

APPVIDEO

Tecnologías de Desarrollo Software



*Convocatoria Enero
2021/2022*

Alumnos:

*César Pagán Villafañe
Ángel Tomás Perea
Nieto*

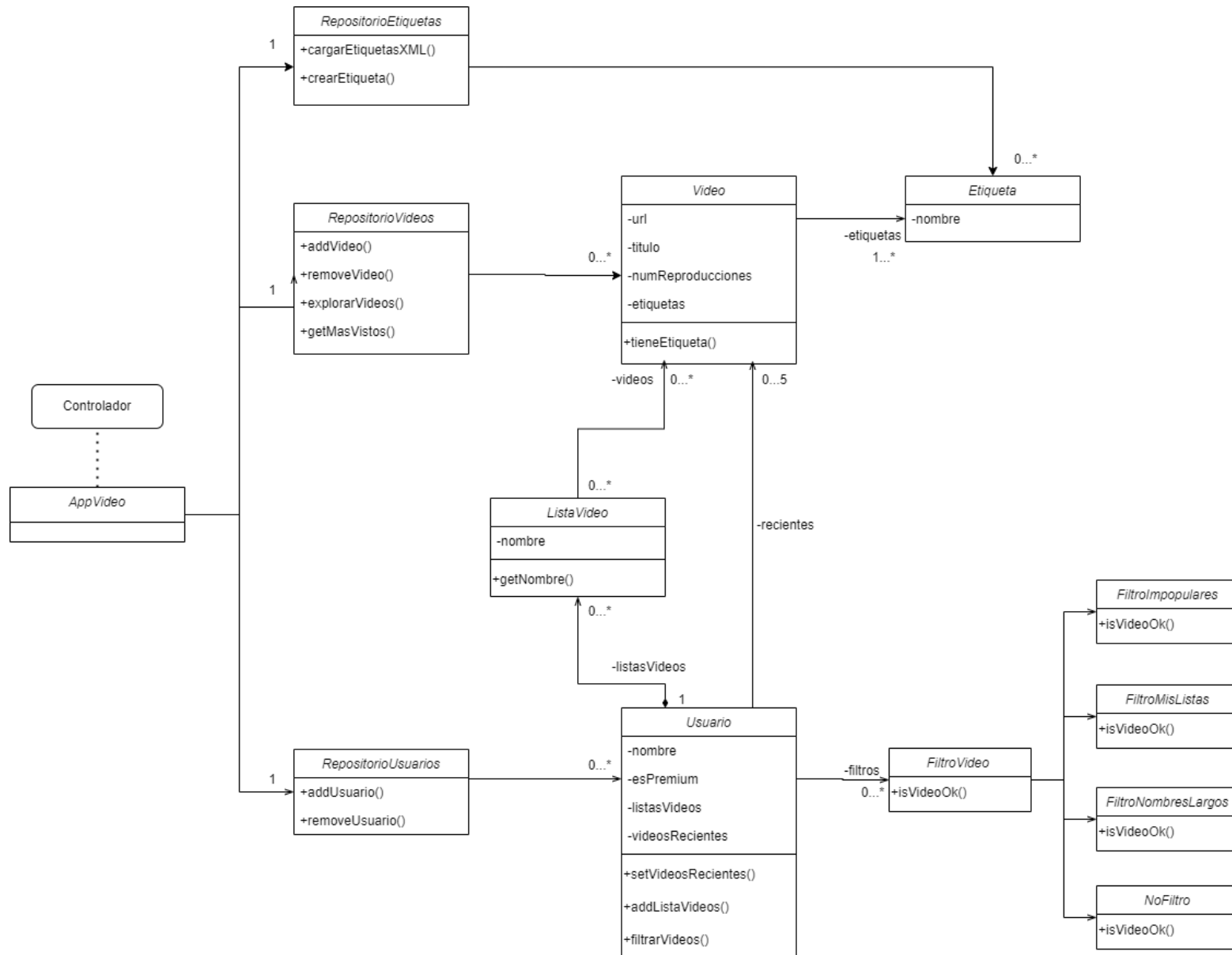
Profesor:

*Jesús Joaquín García
Molina*

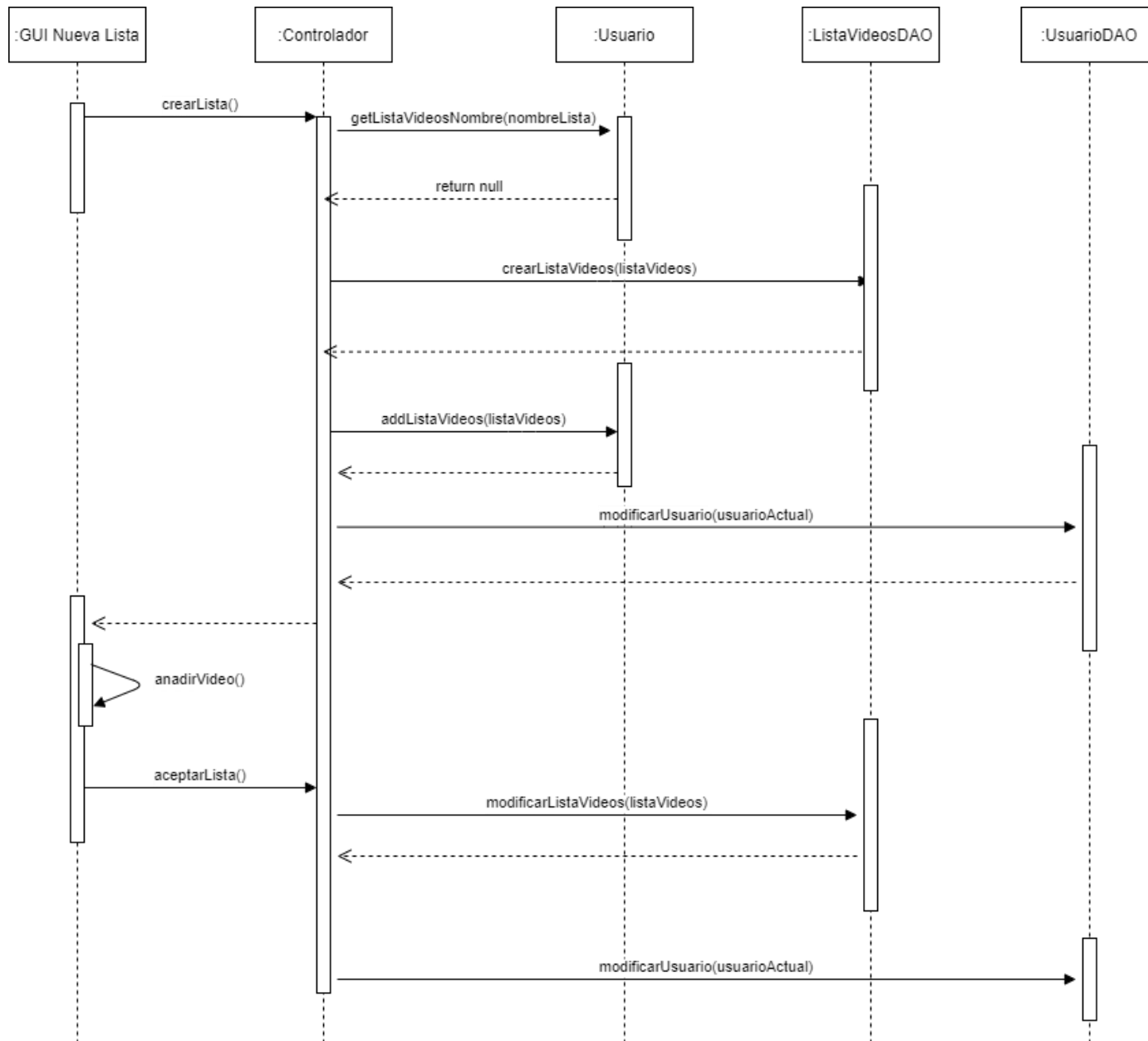
Índice de contenidos

1.	Diagrama de clases del dominio.....	3
2.	Diagrama de interacción UML.....	4
3.	Arquitectura y diseño de la aplicación.....	5
4.	Patrones de diseño.....	7
5.	Componentes utilizados.....	9
6.	Test unitarios.....	10
7.	Manual de la aplicación.....	12
8.	Observaciones finales.....	20

1. Diagrama de clases del dominio.



2. Diagrama de interacción UML.



3. Arquitectura y diseño de la aplicación.

La aplicación APPVideo consta de hasta 9 paquetes distintos, el objetivo era conseguir un proyecto bien planificado y ordenado:

APPVIDEO.TEST

Este paquete que se encuentra dentro de la carpeta src/test/java contiene la clase AppTest.java la cual contiene todos los test unitarios para la comprobación de una clase del modelo, en concreto, sobre RepositorioVideos. Sobre estos tests se hablará más adelante en el apartado de los Test Unitarios.

APPVIDEO.IMAGENES

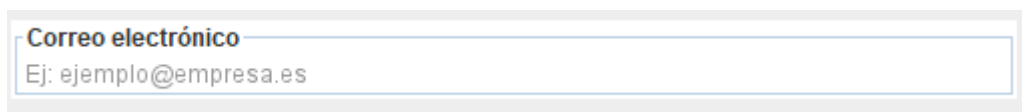
Aquí se encuentran todas las imágenes con extensión .png que han sido utilizadas para el diseño de la interfaz gráfica del proyecto.

APPVIDEO.MAIN

Únicamente se encuentra la clase Launcher.java, se trata del lanzador de la aplicación, se encarga de crear la vista principal desde la cual se da paso al resto de vistas y de inicializar el componente VideoWeb sobre el que más tarde especificaremos.

APPVIDEO.EXTRA

Contiene 3 clases auxiliares distintas las cuáles hemos querido distinguir añadiendo en este paquete. RenderVideos.java se encarga de renderizar una tabla y lista de videos correctamente. VideoTableModel nos ayuda a crear un modelo de tabla destinado a la clase Video. TextPrompt recrea un placeholder, ayuda a que mostremos un mensaje dentro de un JTextField. Ejemplo de uso:



APPVIDEO.CONTROLADOR

En este paquete se encuentra únicamente la clase Controlador.java que se encarga de que la comunicación entre el servidor, interfaz gráfica y servicio de persistencia sea posible sin realizar ninguna violación de los patrones vistos.

APPVIDEO.VISTAS

Dentro de este paquete aparecen hasta 7 clases distintas dedicadas a la interfaz gráfica. Como ya se ha mencionado antes mientras se comentaba el paquete appvideo.main, tenemos una clase VistaPrincipal desde la cual se encuentra definido un panel por defecto que podemos considerar como una especie de header que se podrá observar en todo momento para navegar por las distintas opciones que ofrece la aplicación, únicamente

estará oculto en el momento en el que se muestra vista de login (que está implementada en la vista principal) y registro. A continuación del header tendremos otro panel sobre el cual iremos sustituyendo por la vista que sea seleccionada en cada momento.

APPVIDEO.DOMINIO

El paquete del dominio almacena todas las clases que definen entidades que serán persistentes en el servidor: Usuario, Video, Etiqueta y ListaVideos. También se encuentran los repositorios locales de las entidades anteriores excepto de ListasVideos ya que es una propiedad de cada Usuario. Finalmente se encuentra la interfaz de FiltroVideo con 4 clases que lo implementan: FiltroImpopulares, FiltroMisListas, FiltroNombresLargos y NoFiltro.

APPVIDEO.PERSISTENCIA

Contiene las clases TDSFactoriaDAO, FactoriaDAOAbstracto y DAOException que son necesarias para poder realizar la implementación de un abstract factory. La clase TDSFactoriaDAO contiene los métodos get que devuelven la única instancia de todos los DAO implementados para acceder al servicio de persistencia de la forma correcta.

Los DAO implementados son: TDSEtiquetaDAO, TDSListaVideosDAO, TDSUsuarioDAO y TDSVideoDAO. Todas estas clases tienen sus métodos adaptados de creación, eliminación, modificación y la obtención de la entidad en la persistencia. Además según las propiedades de cada clase, tiene su método parse implementado para conseguir correctamente la transformación objeto-entidad y entidad-objeto.

4. Patrones de diseño.

En este apartado explicamos los patrones que hemos utilizado separados por categorías:

Patrones creacionales

El patrón Factoría Abstracta se ha utilizado para crear los adaptadores que utilizamos en el proyecto y nos permite crear, mediante una interfaz, conjuntos o familias de objetos que dependen mutuamente sin especificar cual es el objeto concreto.

El patrón Singleton lo hemos empleado en las clases de persistencia, controlador, factoría, adaptadores y catálogos, nos ha permitido asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia.

Patrones estructurales

El patrón Adaptador se utiliza junto a los patrones Factoría Abstracta y Singleton para implementar correctamente el patrón DAO, es un patrón de diseño estructural que permite la colaboración entre objetos con interfaces incompatibles, en nuestro caso lo hemos utilizado para el acceso al servidor de persistencia desde las distintas entidades.

El patrón Fachada tiene la característica de ocultar la complejidad de interactuar con un conjunto de subsistemas proporcionando una interfaz de alto nivel, la cual se encarga de realizar la comunicación con todos los subsistemas necesarios. La fachada es una buena estrategia cuando requerimos interactuar con varios subsistemas para realizar una operación concreta ya que se necesita tener el conocimiento técnico y funcional para saber qué operaciones de cada subsistema tenemos que ejecutar y en qué orden, lo que puede resultar muy complicado cuando los sistemas empiezan a crecer demasiado. Nosotros hemos utilizado este patrón en el controlador para separar el modelo-vista dejando que ambas vistas se comuniquen mediante el controlador, siendo este la "fachada" de la capa de presentación y al contrario. Otro sitio donde lo hemos utilizado es en el servicio de persistencia, ofreciéndonos este componente una serie de funciones y métodos para hacer uso del mismo, pero sin hacer referencia directa a los propios métodos internos del servicio.

El patrón Composite sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol, nosotros lo hemos puesto en funcionamiento con la herramienta Swing ya que las vistas que hemos creado utilizan JPanel que son contenedores que pueden contener otros JPanel u otros elementos como botones o etiquetas entre otros.

El patrón Decorador permite añadir funcionalidades a objetos colocando estos objetos dentro de objetos encapsuladores especiales que contienen estas funcionalidades, nosotros lo hemos utilizado en las vistas, por ejemplo, con el desplazamiento vertical.

Patrones de comportamiento

El patrón estrategia permite establecer en tiempo de ejecución el rol de comportamiento de una clase, este comportamiento se basa en el polimorfismo para implementar una serie de comportamientos que podrán ser intercambiados durante la ejecución del programa, logrando con esto que un objeto se pueda comportar de forma distinta según la estrategia establecida. Este patrón lo hemos utilizado para implementar los distintos tipos de filtros, para ello hemos creado una interfaz que es implementada por las subclases que hemos creado para cada tipo de filtro, esto nos permite generalizar las operaciones llamando directamente a los métodos de la interfaz independientemente de la clase de filtro con la que queramos trabajar.

El patrón Observer define una dependencia del tipo uno a muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes, en nuestro caso lo hemos utilizado por ejemplo con los `actionListeners`.

5. Componentes utilizados.

Swing: es una librería de Java que hemos utilizado para crear la aplicación gráfica que permite la interacción con el usuario.

Cargador de vídeos: Es un componente necesario para cargar los videos en la plataforma desde una fuente externa. Este componente contiene la clase VideoListener que se implementa en el Controlador y que dice lo que hay que hacer una vez se notifique del evento.

Luz: Es un componente que se nos ha facilitado desde la asignatura y lo hemos utilizado para que cuando sea pulsado se habrá FileChooser que permita seleccionar el fichero .xml que será el que use el cargador de vídeos para cargar los vídeos de dicho fichero, cuando se completa mediante un evento el componente cambia de color verde a azul.

VideoWeb: Por otro lado, el componente VideoWeb también facilitado por la asignatura cuyo uso nos ha permitido reproducir los vídeos.

JCalendar: Es un componente también obtenido del aula virtual y lo hemos utilizado en la pantalla de registro para abrir un desplegable cuando sea seleccionado para elegir una fecha del calendario, en vez de tener que introducirla a mano evitando así problemas de formato.

6. Test unitarios.

Para la realización de test JUnit hemos decidido optar por comprobar los métodos de la clase RepositorioVideos. Todos los tests están incluidos en la clase AppTest de la carpeta src/test/java. Podemos comprobar que estos cubren todos los métodos de la clase RepositorioVideos y de forma correcta con la herramienta de Eclipse:

Click derecho en el proyecto -> Coverage As -> JUnit Test

Al seleccionar esta opción vamos a la clase RepositorioVideos y observamos como todo está cubierto en verde:

```
public void addVideo(Video video) {
    coleccionVideosID.put(video.getId(), video);
    coleccionVideosTitulo.put(video.getTitulo(), video);
}

public void removeVideo(Video video) {
    coleccionVideosID.remove(video.getId());
    coleccionVideosTitulo.remove(video.getTitulo());
}

public List<Video> explorarVideos(String titulo, LinkedList<Etiqueta> etiquetas) throws DAOException {
    List<Video> videos = new LinkedList<>();
    for (Video video : coleccionVideosTitulo.values()) {
        if ((titulo.isEmpty() && video.tieneEtiqueta(etiquetas)) || ( video.getTitulo().contains(titulo) && video.
            videos.add(video);
    }
    return videos;
}

public List<Video> getMasVistos(){
    List<Video> listaVideoTopTen = coleccionVideosTitulo.values().stream()
        .sorted((video1, video2)->video2.getNumReproducciones()-video1.getNumReproducciones())
        .limit(10)
        .collect(Collectors.toList());

    return listaVideoTopTen;
}
```

Antes de realizar los tests fue necesario crear una función setUp con la etiqueta @Before para poder manejar distintos datos de forma más cómoda:

```
public class AppTest {

    public RepositorioVideos repositorio;
    public List<Video> coleccionVideos;
    public Video video1;
    public Video video2;
    public Video video3;

    @Before
    public void setUp() throws DAOException {
        repositorio = RepositorioVideos.getUnicaInstancia();

        // Quitamos los posibles videos que haya debido a la persistencia para poder ser
        // conscientes al 100% de los videos que manejamos.
        for (Video video : repositorio.getVideos()) {
            repositorio.removeVideo(video);
        }

        coleccionVideos = new LinkedList<Video>();

        video1 = new Video("Invasion", "https://www.youtube.com/watch?v=DlaHxL3mHAU");
        video1.setId(1);
        video1.setNumReproducciones(0);
        coleccionVideos.add(video1);
        video2 = new Video("Arctic Monkeys - Do I Wanna Know?", "https://www.youtube.com/watch?v=bp0SxM0rNPM");
        video2.setId(2);
        video2.setNumReproducciones(5);
        coleccionVideos.add(video2);
        video3 = new Video("Me at the zoo", "https://www.youtube.com/watch?v=jNQXAC9IVRw", new Etiqueta("Zoo"));
        video3.setId(3);
        video3.setNumReproducciones(10);
        coleccionVideos.add(video3);
    }
}
```

Al ser 4 métodos fueron necesarios 3 test sumando 2 más del método explorarVideos ya que tiene dos situaciones distintas:

```
@Test
public void testAddVideo() throws DAOException {

    // Pruebo el método addVideo al repositorio
    for (Video video : coleccionVideos) {
        repositorio.addVideo(video);
    }

    LinkedList<Video> videosRepositorio = repositorio.getVideos();

    // Compruebo que se han añadido
    for (Video video : coleccionVideos) {
        assertTrue(videosRepositorio.contains(video));
    }
}

@Test
public void testRemoveVideo() throws DAOException {
    // Para borrar primero debemos añadir, sabemos que funciona correctamente por el test anterior
    for (Video video : coleccionVideos) {
        repositorio.addVideo(video);
    }

    // Pruebo el método removeVideo del repositorio
    for (Video video : coleccionVideos) {
        repositorio.removeVideo(video);
    }

    LinkedList<Video> videosRepositorio = repositorio.getVideos();

    // Compruebo que se han añadido
    for (Video video : coleccionVideos) {
        assertFalse(videosRepositorio.contains(video));
    }
}
```

```
@Test
public void testExplorarVideoTitulo() throws DAOException {
    // Para buscar primero debemos añadir, sabemos que funciona correctamente por el test anterior
    for (Video video : coleccionVideos) {
        repositorio.addVideo(video);
    }

    // Pruebo el método explorarVideos del repositorio
    List<Video> videosExplorar = repositorio.explorarVideos("Me at", null);
    // Compruebo que la búsqueda es correcta
    assertEquals(videosExplorar.get(0).getTitulo(), video3.getTitulo());
}

@Test
public void testExplorarVideoSinTitulo() throws DAOException {
    // Para buscar primero debemos añadir, sabemos que funciona correctamente por el test anterior
    for (Video video : coleccionVideos) {
        repositorio.addVideo(video);
    }

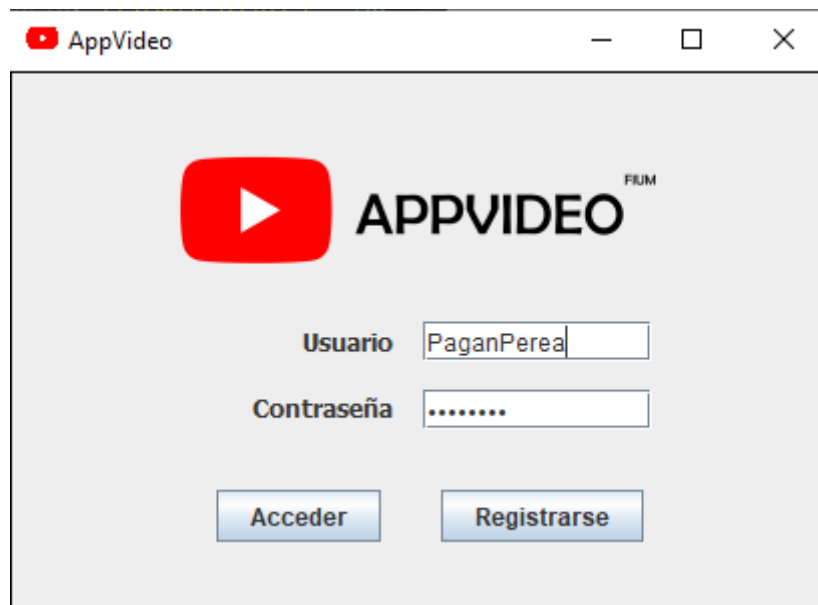
    // Pruebo el método explorarVideos del repositorio
    List<Video> videosExplorar = repositorio.explorarVideos(new String(), video3.getEtiquetas());
    // Compruebo que la búsqueda es correcta
    assertEquals(videosExplorar.get(0).getTitulo(), video3.getTitulo());
}
```

```
@Test
public void testgetMasVistos() throws DAOException {
    // Para buscar primero debemos añadir, sabemos que funciona correctamente por el test anterior
    for (Video video : coleccionVideos) {
        repositorio.addVideo(video);
    }

    // Pruebo el método explorarVideos del repositorio
    List<Video> videosMasVistos = repositorio.getMasVistos();
    // Compruebo que devuelve una lista con el orden correcto
    assertEquals(videosMasVistos.get(0).getTitulo(), video3.getTitulo());
    assertEquals(videosMasVistos.get(1).getTitulo(), video2.getTitulo());
    assertEquals(videosMasVistos.get(2).getTitulo(), video1.getTitulo());
}
```

7. Manual de la aplicación.

Al iniciar la aplicación nos encontramos la ventana para iniciar sesión que también nos da la posibilidad de registrarnos:



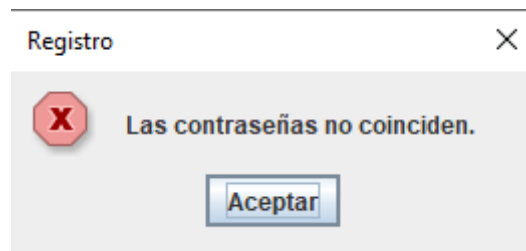
The screenshot shows a window titled "AppVideo" with standard Windows window controls (minimize, maximize, close). The window content features the AppVideo logo, which consists of a red play button icon and the text "APPVIDEO" with "FUM" in smaller letters above the "O". Below the logo, there are two input fields: "Usuario" with the text "PaganPerea" and "Contraseña" with masked characters ".....". At the bottom, there are two buttons: "Acceder" and "Registrarse".

En caso de elegir la opción de registrarse porque no tenemos una cuenta, nos da una nueva vista que nos pide rellenar los siguientes campos:

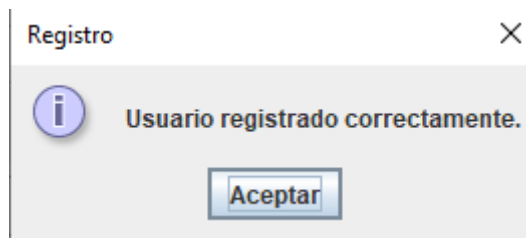


The screenshot shows a registration form titled "Registro para AppVideo". It contains several input fields: "Nombre", "Apellidos", "Correo electrónico" (with a hint "Ej: ejemplo@empresa.es"), "Usuario", "Fecha de Nacimiento" (with a calendar icon), "Contraseña", and "Confirmar Contraseña". At the bottom, there are two buttons: "Volver" and "Registrarse".

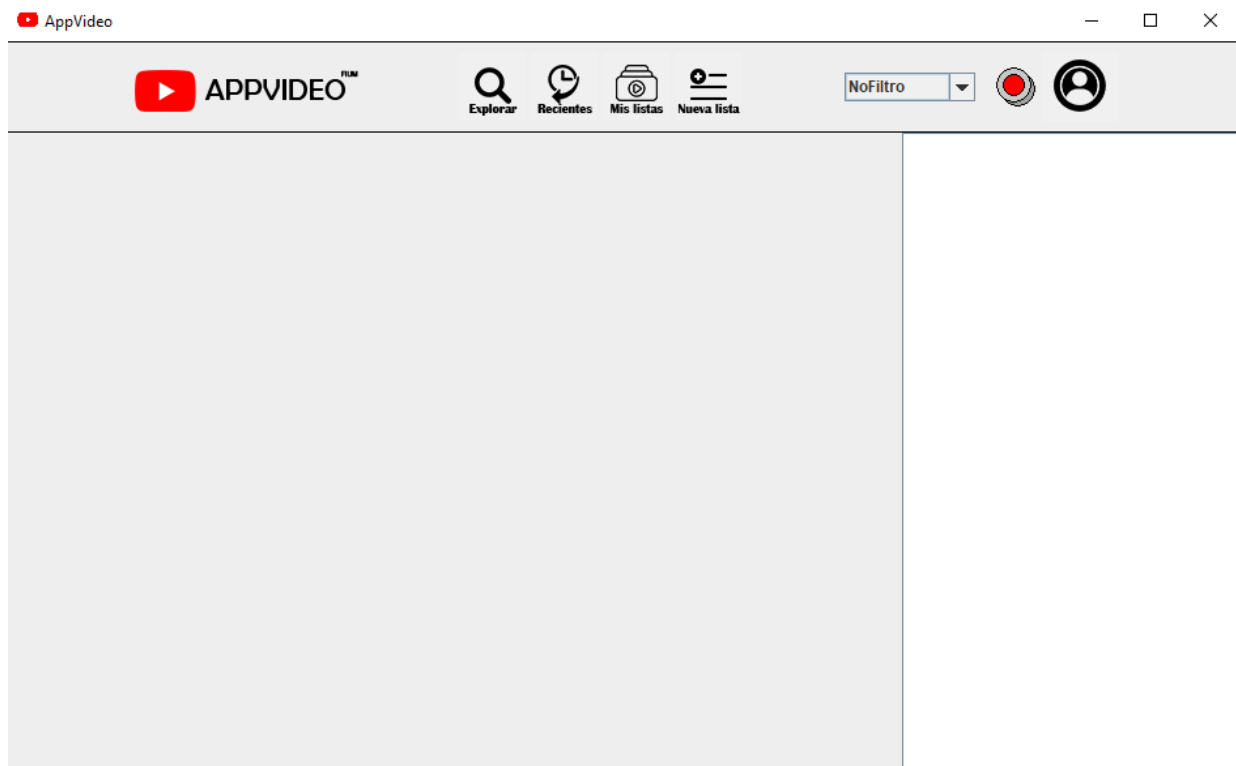
En caso de que no rellenemos algún campo, el email no siga la sintaxis correcta, se introduzca un usuario que ya está registrado o las contraseñas no coincidan se da un mensaje de error, por ejemplo, en este las contraseñas no coinciden:



Cuando nos registramos no sale un mensaje informativo de que todo ha ido bien y nos vuelve a llevar a la pantalla anterior para iniciar sesión.



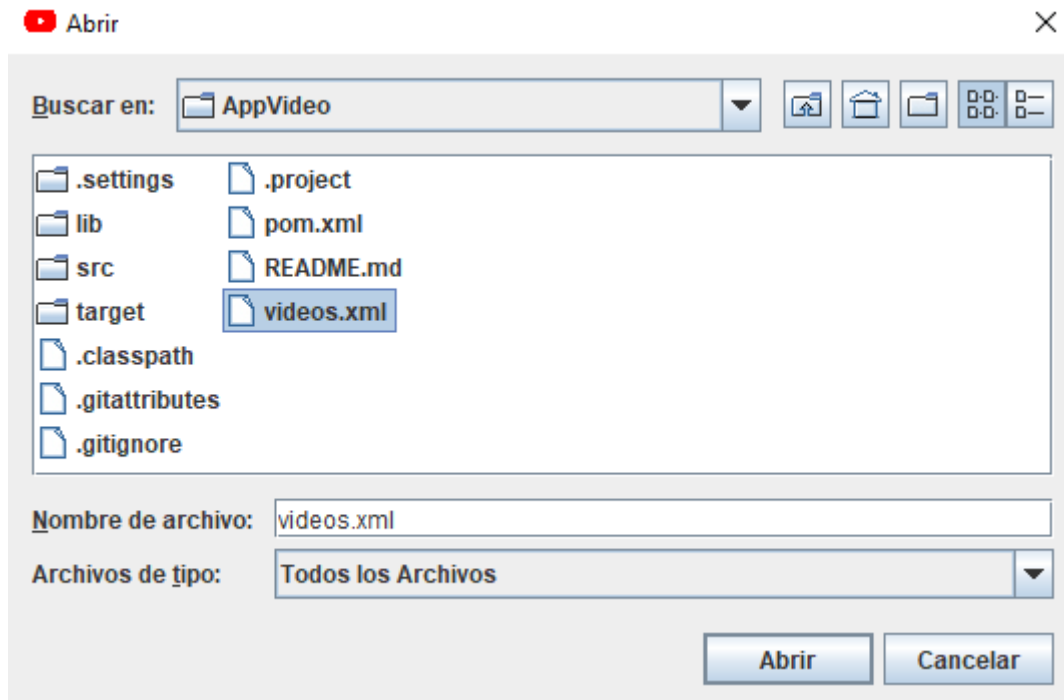
Al igual que en el registro, en caso de introducir un usuario que no exista saldrá un mensaje de error.



Cuando iniciamos sesión por primera vez esta es la ventana que nos encontramos, totalmente vacía, esto se debe a que, al iniciar sesión, tal y como se pedía, enseñamos la lista de videos recientes del usuario y hasta ahora el usuario no ha visto ningún video, por lo que la lista se encuentra vacía.

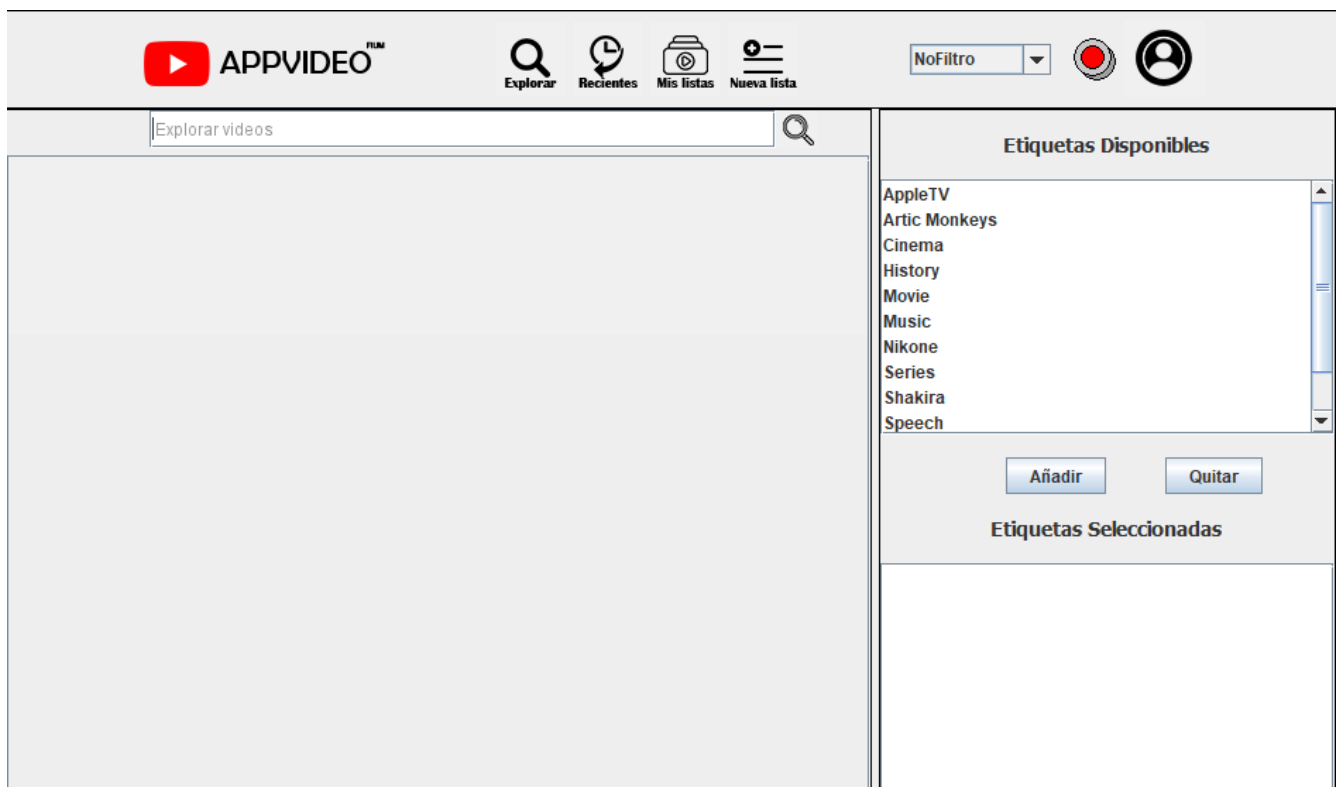
Antes de poder visualizar algún video es necesario exportarlos en caso de no tenerlo todavía, para ello pulsaremos en el botón rojo que hace uso del componente Luz.

Aparecerá una nueva ventana desde la cual tendremos que seleccionar el archivo videos.xml:



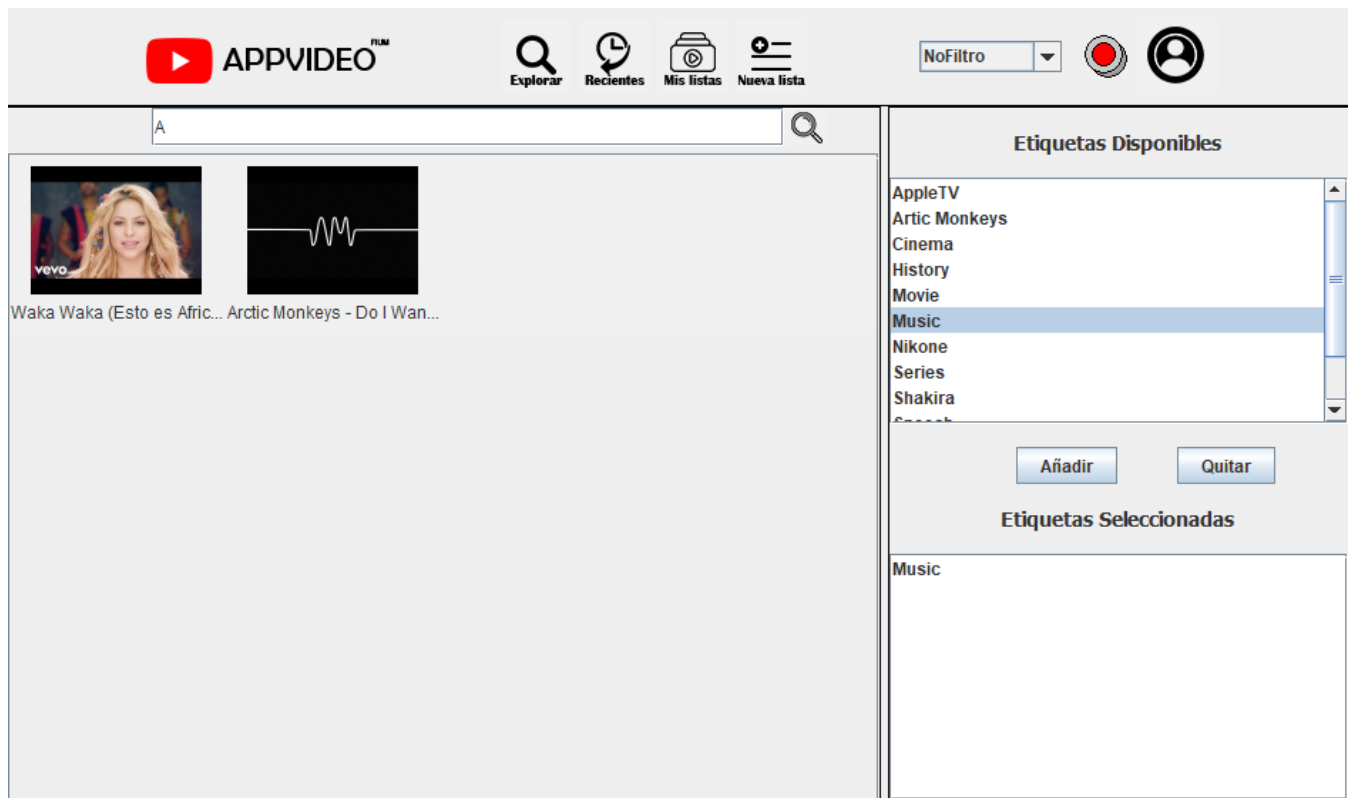
Una vez tenemos los videos exportados podemos tratar con todas las funciones que contiene la aplicación, las iremos repasando de izquierda a derecha (según el header).

Empezamos con la funcionalidad del botón explorar:



Cuando hacemos click nos muestra una ventana que contiene un buscador para filtrar los videos por títulos, también a la derecha nos aparece una lista de todas las etiquetas que conjuntan todos los videos para que podamos filtrar por estas también. En caso de

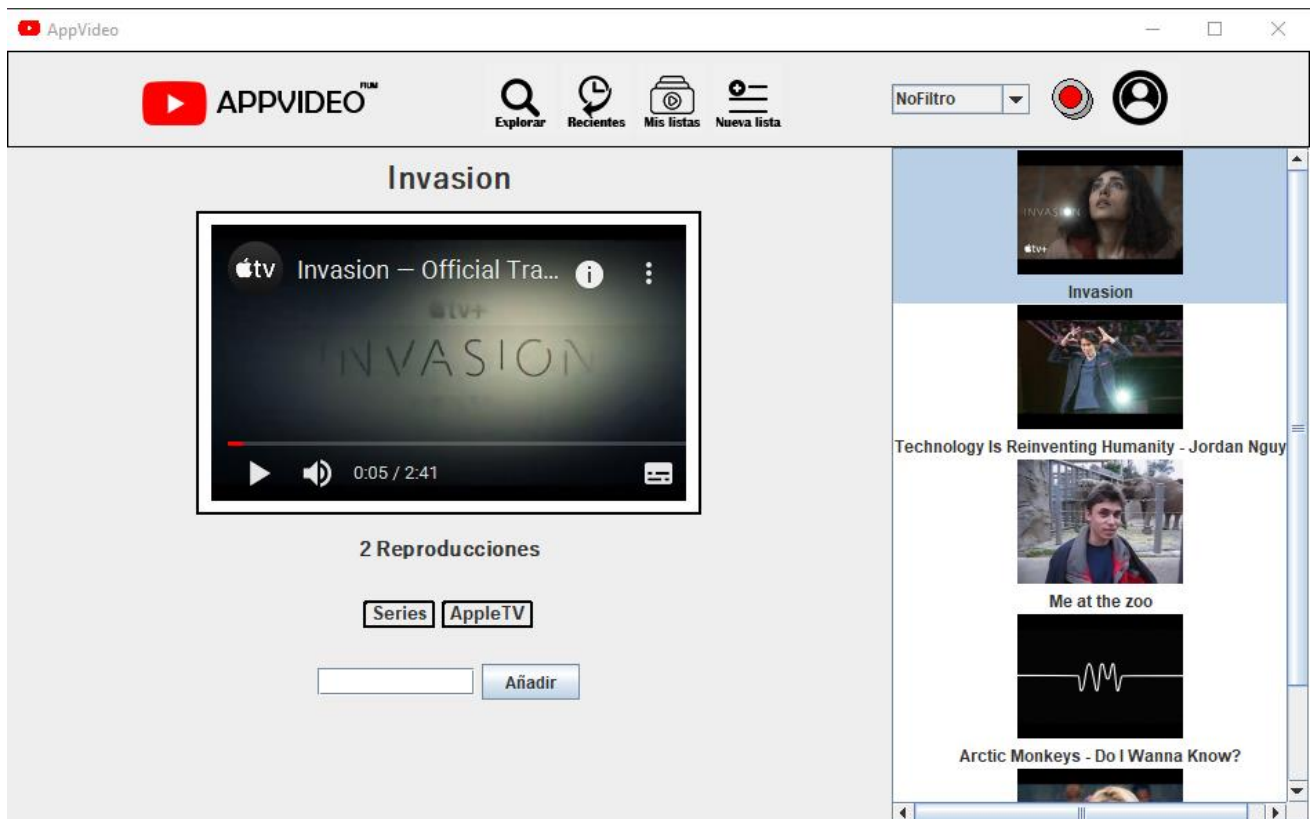
no escribir nada en el título y no añadir ninguna etiqueta si hacemos click en la lupa del buscador hará que nos aparezcan todos los videos. Vamos a mostrar un ejemplo de búsqueda:



Si hacemos doble click en alguno de los videos mostrados cambiará la vista desde la cual comenzará a reproducirse y encontraremos más información. Además, nos da la posibilidad de añadir alguna etiqueta nueva al video en caso de querer.

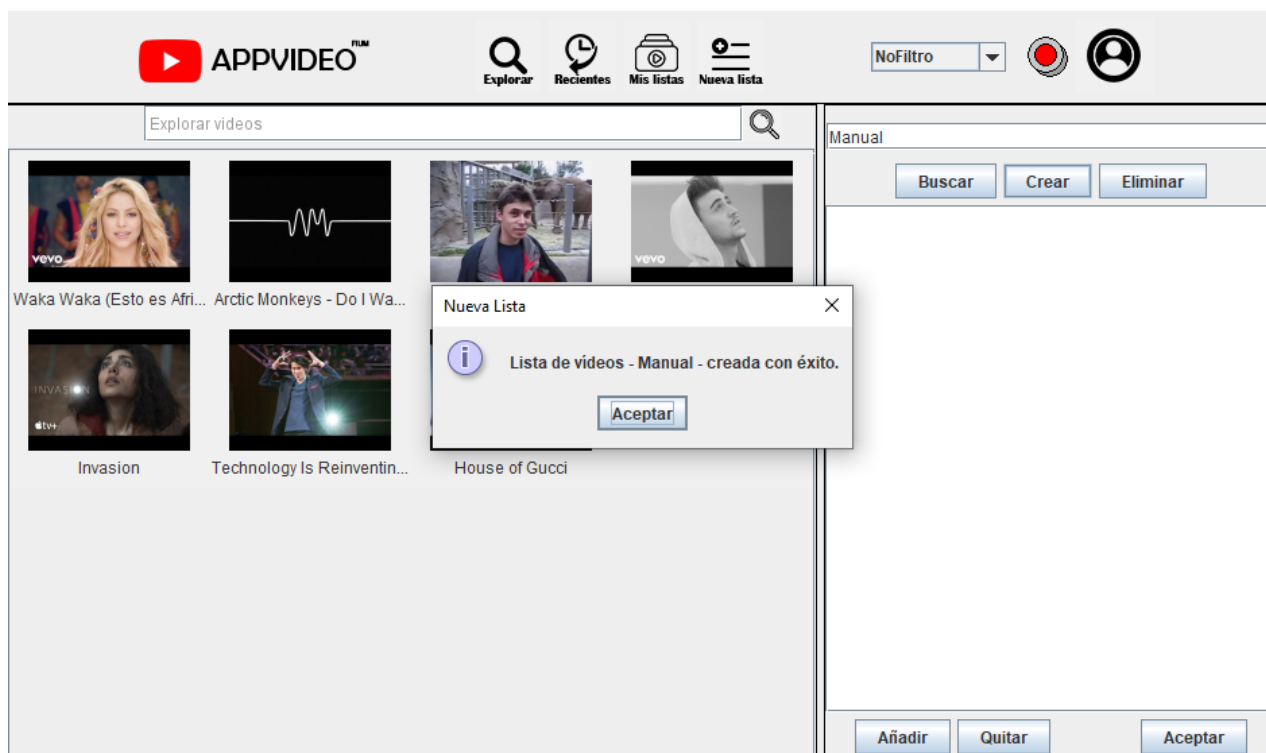


La siguiente opción es la de recientes, la cual muestra los últimos 5 videos que han sido reproducidos por el usuario, a la derecha aparecen los 5 (o menos en caso de no haber reproducido 5 aún) videos con su miniatura y título, si hacemos doble click en alguno se reproducirá, acompañará con información y también dará la opción de añadir etiquetas.

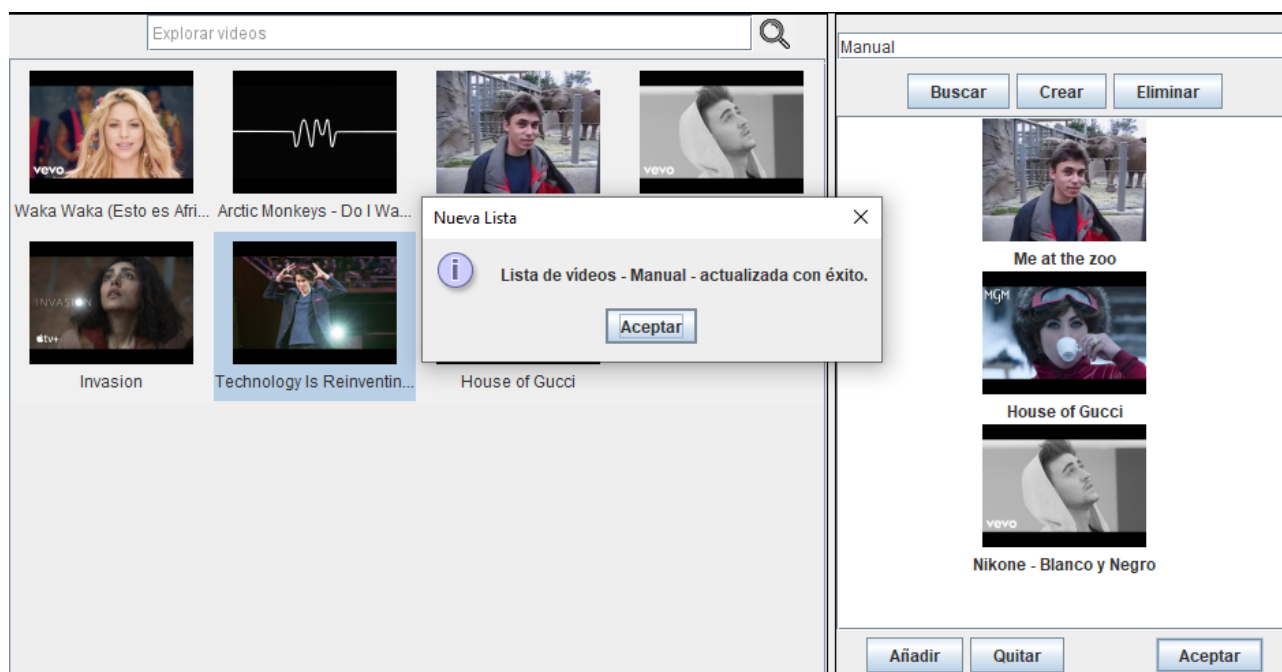


La siguiente opción es la de Mis Listas, pero si somos un usuario nuevo al no tener ninguna, para hacer uso de esta opción primero deberemos tener listas, por tanto, iremos a la opción de Nueva Lista.

Al hacer click nos sale una nueva vista en la que se distinguen dos zonas, a la izquierda tenemos todos los videos que contiene la aplicación con su explorador y a la derecha el panel para manejar las listas. Para crear una lista introduciremos el nombre y le daremos a crear, si antes de crearla añadimos videos y luego la creamos esta lista no incorporará los videos añadidos.

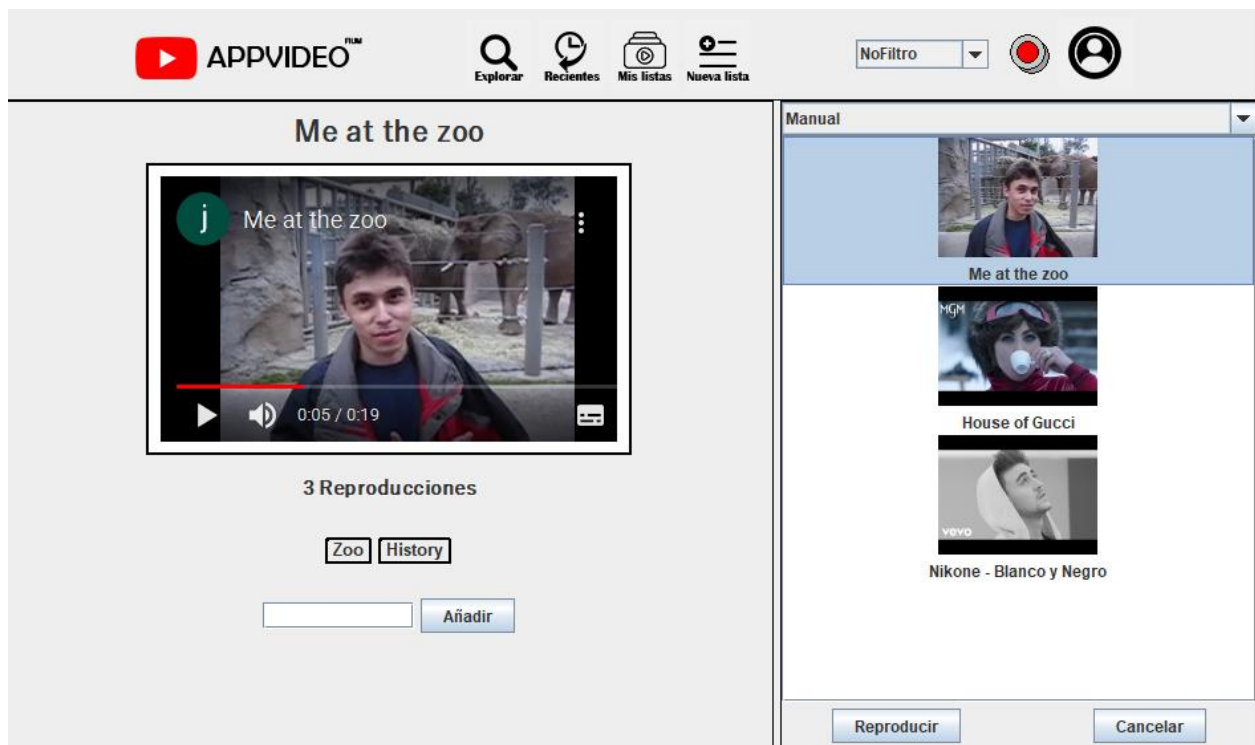


Tras esto podemos seleccionar videos del explorador y añadirlos o en caso de equivocarnos quitarlos, cuando tengamos la lista preparada presionamos el botón de aceptar y se actualizará.



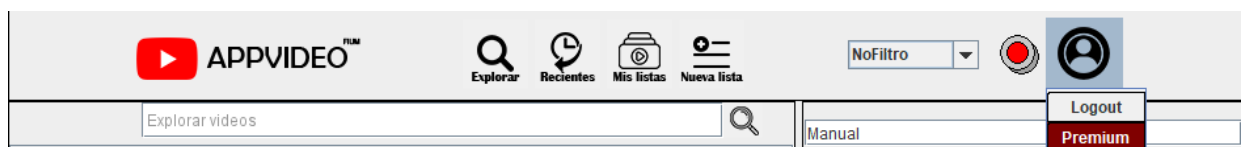
En caso de querer buscar una lista ya creada para modificarla tendremos que buscarla antes, para eliminarla con poner el nombre correctamente es suficiente. En caso de realizar acciones que no se deben, como eliminar una lista que no existe aparecerán mensajes de error.

Ahora si nos dirigimos a la opción de Mis Listas en el desplegable aparecerán las distintas listas que tenemos, al seleccionar una nos saldrá en el panel derecho todos los videos que contiene la lista, podemos seleccionar el que queramos y darle al botón de reproducir, empezará a reproducirse el video de la misma forma que hace en la opción de Recientes.

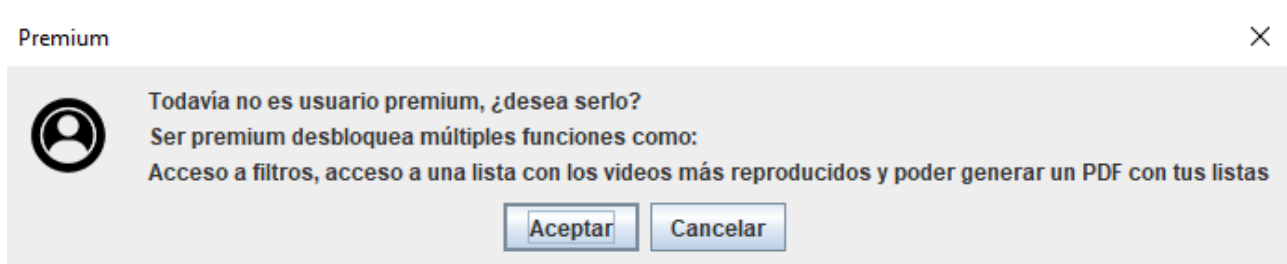


Además, se incluye la opción de Cancelar para cancelar la reproducción del video.

La siguiente opción es el desplegable en el que pone "NoFiltro" pero como no somos usuarios premium todavía no podemos darle uso. Por lo que pasaremos a ver la funcionalidad que incluye la "foto de perfil"



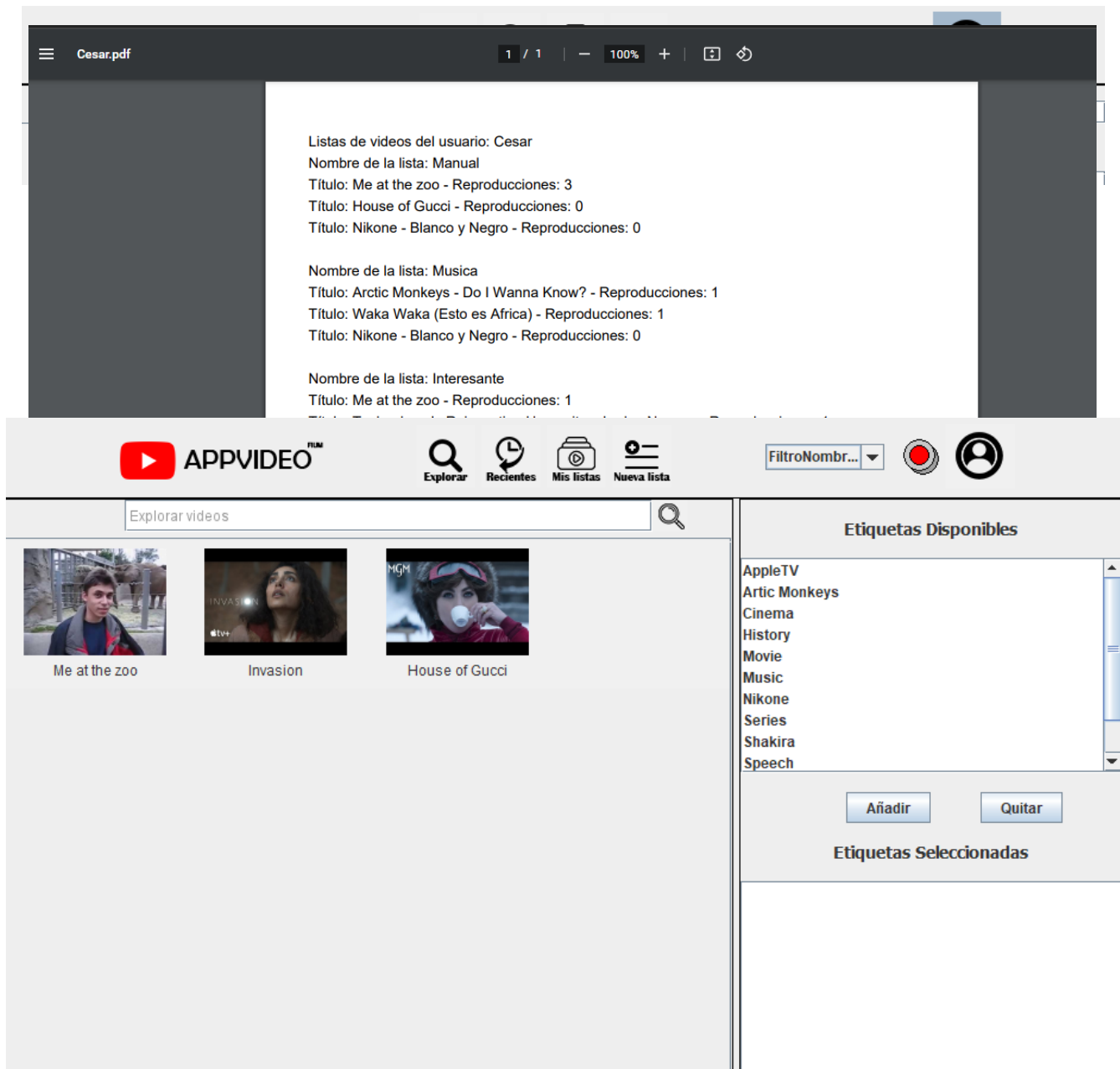
Por ahora tenemos dos opciones, logout para cerrar sesión (volvemos a la pantalla de inicio de sesión) y el de premium para convertirnos en usuarios premium:



Al aceptar, si volvemos a dar click en la “foto de perfil” ahora saldrán nuevas opciones:

El botón de PDF generará un PDF en la ruta E:\\ que se llamará “nombre_real_usuario.pdf” que explica cuales son las listas de videos del usuario:

El botón Top 10 nos mostrará una vista similar a la de Recientes, el único cambio es que sale ordenado los 10 videos con más reproducciones, el primero el que más tiene y el décimo el que menos.



Como última función tenemos desbloqueados todos los filtros implementados, si abrimos el desplegable vemos 4 opciones, al escoger una, cuando exploramos videos se aplica este filtro, de este modo nos mostrará los videos que no cumplen las condiciones del filtro, por ejemplo, este es el resultado usando filtro nombre largo:

Estas serían todas las funciones de AppVideo.

8. Observaciones finales.

Como observaciones finales de la práctica, por lo general nos ha parecido una práctica entretenida y rica en conocimientos, tiene un objetivo que se aleja mucho de los proyectos de prácticas realizados en el resto de las asignaturas de la carrera (excepto TP en el primer año), hemos aprendido sobre diversas tecnologías de la programación en general y en Java (Swing, Maven, Servidor de Persistencia, Git, Java Beans y JUnit). Es una práctica muy completa y entretenida de hacer pues al notar avances en el proyecto viendo como cada vez uno es capaz de implementar más opciones y que funciona uno se motiva y tiene más ganas de seguir trabajando.

De hecho, por nuestra situación en la que no fuimos ser capaces de realizar este proyecto durante el curso anterior, pero adquirimos los conocimientos necesarios para hacerlos, en cuanto nos pusimos a realizarla en nuestro tiempo libre la terminamos relativamente rápido ya que teníamos muchas ganas y nos entreteníamos.

Es difícil llevar una estimación concreta del tiempo que nos ha llevado, desde el repositorio de GitHub no es posible pues empezamos a usarlo más adelante ya que nuestra metodología de trabajo era en pareja durante todo momento, sólo usábamos un ordenador a la vez porque al ser compañeros de piso lo consideramos como la mejor opción. Calculamos que el tiempo total de trabajo dedicado puede rondar las 110 horas.

En conclusión, pensamos que la metodología de realizar esta práctica es muy buena y no se debería de perder esta dinámica con relación a los próximos cursos.