```
package br.com.optimedia.autor.view
{
    import br.com.optimedia.assets.CommandConstants;
    import br.com.optimedia.assets.NotificationConstants;
    import br.com.optimedia.assets.vo.MediaVO;
    import br.com.optimedia.assets.vo.QuestionVO;
    import br.com.optimedia.autor.model.RepositoryManagerProxy;
    import br.com.optimedia.autor.view.components.SendMediaPopUp;

    import flash.events.Event;

    import mx.events.CloseEvent;

    import org.puremvc.as3.multicore.interfaces.INotification;
    import org.puremvc.as3.multicore.patterns.mediator.Mediator;

    public class SendMediaPopUpMediator extends Mediator
    {
        public static const NAME:String = 'SendMediaMediator';

        private var repositoryProxy:RepositoryManagerProxy;

        public function SendMediaPopUpMediator(viewComponent:Object=null)
        {
            super(NAME+viewComponent.uid, viewComponent);
        }

        override public function onRegister():void
        {
            trace(NAME+".onRegister()");
            view.addEventListener(SendMediaPopUp.UPLOAD_FILE_EVENT, uploadMediaFile);
            view.addEventListener(SendMediaPopUp.UPLOAD_TEXT_EVENT, uploadMediaText);
            view.addEventListener(SendMediaPopUp.CREATION_QUESTION_EVENT, saveQuestion);
            view.addEventListener(CloseEvent.CLOSE, closeMe);

            repositoryProxy = facade.retrieveProxy( RepositoryManagerProxy.NAME ) as 
RepositoryManagerProxy;
        }

        override public function onRemove():void {
            view.removePopup();
        }

        public function get view():SendMediaPopUp
        {
            return viewComponent as SendMediaPopUp;
        }

        override public function listNotificationInterests():Array
        {
            return [NotificationConstants.UPLOAD_MEDIA_FILE_RESULT,
                    NotificationConstants.UPLOAD_MEDIA_TEXT_RESULT,
                    NotificationConstants.CREATION_QUESTION_RESULT];
        }
        override public function handleNotification(note:INotification):void
        {
            switch (note.getName())
            {
                case NotificationConstants.UPLOAD_MEDIA_FILE_RESULT:
                    repositoryProxy.getMedias(view.subjectID);
                    closeMe(null);
                    break;
                case NotificationConstants.UPLOAD_MEDIA_TEXT_RESULT:
                    repositoryProxy.getMedias(view.subjectID);
                    closeMe(null);
                    break;
                case NotificationConstants.CREATION_QUESTION_RESULT:
                    repositoryProxy.getMedias(view.subjectID);
                    closeMe(null);
                    break;
                default:
                    break;
            }
        }

        private function closeMe(event:CloseEvent):void {
            sendNotification( CommandConstants.SEND_MEDIA_POPUP_DISPOSE, view );
        }

        private function uploadMediaFile(event:Event):void {
            var media:MediaVO = new MediaVO();
            media.category_id = view.category;
            media.title = view.nameTextInput.text;
            repositoryProxy.uploadMediaFile(view.fileVO, media, view.subjectID);
        }

        private function uploadMediaText(event:Event):void {
            var media:MediaVO = new MediaVO();
            media.category_id = view.category;
            media.title = view.nameTextInput.text;
            if( view.currentState == 'text' ) media.body = view.textArea.text;
            else media.body = view.fileTextInput.text;
            repositoryProxy.uploadMediaText( media, view.subjectID );
        }
        private function saveQuestion(event:Event):void {
            var question:QuestionVO = new QuestionVO();
            question=view.questionVO;

            var media:MediaVO = new MediaVO();
            media.category_id = view.category;
            media.title = view.nameTextInput.text;
            media.body = question.title;

            repositoryProxy.saveQuestion(media, question, view.subjectID);
        }
    }
}
```