Instituto superior de Engenharia de Lisboa

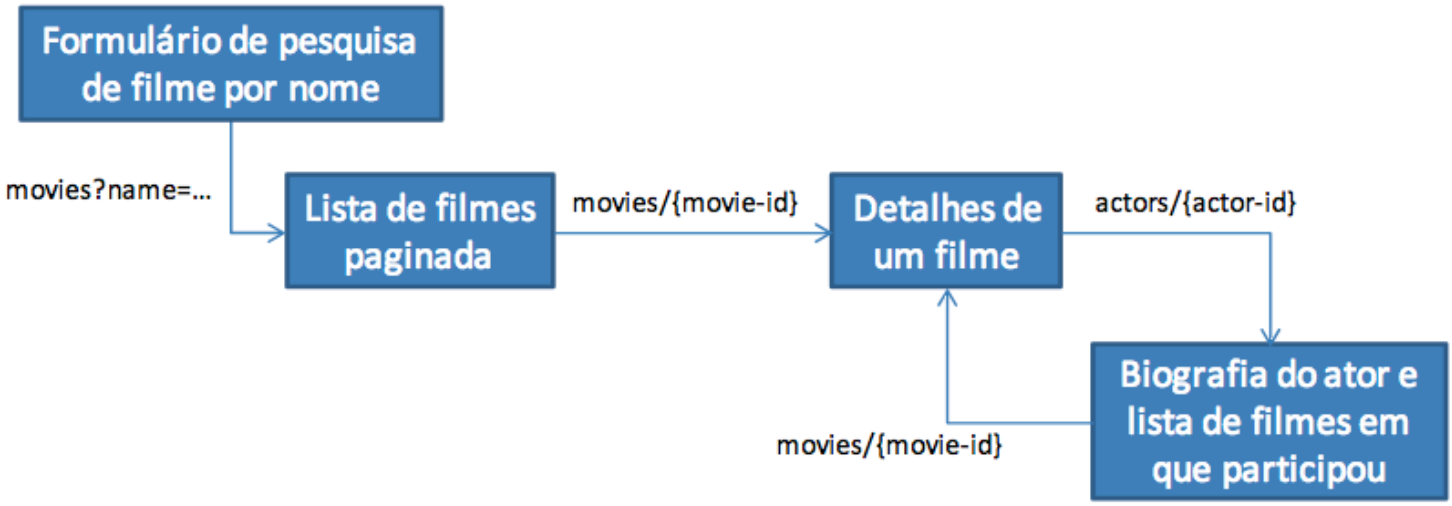
Programação de Internet

Grupo 5 :
Gonçalo Santos, 40599
João Leitão, 41510
Ricardo Moreira, 40677

Índice

Aplicação Web, fase I

Esquema de navegação



Endpoints

URI *endpoint*	Descrição do resultado	Exemplo
/search?name={query}	Página HTML com a listagem paginada dos filmes que correspondem à pesquisa de {query}.	/search?name=Ninja
/movies/{movie-id}	Página HTML com os detalhes de um filme, que incluem o nome, a imagem correspondente ao poster do filme e a informação do realizador e dos atores participantes no filme, com o nome do ator e o nome da personagem interpretada. O nome do actor é um link para a página com os seus detalhes.	/movies/36326
/actors/{actor-id}	Página HTML com os detalhes de ator, com a sua biografia e a lista de filmes em que participou. O nome de cada filme deve constituir uma hiperligação para a página de detalhe do filme.	/actors/680

MovieWebApp

Contem um conjunto de *handlers/actions* que atendem os pedidos exigidos por todos os *endpoints*.
Foi criado um objeto para corresponder à chamada de cada *endpoint*. Estes objetos fazem um pedido a um serviço e de seguida obtêm um *template* preenchendo-o com o retorno do serviço.

Model – Entidades de Domínio

- Ator
- Movie

Ator

id: Int
name: String
biography: String
profile_path: String
filmography: String

Movie

id: Int
title: String
vote_average: Double
release_date: Date
poster_path: String
cast: Actor[ ]
directorName: String

Serviços

- movieService
- cacheService

MovieService, onde é feita a conexão entre a aplicação e a *API*, possui todo o grosso do código.

CacheService, onde é feito o armazenamento dos detalhes de todos os filmes e atores para com isto caso haja pedidos subsequentes ao mesmo *URI* não serão desencadeados novos pedidos à *API*.

Fonte de Dados

RESTful API

- https://api.themoviedb.org

Aplicação Web, fase II

Pretende-se gerar um novo projeto através do *express-generato*.
Dividindo a aplicação em 2 componentes:

- app.js – Instância de *express* como *Pipeline de Middlewares*
- bin\www – servidor *HTTP* que usa app.js como *listener*

Tabelas na CouchDb

User

email: String
list: List[ ]
name: String
password: Pass
username: String

List

listId: Int
name: String
results: Movie[ ]

Nota: Todas as bases de dados contêm os campos *_id* e *_rev* que são gerados automaticamente pela *couchDb*

User (PK – username)

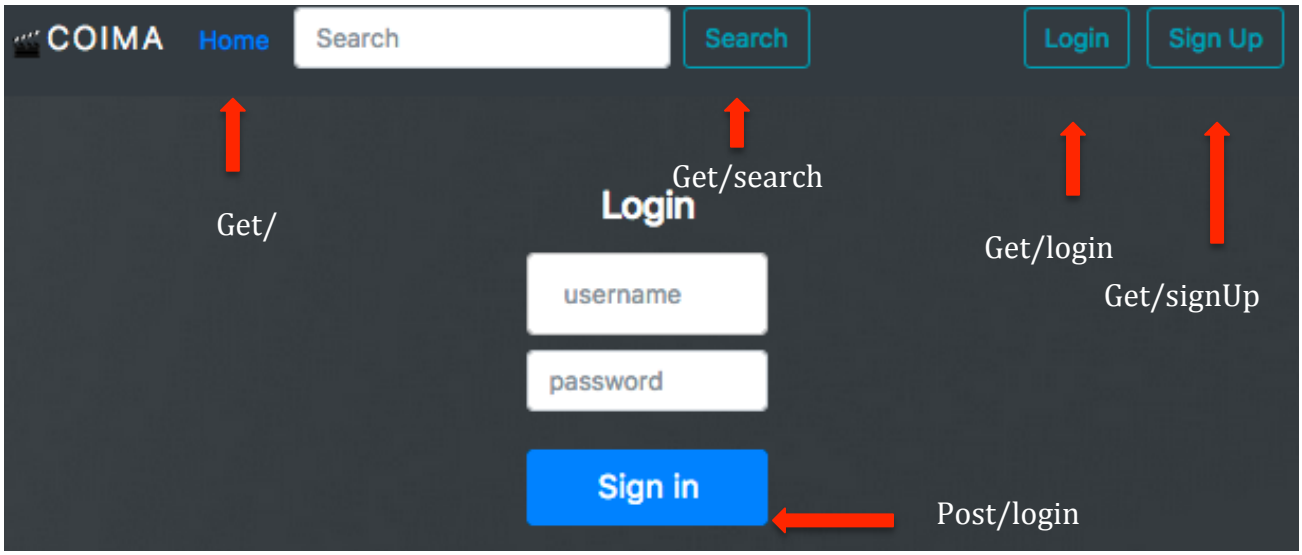
- email – e-mail introduzido pelo utilizador
- list – conjunto de listas favoritas
- name – nome escolhido pelo utilizador
- password – palavra-passe escolhida pelo utilizador
- username – nome de usuário escolhido pelo utilizador

List (PK – listId)

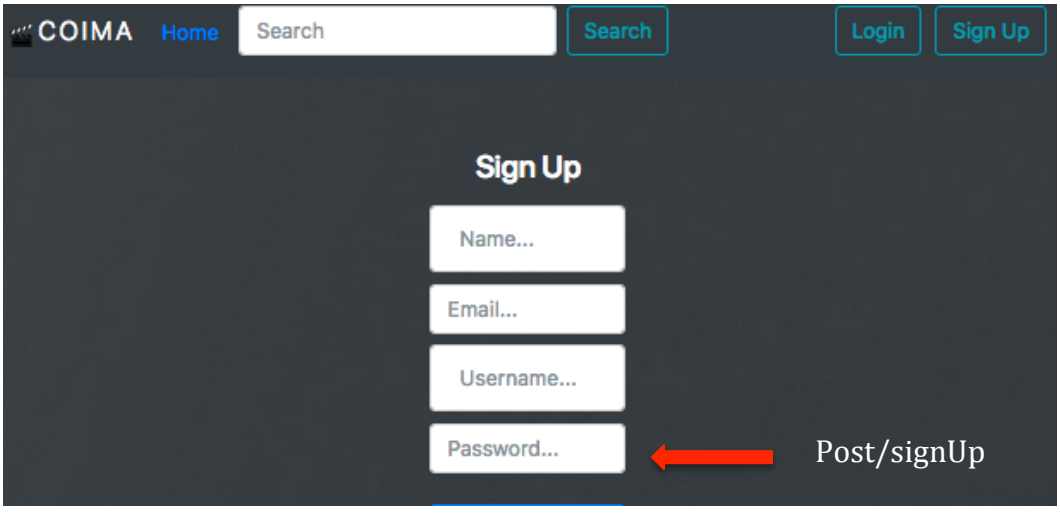
- listId – identificador gerado pelo valor da data de criação da lista
- name – nome escolhido pelo utilizador
- results – conjunto de filmes adicionados pelo utilizador

Registo e autenticação de utilizadores

Todos os utilizadores tem acesso à página de *login* onde podem introduzir os dados da sua conta para com isto ficarem logados. Caso não tenham conta podem criar uma.



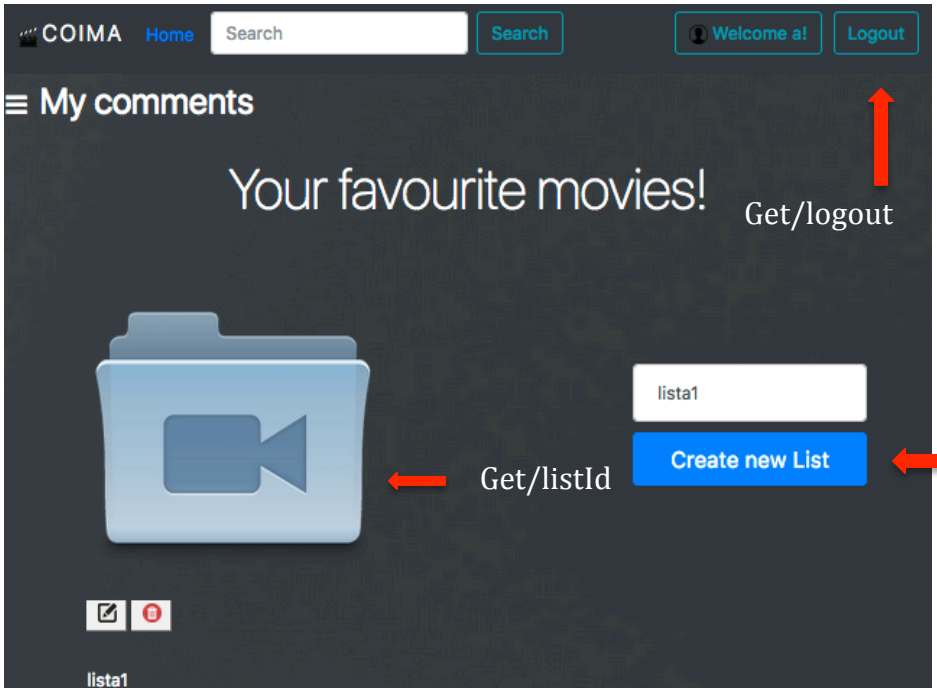
- **Get/** – apresenta a página *home* ;
- **Get/login** – apresenta a página onde se introduzem as credências para a autenticação ;
- **Post/login** – verifica se na base de dados existe algum utilizador que corresponda aos dados introduzidos no formulário. Em caso de sucesso redireciona para a página do *user* ;
- **Get/signUp** – apresenta a página onde se pode fazer o registo de um novo *user* ;
- **Get/search** – faz um pedido à *API* com o nome do filme introduzido no formulário e redireciona para a página onde expõe todos os filmes encontrados ;
- **Post/signUp** – verifica se na base de dados o *username* obtido é diferente do dos utilizadores já existentes. Confere também se o e-mail colocado possui um '@'. Por fim caso todos os parâmetros sejam aceites adiciona um novo utilizador na base de dados.



Gestão dos eventos criados por um utilizador

1)

Para cada utilizador pode ser realizada a operação de adição de listas. Esta apenas pode ser feita na página do *user* via formulário.



- **Get/listId** – apresenta a página onde exibe os filmes da lista1 ;
- **Post/addLst** – cria uma lista nova com o nome introduzido pelo utilizador e coloca-a na *couchDb* ;
- **Get/logout** – apresenta a página onde o utilizador pode sair da sua conta

2)

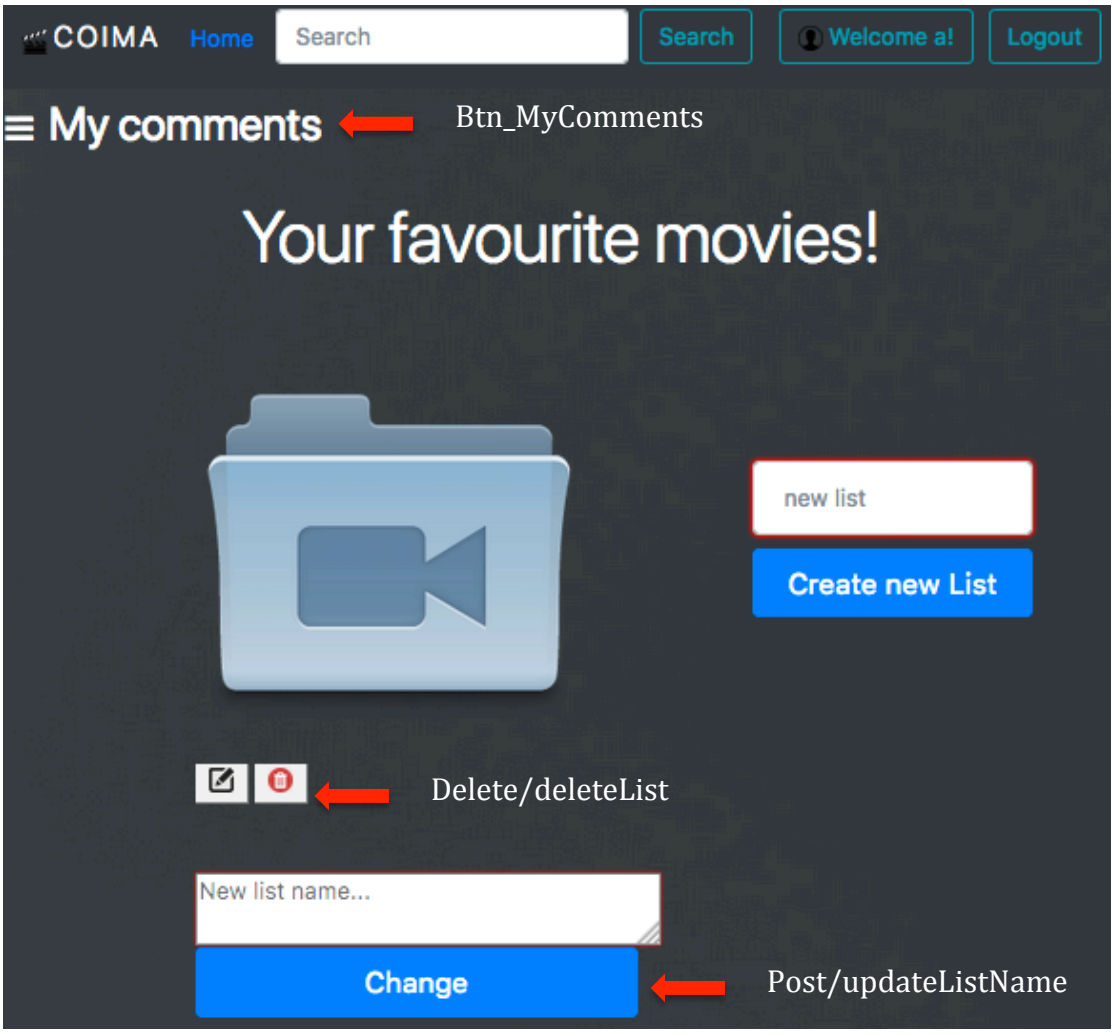
Cada utilizador pode também adicionar filmes nas suas próprias listas

Post/addMovie – adiciona à base de dados no campo *results* da lista escolhida., neste caso, lista1.
Desta maneira as informações deste filme poderão ser visualizadas na página da lista1 ;

Aplicação Web, fase III

Todas as alterações visuais realizadas nesta fase foi usado a técnica *AJAX* e *DOM* para realizar alterações dinâmicas às páginas HTML através de código *JavaScript* de cliente.

A edição e remoção de listas de filmes foi realizada implementando os seguintes *endpoists*.



- **Delete/deleteLis**t – obtém o *id* da lista e remove a mesma da base de dados ;
- **Post/updateListName** – obtém o novo nome introduzido pelo utilizador no formulário e reescreve o mesmo no campo *lst.name* da base de dados ;
- **Btn_MyComments** – apresenta a lista de comentários feitos até à data por este utilizador ;

Alterações na CouchDb

User

email: String
list: List[ ]
name: String
password: Pass
username: String
comments: Comment[ ]

Comment

Id: Int
username: String
comment: String
title: String
movieId: Int
responses: Comment[ ]

Movie

movieId: Int
comments: Comment[ ]

User (PK – username)

- Componentes explicadas anteriormente
- comments – conjunto de comentários realizados pelo utilizador

Comment (PK – id)

- id – identificador gerado pelo valor da data de criação do comentário
- username – nome do utilizador que criou o comentário
- comment – corpo do comentário
- title – titulo do filme comentado
- movieId – identificador do filme comentado
- responses – conjunto de respostas a este comentário

Movie (PK – movieId)

- movieId – identificador do filme
- comments – conjunto de comentários sobre o filme

Comentários a um filme

Os comentários e respostas de um filme só podem ser feitos por utilizadores registados mas são visíveis por qualquer tipo de utilizador. As respostas têm uma margem adicional do lado esquerdo para com isto tornar perceptível a diferença entre o comentário a que está a responder.

- **Post/comments** – cria um objecto comentário com o texto introduzido pelo utilizador, o *username* do utilizador, o titulo do filme e o identificador do filme. Por fim escreve o comentário na base de dados e adiciona à página via *DOM* ;
- **Post/reply** – cria um objecto em tudo semelhante ao explicado anteriormente, com 2 diferenças :
    - Onde é guardado na base de dados ;
    - Na adição à página via *DOM* ;
- **Btn WriteComment** – apresenta a caixa de texto para ser realizada a resposta a um comentário, caso seja clicado novamente a caixa de texto esconde-se ;

Quando é criado um comentário o mesmo é guardado na base de dados sobre o objecto do filme e utilizador correntes.

No caso de resposta é guardada como resposta ao comentário, no filme e no utilizador que tinha feito o comentário. Também é adicionada como comentário normal no utilizador que realizou a resposta, desta forma cada utilizador tem acesso a todas as respostas que fez até á data.