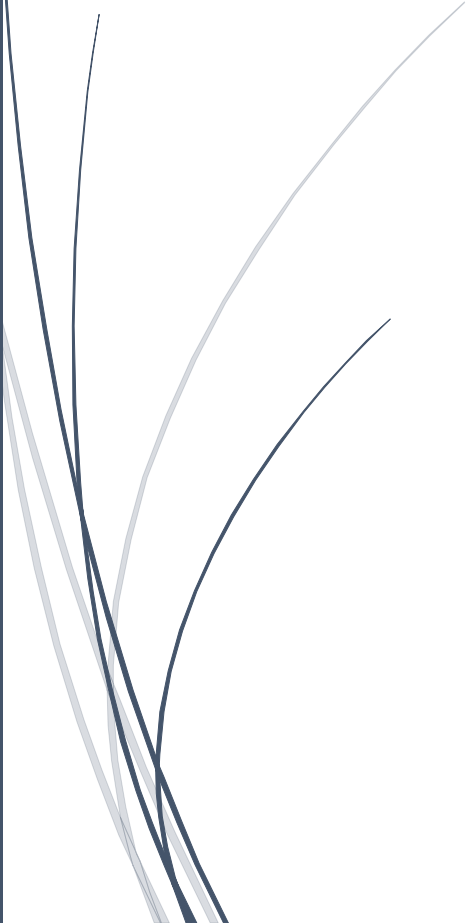


A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

23-05-2021

Relatório M2

Programação Web

Several thin, curved lines in dark blue and light grey originate from the bottom left and sweep upwards and to the right.

Hugo Miguel Campinho Rodrigues – A033005;João
de Jesus Chivarria – A033882;Dérick José Lopes
Costa – A037672

ISMAI – INSTITUTO UNIVERSITÁRIO DA MAIA

Conteúdo

Introdução	2
Contextualização do tema.....	3
Parte 1 – Construção Lógica da Base de Dados	4
Diagrama de classes	4
Explicação do Diagrama de classes	5
Parte 2 – Construção da Base de Dados.....	9
Criação das Tabelas	9
Inserção de Registos nas Tabelas	10
Arquitetura do tipo REST.....	10
Tabelas Criados na Base de Dados	16
Parte 3 – Documentação, JSON e Node.js.....	20
Documentação da API com recurso ao formato OpenAPI 3.0	20
XAMPP.....	24
POSTMAN	26
Conclusão	28

Introdução

Neste momento de avaliação, foi solicitado ao grupo de trabalho a criação de uma REST API que contivesse uma base de dados integrada. O tema da mesma foi livre, ou seja, ficou ao critério de cada grupo de trabalho escolher o seu próprio tema.

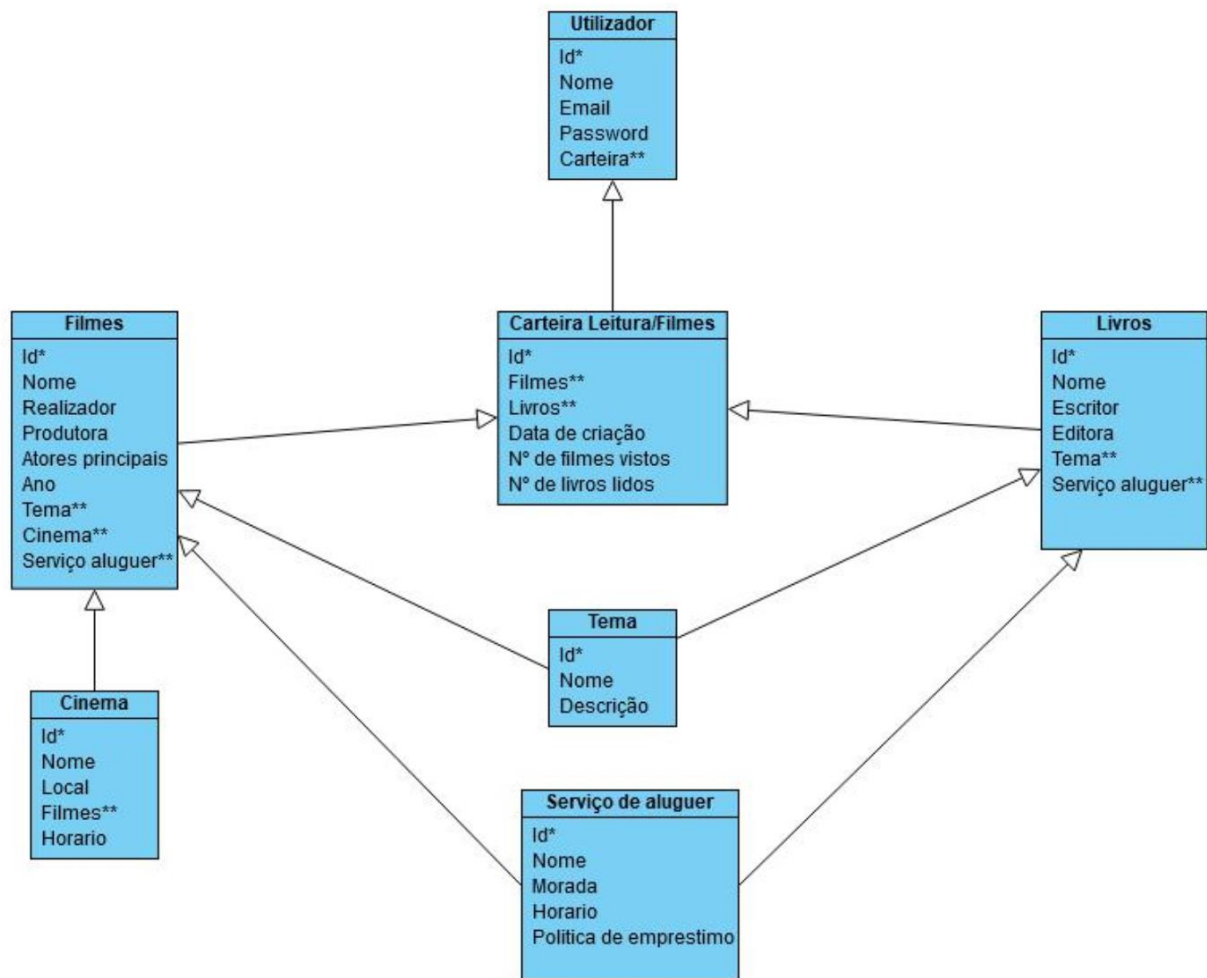
Após falarmos com o professor da disciplina, o nosso grupo de trabalho escolheu como tema uma API de gestão de livros e filmes.

Contextualização do tema

“Pretendemos criar uma API para fornecer um serviço ao utilizador que permite guardar um registo de todos os livros que já leu ou comprou assim como de filmes. Pretendemos ainda que seja possível procurar livros por várias categorias assim como ser possível ir diretamente as páginas das editoras para os adquirir. Com os filmes desejamos implementar as mesmas funcionalidades referidas nos livros mas também criar funcionalidades novas como, saber os filmes a estrear e em que salas de cinema. A estrutura que visualizamos nesta fase para o projeto, assenta numa conta de utilizador que contem os dados das leituras, a API usa-os para definir padrões de leitura que irão preferencialmente exibir livros associados ao “gosto” utilizador, seja por autores, editoras ou outra qualquer categoria associada aos livros. Este também será o caso para os filmes, pesquisas por atores, temas ou editoras. Ainda esperamos incluir funcionalidades para informar o utilizador dos seus hábitos de leitura ou hábitos de cinematográficos com o objetivo comercial de poder ou não vender esta informação as editoras. Por fim, seria uma mais-valia a implementação de funcionalidades de também associar os dados a serviços de biblioteca/aluguer de filmes com o objetivo de enriquecer a informação de onde o utilizador pode adquirir os produtos em questão.”

Parte 1 – Construção Lógica da Base de Dados

Diagrama de classes



Explicação do Diagrama de classes

Construímos a nossa Base de Dados com a seguinte lógica:

Existem utilizadores que possuem:

- Id -> chave primária;
- Nome;
- Email;
- Password;
- Carteira -> chave estrangeira, obtida através da tabela Carteira Leitura/Filmes.

Cada utilizador tem uma ou mais carteira/s de Leitura/Filmes. Nessa/s carteira/s, estão presentes os elementos:

- Id -> Cada carteira tem um id único que é usado como chave primária;
- Filmes -> Podem existir um ou vários filmes em cada carteira, sendo que os filmes são obtidos através da Tabela Filmes. Filmes é uma chave estrangeira;
- Livros -> Podem existir um ou vários livros em cada carteira, sendo que os livros são obtidos através da Tabela Livros. Livros é uma chave estrangeira;
- Data de criação;
- Número de Filmes vistos;
- Número de Livros lidos.

Livros, que é passado como chave estrangeira para a carteira, é uma tabela, na qual

estão incluídos os seguintes parâmetros:

- Id -> chave primária;
- Nome;
- Escritor;
- Editora;
- Tema -> Podem existir um ou vários temas em cada livro, sendo que os temas são obtidos através da Tabela Tema. Tema é uma chave estrangeira;
- Serviço aluguer -> Podem existir um ou vários alugueres de cada livro, sendo que os alugueres são obtidos através da Tabela Serviço de Aluguer. Serviço aluguer é uma chave estrangeira;

Filmes, que é passado como chave estrangeira para a carteira, é uma tabela, na qual

estão incluídos os seguintes parâmetros:

- Id -> chave primária;
- Nome;
- Realizador;
- Produtora;
- Atores Principais;
- Ano;

- Tema -> Podem existir um ou vários temas em cada filme, sendo que os temas são obtidos através da Tabela Tema. Tema é uma chave estrangeira;
- Cinema -> Cada filme pode estar em um ou mais cinemas, sendo que cinema é uma chave estrangeira obtida através da tabela Cinema;
- Serviço aluguer -> Podem existir um ou vários alugueres de cada Filme, sendo que os alugueres são obtidos através da Tabela Serviço de Aluguer. Serviço aluguer é uma chave estrangeira;

Tema que é passado como chave estrangeira para os filmes e para os livros, tem os seguintes parâmetros:

- Id -> chave primária;
- Nome;
- Descrição.

Serviço aluguer que é passado como chave estrangeira para os filmes e para os livros, tem os seguintes parâmetros:

- Id -> chave primária;
- Nome;
- Morada;

- Horário;
- Política de Empréstimo.

Cinema que é passado como chave estrangeira para os filmes, tem os seguintes parâmetros:

- Id -> chave primária;
- Nome;
- Local;
- Filmes -> Podem existir um ou vários filmes em cada cinema, sendo que os filmes são obtidos através da Tabela Filmes. Filmes é uma chave estrangeira;
- Horário.

Parte 2 – Construção da Base de Dados

Após a criação do Diagrama de classes e de fazermos a consequente análise, verificou-se a seguinte situação:

- Existe a necessidade de fazermos a criação de tabelas;
- Depois de termos tabelas, temos de fazer a inserção manual dos registos;
- Aquando da criação dos registos, é necessário estruturarmos a arquitetura REST da Base de Dados, ou seja, implementar os métodos que iríamos utilizar;
- Fazer a ligação ao XAMPP;
- Utilizar JSON e Node.js

Tendo estes tópicos definidos, passamos á construção da Base de Dados propriamente dita.

Criação das Tabelas

Seguindo os pdf's disponibilizados pelo professor da disciplina, fizemos a criação das tabelas de acordo com o exemplo fornecido, o Schedule.

































































































Tabela	Acções	Registos	Tipo	Agrupamento (Collation)	Tamanho	Suspensão
<input type="checkbox"/> carteira	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	30	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> cinema	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	32	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> filme	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	30	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> livro	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	30	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> servico	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	30	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> tema	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	30	InnoDB	utf8mb4_general_ci	16.0 KB	-
<input type="checkbox"/> utilizador	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	30	InnoDB	utf8mb4_general_ci	16.0 KB	-
7 tabelas	Soma	212	InnoDB	utf8mb4_general_ci	112.0 KB	0 Bytes

Inserção de Registos nas Tabelas

Seguidamente, tal como requerido no enunciado, passamos á inserção manual dos registos em cada tabela. Devido á grande quantidade de registos requeridos, esta foi das tarefas mais morosas deste trabalho.

+ Opções

Clique na seta para alternar a visibilidade da coluna.

	id	nome	local	horario	filmeid
<input type="checkbox"/>  Editar  Copiar  Apagar	2	Cinemas Arrabida	V.N de Gaia	Das 10:00 as 02:00	1
<input type="checkbox"/>  Editar  Copiar  Apagar	3	Cinemas Barroso	V.N de Tomego	Das 10:00 as 20:00	2
<input type="checkbox"/>  Editar  Copiar  Apagar	22	Cinemas Acrobata	V.N. das Luzinhas	Das 06:00 as 18:00	3
<input type="checkbox"/>  Editar  Copiar  Apagar	6	Cinemas MarSol	S.Mamede do Monte	Das 08:00 as 20:00	4
<input type="checkbox"/>  Editar  Copiar  Apagar	7	Cinemas Solpoente	Francelinhos	Das 08:00 as 20:00	5
<input type="checkbox"/>  Editar  Copiar  Apagar	23	Cinemas Acrobata	V.N. das Luzinhas	Das 06:00 as 18:00	6
<input type="checkbox"/>  Editar  Copiar  Apagar	9	Cinemas GotadeAgua	Rios-entre-Rios	Das 18:00 as 03:00	7
<input type="checkbox"/>  Editar  Copiar  Apagar	10	Cinemas Sapo	Lagoa Azul	Das 06:00 as 12:00	8
<input type="checkbox"/>  Editar  Copiar  Apagar	11	Cinemas Julia	Monte do lado	Das 06:00 as 15:00	9
<input type="checkbox"/>  Editar  Copiar  Apagar	8	Cinemas Sombra	Francelinhos	Das 15:00 as 03:00	10
<input type="checkbox"/>  Editar  Copiar  Apagar	12	Cinemas João Batista	Braga	Das 10:00 as 00:00	11
<input type="checkbox"/>  Editar  Copiar  Apagar	20	Cinemas Sapo	Lagoa Azul	Das 06:00 as 12:00	11
<input type="checkbox"/>  Editar  Copiar  Apagar	13	Cinemas João Batista	Ponte de Lima	Das 10:00 as 00:00	12
<input type="checkbox"/>  Editar  Copiar  Apagar	24	Cinemas Arrabida	V.N de Gaia	Das 10:00 as 02:00	13
<input type="checkbox"/>  Editar  Copiar  Apagar	14	Cinemas João Batista	Faro	Das 10:00 as 00:00	14
<input type="checkbox"/>  Editar  Copiar  Apagar	15	Cinemas Futurama	Galinhos	Das 10:00 as 20:00	15
<input type="checkbox"/>  Editar  Copiar  Apagar	25	Cinemas Arrabida	V.N de Gaia	Das 10:00 as 02:00	16
<input type="checkbox"/>  Editar  Copiar  Apagar	21	Cinemas Acrobata	V.N. das Luzinhas	Das 06:00 as 18:00	17
<input type="checkbox"/>  Editar  Copiar  Apagar	16	Cinemas Gaiatos	Castelo Preto	Das 10:00 as 18:00	18
<input type="checkbox"/>  Editar  Copiar  Apagar	27	Cinema Fantasmilha	Reboita	Das 16:00 as 02:00	19
<input type="checkbox"/>  Editar  Copiar  Apagar	18	Cinema Municipal de Tia	V.N de Tia	Das 06:00 as 13:00	20
<input type="checkbox"/>  Editar  Copiar  Apagar	17	Cinemas Hélio Sampaio	Marcosa	Das 10:00 as 15:00	21
<input type="checkbox"/>  Editar  Copiar  Apagar	26	Cinema Matias	São Bernardo de Cima	Das 06:00 as 16:00	22
<input type="checkbox"/>  Editar  Copiar  Apagar	28	Cinema Fantasmilha	Reboita	Das 16:00 as 02:00	23
<input type="checkbox"/>  Editar  Copiar  Apagar	30	Cinema Gonsalves Lacerda	V.N de Soalhos	Das 10:00 as 02:00	24
<input type="checkbox"/>  Editar  Copiar  Apagar	32	Cinema Publico Barco	Lagoas Soltas	Das 10:00 as 20:00	25
<input type="checkbox"/>  Editar  Copiar  Apagar	1	Cinema Matias	São Bernardo de Cima	Das 14:00 as 02:00	26
<input type="checkbox"/>  Editar  Copiar  Apagar	19	Cinemas Joia	Taifões	Das 06:00 as 16:00	26
<input type="checkbox"/>  Editar  Copiar  Apagar	5	Cinemas Tobias	V.N de Kunami	Das 08:00 as 15:00	27
<input type="checkbox"/>  Editar  Copiar  Apagar	29	Cinema Fantasmilha	Reboita	Das 16:00 as 02:00	28
<input type="checkbox"/>  Editar  Copiar  Apagar	31	Cinema Gonsalves Lacerda	V.N de Soalhos	Das 10:00 as 02:00	29
<input type="checkbox"/>  Editar  Copiar  Apagar	33	Cinema Publico Barco	Lagoas Soltas	Das 10:00 as 20:00	30

Seguindo o que é pretendido no enunciado, tivemos de fazer a implementação do Protocolo HTTP – CRUD

Pela lógica do que tínhamos feito anteriormente fazia todo o sentido fazermos a implementação dos seguintes métodos:

- GET
- POST
- PUT
- DELETE

PUT /cinemas/{id}

Parameters

CancelReset

Name	Description
id * required number (path)	<input type="text" value="10"/>

Request body

application/json

```
{
  "id": 10,
  "nome": "Cinemas Pardal",
  "local": "Rua Cessano 403",
  "horario": "Das 08:00 as 17:00",
  "filmeId": 8
}
```

GET /cinemas/{id}

Parameters

Cancel

Name	Description
id * required number (path)	<input type="text" value="1"/>
filter object (query)	<div><pre>{ "offset": 0, "limit": 100, "skip": 0, "order": "string", "fields": { "id": true, "nome": true, "local": true, "horario": true, "filmeId": true }, "include": [{ "relation": "string", "scope": { "offset": 0, "limit": 1, "skip": 0, "order": "string" } }] }</pre></div>

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  http://[::1]:3000/cinemas/1?filter=%7B%0A%20%20offset%22%3A%200%2C%0A%20%20limit%22%3A%20100%2C%0A%20%20skip%22%3A%200%2C%0A%20%20order%22%3A%20string%22%2C%0A%20%20fields%22%20%3A%7B%0A%20id%3Atrue%2C%0A%20nome%3Atrue%2C%0A%20local%3Atrue%2C%0A%20horario%3Atrue%2C%0A%20filmeId%3Atrue%7D%26include%3D%5B%7B%0A%20relation%3A%22string%22%2C%0A%20scope%3A%7B%0A%20offset%3A0%2C%0A%20limit%3A1%2C%0A%20skip%3A0%2C%0A%20order%3A%22string%22%7D%5D%7D%26accept%3Aapplication%2Fjson%27
  >
```


POST

/cinemas

Parameters

Cancel

Reset

No parameters

Request body

application/json

```
{  "nome": "exemplonome",  "local": "exemplolocal",  "horario": "De xx:xx as yy:yy",  "filmeId": 6}
```

Execute

Clear

Responses

Responses

Curl

```
curl -X 'POST' \  'http://[::1]:3000/cinemas' \  -H 'accept: application/json' \  -H 'Content-Type: application/json' \  -d '{    "nome": "exemplonome",    "local": "exemplolocal",    "horario": "De xx:xx as yy:yy",    "filmeId": 6  }'
```

Request URL

```
http://[::1]:3000/cinemas
```

Server response

Code

Details

200

Response body

```
{  "id": 39,  "nome": "exemplonome",  "local": "exemplolocal",  "horario": "De xx:xx as yy:yy",  "filmeId": 6}
```

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: *
connection: keep-alive
content-length: 95
content-type: application/json
date: Sun, 23 May 2021 22:45:19 GMT
keep-alive: timeout=5
x-powered-by: Express
```

CinemasControllerController

GET

/cinemas/count

Parameters

Cancel

Name	Description
where object (query)	<pre>{ "additionalProp1": {}}</pre>

Execute

Clear

Responses

Curl

```
curl -X 'GET' \  'http://[::1]:3000/cinemas/count?where=%7B%0A%20%22additionalProp1%22%3A%20%7B%7D%0A%7D' \  -H 'accept: application/json'
```

Request URL

```
http://[::1]:3000/cinemas/count?where=%7B%0A%20%22additionalProp1%22%3A%20%7B%7D%0A%7D
```

Server response

Code

Details

200

Response body

```
{
  "count": 32
}
```



Download

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: *
connection: keep-alive
content-length: 12
content-type: application/json
date: Sun, 23 May 2021 22:46:54 GMT
keep-alive: timeout=5
x-powered-by: Express
```

Responses

Code

Description

Links

200

Cinema model count

No links

Media type

application/json

Controls Accept header

Example Value | Schema

```
{
  "count": 0
}
```

Após a implementação dos métodos, surgiram-nos alguns problemas que associamos a erros na construção das relações $m : n$. Contudo, devido a questões de gestão de tempo e para conseguirmos cumprir com o prazo dado, vimo-nos forçados a manter esses erros, com a esperança que no futuro nos fosse dada a possibilidade de os corrigirmos.

Responses

Curl

```
curl -X 'GET' \
'http://[::1]:3000/cinemas?filter=%7B%0A%20%22offset%22%3A%20%2C%0A%20%22limit%22%3A%20%2C%0A%20%22skip%22%3A%20%2C%0A%20%22order%22%3A%20%22string%22%3A%20%22where%22%3A%20%22%3A%20%22%7D' \
-H 'accept: application/json'
```

Request URL

[illegible]

Server response

Code Details

500

UndocumentedError: Internal Server Error

Response body

```
{
  "error": {
    "statusCode": 500,
    "message": "Internal Server Error"
  }
}
```

 [Download](#)

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: *
connection: keep-alive
content-length: 62
content-type: application/json; charset=utf-8
date: Sun, 23 May 2021 19:02:09 GMT
keep-alive: timeout=5
x-content-type-options: nosniff
x-powered-by: Express
```

[illegible]

Tabelas Criados na Base de Dados

CarteiraFilmeController		▼
POST	/carteiras/{id}/filmes	
PATCH	/carteiras/{id}/filmes	
GET	/carteiras/{id}/filmes	
DELETE	/carteiras/{id}/filmes	
CarteiraLivreController		▼
POST	/carteiras/{id}/livros	
PATCH	/carteiras/{id}/livros	
GET	/carteiras/{id}/livros	
DELETE	/carteiras/{id}/livros	
CarteiraUtilizadorController		▼
POST	/carteiras/{id}/utilizador	
PATCH	/carteiras/{id}/utilizador	
GET	/carteiras/{id}/utilizador	
DELETE	/carteiras/{id}/utilizador	
CinemasControllerController		▼
GET	/cinemas/count	
PUT	/cinemas/{id}	
PATCH	/cinemas/{id}	
GET	/cinemas/{id}	
DELETE	/cinemas/{id}	
POST	/cinemas	
PATCH	/cinemas	
GET	/cinemas	
CarteirasControllerController		▼
GET	/carteiras/count	
PUT	/carteiras/{id}	
PATCH	/carteiras/{id}	
GET	/carteiras/{id}	
DELETE	/carteiras/{id}	
POST	/carteiras	
PATCH	/carteiras	
GET	/carteiras	

UtilizadoresControllerController		▼
GET	/utilizadors/count	
PUT	/utilizadors/{id}	
PATCH	/utilizadors/{id}	
GET	/utilizadors/{id}	
DELETE	/utilizadors/{id}	
POST	/utilizadors	
PATCH	/utilizadors	
GET	/utilizadors	
UtilizadorCarteiraController		▼
POST	/utilizadors/{id}/carteira	
PATCH	/utilizadors/{id}/carteira	
GET	/utilizadors/{id}/carteira	
DELETE	/utilizadors/{id}/carteira	
CinemaFilmeController		▼
POST	/cinemas/{id}/filmes	
PATCH	/cinemas/{id}/filmes	
GET	/cinemas/{id}/filmes	
DELETE	/cinemas/{id}/filmes	
FilmesControllerController		▼
GET	/filmes/count	
PUT	/filmes/{id}	
PATCH	/filmes/{id}	
GET	/filmes/{id}	
DELETE	/filmes/{id}	
POST	/filmes	
PATCH	/filmes	
GET	/filmes	
FilmeCarteiraController		▼
GET	/filmes/{id}/carteira	
FilmeCinemaController		▼
POST	/filmes/{id}/cinemas	
PATCH	/filmes/{id}/cinemas	
GET	/filmes/{id}/cinemas	
DELETE	/filmes/{id}/cinemas	
FilmeServicoController		▼
GET	/filmes/{id}/servico	
FilmeTemaController		▼
POST	/filmes/{id}/temas	
PATCH	/filmes/{id}/temas	
GET	/filmes/{id}/temas	
DELETE	/filmes/{id}/temas	

ServicoFilmeController		▼
POST	/servicos/{id}/filmes	
PATCH	/servicos/{id}/filmes	
GET	/servicos/{id}/filmes	
DELETE	/servicos/{id}/filmes	
ServicoLivreController		▼
POST	/servicos/{id}/livros	
PATCH	/servicos/{id}/livros	
GET	/servicos/{id}/livros	
DELETE	/servicos/{id}/livros	
TemasControllerController		▼
GET	/temas/count	
PUT	/temas/{id}	
PATCH	/temas/{id}	
GET	/temas/{id}	
DELETE	/temas/{id}	
POST	/temas	
PATCH	/temas	
GET	/temas	
TemaFilmeController		▼
GET	/temas/{id}/filme	
TemaLivreController		▼
GET	/temas/{id}/livro	
LivreTemaController		▼
POST	/livros/{id}/temas	
PATCH	/livros/{id}/temas	
GET	/livros/{id}/temas	
DELETE	/livros/{id}/temas	
PingController		▼
GET	/ping	
ServicosControllerController		▼
GET	/servicos/count	
PUT	/servicos/{id}	
PATCH	/servicos/{id}	
GET	/servicos/{id}	
DELETE	/servicos/{id}	
POST	/servicos	
PATCH	/servicos	
GET	/servicos	

LivrosControllerController		▼
GET	/livros/count	
PUT	/livros/{id}	
PATCH	/livros/{id}	
GET	/livros/{id}	
DELETE	/livros/{id}	
POST	/livros	
PATCH	/livros	
GET	/livros	
LivroCarteiraController		▼
GET	/livros/{id}/carteira	
LivroServicoController		▼
GET	/livros/{id}/servico	

Parte 3 – Documentação, JSON e Node.js

Documentação da API com recurso ao formato OpenAPI 3.0

API Filmes/Livros

A API permite que um utilizador tem uma carteira associada na qual estão registados os filmes visualizados e os livros lidos.

A API também permite criar, alterar, apagar e consultar filmes e livros, assim como temas associados e locais com serviços de aluguer dos respetivos. Também é possível consultar, editar, criar e apagar cinemas que permitem ver o filme em exibição.

Url básico:

`http://localhost:8080`

Funcionalidades:

Tema

- Retrieve temas

- o Funcionalidade do tipo GET que retorna todos os registos inseridos na tabela.

- /Temas

- Create tema

- o Funcionalidade do tipo POST que permite criar um novo registo na tabela.

- /Temas

- Retrieve tema

- o Funcionalidade do tipo GET que retorna um registo que é passado através de um parâmetro do tipo id.

- /Temas/?id=ID

- Update tema

- o Funcionalidade do tipo PUT que permite alterar um registo na tabela através de um parâmetro do tipo id.

- /Temas/?id=ID

- Delete tema

- o Funcionalidade do tipo DELETE que permite eliminar um registo na tabela através de um parâmetro do tipo id.

- /Temas/?id=IDUtilizador

- Retrieve utilizadores

o Funcionalidade do tipo GET que retorna todos os registros inseridos na tabela.

▪/Utilizadores

- Create utilizador

o Funcionalidade do tipo POST que permite criar um novo registro na tabela.

▪/Utilizadores

- Retrieve utilizador

o Funcionalidade do tipo GET que retorna um registro que é passado através de um parâmetro do tipo id.

▪/Utilizadores/?id=ID

- Update utilizador

o Funcionalidade do tipo PUT que permite alterar um registro na tabela através de um parâmetro do tipo id.

▪/Utilizadores/?id=ID

- Delete utilizador

o Funcionalidade do tipo DELETE que permite eliminar um registro na tabela através de um parâmetro do tipo id.

▪/Utilizadores/?id=ID

Carteira

- Retrieve carteiras

o Funcionalidade do tipo GET que retorna todos os registros inseridos na tabela.

▪/Carteiras

- Create carteira

o Funcionalidade do tipo POST que permite criar um novo registro na tabela.

▪/Carteiras

- Retrieve carteira

o Funcionalidade do tipo GET que retorna um registro que é passado através de um parâmetro do tipo id.

▪/Carteiras/?id=ID

- Update carteira

o Funcionalidade do tipo PUT que permite alterar um registro na tabela através de um parâmetro do tipo id.

▪/Carteiras/?id=ID

- Delete carteira

o Funcionalidade do tipo DELETE que permite eliminar um registo na tabela através de um parâmetro do tipo id.

- /Carteiras/?id=ID

Filme

- Retrieve filmes

o Funcionalidade do tipo GET que retorna todos os registos inseridos na tabela.

- /Filmes

- Create filme

o Funcionalidade do tipo POST que permite criar um novo registo na tabela.

- /Filmes

- Retrieve filme

o Funcionalidade do tipo GET que retorna um registo que é passado através de um parâmetro do tipo id.

- /Filmes/?id=ID

- Update filme

o Funcionalidade do tipo PUT que permite alterar um registo na tabela através de um parâmetro do tipo id.

- /Filmes/?id=ID

- Delete filme

o Funcionalidade do tipo DELETE que permite eliminar um registo na tabela através de um parâmetro do tipo id.

- /Filmes/?id=ID

Cinema

- Retrieve cinemas

o Funcionalidade do tipo GET que retorna todos os registos inseridos na tabela.

- /Cinemas

- Create cinema

o Funcionalidade do tipo POST que permite criar um novo registo na tabela.

- /Cinemas

- Retrieve cinema

o Funcionalidade do tipo GET que retorna um registo que é passado através de um parâmetro do tipo id.

▪/Cinemas/?id=ID

- Update cinema

o Funcionalidade do tipo PUT que permite alterar um registo na tabela através de um parâmetro do tipo id.

▪/Cinemas/?id=ID

- Delete cinema

o Funcionalidade do tipo DELETE que permite eliminar um registo na tabela através de um parâmetro do tipo id.

▪/Cinemas/?id=IDLivro

- Retrieve livros

o Funcionalidade do tipo GET que retorna todos os registos inseridos na tabela.

▪/Livros

- Create livro

o Funcionalidade do tipo POST que permite criar um novo registo na tabela.

▪/Livros

- Retrieve livro

o Funcionalidade do tipo GET que retorna um registo que é passado através de um parâmetro do tipo isbn.

▪/Livros/?isbn=ISBN

- Update livro

o Funcionalidade do tipo PUT que permite alterar um registo na tabela através de um parâmetro do tipo isbn.

▪/Livros/?isbn=ISBN

- Delete livro

o Funcionalidade do tipo DELETE que permite eliminar um registo na tabela através de um parâmetro do tipo isbn.

▪/Livros/?isbn=ISBN

Serviço de aluguer

- Retrieve serviços

o Funcionalidade do tipo GET que retorna todos os registos inseridos na tabela.

▪/Servicos

- Create serviço

o Funcionalidade do tipo POST que permite criar um novo registo na tabela.

▪/Servicos

- Retrieve serviço

o Funcionalidade do tipo GET que retorna um registo que é passado através de um parâmetro do tipo id.

▪/Servicos/?id=ID

- Update lserviço

o Funcionalidade do tipo PUT que permite alterar um registo na tabela através de um parâmetro do tipo id.

▪/Servicos/?id=ID

- Delete serviço

o Funcionalidade do tipo DELETE que permite eliminar um registo na tabela através de um parâmetro do tipo id.

▪/Servicos/?id=ID

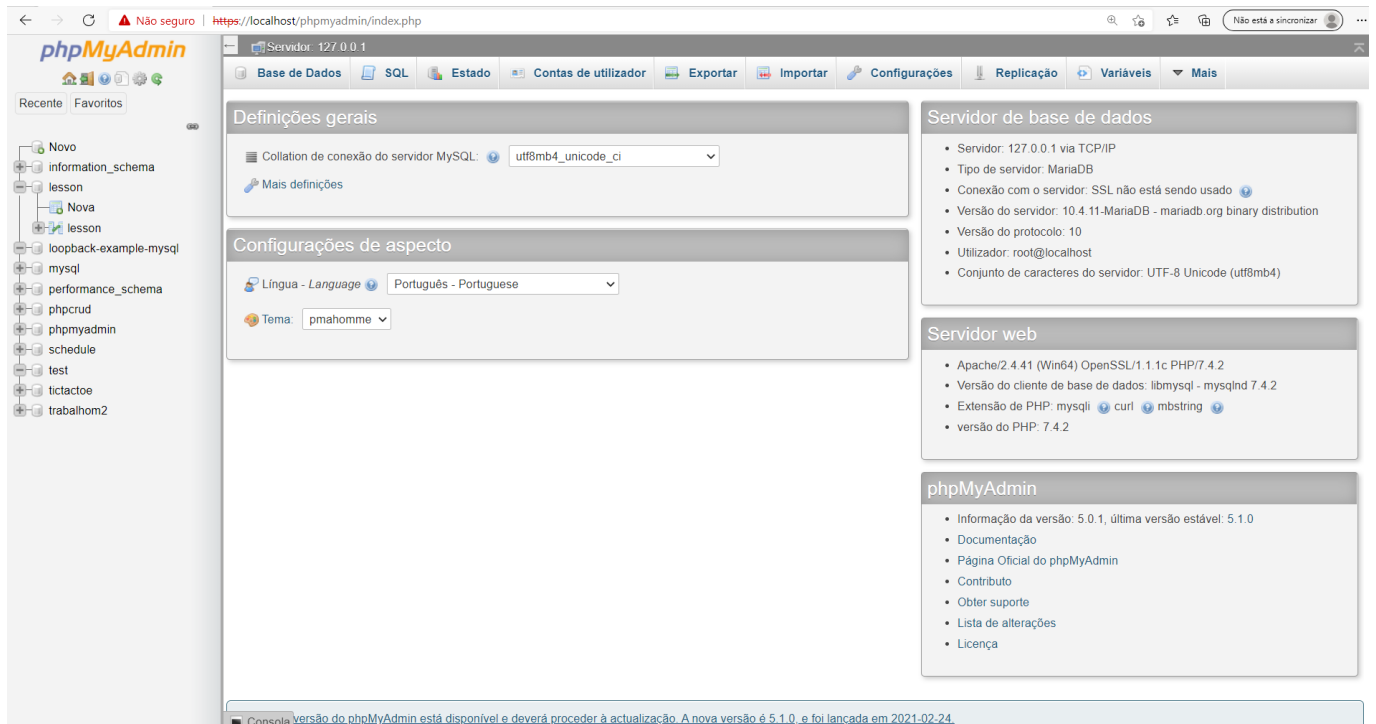
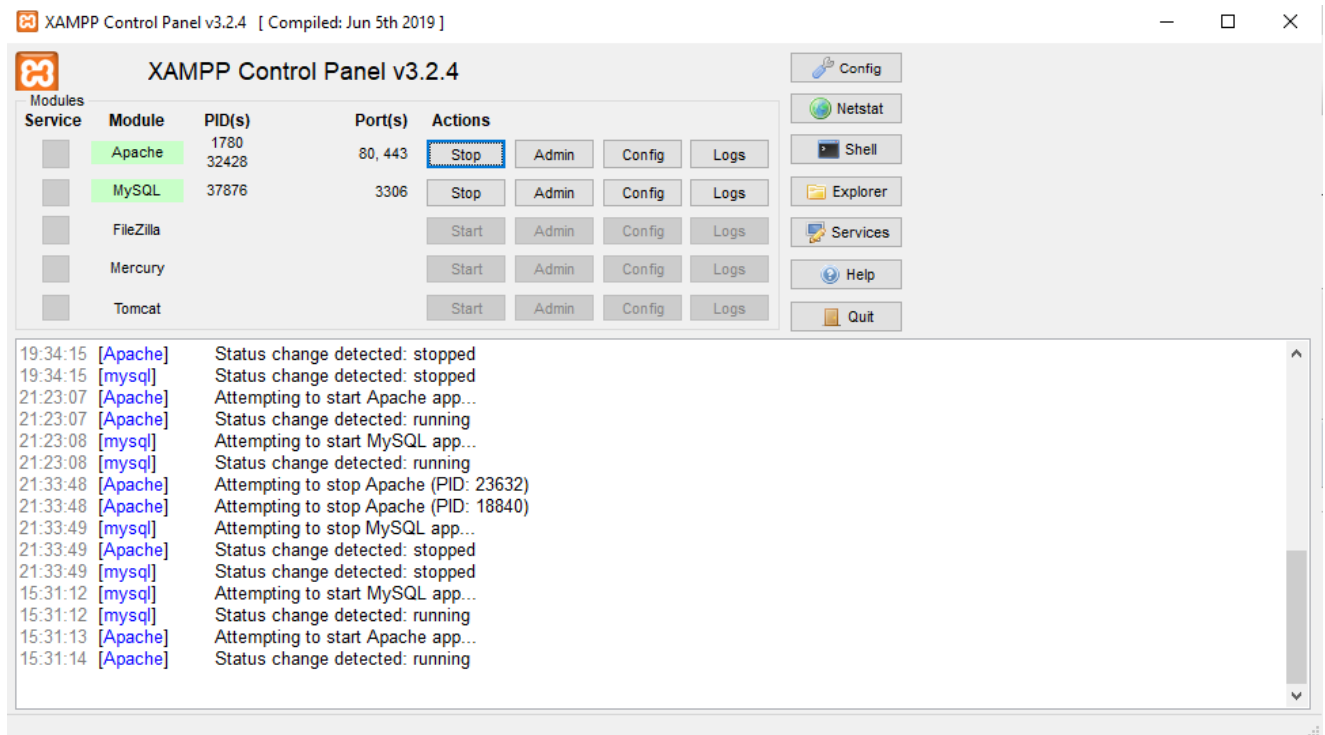
XAMPP

Na configuração do loopback Data Source, resultou o ficheiro de configuração sqldb.datasource.ts.

```
const config = {  
  name: 'sqldb',  
  connector: 'mysql',  
  url: '',  
  host: 'localhost',  
  port: 3306,  
  user: 'root',  
  password: '',  
  database: 'trabalhom2'  
};
```

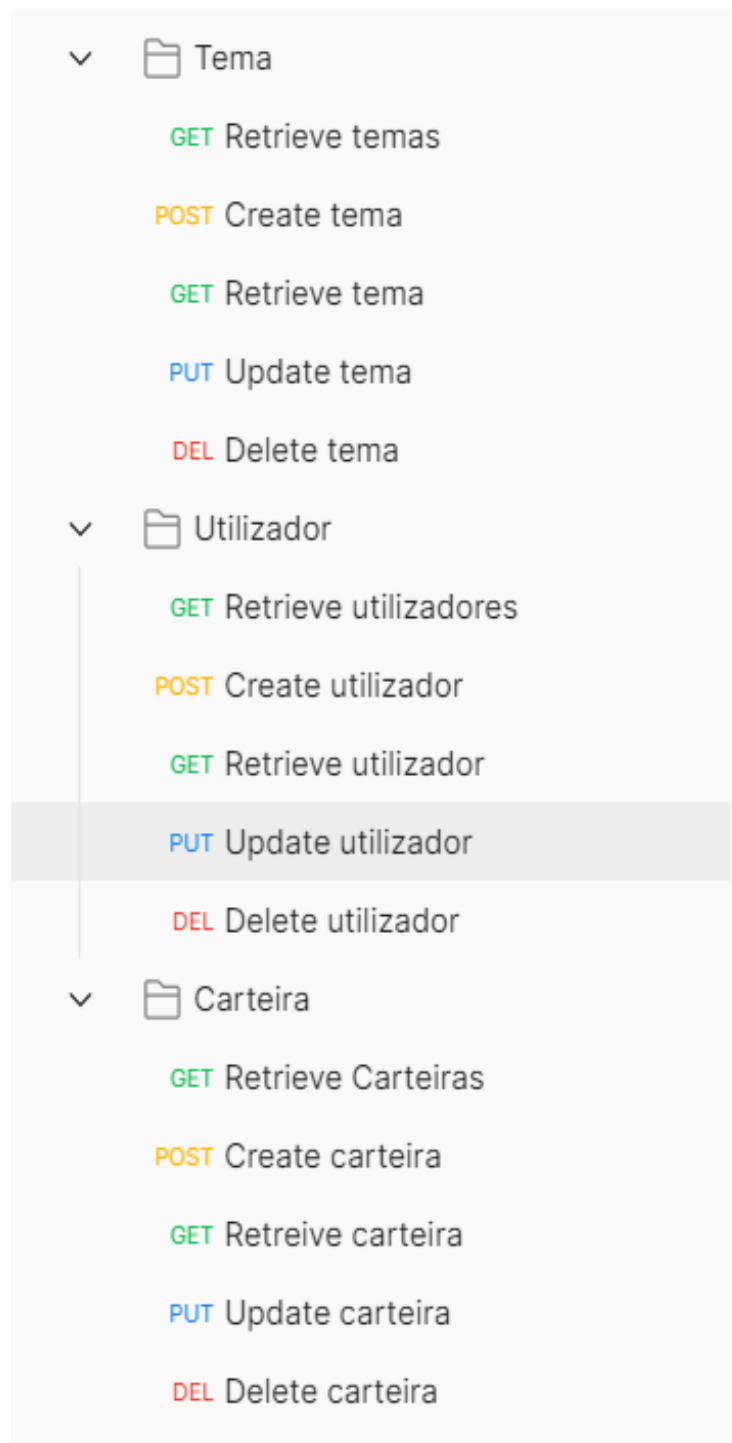
Esta é a configuração necessária por defeito para o funcionamento do MySQL do XAMPP.

Representação do painel de controlo do XAMPP a funcionar corretamente.



POSTMAN

Aquando da finalização da parte da construção da Base de Dados, foi-nos solicitado que fizéssemos a implementação no POSTMAN



✓	📁 Filme	⋮
	GET Retrieve filmes	
	POST Create filme	
	GET Retrieve filme	
	PUT Update filme	
	DEL Delete filme	
✓	📁 Cinema	
	GET Retrieve cinemas	
	POST Create cinema	
	GET Retrieve cinema	
	PUT Update cinema	
	DEL Delete cinema	
✓	📁 Livro	
	GET Retrieve livros	
	POST Create Livro	
	GET Retrieve livro	
	PUT Update livro	
	DEL Delete livro	
✓	📁 Servicos de aluger	
	GET Retrieve servicos	
	POST Create servico	
	GET Retrieve servico	
	PUT Update servico	
	DEL Delete servico	

Conclusão

Tal como foi sendo dito durante este relatório, existiram alguns problemas, nomeadamente nos GET. Chegamos á conclusão que foi devido a erros na construção da lógica da Base de Dados, nomeadamente na relação $m : n$. Estes erros são críticos para o bom desenvolvimento da Base de Dados e tal situação só tem uma solução: Começar do zero e fazer tudo de novo.

Se possível, gostaríamos de ter a possibilidade de corrigir estes erros, pois estes devem-se a problemas de gestão de tempo, sendo que dois dos elementos do grupo de trabalho se encontram no final do curso (estágio/projeto) e trabalharem, revelou-se impossível a realização do trabalho mais cedo.