Software Architecture Description, Deliverables Week 2 / Sprint 1

Diagram of Runtime Components:

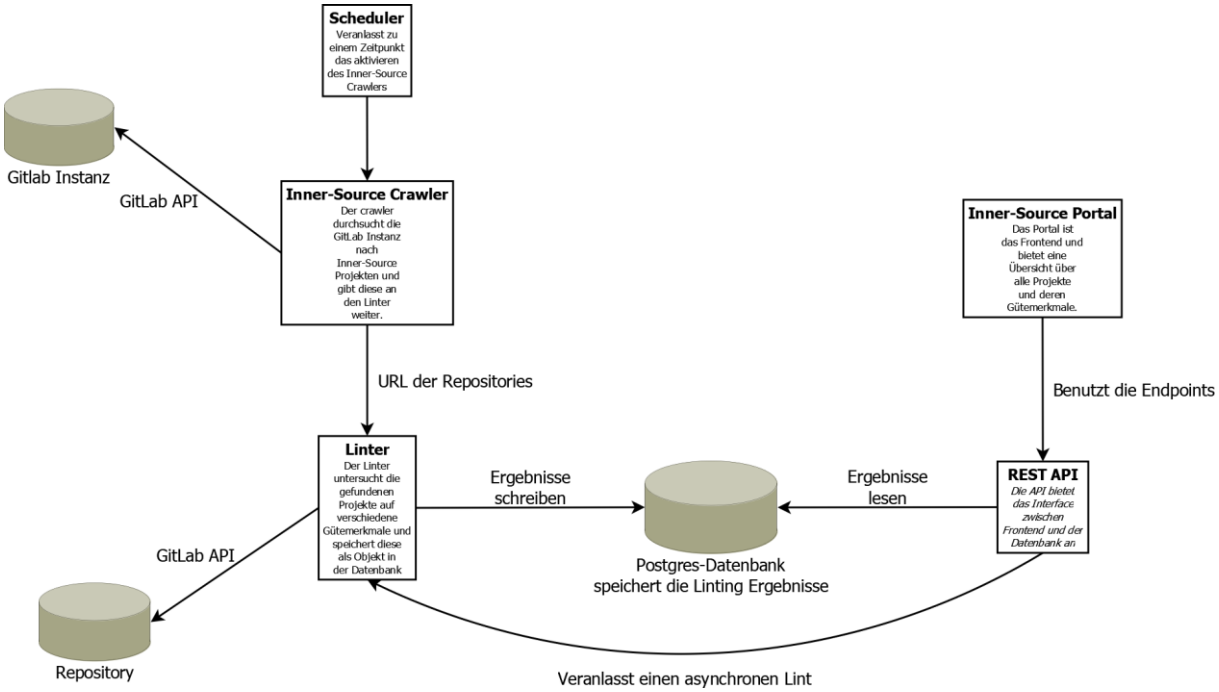

Abbildung 1 Diagram of Runtime Components

Diagrams of Code Components

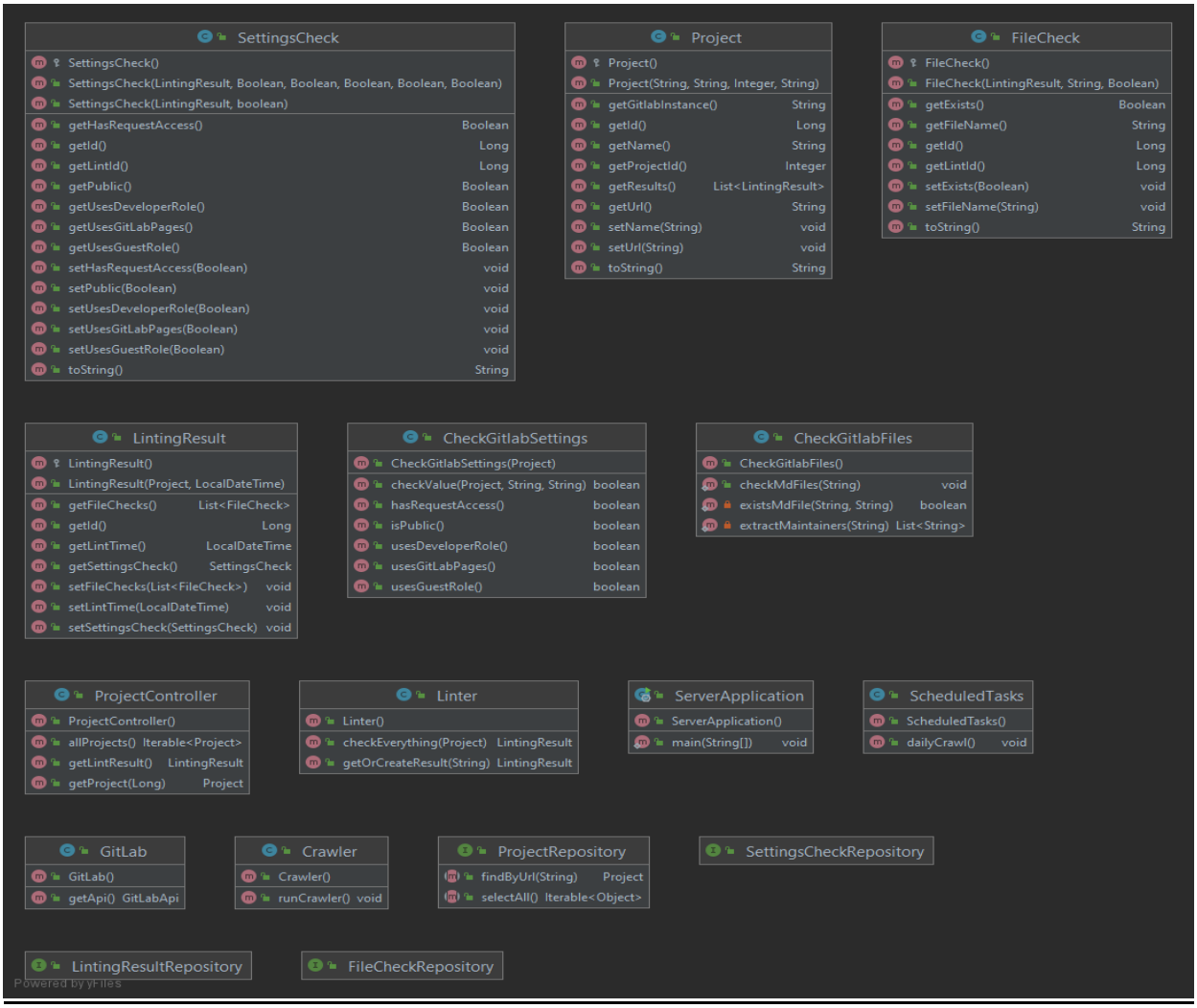

Figure 2 UML digram

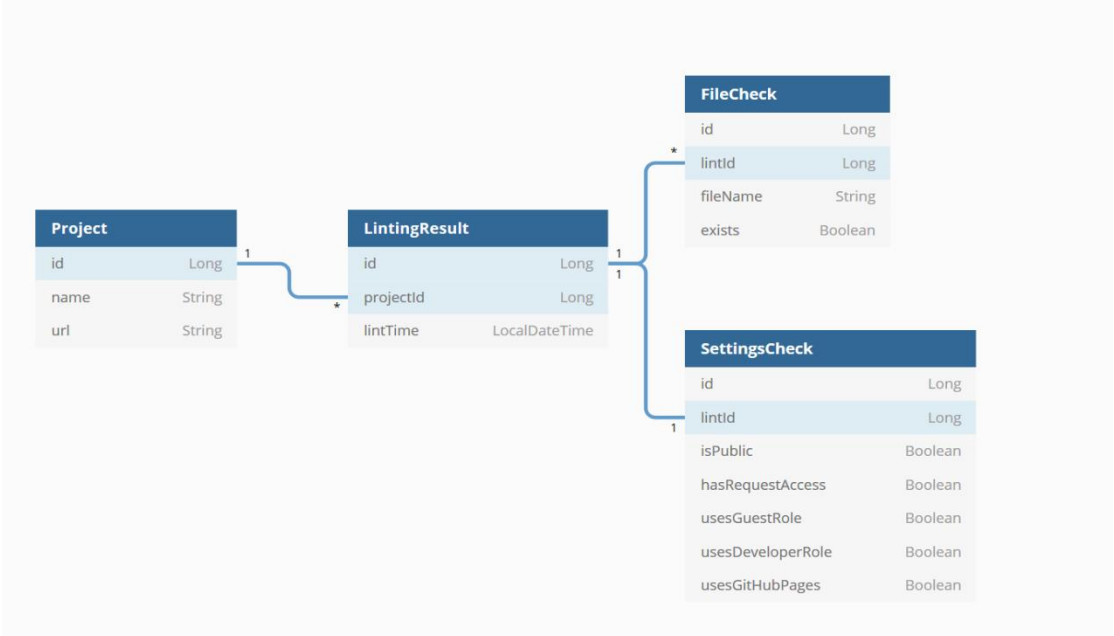

Figure 3 Database ER diagram

Summary of underlying Technology Stack:

\	Context	Name	Version	License	Comment
1	Frontend Framework	Angular	11.2.11	MIT	
2	Backend Framework	Spring (Boot)	2.4.5	Apache License 2.0	
3	Backend Database	Postgres SQL	12	PostgreSQL Licence	Similar to MIT
4	Backend runtime	Docker	20.10.6	Apache License 2.0	
5	Backend orchestrator	Docker-Compose	1.29.1	Apache License 2.0	
6	Backend Framework	GitLab4J-API	4.8.0	MIT	
7	Backend testing	JUnit	5	Eclipse Public License	
8	Backend scheduler	awaitility	3.1.2	Apache License 2.0	
9	Testing Database	H2 in-memory datbase	1.0.200 61217	Eclipse Public License or Mozilla Public License 2.0	
10	Http Testing	HttpClient	4.5.13	Apache License	
11	Java runtime	OpenJDK	11.0.11	GNU GPL v2	with linking exception

Figure 4 Technology Stack

Description:

Angular is used to create the frontend for application.

Spring (Boot) is used to create a stand-alone web application using the embeded Tomcat server.

PostgreSQL is used as relational database managment system with SQL compliance and transactions with ACID properties.

Docker and Docker-Compose are used for porting the application to different platforms.

The GitLab4J-Api supports the usage oft he GitLab-Api inside Java.

Awaititility is used as a schedular to run a job every day.

JUnit, H2 and HttpClient are used to test the backend, the database and the frontend.

Java is used as the programming language and runtime environment.

Descriptions:

Figure 1: Diagram of Runtime Components

The inner-source portal, displays a website built via angular, which takes an URL as an input and sends it to the REST-API. Either there is already an entry in the database then the results are returned or the URL gets sent to the Linter where it is checked if a repository exists or not. If so then the Linter extracts required information on quality criterions and saves it the database. The database consists of a list of inner-source repositories, where each entry has a number of linting results with their corresponding timestamps. This creates the possibilty to display the development of a repository over time.
There is also a total check of all inner-source repositories under a certain gitlab instance. For this task the scheduler runs each day at a certain timestamp and starts the crawler, which extracts a list of all inner-source repositories and hands them over to the linter. The linter then creates a linting result which is saved to the database with a corresponding timestamp.

Figure 2: UML Diagram

The Server Application starts the SpringBoot App.

The Project class contains all information about a repository, which is stored in the database. The information itself is saved via linting results (see Abbildung 4).

The Linting Result is used as container for all needed information e.g. Time stamps, file checks and setting checks (see Abbildung 4).

The File Check class stores currently only the existence of certain required files. A linting result can have multiple File Check objects (see Abbildung 4).

The Settings Check class stores the current settings, like visibility, allowance of forking, etc.., of a repository at a timestamp specified in the Linting Result (see Abbildung 4).

The Linter class is used for extracting all required information via the gitlab api for a given repository. The method checkEverything() then creates a lintingResult object where all information is saved. The Linting Result object is then returned.

The linting Result class the references on all information of the executed checks.

The classes CheckGitlabSettings and CheckGitlabFiles are being called in checkEverything() and are used to perform needed checks.

The GitLab class is the entry point of the gitlab4J library.

The ProjectController class communicates betweend api, backend and frontend.

The ScheduledTasks class is used to perform an update on the database for all inner-source projects as well as the addition of new ones. The crawler class is then activated to get all inner-source repositories, to lint all.