**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:

* Hosting i accés a l’aplicació.
* 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**

**Introducció**

L'aplicació és accessible a través de la URL: *https://pfc-family-search.herokuapp.com/*

Per tal d'accedir a la zona d'exemples de l'aplicació cal identificar-se primer amb FamilySearch. Per fer-ho, cada professor necessitarà un compte de FamilySearch diferent. A l'apèndix [] s'inclouen diferents usuaris creats per cada un dels membres del tribunal, però cada un també pot crear el seu propi.

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

No obstant, no són només les facilitats pels nous desenvolupadors el que ens van portar a desplegar l'aplicació a Heroku, sinó també que el seu pla gratuït s'ajustava en gran mesura el que el projecte requeria i sempre existia la possibilitat d'augmentar la capacitat de processat necessària sota demanda.

Les característiques que més ens atreien de la plataforma Heroku són presentades en els següents apartats.

**Fàcil configuració**

Per una aplicació simple com la desenvolupada per aquest projecte, no fa falta canviar res en el codi d'una aplicació Node.js normal. Només hem d'incloure un fitxer a l'arrel del projecte amb el nom 'Procfile', que indica la comanda que s'ha d'utilitzar per iniciar l'aplicació. En el nostre cas, el fitxer 'Procfile' conte la línea de codi: '*web: node app.js'*

**Desplegament fàcil al núvol**

Heroku es troba molt ben integrat amb Github, l'eina que utilitzem 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 de repositori a les nostres carpetes de desenvolupament.

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

Un cop tenim el remot de Heroku configurat, per desplegar l'aplicació al núvol només hem d'utilitzar la comanda:

*> git push heroku master*

Aquesta comanda indica que volem penjar a Heroku el contingut actual del nostre repositori.

**Entorn de proves local**

Apart del entorn de proves local que podem configurar mitjançant l'instalació de Node.js en el nostre sistema, Heroku també ens permet simular la nostre aplicació en un entorn de producció Heroku a nivell local mitjançant la comanda:

*> heroku local web*

Tot i que no aporta gaires diferencies respecte a desplegar l'aplicació en un entorn local mitjançant Node.js de la forma regular, pot esdevenir una eina útil sota certes circumstàncies.

**Versió gratuïta decent**

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

Aquest ofereix les següents funcionalitats:

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

**Escalatge fàcil de l'aplicació**

En cas de que es desitgi millorar la capacitat de concurrència de l'aplicació per poder rebre més peticions 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 usuari. Per la nostre aplicació web bàsicament es necessiten 'dynos' que processin el tràfic HTTP i HTTPS. Gracies a que el nostre servidor és bastant simple (hem posat la lògica d'interacció de FamilySearch en la capa del controlador) és probable que no faci falta augmentar el nombre de 'dynos' inicial.

De totes formes, aquest és fàcilment escalable mitjançant una simple comanda sempre i quan el nostre pla contractat amb Heroku no sigui el bàsic. Si volem augmentar a 2 els 'dynos' disponibles, executaríem la següent comanda:

*> heroku ps:scale web=2*

**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 de 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 la que 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 al usuari observar la 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 pe un país al voltant d'un any en 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 de 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 la API de FamilySearch simultànies per la mateixa funcionalitat des de la mateixa fulla del navegador.

**Requisits no funcionals**

* L'aplicació web ha de funcionar i ser visualitzada de forma correcta en els principals navegadors web moderns d'escriptori.
* El codi de l'aplicació serà codi obert.
* Els formularis de l'aplicació web que puguin generar errors han de proporcionar informació a l'usuari en el moment que el camp és abandonat o si s'intenta enviar el formulari amb errors.
* 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 la 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 clar feedback a l'usuari quan la interacció amb la API de FamilySearch finalitza.
* L'aplicació ha de ser navegable de forma acceptable mitjançant dispositius mòbil. Les integracions amb la 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 facilitar-ne la visualització quan l'usuari navega entre les diferents pàgines.
* La llengua utilitzada en el web és 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 que 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.
* És 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**

L'aplicació web és relativament simple pel que fa referència a la navegació i seccions d'aquesta.

La figura [] mostra l'arbre de continguts accessibles. Cal indicar que aquesta figura no representa les úniques rutes de navegació possibles entre les diferents seccions, sinó un breu mapa del contingut total disponible a través del website.

[]

A continuació explicarem en més detall en que consisteix cada una de les pàgines de la nostre aplicació web, però primer volem presentar l'estructura general que presenten cada una de les pàgines del web.

La figura [] mostra l'esquema bàsic que les pàgines segueixen. Aquest pot ser descrit o desglossat en les següents seccions:

1. **Barra de navegació:** Permet desplaçar-se per les diferent 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.

Instem als usuaris de l'aplicació web a redimensionar la finestra web per tal d'observar com el contingut s'adapta al dispositiu que la mostra.

**Home o pàgina principal**

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

El bloc de contingut principal de la 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 d'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 l'origen, el context i les motivacions que ens van portar a realitzar aquest projecte. Consisteix en un breu resum d'alguns dels apartats de 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, adaptada a la grandària del dispositiu que la conté.

**Propostes de projecte**

L'apartat de propostes de projecte recull les diferents propostes, que s'han generat per servir com a futurs projectes finals de carrera per a estudiants d'informàtica, en la mateixa secció.

El bloc de contingut principal per aquesta pàgina consisteix en dos blocs composats 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 els detalls específics 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 de propostes.

Les diferents propostes són totes generades des del mateix document HTML. És el servidor l'encarregat d'enviar un conjunt d'informació diferent segons la proposta que ha estat seleccionada.

El bloc del contingut principal per aquesta pàgina està conformat per una breu descripció de en que consisteix el projecte, una llista d'objectius que serveixi per donar un tret de sortida al projecte, però pensat perquè els estudiants iterin sobre ells, un conjunt de consideracions que creiem que cal tenir en compte i finalment una valoració personal sobre la dificultat del projecte.

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 les seccions 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 la 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 no es troba identificat amb FamilySearch. La pàgina permet dues accions simples, tornar enredera (o a la home si s'ha accedit a la pàgina mitançant la URL directament) o identificar-se amb FamilySearch.

El procés d'identificació s'inicia mitjançant el llançament d'un pop-up a la pàgina de FamilySearch en la que es demana nom d'usuari i contrasenya. Un cop aquest és verificat, el servidor redirigeix a l'usuari a la pàgina que havia demanat accedir.

La pàgina d'identificació tampoc pateix cap reestructuració de contingut quan es veu amb dispositius més petits. Simplement, s'adapta a la pantalla que la mostra.

**Exemples Implementats**

La pàgina d'exemples implementats permet a l'usuari descobrir les diferents eines que s'han implementat i accedir a cada una d'elles.

El bloc de contingut principal, en un estil molt similar a la pàgina de propostes de projecte, 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. Cal indicar que aquestes dues pàgines tenen un comportament tècnic idèntic i l'únic que les diferencia és el concepte semàntic que presenten i evidentment, en conseqüència, el contingut.

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

Aquestes pàgines segueixen el mateix patró que la gran majoria de pàgines del web explicat en la figura [].

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 diferencies 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.

El conjunt de fitxers programats representa un total de \_\_\_\_\_\_\_ línees de codi de les que un XX% són codi HTML, un XX% codi Javascript i un X% codi CSS.

|  |  |
| --- | --- |
| **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 mante la traçabilitat sobre l'estat actual de l'usuari i gestiona els ports de l'aplicació. |
| package.json | Conté la descripció de l'aplicació conjuntament amb les dependencies 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 i el fitxer a executar per crear 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 l'exemple evolució d'esdeveniments. |
| views/index.html | Fitxer HTML utilitzat per pintar la pàgina principal 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. |
| views/proposalsTemplate.html | Fitxer HTML utilitzat per pintar els detalls d'una proposta en concret. El contingut es variable segons la proposta seleccionada per l'usuari. |
| views/search.html | Fitxer HTML utilitzat per pintar l'exemple de funcionalitat de cerca. |
| views/surnames.html | Fitxer HTML utilitzat per pintar l'exemple d'expansió geogràfica de cognoms. |
| views/globals/footer.html | Fitxer HTML encarregat de pintar el footer de la pàgina en 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'adjuntar 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 que 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 dels camps del pare per l'exemple de cerca. |
| views/globals/searchMainPerson.html | Fitxer HTML encarregat de pintar el formulari de cerca dels camps del usuari principal per l'exemple de cerca. |
| views/globals/searchMother.html | Fitxer HTML encarregat de pintar el formulari de cerca dels camps de la mare per l'exemple de cerca. |
| views/globals/searchSpouse.html | Fitxer HTML encarregat de pintar el formulari de cerca dels camps de la parella per 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 la 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 la 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 la 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 pre càrrega dels document que s'enviïn contra el complement. En la nostra aplicació l'utilitzem per pre carregar les imatges de les capçaleres de secció abans de 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 de l'usuari de la pàgina d'identificació i les connexions amb la 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 la API de FamilySearch d'aquesta funcionalitat. |

**Funcionament general de l'aplicació web**

**Introducció**

L'objectiu d'aquest apartat és explicar com els diferents components de la web interactuen per tal de crear l'aplicació web que es pot trobar desplegada a Heroku.

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

**Creació del servidor**

L'aplicació comença a funcionar quan aquesta és executada en els servidors al núvol de Heroku, la plataforma d'hostalatge. Per arrancar el servidor, aquest és configurat mitjançant els fitxers 'Procfile' i 'package.json', que s'encarreguen d'assegurar que tots els complements Javascript necessaris s'instal·lin i s'executi el fitxer 'app.js' que configurarà el servidor.

Arribats aquest punt, el servidor està 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 nostre 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 per tal que el navegador pugui mostrar la pàgina i carregar totes les funcionalitats o interaccions possibles d'aquesta al controlador.

**Interacció amb la API de FamilySearch**

El moment en que l'usuari vol demanar dades provinents de l'API de FamilySearch, aquest es veu forçat a interactuar amb un element HTML del navegador. 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 activat, captura l'esdeveniment i avalua la petició. En cas de que no hi hagi cap problema, aquest fa una crida asíncrona al SDK de FamilySearch i n'espera la resposta.

De forma transparent a l'aplicació, el SDK es comunica amb la API de FamilySearch i si hi ha cap problema en la comunicació i la petició es vàlida, aquesta retorna les dades demanades en format XML o JSON. Posteriorment, el SDK transforma la resposta de la API en un objecte Javascript amb funcions de conveniència que facilitaran l'accés a les dades 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. El controlador processa i transforma les dades contingudes a l'objecte i un cop finalitzades les operacions necessàries, modifica la vista del client introduint els canvis pertinents en aquesta.

**Interacció amb elements HTML estàndard controlats pel controlador**

Quan els usuaris interactuen amb elements bàsics del HTML, per exemple, quan interactuen amb les caixes que poden expandir-se o contraure's i contenen camps de formulari en les funcionalitats implementades del cercador o evolució geogràfica de cognoms, la resposta per part del controlador és bastant simple.

El moment en que el controlador detecta que s'ha interactuat amb algun dels elements als que escolta, captura l'esdeveniment, avalua com s'ha de procedir segons el context de l'acció, en aquest cas, plegar o desplegar contingut d'un formulari, i realitza de forma immediata els canvis al navegador.

**Conclusió**

Els cassos 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 que 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.

**Detalls específics de la implementació**

**Introducció**

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 implementades, doncs aquests seran explicats 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 és busca una correlació exacte de les línees de codi, en el codi font de l'aplicació, s'observarà que a les línees de codi de la memòria són incomplertes i manquen molts paràmetres de configuració.

**Estructura general d'una pàgina HTML: La potencialitat de Mustache**

A excepció de la pàgina d'identificació amb FamilySearch, la resta de pàgina de l'aplicació web segueixen la mateixa estructura com bé s'ha explicat en el primer apartat d'aquesta secció.

Per evitar la duplicació de codi, els components comuns s'han creat 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 de les nostres pàgines segueix l'estructura següent:

*<!-- 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 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 components 'navbar' i 'pageTitle'.

També cal destacar que abans del 'footer' deixem un espai per incloure fitxers Javascript que són utilitzats únicament per la pàgina carregada per evitar carregar recursos no necessaris en la resta de pàgines del domini web.

**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 s'adapti de forma adequada a qualsevol pantalla i l'etiqueta necessària per indicar al navegador el conjunt de caràcters que ha de mostrar sense condificar.

*<meta name="viewport" content="width=device-width, initial-scale=1">  
<meta charset="utf-8">*

**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 l'estructura 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.

*<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.

*<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">  
 <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: La potencialitat de 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 com a '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 continguts 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.

Destacar que també esdevé interessant fixar-se en com l'aplicació diferència entre si s'ha de mostrar el subtítol per escriptori o tablet, ja que la lògica és similar per la resta de l'aplicació.

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' i 'visible-xs' en els subtítols del a 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 example-search-bkg **{{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 text-center sm-buffer hidden-sm hidden-xs">  
 <h2 class="****{{highlight}}*** *text-italic">****{{subtitleDesktop}}****</h2>  
 </div>  
 <!-- Subtitles: Tablet -->   
 <div class="col-md-12 text-center sm-buffer visible-sm">  
 <h2 class="****{{highlight}}*** *text-italic">****{{subtitleTablet}}****</h2>  
 </div>  
 </div>*  ***{{#button}}*** *...* ***{{/button}}****</div>*

Al final del bloc de codi mostrat, s'inicia una capçalera similar per dispositius de pantalla extra reduïda. El motiu pel que diferenciem les capçaleres és per mostrar una capçalera amb menys alçada i aprofitar millor l'espai disponible en dispositius mòbils.

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

Aquest fitxer s'encarrega de carregar tots els scripts necessaris a les pàgines del nostre domini web per tal de que les interaccions funcionin correctament.

El fitxer no conté misteri excepte que l'ordre de declaració és important per assegurar que no es poden executar interaccions amb el HTML sense que el complement jQuery hagi estat carregat primer.

Entre d'altres, aquest fitxer càrrega el complement jQuery, el SDK de FamilySearch, els fitxers Javascripts encarregats de manipular les galetes, la declaració del 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 del nostre website.

Les línees de codi més interessants són les que s'encarreguen de configurar el compte de Google Analytics i monitorar la pàgina que està sent vista i tota la informació relativa a la sessió que Analytics guarda de forma automàtica.

*ga('create', 'UA-80847078-1', { 'cookieDomain' : 'none' });  
 ga('send', 'pageview');*

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

Aprofitarem aquest fitxer per mostrar un exemple del grid de Bootstrap explicat en la secció sobre l'estudi tècnic del projecte.

El contingut principal d'aquest fitxer 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 de forma horitzontal, 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 de bootstrap està format per contenidors, o blocs d'espai, en els que es pot crear el nombre de fileres desitjat. Cada una d'aquestes fileres és dividida a la vegada 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 també funciona per dispositius mòbils, apilant els blocs de quatre columnes un sobre els altres?

Per comprendre-ho hem d'explicar primer que bootstrap divideix les aplicacions en quatre grandàries segons l'amplada en píxels 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 la taula anterior són el codi que s'ha d'utilitzar en les classes columna (col) del HTML per tal d'indicar sobre quin tipus de dispositiu estem marcant el nombre de columnes.

D'aquesta forma, la classe utilitzada en el bloc de codi anterior (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' relativament.

En altres paraules, cada bloc d'escriptori de quatre columnes passarà a ocupar-ne dotze en els dispositius petits i extra reduïts, forçant així que cada bloc ocupi tot l'espai disponible en una fila, i per tant, desplaçarà els altres blocs i provocarà l'apilament.

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

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

Degut a que aquesta pàgina de la nostre web està formada únicament per text, volem intentar augmentar una mica més la llegibilitat d'aquesta. Existeixen diversos estudis que demostren que l'esser humà llegeix de forma més còmoda línees curtes que contenen al voltant de 45 a 75 caràcters.

Per aquest motiu, en aquesta pàgina en que tot el contingut principal és text, hem volgut ajustar una mica l'espai ocupat per aquest i en comptes d'utilitzar les dotze columnes del grid de Bootstrap, hem decidit que el contingut n'ocupi només vuit.

Per tal que el contingut no se'ns quedi a l'esquerra, deixant quatre columnes en blanc a la dreta, utilitzem una classe especial de bootstrap que permet deixar columnes en blanc al començament de les dotze columnes. Aquesta classe segueix la forma: 'col-md-offset-\d'.

Així doncs, si volem centrar un contingut de 8 columnes en un grid de 12 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, tenim 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. Aquestes es caracteritzen per utilitzar imatges que s'adapten a la grandària del contenidor i la possibilitat d'incloure un títol i descripció. En el bloc de codi següent és mostra l'esquelet principal d'una d'aquestes caixes.

***<div class="thumbnail">*** *<****img*** *src="/images/thumbnails/search-min.png">  
 <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 de l'aplicació.

*"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 des del frontal 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 crean 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

És important de cara a que l'aplicació sàpiga quina mena de fitxer ha de pintar per poder interpretar el codi d'aquests de forma correcta.

En el nostre cas calia indicar-li a l'aplicació que els fitxers a renderitzar 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 fos capaç de resoldre les diferents URLs 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 components. 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 de les galetes.

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

Per altre banda, el complement bodyParser el preparem per rebre paràmetres de la URL i decodifcar paràmetres JSON.

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

Funcions de redirecció

Aquesta secció del arxiu s'encarrega de decidir i configurar els fitxers HTML que s'han de retornar segons la petició del client.

La resposta per totes les peticions GET dels client són solucionades d'una forma similar. Primer es capturen, després es recullen tots els paràmetres necessaris per emplenar amb Mustache i finalment es pinta el HTML demanat.

***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 mostra com es gestiona la petició de la pàgina inicial de l'aplicació web.

Un cop es captura la petició GET per la pàgina '/', és demana a l'objecte Javascript 'pageTitlesIns' carregat al inici del servidor, els paràmetres a utilitzar per pintar en el fitxer 'pageTitle.html'. Un cop rebuts aquests paràmetres, es pinta la pàgina 'index.html' i s'envien els paràmetres que es veuen llistats.

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

*app.get('/proposals/****:project****', function(req, res) {   
 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 la API de FamilySearch.

Quan és rep una d'questes peticions s'actualitza la galeta de sessió amb el token retornat per la API de FamilySearch o se n'esborra el contingut, i s'envia un paràmetre 'redirect' que indica a la capa del controlador a quina pàgina s'ha de redireccionar un cop el servidor finalitza el processat de la petició post.

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 te els drets necessaris 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 s'insereix com a 'middleware' 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 a quin port s'ha d'escoltar. La línea de codi utilitzada, permet alternar entre un valor per defecte o el especificat en els paràmetres en el moment d'executar 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'especte més interessant 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 cinc paràmetres: 'categoria', 'acció', 'etiqueta' i 'valor'. Els paràmetres 'etiqueta' i 'valor' són opcionals. El sistema plantejat ofereix la possibilitat de que el paràmetre 'category' s'ompli de forma automàtica amb el path de la pàgina que dispara l'esdeveniment.

*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.

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; }*

**Certificant l'aplicació amb Familysearch**

**Introducció**

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

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

En el cas de la nostre aplicació, el conjunt de passos a realitzar per tal de certificar-la 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 la API.

**Certificat d'autentificació**

Un dels dos certificats que l'aplicació desenvolupada requereix és relacionat amb les operacions d'identificació dels usuaris amb la 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 de que algun d'aquests token es vulgui emmagatzemar al navegador en una galeta, aquesta ha de ser una galeta segura.
* El tràfic és xifrat mitjançant el protocol SSL des de l'usuari fins a la API de FamilySearch.
* L'autentificació del 'usuari és completada mitjançant la crida directa als protocols d'identificació OAuth 2 de FamilySearch. No es permet l'emmagatzemen per part de tercers dels noms, contrasenyes d'usuaris i identificadors de sessió.
  + En una aplicació web, l'autentificació es pot realitzar mitjançant la crida d'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 la certificació.

**Certificat de lectura**

Les principals interaccions que realitza l'aplicació web desenvolupada són operacions de lectura contra la API de FamilySearch. Les principals restriccions a tenir en compte de cara a obtenir el certificat de lectura s'exposen a continuació:

* Autentificació certificada
* 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 públiques 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, al 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 regles de seguretat, s'han utilitzat precaucions per tal d'evitar problemes com la injecció de codi a través dels formularis de l'aplicació.

**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 diferents, dispositius mòbils utilitzats i interaccions bàsiques dels usuaris amb les diferents pàgines.

Gran parta d'aquesta informació es capturada de forma automàtica per Google Analytics pel simple fet d'incloure el 'snippet' de codi en la nostra pàgina web i a canvi es disposa de moltes variables d'informació que es poden creuar per tal d'analitzar el rendiment de la pàgina i l'ús que li donen diferents menes d'usuaris.

De totes formes, no s'espera que la nostre aplicació web disposi de molt tràfic i la implementació de Google Analytics no ha vingut donada pel fet de poder analitzar el rendiment de l'aplicació sota diferents variables, sinó per poder controlar el funcionament de la integració amb la API de FamilySearch.

Durant el desenvolupament de l'aplicació, mentre interactuàvem amb l'entorn sandbox de la API, ens vam donar compte que moltes dies per separat i inclòs a vegades períodes de tres o quatre dies seguits, l'entorn no funcionava i les peticions llençades contra la API es cancel·laven perquè aquesta no aconseguia acceptar-les.

Per tal de poder comprendre el funcionament de les interaccions amb la API, s'han creat quatre nivells d'esdeveniments diferents per cada una de les crides:

* **Formulari incorrecte:** En el cas que un usuari intenti llençar una petició contra la API i aquesta no s'iniciï perquè el formulari de configuració contenia errors, es marca el 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 s'ha llençat contra el SDK de FamilySearch i els paràmetres d'aquesta petició.
* **Petició rebutjada:** En cas de 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 (Timeout, no existeix el recurs, masses peticions en un període de temps, etcètera).
* **Petició retornada amb èxit:** Quan el SDK processa la petició i retorna un resultat, s'envia un esdeveniment d'èxit.

Mitjançant l'existència d'aquestes quatre nivells d'esdeveniments per cada una de les funcionalitats implementades, podrem conèixer l'estat de la connexió amb la 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 de la mateixa era incorrecte.
* **Petició llençada:** *facts\_Deaths\_1937\_1947\_unitedstates.* Com es pot veure, quan es llança una petició contra la API per aquesta funcionalitat, també capturem el tipus d'esdeveniment cercat (defuncions), les dates per les que es busca (1937-1947) i el país seleccionat (Estats Units).
* **Petició rebutjada:** *facts\_error\_gateway\_timeout.* Aquest és un exemple d'esdeveniment quan la API no ha ni intentant resoldré la petició. On simplement, no ha contestat.
* **Petició retornada amb èxit:** *facts\_successful.* Esdeveniment que s'envia quan tot ha funcionat com s'esperava i la API ha retornat resultats.

Els exemples d'esdeveniments anteriors serveixen per desmostrar el potencial que pot arribar a desencadenar una eina d'analítica web com Google Analytics. Si ens fixem en l'esdeveniment relatiu a la petició llençada, observem que donat un volum suficientment de dades, podríem estudiar quin tipus d'esdeveniment és el més interessant pels usuaris i el país i període de temps més cercats.

Amb tota aquesta informació podríem arribar a modificar els camps marcats per defecte en el formulari o donar feedback als usuaris en cas d'observar que la majoria d'usuaris està realitzant cerques, que donada la naturalesa de les dades contingudes al arbre familiar de FamilySearch, són candidates a no tenir resultats. Per exemple, si el conjunt d'usuaris cerques de forma predominant per dades contemporànies a Espanya.

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 possibilitat de monitoratge d'una web com la que s'ha implementat, bé podria ser un projecte propi.

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

**Introducció**

Un dels elements més importants quan és desenvolupa una pàgina web és que aquesta carregui de forma ràpida. La diferència entre una pàgina lenta i una de ràpida és generalment traduïble a una pàgina web sense usuaris o una amb usuaris.

Tot i que no era objectiu del projecte fer una pàgina web el més optimitzada possible, doncs els coneixements tècnics necessaris costa temps adquirir-los, si que es volia realitzar les optimitzacions més típiques que són conegudes per endarrerir més la càrrega de les pàgines web.

Aquest apartat cobra relativa importància si recordem que estem utilitzat un servei d'hostalatge gratuït i que per tant, la velocitat inicial del servidor, ja 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 de la perceptible per l'ull humà en circumstàncies normals.

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

Per optimitzar les imatges de la nostre aplicació web s'ha utilitzat l'aplicació amb serveis gratuïts anomenada Optimizilla. Aquesta eina penjada al núvol utilitza una combinació de tècniques d'optimització i compressió amb pèrdues per reduir el pes d'imatges JPG i PNG el màxim possible 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 la nostre aplicació web. Fet considerable si tenim en compte que no s'ha reduït la qualitat perceptible de les imatges.

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

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