ce**N**tralized, s**E**rvice orien**T**ed, sel**F** adaptive, app**LIC**ation for movie **S**treaming - **NETFLICS**

Software Engineering for Autonomous Systems/Service Oriented Software Engineering

Valentina Cecchini – 255596

Stefano Valentini – 254825

demo video:

*\*full resolution images can be found along with this document*

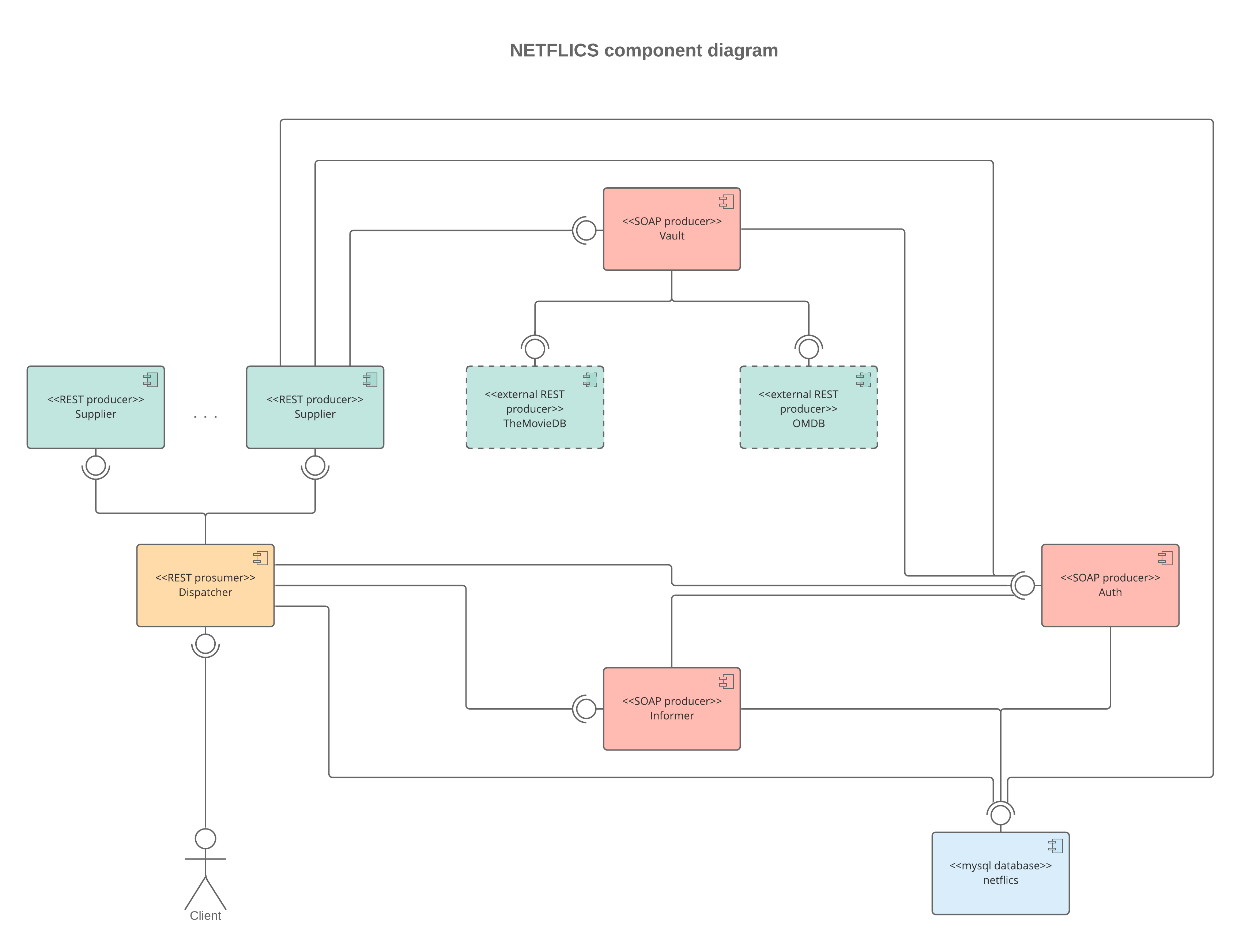
**OVERVIEW**

The aim of this project is to build a video streaming application project based on web services, with particular focus on self-adaptivity.

The system exploits the interactions between services and to achieve a great level of availability while avoiding waste of resources through the implementation of a load-balancing mechanism that is able to “turn off” certain components of the system when not needed and to wake them back up in case of an high volume of incoming requests is detected.

**E-R MODEL**

**SYSTEM’S ARCHITECTURE**



**SOAP** services:

* **Auth**: is the service that is responsible for checking the user’s credentials to allow it to log in/out and access the system’s functionalities, endpoints**:**
  + **logIn(email, password) -> {result, role, token}**

checks the user’s credentials and performs the log in action, returning the role and a new token for the user to use

* + **logOut(token) -> {result}**

deletes the session associated to the sent token

* + **checkToken(token) -> {result, role}**

checks the token and returns the role of the user

* **Vault**: is the service that is responsible for keeping all the movie files on disk and to send them to the suppliers that request them**:**
  + **getMovie(imdb\_id, token) -> {movie, result}**

returns the movie data associated to the imdb\_id, if the user is authorized

* + **addMovie(imdb\_id, movie, token) -> {result}**

saves to disk the movie data that it has received, retrieves the metadata of the movie from TheMovieDB and OMDB and stores them into the db.

* **Informer:** is the service that is responsible for retrieving information from the database to be used for the render of the webclient gui (such as the list of the most viewed movies, etc.).

**REST** services

* **Supplier**:is the service that is responsible for retrieving the movie data from the Vault service and make it available for the Dispatcher; endpoints:
  + **GET /{token}/movie/{id}**

returns the requested movie data as a StreamingOutput

* + **GET /{token}/availability**

returns the current system resource occupation

* + **POST /{token}/movie/{id}**

commands this supplier to fetch the movie data from the Vault service

* + **POST /{token}/wakeup**

commands this supplier to wake up

* + **POST /{token}/sleep**

commands this supplier to go to sleep

* **Dispatcher**: is the only component of the system that communicates with the client(s), it’s main task is the one that consist on permitting the client to stream the movie by requesting it from the “best” supplier (the mechanism by which this is accomplished is detailed in the “Load Balancing” section of this document; endpoints:
  + **PUT /{token}/movie/{imdbId}**

forwards the received movie data to the Vault.addMovie service to store it (reserved to admins)

* + **POST /{token}/logout**

calls Auth.logOut

* + **GET /movie/mostviewed**

returns the data that is retrieved by the Informer.mostViewed service

* + **GET /movie/bestones**

returns the data that is retrieved by the Informer.bestOnes service

* + **GET /{token}/movie/lastviewed**

returns the data that is retrieved by the Informer.lastViewed service

* + **GET /{token}/movie/stream/{imdbId}**

main functionality of this component, finds the “best” supplier and forwards to the client the StreamingOutput that is returned by it.

**LOAD BALANCING**

// TODO

**TESTS**