

# Proyecto MIXEET

**Grupo**  
Roxanne

## INFORME DE SEGUIMIENTO

Hito: 1  
Fecha entrega: 23-12-2015  
Versión: 1.2

Componentes:

- **Roxanne López van Dooren**

## 1. Introducción

A continuación se va a detallar el seguimiento de las iteraciones definidas en Microsoft Project para este primer hito.

Antes de comenzar a profundizar en ello, cabe destacar que se ha completado gran parte de las funcionalidades propuestas del ABP para las siguientes asignaturas de este primer cuatrimestre en función de las horas asignadas al equipo:

- Proyectos Multimedia
- Postproducción Digital
- Sistemas de Difusión Multimedia

De las cuales a rasgos generales se han tenido que hacer las siguientes tareas:

PM	Actualizar documentos	100 Hrs
	Actualizar iteraciones Microsoft Project	
	Aplicación del modelo EVA	
	Preparar soporte presentación	
	Practicar presentación	
	Actualizar Cloud	
PD	Crear y diseñar logotipo	90 Hrs
	Definir paleta colores a usar	
	Crear y diseñar poster	
	Definir primeros bocetos logo de empresa	
SDM	Diseñar Lean Canvas	100 Hrs
	Crear esqueleto del proyecto	
	Definir dependencias del proyecto	
	Configurar servidor y base de datos	
	Conectar aplicación a Youtube API v3	
	Desarrollar FRONTEND del proyecto	
	Desarrollar BACKEND del proyecto	
	Maquetar página web	

## 2. Seguimiento iteraciones

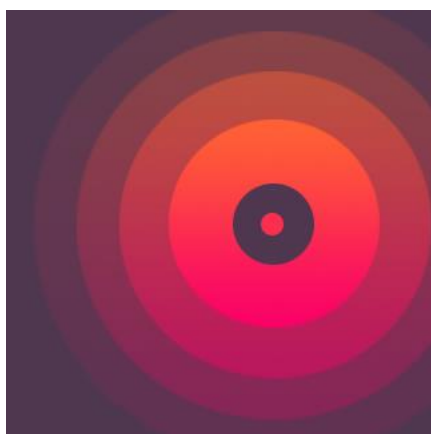
En el caso del proyecto de Mixeet, cuyo equipo es de un componente, se ha estimado la creación de un solo entregable ya que éste incluye las funcionalidades básicas pensadas para el proyecto y carece de sentido mostrarlas por separado.

De esta forma, se presenta a continuación el desarrollo paso a paso de las funcionalidades que conforman el entregable del proyecto con todo el esqueleto completamente formado, con el acceso (registro e inicio de sesión) del usuario y la obtención de sus datos sacados de la API de Google+ y Youtube v3.

Funcionalidad/Entregable	Estado	Observaciones
Acceso a página web desde la Landing	✓	-
Recolección de datos del usuario desde su cuenta de Google+	✓	Utiliza el API de Google+ y Youtube v3
Obtención de listas de reproducción del usuario	✗	Problemas con el API de Youtube
Geolocalización del usuario	✓	Utiliza el API de Google Maps
Edición de datos del usuario	✓	-
Navegación por las distintas secciones de la aplicación	✓	Creado con AngularJS, Express y NodeJS
Cerrar sesión en la aplicación	✓	-
Entregable (Proyecto con todos los apartados creados, acceso a la parte privada y obtención de datos del usuario mediante la API elegida)	✓	Por consola hacer un: <b>npm update</b> <b>bower update</b> Y por último para iniciar el servidor: <b>nodemon server</b> El puerto usado es el 3000 <b>http://localhost:3000</b>

En primer lugar, además de las funcionalidades descritas anteriormente para la aplicación se han realizado otras tareas aparte como por ejemplo:

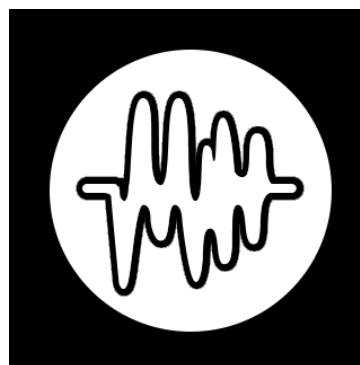
- Diseños de logotipos (PD): realizados con Photoshop CS6  
**Horas Previstas: 15 horas – Horas Reales: 12.5 horas**



*Prototipos v1, v1.2*

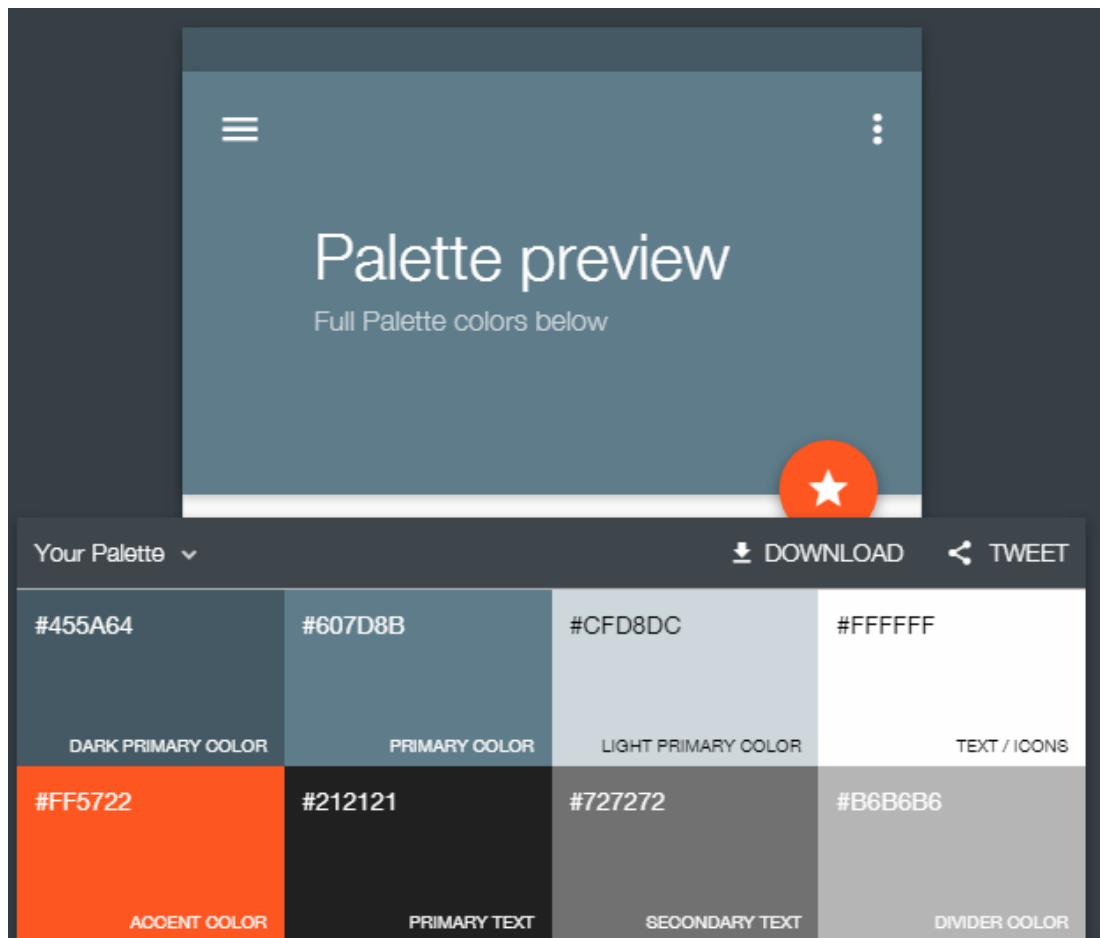


*Prototipo v2 (final)*

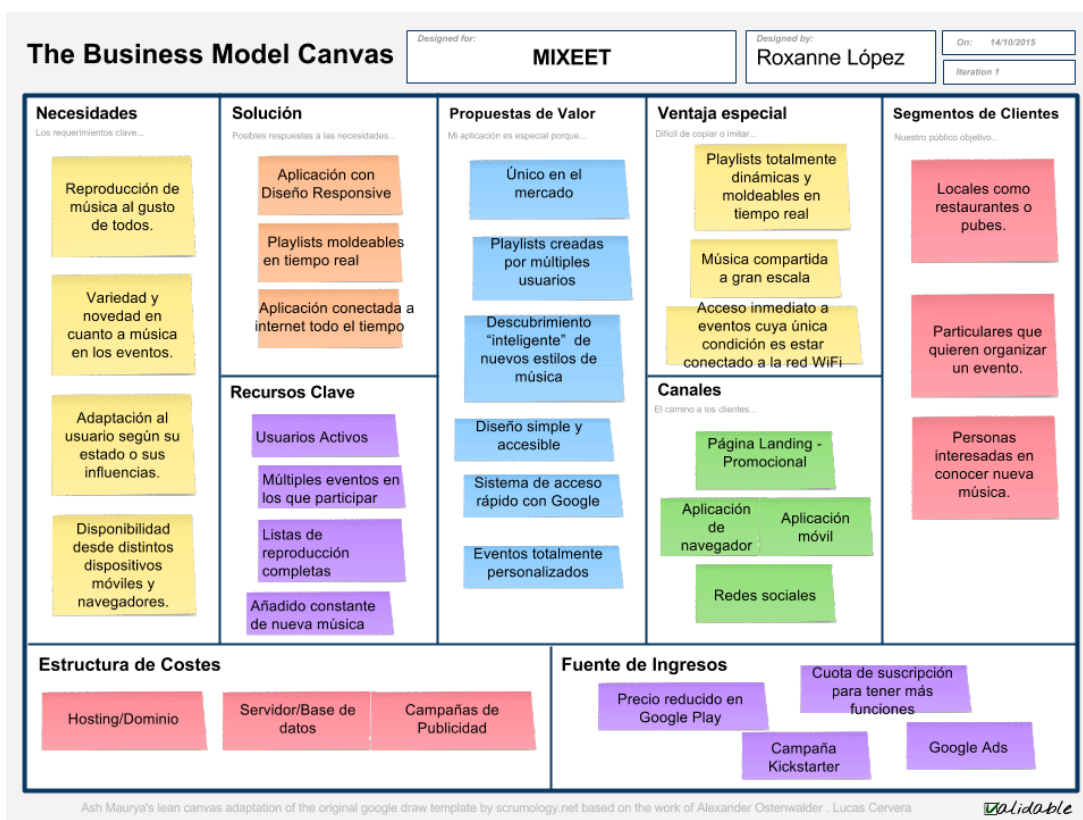


*Prototipo v2 (b&w)*

- Selección de paleta de colores: utilizando el material palette.  
**Horas Previstas: 5 horas – Horas Reales: 7 horas**



- Definición del Lean Canvas: usado el modelo que proporciona Google Drive.  
**Horas Previstas: 15 horas – Horas Reales: 14.5 horas**



- Diseño de la tabla comparativa entre Youtube vs Spotify.  
**Horas Previstas: 20 horas – Horas Reales: 18 horas**

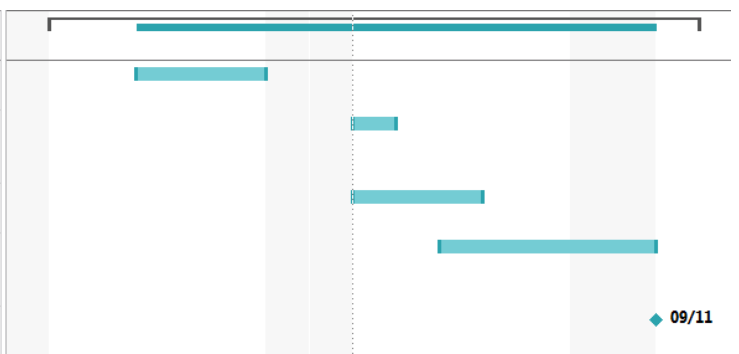
Tabla Comparativa	
Youtube	Spotify
Playlists inventadas por Youtube	Imposible comercializar app
Videos que podrían no ser música	Sólo podrían usarlo usuarios Premium de Spotify
Videos sin definir (sin tags o títulos correctos)	Música únicamente
Todo el mundo tiene una cuenta de google	Canciones bien organizadas
Posibilidad de descarga	Imposibilidad de descarga
Acceso a música amateur	
Illegal separar el Audio del Video	

El desarrollo de las funcionalidades pensadas para el hito 1 se ha realizado siguiendo los siguientes pasos:

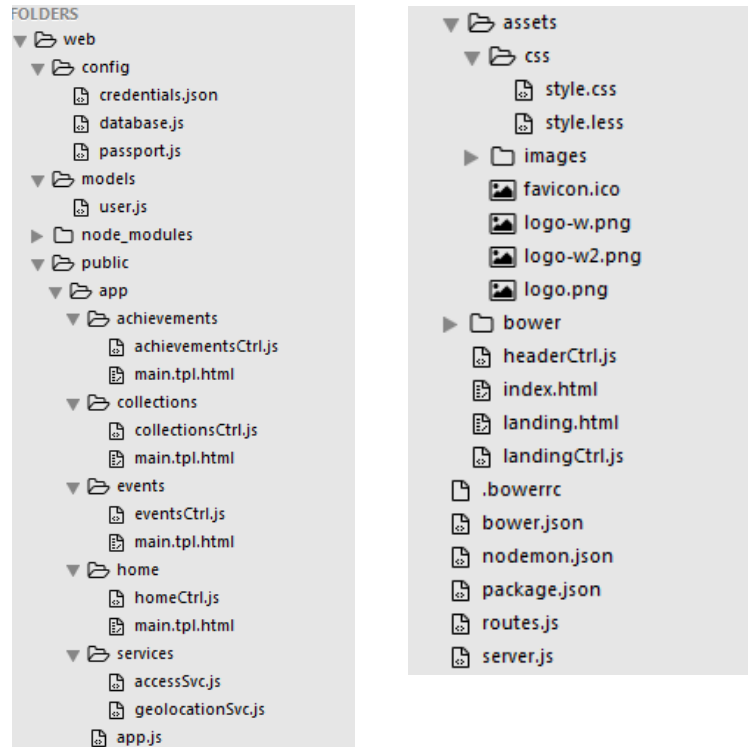
- 1) En nuestra primera iteración hemos previsto realizar un total de **45 horas** (si tenemos en cuenta que en un día trabajamos un total de 5 horas).

Las tareas de la Iteración 1 se han completado en un total de **40 horas**.

Iteración 1 - Proyecto inicial	11 días	lun 26/10/15	lun 09/11/15	
Diseñar logo de la aplicación	3 días	mié 28/10/15	vie 30/10/15	
Crear proyecto inicial con dependencias	1 día	lun 02/11/15	lun 02/11/15	
Definir Lean Canvas (SDM)	3 días	lun 02/11/15	mié 04/11/15	
Estudiar y analizar APIs de Youtube y Spotify	4 días	mié 04/11/15	dom 08/11/15	
Proyecto inicial y elección de API	0 días	lun 09/11/15	lun 09/11/15	



Como ya hemos descrito en el informe previo, las tecnologías usadas para la aplicación web son **AngularJS**, **NodeJS** y **Express**. Al seguir un modelo vista controlador, el esqueleto del proyecto tiene la siguiente estructura:



Podemos ver que el directorio principal está compuesto por las carpetas:

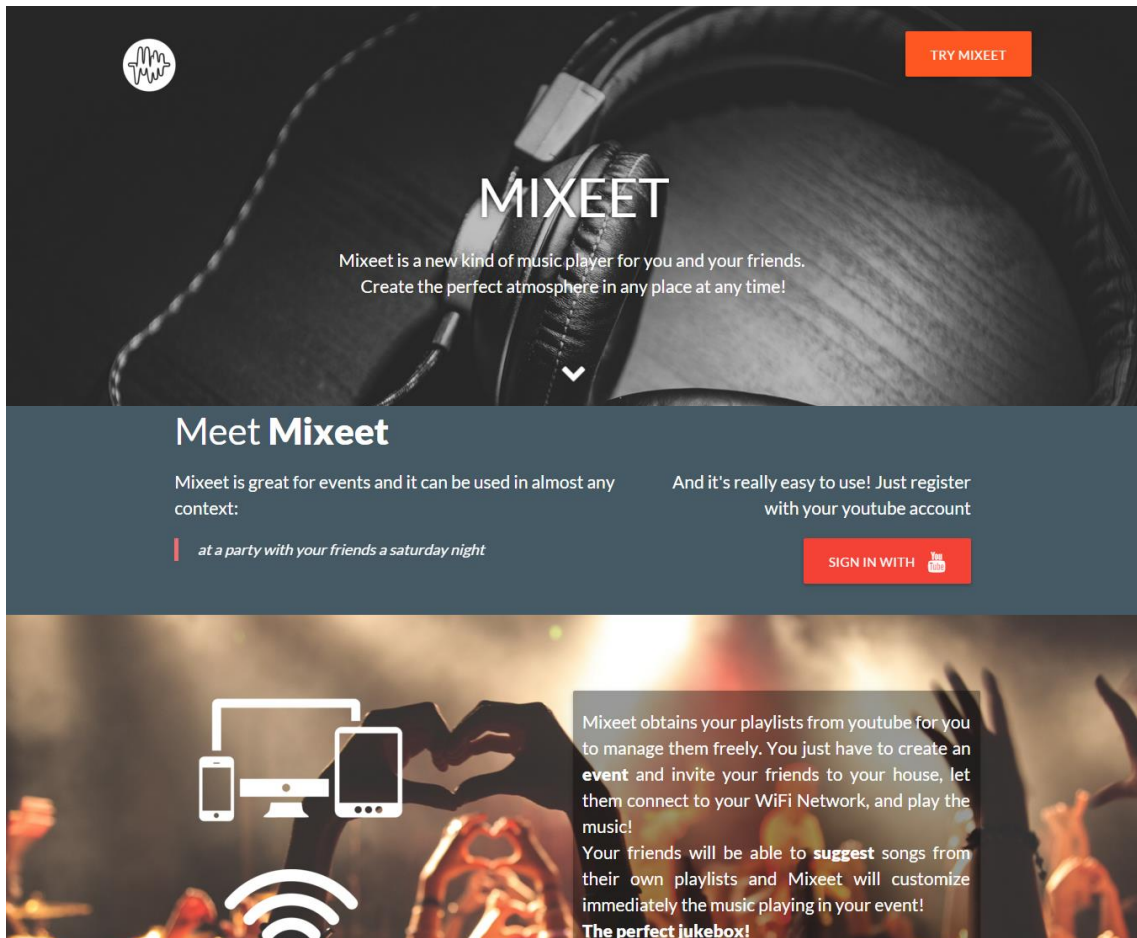
- **Config**: en la que se guardan todos los datos de configuración tanto de la Base de Datos como de la consola de Developer de Google.
- **Models**: en la que tenemos nuestros modelos de Mongoose y MongoDB para el proyecto.
- **Node\_models**: donde guardamos todas las dependencias instaladas con NODE en la parte del servidor.
- **Public**: que contiene el archivo 'routes' que indica las distintas rutas que tiene nuestra aplicación, el archivo 'server' donde inicializamos nuestro servidor e indicamos el puerto que estamos usando, nuestros archivos de inicialización de dependencias y a su vez está compuesto por las carpetas:
  - **Bower**: donde guardamos todas las dependencias instaladas con Bower en la parte del cliente.
  - **App**: en la que tenemos todas las partes que conforman nuestra aplicación ordenadas con sus vistas y controladores.

- **Assets:** que guarda las imágenes y el estilo utilizado en nuestra aplicación.

2) En la iteración número dos se ha estimado un total de **70 horas** a trabajar. El tiempo real trabajado ha sido de **75 horas**.

Iteración 2 - Inicio de Sesión con API	11 días	lun 09/11/15	lun 23/11/15	
Análisis de Riesgos	1 día	lun 09/11/15	lun 09/11/15	
Diseñar primeros bocetos poster (PD)	6 días	lun 09/11/15	dom 15/11/15	
Hacer pruebas con API elegida	6 días	mié 11/11/15	mié 18/11/15	
Desarrollar Landing o Página de Inicio	9 días	mié 11/11/15	lun 23/11/15	
Implementar inicio de sesión con API	6 días	lun 16/11/15	lun 23/11/15	
Proyecto con acceso	0 días	lun 23/11/15	lun 23/11/15	

A la hora de desarrollar la página de inicio:




**MIXEET**

Mixeet is a new kind of music player for you and your friends.  
Create the perfect atmosphere in any place at any time!

**Meet Mixeet**

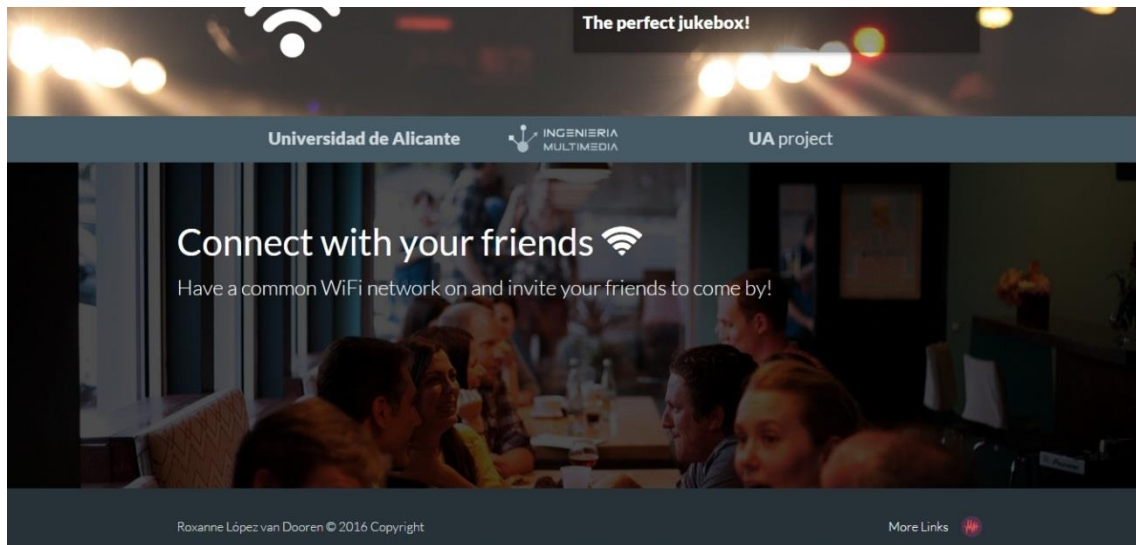
Mixeet is great for events and it can be used in almost any context:  
*at a party with your friends a saturday night*

And it's really easy to use! Just register with your youtube account

**SIGN IN WITH** 

Mixeet obtains your playlists from youtube for you to manage them freely. You just have to create an **event** and invite your friends to your house, let them connect to your WiFi Network, and play the music!  
Your friends will be able to **suggest** songs from their own playlists and Mixeet will customize immediately the music playing in your event!  
**The perfect jukebox!**





Hemos usado la librería de apoyo de estilo Materialize.css y Font Awesome. La página está compuesta por elementos muy sencillos como divs e imágenes de fondo con efecto 'Parallax' para dar una sensación de movimiento y dinamismo. Aparte de eso, se han incluido algunas animaciones programadas con JQuery y CSS como por ejemplo en el slider de la parte inferior de la página.

Podemos notar dos botones que dirigen al mismo sitio para que el usuario pulse y se registre en Mixeet. Esta es la parte complicada de la aplicación ya que tenemos que conectar con la API de Google + y Youtube para registrar e iniciar a nuestros usuarios.

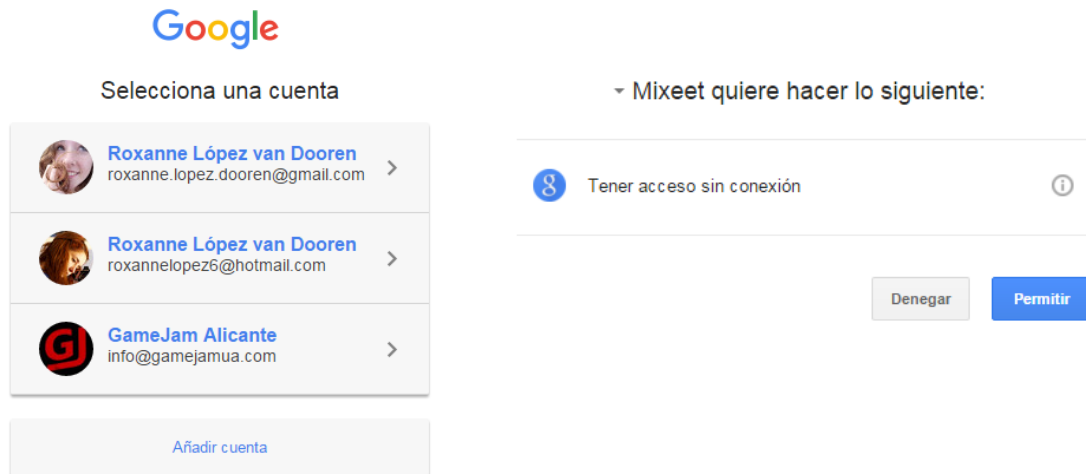
Una vez creadas nuestras credenciales en la consola de desarrollador de Google, hemos tenido que indagar qué librerías en concreto había que usar para recibir la información que nos interesaba de la cuenta del usuario en cuestión (el nombre, el correo electrónico y la imagen de avatar).

```
users.get('/signin', function(req, res){
  if(!req.query.code){
    var scopes = ['https://www.googleapis.com/auth/youtube.readonly',
                  'https://www.googleapis.com/auth/plus.me',
                  'https://www.googleapis.com/auth/plus.login',
                  'https://www.googleapis.com/auth/userinfo.email'
                ];

    var url = oauth2Client.generateAuthUrl({
      access_type: 'online',
      scope: scopes
    });

    res.redirect(url);
  }
});
```

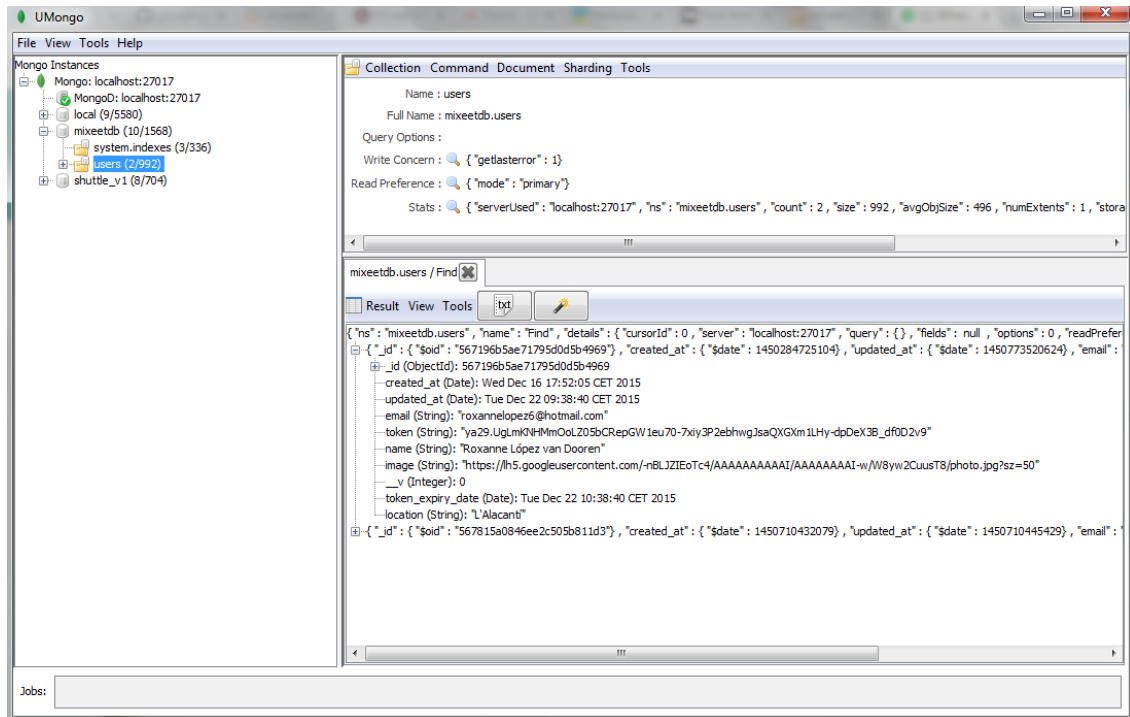
En la imagen superior podemos ver qué 'scopes' se han utilizado para la obtención de estos datos mencionados. A través de esta API, cuando el usuario pulsa sobre el botón correspondiente, le dirigimos a la siguiente página:



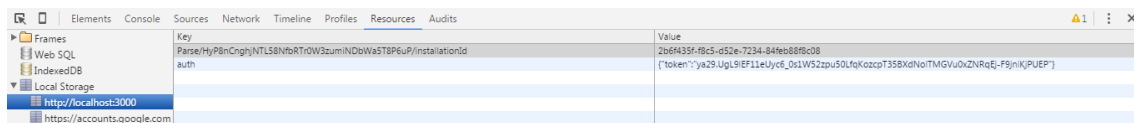
Cuando el usuario elige su perfil o cuenta, el servidor de Mixeet recibe un token por parte de Google, que es el que necesitará para poder identificar al usuario posteriormente en su base de datos. Independientemente de si el usuario existe o no en Mixeet, no va a tener que hacer nada, ya que Mixeet sólo necesita saber si debe insertarlo en la base de datos e iniciarlo como siguiente paso. En el caso de ya existir, se salta el paso de la inserción y lo dirige inmediatamente:

```
newuser.save(function(err,res){
  //TANTO COMO SI EXISTE COMO SI NO, LOGUEO AL USUARIO
  if(err){
    user.findOne({email:email})
    .exec(function(err2, user){
      if(err2) callback(true, token, {error:'server error'});
      else{
        if(!user) callback(true, token, {error:'server error - no user'});
        else{
          user.token = token;
          user.token_expiry_date = expirydate;
          user.save(function(err3, res2){
            if(err3) callback(true, token, {error:'server error'});
            else callback(true, token);
          });
        }
      }
    })
  }
  else callback(false, token);
});
```

Para dar más credibilidad al asunto, podemos ver desde la interfaz gráfica UMongo que en nuestra base de datos el usuario Roxanne López van Dooren existe, y con todos sus datos obtenidos de su cuenta de Google:



Para poder hacer peticiones que requieren de la autenticación del usuario (como editar su perfil o hallar su localización) hemos añadido su token de autorización en el Local Storage, además de esta forma evitamos también que un usuario no autenticado pueda entrar a la parte privada de nuestra aplicación.

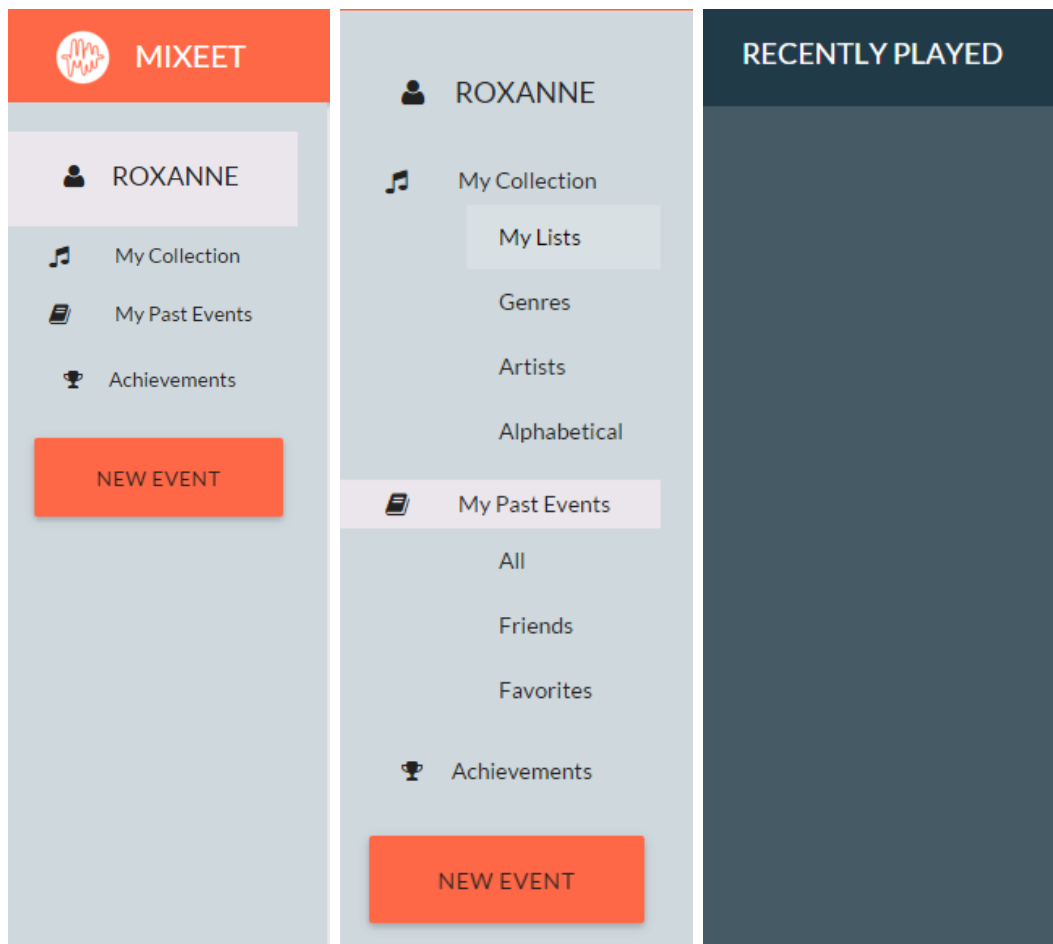


- 3) La iteración 3 se ha centrado en desarrollar la parte interna o privada de Mixeet, además de crear una página de landing más accesible y usable desde dispositivos móviles.

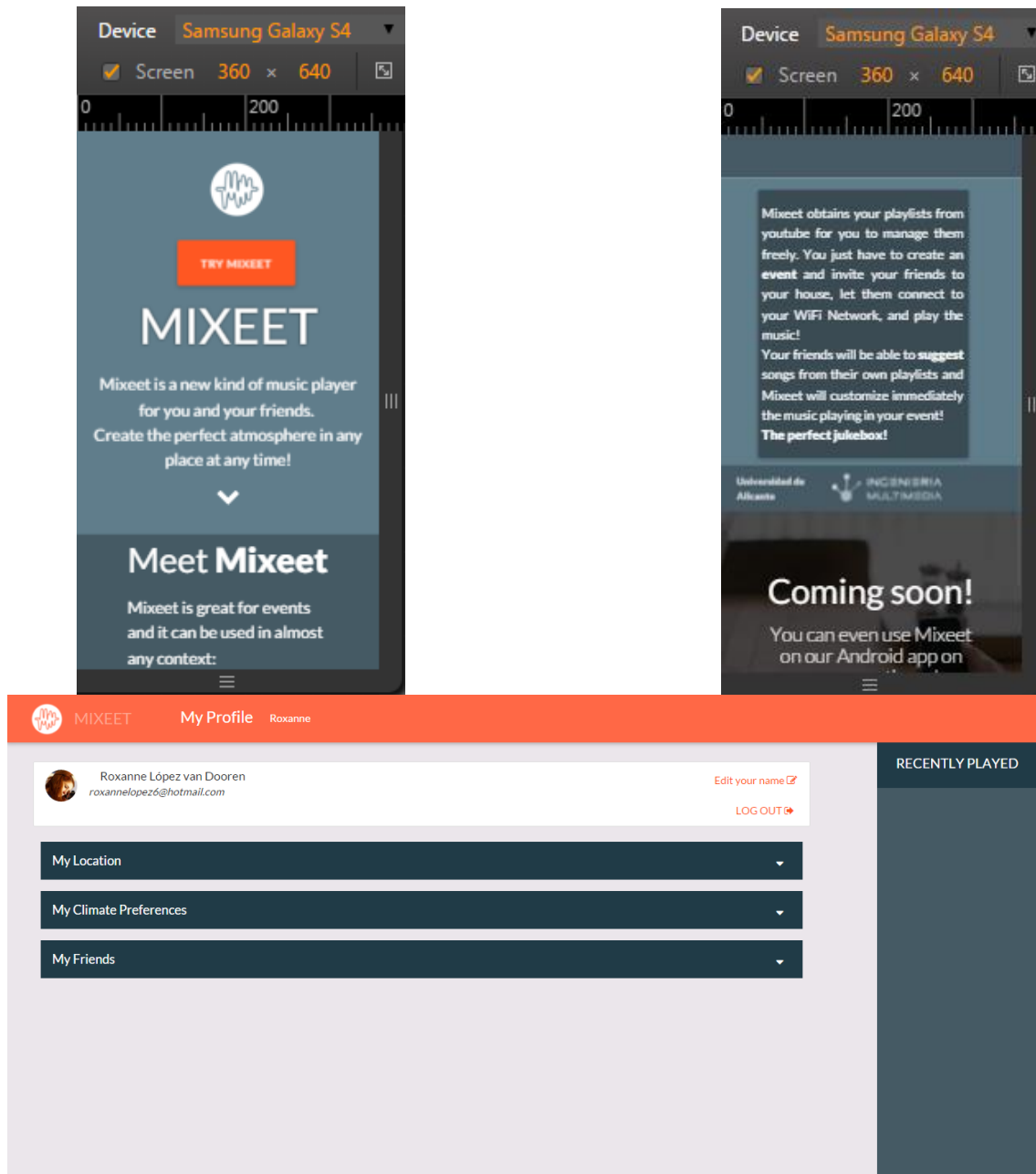
Las horas previstas para este itinerario han sido de **50 horas**, y aunque no se haya podido obtener las listas de reproducción del usuario mediante la API de Youtube, las horas realizadas han sumado un total de **60 horas**.

Iteración 3 - Página Principal y Menús en común	11 días	lun 23/11/15	lun 07/12/15	
Rellenar documentos EVA	1 día	lun 30/11/15	lun 30/11/15	
Diseñar Menús comunes al	7 días	mar 24/11/15	mié 02/12/15	
Diseñar modelo responsive de la Landing	3 días	jue 03/12/15	dom 06/12/15	
Hacer pruebas con obtención de información de youtube	6 días	lun 30/11/15	dom 06/12/15	
Proyecto parte privada inicial	0 días	lun 07/12/15	lun 07/12/15	

Los menús que comparten las distintas vistas se han realizado utilizando algunos atributos de Materialize.css pero casi todo ha sido realizado desde cero con CSS.



Tanto la página principal o landing como los menús de la parte privada tienen un estilo responsive para que se vea bien en todos los dispositivos. Esto se ha hecho utilizando los atributos de @media en CSS.



- 4) En la última iteración nos hemos centrado en desarrollar el perfil de usuario de la parte privada, añadiendo funcionalidades como la de poder editar su información o añadir su localización.

Las horas previstas de esta parte han sido 55, y al final las horas reales han sido 40.


Iteración 4 - Datos del Usuario + Navegación Principal	11 días	lun 07/12/15	lun 21/12/15
Desarrollar página de perfil	5 días	mar 08/12/15	dom 13/12/15
Recuperar datos del usuario y guardar cambios	3 días	lun 14/12/15	mié 16/12/15
Diseñar finales toques al poster	2 días	vie 18/12/15	dom 20/12/15
Desarrollar geolocalización con Google Maps API	4 días	mar 15/12/15	vie 18/12/15
Proyecto con datos del usuario visibles	0 días	lun 21/12/15	lun 21/12/15

Para obtener los datos del usuario hemos creado una ruta personalizada (API REST) de forma que nuestro servidor cuando detecta que se ha entrado en esa URL, saca los datos de la base de datos de ese usuario autenticado.


```
users.get('/me', auth, function(req, res){
  res.json({email:req.user.email, name:req.user.name, image:req.user.image, location:req.user.location});
});
```


Cuando el usuario edita esos datos tendremos que hacer lo mismo pero con una petición de tipo POST para posteriormente guardar esa información:


```
users.post('/me/modify', auth, function(req, res){
  req.user.name = req.body.name;
  req.user.save(function(err, res1){
    if(err) res.status(500).json({msg:'server error'});
    else res.status(200).json({msg:'ok'});
  });
});
```





Roxanne López van Dooren  
 roxannelopez6@hotmail.com

Save 


LOG OUT 

My Location 

My Climate Preferences 

My Friends 

Para el tema de la localización, hemos usado como ya mencionado anteriormente la API de Google Maps y las funciones ya incluidas en HTML5 de geolocalización. Una vez recibidas las coordenadas de la posición el usuario, buscamos con la función del Geocoder el nombre de la ciudad que corresponde a través del archivo JSON que devuelve Google ('results[1].address\_components[1].long\_name').



Roxanne López van Dooren  
roxannelopez6@hotmail.com

Edit your name ✎  
LOG OUT ➔

My Location

You are currently in L'Alacantí
MODIFY LOCATION 📍

My Climate Preferences

My Friends