



BalticLSC Software Design

BalticLSC Admin Tool Technical Documentation, Design of the Computation
Application Language, Design of the BalticLSC Computation Tool
Version 1.0



Priority 1: Innovation

Warsaw University of Technology, Poland
RISE Research Institutes of Sweden AB, Sweden
Institute of Mathematics and Computer Science, University of Latvia, Latvia
EurA AG, Germany
Municipality of Vejle, Denmark
Lithuanian Innovation Center, Lithuania
Machine Technology Center Turku Ltd., Finland
Tartu Science Park Foundation, Estonia

BalticLSC Software Design

BalticLSC Admin Tool Technical Documentation, Design of
the Computation Application Language, Design of the
BalticLSC Computation Tool

Work package	WP5
Task id	A5.2, A5.3, A5.4
Document number	O5.2A
Document type	Design Specification
Title	BalticLSC Software Design
Subtitle	BalticLSC Admin Tool Technical Documentation, Design of the Computation Application Language, Design of the BalticLSC Computation Tool
Author(s)	Agris Šostaks (IMCS), Michał Śmiałek (WUT), Kamil Rybiński (WUT), Radosław Roszczyk (WUT)
Reviewer(s)	Daniel Olsson (RISE)
Accepting	Michał Śmiałek (WUT)
Version	1.0
Status	Final version

History of changes

Date	Ver.	Author(s)	Change description
11.11.2019	0.1	Agris Šostaks (IMCS)	Initial structure.
12.11.2019	0.2	Agris Šostaks (IMCS)	Behavioral model added.
19.11.2019	0.3	Agris Šostaks (IMCS)	Executive summary, Introduction added
12.12.2019	0.4	Agris Šostaks (IMCS)	DTO-s, Conclusion added
17.12.2019	1.0	Agris Šostaks (IMCS)	Finalized, after review.

Executive summary

The overall aim for the Baltic LSC activities is developing and providing a platform for truly affordable and easy to access LSC Environment for end-users to significantly increase capacities to create new innovative data-intense and computation-intense products and services by a vast array of smaller actors in the region.

BalticLSC Software Design document is based on work done within activities of the BalticLSC Work Package 5 (WP5) - Design of the BalticLSC Admin Tool (A5.2), Design of the Computation Application Language (A5.3), and Design of the BalticLSC Computation Tool (A5.4). It involves also workshops and technical sessions with participation of external experts and led mainly by the project's technology partners from - Warsaw University of Technology (WUT), Institute of Mathematics and Computer Science, University of Latvia (IMCS), RISE Research Institutes of Sweden AB (RISE). All workshops and meetings are held during Q1-Q2-Q3-Q4 2019.

BalticLSC Software User Requirements Specification document contains technical design for BalticLSC Software. It is intended to use by BalticLSC technology partners responsible for the development of BalticLSC Software.

Table of contents

History of changes.....	2
Executive summary	3
Table of contents	4
1. Introduction	5
1.1 Objectives and scope	5
1.2 Relations to Other Documents.....	5
1.3 Intended Audience and Usage Guidelines.....	5
2. Architecture Model.....	6
2.1 Cluster Node.....	6
2.2 Main Node: Computations	7
2.3 Main Node: Modules.....	9
2.4 Main Node: Accounts and Billing.....	10
2.5 FrontEnd: App Usage.....	11
2.6 FrontEnd: Network Administration.....	12
3. Behaviour Model.....	13
3.1 Show Cockpit and Show Jobs	13
3.2 Create Task.....	13
3.3 Activate Task.....	14
3.4 Activate Job.....	15
4. Data Model.....	16
5. Conclusion.....	19

1. Introduction

1.1 Objectives and scope

The objective of the document is to describe the design of the BalticLSC Software. The goal is to have the BalticLSC Software design specification which is usable for the BalticLSC Software Development. Since the BalticLSC Software development is being developed using agile methodology (Scrum) and the detailed design is developed during development sprints, this document is considered a “live” document and is a subject of changes after every sprint (approximately once per month).

1.2 Relations to Other Documents

The current BalticLSC Software User Requirements Specification document provides the design specification for the requirements specified in the BalticLSC Software User Requirements Specification (O3.3). It is used also by BalticLSC Platform Design (O4.3) and BalticLSC Operating System Design documents (O4.4)

1.3 Intended Audience and Usage Guidelines

This Design specification is the Output 5.2A as described in the Baltic LSC Application Form and intended for internal use within BalticLSC consortium as basis for future software development activities as well as for reporting purposes for local First Level Control and Baltic Sea Region Program.

2. Architecture Model

The architectural model describes the main components and API interfaces of the BalticLSC Software. The UML Component and Class diagrams have been used.

2.1 Cluster Node (Cluster)

These components are installed on each Cluster Node (the ones that perform the actual computations).

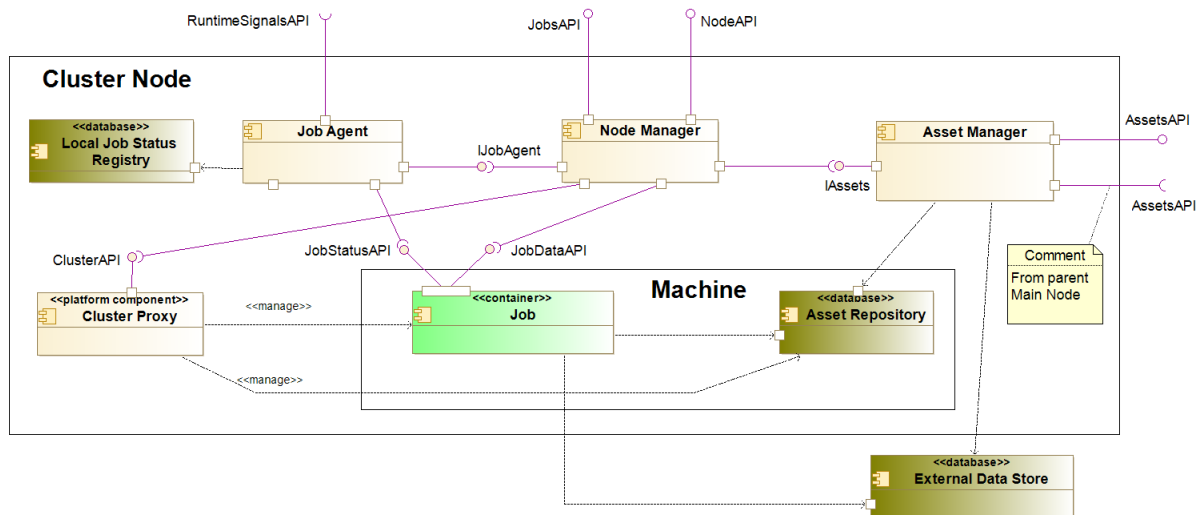


Figure 1. Component Diagram for the Cluster Node

Node Manager

Responsible for managing the node's activities: managing jobs assigned to the current node and managing availability of the whole node.

Job Agent

Responsible for constant monitoring of the status of jobs, and signaling their status to the Runtime Engine and to the Node Manager.

Job

A Docker component installed in the cluster by the Cluster Proxy and performing the actual computations, according to the appropriate Computation Module code.

Cluster Proxy

Responsible for management of containers within the Kubernetes cluster and monitoring of container statuses.

Asset Manager

Responsible for maintaining all the Assets. If the given Asset is not available in the local database, it is requested from the Asset Manager of the respective Master Node, or some underlying Cluster Node (in case it is part of a Master Node).

Asset Repository

A database responsible for storing the Asset contents (data and executables).

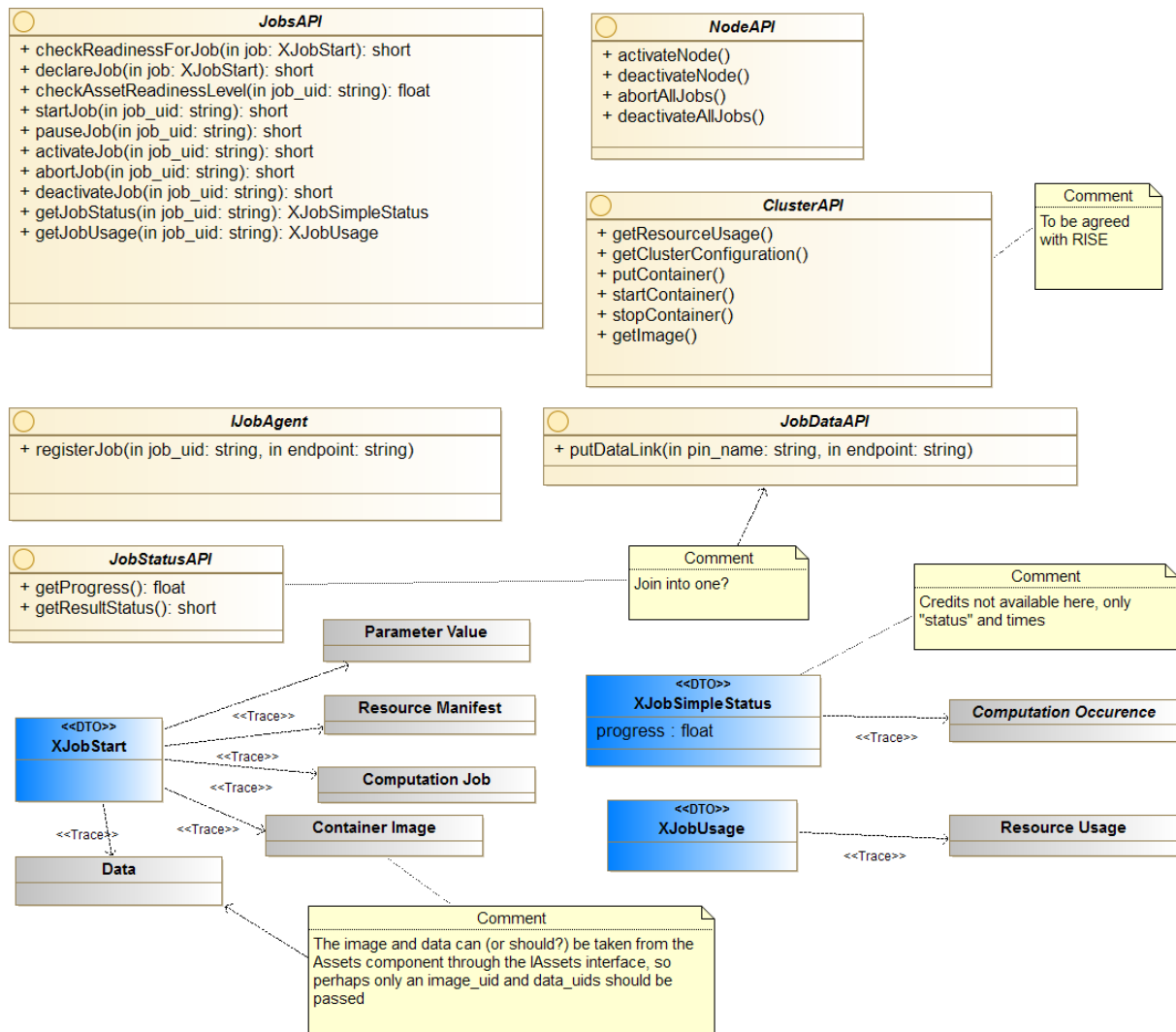


Figure 2. Class Diagram with interface definitions for the Cluster Node

2.2 Main Node: Computations

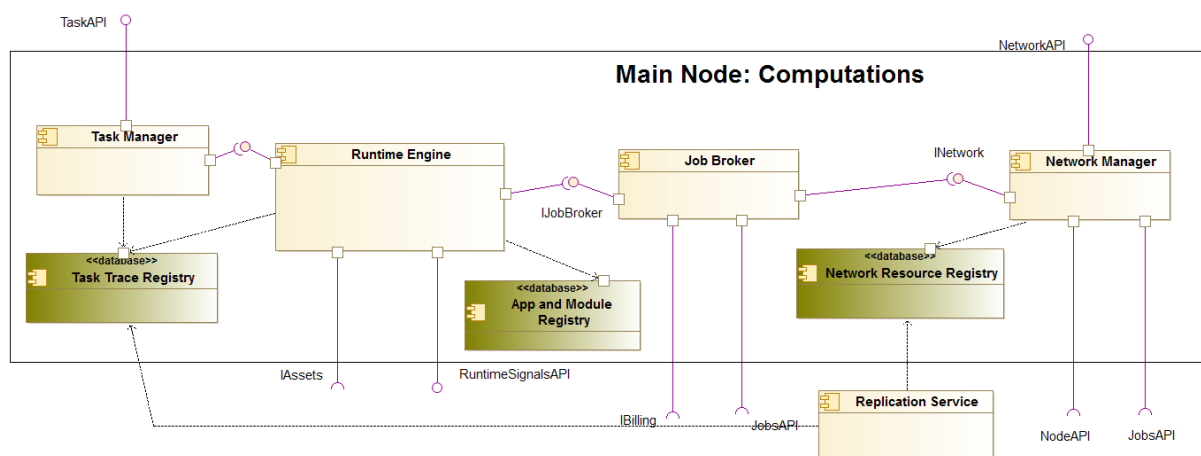


Figure 3. Component Diagram for the Main Node: Computations

Job Broker

Module responsible for distribution of computation task to proper computation resources. Asks for matching nodes from the Network Manager, and for billing information from the Billing component. Accesses the underlying Cluster Nodes through a set of instances of the JobsAPI.

Task Manager

Responsible for managing tasks, as requested by the users. Registers all the tasks in the Task Trace Registry.

Runtime Engine

Responsible for interpreting CAL programs. Allows to interact with the running program through the RuntimeEngineAPI. Registers signals from running jobs through the RuntimeSignalsAPI. Receives the program to be run through IAssets. Sends all the jobs for execution to the JobBroker.

Network Manager

Responsible for managing all the nodes within the network. Allows the user to edit the available resources, and to perform benchmarks on them. Accesses the underlying Cluster Nodes through a set of instances of the JobsAPI and NodeAPI.

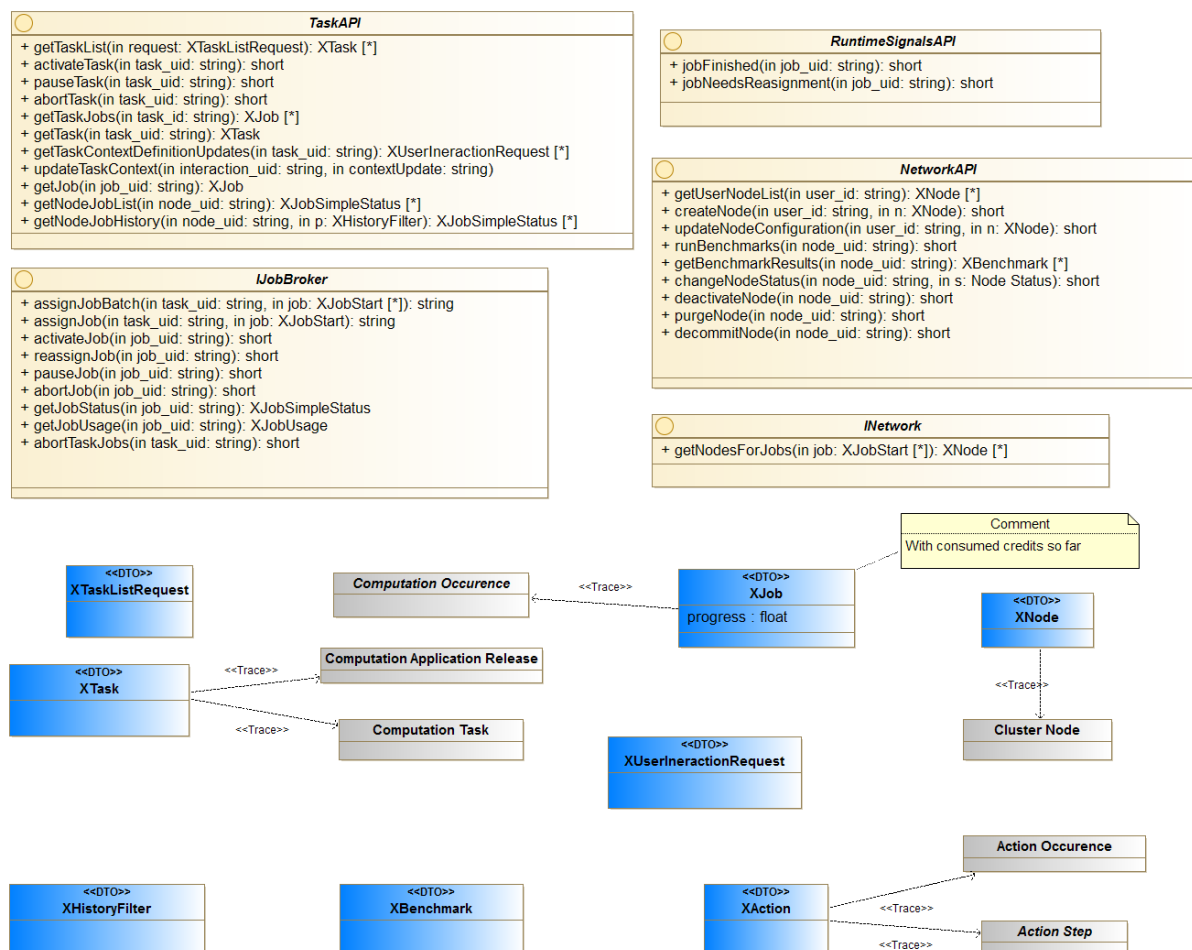


Figure 4. Class Diagram with interface definitions for the Main Node: Computations

2.3 Main Node: Modules

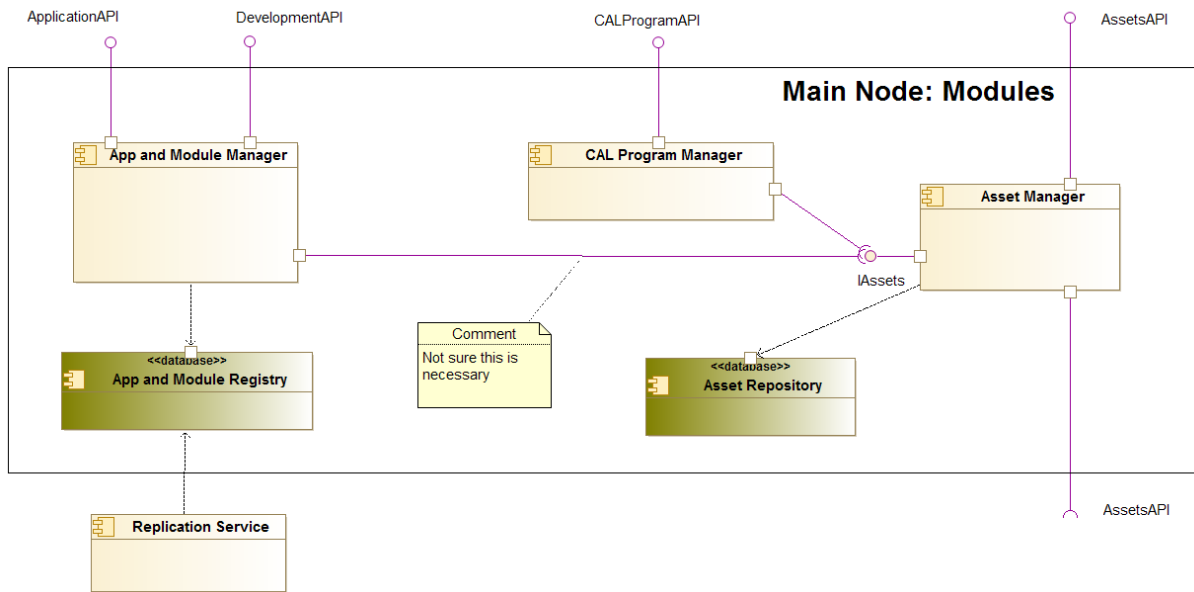


Figure 5. Component Diagram for the Main Node: Modules

CAL Program Manager

Responsible for handling operations on code, related to the currently developed program. The code itself is stored in appropriate containers within the Asset Repository as managed by the Asset Manager.

App and Module Manager

Responsible for giving access to and management of development for applications and computation modules. Stores descriptions of these modules in the App and Module Registry.

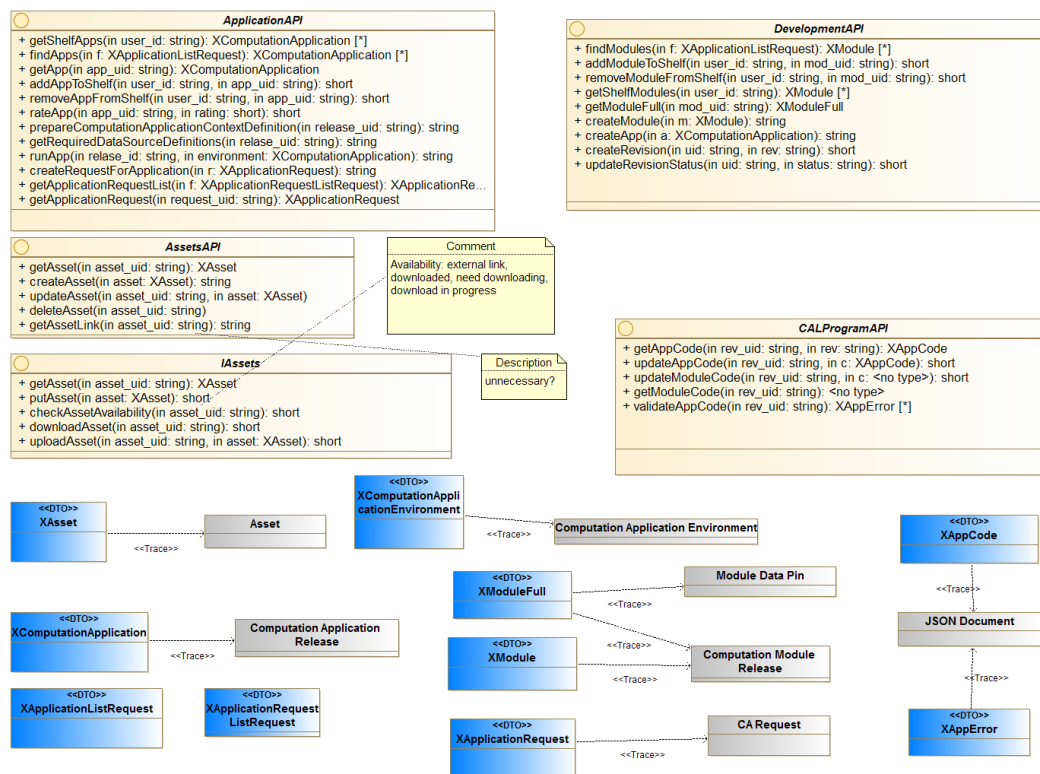


Figure 6. Class Diagram with interface definitions for the Main Node: Modules

2.4 Main Node: Accounts and Billing

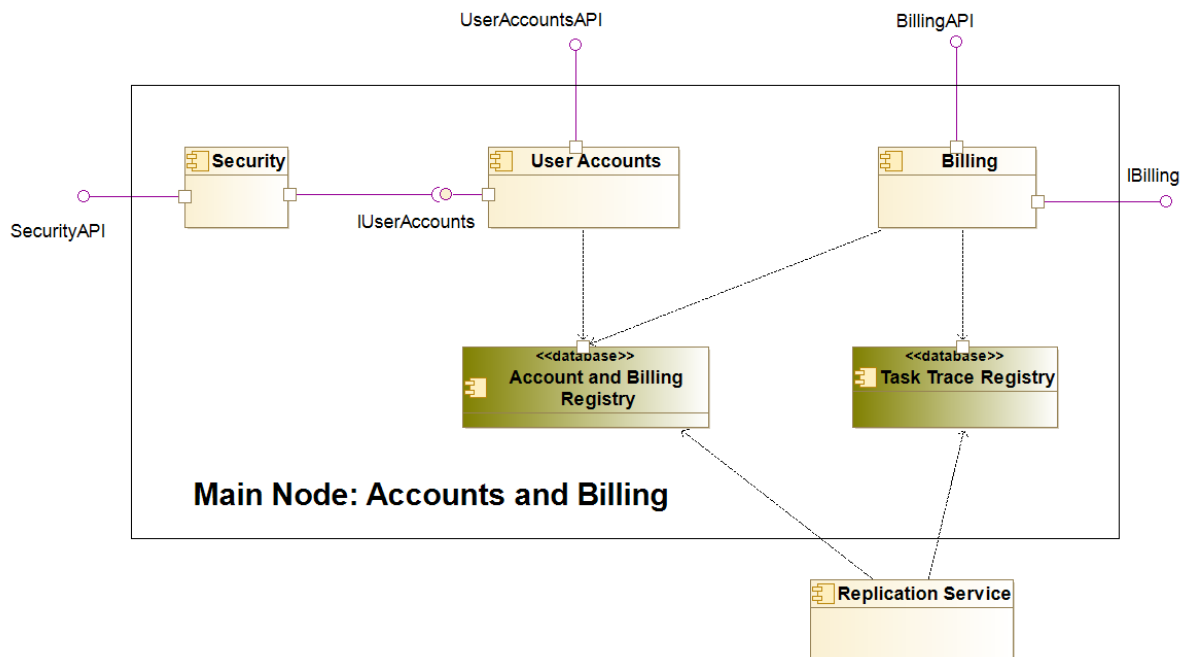


Figure 7. Component Diagram for the Main Node: Accounts and Billing

Billing

Component responsible for computing and providing billing information based on the history of task traces.

User Accounts

Responsible for managing information about the user accounts, including permissions.

Security

Responsible for giving access to the system and assuring security for message passing.

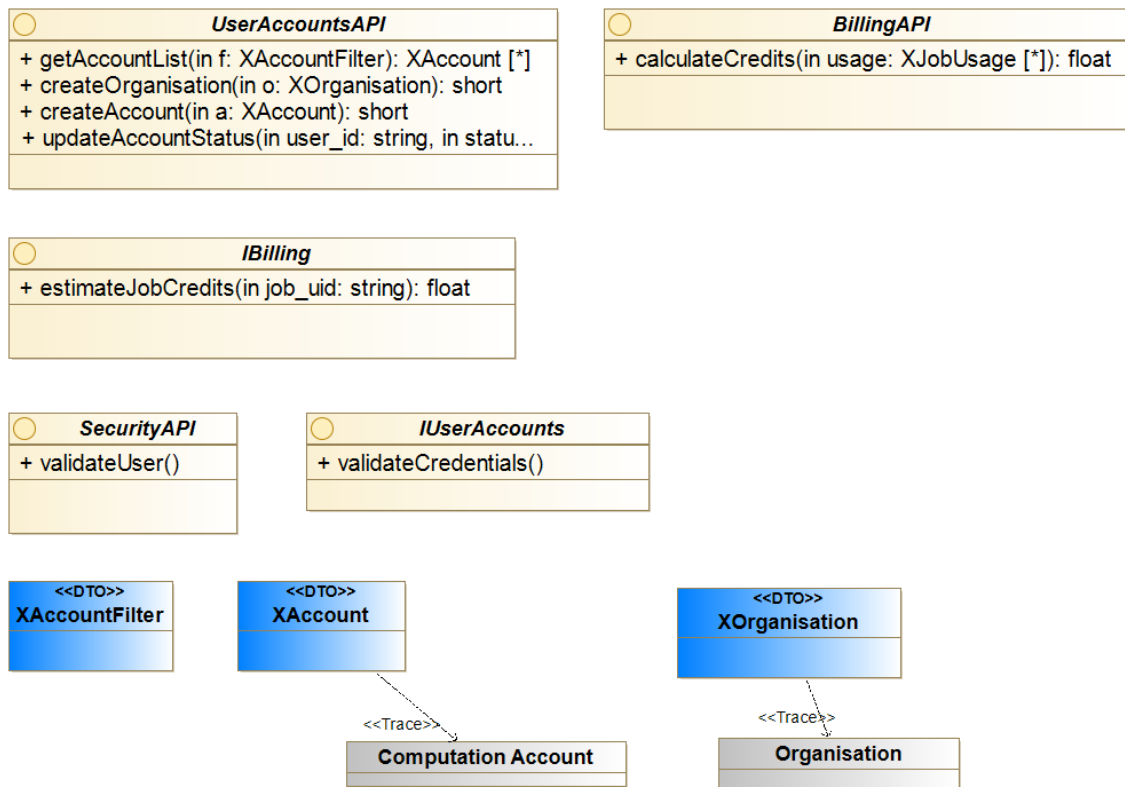


Figure 8. Class Diagram with interface definitions for the Main Node: Accounts and Billing

2.5 FrontEnd: App Usage

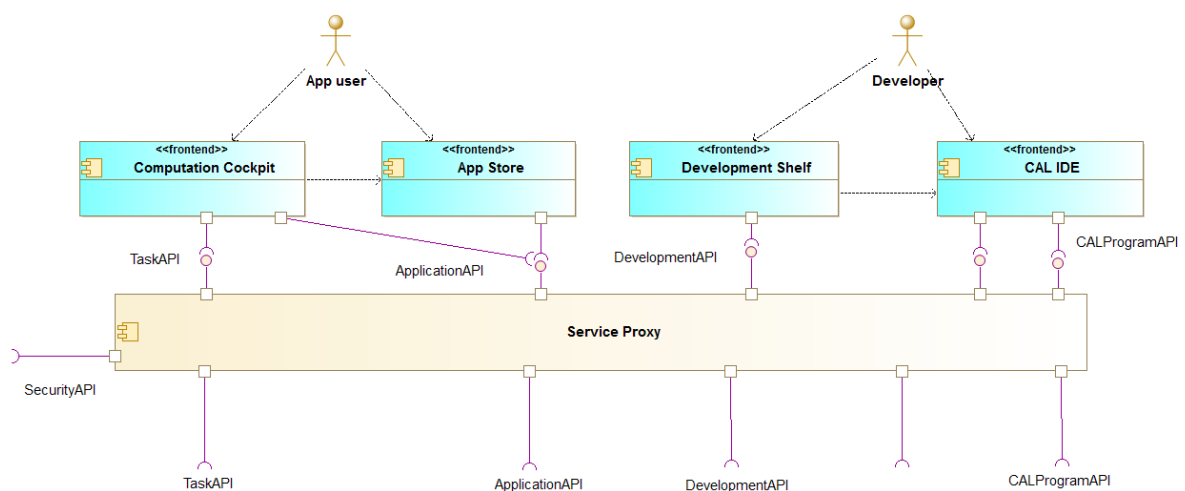


Figure 9. Component Diagram for the FrontEnd: App Usage

App Store

Component responsible for user interactions and application logic for dealing with Computation Applications.

Computation Cockpit

Component responsible for user interactions and application logic for dealing with the Computation Cockpit: managing computation tasks.

Development Shelf

Component responsible for user interactions and application logic for dealing with the Development Shelf: management of the developed Computation Applications and Computation Modules.

CAL IDE

Editor and debugger for CAL programs.

2.6 FrontEnd: Network Administration

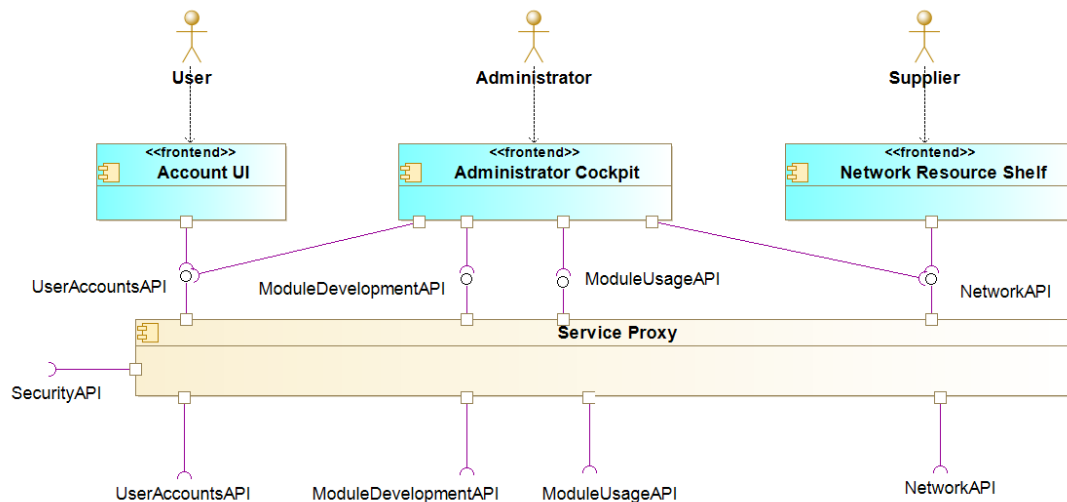


Figure 10. Component Diagram for the FrontEnd: Network Administration

Account UI

Component responsible for user interactions and application logic for dealing with Computation Accounts.

Network Resource Shelf

Component responsible for user interactions and application logic for dealing with the Resource Shelf: management of Cluster Nodes and Master Nodes within the network.

Administrator Cockpit

Responsible for user interactions and application logic for the administrators of the network.

3. Behaviour Model

The behaviour model contains the detailed description of actions taken by a user or a system component in order to complete a use-case. The UML Sequence diagrams are used. They show which components is involved, what the sequence of calls to the API methods is and what the data-flow between components is.

3.1 Show Cockpit and Show Jobs

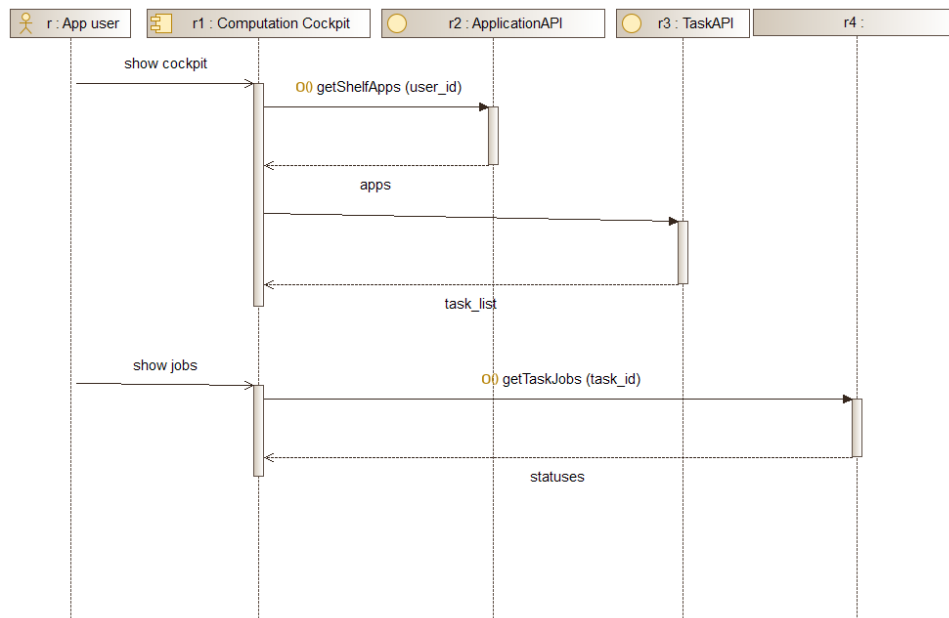


Figure 11. Show cockpit and show jobs

3.2 Create Task

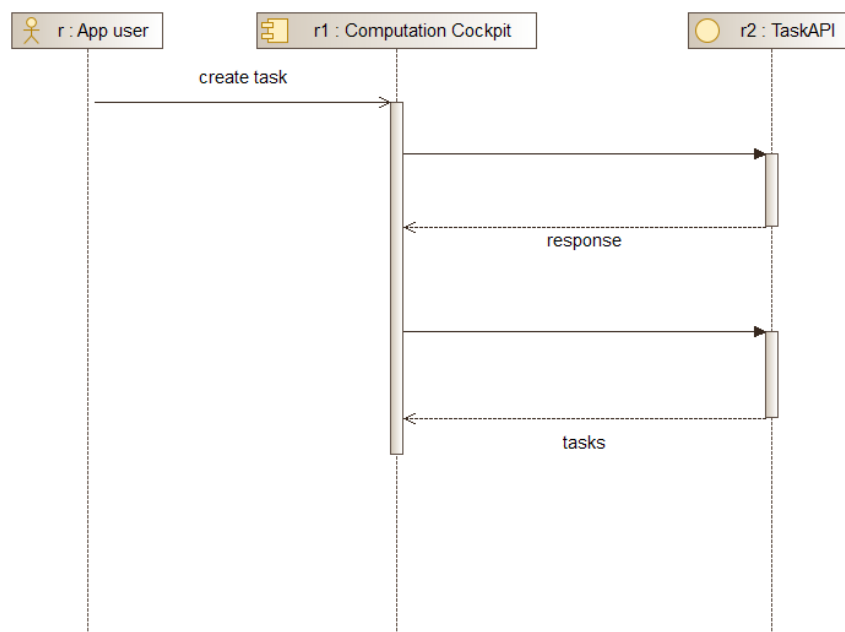


Figure 12. Create task

3.3 Activate Task

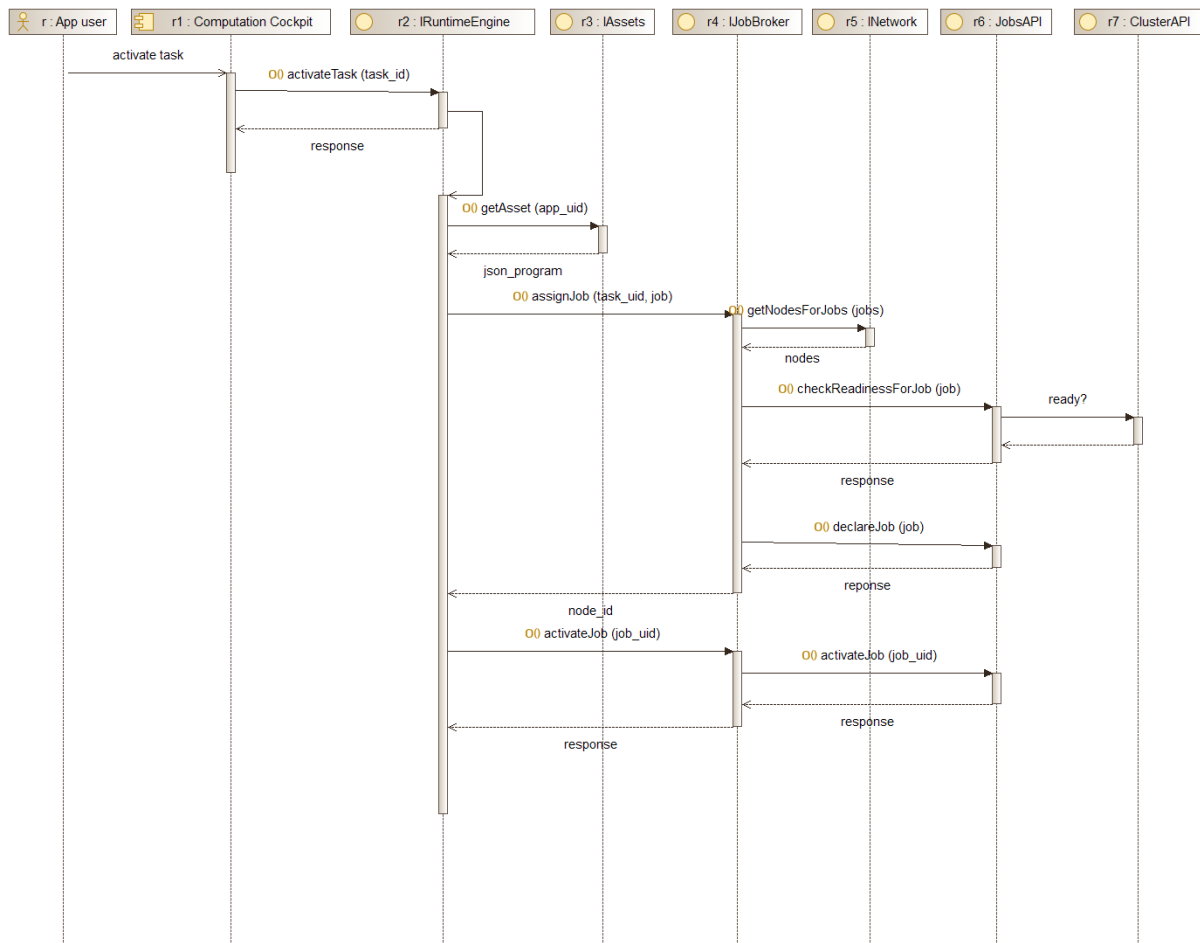


Figure 13. Activate task

3.4 Activate Job

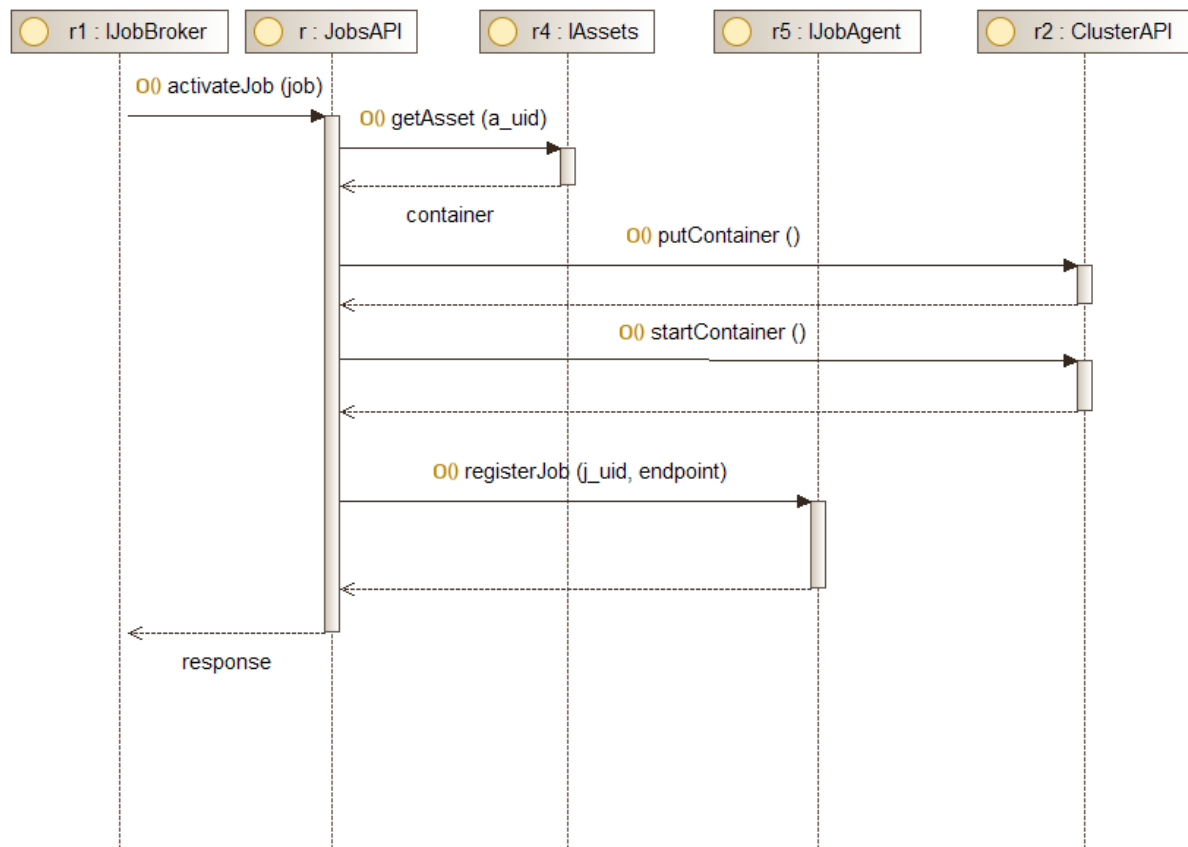
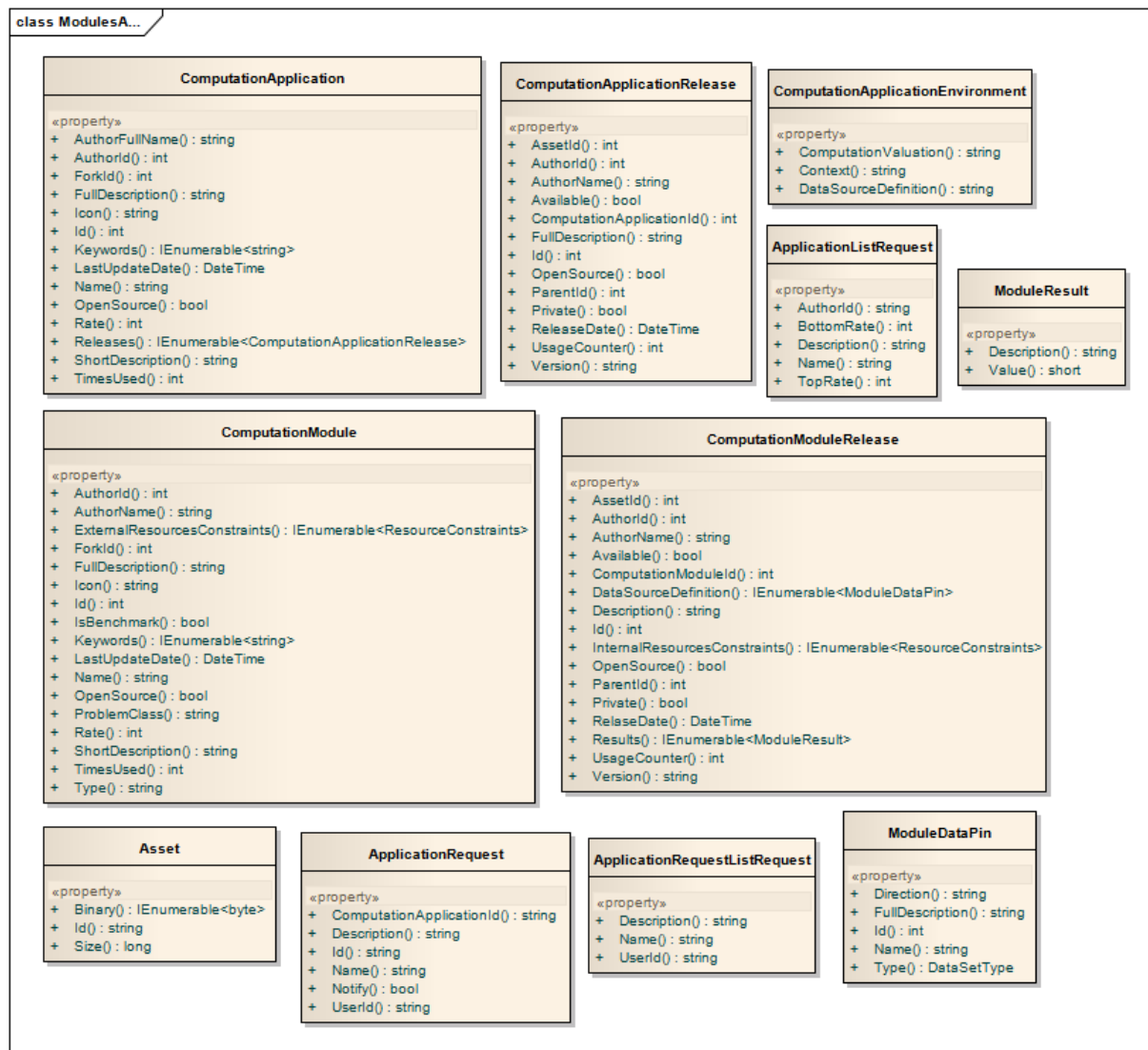


Figure 14. Activate job

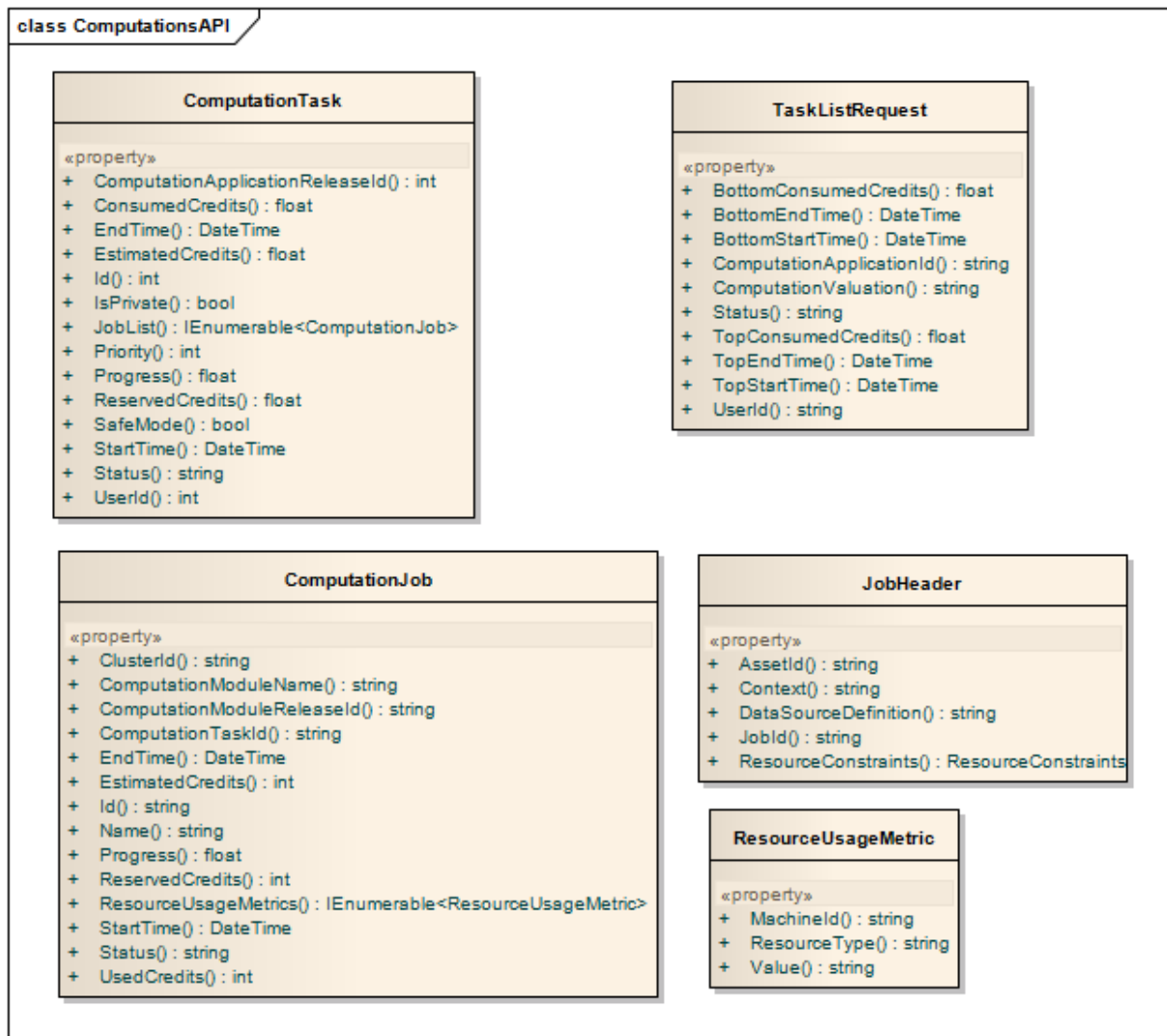
4. Data Model

The data model will contain the detailed description of data flowing through component's interfaces (Data Transfer Objects – DTO's) and data which are persisted in the BalticLSC databases.

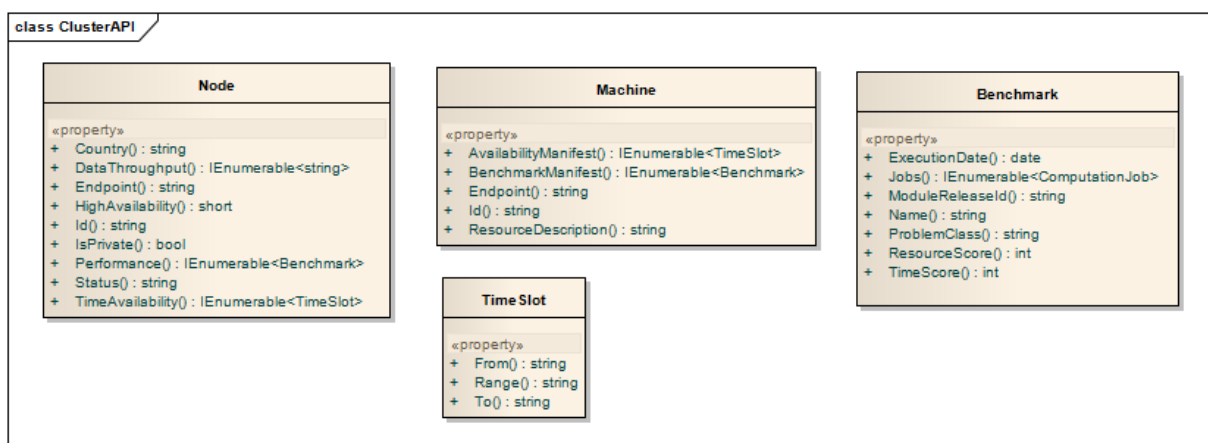
4.1 Modules DTO-s



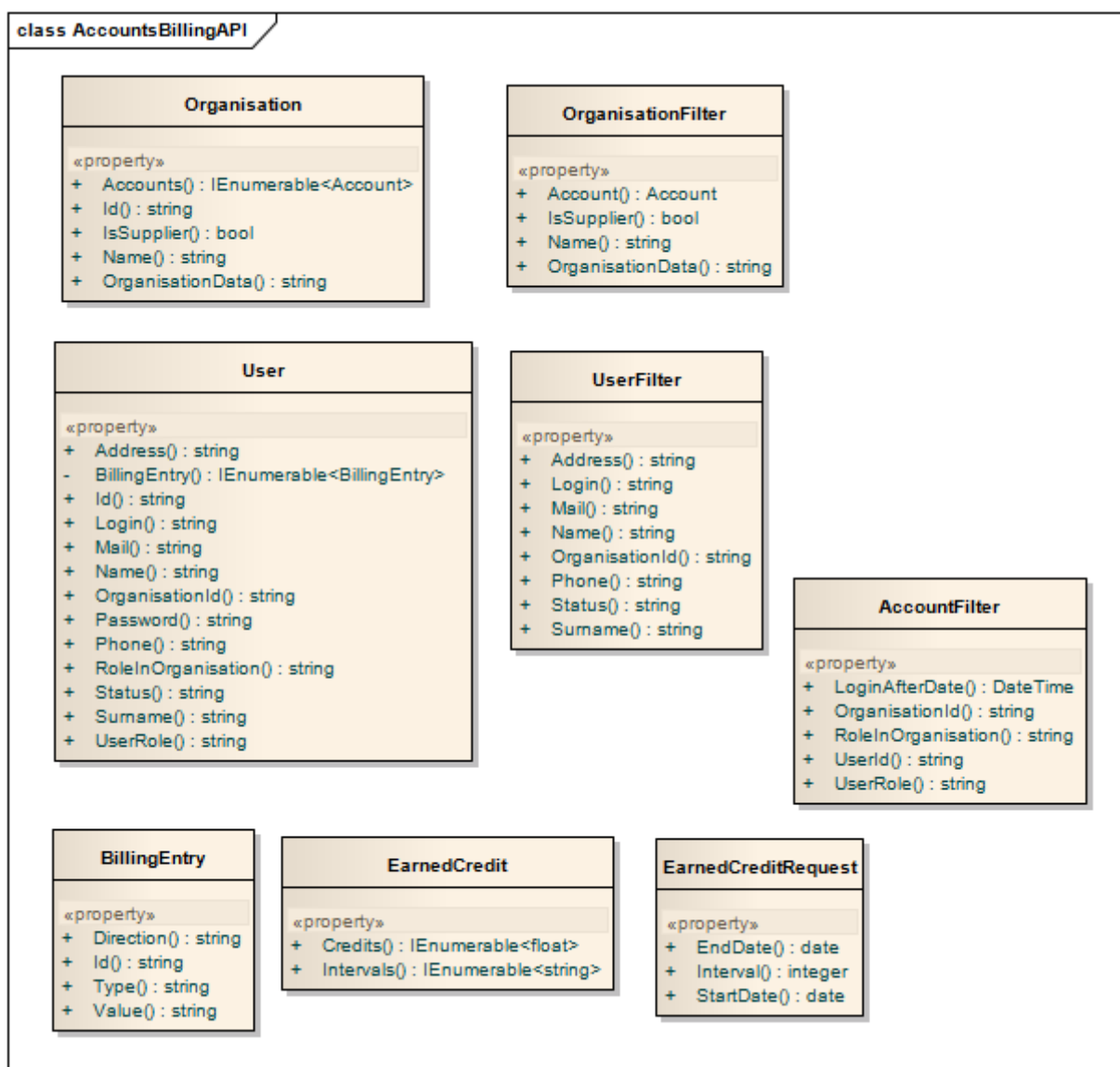
4.2 Computations DTO-s



4.3 Cluster Node API



4.4 Accounts and Billing DTO-s



5. Conclusion

The main objective of the document is to describe the design for BalticLSC Software. The design specification is a “live” document and it is updated during the process of software design and development. The main work is the addition of the necessary details to the behavioural and data models according to the work plan of the agile design and development process.

This document is the main source of the information for the ongoing activities A6.2 Implementation of the BalticLSC Software in order to ensure the match between design and developed software.