Laboratory Activity #05

Distributed Systems Programming

Daniele Bringhenti, Francesco Pizzato

MQTT Features

- MQTT (Message Queuing Telemetry Transport) is a standard client-server **publish-subscribe** messaging transport protocol, usually based on TCP.
- The main features of MQTT are:
  1) **simplicity** (low requirements of processing or battery power)
  2) **efficiency** (lightweight transport);
  3) **scalability** (millions of devices);
  4) **reliability** (support for unreliable networks).

MQTT Features

- MQTT (Message Queuing Telemetry Transport) is a standard client-server **publish-subscribe** messaging transport protocol, usually based on TCP.
- The main features of MQTT are:
  1) **simplicity** (low requirements of processing or battery power)
  2) **efficiency** (lightweight transport);
  3) **scalability** (millions of devices);
  4) **reliability** (support for unreliable networks).

MQTT is suitable for **Internet-of-Things M2M** communications.

Topics of the Laboratory Session

Laboratory Session #05 covers the following activities:

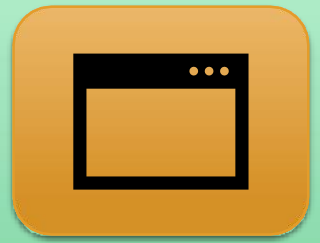
Integration of **MQTT** functionalities in the implementation of the React client

Integration of **MQTT** functionalities in the implementation of the Film Manager service

MQTT

Topics of the Laboratory Session

Laboratory Session #05 covers the following activities:



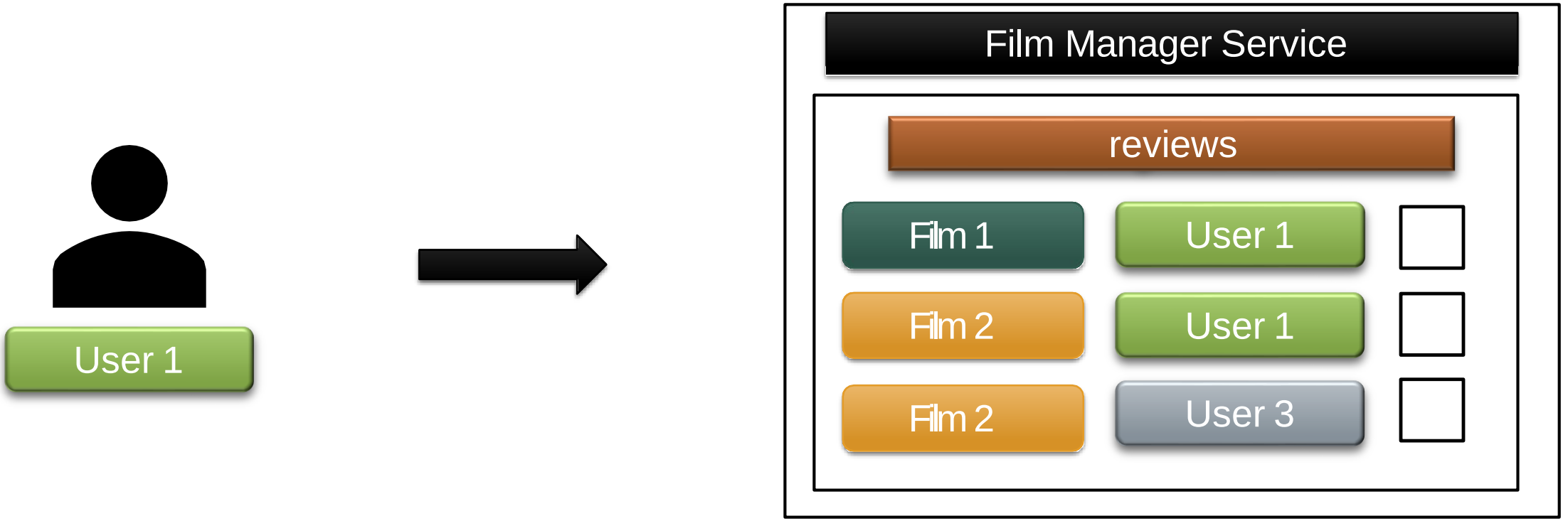
Integration of **MQTT** functionalities in the implementation of the React client

Integration of **MQTT** functionalities in the implementation of the Film Manager service

MQTT

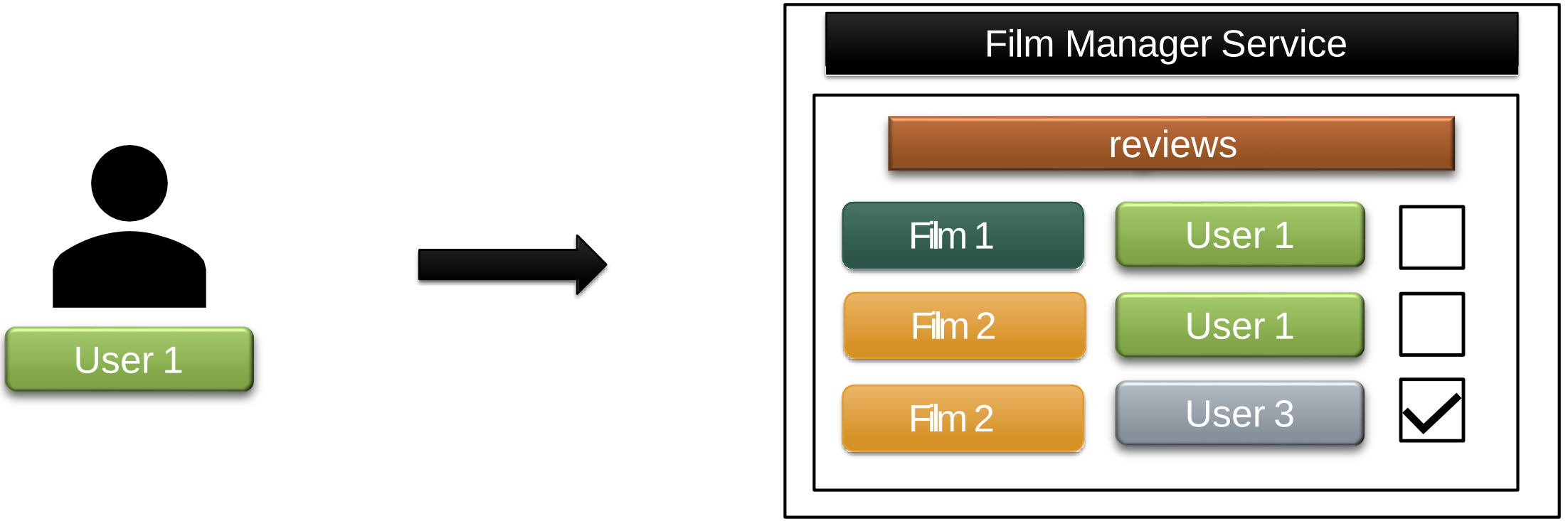
Restriction of the **film selection** operation, with impact on **both** the Film Manager and the React client

Film Selection in the Film Manager service



- A public film can be the active film for **at most one** user at a time.
- In case a user tries to select a film which is active for another user, the operation **fails**.

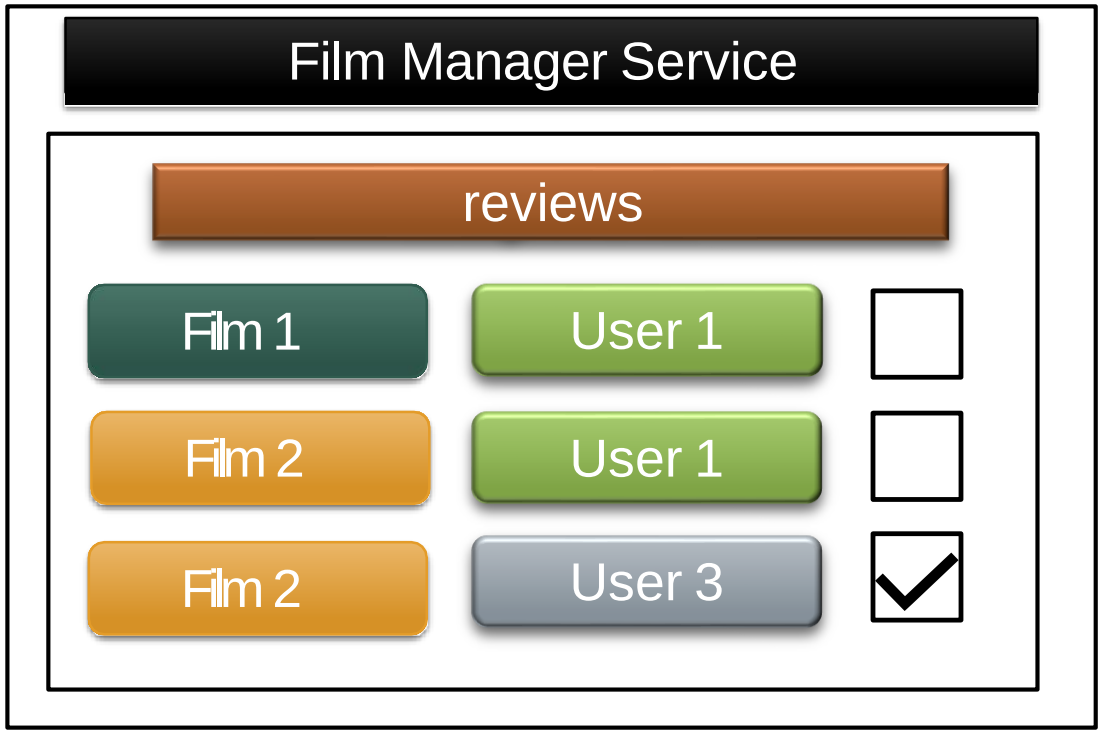
Film Selection in the Film Manager service



- A public film can be the active film for **at most one** user at a time.
- In case a user tries to select a film which is active for another user, the operation **fails**.

Film Selection in the Film Manager service

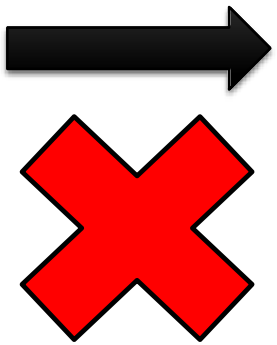
SELECT Film 2

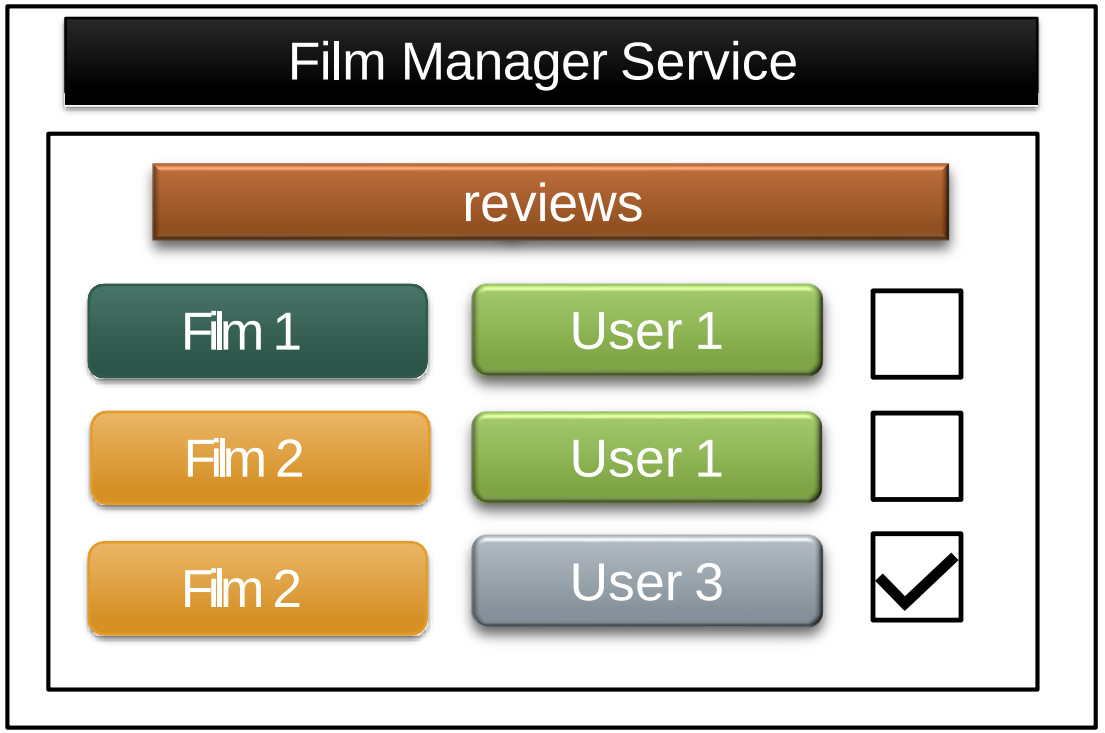


- A public film can be the active film for **at most one** user at a time.
- In case a user tries to select a film which is active for another user, the operation **fails**.

Film Selection in the Film Manager service

SELECT Film 2





- A public film can be the active film for **at most one** user at a time.
- In case a user tries to select a film which is active for another user, the operation **fails**.

Film Selection in the React client

- This new constraint demands for a **synchronization** among clients:
  - **eventual consistency** is acceptable;
  - at the end, all clients will **agree** about film selections.

- When a user tries to select a film, there are **two** options:
  1. the selection remains in a **pending state** until a confirmation or refusal of the selection comes from the server;
  2. the selection appears immediately as active to the user, but if it is later refused by the server, it is undone (**optimistic approach**).

  In both cases, the user must be informed about a failed selection with an **alert**.

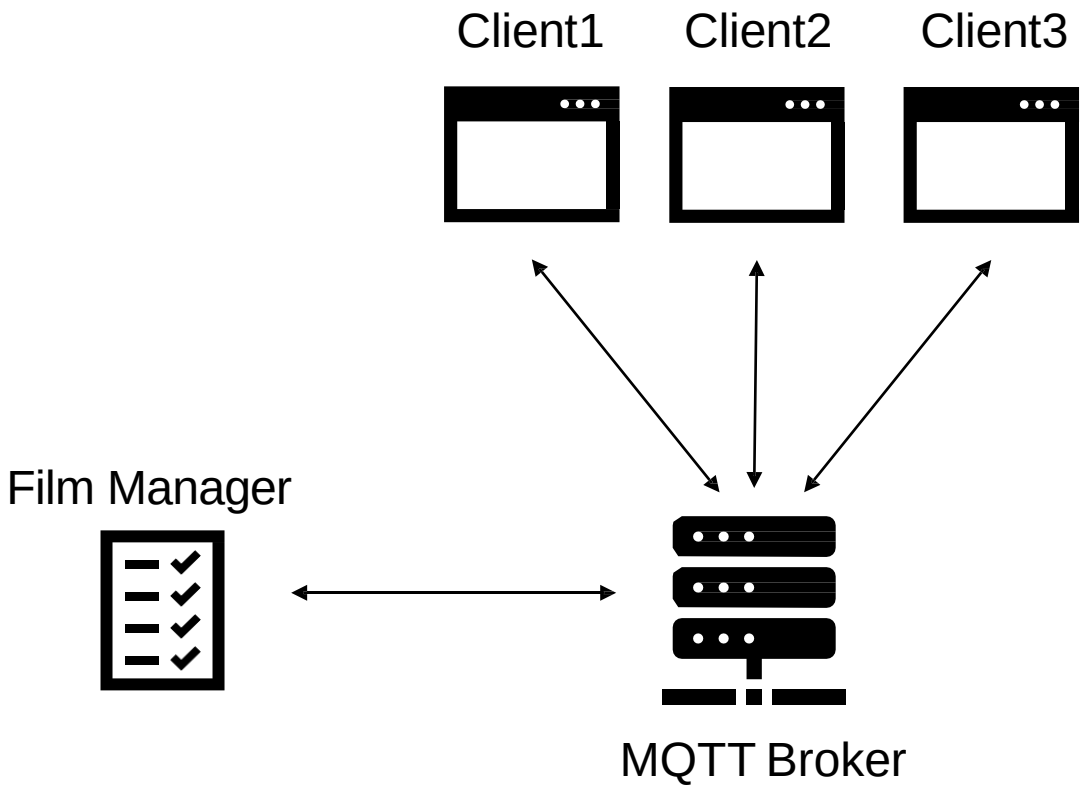
MQTT Communication

- Both the **Film Manager** (FM) service and the **React** client are extended with the functionality to communicate by using MQTT:
  - Film Manager **publishes** MQTT messages;
  - the React client **subscribes** to topics and **receives** MQTT messages.

- The MQTT broker is **Eclipse Mosquitto**:
  - launch Mosquitto with the following command, based on the **mosquitto.conf** file you have been provided with:

```
mosquitto –v –c mosquitto.conf
```

MQTT communication (FM - initial situation)



films

id
1
2
3
4

users

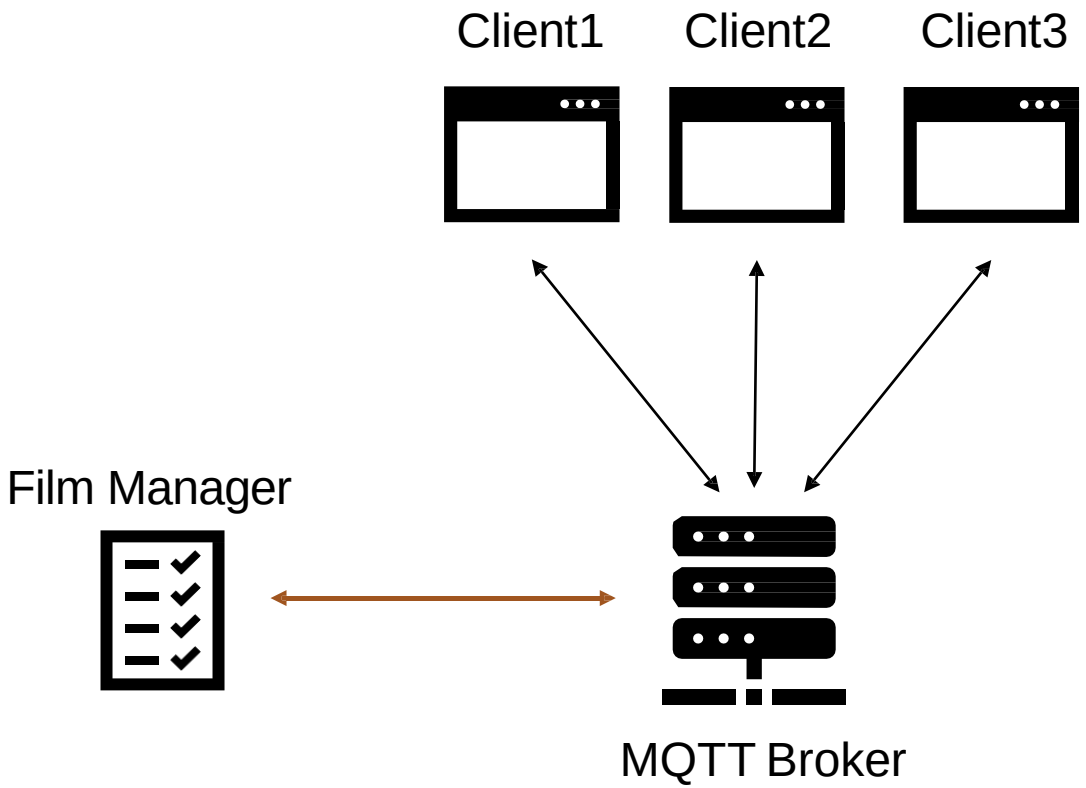
id	name
1	User
2	Frank
3	Karen
4	Rene

reviews

film	user	active
1	2	1
2	2	0
2	3	0
3	4	1

These are public films!

MQTT communication (FM – connection establishment)



films

id
1
2
3
4

users

id	name
1	User
2	Frank
3	Karen
4	Rene

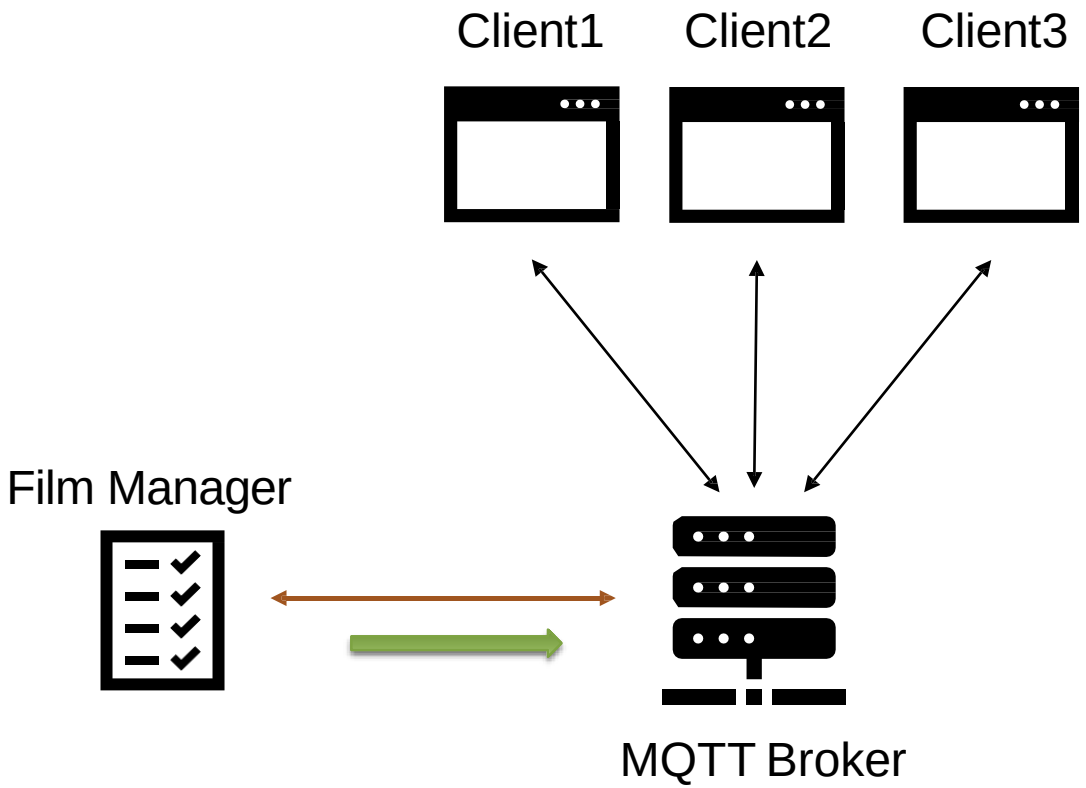
reviews

film	user	active
1	2	1
2	2	0
2	3	0
3	4	1

After the Film Manager service successfully establishes a connection with the broker:

- it **publishes** a message for each existing public film;
- each message must have the **retained** flag set to true.

MQTT communication (FM – connection establishment)



films

id
1
2
3
4

users

id	name
1	User
2	Frank
3	Karen
4	Rene

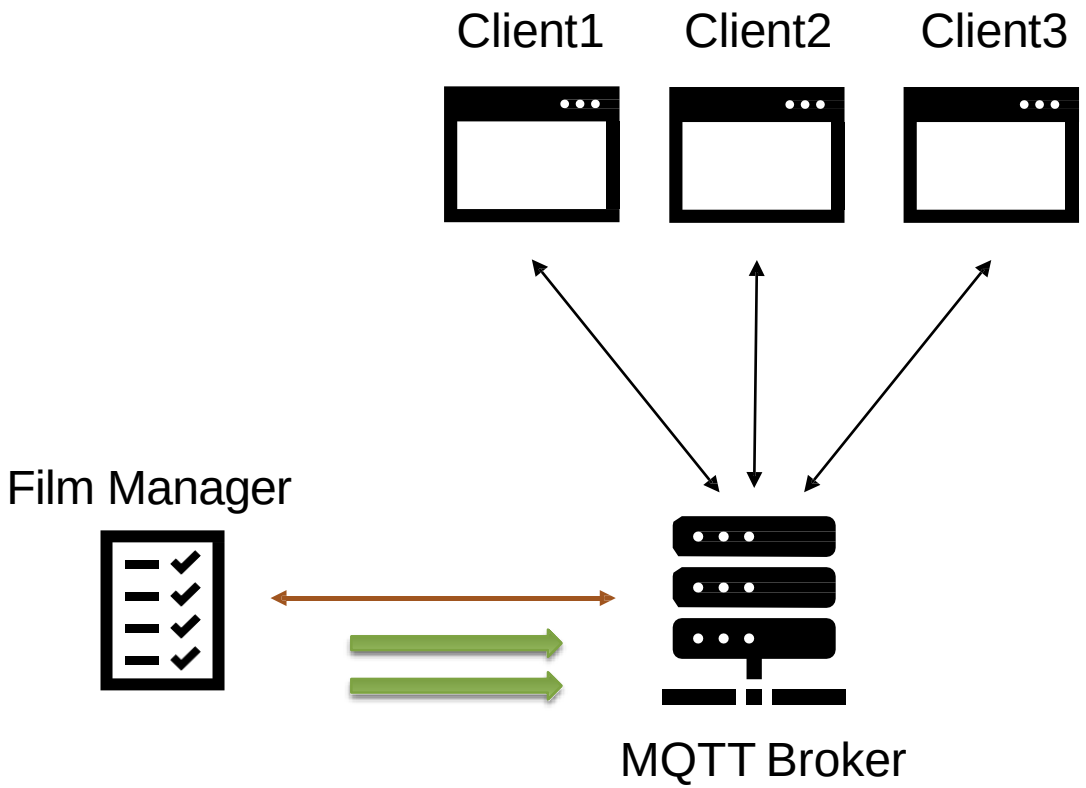
reviews

film	user	active
1	2	1
2	2	0
2	3	0
3	4	1

Topic: “1”

```
{
        “status” : “active”,
        “userId:  “2”,
        “userName”:  “Frank”
}
```

MQTT communication (FM – connection establishment)



films

id
1
2
3
4

users

id	name
1	User
2	Frank
3	Karen
4	Rene

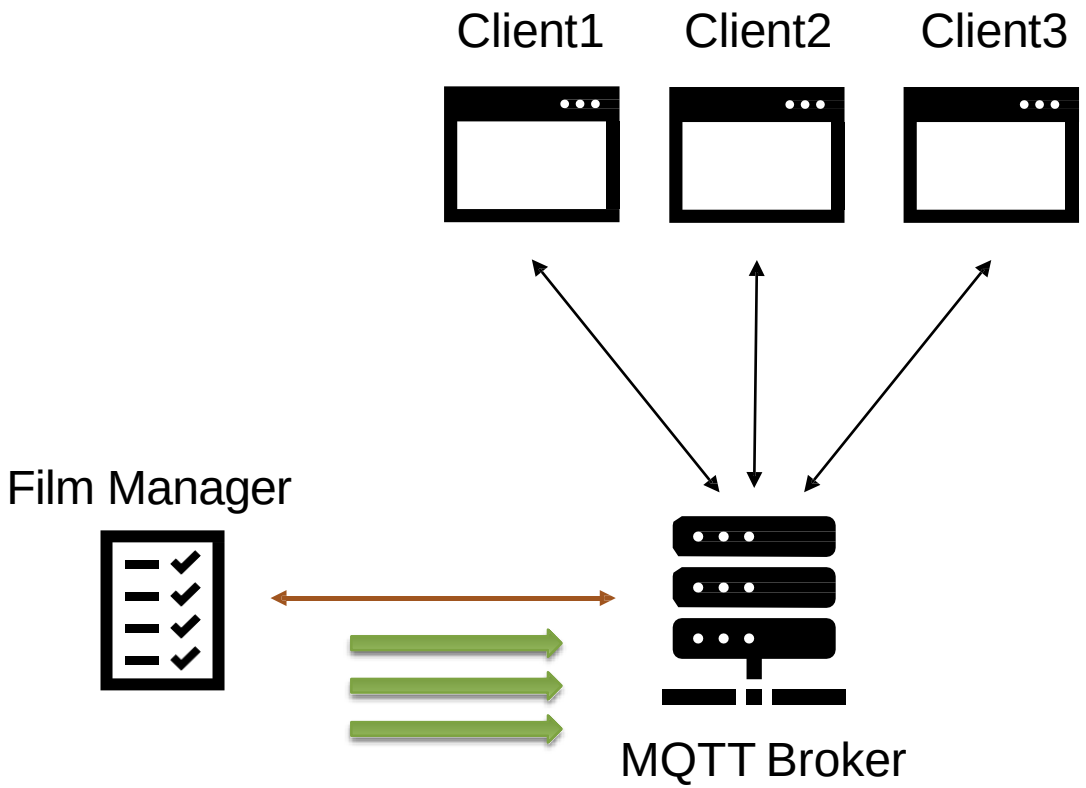
reviews

film	user	active
1	2	1
2	2	0
2	3	0
3	4	1

Topic: “2”

```
{
        “status” : “inactive”
}
```

MQTT communication (FM – connection establishment)



films

id
1
2
3
4

users

id	name
1	User
2	Frank
3	Karen
4	Rene

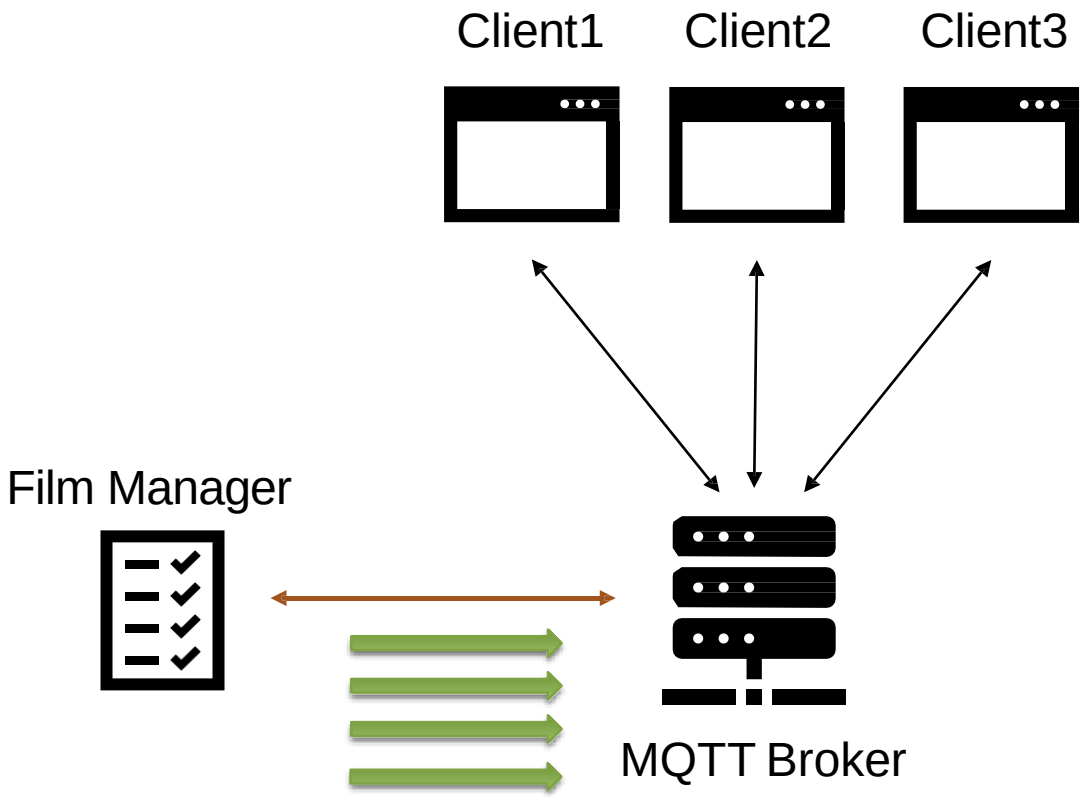
reviews

film	user	active
1	2	1
2	2	0
2	3	0
3	4	1

Topic: “3”

```
{
        “status” : “active”,
        “userId:  “4”,
        “userName”:  “Rene”
}
```

MQTT communication (FM – connection establishment)



films

id
1
2
3
4

users

id	name
1	User
2	Frank
3	Karen
4	Rene

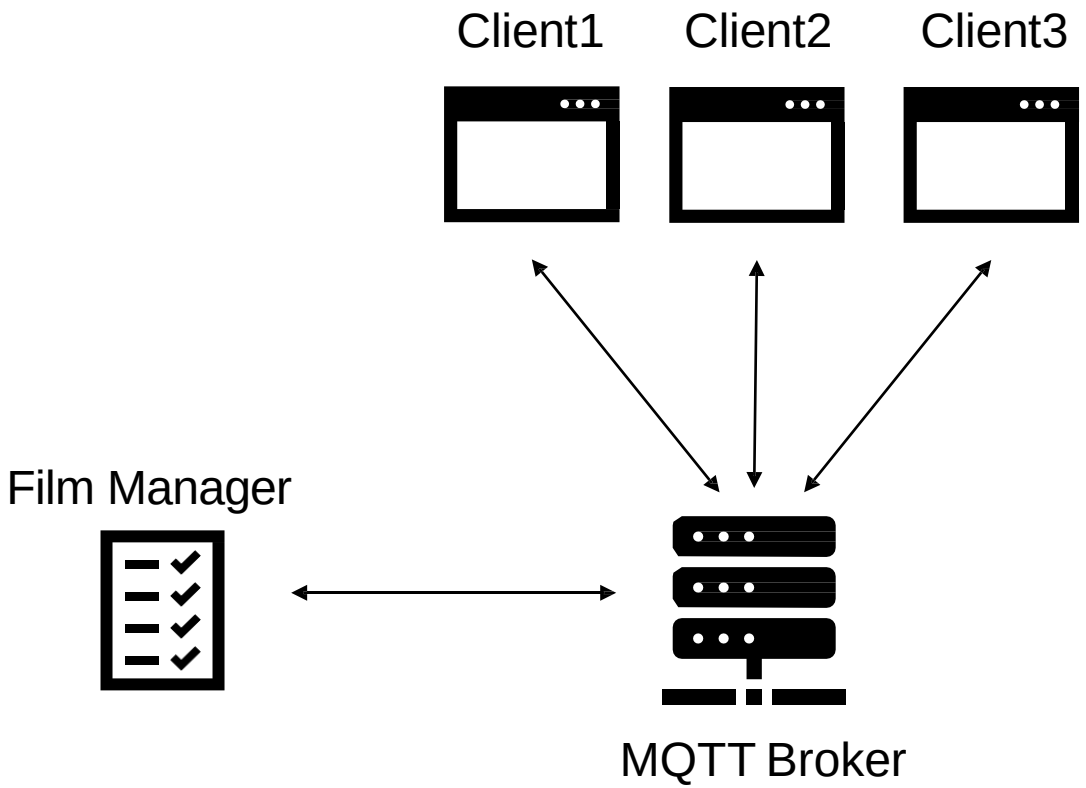
reviews

film	user	active
1	2	1
2	2	0
2	3	0
3	4	1

Topic: “4”

```
{
        “status” : “inactive”
}
```

MQTT communication (FM – film selection)



films

id
1
2
3
4

users

id	name
1	User
2	Frank
3	Karen
4	Rene

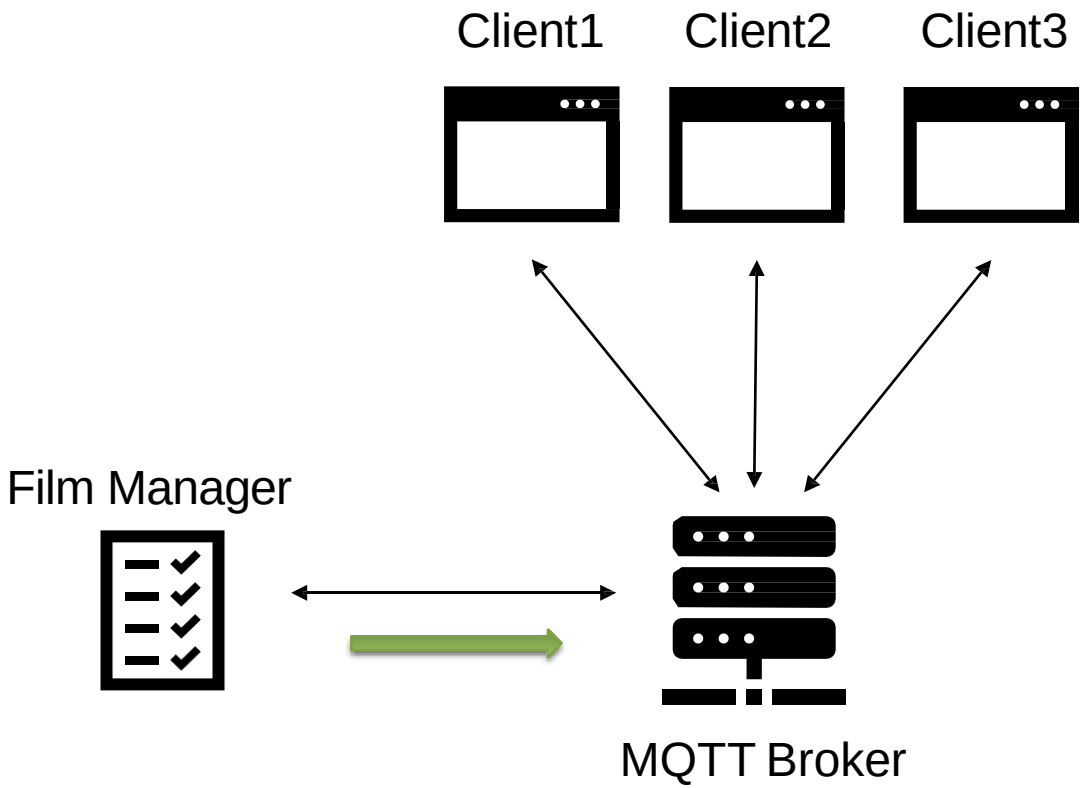
reviews

film	user	active
1	2	**0**
2	2	**1**
2	3	0
3	4	1

When a film becomes **active** for a different user, the Film Manager publishes a retained message, conveying:

- the **active** status of that film;
- the **id** and **name** of the user who selected it.

MQTT communication (FM – film selection)



films

id
1
2
3
4

users

id	name
1	User
2	Frank
3	Karen
4	Rene

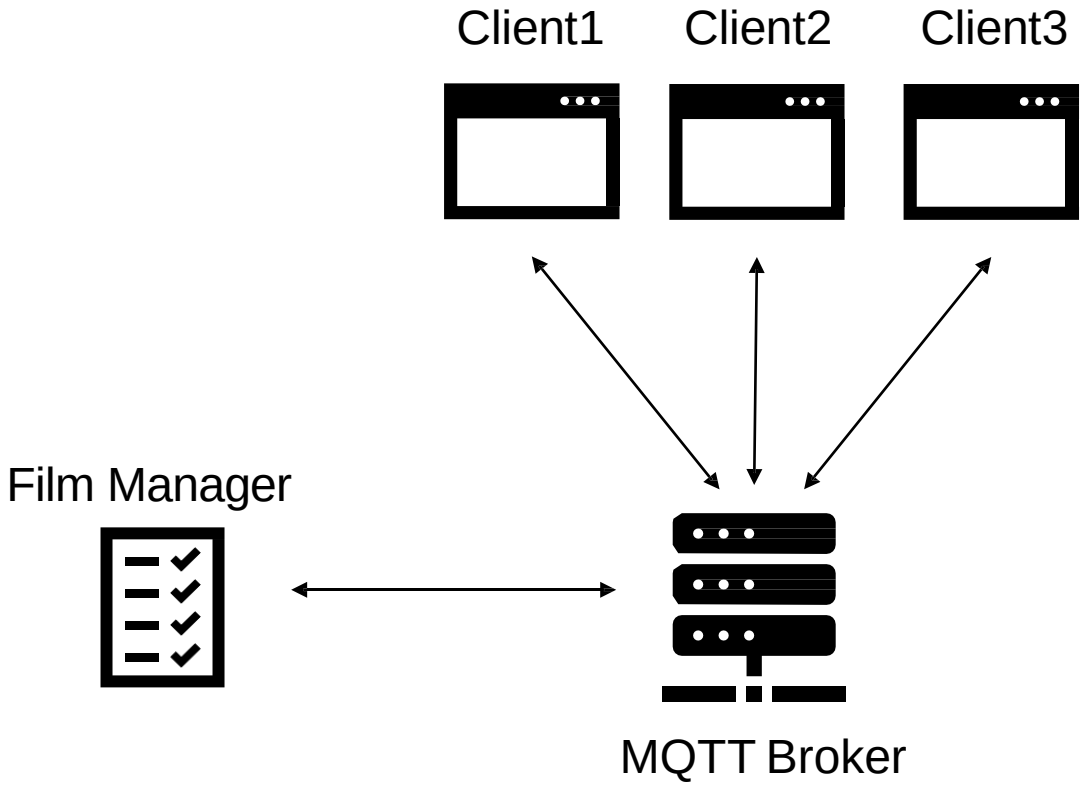
reviews

film	user	active
1	2	**0**
2	2	**1**
2	3	0
3	4	1

Topic: “2”

```
{
        “status” : “active”,
        “userId:  “2”,
        “userName”:  “Frank”
}
```

MQTT communication (FM – film “de”selection)



films

id
1
2
3
4

users

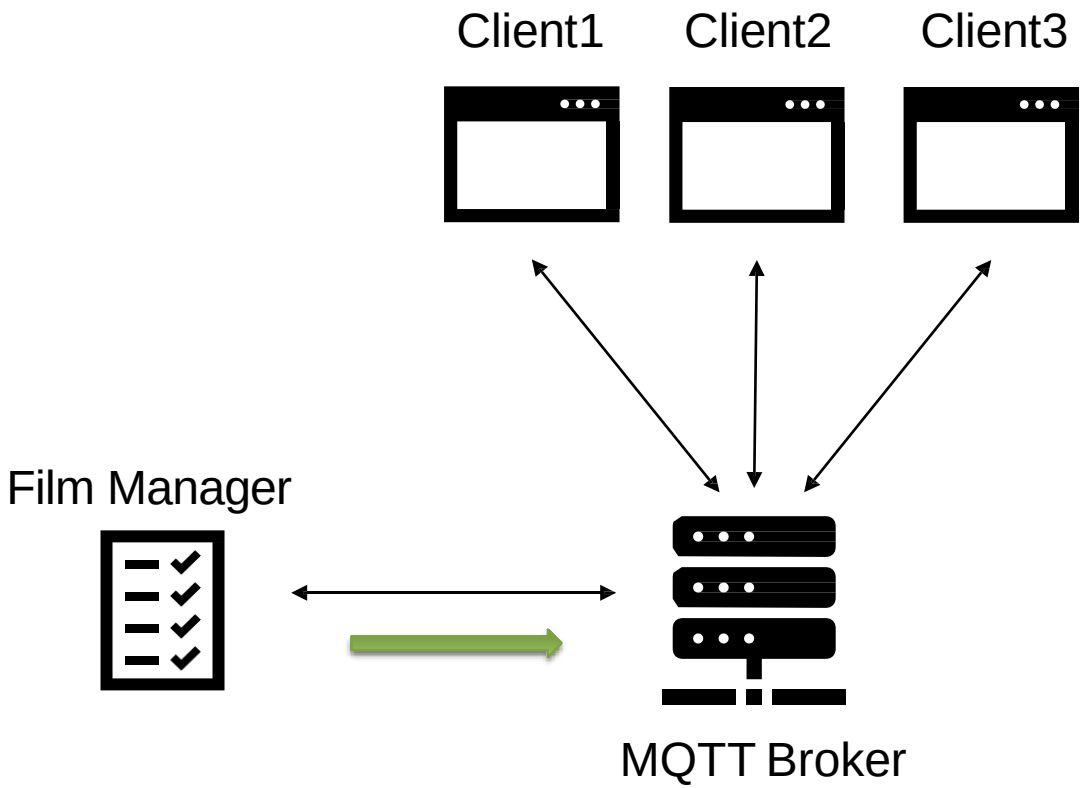
id	name
1	User
2	Frank
3	Karen
4	Rene

reviews

film	user	active
1	2	**0**
2	2	**1**
2	3	0
3	4	1

When a film is **not active** anymore for any user, the Film Manager service publishes a retained message, conveying the **inactive** status of that film.

MQTT communication (FM – film “de”selection)



films

id
1
2
3
4

users

id	name
1	User
2	Frank
3	Karen
4	Rene

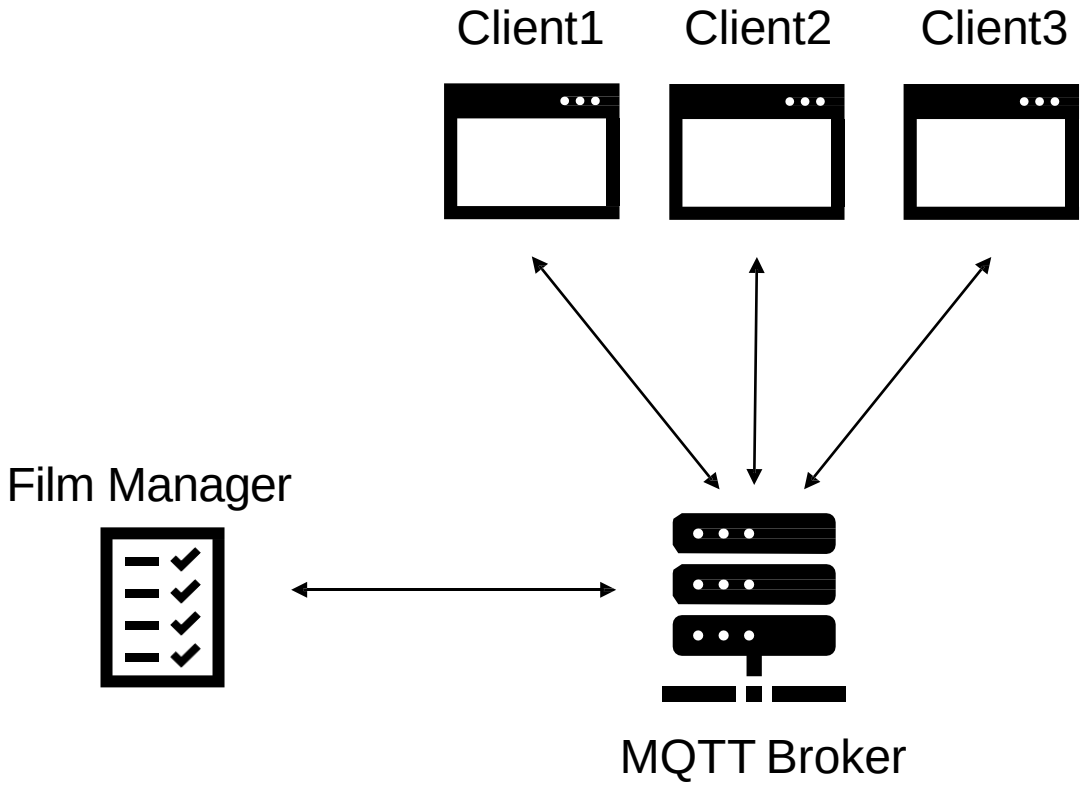
reviews

film	user	active
1	2	**0**
2	2	**1**
2	3	0
3	4	1

Topic: “1”

```
{
        “status” : “inactive”
}
```

MQTT communication (FM – film creation)



films

id
1
2
3
4
5

users

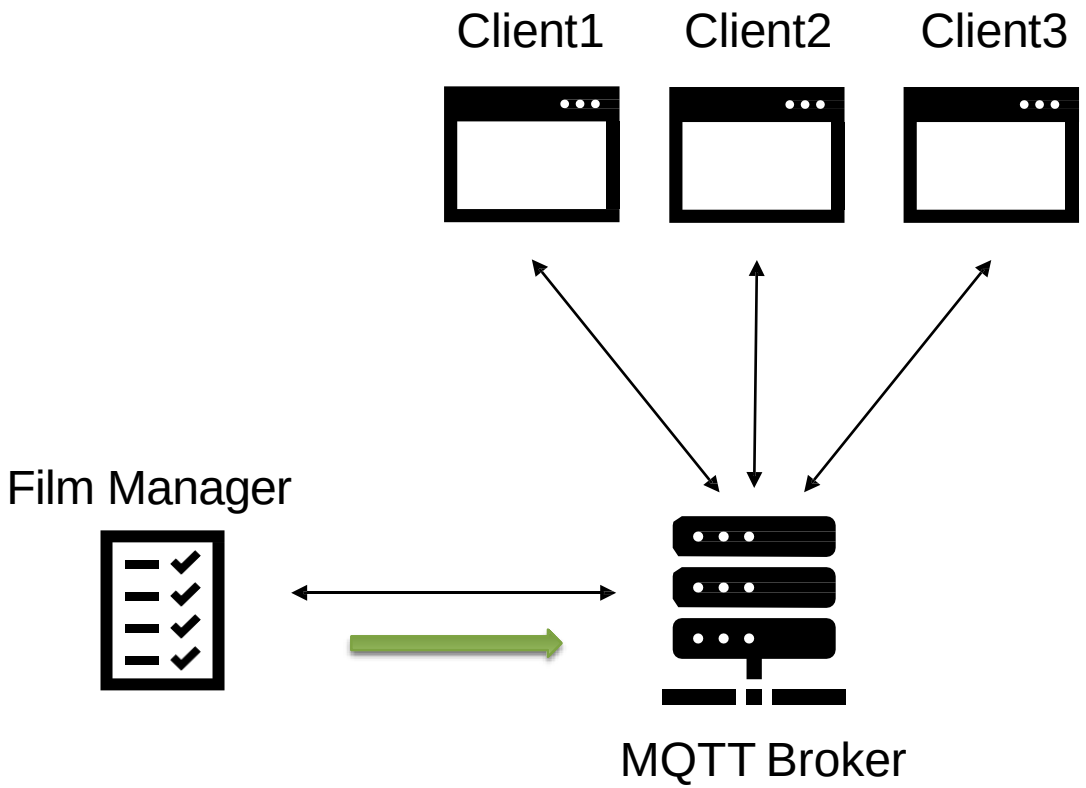
id	name
1	User
2	Frank
3	Karen
4	Rene

reviews

film	user	active
1	2	0
2	2	1
2	3	0
3	4	1

When a public film is **created**, the Film Manager service publishes a retained message, conveying the **inactive** status of that film.

MQTT communication (FM – film creation)



films

id
1
2
3
4
5

users

id	name
1	User
2	Frank
3	Karen
4	Rene

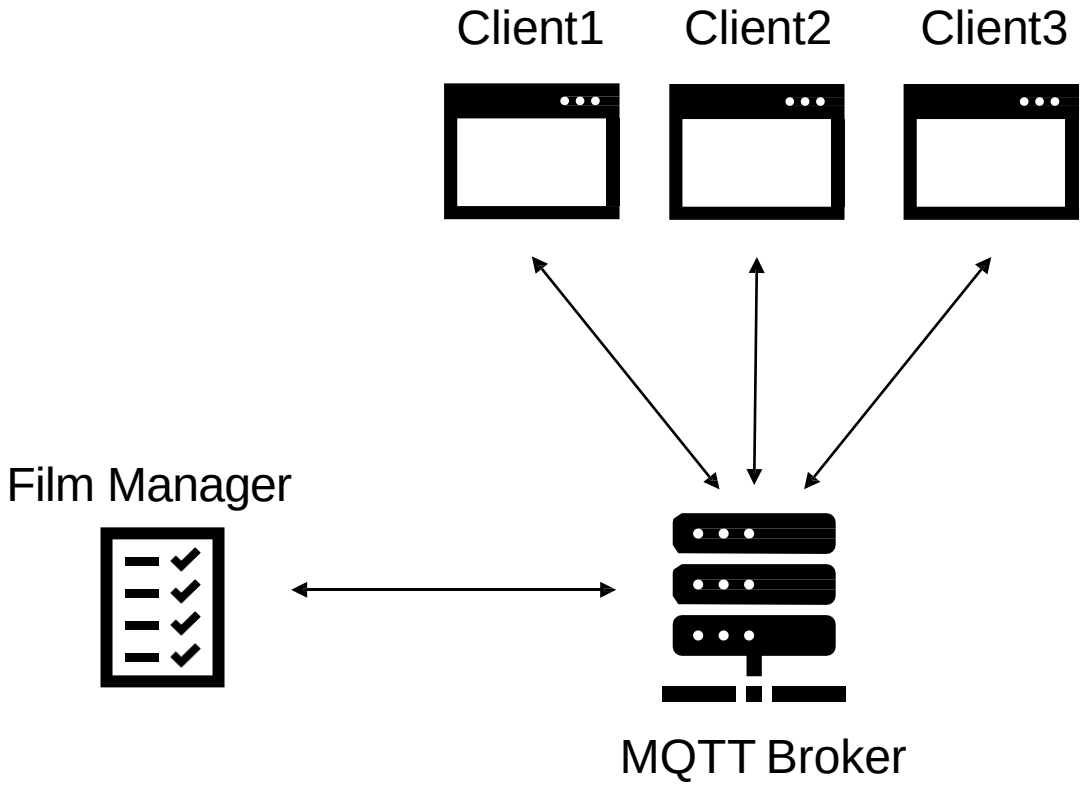
reviews

film	user	active
1	2	0
2	2	1
2	3	0
3	4	1

Topic: “5”

```
{
        “status” : “inactive”
}
```

MQTT communication (FM – film deletion)



films

id
1
2
3
--
5

users

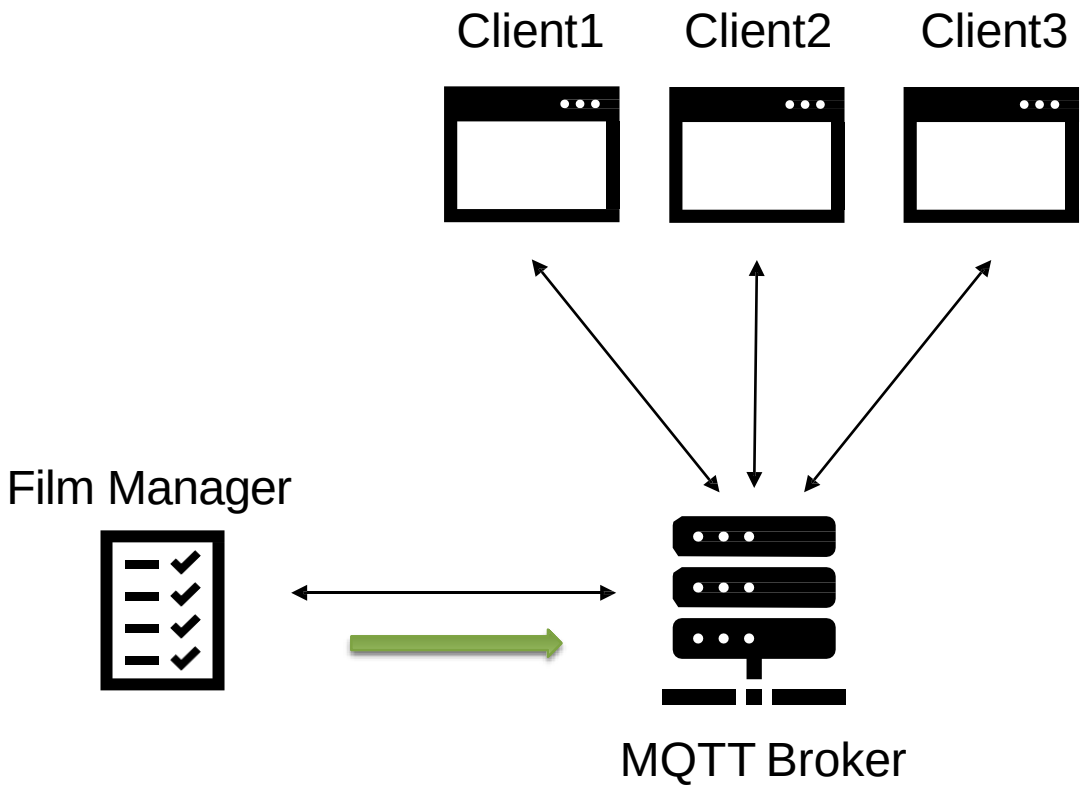
id	name
1	User
2	Frank
3	Karen
4	Rene

reviews

film	user	active
1	2	0
2	2	1
2	3	0
3	4	1

When a public film is **deleted**, the Film Manager service publishes a retained message informing the subscribed clients about this event (i.e., with the film status set to **deleted**).

MQTT communication (FM – film deletion)



films

id
1
2
3
--
5

users

id	name
1	User
2	Frank
3	Karen
4	Rene

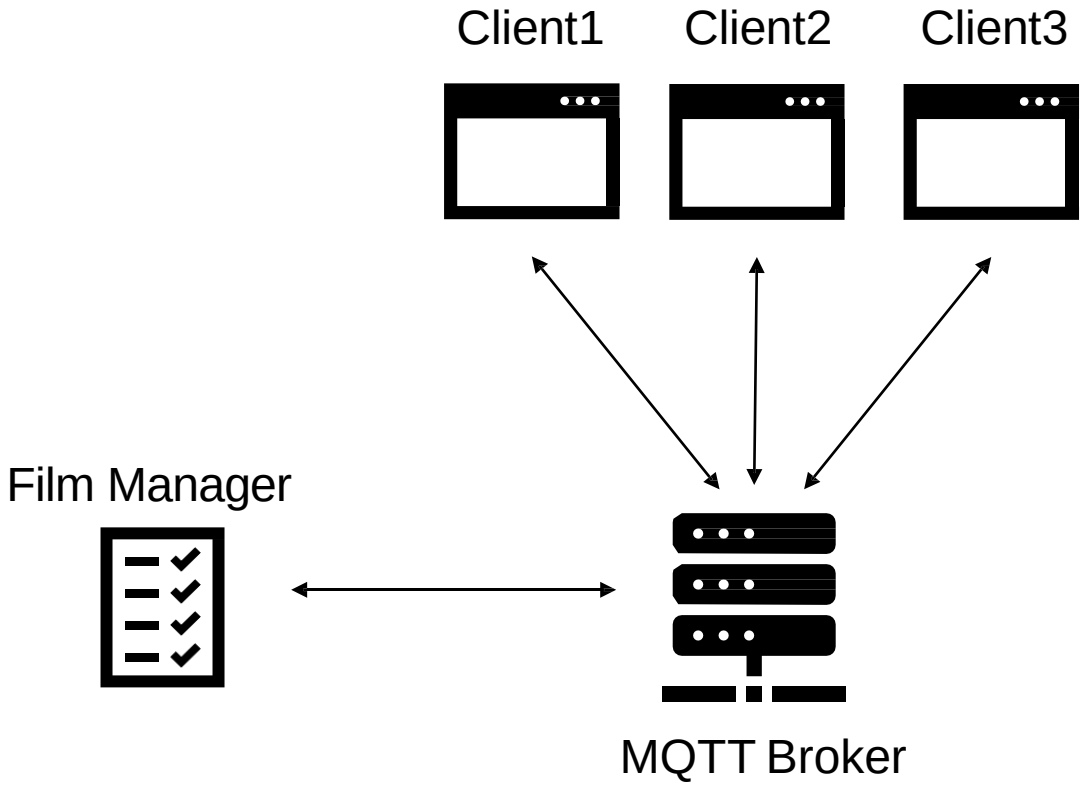
reviews

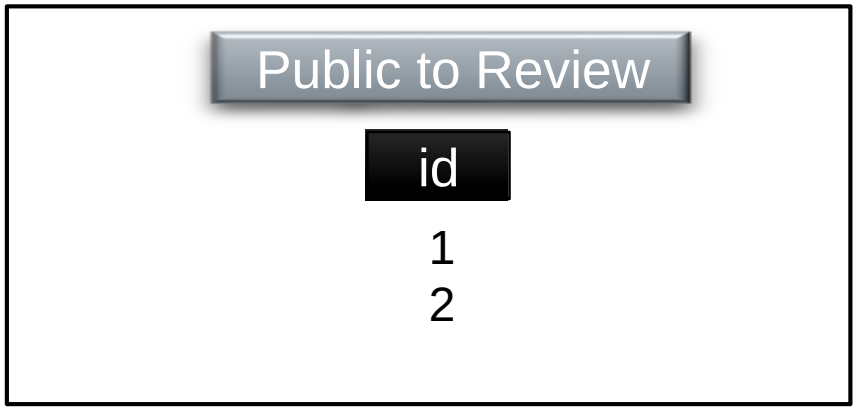
film	user	active
1	2	0
2	2	1
2	3	0
3	4	1

Topic: “4”

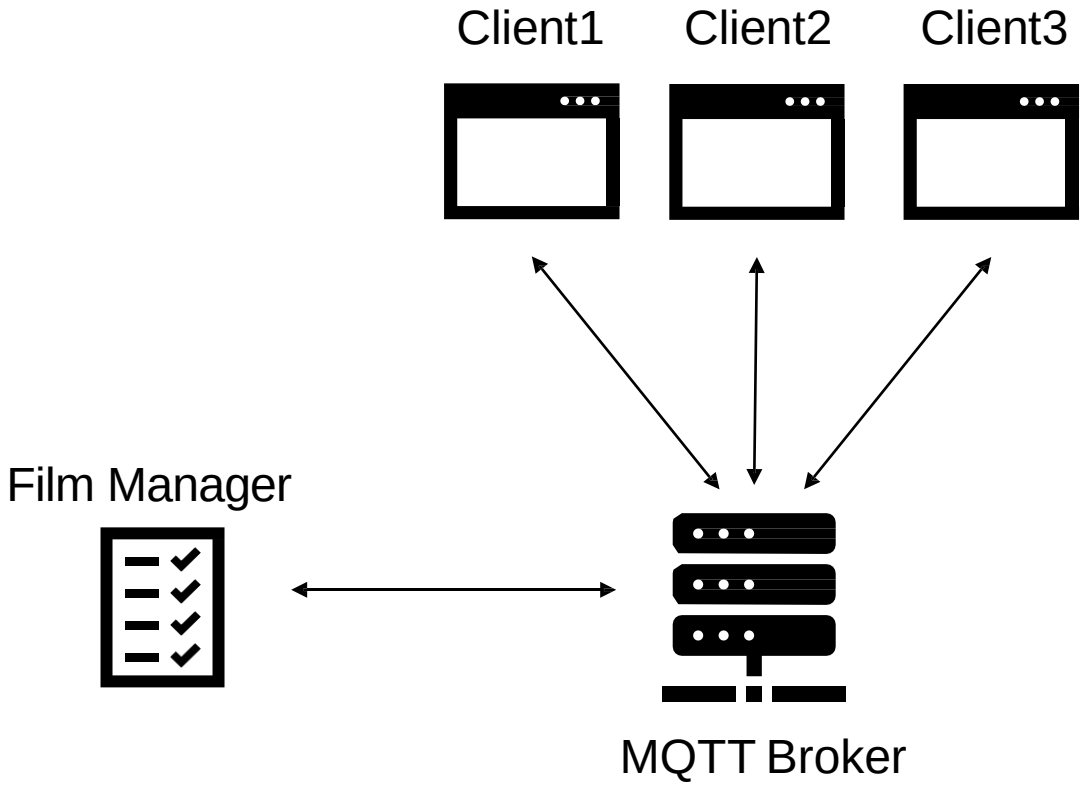
```
{
        “status” : “deleted”
}
```

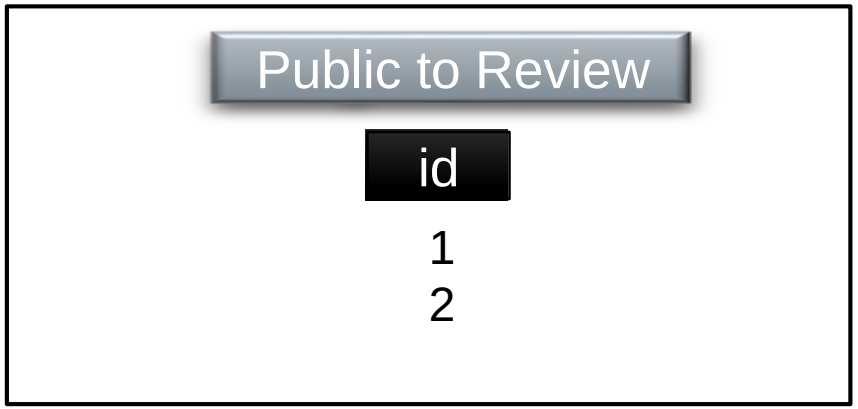
MQTT communication (Client - initial situation)





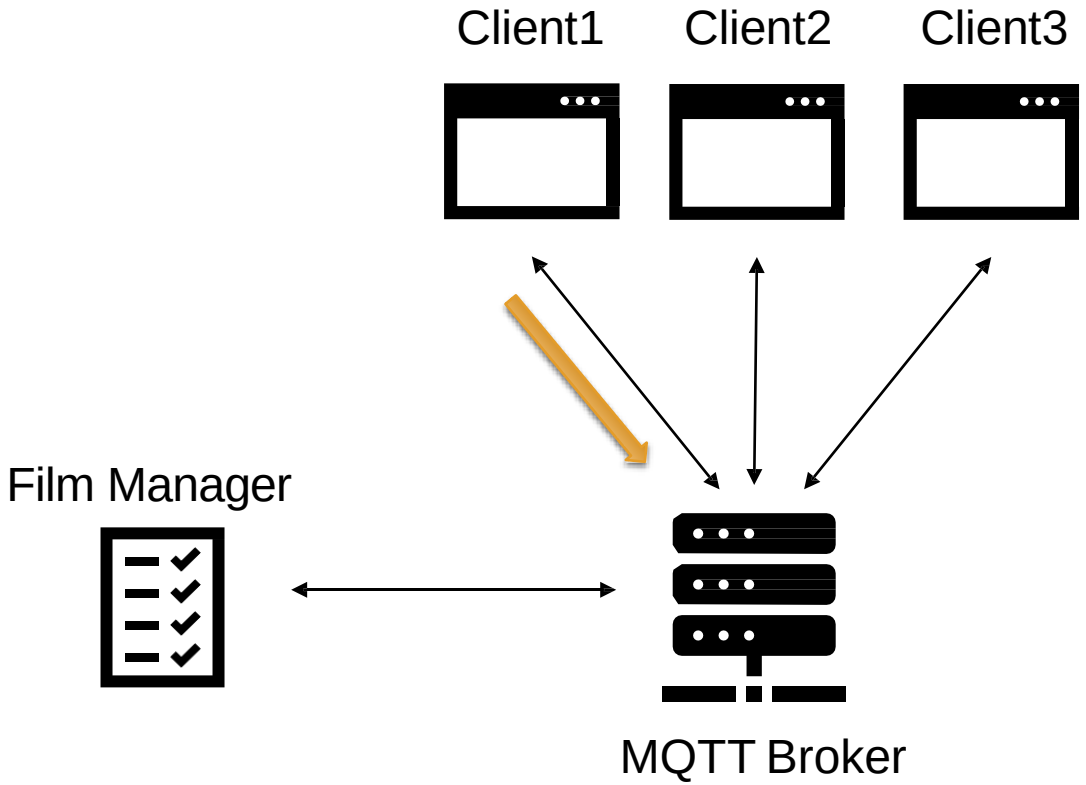
MQTT communication (Client - subscription)

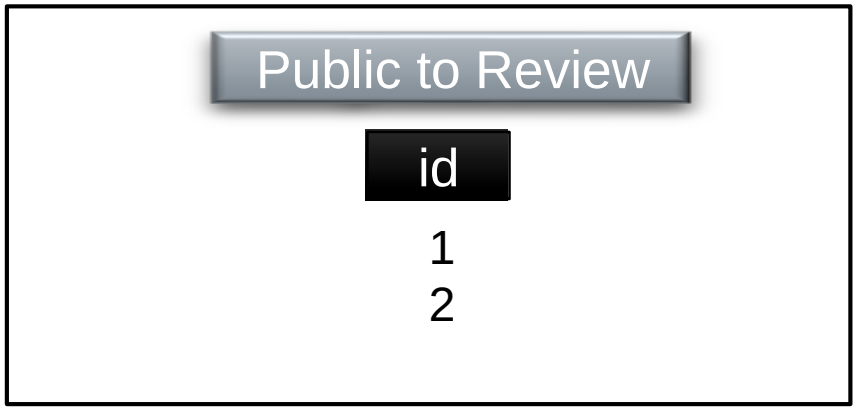




The React client must **subscribe** to topics corresponding to the **id** of each public film for which a review request has been issued to the user and which is **currently** shown in the *Public to Review* page of the GUI.

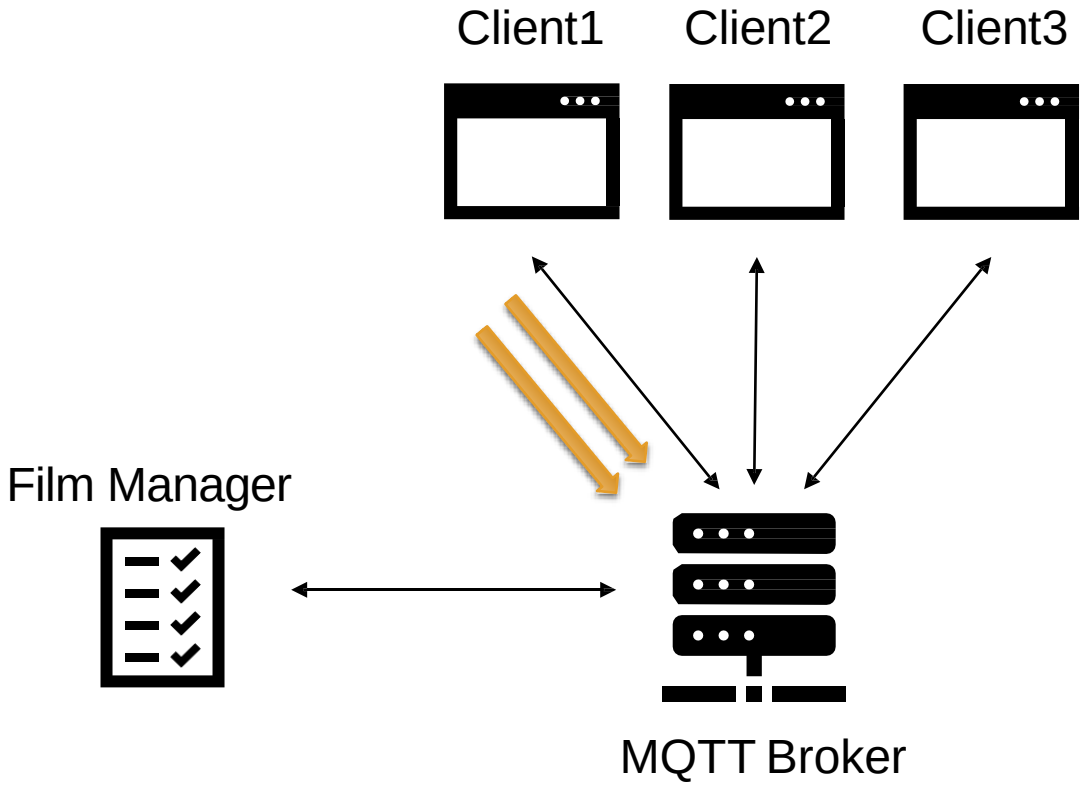
MQTT communication (Client - subscription)

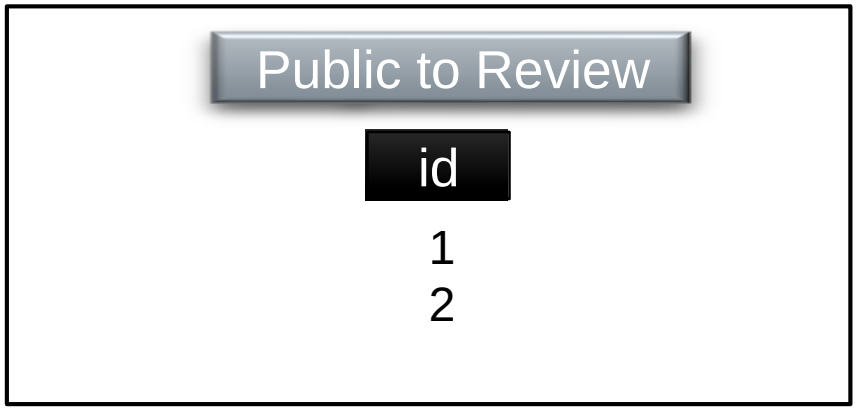




Subscribe to topic: “1”

MQTT communication (Client - subscription)





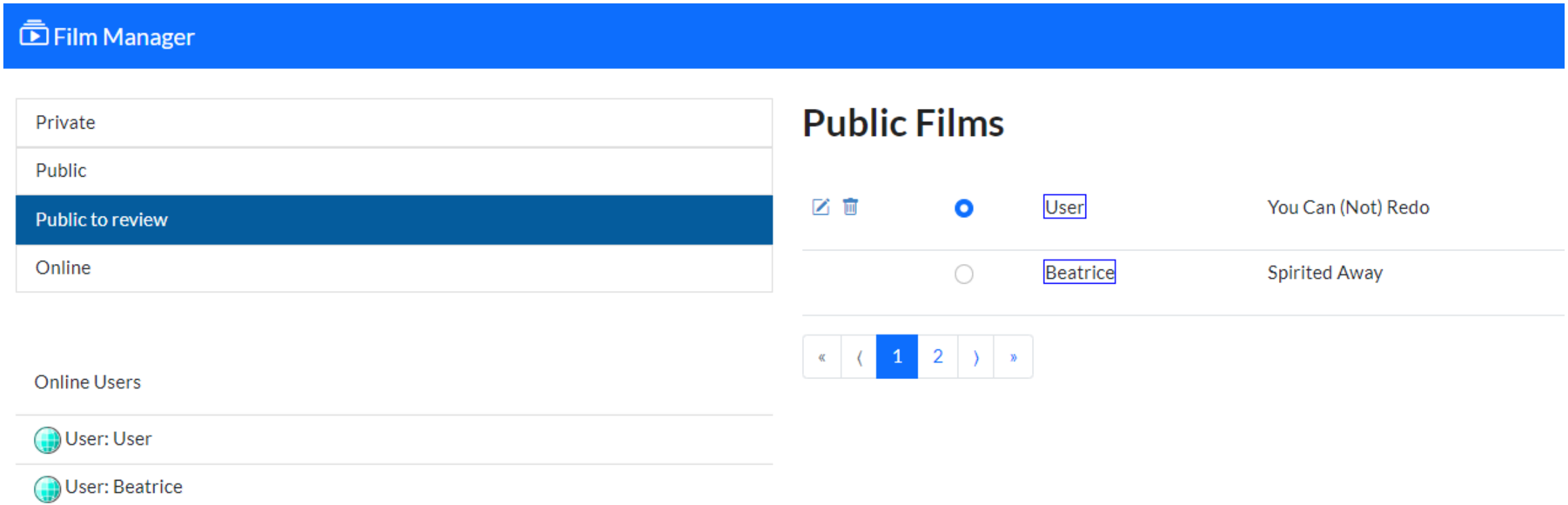
Subscribe to topic: “2”

How does the React client *react*?

- The React client may **receive** MQTT messages for the topic to which there exists a subscription:
  - after the subscription to each topic, retained messages are received;
  - after each status change for film selection, a new message is received.

- The React client **reacts** in the following ways:
  1. whenever the React client receives an MQTT message related to a film, it updates the status of the film in the **Public to Review** page of the GUI;
  2. whenever the selection of a film performed by the logged-in user **fails**, an **alert** message should be shown on the screen.

How does the React client *react*?



Final Tips

- For the communication with a web browser, MQTT messages must be encapsulated into WebSocket frames (**MQTT Over Websockets**):
  - the URL to be specified for the MQTT connection is **ws**://127.0.0.1:8080.

- For the reaction of the *React* client, you only need to use this line of code in **App.jsx**:

  displayFilmSelection(*topic*, *parsedMessage*);

  where:
  - *topic* is a string representing the id of the film;
  - *parsedMessage* is the JSON object retrieved after parsing the MQTT message.

Thanks for your attention!

D. Bringhenti, F. Pizzato

daniele.bringhenti@polito.it

francesco.pizzato@polito.it