**Exemples d'interacció amb la API**

**Introducció**

En aquesta secció de la memòria seran presentats els diferents exemples implementats en la nostre aplicació web que interactuen amb la API de FamilySearch.

Per cada una de las funcionalitat es proporcionarà una descripció adequada de la seva funció, detalls de la implementació i patrons d'usabilitat i exemples d'utilització si s'escau.

**Identificació i desconnexió amb l'API de FamilySearch**

**Descripció de la funcionalitat**

Les funcionalitats d'identificació i desconnexió amb la API de FamilySearch s'utilitzen exactament pel que el seu nom indica, per permetre als usuaris de l'aplicació web identificar-se o desconnectar-se a FamilySearch des de la pàgina d'un tercer, en aquest cas, nosaltres.

Els usuaris no tenen permisos d'interacció amb l'API a menys que tinguin un compte propi de FamilySearch i s'identifiquin en el sistema. El motiu és que un cop identificats cada usuari rep un 'token' o 'fitxa' personal, que un cop ajuntat a les crides a l'API, fa que aquestes siguin acceptades pel sistema.

En l'aplicació web implementada, la possibilitat d'identificar-se amb FamilySearch apareix quan l'usuari intenta accedir a la secció d'exemples o a un exemple concret. La pàgina en qüestió és relativament simple. L'usuari té o bé l'opció de tornar enredera o identificar-se amb FamilySearch.

La figura [] mostra la pàgina d'identificació.

Un cop els usuaris es troben identificats, tenen la opció de desconnectar-se des de qualsevol pàgina del web mitjançant un botó situat a la part dreta de la barra de navegació.

**Detalls de la implementació**

El client de FamilySearch

Degut a que es va decidir interactuar amb l'API de FamilySearch des de la capa controlador de la pàgina web, cal crear una instància del client cada cop que l'usuari canvia de pàgina.

Això és degut a que cada cop que el navegador canvia de URL, també neteja el conjunt de variables globals declarades en els fitxers Javascript. No obstant, això no vol dir que l'usuari s'hagi d'identificar de nou cada cop que canvia de pàgina.

El fitxer 'client.js' és l'encarregat de crear les instàncies del client i gestionar la concordança d'estat entre la connexió a FamilySearch i la informació emmagatzemada en el nostre back end o servidor. Recordem que el servidor és el que decideix si l'usuari pot accedir i utilitzar els exemples i que per tant esdevé important mantenir aquesta concordança d'estats.

A pesar de que l'usuari només ha d'estar identificat a les pàgines d'exemples, al voler oferir la possibilitat de desconnexió des de qualsevol pàgina de l'aplicació, el fitxer 'client.js' forma part dels documents Javascripts inclosos en el fitxer 'javascripts.html' que és inclòs a totes les pàgines del web.

El client FamilySearch és declarat a cada pàgina mitjançant la funció 'new Familysearch()' proporcionada des del SDK.

*var client = new FamilySearch({  
 client\_id: 'a02j000000E5DXqAAN',  
 redirect\_uri: document.location.protocol + '//' + document.location.host + '/',  
 save\_access\_token: saveCookie,  
 access\_token: token,  
 auto\_expire: true,  
 auto\_signin: true,  
 expire\_callback: function(data) { ... },  
 environment: 'sandbox'  
});*

Els paràmetres inclosos en la creació de la instància del client marquen les següents característiques:

* **client\_id:** Número del client que identifica l'aplicació a la plataforma de desenvolupadors de FamilySearch.
* **redirect\_uri:** URL de redirecció que s'ha registrat a la plataforma de desenvolupadors per l'aplicació.
* **save\_access\_token:** Variable que permet emmagatzemar el 'token' retornat per l'API en una galeta quan un usuari s'identifica i permet crear instàncies del client a totes les pàgines del domini web sense la necessitat d'anar demanant a l'usuari que s'identifiqui. En el nostre cas es tracte d'un booleà que contindrà el valor 'False' quasi sempre, ja que utilitzem un mètode alternatiu per emmagatzemar el 'token' d'accés que compleix amb els requisits de certificació.
* **test\_token:** En cas de que l'usuari ja es trobi identificat amb FamilySearch li passem el 'token' emmagatzemant de forma automàtica. Això ens permet no haver de demanar a l'usuari que s'identifiqui en cada una de les pàgines.
* **auto\_expire:** Booleà que indica si volem que el sistema netegi el 'token' de forma automàtica en cas de que aquest quedi invalidat.
* **auto\_signin:** Booleà que indica si volem que es demani a l'usuari que s'identifiqui cada cop que intenta realitzar una operació sense trobar-se connectat a FamilySearch. En la nostre aplicació, li passem el paràmetre 'True', no obstant, no s'hauria de poder donar el cas en que un usuari pogués llençar una crida contra la API sense trobar-se identificat. L'utilitzem com a mesura de seguretat.
* **expire\_callback:** Funció a executar quan el 'token' expira. Conté quasi el mateix codi que la funció de desconnexió implementada, així que en veurem el codi més endavant.
* **environment:** Entorn de l'API de FamilySearch en el que ens volem connectar: 'production', 'sandbox', etcètera.

Fins aquí, tot és relativament simple, però a on s'emmagatzema el 'token' per tal de complir amb les regles de certificació i poder reutilitzar-lo? La resposta és en l'espai local del navegador. Els navegadors moderns permeten escriure en el que és coneix com l'espai local del navegador, el mateix lloc on els navegadors emmagatzemen les imatges i recursos d'una aplicació web per tal de reutilitzar-les en futures crides, evitant descarregar-les de nou i augmentant així la velocitat de navegació.

S'utilitza l'espai local per emmagatzemar el 'token' ja que és un espai inacsessible per tercer i es dificulta en gran mesura el robament d'identitat.

Al principi del fitxer 'client.js' és comprova si el navegador suporta l'escriptura a l'espai local, en cas afirmatiu es carrega el 'token' emmagtzemat si aquest existeix o es configura el paràmetre que permetrà guardar el 'token' en un galeta, a pesar de violar les normes de certificació.

Aquets dos paràmetres són els que s'utilitzen en la creació del client descrita fa uns moments.

*if (typeof(Storage) !== "undefined") {  
 token = localStorage.token ? localStorage.token : '';  
} else {  
 saveCookie = true;  
}*

La funció d'identificació amb FamilySearch

La funció d'identificació es realitza des de la pàgina 'login.html' de l'aplicació i el controlador que la gestiona està inclosa en el fitxer 'login.js'.

Quan l'usuari selecciona el botó 'Access FamilySearch' mostrat a la figura [], el controlador executa la funció 'getAccessToken()' proporcionada pel SDK de FamilySearch. Aquesta funció obre un pop-up que permet a l'usuari identificar-se i n'espera la resposta de forma asíncrona.

En cas que l'usuari s'identifiqui de forma correcta, s'emmagatzema el 'token' retornat per la funció del SDK a l'espai local d'emmagatzematge del navegador i és llença la funció 'serverLogIn(token)' que s'encarregarà de mantenir la concordança d'estat entre un token vàlid a FamilySearch i el nostre servidor.

*client.getAccessToken().then(function(testTokenValue) {  
 localStorage.token = testTokenValue;  
 serverLogIn(testTokenValue);  
});*

La funció serverLogin està inclosa en el fitxer 'client.js' que recordem està inclòs en totes les pàgines del web i el que fa és realitzar una crida AJAX al servidor, passant com a paràmetre el 'token' retornat per l'API. Un cop el servidor ha processat la petició, la funció definida en el paràmetre 'success', redirecciona l'aplicació a una pàgina indicada pel servidor.

*function serverLogIn(apiToken) {  
 $.ajax({  
 type: "POST",  
 url: "/token/login",  
 data: JSON.stringify({token: apiToken}),  
 dataType: 'json',  
 contentType: 'application/json; charset=UTF-8',  
 success: function(data) { ... }  
 });  
}*

El servidor, per la seva banda, crea una galeta de sessió segura i no modificable pel client amb el 'token' codificat. Mentre aquesta galeta existeixi, el servidor permetrà l'accés de l'usuari a les pàgines d'exemples implementats.

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

La funció de desconnexió amb FamilySearch

Aquesta funció s'invoca de forma automàtica quan el 'token' expira o quan l'usuari utilitza la funcionalitat de desconnexió situada a la barra de navegació.

La funció de desconnexió amb FamilySearch realitza el procés invers realitzat per les funcions 'getAccessToken()' i 'serverLogIn()' de l'apartat anterior. Aquesta, invalida el 'token' actual configurat en el client, elimina el valor contingut a l'espai local d'emmagatzematge i sincronitza l'estat amb el servidor mitjançant una crida AJAX com la de l'apartat anterior.

*function serverLogOut() {  
 client.invalidateAccessToken();  
 localStorage.removeItem('token');*

*$.ajax({  
 type: "POST",  
 url: "/token/logout",  
 ...  
 });  
}*

Per la seva banda, el servidor elimina la galeta de sessió simplement assigannt-li el valor 'null'.

*app.post('/token/logout', function(req, res) {  
 req.session = null;  
 res.end('{"redirect" : "/"}');  
});*

**Aspectes d'usabilitat considerats**

Les funcions de connexió i desconnexió realitzen molta feina transparent per l'usuari, no obstant, si que s'ha volgut incloure un petit aspecte d'usabiliat per informar a l'usuari de que s'està esperant la seva identificació amb FamilySearch quan aquest prem el botó d'identificació.

Quan el pop up d'identificació amb FamilySearch és mostra, el contingut de la pàgina 'login.html' s'esvaeix lentament i apareix el gif mostrat en la figura [] que indica que s'està esperant una interacció per part de l'usuari. Quan l'usuari acaba el procés d'identificació és redirigit de forma automàtica a la pàgina d'exemples.

[]

Un petit segon exemple d'usabilitat és que el botó de desconnectar-se només es mostra si l'usuari es troba identificat.

**Cerca de persones a l'arbre familiar de FamilySearch**

**Descripció de la funcionalitat**

La funció d'exemple de cerca permet buscar instàncies de persones per tot l'arbre familiar de FamilySearch.

La funcionalitat ha estat habilitada per permetre la utilització de tots els paràmetres de cerca disponibles per l'API i el SDK i que han estat descrits a la secció cinc de la memòria. No obstant, sembla que per alguna raó desconeguda, la cerca d'esdeveniments relacionats a familiars de la persona cercada no funciona, i de fet, FamilySearch no ofereix aquesta possibilitat des de la seva pàgina web. En conseqüència hem decidit desactivar aquests camps fins a comprendre que està passant exactament.

La cerca es realitza amb el paràmetre 'count', que indica el nombre de resultats a retornar, sense especificar. En aquesta situació, l'API retorna quinze persones per defecte i un cop retorna els resultats, el controlador pinta la informació d'aquestes quinze persones en una taula que recull per cada persona informació sobre el seu identificador, nom i dates de naixement i defunció.

La taula de resultats permet la navegació entre totes les persones que complien les condicions de cerca, carregant-les en blocs de quinze persones. Cal tenir en compte que cada cop que és vol canviar de pàgina el client ha de realitzar una crida a l'API de FamilySearch i que per tant, pot tardar alguns segons a recarregar.

Finalment, la taula de resultats permet la selecció d'una de les persones per tal de desplegar tota la informació bàsica de la que es disposa sobre ella, els seus relatius més propers i informació s'obre l'ascendència i descendència de la persona.

**Recomanacions d'utilització**

En aquesta funcionalitat no s'ha restringit els camps de dates a només d'introducció d'un any. Aquest camp accepte diversos formats vàlids. Per exemple: '02-02-1807', '2/2/1807', '1807/02/02', '2 Febrero 1807', '2 February 1807', '2 1807', 'February 1807', '1807, etcètera.

No s'ha volgut restringir la possibilitat d'introducció de dates per donar més flexibilitat a l'eina, però cal tenir en compte que és imprescindible especificar sempre l'any. Si intentem realitzar una cerca especificant només el dia, el més, o el dia i el més, la cerca no obtindrà resultats.

De forma similar, els camps relacionats amb localitzacions accepten una gran diversitat en quant a nivells d'especificació i idiomes suportats, però per tal de que la cerca produeixi resultats, cal sempre introduir com a mínim, el nivell de província o estat. De totes maneres, per l'exploració de la funcionalitat, es recomana cercar a nivell de país.

**Detalls de la implementació**

Abans de comentar en detall alguns aspectes de la implementació, volem indicar que el fitxer del controlador d'aquesta funcionalitat està composat per quasi mil línees de codi i que evidentment és impossible exposar totes les tasques que aquest realitza en aquesta secció.

És per aquest motiu, que només es destaquen les interaccions principals o més importants de cara al funcionament de la interacció amb la API.

El controlador que s'encarrega de gestionar totes les interaccions de l'usuari amb la funcionalitat es troba en el fitxer 'search.js'.

Iniciant la cerca de persones

Quan l'usuari prem el botó de cerca, la primera tasca del controlador és validar que els continguts introduïts en el formulari son correctes. En cas de que el formulari no compleixi les condicions de cerca es dispararà un error per informar a l'usuari i és cancel·larà la crida a la API.

Per realitzar aquesta validació el formulari escapa cada camp que pot ser introduït per l'usuari i comprova si s'ha de realitzar alguna verificació específica del contingut. Escapar el camp d'un formulari significa codificar certs caràcters especials com per exemple '&', '<', '>', '/', etcètera, per tal d'assegurar el seu correcte transport per les URL i evitar atacs mal intencionats que pretenguin trencar el HTML per realitzar injeccions de codi. El fitxer 'formValidation.js' s'encarrega de realitzar aquestes transformacions i validacions sota petició.

Mostrem a continuació el codi que escapa els caràcters no desitjats.

*var entityMap={"&":"&amp;", "<":"&lt;", ... "/": '&#x2F;'};*

*function escapeHtml(string) {  
 return String(string).replace(/[&<>"'\/]/g, function (s) {  
 return entityMap[s];  
 });  
}*

Si no es mostra un error, significa que el formulari és correcte i compleix amb les condicions de cerca. Arribats a aquest punt és prepara el objecte 'params' que emmagatzema tots els paràmetres de cerca descrits en la secció [] de la memòria, s'eliminen de la crida aquells que no han estat introduïts per l'usuari i és demana la cerca dels primers quinze resultats mitjançant la crida a la funció *printPersonsToTable(0).*

La funció printPersonsToTable

Aquesta funció configura els paràmetres de la cerca *inici* i *context* i llença la crida asíncrona cap al SDK.

Si no es produeix cap error l'API retornarà els primers quinze resultats i aquests es pintaran sobre una taula. En cas contrari, l'aplicació ensenyarà un error a l'usuari on s'exposarà el motiu de la fallida.

*client.getPersonSearch(params).then(function(searchResponse) {   
 // Get persons and display id, name, birth date, death date  
})  
.catch(function(e) {  
 // Display error  
});*

L'estructura de llençada d'una petició al SDK i captura del error és l'estàndard que seguim en totes les crides realitzades cap aquest en les diferents funcions de l'aplicació web.

La llista de resultats que s'obté com a resultat d'una cerca és mostra a la figura [].

La gràcia d'aquesta funcionalitat és que la reutilitzem per quan l'usuari vol navegar pels diferents bocs de quinze resultats mitjançant les fletxes mostrades a la figura [] sense la necessitat de revalidar el formulari de cerca.

Quan el SDK retorna els resultats al controlador, aquest emmagatzema en variables globals els paràmetres resultats totals, índex del primer resultat mostrat i context de la cerca, que permet aconseguir els següents resultats de la cerca. Gracies a això, aquesta mateixa funció pot ser l'encarregada de configurar els paràmetres posició d'inici i context just abans de realitzar la cerca i per tant, reutilitzada tant per la cerca inicial com per la navegació dels resultats sense necessitat de configurar la resta de paràmetres emmagatzemats també en l'objecte global 'params'.

*function printPersonsToTable(pos) {  
 // Update start params  
 params.start = pos;  
 params.context = context;*

*// Search with the defined parameters  
 client.getPersonSearch(params).then(function(searchResponse) {  
 // Get parameters  
 count = searchResponse.getResultsCount();  
 start = searchResponse.getIndex();  
 context = searchResponse.getContext();*

*... // tractament dels resultats  
 });  
}*

Mostrant els detalls d'una persona

La selecció d'una persona en la taula de resultats originada per la cerca, provoca una segona crida al SDK de FamilySearch en la s'obté tota la informació disponible sobre la persona seleccionada i la dels seus familiars.

Com en totes les crides a l'API, en cas d'error, aquests es mostren en un secció específica, però en cas contrari, es pinta en diferents taules la informació bàsica de la persona, els noms diferents que s'han utilitzat per referir-se a ella, els esdeveniments relacionats amb la seva vida, informació bàsica sobre els pares, parelles i fills, informació de la seva ascendència i descendència, notes, fonts de dades i historial de canvis.

Es mostra en la figura [] el inici de les taules de resultats per una persona concreta.

La crida a l'API que gestiona questa informació és mostra en el bloc de codi següent.

*client.getPersonWithRelationships(personID).then(function(personResponse) {  
 var mainPerson = personResponse.getPrimaryPerson();  
 personDisplayProperties(mainPerson);  
 personDisplayNames(mainPerson.getNames());  
 personDisplayFacts(mainPerson.getFacts());  
 ...  
 client.getAncestry( ... );  
 client.getDescendancy( ... );  
 ...  
 mainPerson.getChanges( ... );  
});*

Com s'ha pogut observar en el bloc de codi anterior, per obtenir les diferents peces d'informació s'ha de navegar per diferents nivells de la resposta. Alguna informació es accessible directament des de l'objecte inicial, altre cal anar a buscar-la al objecte específic de la persona i la informació relativa a l'ascendència i descendència cal demanar-la de forma explícita al SDK.

Les dades retornades per les crides al SDK de l'ascendència i descendència són interessants ja que la navegació pels resultats es realitza mitjançant la nomenclatura 'Ahnentafel' i 'Aboville' respectivament.

La nomenclatura 'Anhentafel' otroga a la persona sobre la que es cerca l'ascendència el nombre 1. El seu pare rep el número 2 i la mare el 3. Les regles per calcular els nombres dels pares de qualsevol persona en l'ascendència són:

* **Pare:** nombre de la persona \* 2
* **Mare:** nombre de la persona \* 2 + 1

La nomenclatura 'Aboville' utilitza una estructura similar a la de les seccions i apartats d'aquesta memòria. Si s'otorga el nombre 1 a la persona sobre la que es cerca la descendència, s'utilitza:

* **Fills:** 1.1 / 1.2 / 1.3 / etcètera.
* **Els fills dels fill 1.1 són:** 1.1.1 / 1.1.2 / 1.1.3 / etcètera
* **Els fills del fill 1.2 del fill 1.1 són:** 1.1.2.1 / 1.1.2.2 / etcètera.

Generació de taules amb informació sobre un recurs

Com s'ha especificat en l'apartat anterior, la informació referent als diversos recursos disponibles dels detalls d'una persona s'imprimeix en diferents taules.

La idea de com representar aquestes taules neix de l'aplicació d'exemple del SDK de Javascript. Mitjançant l'adaptació del part del codi d'aquest per ajustar-lo a les necessitats del nostre projecte, s'aconsegueix mitjançant Bootstrap, javascript i jQuery la generació de taules dinàmiques.

Es requereix de taules dinàmiques ja que recursos com l'historial de canvis o el nombre de fills d'una parella, no és estàndard i per tant, s'ha d'adaptar a cada persona en concret.

La generació de les taules es realitza mitjançant la creació de matrius de dades on cada cel·la inclou un vector de dos valors. El primer indica el tipus del camp de la cel·la i el segon el contingut.

Mostrem en el següent bloc de codi les dues primeres fileres del a matriu de dades relativa a la informació bàsica d'una persona.

*var displayProperties = createPanelTable('Display information', [  
 [  
 ['th', 'ID'],  
 ['th', 'Gender'],  
 ['th', 'Lifespan'],  
 ['th', 'Living']  
 ],  
 [  
 ['td', person.getId()],  
 ['td', person.getDisplayGender()],  
 ['td', person.getDisplayLifeSpan()],  
 ['td', person.isLiving()]  
 ], ...  
];*

Un cop aquestes matrius de dades són omplertes, s'utilitza la funció *craetePanelTable(header, rows)* per transformar-la en un bloc de codi HTML que pinti la taula dins d'un panell. A continuació, el codi que tradueix el bloc principal de files a HTML.

*for(var i = 0; i < rows.length; i++){  
 var $row = $('<tr>').appendTo($table);  
 for(var j = 0; j < rows[i].length; j++){  
 $('<'+rows[i][j][0]+'>').text(rows[i][j][1]).appendTo($row);  
 }  
}*

**Aspectes d'usabilitat considerats**

Llargada del formulari

Aquesta pàgina pot arribar a presentar un formulari relativament llarg si considerem tots els camps en teoria disponibles de cara a la cerca. Per aquest motiu s'han dissenyat un parell de funcionalitats per tal de facilitar la comprensió i utilització de la funcionalitat.

La primera d'elles és que els formularis referents a la persona cercada i els seus relatius més propers poden ser contrets i expandits mitjançant un clic en la capçalera del formulari.

Per tal d'indicar que es pot interactuar amb aquestes capçaleres, s'ha afegit el signe '-' o '+' a l'esquerra del títol per indicar l'estat actual del formulari i a la dreta de la capçalera s'ha afegit una icona que també reflecteix l'estat i un text en cursiva que explica la funcionalitat i desapareix en dispositius de mida reduïda. Per sobre de tots aquests detalls, el color de la barra també canvia al passar el ratolí per sobre i la icona és transforma en una mà que fa clic.

La segona funcionalitat relacionada te l'objectiu de facilitar la iniciació de la cerca. En moltes circumstàncies els usuaris voldran només emplenar detalls bàsics de la persona a cercar i no dels relatius immediats. En aquestes situacions, sobretot en dispositius mòbils, el botó de cercar quedarà bastant lluny al final del formulari, per tant, per facilitar la cerca, un cop es superen 780 píxels de la posició del formulari de la persona principal i fins que s'arriba a la posició original del boto de cercar, una barra fixa apareix al final de la pàgina amb el botó de cercar.

La figura [] mosta els dos estats de les capçaleres dels formularis i la barra mencionada que apareix sobre impressionada al final de la pantalla quan és compleixen les condicions.

[]

El canvi d'estat entre les capçaleres es gestiona mitjançant la comprovació de l'estat amb jQuery i canviant els valors de les classes segons convingui.

L'efecte de la barra sobre impressionada s'aconsegueix mitjançant l'ús de jQuery per espiar la posició de l'usuari cada cop que es desplaça verticalment conjuntament amb canvis en el css per canviar i fixar la posició del contenidor i controlar la transició d'entrada que la fa aparèixer d'esquerra a dreta.

Validació del formulari

Com s'ha exposat en la descripció de la funcionalitat, aquest formulari no es troba gens restringit de cara a com les dades han de ser introduïdes. No obstant, si que hi ha una limitació i és que com a mínim s'ha d'omplir un camp del formulari.

En cas de no complir-la, un error es dispara al inici del formulari.

Indicació de cerca mentre s'espera la resposta del SDK

De la mateixa forma que en l'operació d'identificació, mentre s'espera la resposta del SDK, s'informa a l'usuari d'aquesta situació mitjançant una imatge animada que es mostrarà a la zona en la que apareixeran els resultats.

Aquest aspecte d'usabilitat s'aplica tant en la cerca inicial de persones com en la petició de cerca dels detalls d'una persona.

La figura [] mostra un exemple d'aquesta situació.

Missatges d'error del SDK

En cas de que alguna operació del SDK no acabi de la forma en que s'esperava, s'ensenya un error indicant que el problema ve donat per la banda de FamilySearch i l'error en concret retornat.

Per exemple, en aquesta funcionalitat, és normal que certes persones retornades no existeixin realment a l'arbre. L'estil utilitzant per representar l'error és el mateix que pels errors de validació de formulari.

La imatge [] ensenya un d'aquests missatges d'error.

Fletxes d'exploració pels resultats de la cerca

S'ha inclòs just per sobre de la taula de resultats de cerca una barra de navegació que permet a l'usuari explorar els diferents blocs de quinze persones que pot retornar el SDK.

Aquesta barra de navegació consisteix de dos fletxes que permeten navegar endavant i enredera pels blocs i un missatge de text que indica els resultats mostrats a la taula respecte el total de resultats disponibles.

S'ha implementat intel·ligència a les fletxes de navegació per assegurar que no s'accedeix a blocs no existents o invàlids.

La imatge [] ensenya aquesta funcionalitat.

Fletxes de retorn al inici dels detalls d'una persona

El conjunt de detalls obtingut sobre una persona pot acabar significant una alta quantitat d'informació disponible.

Per facilitar el retorn cap a les zones d'inici de la funcionalitat, al inici de cada secció dels detalls d'una persona s'ha inclòs una fletxa per retornar al capdamunt de la secció. La imatge [] mostrada en un apartat anterior mostra aquestes fletxes.

Navegació vertical animada

Aquesta funcionalitat canvia la posició de l'usuari a la pantalla donades certes accions com prémer el botó de cercar, seleccionar una persona de la llista de resultats, utilitzar una fletxa de les seccions de detalls d'una persona o l'aparició d'un error.

Per evitar confondre a l'usuari amb un canvi sobtat de la pantalla, el trasllat a la nova secció de la pàgina es realitza de forma animada en una transició d'un segon independentment de la distància a desplaçar.

Representació de l'estat

Una de les característiques principals amb les que s'ha intentat dotar l'aplicació és la capacitat de representar en tot moment l'estat actual de les funcionalitats independentment del que hagi passat anteriorment.

Les diferents micro transicions d'estat que succeeixen per aquesta funcionalitat i que no han quedat cobertes en les seccions anteriors es llisten a continuació:

* Quan es prem el botó de cercar, el text d'aquest canvia a 'Searching now...' i passa a un estat de desactivació que n'impedeix la utilització fins que l'estat actual és resolt. Quan la cerca finalitza o un és produeix un error l'estat del botó tornar a la seva normalitat.
* Els missatges d'error provinents del SDK o validacions del formulari desapareixen en quan es realitza una nova operació.
* Quan s'utilitzen les fletxes de navegació pels resultats de la cerca, la taula desapareix i tornar a aparèixer per indicar que han canviat els continguts i s'actualitza el valor del bloc de persones mostrat.
* Quan és selecciona una persona de la taula de resultats per mostrar-ne els detalls, la secció de detalls canvia el títol de 'Loading information...' a 'Nom\_persona details' quan el SDK retorna la informació demanada.
* Quan és selecciona una persona per mostrar-ne els detalls, quan una altre persona havia estat seleccionats amb anterioritat, els detalls de la persona prèvia son netejats.

**Interès d'ús**

El interès principal d'aquesta funcionalitat és explorar l'arbre familiar de FamilySearch i observar la diferent informació disponible relacionada a les persones.

**Evolució geogràfica d'un cognom**

**Descripció de la funcionalitat**

La funcionalitat evolució geogràfica d'un cognom permet explorar el nombre de persones nascudes enregistrades a les bases de dades de FamilySearch amb un cognom específic en un conjunt de països seleccionats en un any o interval de temps concret.

Aqueta funcionalitat explota la funció de conveniència 'getResultsCount()' inclosa en la resposta de la cerca de persones mitjançant el SDK de FamilySearch.

Com ja s'ha comentat en diversos llocs de la memòria, el resultat d'una cerca de persones retorna el nombre de persones indicades en el paràmetre 'count' o quinze per defecte. Com més persones es demanen, més tarda el SDK en processar la petició.

No obstant, independentment del nombre de persones sobre les que es demanin detalls, el SDK te accés al nombre de registres que compleixen les condicions de cerca i posa aquest número a disposició dels desenvolupadors mitjançant la funció de conveniència esmentada.

D'aquesta forma podem realitzar una cerca utilitzant el paràmetre 'count' configurat a cero i obtenir així el nombre total de persones que compleixen les condicions de cerca mitjançant la funció de conveniència sense sacrificar velocitat. Com que aquesta funcionalitat no pretén accedir als detalls de les persones en cap moment, no importa si aquestes no són retornades.

La cerca permet configurar el cognom, països i any o anys pels que s'ha de realitzar la cerca.

* **Cognom:** Qualsevol dada introduïda serà donada per vàlida excepte deixar el camp del formulari en blanc.
* **Països:** Per seleccionar un país només cal marcar el checkbox situat a la seva esquerra. Els països estan agrupats per continents i dins de cada continent es troben ordenats en ordre alfabètic.
* **Any de naixement:** Indica l'any pel que es vol llençar la cerca o el primer any del interval que vol ser investigat. L'any ha de ser introduït en un format de quatre dígits. Per exemple, 2016.
* **Rang:** El rang és un paràmetre opcional. En cas d'esta omplert indica quin és l'últim any del interval que vol ser considerat. També ha de ser introduït en format de quatre dígits i ve a representar que el període de temps a considerar són els anys compresos entre Any de naixement i Rang.
* **Interval:** El paràmetre interval només és necessari si s'ha especificat el camp Rang. Aquest paràmetre serveix per indicar cada quants anys volem que es realitzi una fotografia del nombre de persones nascudes amb un cognom determinat en el conjunt de països seleccionats. Per exemple, si el paràmetre any de naixement és 1920, el paràmetre rang és 1940 i el interval és 10, es realitzarà una foto de l'estat pels anys 1920, 1930 i 1940.

Per cada combinació d'any i país es realitzarà una crida al SDK demanant el nombre d'instàncies de persones nascudes amb el cognom indicat.

S'utilitza aquesta aproximació perquè l'alternativa consisteix en cercar el nombre de persones nascudes amb el cognom seleccionat en el rang de temps desitjat i navegar pels detalls de cada una de les persones de la resposta per tal de geolocalitzar o cercar el país de naixement. Aquesta aproximació és bastant inviable si tenim en compte el volum de dades de producció.

Com a últim detall tècnic, comentar que s'obté la fotografia cada cert període de temps en comptes d'agafar totes les instàncies de persones nascudes entre els diferents períodes, degut a discrepàncies en al forma de realitzar la cerca entre el SDK i la pàgina oficial de FamilySearch.

Tot i que la documentació del SDK indica que permet la utilització de rangs de dates en el paràmetre data de naixement, a la pràctica, no és així. La utilització d'aquesta opció en el SDK és tradueix en la obtenció de resultats incorrectes.

En cas d'utilitzar-la, segons la inclusió o no del caràcter '~' al final del rang de dates, el SDK o bé no retornava resultats o retornava resultats amb dates similars en comptes de dates dins del interval especificat. Per exemple, si es demanava el interval 1920-1940, la resposta també podia contenir resultats pels any 1820 o 1980. Utilitzant la funcionalitat de cerca de la web de FamilySearch com a comparació, això no pasava.

Després d'indagar una mica vaig descobrir que el motiu era que el SDK i la web oficial de FamilySearch utilitzaven paràmetres diferents per seleccionar resultats inclosos dins d'un interval de temps.

El SDK realitza una crida a l'API de FamilySearch mitjançant el paràmetre especificat: 'birthDate', mentre que la web FamilySearch utilitza el paràmetre 'birth\_year'. La diferència resideix en que el paràmetre 'birthDate' en realitat permet la inclusió del dia i mes de naixement, però no rangs de dates, mentre que el paràmetre 'birth\_year' només considera els anys i si permet demanar intervals de temps.

Així doncs, com que el paràmetre 'birth\_year' no és suportat pel SDK ni tampoc funciona si aquest és forçat de forma manual des del codi, es va decidir que la funcionalitat pintés la foto cada cert nombre d'anys en comptes de recopilar la informació de tots els anys entre els diferents intervals.

Retornant a la funcionalitat, un cop es realitza la cerca, es mostra a l'usuari informació sobre la configuració de la cerca, la duració estimada i el percentatge completat fins al moment.

A mesura que el SDK va retornant dades es pinten diferents gràfics que il·lustren el nombre d'instàncies del cognom trobades a cada país de formes diferents.

En concret existeixen tres menes de gràfics:

* **Mapa geogràfic:** Aquest mapa del globus terraqui desplegat emplena amb diferents tonalitats de color verd els diferents països pels que s'ha cercat. La foscor del color de cada país indica la quantitat d'instàncies del cognom que s'hi han trobat.
* **Gràfic de barres:** Ordenats els països de més instancies a menys instàncies, es mostra la mateixa informació que en el mapa geogràfic però en format gràfic de barres.
* **Gràfic de línees:** Només disponible quan la cerca conté dades de més d'un any. Mostra la quantitat d'instàncies trobades en cada país per cada any. Permet observar l'evolució temporal de cada país respecte els altres.

Un cop finalitza la cerca de dades, es pot navegar pels gràfics dels diferents anys mitjançant uns controls de navegació específics al cap damunt de la zona de resultats.

**Recomanacions d'utilització**

Volem recordar en aquesta secció que les crides al SDK de FamilySearch són asíncrones i que per tant, res ens impedeix realitzar deu, cent o mil crides simultànies a aquest. No obstant, si recordem la funcionalitat de 'throttling' mencionada en seccions anteriors de la memòria, ens impedeix abusar de la connexió i en cas de fer-ho, les nostres peticions passen a no ser processades.

Com que aquesta funcionalitat llença una cerca al SDK per cada país i any del interval, s'ha imposat en el codi una separació entre crides de dos segons. Això es tradueix en que el temps total d'execució aproximat d'una cerca en segons és: 2 \* número de països \* nombre d'anys.

Per aquest motiu, es recomana als usuaris intentar mantenir un nombre baix d'anys i països en les seves cerques de prova.

També instem a recordar que no tots els països disposen del mateix nombre de registres indexats o del mateix període de temps cobert.

**Detalls de la implementació**

Abans de comentar en detall alguns aspectes de la implementació, volem indicar que el fitxer del controlador d'aquesta funcionalitat està composat per cinc-centes línees de codi i que evidentment és impossible exposar en la documentació tot el codi interessant de la funcionalitat.

És per aquest motiu, que només es destaquen les interaccions principals o més importants de cara al funcionament de la interacció amb la API.

El controlador que s'encarrega de gestionar totes les interaccions de l'usuari amb la funcionalitat es troba en el fitxer 'geo-surnames.js'.

Validació del formulari

A diferència de la funcionalitat de cerca general, aquesta funcionalitat si que realitza una validació més exhaustiva del valors introduïts per l'usuari, ja que una configuració incorrecta d'aquests no obtindria resultats.

Existeixen dos tipus de validacions diferents. La coneguda com a validació en línea i la validació del formulari quan es prem el botó de cercar.

La validació en línea s'activa quan l'usuari surt d'un camp del formulari i s'aprofita aquest moment per mostrar el marc del camp en vermell si aquest conté algun error. L'objectiu d'aquesta validació es facilitar a l'usuari l'apreciació de que ha introduït un error i corregir-lo el més aviat possible, reduint així la frustració en el moment de realitzar la cerca.

Per activar aquesta validació s'utilitza la funció jQuery 'onFocusOut' en conjunció a les classes de Bootstrap 'has-success' o 'has-error' que al ser afegides al camp d'un formulari en pinten la bora de color verd o vermell de forma respectiva.

*$('.form-vali').focusout(function() {  
 if(inlineValidation($(this).attr('id'))) {  
 $(this).parent().removeClass('has-success');  
 $(this).parent().addClass('has-error');  
 }  
 else {  
 $(this).parent().removeClass('has-error');  
 $(this).parent().addClass('has-success');  
 }  
 });*

Com es pot observar en el bloc de codi anterior s'aprofita la mateixa funcionalitat de validació per la validació en línea i la validació global del formulari. Les regles de validació per cada un dels camps del formulari s'especifiquen a continuació:

* **Cognom:** Si el paràmetre te longitud cero, es mostra un error.
* **Països:** Si no s'ha seleccionat com a mínim un país, es mostra un error.
* **Any de naixement:** Si la longitud és diferent a quatre o no és un número, es mostra un error.
* **Rang:** Si el camp no està buit i la longitud és diferent a quatre o no és un número, es mostra un error.
* **Interval:** Si els paràmetres any de naixement i rang són diferents i el paràmetre rang no es buit i el interval no està especificat o no és un número, es mostra un error.

Quan el formulari és enviat cap al SDK, s'escapen els paràmetres llegits de la mateixa forma que s'ha explicat per la funcionalitat de cerca.

La imatge [] mostra a la vegada els errors en línea (marc dels camps del formulari en vermell o verd) i la caixa d'errors que informa a l'usuari quan prem el botó de cercar.

Regulació de les crides asíncrones

En cas de que es passi el procés de validació descrit en l'apartat anterior, significa que podem llençar la cerca contra l'API de FamilySearch.

Recordant el que hem descrit en les recomanacions d'utilització, llençarem una crida al SDK per cada combinació de país i any i aquestes peticions, al ser asíncrones, no esperaran a obtenir la resposta abans de llançar la segona. Per evitar que FamilySearch ens bloquegi hem introduït una espera de dos segons entre crida i crida mitjançant el paràmetre *apiDELAY* i la funcionalitat *setTimeout* de Javascript. També s'han encapsulat les crides en una funció que conté els paràmetres que indiquen quin any i país s'executa en aquella iteració.

*for (var i = 0; i < years.length; i++) {  
 ...  
 for(var k = 0; k < countries.length; k++) {  
 (function(k, i) {  
 setTimeout(function() {  
 client.getPersonSearch(params).then(function(searchResponse) { ... }  
 }, apiDELAY\*k+(i\*countries.length\*apiDELAY));  
 } (k, i));  
 }   
}*

El codi anterior permet que totes les crides al SDK es configurin al mateix moment i de forma instantània, però gràcies a la funcionalitat *setTimeout,* s'aniran executant una a una a intervals de *apiDELAY/1000* segons. El paràmetre *apiDELAY* està configurat per executar crides cada dos segons.

Com que les crides són asíncrones significa que els resultats es processaran en un moment incert i que les variables que emmagatzemin les dades retornades necessiten ser globals per tal de poder ser accedides des de qualsevol part del codi.

En concret s'han definit dues variables que emmagatzemen la mateixa informació però de forma diferent. Una pels gràfics del mapa geogràfic i gràfic de barres i un altre pel gràfic de línees. La raó és que l'estructura de dades que hem de passar a cada un dels gràfics és diferent i resulta més còmode crear dues estructures. Els paràmetres *'i'* i *'k'* passats a cada crida són indispensables de cara a guardar les dades al lloc de la matriu que els pertoca.

La variable *geomapCountries* és una matriu on cada fila conté les dades d'un any i cada casella de la fila està formada per un vector de dos elements. El codi del país i el nombre de persones que compleixen les condicions de cerca per aquell any i país.

La variable *linechartRows* és una matriu on cada columna representa un país i cada fila un any de l'interval. Cada casella de la matriu conté només un valor que indica el nombre de persones que compleixen els criteris de cerca en el país indicat per la columna i el any marcat per la fila.

A continuació es mostra el bloc de codi que llança la cerca al SDK i gestiona els resultats.

*client.getPersonSearch(params).then(function(searchResponse) {  
 // Get instances of people with name in country[k]  
 var total = searchResponse.getResultsCount();  
 ...  
 geomapCountries[i].push([countries[k].code, total]);  
 linechartRows[i].push(total);  
 ...  
}*

Impressió dels gràfics

Degut a la naturalesa de l'aplicació resulta fàcil, que a la mínima que es realitza una cerca interessant, aquesta tardi més d'un minut en ser completada.

Per tal de transmetre la sensació de que el sistema no està aturat i que l'usuari no decideixi tancar la finestra, hem introduït a la funcionalitat la capacitat d'anar pintant els gràfics de cada any a mesura que les dades de cada any són completades.

Per aconseguir aquest efecte disposem de variables globals que comptabilitzen el nombre de cerques que han estat retornades pel SDK i cada cop que les dades d'un any són completades, es pinta el mapa geogràfic i el gràfic de barres relatiu a l'any. Un cop es completen les dades de tots els anys també es pinta el gràfic de línees si es compleix la condició de que s'ha cercat per més d'un any.

Els gràfics es pinten mitjançant la crida a l'API de GoogleCharts. Per pintar qualsevol dels gràfics es segueix un procés similar:

1. És transformen les dades a un format que Google utilitzarà després per crear el gràfic.
2. És crea el tipus de gràfic desitjat mitjançant una segona crida a l'API de Google i s'indica a quin contenidor del HTML volem enganxar el resultat.
3. Es pinta el gràfic en el HTML.

Pel mapa geogràfic, gràcies a la forma en que s'han guardat les dades, el procés és relativament simple. Recordem que cada fila de la matriu *geomapCountries* conté els valors d'un any per cada país.

*function printGeomap(i) {  
 var geomapData = google.visualization.arrayToDataTable(geomapCountries[i]);  
 geomap = new google.visualization.GeoChart(document.getElementById('geomap'));  
 geomap.draw(geomapData, geomapOptions);  
}*

Per representar el gràfic de barres, el procés és molt similar al del mapa geogràfic però primer realitzem un mini tractament de les dades per tal d'ordenar els països de més instàncies a menys instàncies i obtenir així una representació més clara.

*function printBarchart(i) {  
 // Sort data  
 var barchartCountries = geomapCountries[i];  
 var first = barchartCountries.splice(0, 1);  
 barchartCountries.sort(compareCountries);  
 barchartCountries.unshift(first[0]);  
 $('#barchart').css('height', 100\*countries.length);  
 var barchartOptions = { ... };*

*// transform and plot*

*var barchartData = google.visualization.arrayToDataTable(barchartCountries);  
 barchart = new google.charts.Bar(document.getElementById('barchart'));  
 barchart.draw(barchartData, barchartOptions);  
}*

Finalment, el gràfic de línees també és bastant simple de pintar ja que no cal realitzar cap modificació sobre les dades.

*function printLinechart() {  
 ...  
 linechartData.addRows(linechartRows);  
 var title = ' ... ';  
 var linechartOptions = { ... };  
 linechart = new google.charts.Line(document.getElementById('linechart'));  
 linechart.draw(linechartData, linechartOptions);  
 ...  
}*

La figura [] mostra una visualització de cada un dels gràfics.

Creació de la secció de 'checkboxes' per cada continent en el HTML

Intentar crear la secció de 'checkboxes' de forma manual en el HTML hagués estat una bogeria. La creació d'aquestes seccions s'ha realitzat mitjançant la creació de HTML a través d'un procés iteratiu.

El servidor, mitjançant Mustache, utilitza la informació continguda en el fitxer *countryParameters.js* per tal de crear un procés iteratiu que genera totes les caselles necessàries. Per exemple, pel continent nord americà, s'utilitza el següent bloc de codi:

*{{#northAmerica}}  
<div class="col-md-3 col-sm-6 col-xs-12">  
 <label class="checkbox-inline">  
 <input type="checkbox" id="{{code}}" ... > {{name}}  
 </label>  
 <div>  
 {{/northAmerica}}*

D'aquesta forma, per cada país contingut en el objecte *northAmercia,* es representa el nom del país i s'emmagatzema el codi d'aquest en el camp ID per tal de ser utilitzat en la creació dels mapa geogràfic.

**Aspectes d'usabilitat considerats**

Aquesta funcionalitat aprofita de nou molts conceptes d'usabilitat explicats en la funcionalitat anterior i que per tant, no repetirem, però si anomenarem.

* La funcionalitat manté la interacció sobre certes capçaleres del formulari per tal d'expandir o contraure parts d'aquest. També utilitza una barra fixa al final de la finestra per facilitar l'accés al botó de cercar.
* Els missatges d'error que puguin aparèixer al prémer el botó de cercar o causats per la fallida del SDK es representen de la mateixa forma que els apartats de la funcionalitat anterior.
* La navegació vertical animada també forma part d'aquesta funcionalitat quan l'usuari realitza interaccions que canvien la seva posició en la pàgina.

A continuació es citen alguns aspectes d'usabilitat propis d'aquesta funcionalitat.

Navegació pels gràfics de diferents anys