Movies challenge

Welcome to the tech challenge for Lodgify's Backend developers candidates.

Please follow the instructions included here. When you are finished, zip it, use a cloud service to upload it and send the link to us by **email**. (Remeber to set the zip file permission to **Read-only to anyone**, we suggest Google Drive, but Dropbox, OneDrive, etc is also ok.)

Thank you and good luck!

Context

We want a C# .Net Core Web Application API project that meets our requirements. We provide you with the solution skeleton and a few features implemented to save time.

You will find the Data layer is implemented and is instructed to be an In-Memory Database.

The application represents a Cinema. We want to manage the showtimes of the cinema, getting some data from the **Provided API.**

The test includes a docker-compose with Redis and the provided Api, you will need Docker to be able run them.

We only want the following features:

- Create showtimes.
- Reserve seats.
- Buy seats.

Note that we also expect you to provide test coverage for the implemented features.

Implementation instructions

Starting the API

- You will need Docker in order to use this API and then run the next command:

```
docker-compose up
```

- By default, the provided API will run on *http://localhost:7172/swagger/index.html* , https://localhost:7443/swagger/index.html
- For GRPC use the **HTTPS** port
- And Redis in the default port.
- When you end the test

```
docker-compose down
```

Commands and queries

- **Create showtime**

  Should create showtime and should grab the movie data from the ProvidedApi.

- **Reserve seats**

  - Reserving the seat response will contain a GUID of the reservation, also the number of seats, the auditorium used and the movie that will be played.
  - After 10 minutes after the reservation is created, the reservation is considered expired by the system.
  - It should not be possible to reserve the same seats two times in 10 minutes.
  - It shouldn't be possible to reserve an already sold seat.
  - All the seats, when doing a reservation, need to be contiguous.

- **Buy seats**

  - We will need the GUID of the reservation, it is only possible to do it while the seats are reserved.
  - It is not possible to buy the same seat two times.
  - Expired reservations (older than 10 minutes) cannot be confirmed.
  - We are not going to use a Payment abstraction for this case, just have an Endpoint which I can use to Confirm a Reservation.

API communication with ProvidedApi

You can use the HTTP API or the GRPC API, you should check the Swagger for more info. We know that the GRPC implementation is faster. But it is not working right now. You can fix it and use it.

The solution includes the proto of Provided API and a small piece of code that tries to connect with the GRPC API, but it does not work.

Cache

We will like to have a cache layer to cache the response from the Provided API because the API is slow and fails a lot. We will like to call the API and in case of failure try to use the cached response. The cache should use the Redis container provided in the docker-compose.yaml

Execution Tracking

We want to track the execution time of each request done to the service and log the time in the Console. By default, we set the loggers to log in to the Console, so you only need to worry where to put the Logger in the code.

Provided API

We know that **Provided API** may have some configuration issues, and we will like them to be found and fixed, if possible.

Add the Request to cUrls file

We added a file next to this readme named `cUrls.txt`. Please add a curl command for each of the commands and queries that you implemented to this file.

Feedback

You can add any feedback you want to send us in the file `Feedback.md` located next to this file. If you find something you cannot do or fix, add it to this document.