**Exemples d'interacció amb la API**

**Introducció a les funcionalitats implementades**

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

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

Recordar també en aquest apartat, que publicar tot el codi implementat de les diferents funcionalitats és impossible, ja que només el controlador d'alguna d'elles, arriba a les quasi mil línies de codi.

És per aquest motiu, que en els següents apartats, es mostraran només certes parts del codi, sempre simplificades, que creiem que poden resultar d'especial interès per tal de comprendre les bàsiques sota les quals funcionen les diferents funcionalitats. Com sempre, tot el codi font pot ser trobat a la pàgina de GitHub.

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

**Descripció de la funcionalitat**

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

Els usuaris no disposen depermisos d'interacció amb l'API, a menys que s'identifiquin primer en el sistema. El motiu, és que un cop identificats, cada usuari rep un `token', que al ser adjuntat a les crides a l'API, fa que aquestes siguin acceptades.

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 disposa de dues opcions, tornar enredera o identificar-se amb FamilySearch.

La figura [] mostra la pàgina d'identificació.

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

**Detalls d'implementació**

El client de FamilySearch

Pel fet que es va decidir interactuar amb l'API de FamilySearch, des de la capa Controlador de la pàgina web, ens trobem en la situació d'haver de crear una instància del client cada cop que l'usuari canvia de pàgina.

Això és degut al fet quequan el navegador canvia d'URL, també neteja el conjunt de variables globals declarades en els fitxers Javascript. De totes maneres, 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 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'estat en tot moment.

A pesar que els usuaris només necessitenestar identificats a les pàgines d'exemples, el fet de voler oferir la possibilitat de desconnexió des de qualsevol punt de l'aplicació, el fitxer *client.js*s'adjunta a totes les pàgines del domini web.

Les instàncies del client FamilySearch són declarades a cada pàgina mitjançant la funció, *new Familysearch(),* proporcionada pel Javascript 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,compleixen les següents funcions:

* **client\_id:** Número del client, que identifica l'aplicació, a la plataforma de desenvolupadors de FamilySearch. Diferent per cada aplicació.
* **redirect\_uri:** URL de redirecció registrada a la plataforma de desenvolupadors de FamilySearch, per indicar el punt de retorn pel procés d'identificació.
* **save\_access\_token:** Variable que permet emmagatzemar el `token' retornat per l'API, en el procés d'identificació, en una galeta del navegador. Aquestapermet 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 tracta d'un booleà que contindrà el valor *false* quasi sempre, ja que utilitzem un mètode alternatiu per emmagatzemar el `token' d'accés, de forma que compleixi amb els requisits de certificació.
* **test\_token:** En cas 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 a cada una de les pàgines de l'aplicació web.
* **auto\_expire:** Booleà que indica si volem que el sistema netegi el `token' de forma automàtica en cas 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 contra l'APIsense trobar-se connectat a FamilySearch. En la nostra aplicació, li passem el paràmetre *true*, de totes maneres, no s'hauria de poder donar el cas en què un usuari pogués llençar una crida contra l'API sense trobar-se identificat. L'utilitzem, simplement, 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 veurem el seu compartament més endavant.
* **environment:** Entorn de l'API de FamilySearch en el que ens volem connectar: `production', `sandbox', `beta', etcètera.

Fins aquí, tot és relativament simple, però on s'emmagatzema el `token' per complir amb les regles de certificació i poder reutilitzar-lo? La resposta és a l'espai local del navegador. Els navegadors moderns permeten escriure en el que es 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-los en futures crides, evitant així, haver de descarregar-los de nou i augmentarla velocitat de càrrega.

S'utilitza l'espai local del navegador per emmagatzemar el `token', ja que és un espai inaccessible per tercers i dificulta, en gran mesura, el robament d'identitat.

Al principi del fitxer *client.js* es comprova si el navegador utilitzat per l'usuari suporta l'escriptura a l'espai local.En cas afirmatiu, es carrega el `token' emmagatzema't, si aquest existeix, en el paràmetre *token*. Altrament, es configura el paràmetre *save\_access\_token*amb el valor *true.*

Aquests dos paràmetres, són els mateixos que s'utilitzen en la creació del client, descrita en el bloc de codi anterior. A continuació, mostrem el bloc de codi que configura els paràmetres *save\_access\_token* i *token.*

*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 pressiona el botó `Access FamilySearch', mostrat a la figura [ref], 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 *testTokenValue* retornat pel SDK a l'espai local del navegador i és llença la funció *serverLogIn(testTokenValue),* que s'encarregarà de mantenir la concordança d'estat entre un token vàlid de 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 a totes les pàgines del web. Aquesta funció, realitza una crida AJAX al servidor passant com a paràmetre el valor *testTokenValue* retornat per l'API. Un cop el servidor ha processat la petició, la funció definida en el paràmetre *success*, redirigeix l'aplicació a la 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, en rebre la petició AJAX del client, crea una galeta de sessió segura i no modificable pel client amb el valor del paràmetre *testTokenValue* codificat. Mentre aquesta galeta existeixi, el servidor permetrà accedir a 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()*descrites a l'apartat anterior. Aquesta, invalida el `token' configurat en la instància del client, elimina el valor contingut a l'espai local del navegador 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 assignant-li el valor *null* i demana al controlador que redirigeixi a l'usuari cap a la pàgina principal de l'aplicació.

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

**Aspectes d'usabilitat considerats**

Les funcions de connexió i desconnexió realitzen feina sense necessitat d'interacció per part de l'usuari i per tant, no s'han pogut tractar gaires aspectes d'usabilitat. No obstant això, sí que s'ha volgut informar a l'usuari de què s'està esperant la seva identificació amb FamilySearch, quan aquest prem el botó d'identificació.

Quan el pop up d'identificació amb FamilySearch és mostrat, el contingut de la pàgina *login.html* s'esvaeix i apareix la imatge animada, mostrada a la figura [ref], que indica que s'està esperant una interacció per part de l'usuari. Quan l'usuari acaba el procés d'identificació,aquest és redirigit de forma automàtica a la pàgina d'exemples o a la pàgina principalen cas d'error.

[]

El segon aspecte d'usabilitat considerat, és que el botó de desconnectar-se només apareix, evidentment, si l'usuari es troba identificat.

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

**Descripció de la funcionalitat**

La funcionalitat 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 disponiblesper l'API i el SDK i que han estat descrits a la secció cinc de la memòria. De totes maneres, per alguna raó desconeguda, sembla que la cerca d'esdeveniments relacionats a familiars de la persona cercada,no acaba de funcionar i de fet, FamilySearch no ofereix aquesta possibilitat de cerca 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 retornats els resultats, el controlador pinta la informació d'aquestes quinze persones en una taula. Aquesta taula recull, per cada persona, informació sobre el seu identificador, nom, data de naixement i data de 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 per cada acció de paginació, el client ha de realitzar una crida a l'API de FamilySearch i que per tant, pot tardar alguns segons a recarregar-se amb nova informació.

Finalment, la taula de resultats permet la selecció d'una de les persones, per tal de desplegar tota la informació principal 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 la introducció d'un any. Aquest camp, accepta 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 dotar de més flexibilitat a l'eina, no obstant això, cal tenir en compte que és imprescindible especificar sempre l'any. Si intentem realitzar una cerca especificant el dia, el més, o el dia i el més, però sense la inclusió d'un any, la cerca no obtindrà resultats.

De forma similar, els camps relacionats amb localitzacions accepten una gran diversitat,pel que fa a nivells d'especificació i idiomes suportats, però per talque 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 escala de país.

**Detalls d'implementació**

Abans de comentar en detall alguns aspectes de la implementació, volem indicar que només el fitxer del controlador d'aquesta funcionalitat, està compost per quasi mil línies 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 l'API i configuració de la pàgina.

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 siguin correctes. En cas que el formulari no compleixi les condicions de cerca, es dispararàun conjunt d'errorsinformant l'usuari de la informació a corregir i no es llençarà cap crida contra l'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 pels URL i evitar atacs malintencionats, que pretenguin trencar el HTML i afectar el servidor, mitjançant injeccions de codi. El fitxer, *formValidation.js*, s'encarrega de realitzar aquestes transformacions i validacions.

Mostrem a continuació, el codi que escapa els caràcters no desitjats. Per escapar un paràmetre, només cal enviar-lo contra la funció *escapeHtml().*

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

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

Si no es mostra cap error de validació, significa que el formulari és correcte i compleix amb les condicions de cerca.

Arribats a aquest punt, es prepara l'objecte *params,* que emmagatzema tots els paràmetres de cerca descrits a la secció [ref] de la memòria, s'eliminen de la crida aquells que no han estat introduïts per l'usuari i es demana la cerca dels primers quinze resultats mitjançant la crida a la funció *printPersonsToTable(0).*

La funció printPersonsToTable

Aquesta funció,acaba de configurar els paràmetres necessaris per poder realitzar la cerca, la llença i en tracte la resposta.

En concret, configura els paràmetres *inici* i *context* i tot seguit,llença la crida asíncrona cap al SDK. Aquests paràmetres, ja han estat descrits en seccions anteriors de la memòria, però recordem que s'encarreguen de delimitar el primer resultat a retornar i a quina cerca es fa referència, en cas de trobar-se demanant més blocs de resultats a l'API de FamilySearch.

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 l'error a l'usuari tot indicant-ne el motiu de 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 codi anterior, encarregada de llençar una petició al SDK i capturar possibles errors, és l'estàndard que segueixen totes les crides realitzades a la nostra aplicació web, que interactuen amb el SDK de FamilySearch.

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

La gràcia de l'operació *printPersonsToTable(pos),* és que poden reutilitzar-la quan l'usuari vol navegar pels diferents blocs o pàgines de resultats, 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.

Gràcies a aquestes variables, la funció es pot encarregar de configurar els paràmetres: posició d'inici i context, just abans de realitzar la cerca contra el SDK i per tant, pot ser reutilitzada amb facilitat, conjuntament a l'objecte *params,* per demanar més blocs de resultats d'una cerca realitzada.

*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,a la taula de resultats originada per la cerca, provoca una segona crida al SDK de FamilySearch en la que s'obté tota la informació disponible sobre la persona seleccionada i els seus familiars.

Com en totes les crides a l'API, en cas d'error, aquest és mostrat en una secció específica. En cas d'èxit, es pinta, en diferents taules, la informació bàsica de la persona, els diferents noms pels quals és coneguda, els esdeveniments relacionats amb la seva vida, informació bàsica sobre els seus pares, parelles i fills, informació de la seva ascendència i descendència, notes de FamilySearch, fonts de dades que verifiquen la informació mostrada i l'historial de canvis realitzat sobre la persona.

Es mostra en la figura [ref], l'inici de la secció de taules resultant de demanar els detalls específics d'una persona.

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

*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 és accessible directament des de l'objecte inicial, mentre que alguns altres paràmetres cal anar a buscar-los al'objecte específic de la persona. Per altra banda, la informació relativa a l'ascendència i descendència, cal demanar-la de forma explícita al SDK.

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

La nomenclatura `Anhentafel',atorga a la persona sobre la qual se 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

Per altra banda, la nomenclatura `Aboville', utilitza una estructura similar a la de les seccions i apartats d'aquesta memòria. Si s'atorga el nombre 1, a la persona sobre la qualse cerca la descendència, s'utilitza per les diferents generacions:

* **Fills:** 1.1 / 1.2 / 1.3 / etcètera.
* **Els fills dels fill 1.1:** 1.1.1 / 1.1.2 / 1.1.3 / etcètera.
* **Els fills del fill 1.2 del fill 1.1:** 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 codi d'aquest, per ajustar-lo a les necessitats del nostre projecte, s'aconsegueix, mitjançant la combinació de Bootstrap, javascript i jQuery, la generació de taules dinàmiques.

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

La generació de les taules es realitza mitjançant la creació de matrius de dades, on cada cel·la és composta per un vector de dos valors. El primer, indica el tipus del camp de la cel·la, mentre que el segon, el contingut.

Mostrem en el següent bloc de codi, les dues primeres fileres dela 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 s'han creat les diferents matrius de dades, s'utilitza la funció *craetePanelTable(header, rows),*per transformar-les en un bloc de codi HTML que pinti la taules dins d'un panell de Bootstrap. Tot seguit, mostrem 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 funcionalitat pot arribar a presentar un formulari relativament llarg si considerem tots els camps, en teoria disponibles, de cara a la cerca de persones. Per aquest motiu, s'han dissenyat un parell de funcionalitats que pretenen facilitar la comprensió i utilització del formulari de cerca.

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

Per tal d'indicar que es pot interactuar amb aquestes capçaleres, s'ha afegit el signe `-' o `+', a l'esquerra del títol i una icona que també reflecteix l'estat actual, del bloc del formulari, a la dreta. A la vegada, un text en cursiva explica la funcionalitat en la mateixa barra i el color de la barra, canvia enpassar el ratolí per sobre i la icona es transforma en una mà d'interacció.

La segona funcionalitat,té l'objectiu de facilitar la iniciació de la cerca. En moltes circumstàncies, els usuaris només voldran emplenar detalls bàsics de la persona principal a cercar i no pas dels seus relatius. En aquestes situacions, sobretot en dispositius mòbils, el botó de cercar quedarà molt allunyat del'inici del formulari i per tant, pot resultar confús pels usuaris com iniciar la cerca.

Per evitar aquesta confusió, un cop se superen 780 píxels de la posicióinicial del formulari de la persona principal i fins que s'arriba a la posició original del botó de cerca, una barra fixa apareix al final de la pàgina amb el botó de cercar i se sobreposa al contingut de la pàgina web. D'aquesta forma, el botó de cercar és accessible en tot moment pels usuaris, sense necessitat d'arribar al final del formulari.

La figura [] mostra els dos estats possibles, de les capçaleres dels blocs del formulari i la barra mencionada, que apareix sobre impressionada al final de la pantalla amb el botó de cerca, quan es compleixen les condicions necessàries.

[]

L'efecte de la barra sobre impressionada amb el botó de cerca, s'aconsegueix mitjançant jQuery i el canvi de classes CSS, en els elements HTML que contenen la barra de cerca.El jQuery s'encarrega d'espiar la posició de l'usuari cada cop que es desplaça verticalment per la pàgina web i les diferents classes CSS, fixen o alliberen, la posició del contenidor de la barra de cerca.

A continuació mostrem la classe CSS que converteix la barra de cerca, en una barra de posició fixa, al final de la pàgina web. Les propietats que ho fan possible son la propietat, *position:fixed* i *bottom:0px.*

Validació del formulari

Com s'ha exposat en la descripció de la funcionalitat, els paràmetres del formulari no estan sotmesos a validacions del contingut. No obstant això, si que es realitza una comprovació abans de llençar la cerca i és que com a mínim, s'ha d’emplenar un camp del formulari, a part del sexe de la persona principal.

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

Animació 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 a la cerca de persones, s'informarà a l'usuari d'aquesta situació mitjançant una imatge animada que apareixerà a la zona dels resultats.

Aquest aspecte d'usabilitat s'aplica tant a la cerca inicial de persones, com en la petició d’obtenir els detalls d’una persona en concret.

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

Missatges d'error del SDK

En cas que alguna operació del SDK no acabi de la forma prevista, s'ensenya un error indicant que el problema ha recaigut a la banda de FamilySearch i s’indica el missatge d’error retornat des de l’organització.

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 possibles 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 persones retornats pel SDK. Recordem, que cada petició retorna un conjunt de quinze persones.

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

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

La imatge [] mostra l’aspecte visual d’aquesta funcionalitat.

Fletxes de retorn a l’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, a l’inici de cada secció dels detalls d'una persona, s'ha inclòs una fletxa per retornar al capdamunt de la secció. La imatge [ref] mostra aquestes fletxes, en el context dels detalls d’una persona.

Navegació vertical animada

Aquesta funcionalitat, s’encarrega d’animar els canvis de posició vertical automàtics, implementats en la funcionalitat. Exemples d’aquests canvis són prémer el botó de cerca, seleccionar una persona de la llista de resultats, utilitzar una fletxa de les seccions dels 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 de les quals 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 es produeix un error, l'estat del botó torna a la seva normalitat.
* Quan es realitza una nova cerca, els resultats de la cerca anterior (si aquesta existia), són esborrats per evitar causar confusió.
* Els missatges d'error provinents del SDK o validacions del formulari, desapareixen quan es realitza un nou intent de cerca.
* Quan s'utilitzen les fletxes de navegació pels resultats de la cerca, la taula desapareix i tornar a aparèixer per indicar que els continguts han estat refrescats i s'actualitza el valor del bloc de persones mostrat.
* Quan es selecciona una persona de la taula de resultats, per mostrar-ne els detalls, la secció de detalls pren el títol de `Loading information...', i el canvia a `Nom\\_persona details', quan el SDK retorna la informació demanada.
* En seleccionar una persona per mostrar-ne els detalls, quan una altra persona ja havia estat seleccionada amb anterioritat, els detalls de la primera persona seleccionada s’esborran per evitar causar confusió.

**Principal Interès d’ús**

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

**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, per un conjunt de països seleccionats, en un any o interval de temps definit.

Aquesta funcionalitat explota la funció de conveniència *getResultsCount(),* inclosa en la resposta de cerca de persones del SDK de FamilySearch.

Com ja s'ha comentat en diversos llocs de la memòria, els resultats d'una cerca de persones, retornen el nombre de persones indicades en el paràmetre *count* o quinze per defecte. Com més persones es demanin, més tardarà el SDK a processar la petició.

Tanmateix, independentment del nombre de persones demanades, el SDK disposa d’accés al nombre total de registres que compleixen les condicions de cerca i posa aquest valor a disposició dels desenvolupadors, mitjançant la funció de conveniència esmentada.

D'aquesta forma, podem realitzar una cerca utilitzant el paràmetre *count,* configurat a zero i obtenir així, el nombre total de persones que compleixen les condicions de cerca, mitjançant la funció de conveniència i sense sacrificar velocitat de resposta. 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 valor introduït serà donat per vàlid, sempre i quan no es deixi el camp en blanc.
* **Països:** Per seleccionar un país, només cal marcar el checkbox situat a l’esquerra del seu nom. Els països estan agrupats per continents i dins de cada continent, els països es troben ordenats en ordre alfabètic.
* **Any de naixement:** Indica l'any pel qual es vol llençar la cerca o el primer any de l’interval que vol ser considerat. 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’omplir-lo, indica quin és l'últim any de l’interval que es vol ser considerat. També ha de ser introduït en format de quatre dígits. Conjuntament, al paràmetre *any de naixement,* representa el període de temps comprés entre: *Any de naixement--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 *l’interval* és 10, es realitzarà una foto de l'estat, pels anys: 1920, 1930 i 1940.

Dit això, cal tenir en compte, que per cada combinació d'any i país, es realitzarà una crida diferent, al SDK, demanant el nombre d'instàncies de persones nascudes amb el cognom indicat.

S'utilitza aquesta aproximació, ja que l'alternativa consisteix a cercar el nombre de persones nascudes, amb el cognom seleccionat, pel 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 d’aquesta. Aquesta aproximació, resulta bastant inviable si tenim en compte el volum de dades contingut a producció.

Com a últim detall tècnic de la funcionalitat, 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, a causa d’unes discrepàncies en la 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, es tradueix en l’obtenció de resultats incorrectes.

En utilitzar la funcionalitat, interval de dates, el SDK no retorna resultats. Després d'indagar una mica, es va descobrir que el motiu pel qual el SDK i la web oficial de FamilySearch, tenien comportaments dispars, era per la utilització d’un paràmetre de cerca diferent de cara a la introducció de dates.

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 què el paràmetre *birthDate,* en realitat permet la inclusió del dia i mes de naixement en diferents formats, però no rangs de dates; mentre que el paràmetre *birth\\_year,* només considera els anys i per tant, si suporta la utilització d’intervals de temps.

Així doncs, sembla que el paràmetre *birth\\_year,* no és suportat pel SDK de forma natural, ni tampoc funciona si aquest és forçat de forma manual des del codi. Per tots aquests motius, es va decidir que la funcionalitat implementada pintes 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 l’usuari llença una 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, de formes diferents, el nombre d'instàncies del cognom introduït trobades a cada país.

En concret s’han implementat tres tipus de gràfics diferents:

* **Mapa geogràfic:** Aquest mapa del globus terraqüi desplegat, emplena, amb diferents tonalitats de color verd, el nombre d’instàncies del cognom cercat pels diferents països. La foscor del color de cada país, indica la quantitat d'instàncies trobades respecte als altres països.
* **Gràfic de barres:** Aquest gràfic, ordena els països cercats, de més a menys instàncies trobades pel cognom seleccionat. Es mostra la mateixa informació que en el mapa geogràfic, però en format de gràfic de barres ordenat.
* **Gràfic de línies:** Només disponible quan la cerca conté dades per més d'un any. Mostra la quantitat d'instàncies trobades, a cada país, per cada any de l’interval. Permet observar l'evolució temporal de cada país respecte als altres.

Un cop finalitza la cerca de dades, es pot navegar pels gràfics dels diferents anys, mitjançant uns controls de navegació específics, situats al capdamunt 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 al SDK i en conseqüència, a l’API de FamilySearch.

Tanmateix, si recordem la funcionalitat de `throttling', mencionada introduïda anteriorment en aquesta memòria, aquesta impedeix l’abús de la connexió i en cas de fer-ho, les nostres peticions quedarien bloquejades.

Com que la funcionalitat evolució geogràfica d’un cognom, llença una cerca al SDK per cada país i any de l’interval, s'ha imposat en el codi, de forma manual, una separació de dos segons entre les diferents crides. Això es tradueix, en què el temps total d'execució aproximat per una cerca, és de: 2 \* nombre de països \* nombre d'anys (segons).

Per aquest motiu, es recomana als usuaris mantenir un nombre baix d'anys i països, en les seves cerques de prova. Recordar també, en aquest punt, que no tots els països disposen del mateix nombre de registres i que per tant, alguns tipus de cerques, són candidates a no obtenir cap resultat.

**Detalls d’implementació**

Abans de comentar en detall alguns aspectes de la implementació, volem indicar que el fitxer del controlador d'aquesta funcionalitat està compost per cinc-centes línies de codi i que evidentment, resulta impossible exposar en aquesta memòria, tots els blocs de codi interessants.

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

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 de cerca

A diferència de la funcionalitat de cerca general, aquesta funcionalitat, sí que realitza una validació més exhaustiva dels valors introduïts per l'usuari a través del formulari, ja que una configuració incorrecta d'aquests, resultaria l’obtenció de zero resultats.

Han estat implementades dues menes de validacions diferents. La coneguda com a validació en línia i la validació del formulari quan es prem el botó de cerca.

La validació en línia 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, o en verd, si el valor introduït és correcte. L'objectiu d'aquesta validació és facilitar a l'usuari la comprensió dels errors i corregir-los al més aviat possible; reduint així, la frustració en el moment de realitzar la cerca.

Per activar la validació en línia, s'utilitza la funció jQuery *onFocusOut()* en conjunció a les classes de Bootstrap *has-success* o *has-error,* que en 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, per tal de comprovar si un camp és vàlid o no, es crida la funció *inlineValidation(param),* amb l’identificador del camp a comprovar. Aquesta funció es reutilitza tant per la validació en línia, com per la validació en el moment de cerca.

Les regles de validació per cada un dels camps del formulari s'especifiquen a continuació:

* **Cognom:** Si el paràmetre té longitud zero, 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 de quatre o no és un número, es mostra un error.
* **Rang:** Si el camp no està buit i la longitud és diferent de quatre o no és un número, es mostra un error.
* **Interval:** Si els paràmetres any de naixement i rang són diferents, el paràmetre rang no és buit i l’interval no està especificat o no és un nombre, es mostra un error.

La imatge [], mostra a la vegada, els errors en línia (marc dels camps del formulari en vermell o verd) i la caixa d'errors que informa l'usuari dels errors produïts, de forma més detallada, quan l’usuari prem el botó de cerca amb paràmetres incorrectes.

Si totes les validacions són superades, 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 en l’arbre familiar.

Regulació de les crides asíncrones

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

Recordant el fet, que ja hem introduït en les recomanacions d’utilització d’aquesta funcionalitat, llençarem una crida al SDK per cada combinació de país i any i aquestes peticions, pel fet de ser asíncrones, no esperaran a obtenir una resposta abans de llançar la següent petició cap al SDK.

Per evitar que FamilySearch ens bloquegi per abús del servei, 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 la representació dels paràmetres *any (i)* i *país (k),* executats 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 siguin configurades a la vegada i de forma quasi instantània, però que gràcies a la funcionalitat *setTimeout(),* s'aniran executant una per una a intervals de dos segons.

El fet que les crides siguin asíncrones, significa que els resultats retornats pel SDK es processaran en un moment incert i en conseqüència, les variables que emmagatzemen les dades retornades, necessiten ser globals per tal de poder ser accessibles des de qualsevol part del codi.

En concret, s'han definit dues variables globals que emmagatzemen la mateixa informació, però de forma diferent. Una pels gràfics del mapa geogràfic i gràfic de barres i una altra pel gràfic de línies.

El motiu principal pel qual s’utilitzen dues estructures diferents és que l'estructura de dades a passar, a l’API de Google per cada un dels gràfics, és diferent i resulta més còmode disposar de dues estructures. Els paràmetres *i* i *k,* passats a cada crida, s’encarreguen de guardar les dades de cada iteració, al lloc que els hi correspon, de les variables globals.

El primer objecte global, la variable *geomapCountries* és una matriu de dades 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 país.

Per altra banda, la variable global *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 pel país denotat en la columna i l’any denotat per la fila.

A continuació, es mostra el bloc de codi que llança la cerca al SDK i emmagatzema els resultats a les variables globals.

*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

A causa de la funcionalitat, resulta fàcil, que a la mínima que es realitza una cerca interessant, aquesta tardi més d'un minut en completar-se.

Per tal de transmetre la sensació de moviment i que el sistema està treballant, hem introduït a la funcionalitat la capacitat d'anar pintant els gràfics de cada any, a mesura que les dades van quedant disponibles. És a dir, que no cal esperar fins al final de la cerca per poder començar a veure resultats.

Per aconseguir aquest efecte, disposem de variables globals que comptabilitzen el nombre de cerques que ja han retornat del 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, es pinta el gràfic de línies.

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

1. Transformació de les dades a un format que Google utilitzarà després per crear el gràfic.
2. Creació del tipus de gràfic desitjat, mitjançant una segona crida a l'API de Google i indicació del contenidor HTML en el qual volem inserir el gràfic.
3. Renderització del gràfic en el HTML.

Pel mapa geogràfic, gràcies a la forma en què s’han anat emmagatzemant les dades, el procés és relativament simple. Recordem que cada fila de la matriu *geomapCountries,* representa els valors d’un any i cada columna, un país diferent.

*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 petit tractament de les dades per tal d'ordenar els països de més a menys instàncies del cognom trobades 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 línies també resulta bastant simple de pintar, ja que gràcies a la forma en què s’han emmagatzemat les dades, no fa falta realitzar cap tractament especial abans de pintar-les.­

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

Las figures [], [] i [], mostren una visualització dels diferents gràfics disponibles.

Creació de les caselles de selecció per cada continent

Intentar crear la secció de caselles de selecció (checkboxes), de forma manual, en el HTML, hauria estat una bogeria. No només per l’elevada quantitat de feina manual, sinó també per la poca impracticabilitat d’aquesta. La creació d'aquestes seccions, s'ha realitzat mitjançant la creació d’HTML dinàmic de Mustache.

El servidor, mitjançant Mustache, utilitza la informació continguda en el fitxer *countryParameters.js,* per crear un procés iteratiu que genera totes les caselles necessàries per a cada continent.

Per exemple, pel continent nord-americà, s'utilitza el següent bloc de codi. Recordem que les etiquetes {{#northAmerica}} i {{/northAmerica}}, indiquen que es vol replicar el HTML situat entre elles per cada element del vector *northAmerica.*

*{{#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 l’objecte *northAmercia,* es representa el nom del país (*{{name}}*  i s'emmagatzema el codi d'aquest en el camp ID (*{{code}})*, per tal de ser utilitzat com a paràmetre en la creació del mapa geogràfic.

La figura [] mostra un exemple de com queden disposades les diferents caselles de selecció.

**Aspectes d'usabilitat considerats**

Aquesta funcionalitat aprofita molts conceptes d'usabilitat explicats en la funcionalitat de cerca a l’arbre familiar i que per tant, no repetirem en aquesta secció, però si anomenarem.

* La funcionalitat també implementa la interacció sobre certes capçaleres del formulari, per tal expandir o contraure parts d'aquest. També utilitza una barra sobreposada al final de la finestra, per facilitar l'accés al botó de cerca.
* Els missatges d'error que puguin aparèixer en prémer el botó de cerca 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 se citen alguns aspectes d'usabilitat propis d'aquesta funcionalitat.

Navegació pels gràfics de diferents anys

Aquesta funcionalitat, disposa d’una barra de navegació, que permeten canviar l’any sobre el qual els gràfics mostren informació. Aquesta barra, es troba a l’inici de la zona de resultats i es manté fixa i sobreposada als altres elements del HTML, si s’arriba a certa profunditat en la pàgina web.

Aquesta barra de navegació, cobreix un doble objectiu. Primerament, informar a l'usuari sobre a quin any pertanyen les dades dels gràfics pintats i en segon lloc, permetre la navegació a través de les dades dels diferents anys, si aquestes existeixen.

La navegació a través dels gràfics de diferents anys, només es troba disponible, si s'han demanat dades per més d'un any. A més a més, s’han creat regles per impedir l'accés a gràfics que no existeixen, a través d’aquests controls.

La imatge [] mostra un exemple d'aquesta barra de navegació.  
  
Progressió de la cerca

Un dels aspectes d'usabilitat més interessants d'aquesta funcionalitat és la secció dedicada a mostrar la progressió de la cerca. Si ja havíem comentat que a mesura que es reben les dades dels diferents anys, aquestes són representades en els gràfics pertinents, aquesta secció de la pàgina compleix un rol similar, però més informatiu.

En el moment que es llança una cerca, l'usuari és transportat a aquesta secció. Durant el temps de cerca, aquesta secció mostra el cognom pel qual s'està realitzant la cerca, la duració estimada calculada mitjançant la fórmula: númeroPaïsos \* númeroAnys \* apiDELAY, la informació sobre el país i any dels quals s'està esperant la resposta per part del SDK i una barra del progrés actual, respecta el total estimat.

Una cosa que cal tenir en compte és que les crides al SDK són asíncrones i que per tant, no està garantit que el retorn d'aquestes segueixi el mateix ordre que l'ordre d'enviament. Això significa, que potser, el bloc HTML que mostra la progressió de la cerca, pot indicar que s’estan esperant dades que ja han arribat o viceversa. De totes maneres, s’espera que en la gran majoria dels casos, hi hagi una correlació entre l’ordre d’enviament i retorn de les peticions al SDK.

En qualsevol cas, el fet més important relatiu a la barra de progrés, és que aquesta arribi al 100% quan s'han processat totes les crides a l’API i que cada crida emmagatzemi el resultat a les caselles de la matriu que li pertoca. Recordem que això és aconseguit gràcies als paràmetres *i* i *k* encapsulats en les crides.

Quan la cerca és completada, canvia l'estat dels components de la secció i s'indica que la cerca ha estat completada pel cognom especificat, s'elimina el temps estimat de finalització i és substituït per una indicació sobre la localització dels resultats i s'indica el nombre total de països i anys cercats. L'efecte de moviment en la barra de progressió, també és eliminat, per tal d'evitar confondre a l'usuari.

La imatge [] mostra dos estats diferents de la secció progressió de la cerca.

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 poden no haver quedat cobertes en les seccions anteriors es llisten a continuació:

* Quan es prem el botó de cerca, el text d'aquest canvia a `Searching now...' i passa a un estat de desactivació que n'impedeix l’utilització fins que l'estat actual és resolt. Quan la cerca finalitza o un és produeix un error, l'estat del botó torna a la seva normalitat.
* Quan es realitza una nova cerca, els resultats de l'anterior s'amaguen per no causar confusió.
* Els missatges d'error provinents del SDK o errors de validació del formulari, desapareixen en quan es llança una nova operació de cerca.
* Quan s'utilitzen les fletxes de navegació, de la barra de navegació, els gràfics apareixen i desapareixen per tal de fer palpable que aquests han estat refrescats.
* S'ha inclòs, dins de cada continent, la possibilitat de seleccionar o desseleccionar tots els checkboxes, mitjançant un botó.
* El gràfic de línees només es mostra si existeix més d'un any de dades, evitant d’aquesta forma la redundància amb el gràfic de barres, en el cas que només s’hagin consultat dades per un any concret.
* El gràfic de barres ordena els països de més instàncies a menys, per facilitar-ne la comprensió i obtenir una millora visual.

**Interès d'ús**

Aquesta funcionalitat, cobra un esepcial interès de cara a l’estudi de tendències d'emigració importants, donades per circumstàncies històriques.

També hem volgut implementar aquesta funcionalitat perquè els usuaris poguessin observar, com les instàncies d'un cognom en concret, poden ser trobades en infinitat de països diferents i que cada un de nosaltres, probablement, tinguem orígens completament dispersos.

La funcionalitat evolució geogràfica d’un cognom, pretenia ser només un petit test de la potencialitat dels estudis que es poden realitzar, al relacionar dades genealògiques amb agrupacions per país o comerca i com s’ha pogut veure en la secció de propostes de projecte, aquesta ha estat ampliada i diversificada, per representar un parell de projectes, que creiem podem ser d'interès pels futurs estudiants.

**Evolució temporal d'esdeveniments**

**Descripció de la funcionalitat**

La funcionalitat evolució temporal d'esdeveniments, permet als usuaris explorar el nombre d'instàncies de naixements, casaments i defuncions, enregistrades a les bases de dades de FamilySearch, per un país, al llarg d'un període d'onze anys.

Aquesta funcionalitat, probablement la més simple de les tres implementades, ja que reutilitza molta part del codi de les anteriors, és també una de les més intersants de cara a les diferents utilitzacions que si li poden donar.

Aquesta funcionalitat, de la mateixa forma que l'evolució geogràfica d'un cognom, utilitza la funció de conveniència del SDK *getResultsCount(),* per millorar l’eficiència en l’obtenció de resultatsi per tant, no entrarem a enumerar els seus avantatge una altre vegada.

La funcionalitat, llançarà un total d’onze crides al SDK, una per cada any de l’interval a considerar.

L'evolució temporal d'esdeveniments permet a l'usuari configurar els següents tres paràmetres:

* **Esdeveniment:** A seleccionar entre neixaments, casaments o defuncions.
* **Localització:** Localització en la que es cercarà les instàncies del esdeveniment seleccionat.
* **Any central:** Any central del període d’onze anys pel que es realitzarà la cerca. La funcionalitat genera l’interval +/- 5 anys, respecte l'any introduït. Així, per exemple, si introduïm l'any 1942, el rang d'anys utilitzat serà 1937-1947, ambdós inclosos.

Un cop el SDK retorna resultats per totes les consultes enviades, es pinta un gràfic de línees que mostra l'evolució del nombre d'instàncies de *esdeveniment,* trobades al llarg del període seleccionat.

**Recomanacions d'utilització**

Aquesta funcionalitat no restringeix el nivell de localitzacions a utilitzar, com fa per exemple la funcionalitat d'expansió geogràfica d'un cognom i replica, de fet, el funcionament de les localitzacions que s’introduien en la funcionalitat de cerca.

El paràmetre localització, per tant, accepta una gran diversitat de nivells d'especificació diferents, però per assegurar que la cerca produeix resultats, cal sempre introduir, com a mínim, el nivell de província o estat. De totes maneres, per l'exploració d'aquesta funcionalitat, es recomana introduir també el país o millor encara, cercar directament per un país.

Una segona consideració a tenir en compte és que les bases de dades de FamilySearch no contenen, per tots els països, localitzacions i anys, la mateixa quantitat de registres. Per tant, utilitzar la funcionalitat en regions i períodes de temps rics en registres, produirà, amb alta probabilitat, resultats més interesants.

**Detalls d'implementació**

Abans de destacar els detalls d'implementació d'aquesta funcionalitat, volem comentar, de la mateixa forma que per les dues funcionalitats anteriors, que només el controlador que gestiona la funcionalitat està format per tres-centes línees de codi i en conseqüènca, manca de sentit intentar representar totes les funcionalitats i aspectes d'aquesta en la memòria.

El controlador encarregat del funcionament de l'evolució temporal d'esdeveniments és el fitxer *facts.js.*

Validació del formulari

Aquesta funcionalitat utilitza el mateix sistema de validació en línea i validació en el moment de cerca, explicada en detall en la funcionalitat anterior, evolució geogràfica d’un cognom.

Evidentment, de la mateixa forma que les dues funcionalitats anteriors, aqueta també escapa els valors obtinguts del formulari, abans d’enviar-los al SDK, per tal d'evitar injeccions de codi i atacs al sistema.

Les regles de validació per cada un dels camps del formulari es llisten a continuació:

* **Esdeveniment:** En principi no pot existir un estat en que cap dels esdeveniment es trobi seleccionat, però en cas de forcar-ho per edició del HTML, es dispara un error en el cas que cap tipus d'esdeveniment estigui seleccionat.
* **Localització:** Qualsevol localització és acceptada sempre i quan el camp no es deixi en blanc. Suggerim des d'aquesta part de la memòria, la utilització d’un país com a varem de proves i que es tinguin presents les consideracions d'utilització descrites en l’apartat anterior.
* **Any central:** Aquest camp és acceptat com a vàlid, si el valor introduït te longitud quatre i és un número.

Per observar el resultat visual de la validació en línea o validació en el moment de cerca, es pot observar al figura [ref] de la funcionalitat anterior.

Regulació de les crides asíncrones

De la mateixa forma que la funcionalitat d'expansió de cognoms geogràfics, aquesta funcionalitat llança múltiples crides asíncrones contra el SDK de FamilySearch.

En concret, aqueta funcionalitat llença onze crides que per evitar ser bloquejades per la funcionalitat de 'throttling' de la API de FamilySearch han estat serialitzades imposant una pausa de dos segons entre la crida de cada una d'elles.

Aquesta serialització s'ha aconseguit, de forma anàloga a la descrita en detall a la funcionalitat d'expansió geogràfica de cognoms, mitjançant la funció *setTimeout()* de jQuery i el paràmetre *apiDELAY* que s'encarrega d'indicar l'interval d'espera entre les diferents crides. També s'han encapsulat les crides en una funció que emmagatzema com a paràmetre a quina de les onze crides correspon la iteració.

A continuació mostrem el bloc de codi reduït que representa la serialització de les crides a la API.

*for(var i = 0; i < 11; i++) {  
 ...  
 (function(i) {  
 setTimeout(function() {  
 ...  
 client.getPersonSearch(params).then(function(searchResponse) {  
 var total = searchResponse.getResultsCount();  
 linechartRows.push([String(firstYear+i), total]);  
 ...  
 });  
 }, apiDELAY\*i);  
 }(i));  
}*

Aquestes crides retornen al controlador una per una però sense poder garantir l'ordre de rebuda. Per aquest motiu, cobra especial importància el empaquetar en cada una de les crides la iteració del bucle al que correspon (paràmetre 'i'), ja que és aquest valor el que s'utilitza per emmagatzemar els resultats on toca.

La variable global encarregada de guardar els resultats de les diferents crides al SDK és la mateixa que la que s'utilitza pel gràfic de línees de la funcionalitat expansió geogràfica d'un cognom i que en aquest cas consisteix d'una matriu d'una sola columna on columna representa la localització cercada i cada fila el valor per un any diferent.

Explicar que utilitzem una matriu d'una sola columna en comptes d'un vector perquè és el format que necessitarem per tal de que el gràfic retornat per l'API de Google entengui les dades i les representi de la forma que nosaltres volem. La quantitat d'instàncies trobades en l'eix vertical i els anys al llarg de l'eix horitzontal.

La línea de codi encarregada de guardar els valors per cada crida a la API és mostra a continuació. Recordem que el paràmetre 'i' fa referència a la iteració del bucle executada i per extensió a quin any dels onze que conformen el interval cercat.

*linechartRows.push([String(firstYear+i), total]);*

Impressió del gràfic

La funcionalitat evolució temporal d'esdeveniments utilitza una variable global per comptar quantes iteracions de les enviades al SDK han estat retornades fins al moment. La variable en qüestió porta el nom de *yearsConsulted.*

En el moment en que una petició al SDK retorna i aquesta variable passa a valer onze, significa que ja s'han rebut totes les dades i que per tant poden ser impreses. Les línees de codi que realitzen la gestió d'aquesta variable dins de les crides del SDK es mostren a continuació.

*client.getPersonSearch(params).then(function(searchResponse) {  
 yearsConsulted = yearsConsulted + 1;  
 linechartRows.push([String(firstYear+i), total]);  
 if(yearsConsulted == 11) printLinechart();  
});*

Per altre banda, la funcionalitat encarrega de imprimir el gràfic de línees segueix la mateixa pauta que qualsevol gràfic que vulgui ser imprès mitjançant l'API de Google.

1. Preparar les dades en el format final que Google requereix per interpretar-les de forma correcte.
2. Crear el tipus de gràfic desitjat mitjançant una crida a l'API de Google i indicant-ne el contenidor HTML en el que serà representat.
3. Pintar el gràfic en el client.

Un exemple del gràfic imprès per aquesta funcionalitat pot ser vist en la figura []

**Aspectes d'usabilitat considerats**

Aquesta funcionalitat, com és normal, reutilitza molts dels aspectes d'usabilitat ja esmentats i explicats en detall en les seccions anteriors.

En conseqüència, molts d'aquests no seran explicats al detall per aquesta funcionalitat però si que els anomenarem per tal de deixar-ne constància.

* 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 la explicada en la funcionalitat de cerca.
* La navegació vertical animada també forma part d'aquesta funcionalitat quan es donen situacions que alteren l'estat de la pàgina. Per exemple, prémer el botó de cerca o l'aparició d'un error.

Progressió de la cerca

La progressió de la cerca és una secció del HTML d'aquesta funcionalitat molt similar al de la funcionalitat expansió geogràfica de cognoms.

La finalitat d'aquesta secció és un altre cop al de proporcionar a l'usuari un indicar del progrés de cerca realitzat fins el moment i el temps estimat per la finalització d'aquest. L'objectiu, reduir la frustració de l'usuari i ensenyar de forma transparent la feina realitzada.

No entrarem en els detalls més tècnics d'aquesta funcionalitat perquè ja han estat explicats en la funcionalitat d'expansió geogràfica de cognoms.

La informació que es mostra a l'usuari en aquesta funcionalitat és el tipus d'esdeveniment pel que s'està realitzant la cerca, el país introduït i l'any central sobre el que es cerca.

Aquesta funcionalitat també representa una barra de progrés que va progressant a intervals de 9% a mesura que el SDK va resolent les peticions enviades sense importar l'ordre en la que aquestes retornen.

Un cop ha finalitzat la cerca i el gràfic de línees ha estat pintat, s'indica en aquesta secció mitjançant text que la cerca de l'esdeveniment seleccionat ha finalitzat i s'elimina l'efecte de moviment en la barra de progressió per tal de no confondre a l'usuari.

La figura [] mostra els dos estats diferents d'aquesta secció de la funcionalitat.

Representació de l'estat

Les funcionalitat s de l'aplicació es caracteritzen per intentar mantenir una representació de l'estat ideal a cada moment independentment del que hagi pogut passar en anterioritat.

Les diferents micro transicions d'estat que succeeixen per aquesta funcionalitat i que poden no haver 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.
* Quan es realitza una nova cerca, els resultats de l'anterior s'amaguen per no causar confusió.
* Els missatges d'error provinents del SDK o validacions del formulari desapareixen en quan es llança una nova operació.

**Interès d'ús**

Aquesta funcionalitat, com ja hem comentat, a pesar de ser de les més simples que hem implementat pot ser la que tingui més interès un cop s'obté accés a les dades de producció.

La funcionalitat va ser programada per tal de poder intentar relacionar fets històrics com la gran recessió del vint-i-nou o la segona guerra mundial en les tasses de natalitat i defuncions.

Un objectiu secundari d'aquesta funcionalitat era el d'intentar comprendre si les bases de dades de FamilySearch representaven una fotografia de la realitat o com de desviades es troben aquestes. Per exemple, és conegut que durant la segona guerra mundial el nombre de matrimonis es va disparar als Estats Units d'Amèrica. L'idea d'aquesta funcionalitat és comprovar si aquestes relacions conegudes són també observables en les dades de l'organització.