Graduation Approval System

SAI Final Assignment

June 2021

Table of Contents

1. Introduction

In this assignment you will integrate a system of several applications for approving start of graduation projects at Fontys ICT school:

- **Client** application is a Java desktop application. It is used for requesting the approval of their graduation project. In this application the student number, company name and project title are filled in and then the *GraduationRequest* is sent to the system. When the system finished processing the *GraduationRequest*, it will send back a *GraduationReply* to the client application, indicating whether this graduation project is approved or rejected by the school.
- **Approval** application is a Java desktop application which is used by graduation coordinators and Exam Board to approve or reject graduation projects.
- **Administration** application is a RESTful API used to get data about students. In this case, it will be used to get the ECs students achieved in the last semester.

2. Integration Flow

The control-flow of the integration starts with the client application creating and sending a GraduationRequest which contains a student number, company name and project title.

Upon receiving a *GraduationRequest*, the system first retrieves *StudentInfo* from the **administration** API for the student with student number specified in the *GraduationRequest*. *StudentInfo* contains the following information about the student: (1) total achieved ECs in the semester 7, and (2) name of the school mentor of this student.

Next, a *ApprovalRequest* is created and forwarded to three **approval** applications. The *ApprovalRequest* contains the same student number, company name and project title from the original *GraduationRequest* and ECs from *StudentInfo*. A *ApprovalRequest* is forwarded to the three **approval** applications according to rules shown in Table 1:

Table 1. Forwarding rules for Approval application

graduation approval application	processes ApprovalRequest
Graduation Coordinator Software (approves the project content)	Students group is 'software' AND ECs > 23
Graduation Coordinator Technology (approves the project content)	Students group is 'technology' AND ECs > 23
Exam Board (approves that the student is ready for graduation)	ECs are in range [24..29] .
None – Invalid Request (do not froward to approval; directly send back the REJECTED GraduationReply)	ECs < 24

Each **approval** application sends back *ApprovalReply* containing the approval decision (approved or rejected) and function of the person who rejected (rejectedBy). If the decision is approved, then field rejectedBy should be left empty.

After replies of all **approval** applications are received, a *GraduationReply* is sent back to the **client** application. This *GraduationReply* contains fields whether the request was approved (if all **approval** applications have approved) or rejected (if at least one **approval** application has rejected). If *GraduationReply* is rejected, then field rejectedBy contains names of all **approval** applications which have rejected it.

Examples of three possible flows are shown in Figure 1, Figure 2 and Figure 3

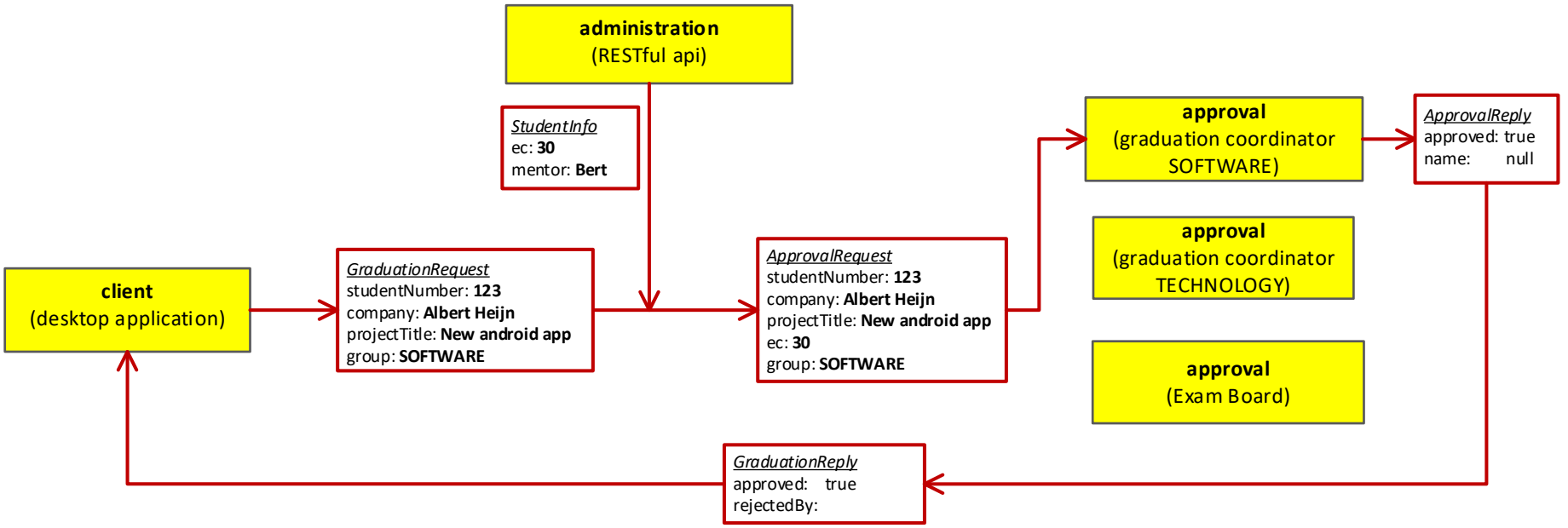


Figure 1. When ECs=30, the request is forwarded only to the group's Graduation Coordinator

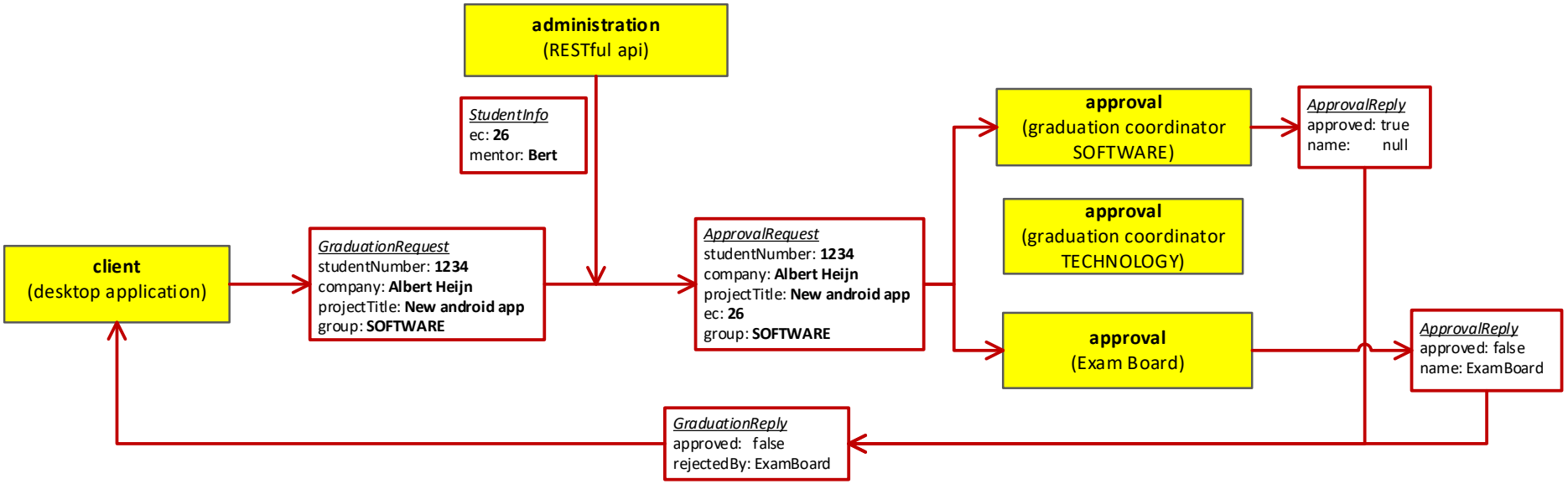


Figure 2. When ECs in range [24..29], the request is forwarded to group's Graduation Coordinator and Exam Board

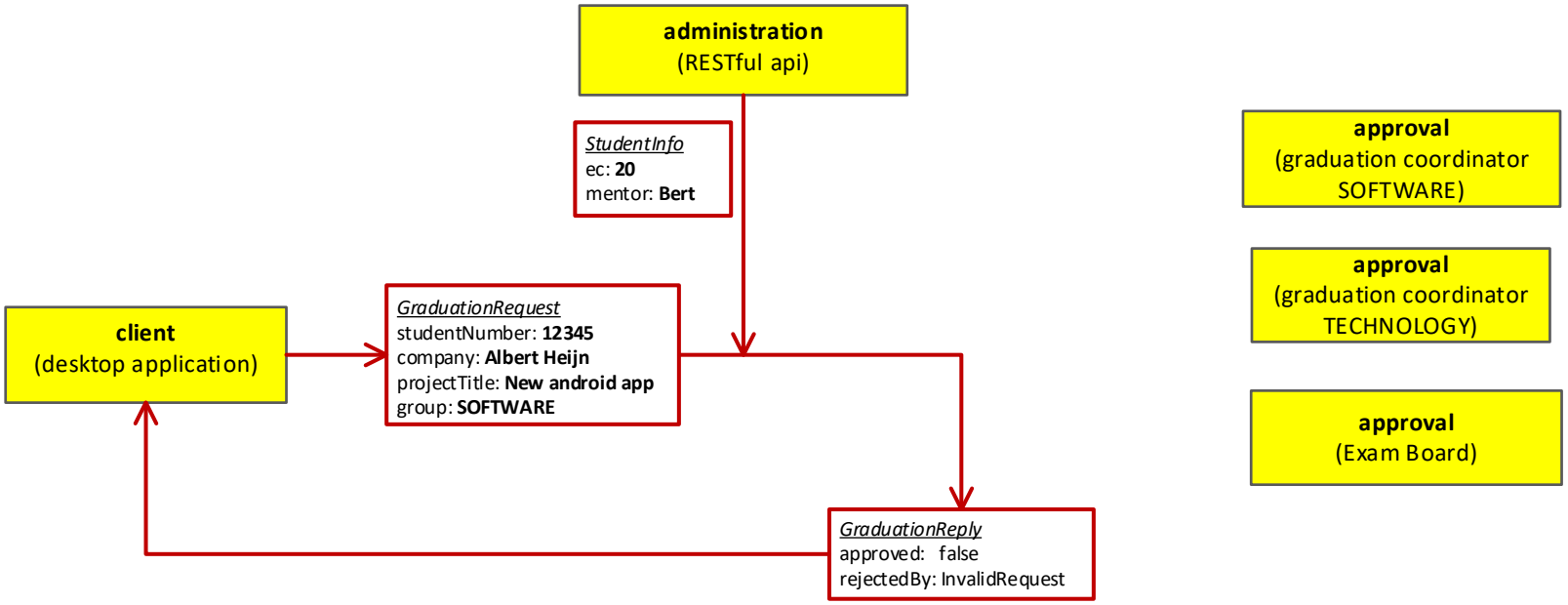


Figure 3. When ECs< 24, the request is automatically rejected

3. Start-up Code

In SAI-final-startup-code.zip you can find (a) the graduation approval start-up project and (b) administration service.

a) Graduation approval start-up project

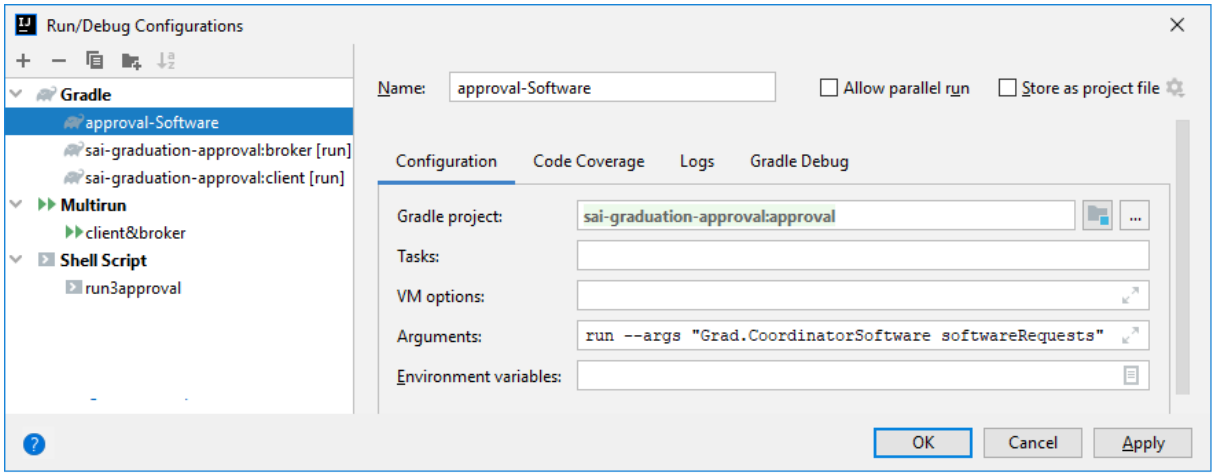
File "graduation-project-approval.zip" contains a GRADLE project with three sub-projects:

- **client** application is a JavaFX application.
- **approval** application is a JavaFX application
- **shared** is a library module where you can find (and also add) some code which will be shared by other applications (see dependencies in build.gradle in **client** and **approval**).

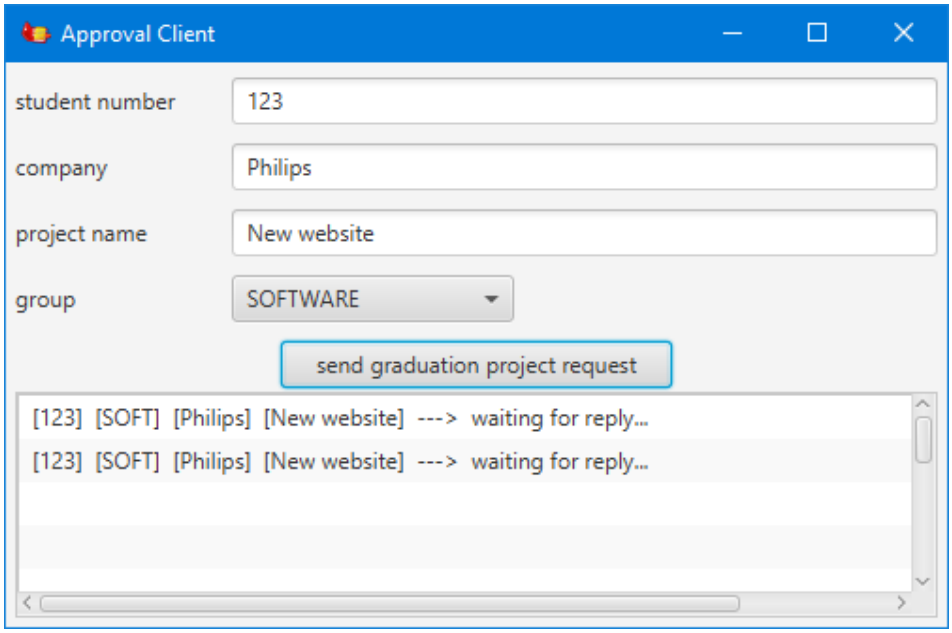
Important note for running the approval application. Arguments for the approval name and approval queue must be provided when running the application. Normally, you can run one or all three approval applications with three custom RUN tasks (see build.gradle file or in IntelliJ/GradleToolWindow/approval/tasks/other/):

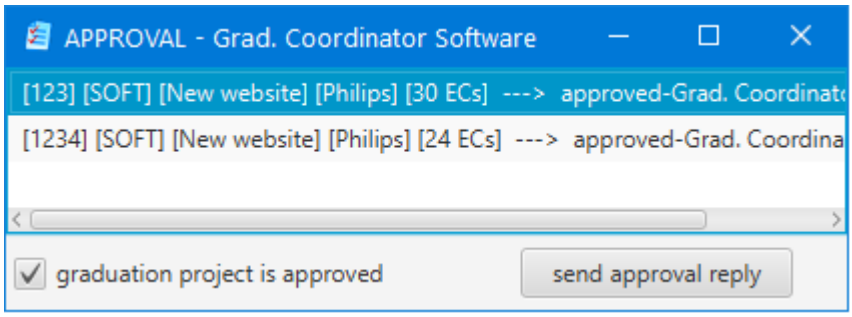
- `gradlew approval:run --args "Grad.CoordinatorSoftware softwareRequests"`
- `gradlew approval:run --args "Grad.CoordinatorTechnology technologyRequests"`
- `gradlew approval:run --args "ExamBoard examBoardRequests"`

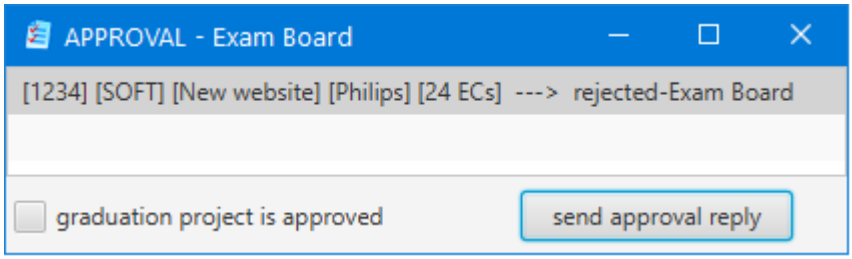
For some reason, newer versions of IntelliJ will not start Gradle run tasks of one module (approval) in parallel. Therefore, you can use file **run3approval.cmd** to run three approval applications in parallel. If you want to run only one approval application, you do need to use this file: you can simply run one of these three Gradle tasks from IntelliJ, but make sure to make a Run Configuration as shown below:

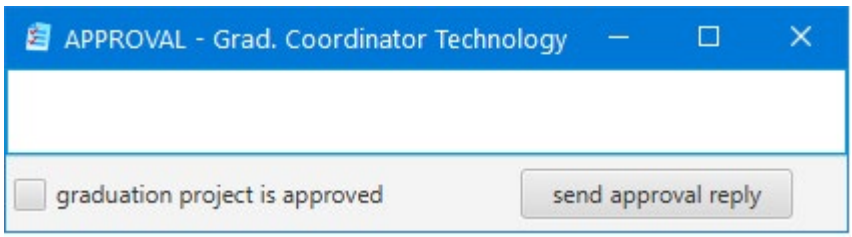


Below you can see screen shots of the client application and three approval applications:









b) Administration service

Administration service is a RESTful API of the student administration office. It is delivered as executable "**administration.jar"**. You can run this service with "**run_administration.cmd"** file. This API generates student info on a random basis as follows:

a. Exactly 30 ECs are generated for student numbers in range 0..999. For example: http://localhost:9091/administration/students/123.
b. ECs in range [24..29] are randomly generated for student numbers in range 1000..9999. For example: http://localhost:9091/administration/students/1234.
c. ECs in range [0..23] are randomly generated for student numbers higher than 10000. For example: http://localhost:9091/administration/students/12345.

4. Assignment

Implement Health Insurance integration system as described in this document. You should make use of the following integration patterns:

- Message Broker
- Correlation Identifier,
- Return Address,
- Messaging Gateway
- Chained Gateways
- Content-Based Router,
- Content Enricher,
- Recipient List,
- Aggregator, and
- Scatter-Gather.

5. Grading Criteria

This assignment is INDIVIDUAL, i.e., it is not allowed to work in groups with other students. The grade you get for this assignment is between 1 and 10, and this will be your grade for the Software Applications Integration (SAI) course.

Gradle project(s) including full source code and all necessary libraries (settings.gradle, build.gradle, etc.) must be submitted. All submitted projects must compile and run correctly on the computer of the teacher. If the teacher does not have your full source code or cannot run your project(s) due to compiling errors, missing files, or exceptions, then your SAI grade will be 1. Otherwise, SAI grades will be determined based on implemented Application Integration Patterns in the following way:

Description	Implemented features	SAI grade
The system works correctly with **one approval application** (e.g., Graduation Coordinator Software).	Message Broker	6
	Correlation Identifier	
	Return Address	
	Messaging Gateway	
	Chained Gateways	
The system works correctly with **one approval application** and **administration service**.	Content-Based Router	7
	Content Enricher	
The system works correctly with **three approval applications** and **administration service**.	Recipient List	8
	Aggregator	
	Scatter-Gather	
	Use of mXparser (or similar) instead of hard-coded approval routing rules.	9
Well organized code, with comments, proper variable and method names, no redundant code.		10

6. Submitting the assignment

Submit the assignment before the deadline on Canvas. It is not possible to submit after the deadline. To submit your source-code do the following:

1. *In IntelliJ:* Execute Gradle task **build/clean** in the root Gradle project **sot-final-assignments**. This will delete temporary gradle directories/files (e.g., build directories) in all sub-projects.*In the File Explorer*: In the root directory **sot-final-assignments** delete manually all temporary IntelliJ and Gradle directories, the only left directories and files should be the *source-code*, *build.gradle* files and *settings.gradle* (similar like in the starup-

project you downloaded):

2. *In the File Explorer:* Zip root directory **sot-marktplaats** and submit it.

7. Submission and Deadlines

Submission of the source code

The IntelliJ project(s) with full source code and all necessary libraries (via gradle, maven or *.jar) must be submitted via Canvas. The deadline for submission is set in this Canvas assignment. It is not possible to submit after this deadline. If you do not submit your source code before the deadline, you will not receive a SAI grade in this block (i.e., you will not pass the SAI course in this block).

Defense of your assignment

In week 8 or 9 SAI exam is scheduled (see class schedules). During this exam you will speak in person to the teacher about your assignment: you will be asked to explain your code, suggest ideas for improvement, etc. If you are not present during this exam, then you will not get a grade for SAI. It is not possible to mote your exam at another time.

Only students who submitted their source code via Canvas before the deadline specified in Canvas will be invited for this exam. You will receive this invitation with your specific time slot from you teacher several days before the exam. In this invitation it will be specified at which time you should be present for this exam. Each student will have his/her own time slot, and you should and can be present only during your own time slot.