VISITES VIRTUALS INTERACTIVES

DESENVOLUPAMENT D'UNA APLICACIÓ WEB PER A LA CREACIÓ

DE VISITES VIRTUALS INTERACTIVES

Batxillerat Tecnològic

Tutor: Manuel Gil Pérez

Escola Arrels II

Marc Pujol Gualdo

2n de Batxillerat

Curs 2016-2017

AGRAÏMENTS

Agraeixo l'ajuda a tots aquells qui me l'han donat durant tots aquests mesos de recerca.

Al meu tutor Manuel Gil, per ajudar-me i donar-me consell, no només durant les revisions sinó sempre que l'he necessitada. Als amics i familiars, sobretot els meus pares, que s'han interessat pel meu treball i m'han transmès la seva opinió i els seus suggeriments. Especialment a Marc Segués, per confiar en el meu treball de recerca i utilitzar-lo com a eina per realitzar una part del seu treball.

Però no només vull donar les gràcies a qui m'ha ajudat durant el treball de recerca. Vull agrair a tots els que m'han estat ajudant i han facilitat el meu procés d'autoaprenentatge en la programació.

Al Gerard Farràs i altres professors de l'escola i sobretot a molts usuaris (en molts casos anònims) d'internet que m'han resolt dubtes i m'han guiat en aquest procés.

En particular vull donar les gràcies a Ignacio Puig, Juan Badino, Antonella Re, Facundo Rubin i Hernán Rajchert, membres de l'equip d'Acamica, una *startup* d'Argentina que ofereix cursos de programació; i on, juntament amb ells, sóc moderador dels fòrums de la comunitat.

Per últim, només em falta agrair-te a tu, lector, haver-te interessat i destinar una part del teu temps en el meu treball.

Moltes Gràcies.

ÍNDEX

1. ABANS DE COMENÇAR

Sempre he cregut que les imatges, especialment els vídeos, són una eina que facilita la comprensió de qualsevol text. És per això que he buscat un mètode per poder inserir vídeos al llarg del meu treball escrit. Ja que no és possible fer-ho de manera física, ho he fet de manera virtual.

Aquests vídeos es poden veure sobre el paper, tot i que a través de la càmera del telèfon mòbil. És el que s'anomena realitat augmentada.

Per poder-los veure, cal descarregar-se l'aplicació "Aurasma" al telèfon mòbil. Després de registrar-se o d'iniciar sessió cal buscar "mgualdo" a la barra de cerques i prémer el resultat "Visites Interactives". Per últim, cal prémer el botó "follow" que es troba en aquesta pantalla.

Ara ja es pot tornar al menú principal de l'aplicació i prémer el botó d'escanejar perquè s'obri la càmera del mòbil.

No veurà cap vídeo fins que no enfoqui una imatge que contingui realitat augmentada. En fer-ho, automàticament apareixerà el vídeo corresponent.

Les imatges que disposen d'aquest tipus de contingut contenen el logotip de l'aplicació a la part inferior dreta, a més d'una explicació al peu d'imatge.

2. GLOSSARI

El meu treball de recerca es basa en el desenvolupament web, acció per la qual és imprescindible tenir certes nocions en programació. He elaborat un glossari on faig un recopilatori d'alguns tecnicismes d'aquesta disciplina per facilitar la comprensió del treball a usuaris sense aquests coneixements. Més enllà de conèixer el significat i la funció de les següents paraules, la meva intenció és acostumar el lector al codi per tal que pugui entendre els petits fragments que utilitzo de mostra al llarg del treball.

Poden trobar el glossari a l'**ANNEX 1**.

Al llarg del treball apareixeran diferents fragments de codi per les explicacions del funcionament. Resulta impossible contextualitzar els fragments dins el codi que els engloba, per la qual cosa no és possible explicar-ne totes les línies amb la màxima precisió.

Cal parar molta atenció als comentaris de codi, ja que aquests expliquen què està passant a nivell de codi en tot moment.

```
/* Això és un comentari. No afecta al codi i serveix per explicar-ne el funcionament*/
```

Com que els arxius de codi són molt extensos, en mostraré tan sols aquelles parts que tinguin una vital importància i relació amb el tema que s'està parlant en cada moment. Escriuré tres punts en un comentari per indicar fragments que no mostro però que hi són. En moltes ocasions poden ser pàgines i pàgines senceres.

```
/* ... */ en Javascript i  <!-- ... --> en HTML
```

3. INTRODUCCIÓ

Aquest treball de recerca és per a mi un repte que em vaig plantejar per aprofundir els meus coneixements i habilitats en la programació. Concretament en el desenvolupament web.

Vaig decidir partir de la base d'un servei que s'ofereix actualment per desenvolupar-ne un de molt més complex i alhora accessible.

El servei en el qual em vaig basar són les visites virtuals.

Una visita virtual és un conjunt de fotografies esfèriques connectades entre sí que permeten recrear la visita que faria una persona a un indret físic determinat d'una manera virtual. Permeten moure's entre fotos que engloben tots els angles de visió: de dalt a baix (180º) i d'esquerra a dreta (360º). És a dir, fotos que ocuparien tot l'interior d'una esfera.

Hi ha diferents mètodes per aconseguir una fotografia esfèrica. Des d'utilitzar càmeres especials, càmeres normals amb accessoris o simplement la càmera del mòbil.

Si aquestes fotos són capturades en intervals de poca distància, el resultat és encara més realista, ja que llavors moure's d'una fotografia a una altra crea l'efecte de caminar.

Quan es visualitza una visita, no es veu la fotografia esfèrica com una imatge plana, sinó que s'utilitzen visualitzadors que mostren tan sols una part de la imatge i en deformen les cantonades, per simular el que veurien els ulls humans.

Fotografia esfèrica vista com una fotografia plana.

Fotografia esfèrica vista des d'un visualitzador. Escanegi la imatge per veure-la en moviment.

En aquests visualitzadors, doncs, un usuari pot interaccionar mitjançant dues accions. La de **girar/rotar** (canviar l'angle de visió de la fotografia) o la d'**avançar** (canviar de fotografia).

Em vaig adonar que gairebé tots els serveis de visites virtuals que existeixen es limiten a connectar imatges esfèriques entre sí per permetre'n la navegació. Només hi ha alguns serveis puntuals que permetin afegir text o música de fons.

La meva idea era crear un tipus de visita virtual molt més interactiva, on l'usuari pogués fer moltes més accions que no només avançar i girar. Això ho aconseguiria afegint punts interactius.

Punt interactiu de tipus "text" adjuntat a la finestra.

Els punts interactius són una icona gràfica que apareix en el visualitzador superposada en un punt de la imatge, i s'hi manté fixa.

Així doncs, per exemple, en una visita virtual en un restaurant, es podria assignar un punt interactiu a una taula en concret, i aquest només es veuria quan l'angle de visió engloba aquella taula i es mouria d'acord amb els moviments de rotació. A més a més, en passar el cursor per sobre d'un punt interactiu, en veuríem la seva funcionalitat.

És a dir, cada punt interactiu està representat per una icona (cada tipus de punt interactiu té una icona diferent que el representa), però en passar-hi per sobre poden aparèixer molts més continguts.

Aquests continguts poden ser imatges, textos, vídeos, mapes, enllaços… En l'exemple del restaurant, en passar el cursor pel punt interactiu es podria mostrar un formulari per reservar aquella taula. I en ser completat, enviar-lo al propietari.

Així doncs, la meva idea era desenvolupar una aplicació web mitjançant la qual hom fos capaç de generar visites virtuals interactives. Per fer-ho, vaig adonar-me que el millor era separar el desenvolupament en dos entorns web:

- **Una aplicació web** on, entre d'altres funcionalitats, qualsevol usuari registrat pogués generar una visita.
- **Una plataforma web** on es renderitzessin les visites creades mitjançant l'aplicació web. A més, s'hauria de poder compartir i inserir amb facilitat.

És evident que ambdós entorns estan molt relacionats. A grans trets podem dir que l'aplicació web **genera** les dades que necessita una visita i la plataforma web les **rep**.

La metodologia que he emprat per realitzar aquest treball ha estat l'autoaprenentatge a través d'Internet. Per tant, el meu treball no ha requerit realitzar enquestes ni recopilar opinions, ja que la meva recerca ha consistit en aconseguir portar a la pràctica les idees que em passaven pel cap. O més ben dit, aconseguir transformar-les en codi, descobrint la forma més adient per fer-ho.

La meva gran pregunta abans de començar era si fóra capaç de crear tot sol i en pocs mesos un sistema com el que em plantejava, afegint funcionalitats que les empreses actuals no ofereixen.

4 D'ON PARTEIXO?

Un dels principals fonaments en la programació i en la ciència en general és no reinventar la roda. És a dir, no desenvolupar de nou allò que ja està desenvolupat.

Seguint aquest conveni, he seleccionat dos projectes per tal de començar a desenvolupar-hi el meu a partir d'aquests.

4.1 Metronic - Responsive Admin Dashboard Template.

El primer és una plantilla de disseny web basada en Bootstrap anomenada *Metronic - Responsive Admin Dashboard Template*.

El que em permet aquesta plantilla és donar estil, forma i colors al meu contingut d'una manera molt ràpida, permetent-me retocar tot el que vulgui i afegint al mateix temps els meus propis dissenys.

Per conèixer més informació d'aquesta plantilla i com l'he utilitzada, vegeu l'**ANNEX 2.1**.

4.2 Pannellum

El segon projecte que he utilitzat per desenvolupar el meu projecte és Pannellum.

Pannellum és un visualitzador de fotografies esfèriques per la web de codi lliure. El gran avantatge d'aquest visualitzador és que disposa d'un sistema per referir-se a qualsevol punt de la imatge i afegir-hi punts interactius de tipus "text".

Així doncs, he utilitzat Pannellum com a base de la plataforma web.

Per a més informació de les característiques i del funcionament d'aquest visualitzador, vegeu l'**ANNEX 2.2**

5. DOMINI I HOSTING

És obvi que per realitzar el meu treball pràctic necessitava "una web".

Però què és "una web"? Més enllà del contingut, per publicar una web a internet necessitem un nom que la identifiqui i que hi accedeixi. Aquest nom és el domini (www.visitesinteractives.cat).

Per altra banda, també necessitem penjar els arxius que contenen el nostre codi a "algun lloc", perquè es vegin a la web. El servei per gestionar-ho s'anomena Hosting.

Jo he contractat aquests dos serveis en una empresa que ja havia utilitzat anteriorment: swhosting (www.swhosting.com/ca). He triat el pla "Hosting Micro". Inclou un domini, un GB de disc, un compte de correu i un compte FTP (compte necessari per penjar els arxius al hosting contractat). Encara que no ofereixi una gran quantitat d'espai, a mi ja em serveix, perquè la majoria de dades les guardaré en una base de dades i al núvol.

6. DESENVOLUPAMENT DE L'APLICACIÓ WEB

6. 1 NAVEGACIÓ

El meu treball pràctic necessita de centenars d'arxius per funcionar. Els d'estil (CSS) i els scripts (JS) es

connecten als de contingut (HTML) per dotar-los d'estil i funcionament; i són aquests últims als quals l'usuari pot accedir mitjançant l'URL. (www.domini.com/**contactar.html** ens indica que ens trobem a l'arxiu *contactar.html*)

Cal dir que no sempre és necessari escriure el **/document.html**. L'adreça www.domini.com carregarà l'arxiu per defecte (normalment **index.html**) encara que no s'indiqui.

En la meva web hi ha tan sols tres pàgines per les quals es poden moure els usuaris:

- **Visita.html** → Pertany a la plataforma web. És la pàgina on es visualitzen les visites creades, encara que www.visitesinteractives.cat/visita per sí sol no mostra cap visita. Necessita rebre més informació que més endavant detallaré.
- **Login.html** → Pertany a l'aplicació web. És la pàgina on entren els usuaris quan no tenen o no han iniciat una sessió.
- **Index.html** → Pertany a l'aplicació web. És la pàgina principal. Per accedir-hi és imprescindible tenir una sessió iniciada.

Encara que **index.html** sigui un sol arxiu, dins seu en mostra molts més.

Per tant, aquest arxiu no conté cap contingut propi, sinó que conté divs (caixes) que s'omplen amb contingut d'altres arxius.

Distribució dels arxius en index.html

Algunes de les caixes són el *header* (capçalera), el *footer* (peu de pàgina), el *sidebar* (menú lateral) o la barra superior. Totes aquestes parts són arxius individuals que s'insereixen a la zona assignada a *index.html*.

Hi ha una altra part (CONTINGUT a la captura de pantalla) que és dinàmica. Això vol dir que en aquesta part es mostra un arxiu o un altre segons requereix l'aplicació.

Gràcies a AngularJS, puc gestionar aquest contingut depenent de la URL.

Una URL que sempre seguirà aquesta forma: www.visitesinteractives.cat/#/**part-variable**, que seria l'equivalent a www.visitesinteractives.cat/index.html#/**part-variable**.

On /#/**part-variable** és la part que canvia (/#/***dashboard***, /#/***usuaris***, /#/***visites***, /#/***xat***, /#/***estadístiques***...).

És molt important entendre que */#/quelcom* és el mateix que www.visitesinteractives/#/quelcom, ja que constantment faré referència a la forma abreujada d'escriure-ho.

En el següent fragment de codi es pot entreveure el funcionament d'$stateProvider, l'eina d'Angular encarregada de la navegació.

```
$stateProvider.state('dashboard', {
    url: "/dashboard", // Quan la url és www.visitesinteractives.cat/#/dashboard
    templateUrl: "views/dashboard.html", // especifiquem l'arxiu que es mostrarà a l'apartat
CONTINGUT
    data: {
        pageTitle: 'Taulell principal'
    },
    controller: "DashboardController", //especifiquem el controlador que s'utilitzarà
    resolve: {
        deps: ['$ocLazyLoad', function ($ocLazyLoad) {
            return $ocLazyLoad.load({
                name: 'VisitesVirtualsApp',
                       insertBefore: '#ng_load_plugins_before', // especifiquem alguns arxius
       (estils i scripts) que requereix específicament aquesta pàgina.
                files: [
                    '../assets/pages/scripts/dashboard.min.js',
                    'js/controllers/DashboardController.js',
                ]
            });
        }]
    }
})
```

En aquest cas hem vist la configuració per a /#/**dashboard**, però aquest codi es repeteix tantes vegades com pàgines hi ha.

També hi ha URLs que tenen més d'un nivell, com www.visitesinteractives.cat/#/**visita/-KLa46rql50gF-siU-zi**.

En aquest cas /#/**visita** ens informa que necessita mostrar l'arxiu que dóna informació de cada visita (una plantilla igual per totes les visites), i /#/**-KLa46rql50gF-siU-zi** és un ID que ens indica de quina visita estem parlant per poder-ne obtenir la informació i mostrar-la en la plantilla (més endavant explicaré com).

Així doncs, una vegada entenem el funcionament dels URLs és fàcil moure's per l'aplicació canviant-los.

Per exemple, /#/**estadístiques** ens mostrarà les estadístiques de totes les visites creades, però sabent l'ID d'una visita podem moure'ns a /#/**estadístiques/-KLa46rql50gF-siU-zi** per obtenir les estadístiques detallades d'aquella visita.

El mateix passa amb /#/**perfil/*id***, /#/**xat/*id***...

També hi ha l'opció de redirigir enllaços

```
// envia de /gràfics a /estadístiques
$urlRouterProvider.when('/gràfics', '/estadistiques/');
// per una URL desconeguda redirigeix a /dashboard
$urlRouterProvider.otherwise("/dashboard");
```

Per veure l'arxiu Javascript complet on es detalla la configuració de la navegació, vegi's l'**ANNEX 4.1.**

6.2 AUTENTIFICACIÓ

Per utilitzar l'aplicació web és necessari registrar-se.

Els usuaris no registrats tan sols poden veure les visites creades a la plataforma web.

A part de crear visites, hi ha altres funcionalitats pròpies d'una visita que només estan disponibles a l'aplicació web, com els comentaris, la possibilitat de veure i descarregar les imatges originals de la visita, veure totes les versions d'aquella visita, veure el perfil del creador i veure'n les estadístiques.

Per posar un exemple, en una visita amb ID **-KLa46rql50gF-siU-zi**, un usuari registrat podrà accedir a www.visitesinteractives.cat/visita?id=-KLa46rql50gF-siU-zi (plataforma web) i veure'n la informació addicional que he esmentat a www.visitesinteractives.cat/#/visita/-KLa46rql50gF-siU-zi (aplicació web). Un usuari no registrat només podrà veure www.visitesinteractives.cat/visita?id=-KLa46rql50gF-siU-zi (plataforma web).

Per manejar els usuaris i les seves sessions he utilitzat Firebase Auth, un dels serveis de Firebase. Vegin més informació de Firebase a l'**ANNEX 1.7**.

Les funcions principals d'aquest servei són les de registrar usuaris i iniciar-ne la sessió.

Captura de pantalla de login.html

Totes aquestes accions es porten a terme a la pàgina www.visitesinteractives.cat/login.html, la qual consta de tres formularis. Un per iniciar sessió, un per crear un compte i un per restablir la contrasenya en cas d'oblidar-la.

Per veure el codi HTML de la pàgina login.html, vegeu l'**ANNEX 3.2**.

Cal recordar que aquesta pàgina (login.html) només és accessible per a usuaris no autentificats. Per tant, necessito un mètode per detectar usuaris autentificats i redirigir-los a la pàgina d'inici (index.html).

```
    //el següent codi s'executa cada vegada que canvia l'estat d'autentificació (quan passa a
haver-hi/deixar-hi d'haver un usuari) i també quan es carrega la pàgina.
    firebase.auth().onAuthStateChanged(function (user) {
        if (user) {
            /* HI HA UN USUARI ACTIU */
            /*... COMPROVA ELS CASOS DE SOCIAL LOGIN (EXPLICAT MÉS ENDAVANT) ...*/
          window.location = "http://www.visitesinteractives.cat"; //Redirigim a la pàgina
       principal
        } else {
           /* NO HI HA CAP USUARI ACTIU */
        }
    });
```

6.2.1 REGISTRE D'USUARIS

El primer mètode i el més tradicional per registrar un usuari és fer-ho a partir d'un correu electrònic i una contrasenya que ell mateix proporciona.

Firebase dota cada usuari d'un ID únic, i l'emmagatzema junt amb el correu i la contrasenya, però no a la base de dades (per seguretat). Per tant, una vegada un usuari es registra, n'agafo l'ID que Firebase em proporciona i creo un apartat a la base de dades (/usuaris/ID) on podré guardar totes les dades que aquest usuari pugui generar, començant per les dades que havia entrat al formulari de registre, la data de registre i una fotografia de perfil per defecte.

A continuació adjunto un fragment de codi per "il·lustrar" el que acabo d'explicar.

```
// Funció cridada quan es prem el botó REGISTRAR
registrar = function () {
```

```
    // obté les dades del formulari i les guarda en variables.
    var nom = $(".register-form .nom").val() + 3" " + $(".register-form .cognoms").val(),
        email = $(".register-form .email").val(),
        pass = $(".register-form .pass").val();

    /* ... CODI QUE VALIDA LES DADES DONADES ... */

    // crida la funció de Firebase encarregada de registrar usuaris, passant el correu i la
contrasenya com a paràmetres
    firebase.auth().createUserWithEmailAndPassword(email, pass).then(function (data) {
        /* JA S'HA REGISTRAT L'USUARI
       PRENEM L'ID (data.uid) I L'AFEGIM A LA BASE DE DADES  */
    }).catch(function (error) {
        /* MANEJEM L'ERROR */
    });
};
```

El segon mètode per registrar usuaris és el que es coneix com a *Social Login.* Aquest mètode consisteix en aprofitar un compte existent de l'usuari en algun dels serveis més populars d'internet com Google o Facebook.

D'aquesta manera, no cal que l'usuari generi i recordi una nova contrasenya, i és probable que ja tingui la sessió iniciada quan entra a l'aplicació web.

Els proveïdors compatibles que he configurat són Google, Facebook, Twitter i GitHub.

Tal com he explicat abans, una vegada l'usuari està registrat necessito afegir-lo a la base de dades, però el codi que registra un usuari amb aquest mètode no em torna un *callback* (m'avisa quan s'acaba de registrar) amb les dades de l'usuari registrat per poder-ho fer, sinó que em torna a carregar la pàgina *login.html* amb l'usuari actiu registrat a Firebase (però no a la base de dades) i amb la sessió iniciada.

Puc detectar aquest usuari amb el codi que he mencionat anteriorment.

Amb aquest codi, en cas que l'usuari hagi estat registrat mitjançant els proveïdors socials, li atribueixo un senyal en afegir-lo a la base de dades per saber que ja ha estat afegit. Del contrari, cada vegada que iniciés sessió s'afegiria novament a la base de dades.

Això no passa amb el primer mètode perquè, a diferència d'aquest, té dues funcions. Una per registrar els usuaris (i afegir-los a la base de dades) i una per iniciar la sessió.

6.2.2 INICI DE SESSIÓ

Per iniciar sessió en un compte creat a través de *Social Login* cal prémer la icona del proveïdor corresponent, com en el cas de voler-se registrar. En veure que ja ha estat afegit a la base de dades, l'aplicació el portarà a la pàgina d'inici amb les seves dades.

Per iniciar sessió amb un compte de correu electrònic i contrasenya, utilitzo aquest codi:

```
//funció cridada en prémer el botó d'iniciar sessió
$scope.iniciarSessio = function () {
    //agafem el correu i contrasenya del DOM i el guardem en variables
    var email = $(".login-form .email").val(),
        pass = $(".login-form .pass").val();

    $scope.addAlert('Intentant iniciar sessió...', 'success');

    //cridem a la funció de Firebase per iniciar sessió passant el correu i la contrasenya per
paràmetres
    firebase.auth().signInWithEmailAndPassword(email, pass).catch(function (error) {
        //en cas d'error, el manegem
        });

    };
```

En aquest codi sembla que no es contempli l'opció d'haver iniciat sessió amb èxit, però en cas d'inici de sessió, l'encarregat de detectar-ho i manejar-ho seria la funció explicada anteriorment ***onAuthStateChanged***.

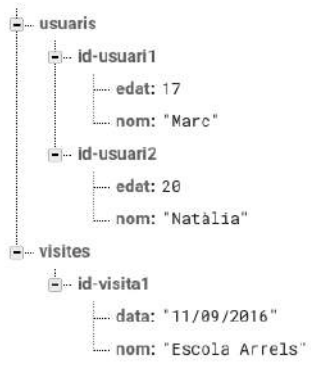
Per veure el codi Javascript encarregat de l'autentificació, vegeu l'**ANNEX 4.6**.

6.3 TRACTAMENT DE DADES

En l'aplicació web hi ha molts textos ja definits mitjançant codi; per exemple els del menú, els dels botons... però molts d'altres són generats per l'usuari, i els hem de guardar en algun lloc.

Aquest lloc és una base de dades; jo he fet servir la que em proporciona Firebase: *Firebase Realtime Database*. El que té d'especial aquesta base de dades és que qualsevol canvi que s'hi realitza es veu reflectit en tots els dispositius connectats automàticament. No és necessari refrescar la pàgina per veure els canvis.

El funcionament d'aquesta base de dades és molt simple.



Exemple d'organització d'una base de dades de tipus JSON

Funciona amb format JSON. Això significa que les dades s'emmagatzemen en forma d'arbre. Puc començar amb tres branques (visites, usuaris i xat), i de cada una, crear-ne les que vulgui, i així successivament.

Per llegir, afegir, editar o eliminar-ne qualsevol element, necessitem crear-hi una referència.

La referència és la ruta dins la base de dades de l'element que estem buscant.

```
// id_usuari és una variable que conté l'id de l'usuari
// la següent referència apunta a la informació de l'usuari amb id id_usuari.
var ref = firebase.database().ref("usuari/" + id_usuari + "/info");
```

Ara que tenim la referència, podem utilitzar-la per diferents operacions.

```
// ref = la referència que hem creat

// utilitzem AngularFire per LLEGIR les dades que hi ha a la referència
var email_usuari = $firebaseObject(ref);

// accedim al subapartat email i n'EDITEM el valor existent
ref.child("email").set("exemple@correu.com");

// apuntem al subapartat telèfon i n'ELIMINEM el valor
ref.child("telefon").remove();

// Afegim un element dins el subapartat aficions. Això genera un id únic.
// A la base de dades hi haurà --> /usuaris/id_usuari/info/aficons/id_afició
ref.child("aficions").push({
    nom: "Programar"
});
```

Ara que sabem com funciona la base de dades, anem a veure com la manegem per adaptar-la a l'usuari.

Recordem que una vegada ja hi ha un usuari actiu, aquest és redirigit a la pàgina index.html. I que index.html és una sola pàgina però que mostra diferents arxius segons la **/#/url**.

Aquests arxius mostren un contingut dinàmic que, a diferència de l'estàtic, varia segons l'usuari que el visualitza i moltes altres circumstàncies.

Per poder oferir aquest contingut personalitzat és necessari saber qui és l'usuari.

Per fer-ho, quan carrega la pàgina detectem l'usuari actiu i n'obtenim el seu id únic. Tot seguit podem anar

a la base de dades i entrar en /usuaris/**id**. Allà hi haurà totes les dades de l'usuari, a les quals tindran accés els controladors d'Angular per poder manipular i mostrar l'HTML segons convingui.

Aquesta és la teoria; ara anem a veure com transformar-ho en codi.

El codi encarregat d'obtenir les dades de l'usuari i oferir-les als controladors es troba en una *factory* (factoria) anomenada **usuari**. Dins de les factories s'executa el codi que es necessiti per obtenir uns certs valors, els quals seran accessibles pels controladors.

Detectar l'usuari actiu per obtenir-ne l'id és un procés asincrònic que es realitza a la factoria (requereix un cert marge de temps des que es fa la petició fins a aconseguir el resultat), així com també rebre les dades sol·licitades de la base de dades.

Desafortunadament la factoria ha d'aconseguir les dades que retorna instantàniament, per la qual cosa no podem retornar directament les dades de l'usuari.

Vaig estar bastant temps buscant un mètode per solucionar el problema, fins que, amb l'ajuda d'un company d'Acamica[1], vaig descobrir-lo.

El que podem fer és retornar una *promise* (promesa) a les dades de l'usuari.

Tal com el seu nom indica, una promesa promet retornar un valor, de manera que temporalment només és una "promesa" fins que aquesta és resolta (i retorna les dades que promet) o rebutjada (en cas d'error).

Així doncs, la factoria retorna una promesa a les dades de l'usuari que serà resolta en un futur així com altres dades i funcions sobre el funcionament general de l'aplicació que podran necessitar els controladors.

```
//BD = Base de Dades
VisitesVirtualsApp.factory('usuari', ['$rootScope', '$firebaseObject', '$firebaseArray',
'$state', '$q', function ($rootScope, $firebaseObject, $firebaseArray, $state, $q) {

    var UsuariPromise = $q.defer(); //creo una promesa

    firebase.auth().onAuthStateChanged(function (user) {
        if (user) {
            //hi ha un usuari actiu
            var ref = firebase.database().ref("usuaris/" + user.uid + ""); //referència a
l'usuari en la BD
            var usuari = $firebaseObject(ref); // petició a la referència
            usuari.$loaded().then(function () {
                //quan la petició es resol, resolem la promesa retornant les dades de l'usuari
```

[1] [https://www.acamica.com/comunidad/6873/como-devolver-valores-asincronicos-en-una-factory]

```
                UsuariPromise.resolve(usuari);
            });
        } else {
            //no hi ha cap usuari actiu
            UsuariPromise.reject('Usuari no registrat'); //rebutgem la promesa
        }
    });

    // seguit de funcions que retornem
    return {
        usuari: function () {
            return UsuariPromise.promise; //retornem la promesa.
        },
        /* ... ALTRES VALORS I FUNCIONS QUE NECESSITEN ELS CONTROLADORS ... */
    };
}]);
```

Per veure el codi Javascript complet de la factoria "usuari", vegeu l'**ANNEX 4.1**.

Ara tots els controladors poden accedir a aquesta promesa. Tan sols els falta esperar que es resolgui o es rebutgi per començar a funcionar amb les dades que necessiten.

```
VisitesVirtualsApp.controller('HeaderController', ['$scope', '$firebaseObject',
'$firebaseArray', 'usuari', '$state', function ($scope, $firebaseObject, $firebaseArray,
usuari, $state) {
    $scope.$on('$includeContentLoaded', function () {

        //accedim a la promesa (propietat usuari de la factoria usuari)
        var promesa = usuari.usuari();

        promesa.then(function (valor) {
                // en cas de ser resolta ja tenim accés a les dades de l'usuari
                var usuari = valor;
                /* ... CODI DEL CONTROLADOR ... */
            },
            function (error) {
                //en cas de ser rebutjada fem que l'usuari torni a iniciar sessió
                window.location = "login.html";
            });

    });
}]);
```

6.4 TRACTAMENT D'ARXIUS

Així com faig amb les dades, també necessito emmagatzemar molts arxius que es generen a la web.

Com és obvi, no puc guardar imatges, vídeos, àudios i altres arxius a una base de dades, ja que aquesta només emmagatzema cadenes de caràcters.

Firebase em proporciona un servei per això, anomenat *Firebase Storage*.

Firebase Storage funciona d'una forma similar a la base de dades.

Els arxius també s'emmagatzemen en forma d'arbre, i necessito una referència per afegir-ne o eliminar-ne.

```
//referència a Firebase Storage
var ref = firebase.storage().ref().child("visites");

//afegim un arxiu (cal que especifiqui el nom amb que es guardarà a la referència)
var PenjarFoto = ref.child("visites/pano1").put(file, metadata);
PenjarFoto.on('state_changed', function (snapshot) {
    //mentre s'està penjant ens manté informat del procés de càrrega (podem calcular el
progrés)
    var progress = (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
}, function (error) {
    //en cas d'error, el manegem
}, function () {
    //si es penja amb èxit podem obtenir una URL que apunta a l'arxiu que acabem de penjar
    var url = PenjarFoto.snapshot.metadata.downloadURLs[0];
});

//eliminem un arxiu (cal que la referència sigui a l'arxiu en concret)
firebase.storage().ref().child("visites/pano1").delete().then(function () {
    //accions una vegada s'ha eliminat
}, function (error) {
    //en cas d'error, el manegem
});
```

També hi ha un mètode per aconseguir la URL d'un l'arxiu a partir de la seva referència, però no l'utilitzo, ja que una vegada es penja un arxiu, el guardo a la base de dades.

En algunes ocasions, s'han d'emmagatzemar imatges que seran mostrades a unes dimensions relativament baixes, com les imatges del menú o el plànol. Aquestes imatges, doncs, no cal que siguin d'una extrema qualitat, ja que ocuparien espai innecessari a *Firebase Storage* i tardarien més en penjar-se i en carregar-se. Per solucionar aquest problema, vaig decidir redimensionar aquelles imatges que sobrepassessin una amplada màxima preestablerta. Després de molta recerca, vaig trobar un mètode per fer-ho des del navegador. És el següent:

```
dataURLtoBlob = function (url) {
    /* Funció necessària per convertir l'url d'un "canvas" a un objecte de tipus Blob */
};

var file; //file = l'arxiu que s'ha seleccionat
var _URL = window.URL || window.webkitURL;
var img = new Image(); //renderitzem la imatge que s'ha seleccionat per obtenir-ne les mides
reals
img.src = _URL.createObjectURL(file);
img.onload = function () {
    var MAX_WIDTH = 800; //definim una amplada màxima
    var width = img.width; //obtenim l'amplada real de la imatge
```

```
    var height = img.height; //obtenim l'alçada real de la imatge
    if (width > MAX_WIDTH) {
        //en cas que l'amplada real de la imatge superi la màxima preestablerta, generem una
imatge més petita mitjançant un "canvas".
        var canvas = document.createElement("canvas");
        var ctx = canvas.getContext("2d");
        ctx.drawImage(img, 0, 0);
        width = MAX_WIDTH; //reduïm l'amplada a la màxima establerta
        height = height / (width / MAX_WIDTH); //reduïm l'alçada amb les mateixes proporcions
amb que s'ha reduït l'amplada
        //dotem el "canvas" d'aquestes noves mides
        canvas.width = width;
        canvas.height = height;
        var ctx = canvas.getContext("2d");
        ctx.drawImage(img, 0, 0, width, height);
        var dataurl = canvas.toDataURL("image/png"); //obtenim una URL del canvas
        file = dataURLtoBlob(dataurl) //passem d'una url a un objecte que puguem penjar a
Firebase Storage
    }
    //ja podem penjar l'arxiu redimensionat (file)
    var PenjarPlanol = storageRef.child("ruta/a/la/base/de/dades").put(file, metadata);
}
```

Una diferència entre *Firebase Storage* i la base de dades és que la referència per eliminar un element ha d'apuntar a aquell element en concret. Mentre a la base de dades puc eliminar */usuaris/id_usuari* (elimina totes les dades de l'usuari, incloses les branques que en deriven) a *Firebase Storage* no puc eliminar una carpeta (és a dir, un element que encara tingui més branques).

Per tant, a l'hora d'eliminar una visita se'm presenta un problema, ja que he d'eliminar arxiu per arxiu tots els que aquesta té emmagatzemats, i això implica saber-ne totes i cada una de les referències.

Aquests arxius són, a més dels panorames, imatges del menú, plànols, música...

Per solucionar aquest problema se'm va acudir barrejar els dos serveis. Així que cada vegada que un arxiu d'una visita és penjat amb èxit a *Firebase Storage*, afegeixo la referència d'aquest arxiu a l'apartat *storage* de la base de dades d'aquella visita.

Llavors, quan vull eliminar una visita, tan sols em cal recórrer totes les referències guardades a la base de dades i fer que se n'elimini l'arxiu que contenen a *Firebase Storage*.

```
//creo una referència al llistat de referències de la base de dades
var refArxius = firebase.database().child("visites/" + id_visita + "/storage");
//demano un array amb els valors d'aquesta referència
var arxius = $firebaseArray(refArxius);
//espero a obtenir aquest array (és un procés asincrònic)
arxius.$loaded().then(function () {
    if (arxius.length > 0) { //comprova si hi ha arxius
```

```
        //crea un comptador, definint el màxim d'aquest com la quantitat d'elements que hi ha
a l'array, és a dir, la quantitat de referències.
        var comptador_max = arxius.length,
            comptador = 0;
        //per cada un dels elements d'aquest array s'executa el següent codi
        arxius.forEach(function (arxiu) {
            //creem una referència a l'arxiu que volem eliminar (la referència és el valor de
l'ítem actual de l'array = arxiu.$value)
            firebase.storage().ref().child(arxiu.$value).delete().then(function () {
                //una vegada s'ha eliminat l'arxiu
                contador++; //augmentem el comptador en 1.
                if (comptador === comptador_max) {
                    // Si el número actual del comptador és el mateix que el màxim establert,
significa que ja hem eliminat tots els arxius.
                    // Llavors eliminem la visita de la Base de Dades
                };
            }, function (error) {
                // en cas d'error, el manegem.
            });
        });
    } else {
        //no hi ha arxius, per tant només cal que eliminem la visita de la Base de Dades
    };
});
```

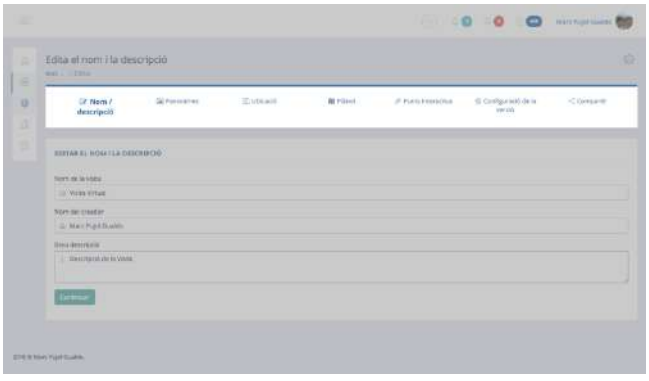
6.5 CREACIÓ / EDICIÓ DE VISITES

La creació de visites virtuals interactives així com la seva posterior edició és la funcionalitat principal de l'aplicació web.

És un procés que he dividit en set passos, on hi ha sempre un petit menú present per navegar entre ells.

S'accedeix a ells mitjançant l'acabament d'URL següent: /#/editar/**id**/**pas**

On **id** és l'id de la visita i **pas** és el nom d'un dels set passos.



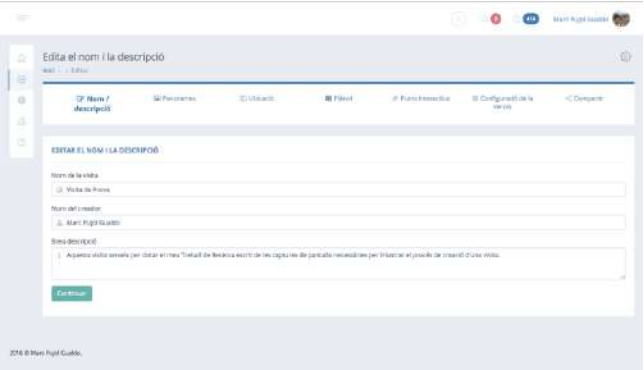
Menú per navegar entre els set passos del procés de creació d'una visita

Com es pot intuir per l'adreça, aquests passos no només es segueixen quan es crea la visita, sinó també quan se n'**edita** una de ja existent.

En cas d'estar creant-ne una, tots els passos excepte el primer romandran bloquejats, i només serà possible desbloquejar-los completant l'anterior.

Anem a entrar en detall amb cada un dels set passos:

6.5.1 NOM / DESCRIPCIÓ (/#/crear/***id***/nom)



Captura de pantalla del primer pas

En aquest pas es defineix el nom, l'autor (per defecte és l'usuari que crea la visita) i la descripció de la visita.

Depenent de com s'hi arriba aquest pas es comporta d'una manera o d'una altra.

Si s'hi arriba amb la intenció d'editar una visita ja existent, òbviament l'id es correspondrà al de la visita en qüestió i en modificar-ne les dades, les modificarà també a la base de dades.

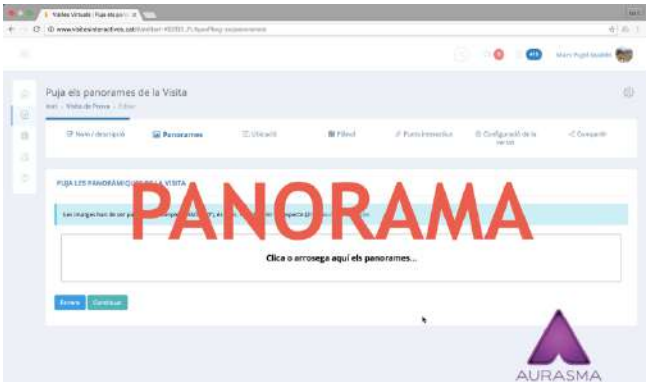
Però si s'accedeix a aquest apartat per crear una visita nova encara no podem utilitzar cap id, ja que no ha estat entrada a la base de dades. El que fem llavors és deixar el camp id de l'URL amb el text "id".

No serà fins que es premi "Continuar" que no agafarà les dades i les afegirà a la base de dades, creant un id únic per aquella visita i, ara sí, incorporar-lo a la URL portant l'usuari al segon pas.

PRIMER PAS		SEGON PAS
/#/editar/12345/nom	s'està editant la visita amb id 12345	/#/editar/12345/panorames
#/editar/id/nom	s'està creant una nova visita	/#/editar/67890/panorames (67890 és un nou id creat en acabar el primer pas)

Poden veure el codi Javascript d'aquest pas a l'**ANNEX 4.3** i l'HTML a l'**ANNEX 3.21**.

6.5.2 PANORAMES (/#/editar/***id***/panorames)



Captura de pantalla del segon pas. Escanegi la imatge per veure'n el funcionament

En aquest pas (i tots els següents) sí que ja és imprescindible disposar d'un id.

El que es fa en el segon pas és penjar els panorames que es vulguin utilitzar. Les imatges han de ser panorames complerts (360ºx180º) i no poden pesar més de 5MB cadascuna.

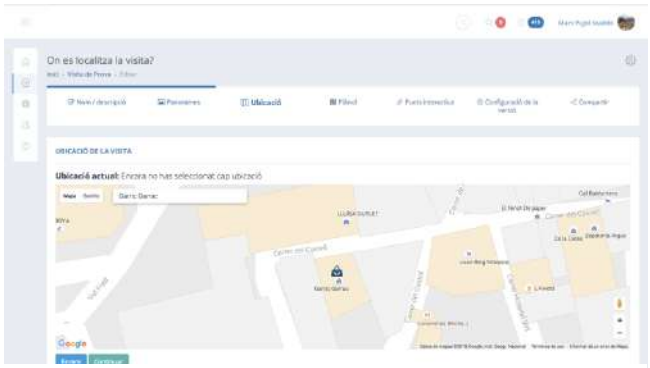
Hi ha dues columnes: en la de l'esquerra apareix

una zona habilitada per arrossegar-hi les imatges, a més d'aparèixer-hi aquelles que s'estan pujant, junt amb la seva barra de progrés; en la de la dreta, apareixen aquelles que ja han estat penjades, amb la possibilitat d'eliminar-les i d'afegir-hi un nom que doni títol a aquella escena (una escena és un panorama junt amb tots els seus punts interactius).

Quan cada un dels panorames es penja a *Firebase Storage*, es crea una nova entrada a la base de dades que es refereix a l'escena en la qual, entre moltes altres dades, hi ha el panorama que s'acaba de penjar i el nom que se li dóna.

Òbviament el codi Javascript per penjar els panorames és molt més complex. Poden veure'l a l'**ANNEX 4.8**, així com també poden veure el codi HTML a l'**ANNEX 3.22**.

6.5.3 UBICACIÓ (/#/editar/***id***/ubicacio)



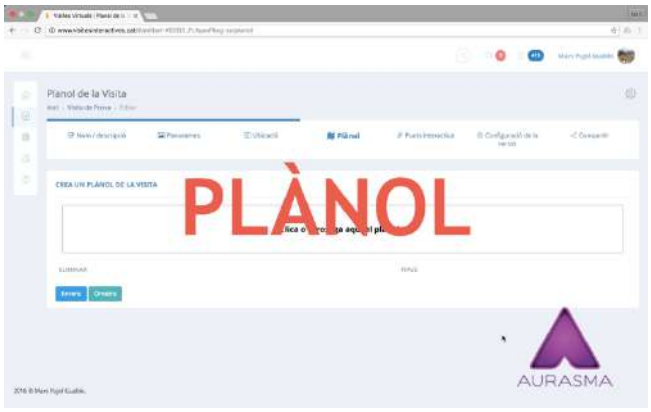
Captura de pantalla del tercer pas

En aquest pas l'usuari pot ubicar la visita en un mapa. He fet servir els mapes de Google Maps. Primer vaig pensar de definir la ubicació segons coordenades, però al final vaig trobar l'API de Google Places. Gràcies a aquesta API, no només puc buscar segons les adreces, sinó que també puc buscar directament els noms de les empreses.

Una vegada s'ha definit la ubicació i es prem "Continuar" es guarden les dades del mapa a la base de dades. Poden veure el codi HTML d'aquesta part en l'**ANNEX 3.23** i el codi Javascript en l'**ANNEX 4.3**.

6.5.4 PLÀNOL (/#/editar/***id***/planol)

En aquest pas s'ofereix la possibilitat d'afegir un plànol. Primer es puja el plànol, i després permet seleccionar-ne diferents zones i associar-les a l'escena que corresponguin. Així, després, els usuaris finals de la visita podran navegar entre les escenes prement les diferents zones del plànol.

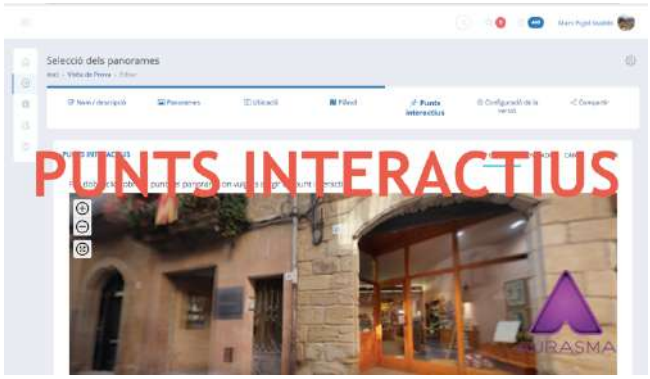


Captura de pantalla del quart pas. Escanegi la imatge per veure'n el funcionament

Per crear aquestes seccions dins una imatge, he utilitzar l'element map d'HTML. Per crear elements map interactivament (arrossegant sobre la imatge) he utilitzat el Framework jQuery "imgAreaSelect", de odyniec.net.

Poden veure el codi HTML d'aquesta part en l'**ANNEX 3.24** i el Javascript en l'**ANNEX 4.3**

6.5.5 PUNTS INTERACTIUS (/#/editar/***id***/punts)



Captura de pantalla del cinquè pas. Escanegi la imatge per veure'n el funcionament

Aquest pas és el més complex de tots, i el que dota d'interactivitat a les escenes de la visita.

Consisteix en afegir els punts interactius (*hotSpots*) a les escenes.

Per fer-ho, l'usuari pot escollir en tot moment, mitjançant un menú, l'escena a la qual vol afegir *hotSpots*. Disposa d'un previsualitzador d'aquella escena per veure-la i decidir on els vol afegir. Per fer-ho, ha de fer doble clic en el lloc exacte on el desitgi. Llavors s'obrirà un modal on podrà triar quin tipus de punt interactiu vol afegir. Quan premi el tipus que ha triat s'obrirà un segon modal demanant la informació que aquell tipus de punt interactiu necessiti. En prémer "Afegir" s'afegirà aquell punt a l'escena. Es tancaran tots els modals oberts i en el previsualitzador apareixerà automàticament el punt que acabem d'afegir, on podrà provar-lo i interactuar-hi. El punt interactiu també s'afegirà en un llistat a l'esquerra de la pantalla on podrà eliminar-lo o entrar a la seva configuració / editar.ne el contingut (només en algun tipus de *hotSpots*).

A nivell de codi, el que necessitem per afegir un hotSpot és obtenir tota la informació que requereixi el seu tipus i afegir-la a la base de dades. L'encarregat de fer que funcionin és la plataforma web (en aquesta pàgina apareix inserida com a previsualitzador).

Per tant, com que hi ha molts tipus de *hotSpots*, podem crear una funció que en rebi per paràmetres la informació i el tipus i ho pengi a la base de dades.

```
//funció que rep com a paràmetres el tipus de hotSpot i la seva informació i el penja a la base de dades
afegirHotspot = function (type, array) {
    //creem una referència a la base de dades de l'escena que estiguem editant en aquell moment
    var x = refVisita.child("scenes/" + $scope.panorama + "/hotSpots");
    //creem un objecte que representa el hotSpot (el que serà penjat a la base de dades)
    var object = {
        "type": type, //definim el tipus de hotSpot com el primer paràmetre que es passa a la funció
        "pitch": $scope.pitch, //aquesta variable ("pitch") s'ha guardat en el moment en què s'ha fet doble clic per afegir el hotSpot
        "yaw": $scope.yaw //aquesta variable ("yaw") s'ha guardat en el moment en què s'ha fet doble clic per afegir el hotSpot
    };
    //afegim els valors que hem passat com a segon paràmetre a l'objecte (inclouen tant el nom de la propietat que ha de tenir l'objecte com el seu valor)
    array.forEach(function (value) {
        //afegim la propietat a l'objecte i n'hi determinem els valors
        object[value[0][0]] = value[1][0];
    });
    x.push(object); //afegeix l'objecte que hem definit a la base de dades
    refrescarIframes(); //funció que refresca els iframes "previsualitzadors" de cada escena perquè aparegui el hotSpot que acabem d'afegir
    tancaModals(); // tanca els modals que hi havia oberts (el de tipus de hotSpot i el de la seva configuració)
    actualitzarData(); //actualitza la data d'última modificació de la visita a la base de dades
    comprovarConexions(); //comprova que tots els panorames estiguin connectats entre sí
};
```

Ara doncs, podem cridar a la funció afegirHotspot, passant com a primer paràmetre el tipus de hotSpot (ex: "vídeo") i com a segon un array amb les propietats que ha de tenir aquell hotSpot i els seus respectius valors (ex: [[["propietat1"], ["valor1"]], [["propietat2"], ["valor2"]]]).

Cada tipus de *hotSpot* té el seu modal per aconseguir la informació que necessita (modals dels quals poden veure el codi en l'**ANNEX 3.25** [HTML] i l'**ANNEX 4.3** [JavaScript]). En prémer "Afegir", es crida a una funció que prepara els paràmetres que cal passar a la funció afegirHotspot, i la crida.

Vegem els diferents tipus de *hotSpot* i les funcions que criden:

6.5.5.1 PANORAMA

Punt interactiu de tipus panorama

Aquest *hotSpot* serveix per connectar escenes permetent la navegació entre elles. Connecta l'escena actual amb una altra (escollida en el modal d'aquest tipus de *hotSpot*) que identifiquem mitjançant el seu id (*sceneId*).

```
$scope.hotspot_pano = function (panoid, text) {
     afegirHotspot("scene", [["text"], [text]], [["sceneId"], [panoid]]);
 };
```

6.5.5.2 TEXT, ENLLAÇ, TRUCAR, ENVIAR EMAIL

Punt interactiu de tipus text

En aquests quatre casos, el nom de la propietat que passem com a paràmetre és "text", i el seu valor és un element HTML que representa la informació que vol transmetre el *hotSpot*.

```
afegirHotspot("tipus", [["text"], ["codi HTML"]]);
```

A continuació podem veure els quatre tipus esmentats en el títol (les paraules subratllades són variables que s'obtenen en la funció que crida aquell tipus de *hotSpot*):

```
afegirHotspot("info", [["text"], [text]]); //TEXT

afegirHotspot("info", [["text"], ["<a target='_blank' href='" + arreglarUrl(adreça) + "'>" +
text_a_mostrar + "</a>"]]); //ENLLAÇ - la funció arreglarUrl permet que l'usuari introdueixi
l'enllaç sense http://, ja que l'hi afegeix automàticament si és necessari.

afegirHotspot("info", [["text"], ["<a target='_blank' href='tel:" + telefon + "'>" + telefon +
"</a>"]]); //TRUCAR

afegirHotspot("info", [["text"], ["<a target='_blank' href='mailto:" + email + "'>" + email +
"</a>"]]); //ENVIAR EMAIL
```

6.5.5.3 PÀGINA WEB INSERIDA, VÍDEO

Punt interactiu de tipus vídeo

Genera un espai on es veu la web o el vídeo establerts sense haver de sortir de la visita, de forma superposada. Passem l'adreça de la pàgina que s'ha de veure com a valor d'una propietat que anomenem "link"

```
// PÀGINA WEB INSERIDA
$scope.hotspot_web = function (prev_link) {
     var link = arreglarUrl(prev_link);
```

```
     afegirHotspot("iframe", [[["link"], [link]], [["text"], [""]]]);
                                };

// VÍDEO (Ha de ser de YouTube)
$scope.hotspot_video = function (yturl) {
     var ytcodi = "" + getUrlParameters("v", yturl); //aquesta funció obté la variable "v" de
de l'URL del vídeo. Conté l'id del vídeo
     var ytembed = "https://www.youtube.com/embed/" + ytcodi + "?enablejsapi=1";
     afegirHotspot("video", [[["link"], [ytembed]], [["text"], [""]]]);
 };
```

6.5.5.4 MÚSICA / SO, IMATGE

Punt interactiu de tipus música/so

Aquests tipus de *hotSpots* necessiten penjar arxius a *Firebase Storage*.

En el cas de la música, abans de penjar l'arxiu es poden configurar algunes opcions com la repetició de l'àudio en acabar-se o l'inici automàtic d'aquest.

Igual que en tots els arxius, abans de penjar-los es comprova que el format sigui l'adequat, calcula el progrés de la càrrega, i en acabar-se de penjar, afegeix l'arxiu al llistat d'arxius de la base de dades i executa el codi que afegeix el *hotSpot* també a la base de dades:

```
 // MÚSICA/SO
 //opcions = configuració elegida al DOM
 //url = url que proporciona Firebase Hosting quan s'acaba de penjar l'arxiu
 var text = "<audio controls " + opcions + "><source src='" + url + "' type='" + file.type +
"'>El teu navegador no suporta l'audio d'HTML5.</audio>"; //creem un tag HTML d'àudio amb les
dades corresponents
 afegirHotspot("music", [[["text"], [text]], [["panorama"], [panorama]], [["data"], [data]]]);

 // IMATGE
//url = url que proporciona Firebase Hosting quan s'acaba de penjar l'arxiu
 var text = "<img alt=”imatge Firebase Storage” src=" + url + ”>"; //creem un tag HTML
d'imatge amb les dades corresponents
 afegirHotspot("imatge", [[["text"], [text]]]);
```

Aquests dos tipus de *hotSpot*, a diferència de tota la resta, pengen un arxiu a *Firebase Storage*. Per tant, a l'hora d'eliminar aquests punts, cal eliminar abans l'arxiu.

```
//funció que elimina el punt interactiu que se li passi com a paràmetre (tots els punts
interactius se sotmeten a aquesta funció)
$scope.eliminarPunt = function (key) {
    refVisita.child("scenes/" + $scope.panorama + "/hotSpots/" + key).remove(); //l'elimina de
la base de dades
    actualitzarData(); //actualitza la data de "última edició"
```

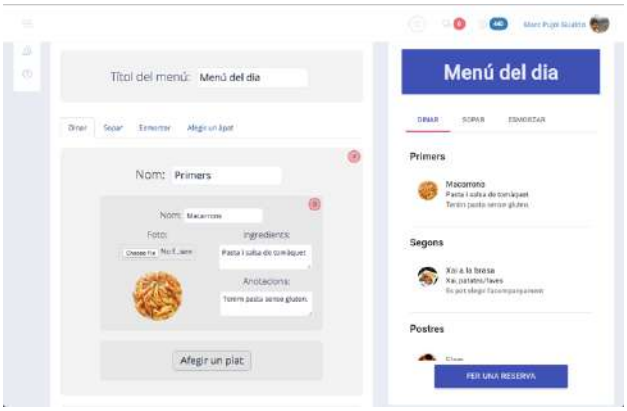
```
    comprovarConexions(); //comprova de nou que els panorames estiguin correctament connectats
(es podria haver eliminat un punt de tipus "panorama")
    refrescarIframes(); //refresca els iframes perquè es vegin els canvis reflexats en el
previsualitzador
};

//funció per eliminar un punt interactiu de tipus "música"
$scope.eliminarMusica = function (index, panorama, data) {
    //elimina l'àudio de Firebase Storage
    firebase.storage().ref().child("visites").child(id_visita).child("musica").child(panorama)
.child(data).delete().then(function () {
        console.log("S'ha eliminat la cançó / el so correctament");
    }, function (error) {
        console.log("Error liminant la cançó / el so", error);
    });
    //finalment crida la funció que elimina el punt interactiu de la base de dades
    $scope.eliminarPunt(index);
};

//funció per eliminar un punt interactiu de tipus "imatge"
$scope.eliminarFoto = function (index, panorama, data) {
    //elimina la imatge de Firebase Storage
    firebase.storage().ref().child("visites/" + id_visita + "/imatges/" +
data).delete().then(function () {
        console.log("S'ha eliminat la imatge correctament");
    }, function (error) {
        console.log("Error liminant la imatge", error);
    });
    //finalment crida la funció que elimina el punt interactiu de la base de dades
    $scope.eliminarPunt(index);
};
```

6.5.5.5 MENÚ RESTAURANT

Punt interactiu de tipus menú

Aquest cas és més complex. El modal que recull la informació consisteix en un "editor de menú" que he creat (que posteriorment pot ser utilitzat per editar-lo). Permet afegir àpats, apartats dins els àpats i plats dins els apartats.

També permet definir el preu de cada àpat.



Captura de pantalla de l'editor del menú

Cada plat compta amb el seu nom, ingredients, anotacions i una foto (pot ser capturada en aquell moment des d'un mòbil, per exemple).

El que faig és guardar tota la informació que necessito a la base de dades. Després aquestes dades seran passades a una pàgina HTML externa que he dissenyat expressament, i finalment aquesta pàgina serà inserida a la visita com a "Pàgina web inserida" (el tipus de *hotSpot* anterior).

Per a més informació del menú i del seu editor vegeu l'**ANNEX 3.31**.

```
$scope.hotspot_menu = function () {
     var link = "http://www.visitesinteractives.cat/sub/menu/index.html?visitaid=" +
id_visita; //la pàgina que conté el menú, passant-li com a paràmetre l'id de la visita perquè
pugui anar a buscar les dades a la base de dades.
     afegirHotspot("menu", [["link"], [link]], [["text"], [""]]);
 };
```

6.5.5.6 RESERVAR TAULA RESTAURANT, RESERVAR HABITACIÓ HOTEL

Aquests tipus de *hotSpots* estan destinats a restaurants i hotels, i també són diferents de la resta. De fet, m'atreviria a dir que són els més complexos.

A l'hora de configurar-los, tan sols es demana el nom de la taula o habitació a la que s'està afegint aquest *hotSpot*, però el que veu l'usuari final de la visita és un formulari amb les dades necessàries per reservar una taula a un restaurant o una habitació en un hotel (nom, persones, data, hora, correu electrònic, comentaris...). Una vegada l'usuari omple les dades i prem el botó de reservar, automàticament es guarden aquestes dades, s'envia un correu al propietari de la visita informant que té una nova reserva i un altre correu al client dient que s'ha rebut la petició i que aviat serà acceptada o rebutjada pel propietari.

Punt interactiu de tipus "reservar habitació". Escanegi per veure com funciona els sistema de reserves i correus electrònics

El correu que ha rebut el propietari inclou un enllaç a l'apartat de l'aplicació web que maneja les reserves.

En aquest apartat, el propietari pot veure les dades de la reserva, i acceptar-la o rebutjar-la. En ambdues possibilitats s'ofereix un text predeterminat i personalitzat per respondre el client. Aquest text pot ser editat, i una vegada es prem "Enviar", s'envia un últim correu al client confirmant o rebutjant la seva reserva a la taula o habitació que havia sol·licitat amb el text editat pel propietari. A més, la reserva també queda marcada com a reservada o rebutjada en l'aplicació web.

Tots aquests correus posseeixen dissenys compatibles amb la majoria de dispositius i dades personalitzades. Per veure tot aquest procés, si us plau escanegin la imatge adjuntada.

Òbviament, tota aquesta explicació teòrica l'he hagut de transformar en codi. Aquest codi és tan extens que he cregut oportú separar-lo del treball, així que el podeu trobar en els **ANNEXOS 3.14, 3.15, 3.31, 3.32, 3.33, 4.9, 4.15, 5.1, 5.2, 5.3, 5.4 i 5.5.**

De la mateixa manera que ho fèiem amb el menú, aquest *hotSpot* serà afegit com a "Pàgina web inserida", apuntant a una pàgina HTML externa que he desenvolupat a la qual li passem l'id de la visita perquè en pugui obtenir les dades necessàries.

```
$scope.hotspot_reservartaula = function (taula) {
     var link = "http://www.visitesinteractives.cat/sub/reservar/index.html?visitaid=" +
id_visita + "&taula=" + taula;
     afegirHotspot("reservar", [["link"], [link]], [["text"], [""]]);
 };

$scope.hotspot_reservarhabitacio = function (habitacio) {
     var link = "http://www.visitesinteractives.cat/sub/reservarhotel/index.html?visitaid=" +
id_visita + "&habitacio=" + habitacio;
     afegirHotspot("reservartaula", [["link"], [link]], [["text"], [""]]);
 };
```

6.5.5.7 PERSONALITZAT (HTML)

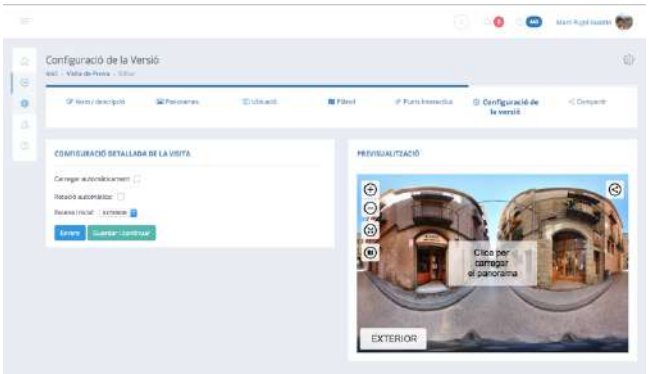
Per últim, disposem d'un tipus de *hotSpot* que permet afegir funcionalitats diferents de les que ofereixen la resta d'opcions. Per utilitzar-lo s'ha de tenir un mínim de coneixement en HTML, ja que és el llenguatge que s'utilitza per configurar aquest tipus de *hotSpot*.

Punt interactiu de tipus personalitzat Així com en la resta de tipus es demanava les dades i després es creava un element HTML a partir de les dades obtingudes (perquè l'usuari no hagi d'escriure codi d'HTML), ara l'usuari pot escriure el codi HTML sencer.

El fet de que l'usuari pugui desenvolupar per complet la funcionalitat d'aquest punt interactiu el converteix en el tipus de punt més versàtil i interessant. Una de les moltes possibilitats d'aquest punt seria, per exemple, poder crear un punt interactiu per cada objecte d'una botiga, des del qual es pogués comprar-lo en la seva respectiva botiga on-line.

```
$scope.hotspot_html = function (codi, amplada, alcada) {
    afegirHotspot("personalitzat", [[["text"], ["<div style='width:" + amplada + "px;height:"
+ alcada + "px;clear:both;'>" + codi + "</div>"]]]);
};
```

6.5.6 CONFIGURACIÓ DE LA VERSIÓ (/#/editar/***id***/versio)



Captura de pantalla del sisè pas.

Una vegada acabem amb els punts, l'usuari passa al següent pas: la configuració de la versió.

Una visita pot ser visualitzada de diferents maneres. Pot carregar-se automàticament o esperar a que premin "Carregar", pot començar a rodar automàticament o no, fer-ho a una velocitat determinada, deixant passar uns segons abans de començar... fins i tot pot mostrar primer una escena o una altra.

Totes aquestes possibles configuracions són les que es tracten en aquest pas. Podem configurar la visita com més ens agradi i prémer "Continuar"; això crearà una versió d'aquesta visita i ens portarà al següent pas.

Però una vegada ja n'hem creat una, podem tornar a aquest pas i crear totes les que vulguem. Potser en volem una que es carregui automàticament per enviar per correu i una altra que no ho faci per publicar-la a una web sense augmentar-ne el temps de càrrega. Aquest és el pas per fer-ho.

El funcionament és el següent:

Quan l'usuari prem "Continuar" s'agafa la configuració que hi ha en aquell moment, i es guarda en una nova entrada a la base de dades de la visita, generant així un nou id propi d'aquesta versió.

Llavors s'utilitza aquest id per passar-lo com a paràmetre a la URL final de la visita www.visitesinteractives.cat/visita?id=*id_visita*&v=***id_versio***, (la plataforma web l'utilitzarà per buscar a la base de dades les característiques d'aquesta versió i mostrar-la d'acord amb elles).

Aquest ja seria un URL vàlid per compartir amb el món la nostra visita, però és un pèl massa llarg per ser compartit. L'ideal seria aconseguir una URL més curta, d'entre 10 – 15 caràcters enlloc dels 70 que ocupa aquesta. Això ho podem fer gràcies a *Google Shortener* (goo.gl), un servei que ens porporciona una URL del tipus goo.gl/id_curt que redirecciona a la URL que li diguem. Un gran avantatge d'aquest servei és que podrem obtenir estadístiques d'aquestes URLs. Així doncs, una vegada generem aquesta URL curta, la guardem també a la base de dades.

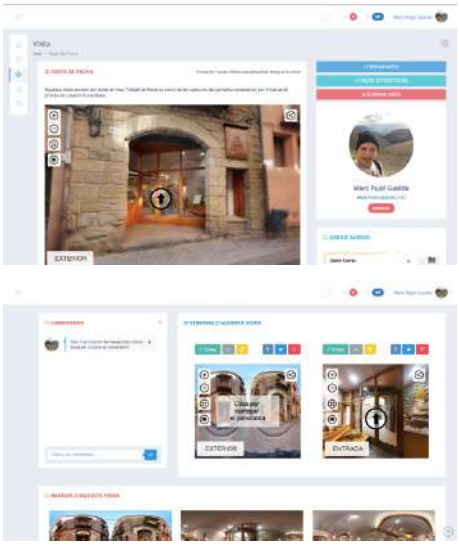
6.5.7 COMPARTIR (/#/editar/***id***/compartir)

Captura de pantalla del setè pas.

En aquest últim pas, el que se'ns mostra són totes les versions que haguem creat de la visita. Per cada una, se'ns ofereix múltiples botons per compartir-la. Un botó que ens mostra l'enllaç, un altre que ens mostra el codi HTML per inserir la visita en qualsevol web, un altre ens ofereix un codi QR per ser escanejat i per últim tres botons més per compartir-la a les xarxes socials principals (Facebook, Twitter, Google Plus).

6.6 PÀGINA DE LA VISITA (/#/visita/***id***)

Cada visita disposa de la seva pàgina única. Aquesta pàgina es troba a /#/visita/**id**, on, òbviament, **id** n'és el seu id. En aquesta pàgina, a diferència de la de crear/editar (/#/editar/id/nom), hi poden accedir tots els usuaris registrats, no només el creador.



Captures de pantalla de la pàgina d'una visita

Aquesta pàgina consta de diferents "widgets". El principal, i el més gran, és la visita pròpiament dita. Consta del seu títol, la seva descripció, la data de creació i última modificació, i la visita (la plataforma web inserida)

A la dreta d'aquest "widget", veiem uns botons de colors. Si no som el creador de la visita, tan sols en veurem un que ens envia a les estadístiques d'aquella visita (/#/estadistiques/id).

Si en som els creadors, apareixeran com a mínim dos botons més, el d'editar la visita (/#/editar/id/nom) i el d'eliminar-la (apareix un modal de confirmació).

Si les característiques de la visita ho requereixen, però, poden aparèixer altres botons com el d'editar el menú (/#/visita/id/menu) o administrar les reserves (/#/visita/id/reserves/taula o /#/visita/id /reserves/habitació).

A sota dels botons apareix un altre "widget" amb informació del creador de la visita, amb la possibilitat d'anar al seu perfil (/#/perfil/id-usuari) o començar-hi un xat (/#/xat/id-xat).

Per sota d'aquest, n'hi ha un altre que compta amb un mapa de la ubicació que s'hagi establert de la visita.

A continuació ens tornem a trobar dos "widgets" de costat. El de l'esquerra serveix per afegir i llegir comentaris sobre la visita en qüestió, i el de la dreta ens mostra totes les versions que té la visita. Cada una de les versions compta amb les mateixes opcions per ser compartida que les que disposa el creador a l'últim pas de la creació de visites (/#/editar/id/compartir).

Per últim, hi ha un "widget" que conté els panorames originals de la visita. Es poden obrir en una pestanya nova i també es poden descarregar mitjançant el botó habilitat per fer-ho.

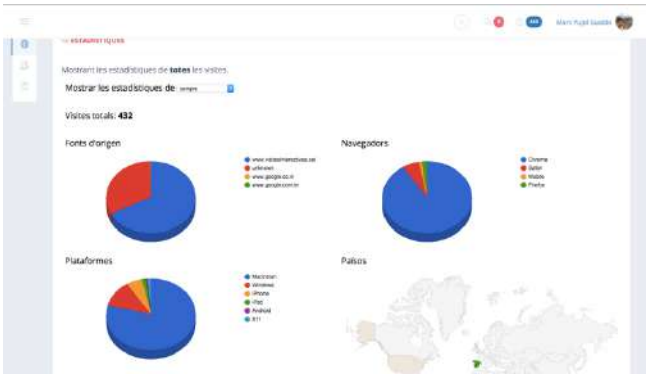
Totes questes funcionalitats són possibles gràcies al codi Javascript encarregat d'aquesta pàgina, que podeu trobar totalment comentat a l'**ANNEX 4.12**. Per veure'n el codi HTML, vegin l'**ANNEX 3.17**.

També cal saber que a /#/visites hi trobarem un llistat de totes les visites i a /#/mapa les hi trobarem situades

en un gran mapa amb la possibilitat de veure'n més informació de cada una en fer-hi clic a sobre.

El codi que maneja aquestes dues pàgines es troba en els **ANNEXOS 3.12 i 3.18** (HTML) i els **ANNEXOS 4.7 i 4.13** (Javascript)

6.7 ESTADÍSTIQUES DE LES VISITES (/#/estadistiques/*id*)



Captura de pantalla de la pàgina d'estadístiques

Les estadístiques de les visualitzacions a les visites són una eina molt potent. Permeten descobrir des de quin dispositiu s'han vist més vegades, des de quin navegador, com han arribat els usuaris a la visita... Totes aquestes dades poden servir tant per millorar l'aplicació web (fer-la més amigable per a mòbils, prioritzar compatibilitat amb navegadors...) com per saber quina és la millor manera de compartir-les. (l'origen de procedència ens mostrarà on les troba la gent. Facebook, Twitter, blogs...)

Per aconseguir les estadístiques, faig servir l'API de *Google Shortener* (l'eina que també m'ajuda a escurçar els URLs). Jo li dóno l'URL escurçat del qual vull aconseguir estadístiques i n'obtinc un objecte Javascript com a resposta que conté totes les dades.

He utilitzat l'API de *Google Charts* per mostrar aquestes dades en gràfiques de diferents tipus.

La pàgina de les estadístiques és /#/estadístiques, encara que la major part d'HTML i Javascript que requereixen està en un arxiu extern. De fet, l'arxiu HTML que es mostra a /#/estadístiques conté un iframe a l'arxiu extern, on realment es troben les estadístiques. Veure **ANNEX 3.3**.

Així doncs, cal diferenciar el desenvolupament d'aquest arxiu extern i el de "dins" de l'aplicació web.

El que es fa des de l'aplicació web és obtenir les URLs que s'han d'analitzar i passar-les a l'arxiu extern.

Hi ha dues possibilitats, que l'usuari vulgui les estadístiques de totes les visites (/#/estadístiques) o que vulgui les estadístiques detallades d'una sola visita (/#/estadístiques/id).

En el primer cas, hem d'obtenir totes les URLs. És a dir, l'URL de cada versió de cada visita.

En el segon pas, només hem d'obtenir les URLs de les versions de la visita especificada.

Per veure tot aquest procés, vegi's l'**ANNEX 4.5**.

Una vegada tenim les URLs, les passem a l'arxiu extern com a paràmetre en la URL d'aquest: www.visitesinteractives.cat/estadistiques.html?url=['http://goo.gl/qkTsTB','https://goo.gl/oGsBKz'].

Ara és el torn de l'arxiu extern.

Primer de tot agafa l'array d'URLs que se li ha passat com a paràmetre.

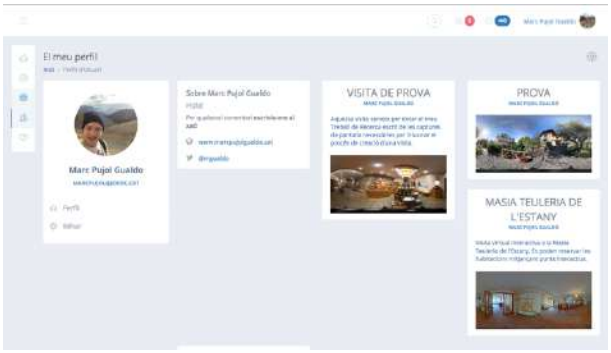
Amb cada un d'aquests URLs fa una petició per rebre'n les estadístiques. Un cop les té, les tracta per poder crear els gràfics. Aquest procés es repeteix per cada una de les URLs, i les dades es van sumant.

També es pot restringir el període de temps a analitzar (sempre, aquest mes, avui, les últimes dues hores...)

6.8 PERFIL D'USUARI (/#/perfil/*id-usuari*)

Els usuaris tenen una forta importància en l'aplicació web. Hi ha un xat entre ells, poden fer comentaris, apareixen a la pàgina de la visita... I també tenen un perfil.

Per veure tots els usuaris, podem anar a /#/usuaris. Des d'aquest apartat veurem la informació més important de cada usuari, la seva última connexió i un accés directe per començar un xat amb ell.



Captura de pantalla del perfil de l'usuari actiu

Per veure un perfil en concret, s'ha d'anar a /#/perfil/**idusuari**, on **idusuari** és l'id únic de cada usuari (com en les visites). Per veure el perfil d'un mateix, l'id ha de ser el nostre o simplement no ha d'haver-n'hi (/#/perfil). En aquest cas veuràs la teva informació tal i com la veuries de qualsevol altre usuari. L'única diferència és que no hi ha el botó per començar un xat, i n'hi ha un altre per editar el perfil.

Si el cliquem accedirem a /#/perfil/editar. Aquest apartat consta de 4 subapartats: informació personal (canvia el nom, afegeix una descripció o enllaços a xarxes socials), canviar la foto de perfil (canvia la foto penjant la nova a *Firebase Storage* i actualitzant-ne la URL a la base de dades), canviar el correu i canviar la contrasenya (aquests dos últims funcionen mitjançant funcions que em proporciona Firebase Auth).

També hi ha un botó per eliminar el compte. En prémer-lo, es demana una confirmació amb un modal i es

deixa clar que amb aquesta acció no se n'eliminaran les visites.

Tornant al perfil d'un usuari, si l'id no és el nostre, es mostrarà la informació de l'usuari en qüestió.

La informació inclou el nom, el correu electrònic, l'última connexió, un petit text d'informació sobre l'usuari, enllaços a les seves xarxes socials i webs d'interès (algunes d'aquestes dades han d'haver estat afegides per l'usuari en editar el seu perfil). També es mostra un llistat de les visites que ha creat, i com he dit abans, un botó per iniciar un xat amb ell.

Tot aquest contingut i aquestes funcionalitats aquí esmentades, es troben explicades detalladament en els **ANNEXOS 3.16, 3.28, 3.29 i 3.30** (HTML), i els **ANNEXOS 4.10 i 4.11** (Javascript).

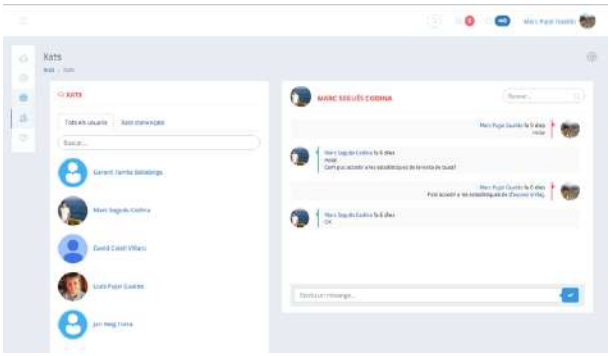
6.9 XAT (/#/xat/***id-xat***)

Com ja he mencionat en algunes ocasions al llarg del treball, existeix un xat entre els usuaris.

La meva primera idea fou la de crear un xat d'ajuda perquè qualsevol usuari pogués contactar fàcilment amb mi. Després vaig pensar que seria força interessant permetre que tots els usuaris poguessin contactar entre ells fàcilment, ja sigui per felicitar algú per una visita que ha creat, per demanar com s'ho ha fet per afegir algun tipus de funcionalitat o qualsevol altra cosa. Així que vaig crear un xat per a tots els usuaris.

Per accedir al xat, s'entra a /#/xat (o /#/xat/*idxat* si volem anar directament a un xat en concret).

Sempre hi ha dues columnes. A la de l'esquerra hi ha una llista on hi podrem veure i buscar tots els usuaris registrats o, si així ho indiquem, només els usuaris amb qui ja tinguem un xat començat. A la columna de la dreta, hi trobem la zona del xat pròpiament dit.



Captura de pantalla d'un xat

D'entrada, en /#/xat, la columna de la dreta apareix en blanc, mostrant un missatge que indica com començar un xat amb un usuari. Quan en comencem (o re-emprenguem) algun, aquest s'hi mostrarà.

Si ens fixem en la URL, veurem que en haver-hi un xat actiu aquesta és /#/xat/**idxat**. Com es pot intuir, cada xat té el seu id (**idxat**). Segueix la mateixa lògica que /#/estadístiques/*id*, /#/perfil/*id* o /#/visita/*id*.

Passem ara a un altre aspecte del xat, les notificacions.

Per una banda, si ens fixem en el *header* (capçalera), podem veure una icona que representa un xat amb un número en forma de subíndex. Aquesta icona ens indica el nombre de missatges que tenim pendents de llegir al xat. Passant el cursor per sobre podem veure aquests missatges, i fent clic a sobre anirem al xat en qüestió i la notificació desapareixerà.

Per altra banda, en el moment en què s'envia un missatge, si fa més de 10 minuts que el xat en qüestió estava inactiu, es notificarà de la seva presència per correu electrònic. En aquest correu es mostra la imatge de l'usuari que ha enviat el missatge, el cos del missatge i un botó per obrir el xat al navegador.

Tot el codi (degudament comentat) HTML del xat es pot veure a l'**ANNEX 3.19**, i el Javascript a l'**ANNEX 4.14**. El codi de les notificacions al *header* es troba en l'**ANNEX 3.4** (HTML) i l'**ANNEX 4.1** (Javascript). I el codi PHP encarregat d'enviar el correu de notificació es troba en l'**ANNEX 5.6**.

6.10 ALTRES FUNCIONALITATS

En els punts anteriors he explicat el meu treball seguint com a estructura les diferents parts/pàgines de l'aplicació web. Si bé ja he comentat les principals, n'hi ha algunes altres de secundàries, però no per això menys importants.

Una altra funcionalitat d'aquesta *webapp* és el seu buscador, disponible a /#/buscar. Allà hi trobem una entrada de text des d'on podem cercar i filtrar entre els usuaris i visites de l'aplicació. També podem accedir directament a la nostra cerca escrivint la paraula com a segon apartat en l'adreça a /#/buscar/el-text-a-buscar o des de la barra de cerca que sempre està present en el *header*.

El codi d'aquest apartat es troba a l'**ANNEX 3.9** (HTML) i **ANNEX 4.2** (Javascript).

En el header, a més d'haver-hi aquest cercador i les notificacions dels missatges, encara hi ha un parell més de funcionalitats. Una d'aquestes és la icona que es troba al costat de les notificacions del xat. Es tracta d'un "historial" d'accions en l'aplicació web. Passant-hi per sobre veiem una llista de tot el que hem anat fent, ordenat per ordre cronològic descendent

i amb la possibilitat d'eliminar cada un dels ítems o tots a la vegada.

La segona funcionalitat és la icona de la part superior dreta, que conté la nostra imatge de perfil. Passant per sobre d'aquesta zona s'obra un petit desplegable amb dues opcions relacionades amb el perfil: "Editar el Perfil" (redirigeix a /#/perfil/editar) o "Tancar la Sessió" (tanca la sessió i redirigeix a /login.html).

El codi HTML del *header* està en l'**ANNEX 3.4**, i el Javascript en l'**ANNEX 4.1**.

Una altra pàgina de la qual no hem parlat és la d'ajuda (/#/ajuda).

El que trobem en aquesta pàgina són visites guiades per entendre el funcionament de molts aspectes de la web. Són unes visites que ressalten les zones en les quals s'ha de parar atenció, i expliquen pas a pas processos com el d'editar el teu perfil, crear una visita o començar un xat.

El codi d'aquesta pàgina es troba en l'**ANNEX 3.8** (HTML), i en l'**ANNEX 4.1** (Javascript)

7. DESENVOLUPAMENT DE LA PLATAFORMA WEB

La plataforma web és l'entorn web on l'usuari final veu una visita.

És un visualitzador capaç de mostrar la informació i els punts interactius de cada visita.

Al llarg del punt 4, on he tractat el desenvolupament de l'aplicació web, ha aparegut la plataforma web en algunes ocasions. Recordem que durant el procés de creació d'una visita la veiem de tant en tant inserida i actuant com a previsualitzador. Al final d'aquest procés també creàvem una URL que hi apuntava i també l'hem vist inserida en la pàgina d'una visita.

A grans trets, podem dir que la plataforma web és l'encarregada de mostrar totes les dades i arxius que s'han anat creant i confeccionant en l'aplicació web.

Totes aquestes dades (entre elles, URLs als arxius), es troben a la base de dades, concretament a /visita/*idvisita*. Per tant, la plataforma web ha de conèixer l'id de la visita que vol mostrar per anar-ne a buscar abans les dades. Aquesta dada l'aconsegueix interpretant l'URL.

L'usuari que entri en una visita haurà rebut un enllaç similar a http://goo.gl/qkTsTB, però aquest l'haurà

redirigit a http://www.visitesinteractives.cat/visita?**id**=-KR-sRe5Y0unQIJO0XV4&**v**=1473203028309.

Fixant-nos en a questa adreça, veiem que rep dos paràmetres: **id** (l'id de la visita) i **v** (l'id de la versió que volem mostrar).

Ara que ja coneixem l'id de la visita, és qüestió d'anar-ne a buscar les dades a la base de dades.

Per fer-ho, fem una petició AJAX a https://visites-interactives.firebaseio.com/visites/*idvisita*.json. Això ens retornarà un text de tipus JSON amb totes les dades.

A més de les dades referents al contingut, també hem d'anar a buscar les dades referents a la versió, per saber de quina manera s'ha de mostrar la visita. Aquestes dades es troben dins del text JSON que hem obtingut. Per tant, només ens cal buscar-les utilitzant l'id de la versió que obtenim de l'URL.

Una vegada tenim les dades, la major part del funcionament és a càrrec de Pannellum.

A continuació explicaré breument com funciona i els aspectes que jo li he afegit o canviat.

Pel que fa els hotSpots, el que fa Pannellum és crear un div per a cada hotSpot. Aquest div té una icona, i en passar per sobre mostra un span (que fins llavors estava ocult). Aquest span mostra el codi HTML que el hotSpot tingui en la seva propietat "text".

Per tant, es mostra el contingut que l'usuari ha configurat en cada hotSpot. (Encara que l'usuari configurés els hotSpots d'una forma interactiva i visual, en el fons s'estava creant un fragment de codi HTML).

El fet que cada hotSpot tingui determinat un tipus (imatge, text, menú...), em permet, mitjançant CSS (*Cascading Style Sheets*), dotar de la icona i del disseny necessari a cada un d'ells (per exemple, un vídeo requereix un espai més gran que un text).

Hi ha alguns *hotSpot* especials. Són els que contenen altres propietats a part de "pitch" (posició en X), "yaw" (posició en Y), "type" (tipus) i "text" (codi HTML del hotSpot).

En són exemples els que contenen "sceneId" (s'utilitza per saber a quin panorama hem d'anar en un hotSpot de tipus panorama) o els que contenen "link" (automàticament es crea un iframe i s'atribueix el valor de "link" a l'atribut "src" de l'iframe).

Pel que fa als hotSpots de vídeo, he incorporat l'API de Youtube per pausar el vídeo quan el cursor surt d'aquest i tornar-lo a renaudar en passar per sobre (evitant que es reprodueixi el vídeo quan el *hotSpot* no

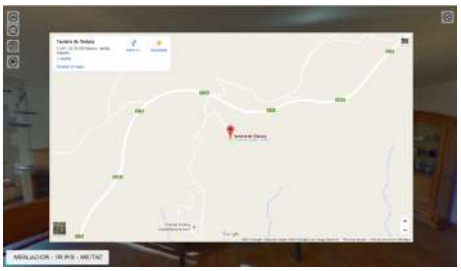
és visible).

Deixem aquí els hotSpots i passem a altres funcionalitats que he afegit o canviat de la plataforma.

En primer lloc, he creat una nova manera de navegar entre les escenes d'una visita.

A més de fer-ho a traves dels hotSpots de tipus "panorama", ara també és possible canviar d'escena mitjançant un plànol. Aquest plànol ha de ser una imatge representativa de l'espai que engloba la visita, per exemple, una fotografia aèria. Aquest plànol és mostrat en fer clic sobre la caixa d'informació (part inferior esquerra), en forma de modal (superposat a la visita).

El plànol conté zones que poden ser premudes. En ser-ho, l'usuari serà enviat a l'escena que s'hagi associat a la zona premuda.



Mapa de la visita en la plataforma web

Una altra funcionalitat que he agregat és el mapa de la ubicació de la visita, que es mostra, a l'igual que el plànol, d'una forma superposada. Per mostrar-lo, cal prémer el botó que conté la icona d'un mapa.

Pel que fa als botons, també he afegit botons per compartir la visita (obtenir l'enllaç, compartir a les xarxes socials i mostrar un codi QR).

També he hagut de fer uns canvis en el funcionament del visualitzador per poder utilitzar-lo com a previsualitzador mentre es crea una visita. Sense aquests canvis, no podríem veure la plataforma web fins que no acabéssim amb el procés de creació.

Per últim, he canviat el disseny del visualitzador modificant el codi CSS i afegint-ne de propi.

Tot el que he mencionat aquí, i **molt** més, es troba degudament comentat i explicat d'una forma més tècnica en el codi de la plataforma web. Vegin-lo en els següents annexos: **ANNEX 3.34** (HTML), **ANNEX 3.35** (CSS) i **ANNEX 4.16** (JavaScript).

8. PROBLEMES

El principal problema, o més ben dit, dificultat, ha estat realitzar el redactat d'aquest treball.

En tractar-se d'un treball de recerca, no puc conèixer amb exactitud els lectors als quals arribarà. Evidentment, no explicaria de la mateixa manera el meu treball a una persona que tingui coneixements de programació que a una persona que no els tingui.

Per tant, m'he trobat amb un dilema: redactar de manera que tothom entengui el que estic escrivint, renunciant a la precisió...? O parlar de paraules, conceptes i procediments que (almenys a mi) m'ha portat anys a aprendre, reduint el nombre de lectors possibles...?

Després de donar-hi moltes voltes, vaig pensar que redactaria de manera que qualsevol persona es pogués fer una idea bastant clara del funcionament del projecte, utilitzant fragments de codi ben comentats i de fàcil comprensió. Per entrar en detall i trobar les explicacions més tècniques i precises, seria necessari accedir als Annexos, on hi hauria tota la programació degudament comentada. A mesura que vaig anar redactant el treball em vaig adonar que aquesta era la millor forma, ja que fent-ho així (col·locant les explicacions més tècniques als annexos) solucionava també un altre problema, la limitació d'espai.

Passant als problemes del projecte pràctic, un dels inconvenients (encara que alhora avantatge) que vaig tenir, va ser l'actualització de Firebase, el servei que utilitzo com a base de dades.

El fet que s'actualitzés em va permetre aprofitar les noves funcionalitats que oferia, sobretot la de "*Firebase Storage*", el servei per emmagatzemar arxius al núvol. El preu a pagar, però, va ser haver de reescriure bastantes parts del meu codi, amb tot el que això comporta[1].

Tan bon punt vaig actualitzar la primera línia de codi, em van aparèixer moltíssims errors, i aquests no van desaparèixer fins que no vaig acabar amb la migració.

Un altre problema va ser la comunicació entre pàgines web diferents.

Hi ha moments en què des de la plataforma web (quan és utilitzada com a previsualitzador) s'ha de cridar una funció que està definida a l'aplicació web. Això només es pot fer si el domini és el mateix en ambdues pàgines. Encara que el domini fos el mateix, vaig tenir problemes si accedia a la pàgina amb o sense "www." al davant. Després de molta recerca vaig trobar un mètode[2] perquè sempre es carregués la pàgina amb "www." al davant, eliminant així el problema.

Un altre problema fou utilitzar com a panorames de la visita els panorames penjats a Firebase Hosting. No podia mostrar les imatges, ja que l'URL de la meva pàgina no era acceptada per Firebase. Finalment, vaig trobar la solució[3]. Vaig afegir la meva URL d'entre les permeses al projecte de Firebase.

Aquests són alguns dels problemes que he tingut.

Cal dir, però, que l'apartat "problemes" en un treball de programació pot ser una mica paranoic, ja que la programació consisteix precisament en resoldre'n.

Programant, contínuament et trobes amb dificultats, i amb cada una d'elles pots estar batallant-hi durant molt de temps.

Però, en el fons, si del que es tracta és d'aprendre i millorar no hauríem de veure els problemes com quelcom negatiu, ans el contrari, ja que són aquests els que ens fan progressar.

9. POSSIBLES MILLORES

Desenvolupar una aplicació web pot ser un procés infinit. Sempre es poden afegir noves funcionalitats o trobar maneres de funcionar diferents de l'actual.

Algunes de les funcionalitats i millores que tinc en ment a dia d'avui són:

- Compatibilitat amb ulleres de realitat augmentada.
- Mantenir una relació entre el volum de la música i l'angle de visió en la visita.
- Afegir més d'un plànol per suportar visites multi-planta.
- Permetre que més d'un usuari tinguin drets d'editor per una sola visita.
- Millorar la identificació de panorames incomunicats.

M'hauria agradat desenvolupar aquestes millores i moltes més, però per qüestió de temps m'ha resultat impossible.

[1] [https://github.com/firebase/angularfire/blob/master/docs/migration/1XX-to-2XX.md]

[2] [http://stackoverflow.com/questions/4159088/how-to-modify-htaccess-file-to-always-redirect-to-www]

[3] [http://stackoverflow.com/questions/37760695/firebase-storage-and-access-control-allow-origin]

10. CONCLUSIONS

La conclusió a la qual arribo és molt bona.

Em responc positivament al meu plantejament inicial; he constatat que és possible desenvolupar un sistema de creació de visites virtuals interactives, desenvolupant moltes funcionalitats totalment noves, i fer-ho sol, en pocs mesos, i amb un cost quasi bé nul (uns 17€/l'any per el domini).

A part de la resposta al meu plantejament inicial, he obtingut moltes altres conclusions.

Una que considero molt important és que m'he adonat que la programació web és molt poderosa. Et permet crear quasi bé tot allò que et puguis imaginar i oferir-ho a tothom mitjançant qualsevol dispositiu amb connexió a internet. Sembla una afirmació molt idíl·lica, fins i tot pot semblar exagerada. Però, realment, en aquests mesos he anat veient, amb gran satisfacció, com esdevenien realitat les meves idees. Sorprenentment, m'he adonat que les limitacions que he tingut han estat més causades pel límit de temps que no pas per la impossibilitat de dur-les a terme.

A nivell personal, puc afirmar que aquest treball m'ha ajudat moltíssim en el meu aprenentatge.

He vist com els meus coneixements augmentaven progressivament en els darrers mesos. De fet, en el tram final del treball, he canviat bastants fragments de codi que ja havia creat en els inicis del treball per uns altres de més curts i eficients que no era capaç de desenvolupar quan els vaig escriure.

També he descobert que la programació requereix un estat de recerca permanent, ja que està en una contínua i ràpida evolució. No hi ha més sortida que l'aprenentatge constant i la recerca per trobar les solucions més eficaces, adequades i modernes. Per aquest motiu, crec que és més important l'habilitat de buscar, ordenar i comprendre la informació que no pas la de memoritzar-la, ja que resulta impossible i de seguida queda desfasada.

Més enllà d'aquestes conclusions escrites, la conclusió real de tot el meu treball és l'aplicació i la plataforma web. Per tant, us convido a entrar-hi i a utilitzar totes les seves funcionalitats.

www.visitesinteractives.cat

11. BIBLIOGRAFIA WEB

En moltes d'aquestes pàgines (documentacions i fòrums) hi he accedit en diverses ocasions per consultar diferents aspectes / preguntes. Per raons d'espai, no cito cada una de les pàgines consultades per separat.

Documentacions:

https://docs.angularjs.org - Documentació d'AngularJS

http://api.jquery.com/ - Documentació de jQuery

https://firebase.google.com/docs/ - Documentació de Firebase

https://github.com/firebase/angularfire - Documentació d'AngularFire

https://pannellum.org/documentation/overview/ - Documentació de Pannellum

Fòrums:

http://stackoverflow.com/ - StackOverflow - Fòrum de programació

https://www.acamica.com/comunidad - Fòrum de la comunitat d'Acamica

http://www.codingforums.com/ - CodingForums - Fòrum de programació

http://www.dreamincode.net/forums/ - </dream.in.code> - Fòrum de programació

Altres:

http://keenthemes.com/preview/metronic/ - Metronic (KeenThemes) - Tema CSS

https://pannellum.org/ - Pannellum - Visualitzador de Panorames

http://www.w3schools.com/ - w3schools - Explicacions d'HTML, CSS i Javascript

https://www.acamica.com/cursos - Acamica - Cursos de programació i tecnologia

https://github.com/yaru22/angular-timeago - TimeAgo - Framework per AngularJS

http://introjs.com/ - IntroJs - Servei per crear visites guiades en un lloc web.

https://dummyimage.com/ - Dummyimage - Generador d'imatges a partir d'un text

https://www.swhosting.com/ca/ - Swhosting – Domini i hosting

https://developer.mozilla.org/ca/docs/Web/API/notification - MDN - API Notificacions d'escriptori

https://github.com/kemayo/maphilight - Map Highlight (David Lynch) – Framework per jQuery