%] en HTML, un [y%] en Javascript i jQuery i un [z%] en css.

Tot el codi de l’aplicació pot ser trobat en el repositori GitHub accessible a través del següent URL:

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

L’estructura del codi serà presentada més endavant, en aquesta mateixa secció de la memòria, però principalment, el servidor està compost pel fitxer *app.js*, els fitxers HTML es troben a la carpeta *views* i els fitxers Javascript i jQuery, a la carpeta *assets.*

**Requisits de l'aplicació**

Aquesta llista pretén oferir un tast dels requisits o manaments que s'han tingut en compte durant el desenvolupament de l'aplicació web.

**Requisits funcionals**

* La web ha de permetre identificar-se amb FamilySearch mitjançant el sistema d'autentificació per pop-up.
* La web ha de permetre a l'usuari tancar la connexió amb FamilySearch mitjançant una funcionalitat de `Sign Out'.
* L'aplicació ha de ser capaç de tancar automàticament la connexió amb FamilySearch si aquesta expira.
* La web ha d'incloure una secció que ofereixi un petit resum del rerefons que va originar el projecte.
* La web ha de disposar d'una secció en què s'enumerin i exposin les diferents propostes de projecte generades pels futurs estudiants.
* L'aplicació ha d'oferir la possibilitat de cercar persones en l'arbre familiar de FamilySearch i observar-ne els detalls d'alguna en concret.
* L'aplicació ha de permetre a l'usuari observar l'evolució geogràfica d'un cognom donat un conjunt de països i període de temps.
* L'aplicació ha de permetre la visualització del nombre de naixements, casaments i defuncions enregistrades per un país al voltant d'un any concret.
* La secció d'exemples ha de ser només accessible si l'usuari es troba identificat a FamilySearch i ha rebut el token d'ús pertinent.
* En cas que el token expiri, l'usuari ha de ser redirigit a la pàgina principal en el moment d'expiració o en la seva següent interacció si aquest es troba dins de l'àrea d'exemples.
* L'aplicació ha d'emmagatzemar el token proporcionat per FamilySearch que rep l'usuari en un recurs que no sigui accessible ni modificable per tercers.
* No es permetrà a l'usuari llençar dues crides contra l’API de FamilySearch simultànies per la mateixa funcionalitat des de la mateixa pestanya del navegador.
* L’aplicació ha d'aportar la informació bàsica sobre l’origen del projecte i el seu rerefons. L'aplicació també ha d’enllaçar en algun lloc amb el codi font del projecte.

**Requisits no funcionals**

* L'aplicació web ha de funcionar i ser visualitzada de forma correcta en els principals navegadors web moderns.
* Els formularis de l'aplicació web que puguin generar errors han de proporcionar informació bàsica a l'usuari en el moment que el camp és abandonat o informació més detallada si envia el formulari amb errors.
* Els formularis de l’aplicació han de donar un feedback positiu en cas que els camps siguin omplerts de forma correcta.
* La web ha de ser relativament fàcil d'utilitzar, oferint les eines necessàries als usuaris i facilitant la navegació per les diferents seccions.
* Mentre l'aplicació web espera resposta de l’API de FamilySearch, s'ha de mostrar a l'usuari que l'aplicació es troba esperant resultats i el progrés realitzat fins al moment.
* L'aplicació ha de donar un feedback clar a l'usuari quan la interacció amb l’API de FamilySearch finalitza.
* L'aplicació ha de ser navegable de forma acceptable mitjançant dispositius mòbil. Les integracions amb l’API de FamilySearch també han de ser utilitzables, però no cal que la informació resultant es trobi completament adaptada a aquests dispositius.
* Les imatges de l'aplicació s'han de trobar optimitzades en la mesura que sigui possible per intentar que aquesta carregui el més ràpid possible en un entorn d'hostalatge gratuït.
* Les imatges principals de l'aplicació s'han de carregar de formar transparent quan l'aplicació és iniciada i la primera pàgina és carregada, per millorar la visualització de la web quan l'usuari navega entre les diferents seccions.
* La llengua utilitzada en el web serà l'anglès per tal d'ajudar i facilitar el procés de certificació.
* El projecte s'ha de trobar sota una certificació Creative Commons d'atribució no comercial.
* La informació sobre el codi font del projecte, la llicència i la facultat d'informàtica ha de trobar-se disponible en el peu de pàgina de les pàgines de la web.
* Es podrà monitorar la navegació dels usuaris pel web, així com les seves accions principals i errors generats.
* Es podrà obtenir la configuració dels sistemes amb els quals s'ha navegat per la web i veure si el comportament d'algun d'ells és més propici a la generació d'errors.
* Els fitxers Javascript s'han de trobar el més al final possible dels arxius HTML per facilitar la càrrega del contingut.
* Es reutilitzarà codi HTML i Javascript en la mesura que sigui possible per tal d'evitar la duplicació de contingut.

**Estructurà de l'aplicació web**

**Introducció a l'estructura de l'aplicació**

L'aplicació web és relativament simple pel que respecta a la navegació i les diferents seccions que la conformen.

La figura [] mostra l'arbre de continguts accessibles. Cal indicar, que aquesta figura no representa les úniques rutes de navegació existents entre les diferents seccions, sinó un breu mapa del contingut total disponible a través del web i d’on penja cada secció.

[]

Més endavant, s’explicarà més en detall en què consisteix cada una de les pàgines o seccions de la nostra aplicació web, però primer volem presentar l'estructura general o esquelet, que segueixen gairebé totes les pàgines del nostre web.

La figura [] mostra l'esquema bàsic sobre el qual les pàgines són construïdes. Aquest pot ser descrit o desglossat en les següents seccions:

1. **Barra de navegació:** Permet desplaçar-se per les diferents seccions principals.
2. **Capçalera de secció:** Conté el títol i subtítol de la pàgina sobre impressionat a una imatge relacionada.
3. **Contingut principal:** El contingut principal i únic d'aquesta pàgina.
4. **Footer:** Peu de pàgina. Inclou informació sobre el codi font del projecte, la llicència i la facultat d'informàtica de Barcelona.

Convidem des de la memòria, als usuaris de l'aplicació web, a redimensionar la finestra web per tal d'observar com els continguts de cada pàgina s’adapten al dispositiu que els mostra.

A continuació descriurem el contingut de cada secció o apartat que conforma la nostra aplicació web.

**Home o pàgina principal**

La home és la primera pàgina que veu l'usuari quan entra a l'aplicació web. Aquesta no té cap altre propòsit que el de donar la benvinguda i enllaçar als diferents continguts.

El bloc de contingut principal d’aquesta pàgina inclou enllaços i breus descripcions, a les tres seccions principals del projecte. A saber: rerefons, propostes de projecte i exemples d'implementació.

La principal diferència entre la versió per dispositius mòbils i la d'escriptori és que la primera fa desaparèixer el subtítol, simplifica el títol, transforma la barra de navegació en una usable per dispositius mòbil i transforma la secció de contingut principal apilant-ne el contingut de cada una de les tres columnes que la conformen.

**Rerefons**

La pàgina de rerefons conté la informació bàsica respecte a l'origen, context i motivacions, que ens van portar a realitzar aquest projecte. Consisteix en un breu resum, molt superficial, d’alguns dels apartats exposats en la primera secció de la memòria.

El bloc de contingut principal d'aquesta pàgina consisteix en dos grans blocs de text. El primer, descriu el rerefons del projecte, mentre que el segon ofereix una petita descripció del context i motivacions.

La principal diferència entre les versions d'escriptori i mòbil és que la segona presenta un títol més simple, la desaparició del subtítol i una barra de navegació adaptada a dispositius mòbils. L'estructura del contingut roman igual, això si, adaptat a la grandària del dispositiu que el conté.

**Propostes de projecte**

L'apartat de propostes de projecte recull les diferents propostes que han estat generades per servir com a projectes finals de carrera pels estudiants d'informàtica.

El bloc de contingut principal per aquesta pàgina consisteix en dos blocs composts per petites caixes que contenen una imatge, un títol i una petita descripció de la proposta que representen. Cada una d'aquestes caixes enllaça també amb una pàgina que conté alguns detalls de la proposta.

El primer bloc de caixes representa les propostes generades pels futurs estudiants, mentre que el segon bloc està format per les propostes relacionades amb els exemples implementats.

Les principals diferencies entre les versions d'escriptori i dispositius mòbils, és que la segona presenta un títol simplificat, la desaparició del subtítol, la barra de navegació adaptada i diferent nombre de caixes per fila segons el dispositiu utilitzat. Tres columnes per escriptoris, dues per tauletes gràfiques i una per mòbils.

**Detalls específics d'una proposta**

Com bé indica el nom de la secció, aquesta pàgina mostra els detalls específics de la proposta seleccionada des de la pàgina propostes.

Les diferents propostes són totes generades des del mateix document HTML. És en aquest cas el servidor, l'encarregat d'enviar un conjunt d'informació diferent segons la proposta que ha estat seleccionada. Veurem en més detall com aquest procés funciona en futurs apartats d’aquesta secció.

El bloc del contingut principal per cada una de les propostes, està conformat per una breu descripció del projecte i en alguns casos, certs exemples, preguntes o possibilitats d’extensió, que la proposta pot abordar. Les propostes també disposen d’una petita valoració numèrica, que representa una valoració de la dificultat d’execució de la proposta. Com més gran és el valor, més complexa.

La versió d'escriptori i mòbil no es diferencien en grans aspectes excepte en l'adaptació del contingut a la pantalla del dispositiu i els típics canvis esmentats en els apartats anteriors sobre el títol, subtítol i barra de navegació.

**Identificació amb FamilySearch**

Aquesta pàgina s'utilitza per assegurar que l'usuari no pot utilitzar els exemples sense identificar-se abans amb l’API de FamilySearch.

La pàgina apareix quan l'usuari intenta accedir a la pàgina d'exemples o la pàgina d'un exemple en concret, però encara no s’ha identificat amb FamilySearch. La pàgina permet dues accions simples, tornar enredera (o a la home si s'ha accedit a la pàgina mitjançant la introducció directa de l'URL) o identificar-se amb FamilySearch.

El procés d'identificació s'inicia mitjançant el llançament d'un pop-up, que obre la pàgina de FamilySearch. Aquesta demana la introducció del nom d'usuari i contrasenya. Un cop aquesta informació ha estat verificada, el servidor redirigeix a l'usuari a la pàgina que havia demanat accedir.

La pàgina d'identificació tampoc pateix cap reestructuració especial del contingut quan es veu amb dispositius més petits. Simplement, s'adapta a la pantalla que la mostra i reorganitza els botons que permeten tornar endarrere o obrir el procés d’identificació.

**Exemples Implementats**

La pàgina d'exemples implementats permet a l'usuari descobrir les diferents eines que han estat implementades, per demostrar possibles interaccions amb l’API de FamilySearch i accedir a cada una d'elles.

El bloc de contingut principal, segueix un estil molt similar a la pàgina propostes de projecte, descrita tres apartats endarrere. Aquesta, mostra per cada un dels exemples implementats un títol, una breu descripció i permet a l'usuari navegar cap a les pàgines que contenen les implementacions concretes.

Les principals diferencies entre les versions d'escriptori i dispositius mòbils són exactament les mateixes que per la pàgina de propostes de projecte. És a dir, títol adaptat, eliminació del subtítol, adaptació de la barra de navegació i nombre de propostes per fila segons la grandària del dispositiu.

Realment, aquestes dues pàgines tenen un comportament tècnic idèntic i l'únic que les diferencia és el concepte semàntic que representen i en conseqüència, el contingut.

**Funcionalitats de cerca, expansió geogràfica d'un cognom i evolució d'esdeveniments**

Aquestes pàgines segueixen el mateix patró que la gran majoria de pàgines web de l’aplicació. Recordem que l'esquelet d’aquestes pàgines es mostrava en la figura [ref] d'aquesta mateixa secció de la memòria.

La gran diferencia, d'aquestes pàgines amb la resta és que la part del contingut principal és relativament més complexa i diferent per cada una d'elles. És per aquest motiu, que el comportament exacte de cada una d'aquestes pàgines serà exposat per separat a la secció onze de la memòria.

Pel que fa a l'estructura en comú que comparteixen amb la resta de pàgines, les diferències entre la visualització entre dispositius mòbils i escriptori són les ja conegudes: minimització del títol, desaparició del subtítol i adaptació de la barra de navegació.

**Els fitxers de l'aplicació i les seves funcions**

En aquest apartat de la memòria volem presentar l'arbre d'arxius generats per tal de programar la pàgina web i exposar la funció que desenvolupa cada un d'ells en el marc de l'aplicació.

Recordem que tot el codi programat per tal de fer funcionar l’aplicació web pot ser trobat a l'URL:

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

El conjunt de fitxers programats, representa un total de [línies codi] línies de codi, de les quals un XX% són codi HTML, un XX% codi Javascript i un X% codi CSS.

La taula [ref] mostra la localització relativa de cada arxiu i en descriu breument la seva funcionalitat.

|  |  |
| --- | --- |
| **Arxiu** | **Funció** |
| app.js | Servidor Node.js. Aquest fitxer s'encarrega de gestionar totes les peticions HTTP i HTTPS i de configurar l'aplicació web. També manté la traçabilitat sobre l'estat actual de l'usuari i gestiona els ports de l'aplicació. |
| package.json | Conté metadades sobre l’aplicació web així com les dependències d’aquesta. S'utilitza sobretot per indicar al servidor Node.js quins paquets han de ser instal·lats per assegurar el correcte funcionament de l'aplicació. |
| Procfile | Fitxer utilitzat per indicar al proveïdor del servei d'hostalatge, Heroku, el tipus d'aplicació a configurar i el fitxer a executar per iniciar el servidor. |
| README.md | Breu fitxer que descriu el motiu de creació del projecte i l'ús de cada un dels fitxers que el conformen. |
| views/background.html | Fitxer HTML utilitzat per pintar la pàgina rerefons. |
| views/exemples.html | Fitxer HTML utilitzat per pintar la pàgina d'exemples. |
| views/facts.html | Fitxer HTML utilitzat per pintar la funcionalitat evolució d'esdeveniments. |
| views/index.html | Fitxer HTML utilitzat per pintar la pàgina inicial de l'aplicació. |
| views/login.html | Fitxer HTML utilitzat per pintar la pàgina d'identificació amb FamilySearch. |
| views/proposals.html | Fitxer HTML utilitzat per pintar la pàgina que recull el conjunt de propostes de projecte per futurs estudiants. |
| views/proposalsTemplate.html | Fitxer HTML utilitzat per pintar els detalls d'una proposta en concret. El contingut és variable segons la proposta seleccionada per l'usuari. |
| views/search.html | Fitxer HTML utilitzat per pintar la funcionalitat de funcionalitat de cerca. |
| views/surnames.html | Fitxer HTML utilitzat per pintar la funcionalitat d'expansió geogràfica d'un cognom. |
| views/globals/footer.html | Fitxer HTML encarregat de pintar el footer de la pàgina (per aquelles pàgines que el contenen). |
| views/globals/future-proposals.html | Fitxer HTML encarregat de pintar les propostes de projecte per futurs estudiants a la pàgina de propostes. |
| views/globals/header.html | Fitxer HTML encarregat de pintar l'encapçalament de la web en totes les pàgines. |
| views/globals/implemented-proposals.html | Fitxer HTML encarregat de pintar les caixes dels exemples implementats en les pàgines de recopilació de propostes i exemples. |
| views/globals/javascripts.html | Fitxer HTML encarregat d’inserir els Javascripts comuns a totes les pàgines a cada pàgina. |
| views/globals/navbar.html | Fitxer HTML encarregat de pintar la barra de navegació a cada pàgina en què vol ser mostrada. |
| views/globals/pageTitle.html | Fitxer HTML encarregat de pintar la capçalera de secció de cada pàgina. Els continguts d'aquesta varien segons la pàgina seleccionada, mitjançant els paràmetres enviats pel servidor. |
| views/globals/searchFather.html | Fitxer HTML encarregat de pintar el formulari de cerca, pels camps del pare, a l'exemple de cerca. |
| views/globals/searchMainPerson.html | Fitxer HTML encarregat de pintar el formulari de cerca, pels camps de l’usuari principal, a l'exemple de cerca. |
| views/globals/searchMother.html | Fitxer HTML encarregat de pintar el formulari de cerca, pels camps de la mare, a l'exemple de cerca. |
| views/globals/searchSpouse.html | Fitxer HTML encarregat de pintar el formulari de cerca, pels camps de la parella, a l'exemple de cerca. |
| assets/css/style.css | Fitxer CSS encarregat de modificar l'estil dels diferents elements HTML, classes i identificadors. |
| assets/js/client.js | Fitxer Javascript que gestiona les funcions d'identificació, tancament de sessió i establiment de la connexió amb l’API de FamilySearch. |
| assets/js/cookies.js | Fitxer Javascript per gestionar les cookies per la banda del client (navegador). Fitxer de codi obert reutilitzat del web. |
| assets/js/countryParameters.js | Fitxer Javascript per emmagatzemar la informació dels països continguts a cada continent. S'utilitza perquè el servidor pugui enviar al HTML els continguts i generar HTML dinàmic. |
| assets/js/facts.js | Fitxer Javascript encarregat de gestionar totes les interaccions que l'usuari pot realitzar en l'exemple evolució d'esdeveniments i les connexions amb l’API d'aquesta funcionalitat. |
| assets/js/formValidation.js | Fitxer Javascript per escapar els camps dels formularis i validar-ne el contingut. |
| assets/js/gaTagging.js | Fitxer Javascript encarregat de configurar els esdeveniments de Google Analytics generals i llençar les crides de tots els esdeveniments. |
| assets/js/geo-surnames.js | Fitxer Javascript encarregat de gestionar totes les interaccions que l'usuari pot realitzar en l'exemple d'evolució geogràfica d'un cognom i les connexions amb l’API de Familysearch d'aquesta funcionalitat. |
| assets/js/index.js | Utilitzat per llençar la càrrega d'imatges quan l'usuari aterra a la pàgina principal de l'aplicació. |
| assets/js/jquery.preload.min.js | Fitxer Javascript encarregat d'executar la precàrrega dels documents que s'enviïn contra el complement. En la nostra aplicació l'utilitzem per carregar les imatges de les capçaleres de secció abans que l'usuari les demani. (Fitxer de codi obert programat per Ben Lin, 2011) |
| assets/js/login.js | Fitxer Javascript encarregat de gestionar les interaccions d’usuari de la pàgina d'identificació i les connexions amb l’API de Familysearch realitzades des de la pàgina. |
| assets/js/pageTitles.js | Fitxer Javascript que emmagatzema els paràmetres de configuració de totes les capçaleres de secció. El servidor l'utilitza per obtenir els valors específics de cada pàgina i utilitzar-los per personalitzar els continguts del fitxer pageTitle.html |
| assets/js/projectProposals.js | Fitxer Javascript que emmagatzema els detalls de les diferents propostes de projecte. El servidor l'utilitza per obtenir els valors específics de la proposta demanada i utilitzar-los per personalitzar els continguts del fitxer proposalsTemplate.html |
| assets/js/proposalExamplesLinks.js | Fitxer Javascript utilitzat per redirigir de forma correcta les caixes de les pàgines de propostes i exemples. |
| assets/js/search.js | Fitxer Javascript encarregat de gestionar totes les interaccions que l'usuari pot realitzar en l'exemple de cerca i les connexions amb l’API de FamilySearch d'aquesta funcionalitat. |

**Funcionament general de l'aplicació web**

L'objectiu d'aquest apartat és explicar com els diferents components de la web interactuen entre ells per tal de crear l'aplicació web desplegada al núvol.

Com a suport a les explicacions que s'oferiran, la figura [ref] mostra un diagrama amb els components principals de l'aplicació, com interactuen entre ells i els principals formats de dades que intercanvien.

En la imatge es poden observar les capes Servidor, Controlador i Client, que conformen l'arquitectura en tres capes explicada en seccions anteriors. També es pot observar el Javascript SDK, del que es detalla com es comunica amb l'API de FamilySearch i a quina capa de l'aplicació es troba connectat.

**Creació del servidor**

L'aplicació comença a funcionar quan aquesta és executada en els servidors al núvol de Heroku, la plataforma d'hostalatge. La configuració del servidor es realitza mitjançant els fitxers *Procfile* i *package.json*, que s'encarreguen d'assegurar que tots els complements Javascript necessaris siguin instal·lats i s'executi el fitxer *app.js* per arrancar i configurar el servidor.

Quan la configuració del servidor acaba, aquest es troba preparat per començar a rebre peticions dels usuaris.

**Accés a una pàgina del domini web**

Quan un usuari demana carregar la pàgina inicial de la nostra aplicació, una petició *http* o *https* és generada i enviada cap al servidor a través del navegador del client.

Quan el servidor la rep, l'avalua i en cas d'èxit, retorna al client els fitxers processats necessaris perquè el navegador pugui mostrar la pàgina demanada i carregar totes les funcionalitats o interaccions possibles d'aquesta al controlador.

**Interacció amb l’API de FamilySearch**

El moment en què l'usuari vol interactuar amb l'API de FamilySearch, aquest es veu forçat a interactuar primer amb un element del codi HTML per comunicar l’acció a realitzar. Per exemple, el botó de cerca de la funcionalitat d'evolució temporal d'esdeveniments.

Quan el controlador detecta que el botó de cerca ha estat pressionat, captura l'esdeveniment i avalua la petició. En cas que no es detecti cap problema ni error en els paràmetres introduïts, el controlador realitza una crida asíncrona al SDK de FamilySearch i n'espera la resposta.

De forma transparent a l'aplicació web, el SDK es comunica amb l’API de FamilySearch i si no hi ha cap problema en la comunicació i la petició és vàlida, aquesta retorna les dades demanades en format XML o JSON. Posteriorment, el SDK transforma la resposta de l’API en un objecte Javascript amb funcions de conveniència que facilitaran l'accés a les dades de la resposta i el retorna al controlador.

En el moment que el SDK retorna l'objecte, la promesa pendent de resolució que havia creat el controlador és resolta i l'objecte retornat pel SDK passa a ser accessible. Arribats a aquest punt, el controlador processa i transforma les dades contingudes a l'objecte de la forma desitjada i un cop finalitzades les operacions necessàries, modifica la vista del client introduint els canvis pertinents en aquesta.

**Interacció amb elements del HTML**

Quan els usuaris interactuen amb elements bàsics del HTML, per exemple, quan interactuen amb les caixes que permeten expandir o plegar seccions del formulari en les funcionalitats de cerca o evolució geogràfica de cognoms, la resposta a realitzar per part del controlador és bastant simple.

En el moment que el controlador detecta que s'ha interactuat amb algun dels elements que escolta, captura l'esdeveniment, avalua com s'ha de procedir segons el context de l'acció, en el cas de l’exemple, plegar o desplegar contingut d'un formulari i realitza de forma immediata els canvis a la vista del navegador.

**Conclusió**

Els casos d'ús que s'han cobert en els apartats anteriors són relativament simples, però són una mostra representativa del conjunt d'accions diferents a les quals l'aplicació pot haver de fer front.

Esperem que aquesta secció hagi servit per il·lustrar el funcionament general de la web i com els diferents components interactuen entre ells. La resta d’accions possibles en el web, requerint més o menys complexitat per part del controlador, segueixen una de les tres rutes descrites en els apartats anteriors, per tal de ser satisfetes.

**Detalls específics de la implementació**

**Introducció als detalls d'implementació**

L'objectiu d'aquesta secció és destacar alguns dels detalls d'implementació dels fitxers de l'aplicació.

En aquest apartat, no s'exposaran els detalls d'implementació de les funcionalitats integrades amb FamilySearch, doncs aquestes seran explicades, més en profunditat, en el seu propi apartat de la següent secció de la memòria.

Els codis que s'enllaçaran en cada un dels següents apartats es troben molt simplificats per tal de facilitar la comprensió dels punts a explicar. Si es busca una correlació exacta de les línies de codi, en el codi font de l'aplicació, s'observarà que les línies de codi de la memòria són incompletes i manquen de molts paràmetres de configuració.

Cal entendre que una pàgina web està formada per moltes línies curtes i altres de molt llargues. En conseqüència, mostrar el codi complet de cada fitxer, resulta impossible i ineficient, de cara a una memòria. A canvi, s'han seleccionat detalls d'implementació, que permeten explicar les bases sota les quals els diferents arxius funcionen i repliquen diverses vegades, al llarg del seu codi.

Instem als lectors de la memòria, a visitar també els fitxers de codi originals, que poden ser trobats a l'URL ja mencionat diverses vegades, si així ho prefereixen.

**Estructura general d'una pàgina HTML: Mustache I**

A excepció de la pàgina d'identificació amb FamilySearch, la resta de pàgines de l'aplicació web segueixen la mateixa estructura, tal com s'ha exposat en l'apartat Estructura de l'aplicació web.

Per evitar la duplicació de codi entre les diferents pàgines, els elements compartits han estat creats en fitxer separats i són carregats a cada pàgina segons si volem mostrar-los o no. Recordem que la tecnologia utilitzada, que fa això possible, és el llenguatge de plantilles Mustache.

L'esquelet del codi d'una pàgina qualsevol del nostre web segueix la forma mostrada en el bloc de codi que segueix.

*<!-- header includes -->  
{{> header }}  
{{> navbar }}  
{{> pageTitle }}*

*<!-- specific page content -->  
...*

*{{> javascripts }}  
<!-- specific javascripts for this file -->  
...*

*<!-- footer -->  
{{> footer }}*

Les etiquetes {{> `fileName' }} s'utilitzen per indicar que es vol importar, en aquesta posició, el codi HTML del fitxer `fileName.html'.

Mitjançant aquesta estructura tenim el control absolut de quins components volem incloure a cada una de les pàgines. Per exemple, l'esquelet de la pàgina d'identificació es diferencia de la resta en que no s'inclouen els fitxers *navbar, pageTitle* ni *footer*.

També cal destacar que abans del *footer,* deixem un espai en blanc, per incloure fitxers Javascript que són utilitzats únicament per la pàgina carregada, d'aqueta forma s'evita carregar tots els controladors a totes les pàgines, si aquests no han de ser utilitzats.

**El fitxer header.html: Configuració de la web**

Aquest fitxer s'encarrega de configurar la capçalera de les pàgines del nostre domini i obrir el cos del codi. També es declaren en aquesta secció els arxius CSS a carregar i les fonts a utilitzar.

Les línees de codi més interessants són les que configuren el `viewport' del dispositiu perquè aquest s'adapti de forma adequada a qualsevol pantalla i l'etiqueta necessària per indicar al navegador la codificació en que es troben els caràcters (línies 1 i 4 del següent codi).

*<meta charset="utf-8">  
<meta http-equiv="X-UA-Compatible" content="IE=edge">  
<title>PFC - Family Search API Study</title>  
<meta name="viewport" content="width=device-width, initial-scale=1">  
<link href="/node\_modules/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet">  
 <link href="/assets/css/style.css" rel="stylesheet">  
<link href='fontNumber1' rel='stylesheet' type='text/css'>  
<link href='fontNumber2' rel='stylesheet' type='text/css'>***El fitxer navbar.html: La barra de navegació adaptativa**

Com el nom del fitxer indica, aquest s'encarrega de configurar la barra de navegació. Aquesta ha estat creada mitjançant la potencialitat d'un dels components de Bootstrap.

Consta principalment de dos blocs de codi HTML. El primer, s'utilitza per indicar la capçalera de la barra de navegació, que conté el botó per expandir-la en dispositius mòbils, la icona de l'aplicació que és mostrarà nomes en dispositius d'escriptori i el nom de l'aplicació web.

La capçalera adaptativa de Bootstrap per la barra de navegació, s'invoca mitjançant la classe *navbar-header*.

*<div class="****navbar-header****">   
 <****button*** *type="button" class="navbar-toggle collapsed" ...>...</button>  
 <a href="/" ...><****img*** *src="/images/littleIco.png"/></a>  
 <a class="navbar-brand* ***navbar-link****" href="/">FamilySearch - PFC</a>  
</div>*

El segon bloc, s'encarrega de generar els enllaços a les diferent seccions de l'aplicació i configurar el botó dedicat al tancament de la connexió amb FamilySearch, si aquesta està oberta.

Cada enllaç és estilitzat i configurat mitjançant la classe *navbar-link,* es decideix si aquesta enllaç ha d'aparèixer per l'esquerra o dreta de la barra de navegació, segons si el contenidor pare dels enllaços, conté la classe *navbar-right.*

*<div class="collapse navbar-collapse" id="...">  
 <ul class="nav navbar-nav">  
 <li><a class="****navbar-link****" href="****/background****">Background</a></li>  
 <li><a class="****navbar-link****" href="****/proposals****">Proposals</a></li>  
 <li><a class="****navbar-link****" href="****/examples****">Examples</a></li>  
 </ul>  
 <ul class="nav navbar-nav navbar-right">  
 <li><a id="****signOut****" class="****navbar-link****" href>Sign Out</a>  
 </ul>  
</div>*

**El fitxer pageTitle.html: Mustache II**

El fitxer pageTitle resulta un dels fitxers HTML més interessant de l'aplicació. Com s'ha comentat en el primer apartat d'aquesta secció, casi totes les pàgines de l'aplicació contenen el bloc de codi que hem anomenat `capçalera de secció'.

Aquesta capçalera consta d'una imatge de fons, amb un títol i subtítol solapats segons el dispositiu des del que s'accedeix a la pàgina. Els paràmetres que marquen la imatge de fons, els texts, el color del ressaltat i si cal mostrar un botó de navegació o no, són enviats pel servidor i carregats de forma dinàmica segons la pàgina visitada.

Recordem que les etiquetes {{ nomEtiqueta }} són referències a codi Mustache i representen paràmetres dinàmics emplenats pel servidor abans de servir el HTML. En el bloc de codi que es mostrarà més endavant, es podran veure importats els paràmetres: backgroundImage, highlight, title, subtitleDesktop, subtitleTabet i button.

Fixem-nos també en que aquests paràmetres es poden importar en qualsevol lloc del codi HTML. En alguns cassos, s'importen com a clase d'un element HTML, indicant-ne l'estil i comportament esperats d'aquest i altres, simplement es tracte de contingut estàtic, com per exemple, el títol.

Destacar també, que pot resultat interessant fixar-se en com l'aplicació diferència entre si s'ha de mostrar el subtítol per escriptori o tauleta gràfica, ja que la lògica és la mateixa per qualsevol altre component del domini web.

Els encarregats de gestionar aquest aspecte són les etiquetes de classe: *hidden-sm* i *hidden-xs* en els subtítols d'escriptori i en contrapartida, les etiquetes de classe: *visible-sm*, en els subtítols de la tauleta gràfica. Aquestes etiquetes poden ser llegides de forma semàntica com: Si la pantalla que mostra la pàgina web és petita (tauletes gràfiques) o ultra reduïda (mòbils), no mostris el subtítol per escriptoris i si el de tablet.

Així doncs, el següent bloc de codi representa la configuració de la capçalera de secció per dispositius de pantalla petita o superiors (fixem-nos en l'etiqueta de classe de la primera línea que només amaga la secció per dispositius *xs*).

<div class="container-fluid **{{backgroundImage}}** hidden-xs">  
 *<!-- main title -->  
 <div class="row"><h1 class="****{{highlight}}****">****{{title}}****</h1></div>  
  
 <!-- subtitle -->  
 <div class="row">  
 <!-- Subtitles: Desktop -->  
 <div class="col-md-12 hidden-sm hidden-xs">  
 <h2 class="****{{highlight}}*** *text-italic">****{{subtitleDesktop}}****</h2>  
 </div>  
 <!-- Subtitles: Tablet -->  
 <div class="col-md-12 visible-sm">  
 <h2 class="****{{highlight}}*** *text-italic">****{{subtitleTablet}}****</h2>  
 </div>  
 </div>  
   
 <!-- display button if required -->*  ***{{#button}}*** *...* ***{{/button}}****</div>*

Al final del bloc de codi mostrat, en el fitxer original, s'inicia una capçalera similar per dispositius de pantalla extra reduïda. El motiu pel que creem dues capçaleres diferents és per mostrar una capçalera més petita en alçada i aprofitar així millor l'espai disponible en dispositius mòbils.

**El fitxer javascripts.html: Càrrega dels controladors**

Aquest fitxer s'encarrega de carregar tots els scripts comuns necessaries a les pàgines del nostre domini web.

El fitxer no representa cap misteri, excepte que l'ordre de declaració és important, per assegurar que no es carrega codi jQuery sense carregar abans el propi complement.

El fitxer càrrega els complement jQuery, SDK de FamilySearch, fitxers Javascripts encarregats de manipular les galetes, la declaració de l'objecte FamilySearch i control de sessió, les interaccions pels components de Bootstrap i el snippet de codi de Google Analytics, que s'encarrega de monitorar el tràfic de l'aplicació web.

Les úniques línies de codi interessants del fitxer són les encarregades de configurar el compte de Google Analytics i monitorar la pàgina visitada. Aquesta pàgina, també emmagatzemarà de forma automàtica, tota la informació relativa a la sessió d'usuari.

*ga('create', 'UA-80847078-1', { 'cookieDomain' : '...'});  
 ga('send', 'pageview');*

**El fitxer index.html: El grid de Bootstrap**

Aprofitarem aquest fitxer per explicar en profunditat el funcionament del grid de Bootstrap, introduït ja en la secció `Estudi tècnic del projecte.'

El contingut principal del fitxer *index.html*, representa un conjunt d'enllaços als tres grans blocs de l'aplicació web, on cada un, és veu representat per una imatge, un títol i una descripció. En aplicacions d'escriptori, aquests tres blocs es mostren un al costat de l'altre, mentre que en aplicacions de pantalles més reduïdes, es mostren un sobre l'altre.

Per aconseguir-ho, es juga amb el concepte de files i columnes del grid de bootstrap. El grid, està format per contenidors o blocs d'espai, en els que es poden crear diferents fileres i cada una d'aquestes fileres, pot ser dividida en dotze columnes.

Com que en la versió d'escriptori volem crear tres blocs idèntics, on cada un contindrà un enllaç a una secció de la pàgina web, a cada un dels blocs li corresponen 12/3 = 4 columnes. Per indicar-ho a bootstrap, s'utilitza la classe *col-md-4*.

Així doncs, el codi extremadament simplificat d'aquesta secció podria ser representat de la següent forma:

*<div class="container">*  
 <div class="row">  
 <div class="**col-md-4"**> ... </div>  
 <div class="**col-md-4"**> ... </div>  
 <div class="**col-md-4**"> ... </div>  
 </div>  
</div>

La configuració mostrada en el bloc de codi anterior representaria exactament el format que volem que la pàgina agafi per escriptoris, però perquè aquesta estructura funciona també ne els dispositius mòbils apilant-ne els blocs de quatre columnes un sobre l'altre?

Per comprendre-ho hem d'explicar primer que bootstrap categoritza les aplicacions en quatre grandàries, segons l'amplada en píxels de la pantalla dels dispositius:

* **Dispositius extra reduïts (xs):** Fa referència als dispositius mòbils o pantalles amb amplitud inferior als 768 píxels.
* **Dispositius petits (sm):** Fa referència sobretot a tauletes gràfiques o pantalles amb amplitud superior o igual a 768 píxels, però inferior a 992 píxels.
* **Dispositius mitjans (md):** Fa referència a escriptoris o pantalles amb amplitud superior o igual als 992 píxels i inferior als 1200 píxels.
* **Dispositius grans (lg):** Fa referència a escriptoris grans o pantalles amb amplitud superior o igual als 1200 píxels.

Les lletres que s'han indicat entre parestèsies en el llistat anterior, són un codi que s'introdueix en la classe columna de bootstrap (col-xx-4), per indicar sobre quina grandària mínima de dispositiu, s'ha d'aplicar la regla.

D'aquesta forma, la classe utilitzada en el bloc de codi d'exemple (col-md-4), indica que es vol declarar un bloc de quatre columnes per dispositius mitjans o superiors. Els dispositius que no compleixin aquesta regla, és a dir, els dispositius petits o extra reduïts, veuran transformada la classe de forma automàtica a *col-sm-12* o *col-xs-12* respectivament.

En altres paraules, en els dispositius petits i extra reduïts, cada bloc passarà a ocupar dotze columnes, forçant així que cada bloc ocupi tot l'espai disponible en una fila i per tant, desplaçarà els altres blocs a noves files, creant aquest afecte d'apilament.

**El fitxer background.html: El grid de bootstrap II**

Utilitzarem aquest fitxer per explicar una altra funcionalitat interessant del grid de Bootstrap.

Degut a que aquesta pàgina està formada únicament per text, volem intentar augmentar la llegibilitat d'aquesta. Existeixen diversos estudis que demostren que l'esser humà llegeix, amb més comoditat, línees curtes que contenen entre 45 i 75 caràcters.

Per aquest motiu, en aquesta pàgina que tot el contingut principal és text, hem volgut reduir l'amplada màxima utilitzada i en comptes d'utilitzar les dotze columnes que el grid permet, hem decidit que utilitzar-ne només vuit.

Per tal que el contingut no se'ns quedi descentrat, és a dir, ocupant vuit columnes des de l'esquerra de la fila i deixant-ne quatre en blanc a la dreta, utilitzem una classe especial de Bootstrap que permet deixar columnes en blanc al començament d'una filera. Aquesta classe segueix la forma: *col-SIZE-offset-NUMBER* i es regeix per les mateixes regles que les explicades a l'apartat anterior.

Això significa que si declarem una regla per dispositius *md,* aquesta no aplicarà per dispositius de grandària inferior. Garantint que el text segueixi ocupant tota l'amplada possible, en dispositius més petits.

Així doncs, si volem centrar un contingut de vuit columnes en un fila de dotze columnes, hem de deixar dues columnes en blanc a cada banda del text. La línea de codi que garanteix aquesta visualització és mostra a continuació:

*<div class="col-md-8 col-md-offset-2"> ... </div>*

**Fitxers future-proopsals.html i implemented-proposals.html: El grid de bootstrap III i el component Thumbnail**

Aquests dos fitxers són utilitzats per crear el contingut principal de la pàgina propostes i exemples de l'aplicació web. En concret, s'encarreguen de crear les caixes formades per una imatge, un títol i una descripció, que representen una proposta de projecte o un projecte implementat.

Aquestes caixes, originalment tres per fila, s'adapten a dos per fila en tauletes gràfiques i a un per fila en dispositius mòbils. El comportament s'aconsegueix de forma similar al explicat en el fitxer index.html, però en aquest cas, considerant tres formats diferents en comptes de dos. Això és pot aconseguir mitjançant la línea de codi:

*<div id="proposal-box-1" class="****col-md-4 col-sm-6****"> ... </div>*

D'aquesta forma, indiquem que en dispositius mitjans o més grans, volem que la capsa ocupi quatre de les dotze columnes, en dispositius petits, sis columnes i en dispositius extra reduïts, per omissió, dotze columnes.

Les caixes que conformen cada una de les propostes, han estat generades mitjançant el component Thumbnail de Bootstrap. Aquests es caracteritzen per utilitzar imatges que s'adapten a la grandària del contenidor *(img)* i la possibilitat d'incloure un títol i descripció a cada caixa *(caption)*. En el següent bloc de codi, és mostra l'esquelet principal d'una d'aquestes caixes.

***<div class="thumbnail">*** <!-- image --> *<****img*** *src="/images/thumbnails/search-min.png">  
 <!-- title + text -->  
 <div class="****caption****">  
 <h3> ... </h3>  
 <p> ... </p>  
 </div>  
</div>*

**El fitxer package.json: Complements de l'aplicació**

El fitxer package.json s'utilitza per configurar l'aplicació web quan aquesta és desplegada al núvol. La part del codi més interessant és la que específica les dependencies de l'aplicació i que dictamina els components que seran instal·lats, per tal de garantir el correcte funcionament d'aquesta.

*"dependencies": {  
 "body-parser": "^1.15.2",  
 "bootstrap": "3.3.6",  
 "cookie-session": "2.0.0-alpha.1",  
 "express": "4.14.0",  
 "mustache-express": "1.2.2"  
 }*

Els diferents paquets compleixen les següents funcionalitats:

* **Body-parser:** Utilitzat per capturar els paràmetres enviats des del frontal al servidor.
* **Bootstrap:** Com ja s'ha comentat, utilitzat per controlar l'estructura de les pàgines i la utilització de components genèrics.
* **Cookie-session:** Utilitzat per crear galetes de sessió no editables i signades, per tal d'evitar atacs a la seguretat dels usuaris.
* **Express:** Framework sobre el que es programarà l'aplicació Node.js
* **Mustache-express:** Càrrega del llenguatge de plantilles pel framework Express.

**El fitxer app.json: El funcionament del servidor**

Introducció

El fitxer *app.json* s'encarrega de configurar el servidor i gestionar les peticions HTTP, HTTPS i AJAX provinents del client. És un dels fitxers principals de l'aplicació i probablement un dels més interessants.

El fitxer es divideix en diferents seccions:

* Creació de variables
* Configuració del motor d'impressió i carpetes
* Configuració dels complements
* Funcions de redirecció
* Processament de peticions 'POST'
* Validació d'identificació
* Configuració del servidor

Creació de variables

En aqueta secció és declarà la utilització dels diferents complements i la creació de l'aplicació mitjançant el framework Express.

També es creen instàncies dels objectes amb funcions de conveniència emmascarats en els fitxers *projectProposals.js, pageTitles.js i countryParameters.js*

Configuració del motor d'impressió i carpetes

Aquest punt esdevé important, ja que indica a l'aplicació quina mena de fitxers ha de renderitzar. En el nostre cas, calia indicar a l'aplicació que els fitxers a pintar en el client eren del format HTML i que s'utilitzava el motor de plantilles Mustache com a complement.

*app.set('view\_engine', 'html');  
app.engine('html', mustacheExpress("views/globals", ".html"));*

De cara a que el servidor sigui capaç de resoldre on es troben localitzats els diferents recursos o fitxers, cal també indicar-ne la ruta de forma relativa a la del servidor. Així doncs, s'emmascara la carpeta *views, assets, node\_modules i images* de l'aplicació de la forma següent:

*app.set('views', path.join(\_\_dirname, 'views'));  
app.use('/assets', express.static(path.join(\_\_dirname, 'assets')));  
app.use('/node\_modules', express.static(path.join(\_\_dirname, 'node\_modules')));  
app.use('/images', express.static(path.join(\_\_dirname, 'images')));*

Configuració dels complements

En aquest apartat és configuren dos complements. El encarregat de gestionar les galetes (cookie-session) i el encarregat d'interpretar els paràmetres enviats al servidor des del client (body-parser).

El complement cookie-session no cal configurar-lo gaire, doncs els paràmetres per defecte ja marquen les galetes a crear com a segures (transmesa si és possible a través de https) i httpOnly (no modificable pel client). Per tant, només hem d'indicar-ne el nom de la galeta i un parell de claus amb les que volem firmar i xifrar el contingut d'aquestes.

*app.use(cookieSession({  
 name: 'session',  
 keys: ['misaholdrin', 'tommarvoloriddle']  
}));*

Per altre banda, el complement bodyParser, es preparat per rebre paràmetres de la URL i descodificar paràmetres JSON.

*app.use(bodyParser.urlencoded({ extended: true }));  
app.use(bodyParser.json());*

Funcions de redirecció

Aquesta secció del fitxer s'encarrega de decidir i configurar els fitxers HTML que han de ser retornats segons les peticions realitzades pels usuaris al servidor.

La resposta a totes les peticions GET dels clients són solucionades de forma similar. Es capturen, després es recullen tots els paràmetres necessaris per emplenar els paràmetres amb Mustache i s'envia el HTML processat cap al client.

*app.get('/', function(req, res){  
var params = pageTitlesIns.getTitle('index');  
res.render('index.html', {  
 backgroundImage: params[0],  
 highlight: params[1],  
 title: params[2],  
 titleMobile: params[3],  
 subtitleDesktop: params[4],  
 subtitleTablet: params[5],  
 button: params[6],  
 buttonHref: ''  
 });  
});*

El codi anterior representa la resposta a la petició d'accés a la pàgina inicial de l'aplicació (/).

Un cop es captura la petició GET, denotada pel URL `/', és demana a l'objecte Javascript `pageTitlesIns', carregat a l'inici del servidor, els paràmetres a utilitzar per pintar el fitxer *pageTitle.html*. Un cop rebuts aquests paràmetres, es processa el fitxer html *index.html* amb els paràmetres desitjats i s'envia al client.

Altres peticions GET més complexes, com per exemple, la de pintar els detalls d'una proposta de projecte específica, funcionen de forma similar però carregant més paràmetres a part dels de l'objecte *pageTitleIns*. Aquest cas, conté una excepcionalitat més i és que es captura part de l'URL, per utilitzar-la com a paràmetre, tal com s'indica en el següent bloc de codi.

*app.get('/proposals/****:project****', function(req, res) {   
 var params = pageTitlesIns.getTitle('index');  
 var proposal = projectProposalsIns.getExample(****req.params.project****);  
 });*

Processament peticions POST

Les dues úniques peticions POST que l'aplicació està preparada per acceptar i respondre, són les crides AJAX que serveixen per indicar al servidor que l'usuari s'ha identificat o desconnectat de l'API de FamilySearch.

Quan és rep una d'aquestes peticions, s'actualitza la galeta de sessió amb el token retornat per l'API de FamilySearch o se n'esborra el contingut, segons la petició tractada. Un cop processada la informació, s'envia un paràmetre *redirect* cap al navegador, que indica a la capa del controlador a quina pàgina s'ha de redireccionar a l'usuari de l'aplicació.

És mostra a continuació, com a exemple, el processament de la petició POST d'identificació.

app.post('/token/login', function(req, res) {  
 req.session.logged = req.session.logged || req.body.token;  
 res.end('{"redirect" : "/examples"}');  
});

Validació d'identificació

La validació d'identificació és una funció que s'utilitza per comprovar si un usuari disposa dels permisos suficients per visitar les pàgines d'exemples implementats.

Quan el servidor rep la petició de mostrar la pàgina d'exemples o un exemple en concret, realitza una crida a aquesta funció, que resoldrà si la petició ha de ser processada o no. Aquesta funció (isAuthenticated), s'insereix com a middleware en la capçalera de la petició de la següent forma:

***app.get('/examples', isAuthenticated, function(req, res) { ... });***

La funció isAuthenticated comprova si la galeta *session* està configurada o no. En cas afirmatiu, es processa la petició inicial de l'usuari i en cas contrari, es mostra la pàgina d'identificació.

*function isAuthenticated(req, res, next) {  
 if(req.session.isPopulated) next();  
 else res.redirect('/login');  
}*

Configuració del servidor

La part de configuració del servidor consisteix, principalment, en indicar quin port ha de ser escoltat. La línea de codi utilitzada, permet alternar entre un valor per defecte o el especificat en els paràmetres de configuració de l'entorn que desplegui el servidor.  
 *var server = app.listen(process.env.PORT || 8080, process.env.IP, function () { ... });*

**El fitxer gaTagging.js: Enviant esdeveniments**

Aquest fitxer s'encarrega de configurar els esdeveniments generals que seran enviats a Google Analytics, però l'aspecte més interessant que conté, és la funció genèrica que s'invoca per enviar esdeveniments des de qualsevol punt de l'aplicació.

Els esdeveniments de Google Analytics estan formats per quatre paràmetres: *categoria, acció, etiqueta i valor*, on els paràmetres *etiqueta* i *valor* són opcionals. El sistema plantejat, ofereix la possibilitat de que el paràmetre *categoria,* sigui omplert automàticament, amb el path de la pàgina que dispara l'esdeveniment.

En el bloc de codi que es mostra a continuació es poden observar dues funcions. La funció, *getCategory,* és l'encarregada d'emplenar el camp *categoria* amb la pàgina i la funció, *sendEvent,* és la funció que s'invoca per disparar tots els esdeveniments de l'aplicació web, de forma centralitzada i cridar a la funció *getCategory* en cas de necessitat.

*function* ***getCategory****() {  
 var currentPage = location.pathname.split('?')[0].slice(1);  
 return currentPage = currentPage != '' ? currentPage : 'home'  
}*

*function* ***sendEvent****(category, action, label, value) {  
 category = category != '' ? category :* ***getCategory()****;   
 ...* ***ga('send', 'event', category, action, label)****;  
}*

**El fitxer style.css: Configurant elements propis**

El fitxer style.css ha estat utilitzat principalment per controlar l'aparença de certs elements mostrats a l'aplicació web.

Els dos element que més han estat manipulats són la font associada a cada element HTML, degut a que les fonts per defecte de Bootstrap són bastant simples i manquen de personalitat, i la creació d'unes noves classes destinades a controlar la distància entre línees del grid de Bootstrap, que per defecte, mostra les diferents línies del grid pràcticament enganxades i com a conseqüència, la pàgina web no respira.

A continuació, mostrem un exemple d'assignació de font als elements HTML <h1> .. <h6> i algun exemple dels espaiadors de classe, que hem creat, seguint la nomenclatura Bootstrap.

*// Fonts for html <h1> .. <h6>  
h1, h2, h3, h4, h5, h6 {  
 font-family: 'Lora', serif;  
 font-weight: 400;  
 line-height: 1.8em;  
}*

*// Spacers  
.xs-buffer { margin-top:10px; }  
.sm-buffer { margin-top:20px; }  
...  
.xxxl-buffer { margin-top: 60px; }*

A part, també s'han creat diferents classes per tal de controlar aspectes com la dissociació dels controls de les funcionalitat d'exemple, de la seva posició fixa, carregar les diferents imatges de fons en les capçaleres de secció, canvia l'estil dels enllaços URL, controlar la visibilitat inicial de certes zones del HTML que volem mantenir ocultes, estils de les taules, barres de progrés, camps del formulari, etcètera, etcètera.

Es recomana donar un cop d'ull al fitxer original: *style.css,* per tenir una idea més global de totes les petites configuracions que aquest realitza.

**Certificant l'aplicació amb Familysearch**

Com s'havia comentat en la quarta secció d'aquesta memòria, en la que s'introduïa l'API de FamilySearch, per tal d'obtenir accés a les dades de producció cal certificar les aplicacions.

El procés de certificació, pot ser vist com una validació de l'aplicació per part de l'organització FamilySearch, per assegurar que no es realitza cap operació que pugui afectar al rendiment de l'API, la integritat de les dades o la seguretat del sistema.

En el cas de la nostra aplicació, el conjunt de passos a realitzar per tal de certificar l'aplicació web s'exposen a continuació:

1. Aplicar l'aplicació per certificació des de la secció `les meves aplicacions' del portal de desenvolupadors de FamilySearch.
2. Completar, signar i retornar l'acord d'afiliació de producte, les regles de seguretat i la petició d'obtenció d'una clau d'accés a producció.
3. Registrar l'aplicació a FamilySearch i monitorar el procés de certificació.
4. L'aplicació serà avaluada de diferents formes segons els diferents certificats que es vulguin obtenir. Aquests certificats van relacionats amb les operacions que realitza l'aplicació contra l'API.

Pel fet que la nostra aplicació s'identifica amb l'API de FamilySearch i realitza operacions de lectura sobre aquesta, són dos els certificats pels quals l'aplicació hauria de ser certificada. El certificat d'autentificació i el certificat de lectura.

**Certificat d'autentificació**

El primer certificat que l'aplicació desenvolupada requereix és el relacionat amb el procés d'identificació dels usuaris a l'API de FamilySearch.

La llista de consideracions a tenir en compte per obtenir el certificat són:

* Cada usuari ha d'adquirir un token d'identificació propi per tal de llegir dades de l'arbre familiar de FamilySearch.
* Els tokens d'identificació han de ser protegits. En cas que es vulgui emmagatzemar un token en una galeta del navegador, aquesta ha de ser una galeta segura.
* El tràfic és xifrat mitjançant el protocol SSL des de l'usuari fins a l'API de FamilySearch.
* L'autentificació de l'usuari és completada mitjançant la crida directa als protocols d'identificació OAuth 2 de FamilySearch. No es permet als tercers emmagatzemar informació sobre els noms d'usuari, contrasenyes i identificadors de sessió. En una aplicació web, l'autentificació pot ser realitzar, mitjançant la crida a un pop-up, al protocol d'autentificació de FamilySearch.

Donat que l'aplicació utilitza els serveis oferts pel SDK oficial de Javascript, de cara a l'autentificació, aquesta operació no hauria de presentar molts problemes de cara a obtenir el certificat.

**Certificat de lectura**

El segon certificat que l'aplicació requereix va relacionat amb les operacions de lectura que realitza contra l'API de FamilySearch. Les principals restriccions a tenir en compte de cara a obtenir el certificat de lectura s'exposen a continuació:

* Procés d'autentificació certificat.
* Demostració de la correcta implementació de diferents funcions de lectura.
* Demostració de l'ús correcte de la cache de FamilySearch.
* Les aplicacions han de guiar a l'usuari a l'hora d'utilitzar les funcionalitats i ajudar-los a superar els possibles errors.
* Les aplicacions han de tenir en compte els estàndards del mercat a l'hora d'evitar atacs a la seguretat mitjançant injeccions de codi i altres vulnerabilitats.
* Complir amb les bones pràctiques de seguretat:
  + Només està premés mostrar informació regulada de persones vives a les persones autentificades amb FamilySearch.
  + La informació de persones difuntes no regulada pot ser mostrada a qualsevol usuari.
  + Les dades locals emmagatzemades durant la sessió en el navegador, han de ser eliminades al final d'aquesta.
  + Són permeses les tasques d'elevat temps de processat sempre i quan es compleixin certes regulacions.
  + Les aplicacions poden guardar informació genealògica, de persones difuntes, obtinguda a través d'usuaris identificats amb FamilySearch.
  + Les aplicacions poden guardar, però no fer pública, informació genealògica de persones vives.
  + Les aplicacions poden emmagatzemar els identificadors de les persones, amb dades regulades, però no les dades regulades per se.
  + Les aplicacions no poden emmagatzemar relacions concretes que indiquin que una persona va trobar-se, en un lloc concret, en una data concreta.

Un altre cop, pel fet d'utilitzar el SDK oficial de Javascript, molts dels punts principals necessaris per obtenir la certificació de lectura haurien d'estar coberts. Un exemple clar, és el de la correcta implementació de les funcions de lectura. Donat que aquestes no són realment controlades per nosaltres, no hi hauria d'haver cap problema.

Pel que fa a les bones pràctiques de seguretat, s'han utilitzat precaucions per evitar problemes com la injecció de codi a través dels formularis de l'aplicació i la nostra aplicació no emmagatzema cap mena d'informació de l'usuari, ni de les dades retornades, més enllà del token d'identificació.  
  
**Procés de certificació**

**Google Analytics**

Google Analytics és un servei d'analítica web, proporcionat per Google, que s'encarrega de monitorar i reportar dades relatives al tràfic d'una aplicació web o aplicació mòbil.

Mitjançant una implementació relativament simple, és possible emmagatzemar informació relativa a les pàgines visitades, la navegació entre pàgines, els sistemes operatius utilitzats, informació sobre els diferents navegadors, resolucions de pantalla, dispositius mòbils utilitzats i interaccions bàsiques dels usuaris amb les diferents pàgines del web.

Gran part d'aquesta informació és capturada de forma automàtica, per Google Analytics, pel simple fet d'incloure el `snippet' de codi de l'eina a les nostres pàgines del domini web. A canvi, es disposa de moltes variables que poden ser creuades per tal d'analitzar el rendiment del domini web i l'ús que li donen els usuaris.

De totes maneres, no s'espera que la nostra aplicació web disposi de grans quantitats de tràfic i la implementació de Google Analytics no ha vingut donada pel fet de poder analitzar el rendiment tècnic o d'usabilitat de l'aplicació, sinó per poder controlar el funcionament de la integració amb l'API de FamilySearch.

Durant el desenvolupament de l'aplicació, mentre interactuàvem amb l'entorn sandbox de l'API, ens vàrem adonar que durant diversos dies i inclòs a vegades, períodes de tres o quatre dies, l'entorn no funcionava i les peticions llençades contra l'API eren cancel·lades.

Per tal de poder monitorar en tot moment el funcionament de les interaccions dels usuaris amb l'API, s'ha utilitzat la funcionalitat de Google Analytics que permet enviar esdeveniments personalitzats, que o bé reflecteixen accions dels usuaris o condicions que s'han donat en l'aplicació web o mòbil.

En concret, s'han creat quatre nivells d'esdeveniments diferents per cada una de les funcionalitats principals que interactuen amb l'API de FamilySearch

* **Formulari incorrecte:** En el cas que un usuari intenti llençar una petició contra l'API i aquesta no s'iniciï perquè el formulari contenia errors, es marca l'intent amb una etiqueta de formular incorrecte.
* **Petició llençada:** Si la validació de tots els camps és correcta, s'envia un esdeveniment indicant que un intent de connexió amb l'API, s'ha llençat amb el SDK de FamilySearch i els paràmetres d'aquesta petició.
* **Petició rebutjada:** En cas que el SDK no pugui resoldre la petició per qualsevol motiu, s'enregistra un esdeveniment que indica el rebuig de la petició i n'especifica el motiu (Per exemple, timeout, no existeix el recurs, excés de peticions en un període limitat de temps, etcètera).
* **Petició retornada amb èxit:** Quan el SDK processa la petició i retorna resultats, s'envia un esdeveniment d'èxit.

Mitjançant l'existència d'aquests quatre nivells per cada una de les funcionalitats implementades, podrem conèixer l'estat de la connexió amb l'API de cada una d'elles, amb un esforç relativament baix.

La figura [] mostra els diferents quatre nivells d'esdeveniments per la funcionalitat evolució d'esdeveniments. Com es pot veure en la imatge, Google Analytics captura quantes vegades s'ha donat un esdeveniment concret en el període de temps definit i quantes sessions l'han contingut en algun moment donat.

A continuació, també citem els diferents exemples mostrats en la figura [] i hi afegim algun comentari.

* **Formulari incorrecte:** *facts\_error\_formValidation.* Esdeveniment que s'envia quan la petició no s'ha arribat a processar perquè la configuració dels paràmetres no era correcta.
* **Petició llençada:***facts\_Deaths\_1937\_1947\_unitedstates.* Com es pot veure, quan es llança una petició contra l'API, també capturem el tipus d'esdeveniment cercat (defuncions), les dates per les quals s'ha cercat (1937-1947) i el país seleccionat (Estats Units).
* **Petició rebutjada:** *facts\_error\_gateway\_timeout.* Aquest és un exemple dels esdeveniments que poden ser retornats quan l'API no ha pogut processar la petició.
* **Petició retornada amb èxit:** *facts\_successful.* Esdeveniment que s'envia quan tot ha funcionat com s'esperava i l'API ha retornat resultats.

Els exemples d'esdeveniments anteriors, serveixen per demostrar el potencial que pot arribar a desencadenar una eina d'analítica web com Google Analytics.

A part dels esdeveniments relatius a les funcionalitats implementades, que interactuen amb l'API de FamilySearch, també s'han implementat esdeveniments per controlar els intents d'identificació contra FamilySearch satisfactoris, les peticions de desconnexió, posicions dels exemples i propostes de projecte amb les que s'ha interactuat, interaccions amb la barra de navegació, posició de la persona seleccionada en la llista de resultats de la funcionalitat de cerca, etcètera.

Tot i que el conjunt d'informació que s'ha exposat fins ara relativa a Google Analytics podria ser considerada simple, no creiem que sigui l'objectiu de la memòria entrar en molt més detall en les possibilitats d'una eina d'analítica web com aquesta, doncs realitzar una proposta profunda i exhaustiva de les possibilitats de monitoratge d'una web com la implementada, bé podria ser un projecte propi.

**Optimització d'imatges i minimització del codi**

Un dels elements més importants quan es desenvolupa una pàgina web és que aquesta carregui de forma ràpida. La diferència entre una pàgina lenta i una de ràpida, s'acaba traduint generalment en una pàgina web sense usuaris o amb usuaris.

No era un objectiu del projecte fer una pàgina web el més optimitzada possible, ja que els coneixements tècnics necessaris requereixen temps per ser adquirits. No obstant això, sí que es volia realitzar les optimitzacions més típiques, que també solen ser les que endarrereixen més la càrrega de les pàgines web.

Aquest apartat, cobra relativa importància, si tenim present que estem utilitzant un servei d'hostalatge gratuït i que per tant, la velocitat de resposta del servidor, no és de les millors del mercat.

**Optimització d'imatges**

La manca d'optimització en les imatges sol ser un dels factors que més afecta a l'hora de carregar una pàgina web. Els programes de disseny utilitzats per generar imatges de gran qualitat, solen emmagatzemar més informació que la perceptible per l'ull humà, en circumstàncies normals.

És per aquest motiu que optimitzar les imatges, els arxius més grans a descarregar de forma general en una web, esdevé un procés relativament comú.

Per optimitzar les imatges de la nostra aplicació web s'ha utilitzat l'aplicació Optimizilla. Aquesta eina, penjada al núvol, utilitza una combinació de tècniques d'optimització i compressió amb pèrdues, per tal de reduir al màxim el pes d'imatges JPG i PNG, sense reduir el nivell de qualitat perceptible per l'ull humà.

La utilització d'aquesta eina ha reduït, de forma aproximada, el 60-80% del pes de totes les imatges que s'utilitzen en l'aplicació web. Fet considerable, si tenim en compte que no s'ha reduït la qualitat perceptible d'aquestes.

**Hosting de l'aplicació**

Com ja s'ha indicat en el primer apartat d'aquesta secció de la memòria, l'aplicació es troba accessible a través del domini:

*https://pfc-family-search.herokuapp.com/*

Hem escollit la plataforma Heroku per desplegar l'aplicació, ja que oferia un ampli ventall d'eines i documentació, que resultaven ideals per un desenvolupador novici de la plataforma Node.js.

No obstant això, no van ser només aquestes facilitats i eines les que ens van portar a desplegar l'aplicació a Heroku, sinó també el fet que el seu pla de hosting gratuït s'ajustava en gran mesura al que el projecte requeria i en cas d'acabar necessitant més, sempre existia la possibilitat d'augmentar la capacitat de processat necessària.

Les característiques que més ens van atreure de la plataforma Heroku es presenten d'una en una en els següents apartats.

**Fàcil configuració**

Per una aplicació simple com la desenvolupada per aquest projecte, no fa falta canviar res en el codi de la nostra aplicació per tal de desplegar-la a Heroku. Només hem d'incloure un nou fitxer, a la carpeta arrel del projecte i amb el nom *Procfile*, que indica el tipus d'aplicació i la comanda que ha ser utilitzada per tal d'iniciar-la.

En el nostre cas, el fitxer *Procfile* conte la següent línea de codi: `*web: node app.js'*

**Fàcil desplegament al núvol**

La plataforma Heroku es troba molt ben integrada amb Github, l'eina que hem utilitzat per mantenir sincronitzats els desenvolupaments en les diferents estacions de treball. Gràcies a aquesta integració, Heroku disposa d'una comanda que ens permet afegir un remot, a la carpeta del nostre projecte.

*>heroku git:remote -a pfc-family-search*

Un cop tenim el remot de Heroku configurat, de la mateixa forma que podem enviar el codi de la nostra estació de treball al núvol, el podem enviar a Heroku per tal de desplegar la nostra aplicació web. Això és tan fàcil com llençar la següent instrucció.

*> git push heroku master*

**Entorn de proves local**

A part de l'entorn de proves local que podem configurar mitjançant la instal·lació de Node.js en el nostre sistema, Heroku també ens permet simular un entorn de producció Heroku, en l'àmbit local. S'aconsegueix mitjançant la següent instrucció.

*> heroku local web*

Tot i que no aporta gaires diferències respecte a desplegar l'aplicació en local mitjançant Node.js de la forma convencional, pot esdevenir útil sota certes circumstàncies.

**Decent versió gratuïta**

Durant tot el procés de desenvolupament, l'aplicació es trobava sota el paquet de funcionalitats gratuït ofert per Heroku.

Aquest ofereix les següents funcionalitats:

* Desplegament des de repositoris GIT.
* Actualitzacions automàtiques.
* Auto reparació d'aplicacions.
* Logs del sistema.
* Nombre de processos diferents suportats: 2
* 1000 hores mensuals de `dyno' actius. L'aplicació s'adorm després de 30 minuts d'inactivitat.
* Dominis personalitzables.
* 512MB de RAM

**Fàcil escalatge de l'aplicació**

En cas que es desitgi millorar la capacitat de concurrència de l'aplicació, per poder rebre i processar més peticions en paral·lel i realitzar més processos diferents al mateix temps, aquesta és fàcilment escalable mitjançant la inclusió de nous `dynos'.

Els `dynos' són els contenidors que executen les comandes dels usuaris. Per la nostra aplicació web, bàsicament es necessiten `dynos' que processin el tràfic HTTP i HTTPS. Gràcies al fet que el nostre servidor és bastant simple (hem posat la lògica d'interacció amb l'API de FamilySearch a la capa del controlador) és probable que no faci falta augmentar el nombre de `dynos' inicial.

De totes maneres, aquest és fàcilment escalable mitjançant una simple comanda, sempre i quan el nostre pla contractat amb Heroku no sigui el gratuït. Si volem augmentar, per exemple, a 2 el nombre de `dynos' disponibles, executaríem la següent comanda:

*> heroku ps:scale web=2*

**Heroku**https://devcenter.heroku.com/

**Optimizilla**http://optimizilla.com/

**Analytics**https://analytics.google.com/