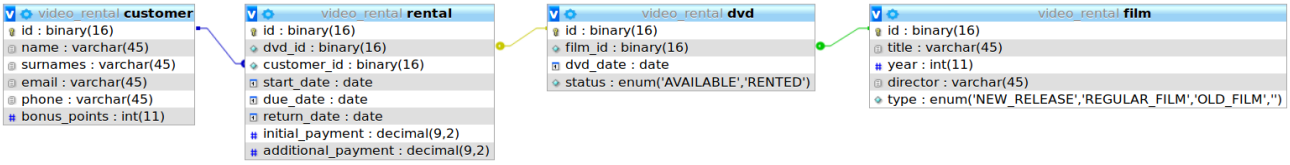
SINGULAR COVER
Technical test - Senior Backend Engineer

Video Rental REST API

- The application is written in Java using Spring framework with the following technologies: Spring Boot, Spring data JPA (Hibernate), MySQL database, Google Gson to convert Java Objects to JSON and back, and Gradle to build the project.
- I choose a relational database since data is consistent, well structured and defined. Also, due to the type and size of the business, a massive growth is not expected so scalability should not be an issue.
- The database has four tables as shown in the diagram below. Films an Customers tables contain generic info for each of them. Dvd table stores all tapes in the rental store, so there can be several dvd tapes of a single film title. Rental table stores information for each rental in store, both the active or ongoing ones and the historical ones that are already terminated.



- All id for each of the tables are implemented using uuids.
- NEW_RELEASE rental price is the total number of days multiplied by the PREMIUM_PRICE (3 euro). REGULAR_FILMS rental price is PREMIUM_PRICE for the first 3 days, and BASE_PRICE (1 euro) multiplied for the total extra days. OLD_FILMS price is the BASE_PRICE (1 euro) for the first 5 days, and that same price multiplied for the total extra days.
- The application keeps track of customers bonus points and adds them to the customer's data. However, there is not any implementation on how to spend or use those points. That would be a feature to do.
- The project provides a **/VideoRental/src/main/resources/video_rental.sql** file to create and populate the database.

The API exposes the following operations:

- **CUSTOMER TABLE**

POST
http://localhost:8080/customer

body
```
{
"name": "Aurora",
"surnames": "Capmany",
"email": "auroracapmany@gmail.com",
"phone": "675342009",
"bonusPoints": "0"
}
```

Saves given JSON object as a new customer in database, and returns saved object.

GET
http://localhost:8080/customer

Returns a list of all customers from database alphabetically sorted by surnames.

GET
http://localhost:8080/customer/0cc8fe93-a7f6-4ebf-a03b-b560d26adfe6

Returns the customer from database with given id.

PUT
http://localhost:8080/customer/eb9826f5-f8fa-4a0b-9e06-caa6bad632bb

body
```
{
"id": "eb9826f5-f8fa-4a0b-9e06-caa6bad632bb",
"name": "Criss",
"surnames": "Fabregas",
"email": "cristinafabregas@gmail.com",
"phone": "666761252",
"bonusPoints": "5"
}
```

Updates customer in database with given id and parameters in JSON body, and returns updated object. For instance, in the example above, changing customer's name from 'Cristina' to 'Criss'.

DELETE
http://localhost:8080/customer/b1047afe-c065-4672-b1bb-1f3ed7bd0064

Deletes customer with given id, and returns deleted object. **Deletes customer only if it is not linked to any rental.** Otherwise, throws 500 Internal Server Error. TODO: implement proper exception handling.

- **FILM TABLE**

POST
http://localhost:8080/film

Body
```
{
"title": "The Godfather",
"year": "1972",
"director": "Francis Ford Coppola",
"type": "OLD_FILM"
}
```

Saves given JSON object as a new film in database, and returns saved object.

GET
http://localhost:8080/film

Returns a list of all films from database alphabetically sorted.

GET
http://localhost:8080/film/c4c6412d-e05b-4496-9931-d536c9669d44

Returns the film from database with given id.

GET
http://localhost:8080/film/keyword/the

Returns a list of films that contain given keyword, for instance 'the' in the example above.

PUT
http://localhost:8080/film/e34d980d-df0b-49be-a33d-ce313917a832

body
```
{
"title": "Toy Story 1",
"year": 1995,
"director": " John Lasseter",
"type": "REGULAR_FILM"
}
```

Updates film in database with given id and parameters in JSON body, and returns updated object. For instance, in the example above, changing Toy Story 4 to 1.

DELETE
http://localhost:8080/film/b8e68424-8bff-4b73-a0c3-9209f357ce00

Deletes film with given id, and returns deleted object. **Deletes film only if it is not is linked to any dvd, and by extension to any rental.** Otherwise, throws 500 Internal Server Error. TODO: implement proper exception handling.

- **DVD TABLE**

POST
http://localhost:8080/dvd

body
```
{
  "filmId": "c4c6412d-e05b-4496-9931-d536c9669d44"
}
```

Saves a new dvd in the database containing the film provided by the JSON id string, and returns the saved object.

GET
http://localhost:8080/dvd

Returns a list of all dvd from database alphabetically sorted by title. There might be several dvds of a same film.

GET
http://localhost:8080/dvd/78536931-0b55-48ed-9fa5-c9240f4131ee

Returns the dvd from database with given id.

GET
http://localhost:8080/dvd/film/a3bc4e8c-0187-40dc-8ffa-c1bf772f9f39

Returns a list of dvds of the same film, given the film id.

PUT
http://localhost:8080/dvd/41f91bd4-0f62-4c24-ac34-4bf302dd70c4

body
```
{
  "filmId" : "4e9f4cf6-3d83-4f16-a922-3d361652c5f8",
  "dvdDate" : "2011-11-11"
}
```

Updates dvd in database with given film and dvd date in JSON body, and returns updated object.

DELETE
http://localhost:8080/dvd/fcc6a80a-1b65-4552-b3dd-b77422a28dcc

Deletes dvd with given id, and returns deleted object. **Deletes dvd only if it is not is linked to any rental.** Otherwise, throws 500 Internal Server Error. TODO: implement proper exception handling.

- **RENTAL TABLE**

POST
http://localhost:8080/rental

body
```
{
"dvdId": "57621294-fcbc-4dec-88ae-9b6b86e42555",
"customerId": "be9a4e32-1e3d-42a2-ba2e-32a14497f0ed",
"numberDays": "5"
}
```

Saves a new rental in the database with the data provided by the JSON string, and returns the saved object.

GET
http://localhost:8080/rental

Returns a list of all active (ongoing) dvd rentals, discards all historical ones that have already been returned.

GET
http://localhost:8080/rental/history

Returns a list of all dvd rentals, included historical ones, sorted by date from most recent to the oldest ones.

GET
http://localhost:8080/rental/c8099fd4-0e3a-4933-b3b2-38bab7268ab2

Returns the rental from database with the given rental id.

GET
http://localhost:8080/rental/dvd/b08af968-2a9e-40df-8048-75b9fdd50008

Returns the rental of the dvd with the given dvd id.

GET
http://localhost:8080/rental/history/5c3ab96a-fa78-4c81-a179-76249e1a4e3a

Returns a list of all rentals that a dvd has had historically, sorted by date from most recent to the oldest ones.

GET
http://localhost:8080/rental/customer/929b6ba3-a592-4ea9-bd68-17ee46efbe2d

Returns a list of all active (ongoing) rentals of a customer by a given customer id.

PUT
http://localhost:8080/rental/terminate/78536931-0b55-48ed-9fa5-c9240f4131ee

Terminates an active rental by dvd id, and calculates if return date exceeds due date and, if so, calculates an additional payment. TODO: implement proper exception handling to avoid 500 Internal Server Error when a rental that is already terminated is called

PUT
http://localhost:8080/rental/d75be1ca-7026-4206-880c-14aea60ea7ee

body
```
{
"startDate": "2019-08-01",
"dueDate": "2019-08-06",
"initialPayment": 15,
"additionalPayment": 0
}
```

Updates rental in database with given data in JSON body, and returns updated object. For instance, in the example above, start and return date and initial payment are updated in rental info.

DELETE
http://localhost:8080/rental/ae6dc272-d7ec-41b2-8541-94cf29e3bb9f

Deletes rental with given id, and returns deleted object.