Phase 4

fbfreitas edited this page 23 days ago · 7 revisions

Phase 4

Introduction

This document describes the requirements for the fourth and final phase of the Software Laboratory project.

Requirements

Using PostgreSQL

Use the PostgreSQL Relational Database Management System, instead of SQL Server.

- Install PostgreSQL on the development machines.
- Obtain and use a JDBC driver for PostgreSQL.
- Change both the DDL and DML statements to use the PostgreSQL dialect.
- Use the `JDBC_DATABASE_URL` environment variable to configure the PostgreSQL connection settings.

Hosting on Heroku

Host the application on the Heroku.

- Create an Heroku free account. Heroku is an example of a Platform-as-a-Service (PaaS) provider.
- Install the Heroku CLI (Command Line Interface) on all the development machines.
- On the Heroku web site, create a new application
  - The name should follow the following structure: `isel-ls-1718-2-li4<turma>-g<número-do-grupo>` .
  - Select the "Europe" region.
- On the application home page, provision the "Heroku Postgres" add-on using the "Hobby Dev - Free" plan.
  - The `JDBC_DATABASE_URL` environment variable will be automatically added to Java the execution environment.
- Change the `build.gradle` file to include the `application` plug-in, define the startup class, and define the `stage` task.
- Create the `settings.gradle` file with the project name.
- Create the `Procfile` file with the Heroku project name and the activation string.
- On the command line, do `heroku login` .
- On the command line, inside the project root folder, do `heroku git:remote -a isel-ls-1718-2-li4XX-gXX` . This will create a Git remote pointing to the Heroku repository.

Use the Movies API

- When creating a movie, use the Movies API to fetch the movie information based on the given movie identifier, which becomes the only mandatory parameter.

Logging

Add logging of any relevant event information into the developed application. Use the Simple Logging Facade for Java (SLF4J), configured with the binding for the Simple implementation. See TimeServlet.java for an example.

Pages 12

Home
calendar
Demonstration
discussion_calendar
environment
Introduction
Phase 0
Phase 1
Phase 2
Phase 3
Phase 4
turnos

Clone this wiki locally

https://github.com/isel-lei

Clone in Desktop

Resource creation via the HTTP interface

The main goal for this requirement is to add support for the commands with `POST` method in the HTTP interface. For that, the representations for the following resources should be augmented with HTML forms:

- `/cinemas` - form to add a new cinema.
- `/movies` - form to add a new movie.
- `/cinemas/{cid}` - form to add a new theater.
- `/cinemas/{cid}/theaters/{tid}` - form to add a new session.
- `/cinemas/{cid}/theaters/{tid}/sessions/{sid}` - form to add a new ticket

Each one of these forms must, when submitted, send a POST request to the associated resource. For instance:

- The representation returned on a `GET /cinemas` request should contain a form with the field `name`.
- This form, when submitted, must send a `POST` request to `/cinemas` containing this field in the body.
- If the `POST` request is successful, i.e. a new cinema was created, its response should be a `303 See Other` with the `Location` header pointing to the created cinema (e.g. `/cinemas/some-id`).

Delivery

The completed project must be delivery until June 9 2018 (end of week 15), via the creation of a 1.0.0 tag on the GitHub repository.