POLITÉCNICO DE LEIRIA

Relatório do Projeto

Serviços e Interoperabilidade de Sistemas (SIS)

TeSP em Programação de Sistemas de Informação (2025/26)

CineLive

Componente de *API REST* e *Messaging*

— PL2 • Grupo A —

Eduardo Carvalho (2024147217)

Diego Teixeira (2024118127)

Leiria, janeiro de 2026

Índice

Contextualização

O projeto **CineLive** consiste num sistema de gestão de cinemas, filmes, sessões e bilhetes, desenvolvido no âmbito da unidade curricular de Projeto em Sistemas de Informação (**PSI**) que integra as UCs de Plataformas de Sistemas de Informação (**PLSI**), Acesso Móvel a Sistemas de Informação (**AMSI**) e Serviços e Interoperabilidade de Sistemas (**SIS**) do curso TeSP em Programação de Sistemas de Informação.

A solução é constituída por uma plataforma web, com recurso à *framework* **Yii2** de PHP, dividida em *front-office* e *back-office*, e uma aplicação móvel Android desenvolvida em Java. O *front-office* e a aplicação móvel suportam a consulta de filmes, a compra de bilhetes e o aluguer de salas, enquanto o *back-office* é responsável pela gestão de filmes, cinemas, salas e sessões de cada cinema.

No contexto da UC de **SIS**, o foco do projeto incide no desenvolvimento de uma **API RESTful *stateless***, responsável por disponibilizar os dados e as operações necessárias ao funcionamento da aplicação Android através do modelo ***request/response***. A API permite à aplicação Android **comunicar com o servidor**, assegurando todas as **operações CRUD** necessárias ao cliente, além da realização de operações personalizadas (*custom actions*), alinhadas com os requisitos funcionais da aplicação.

A aplicação Android necessita, de uma forma geral, de aceder às informações relacionadas com cinemas, filmes, sessões, compras e bilhetes, bem como de realizar operações como **autenticação** e **autorização** dos utilizadores. Deste modo, a API foi estruturada em vários controladores, cada um responsável pela gestão de um conjunto específico de recursos e respetivas relações, incluindo situações de ***master/detail*** entre entidades, como filmes e sessões ou compras e bilhetes.

Para além da comunicação baseada em pedidos **HTTP**, o projeto integra também funcionalidades de ***messaging*** no modelo ***publish/subscribe***, permitindo o envio de mensagens de forma **assíncrona** relativas a eventos ou ações relevantes, tais como a criação de novas compras ou adição de filmes ao catálogo.

Desta forma, a componente **SIS** do projeto **CineLive** demonstra a aplicação prática dos conceitos abordados nas aulas, nomeadamente o desenvolvimento de **APIs RESTful** e a comunicação assíncrona através de **MQTT**, assegurando uma integração eficaz entre a aplicação Android e os serviços do sistema.

Endpoints da API

A API RESTful do projeto **CineLive** encontra-se acessível através do seguinte URL:

Servidor de produção	http://172.22.21.212/CineLive/Web/backend/web/api
Desenvolvimento local	http://localhost/CineLive/Web/backend/web/api

A API disponibiliza ***endpoints* de acesso público**, nomeadamente para a consulta de cinemas, filmes e sessões, além de ***endpoints* protegidos**, que requerem autenticação.

O acesso a *endpoints* protegidos é realizado através de um ***access-token***, obtido através do *endpoint* **/auth**, podendo a autenticação ser efetuada recorrendo a **QueryParamAuth** ou **HttpBearerAuth**, como já mencionado, a API é ***stateless***, ou seja, não guarda sessão.

No desenvolvimento da API, optou-se maioritariamente pela utilização de controladores baseados em **yii\rest\Controller (RestController)**, em vez do **ActiveController**, uma vez que muitas das operações disponibilizadas requerem lógica personalizada, validações específicas e respostas adaptadas às necessidades da aplicação Android.

A utilização de **ActiveController** implicaria a desativação (*unset*) de várias ações CRUD geradas pelo **Yii2**, uma vez que nem todas fazem sentido na sua forma padrão.

Esta abordagem permite um maior controlo sobre as operações da API, garantindo que apenas as ações necessárias estão disponíveis, mantendo uma estrutura clara e alinhada com os requisitos do projeto.

A seguir estão descritos todos os controladores e *endpoints* da API, bem como alguns exemplos de respostas.

1. Cinemas

O **CinemaController** é um **RestController** que permite a consulta de cinemas ativos, o acesso aos detalhes de um cinema específico, os filmes em exibição desse cinema, e outras ações como pesquisa por nome e o número de sessões associadas a um cinema.

Method	***Endpoint***	**Descrição**
GET	/cinemas	Lista de cinemas ativos (?q= → pesquisa por nome opcional)
GET	/cinemas/{id}	Detalhes de um cinema
GET	/cinemas/{id}/filmes	Filmes em exibição de um cinema
GET	/cinemas/{id}/count-sessoes	Número de sessões ativas de um cinema

Exemplos de invocação

• Obter a lista de cinemas ativos

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/cinemas
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/cinemas?q=Leiria
```

Resposta:

```
[
    {
        "id": 1,
        "nome": "CineLive Leiria",
        "morada": "Rua Dr. João Carneiro Nº25, 2400-149 Leiria",
        "telefone": "244123456",
        "email": "leiria@cinelive.pt",
        "horario": "10:00 - 23:30",
        "capacidade": "6 Salas • 680 Lugares",
        "has_sessoes": true
    },
    ...
]
```

• Detalhes de um cinema

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/cinemas/1
```

• Filmes em exibição de um cinema

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/cinemas/1/filmes
```

Resposta:

```
[
    {
        "id": 8,
        "titulo": "Carros 2",
        "poster_url": "/CineLive/Web/frontend/web/uploads/posters/poster_6910b6ad1f9ea.jpg"
    },
]
```

2. Filmes

O **FilmeController** é um **RestController** que permite a consulta e pesquisa de filmes, o acesso aos detalhes de um filme específico, a obtenção das sessões associadas e a aplicação de diversos filtros, como filmes infantis, estreias, género, idioma e popularidade.

Method	***Endpoint***	**Descrição**
GET	/filmes	Lista de filmes • (?cinema_id= cinema opcional) • (?filter= 'kids' ou 'brevemente' opcional para listar filmes para crianças ou brevemente) • (?q= pesquisa por título opcional)
GET	/filmes/count	Contagem de todos os filmes já criados
GET	/filmes/{id}	Detalhes de um filme
GET	/filmes/{id}/sessoes	Sessões do filme (?cinema_id= → cinema opcional)
GET	/filmes/{id}/count-sessoes	Número de sessões do filme (?cinema_id= → cinema opcional)
GET	/filmes/mais-vistos	Filmes mais vistos (?limit = → limite opcional)
GET	/filmes/estreiam-hoje	Filmes que estreiam hoje
GET	/filmes/por-genero	Filmes por género (?q= → género)
GET	/filmes/por-idioma	Filmes por idioma (?q= → idioma)

Exemplos de invocação

• Obter a lista de filmes

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes?q=Interstellar
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes?cinema_id=1
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes?filter=kids
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes?filter=brevemente
```

Resposta:


```
[
    {
        "id": 8,
        "titulo": "Carros 2",
        "poster_url": "/CineLive/Web/frontend/web/uploads/posters/poster_6910b6ad1f9ea.jpg"
    },
    {
        "id": 11,
        "titulo": "Divertida-Mente",
        "poster_url": "/CineLive/Web/frontend/web/uploads/posters/poster_6918b4c3cf56d.jpg"
    },
]
```

• Contagem de todos os filmes já criados

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes/count
```

• Detalhes de um filme

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes/1
```

Resposta:


```
{
    "id": 1,
    "titulo": "Interstellar",
    "poster_url": "/CineLive/Web/frontend/web/uploads/posters/poster_68fa01aecd6d2.jpg",
    "rating": "M12",
    "generos": "Ação, Drama, Ficção Científica",
    "estreia": "23/10/2025",
    "duracao": "2h 49min",
    "idioma": "Inglês",
    "realizacao": "Christopher Nolan",
    "sinopse": "Em um futuro próximo...,
    "has_sessoes": false
}
```


• Sessões de um filme

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes/1/sessoes

curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes/1/sessoes?cinema_id=1
```

Resposta:


```
{
    "21/01/2026": [
        {
            "id": 7,
            "hora_inicio": "16:00",
            "hora_fim": "17:46",
            "cinema_id": 1
        }
    ]
}
```


• Número de sessões de um filme

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes/1/count-sessoes
```

• Filmes mais vistos

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/filmes/1/mais-vistos?limit=5
```

3. Sessões

O **SessoesController** é um **RestController** que permite a consulta de sessões ativas o acesso aos detalhes de uma sessão específica.

Method	***Endpoint***	**Descrição**
GET	/sessoes	Lista de sessões ativas (?cinema_id= → cinema opcional)
GET	/sessoes/{id}	Detalhes de uma sessão

Exemplos de invocação

• Obter a lista de sessões

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/sessoes
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/sessoes?cinema_id=1
```

Resposta:

```
[
    {
        "id": 1,
        "data": "11/02/2026",
        "hora_inicio": "13:00",
        "hora_fim": "18:33",
        "filme_id": 8,
        "cinema_id": 1,
        "sala_id": 1
    },
    ...
]
```

• Detalhes de uma sessão

```
curl -X GET 172.22.21.212/CineLive/Web/backend/web/api/sessoes/1
```

Resposta:

```
{
    "id": 8,
    "data": "11/01/2026",
    "hora_inicio": "10:00",
    "hora_fim": "11:43",
    "filme_id": 2,
    "cinema_id": 1,
    "cinema_nome": "CineLive Leiria",
    "sala": {
        "id": 2,
        "nome": "Sala 2",
        "preco_bilhete": 10,
        "num_filas": 10,
        "num_colunas": 10,
        "lugares_ocupados": ["B4","B5"]
    }
}
```

4. Autenticação

O **AuthController** é um **RestController**, que é responsável pela autenticação do utilizador, com as funcionalidades de *login*, *signup* e validação do *access-token* (campo *auth_key* da tabela *user*).

Method	***Endpoint***	**Descrição**
POST	/auth/login	Efetuar login
POST	/auth/signup	Criar uma conta
POST	/auth/validate	Validar o access-token

Exemplos de invocação

• Login

```
curl -X POST "http://172.22.21.212/CineLive/Web/backend/web/api/auth/login"
  -H "Content-Type: application/json"
  -d '{
        "username": "cliente_exemplo",
        "password": "password123"
      }'
```

• Signup

```
curl -X POST "http://172.22.21.212/CineLive/Web/backend/web/api/auth/signup"
  -H "Content-Type: application/json"
  -d '{
        "username": "cliente_novo",
        "password": "password123",
        "email": "cliente@email.com",
        "nome": "João Silva",
        "telemovel": "912345678"
      }'
```

• Validar o *token* do utilizador

```
curl -X POST 172.22.21.212/CineLive/Web/backend/web/api/auth/validate
?access-token={token}
```

Resposta (de uma forma geral, é semelhante para os 3 *endpoints*):

```
{
    "status": "success",
    "access-token": {token},
    "perfil": {
        "id": 14,
        "username": "cliente1",
        "nome": "Miguel Ribeiro",
        "email": "miguel.ribeiro@email.com",
        "telemovel": "912345678"
    }
}
```

5. Perfil

O **PerfilController** é um **RestController** que permite ao utilizador realizar todas as operações necessárias do seu perfil, incluindo a consulta, a edição e a eliminação.

Method	***Endpoint***	**Descrição**
GET	/perfil	Ver o seu perfil
PUT	/perfil	Editar o seu perfil (qualquer campo ou todos)
DELETE	/perfil	Eliminar o seu perfil

Exemplos de invocação

• Ver o seu perfil

```
curl -X GET http://172.22.21.212/CineLive/Web/backend/web/api/perfil
?access-token={token}
```

Resposta:

```
{
    "id": 14,
    "username": "cliente1",
    "nome": "Miguel Ribeiro",
    "email": "miguel.ribeiro@email.com",
    "telemovel": "912345678"
}
```

• Editar o seu perfil

```
curl -X PUT "http://172.22.21.212/CineLive/Web/backend/web/api/perfil
?access-token={token}"
  -H "Content-Type: application/json"
  -d '{
        "nome": "João Silva",
      }'
```

Resposta:

```
{
    "status": "success",
    "message": "Perfil atualizado com sucesso.",
    "user": {
       "id": 14,
       "username": "cliente1",
       "nome": "Miguel Ribeiro",
       "email": "miguel.ribeiro@email.com",
       "telemovel": "912345678"
    }
}
```

• Eliminar o seu perfil

```
curl -X DELETE http://172.22.21.212/CineLive/Web/backend/web/api/perfil
?access-token={token}
```

6. Compras

O **CompraController** é um **RestController** que permite ao utilizador autenticado consultar o histórico das suas compras, aceder aos detalhes de uma compra específica, efetuar novas compras de bilhetes, bem como obter informação adicional relacionada, como os bilhetes associados, contagens e totais de compras.

Este controlador suporta apenas operações associadas ao utilizador autenticado.

Method	***Endpoint***	**Descrição**
GET	/compras	Lista as compras do utilizador autenticado
GET	/compras/{id}	Detalhes de uma compra
POST	/compras	Criação de uma nova compra
DELETE	/compras/{id}	Elimina uma compra cancelada
GET	/compras/com-bilhetes	Lista compras com os respetivos bilhetes
GET	/compras/{id}/bilhetes	Bilhetes associados a uma compra
GET	/compras/total	Total gasto pelo utilizador
GET	/compras/count	Número total de compras do utilizador
GET	/compras/{id}/count-bilhetes	Número de bilhetes de uma compra
GET	/compras/count-por-filme	Número de compras por filme
GET	/compras/count-por-cinema	Número de compras por cinema

Exemplos de invocação

• Obter a lista de compras do utilizador

```
curl -X GET http://172.22.21.212/CineLive/Web/backend/web/api/compras
?access-token={token}
```

Resposta:


```
[
    {
        "id": 9,
        "cliente_id": 16,
        "data": "03/12/2025",
        "total": "60.00€",
        "estado": "Confirmada",
        "pagamento": "MBWAY",
        "filme_id": 15,
        "filme_titulo": "The Batman",
        "cinema_id": 1,
        "cinema_nome": "CineLive Leiria",
        "sala_id": 2,
        "sala_nome": "Sala 2",
        "sessao_id": 12,
```

```
        "sessao_data": "29/01/2026",
        "sessao_hora_inicio": "18:00",
        "sessao_hora_fim": "20:56",
        "lugares": "F4, F5, F8, F7, F6, F9"
    },
    ...
]
```

• Detalhes de uma compra

```
curl -X GET http://172.22.21.212/CineLive/Web/backend/web/api/compras/1
?access-token={token}
```

• Criar uma compra

```
curl -X POST "http://172.22.21.212/CineLive/Web/backend/web/api/compras
?access-token={token}"
  -H "Content-Type: application/json"
  -d '{
        "sessao_id": 3,
        "pagamento": "mbway",
        "lugares": ["A1", "A2"]
      }'
```

• Bilhetes associados a uma compra

```
curl -X GET http://172.22.21.212/CineLive/Web/backend/web/api/compras/1/bilhetes
?access-token={token}
```

Exemplo:

```
[
    {
        "id": 1,
        "codigo": "57SV7K",
        "lugar": "B4",
        "preco": "10.00€",
        "estado": "Pendente"
    },
    ...
]
```

• Número total de compras do utilizador

```
curl -X GET http://172.22.21.212/CineLive/Web/backend/web/api/compras/count
?access-token={token}
```

• Total gasto pelo utilizador

```
curl -X GET http://172.22.21.212/CineLive/Web/backend/web/api/compras/total
?access-token={token}
```

• Eliminar uma compra cancelada

```
curl -X DELETE http://172.22.21.212/CineLive/Web/backend/web/api/compras/1
?access-token={token}
```

7. Alugueres

O **AluguerSalaController** é um **RestController** que permite ao utilizador autenticado consultar os seus pedidos de aluguer, aceder aos detalhes de um aluguer específico, criar pedidos de aluguer de salas e atualizar ou eliminar alugueres existentes, respeitando as regras de disponibilidade e o estado do aluguer.

Este controlador suporta apenas operações associadas ao utilizador autenticado.

Method	***Endpoint***	**Descrição**
GET	/aluguer-salas	Lista os alugueres do utilizador autenticado
GET	/aluguer-salas/{id}	Detalhes de um aluguer
POST	/aluguer-salas	Criação de um novo aluguer de sala
PUT	/aluguer-salas/{id}	Atualização de um aluguer
DELETE	/aluguer-salas/{id}	Eliminação de um aluguer

Exemplos de invocação

• Obter a lista de alugueres do utilizador

```
curl -X GET http://172.22.21.212/CineLive/Web/backend/web/api/aluguer-salas
?access-token={token}
```

Resposta:

```
[
    {
        "id": 1,
        "cliente_id": 14,
        "cinema_id": 1,
        "sala_id": 1,
        "data": "2026-01-29",
        "hora_inicio": "10:00:00",
        "hora_fim": "12:00:00",
        "estado": "confirmado",
        "tipo_evento": "Festa de aniversário",
        "observacoes": "Somos um grupo de 8 pessoas..."
    },
    ...
]
```

• Detalhes de um aluguer

```
curl -X GET http://172.22.21.212/CineLive/Web/backend/web/api/aluguer-salas/1
?access-token={token}
```

• Criar um aluguer de sala

```
curl -X POST "http://172.22.21.212/CineLive/Web/backend/web/api/aluguer-salas
?access-token={token}"
  -H "Content-Type: application/json"
  -d '{
```

```
        "data": "2026-01-20",
        "hora_inicio": "14:00",
        "hora_fim": "18:00",
        "cinema_id": 1,
        "sala_id": 3,
        "tipo_evento": "Evento privado",
        "observacoes": "Aniversário"
      }'
```

• Atualizar um aluguer de sala

```
curl -X PUT "http://172.22.21.212/CineLive/Web/backend/web/api/aluguer-salas
?access-token={token}"
  -H "Content-Type: application/json"
  -d '{
        "estado": "cancelado"
      }'
```

Resposta:

```
{
    "status": "success",
    "id": 1
}
```

• Eliminar um aluguer de sala

```
curl -X DELETE http://172.22.21.212/CineLive/Web/backend/web/api/aluguer-salas/1
?access-token={token}
```

8. Géneros

Para cumprir o requisito de implementação de **3 controladores com operações CRUD** completas, foi necessário identificar entidades em que a operação de eliminação fizesse sentido no contexto do sistema.

Em vários controladores da **API**, a operação **DELETE** é **pouco frequente ou inexistente**, uma vez que a remoção de dados é raramente adequada do ponto de vista funcional e de integridade do sistema.

Neste contexto, o **GeneroController** é um **ActiveController** que foi concebido para disponibilizar um conjunto completo de operações **CRUD**, permitindo a gestão total dos géneros de filmes. As operações de criação, atualização e eliminação encontram-se restritas a utilizadores com permissões administrativas, sendo o controlo de acessos assegurado através de regras de **RBAC**, implementadas no método **checkAccess**.

Desta forma, garante-se não só o cumprimento dos requisitos do enunciado, como também a aplicação prática de mecanismos de autorização, acrescentando valor ao projeto ao assegurar que apenas utilizadores autorizados podem efetuar alterações estruturais aos dados, preservando a integridade e consistência da informação.

Method	***Endpoint***	**Descrição**
GET	/generos	Lista todos os géneros
GET	/generos/{id}	Detalhes de um género
POST	/generos	Criação de um novo género (*admin*)
PUT	/generos/{id}	Atualização de um género (*admin*)
DELETE	/generos/{id}	Eliminação de um género (*admin*)

Exemplos de invocação

• Listar géneros

```
curl -X GET http://172.22.21.212/CineLive/Web/backend/web/api/generos
```

Resposta:

```
[
    {
        "id": 2,
        "nome": "Ação"
    },
    ...
]
```

• Detalhes de um género

```
curl -X PUT http://172.22.21.212/CineLive/Web/backend/web/api/generos/1
```

• Criar um género (*admin*)

```
curl -X POST "http://172.22.21.212/CineLive/Web/backend/web/api/generos
?access-token={token}"
  -H "Content-Type: application/json"
  -d '{
        "nome": "Drama"
      }'
```

• Atualizar um género (*admin*)

```
curl -X PUT "http://172.22.21.212/CineLive/Web/backend/web/api/generos/1
?access-token={token}"
  -H "Content-Type: application/json"
  -d '{
        "nome": "Drama"
      }'
```

• Eliminar um género (*admin*)

```
curl -X DELETE http://172.22.21.212/CineLive/Web/backend/web/api/generos/1
?access-token={token}
```

Messaging com MQTT

Para além da comunicação síncrona baseada em pedidos **HTTP**, o projeto integra um mecanismo de comunicação assíncrona baseado no modelo ***publish/subscribe***, recorrendo ao protocolo **MQTT**.

Esta abordagem permite a notificação de eventos relevantes em tempo real, sem necessidade de pedidos constantes por parte das aplicações cliente, promovendo o desacoplamento entre os diferentes componentes do sistema.

De forma a simplificar a comunicação MQTT, foi criado um **serviço auxiliar (MqttService)**, localizado em **common/helpers**, responsável por estabelecer a ligação ao broker, publicar mensagens nos respetivos tópicos e tratar eventuais erros de comunicação. Este serviço é reutilizado pelos diferentes modelos do sistema, garantindo consistência e reduzindo a duplicação de código.

O sistema publica mensagens MQTT sempre que ocorrem eventos significativos, como a atualização do estado de um aluguer de sala, a realização de uma nova compra ou a criação e alteração do estado de filmes.

A seguir são descritos os diferentes canais de envio de mensagens disponíveis no sistema, bem como exemplos de mensagens publicadas.

1. Alugueres

- `cinelive/alugueres/{cliente_id}`

Este canal é utilizado para notificar um cliente específico sempre que o **estado de um pedido de aluguer de sala** é alterado.

Sempre que um aluguer é confirmado ou cancelado, o sistema publica uma mensagem contendo a identificação do aluguer, o novo estado e uma mensagem informativa.

```
Exemplo de mensagem
Topic: cinelive/alugueres/12
---

{
  "id": 4,
  "estado": "confirmado",
  "mensagem": "O seu pedido de aluguer #4 foi confirmado.",
  "data": "2026-01-20",
  "hora_inicio": "14:00",
  "hora_fim": "18:00"
}
```

2. Compras

• cinelive/cinema/{cinema_id}/compras

Este canal é utilizado para notificar um cinema sempre que ocorre uma **nova compra de bilhetes** associada a uma das suas sessões.

A mensagem publicada inclui informação sobre a compra, o filme, a sessão e o cliente, permitindo ao cinema acompanhar em tempo real a atividade de vendas.
Este canal é particularmente útil para efeitos de monitorização e estatísticas internas.

```
Exemplo de mensagem
Topic: cinelive/cinema/1/compras
---

{
  "compra_id": 1,
  "cinema_id": 1,
  "cinema": "CineLive Leiria",
  "sessao_id": 14,
  "filme": "Cars",
  "data": "2026-01-18 21:30:00",
  "cliente_id": 7,
  "cliente_nome": "cliente_exemplo",
  "bilhetes": 2,
  "mensagem": "Nova compra realizada no cinema 'CineLive Leiria' para o filme 'Cars'."
}
```

3. Filmes

• cinelive/filmes

Este canal é utilizado para notificar as aplicações cliente sempre que um **novo filme é adicionado ao catálogo** ou quando o estado de um filme é alterado para terminado.

Desta forma, as aplicações podem atualizar automaticamente as listas de filmes disponíveis, garantindo que a informação apresentada ao utilizador está sempre sincronizada com o estado do sistema.

```
Exemplo de mensagem
Topic: cinelive/filmes
---

{
  "filme_id": 24,
  "estado": "em_exibicao",
  "titulo": "Duna: Parte Dois",
  "mensagem": "Um novo filme foi adicionado: Interstellar.",
  "estreia": "2026-01-25"
}
```

Conclusão

No âmbito da UC de **Serviços e Interoperabilidade de Sistemas**, o projeto **CineLive** permitiu aplicar de forma prática os conceitos fundamentais relacionados com o desenvolvimento de **APIs RESTful** e comunicação assíncrona, assegurando a interoperabilidade entre a aplicação Android e os serviços *backend* do sistema.

A **API RESTful** desenvolvida disponibiliza um conjunto abrangente de *endpoints*, organizados por controladores, que permitem suportar as principais funcionalidades da aplicação cliente, incluindo operações **CRUD**, ações personalizadas e relações ***master/detail*** entre entidades.

A utilização de mecanismos de autenticação garante a segurança e o controlo de acesso às operações sensíveis, assegurando que cada utilizador apenas acede aos recursos que lhe são permitidos.

Para complementar, a integração de ***messaging*** baseado em **MQTT**, recorrendo ao modelo ***publish/subscribe***, permitiu implementar uma comunicação assíncrona eficiente, possibilitando a notificação em tempo real de eventos relevantes, como a realização de compras, a atualização de alugueres de salas e alterações ao estado dos filmes.

Em conclusão, a componente de **SIS** do projeto **CineLive** demonstra uma solução funcional, escalável e alinhada com as boas práticas de programação, evidenciando a aplicação correta dos conhecimentos adquiridos ao longo das aulas.