Jorge Segovia Tormo

Grupo

Javier Molpeceres Gómez   
Jorge Segovia Tormo

SPOTIFY

Práctica 2 Ingeniería Web

SPOTIFY

La propuesta del profesor fue:

Publica:

registro, buscar canciones/detalle de canción, oir previews

Privada:

login, buscar canciones/detalle de canción, escucharlas, listas de reproducción

BO: CRUD canciones, estilos, artistas

Dejando lo de tipos e intereses de usuarios, tipos de listas y la publicidad

Después de hablar con el profesor y explicarle nuestros inconvenientes sobre su propuesta por el hecho de que Spotify no tuviera una parte pública usable y que los estilos pertenecían a las canciones pasaba a estar así:

Publica:

Login, registro

Privada:

buscar canciones/detalle de canción, escucharlas(simulado), listas de reproducción, artistas, detalle de artista

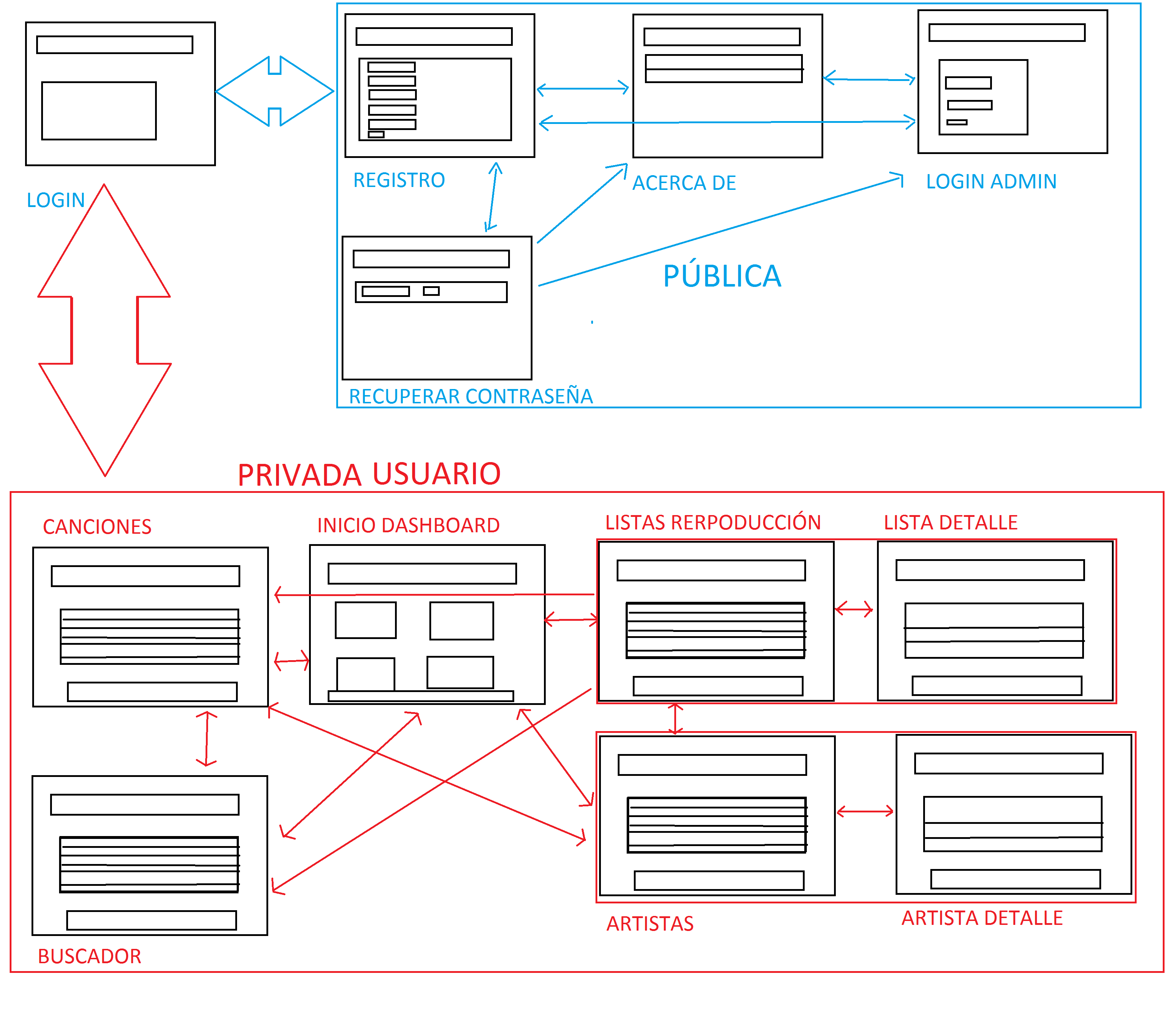
BO: CRUD canciones, artistas, usuarios, albumes

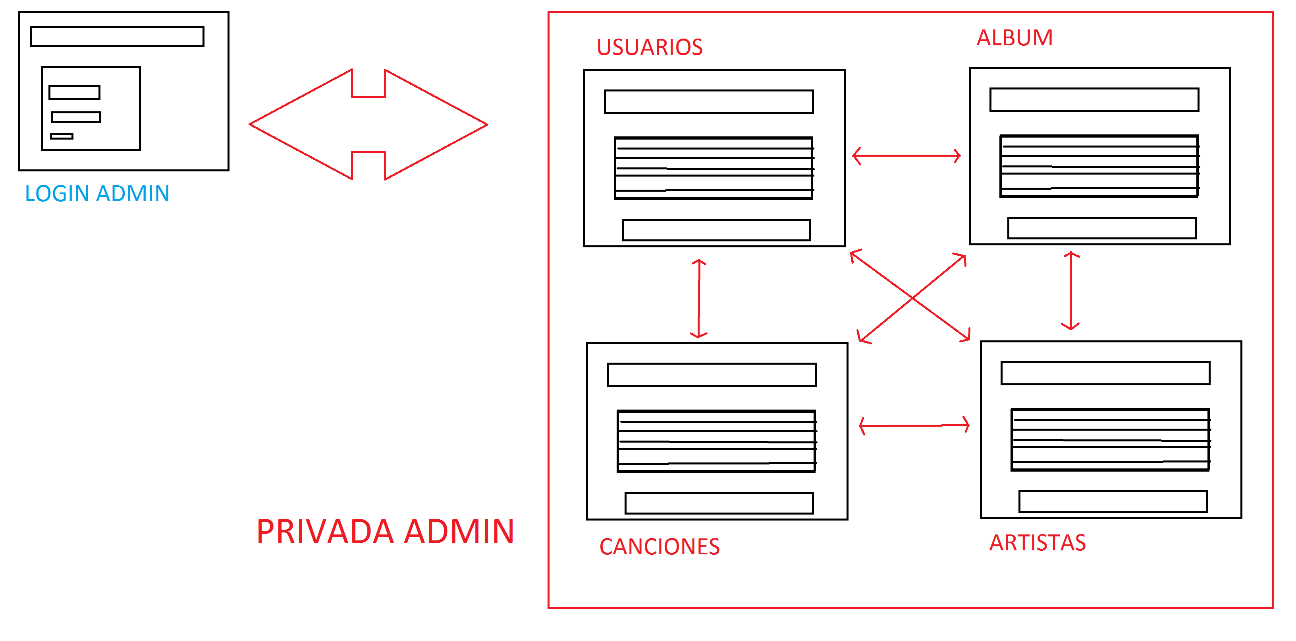
Hemos realizado una implementación de Spotify, en la parte pública hemos desarrollado el Login (de usuarios de la aplicación y administradores), Registro, recuperar contraseña y un “acercade” de la aplicación.

En la parte privada hemos realizado dos partes independientes. Una para administradores y otras para usuarios.

La parte privada de administrador incluye CRUD de Usuarios, Canciones, Artistas y álbumes. En la parte privada de usuario encontramos un menú de navegación(dashboard) como página de inicio, Listas de reproducción del usuario, una lista de canciones paginadas, una lista de artistas paginado y un buscador de canciones. Se puede acceder al detalle de artista, donde se verán sus detalles y sus canciones, al detalle de una canción en modo de reproducción. El modo de reproducción es un pie de página donde se podrá ver los detalles de las canciones.

Wireframes y Mockups





DIAGRAMAS

DESCRIPCIÓN TÉCNICA

Hemos desarrollado una aplicación web usando el Framework CodeIgniter de PHP. La arquitectura de la aplicación está realizada en capas.

Vista<->Controlador<->Modelo<->Base de Datos

Para la base de datos se ha implementado en MySQL utilizando los clientes Workbench y phpMyAdmin.

Por lo tanto, serían necesarios dos servidores, el de la base de datos y el apache donde se mantiene la aplicación web.

Para el desarrollo web se ha utilizado el framework CSS de Bootstrap.

Por último, hemos usado Grocery CRUD para los cruds de la parte privada.

DETALLES DE IMPLEMENTACIÓN

La implementación se ha llevado a cabo utilizando un conjunto de modelo, vista, controlador para cada uno de los elementos de la aplicación. Es decir, uno para Usuario donde se controla el inicio de sesión y la parte pública; otro para la parte privada de usuario de la aplicación que hemos llamado “inapp” que tendría el controlador inapp pero usaría modelos independientes, uno para el buscador, otro para las canciones, otro de artistas y por último uno de listas de reproducción; el último controlador sería el de la parte privada de usuario administrador, que hace uno de un controlador y del modelo del Grocery Crud para obtener los datos.

La sesión es independiente en los dos roles de usuario que tenemos, es decir tenemos dos tipos de usuario uno que es solo de la aplicación y otro de la aplicación y administrador. El usuario básico solo puede entrar a través del login de usuario, pero el administrador puede entrar por ambos login, pero solo se le mantiene la sesión en la parte de la aplicación por el que haya entrado.

Si un usuario intenta entrar a la aplicación si haberse autentificado, la aplicación le manda directamente al login, y de forma contraria si ya está autentificado e intenta entrar en la parte publica le redirecciona al dashboard de la parte privada.

En ningún momento se solapan las capas para la obtención de datos ya que no lo hemos visto necesario y podría dar problemas.

Hemos usado la sesión de CodeIgniter para mantener la sesión, la canción que se escucha en cada momento y la última búsqueda en la aplicación. Las listas que no están delimitadas, como son las canciones, los artistas y las búsquedas tienen paginación realizada con las posibilidades que da CodeIgniter. Las listas que están limitadas como puede ser las canciones de un artista, o las listas de reproducción no tiene paginación ya que son listas acotadas y pensamos que de cara al usuario es mejor tener todos los datos de golpe y al ser limitadas no deberían tener un peso demasiado grande.

La aplicación respeta los estándares HTML5 y se ha programador de tal forma que se mantenga “responsive” para poder usarla desde un móvil o navegador sin problemas.

Problemas Encontrados

La paginación. No conseguíamos hacer la paginación de la forma que teníamos pensada, que era siguiendo un esquema rest enviar peticiones por queryString por el navegador.

La solución ha sido usar la forma predefinida de CodeIgniter, aunque haya que configurarla es mucho más sencilla de aplicar.

Reproducción de archivos desde el navegador. No hemos conseguido implementarlo.

Mejoras y ampliaciones

A parte de la reproducción de música. Usar los álbumes que los hemos obviado durante la implementación para agrupar la música. Una sección de configuración del usuario donde se le permita editar sus parámetros y darse de baja si lo viera conveniente.

La inclusión de publicidad en la interfaz de la aplicación y la personalización de la interfaz son mejoras que se podrían llevar a cabo sin mucho coste y darían un buen resultado en el conjunto de la misma.

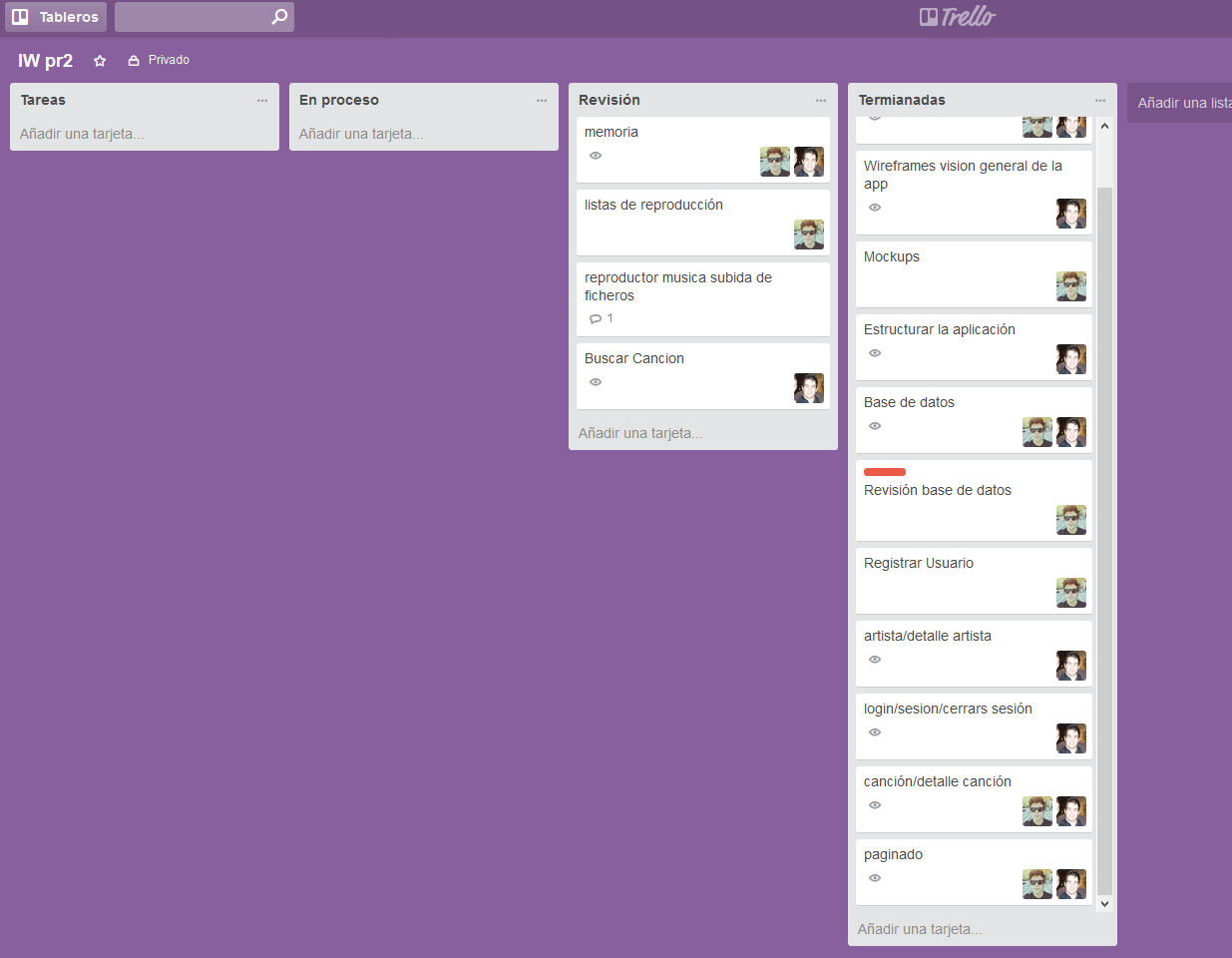
Tareas Adicionales

Github

Hemos utilizado github para poder trabajar juntos.

Trello

Hemos utilizado esta herramienta para organizar el trabajo del equipo.



Referencias

<https://www.codeigniter.com/user_guide/>

<http://www.grocerycrud.com/documentation>

<http://getbootstrap.com/>

<http://www.w3schools.com>

<https://github.com/>

<https://trello.com/>