Phase 1

fbfreitas edited this page on Mar 2 · 1 revision

Goals

The Software Laboratory project is comprised by the analysis, design and implementation of an information system to manage cinemas and associated information. Its development is divided into 4 phases, with incremental requirements published before the beginning of each phase.

Domain

- A *cinema* is characterized by:
  - a name;
  - the city where it is located;
  - a set of theaters.
- A *theather* is associated to one *cinema* and is characterized by:
  - a name;
  - the number of available seats;
  - the number of rows;
  - the number of seats per row;
- A *movie* is characterized by:
  - a title;
  - a release year;
  - a duration.
- A *session* has a date (which includes the time of day) and is associated with a theater and a movie. Sessions in the same theater cannot overlap temporally.

Management application

On the first phase, the interaction with the information system is done via commands for a console application. Each application execution performs one and only one command, presenting the command result on the standard output.

Each command is defined using the following generic structure

```
{method} {path} {parameters}
```

where

- the `method` defines the type of action to perform (e.g. `GET` or `POST`).
- the `path` defines the resource on which the command is executed (e.g. `/cinemas` or `/cinemas/123`).
- the `parameters` is a sequence of `name=value` pairs, separated by `&` (e.g. `name=City+Campo+Pequeno&city=Lisboa`).

Example: `POST /cinemas name=City+Campo+Pequeno&city=Lisboa`

Phase 1 requirements

For phase 1, the management application must support the following commands.

Movie management

Clone this wiki locally

https://github.com/isel-lei

Clone in Desktop

- `POST /movies` - creates a new movie, given the following parameters
  - `title` - the title name.
  - `releaseYear` - the movie's release year.
  - `duration` - the movie's duration (in minutes).

  This command returns the movie unique identifier.

- `GET /movies` - returns a list with all movies.

- `GET /movies/{mid}` - returns the detailed information for the movie identified by `mid`.

Cinema management

- `POST /cinemas` - creates a new cinema, given the following parameters
  - `name` - the cinema name.
  - `city` - the city where the cinema is located.

  This command returns the cinema unique identifier.

- `GET /cinemas` - returns a list with all cinemas.

- `GET /cinemas/{cid}` - returns the detailed information for the cinema identified by `cid`.

- `POST /cinemas/{cid}/theaters` - adds a new theater to a cinema, given the following parameters
  - `name` - theater name.
  - `rows` - theater number of rows.
  - `seats` - number of seats per rows.

  This command returns the theater unique identifier.

- `GET /cinemas/{cid}/theaters` - returns the list of theaters of the cinema identified by `cid`.

- `GET /cinemas/{cid}/theaters/{tid}` - returns the detailed information for the theater identified by `tid`.

Session management

- `POST /cinemas/{cid}/theaters/{tid}/sessions` - creates a new session, given the following parameters
  - `date` - session date
  - `mid` - movie identifier

- `GET /cinemas/{cid}/theaters/{tid}/sessions` - returns a list with all the sessions in the theather identified by `tid` on the cinema identified by `cid`.

- `GET /cinemas/{cid}/sessions` - returns a list with all sessions in cinema identified by `cid`.

- `GET /cinemas/{cid}/sessions/{sid}` - returns the detailed information for the session identified by `sid`.

- `GET /cinemas/{cid}/sessions/today` - returns a list with all sessions for the current day in the cinema identified by `cid`.

- `GET /cinemas/{cid}/theaters/{tid}/sessions/today` - returns a list with all sessions for the current day in the cinema and theater identified by `cid` and `tid` respectively.

Non-functional requirements

The following non-functional requirements will be highly valued.

- Readability.
- Testability, including on machines not belonging to the development team.

Delivery date

The project must be delivery until March 24 (end of week 5), via the creation of a 0.1.0 tag on the GitHub repository.