



ToDo LIST WEB APPLICATION

Design Document

Contents

1. JavaScript framework selection.....	2
1.1 Angular Advantages over React and Vue	2
1.2 Downsides or limitations of React framework	2
1.3 Downsides or limitations of Vue frameworks.....	3
2. Backend technologies	3
2.1 Why Spring Boot?.....	3
3. Database Technologies	4
3.1 Why MySQL database?	4
4. Software architecture	4
4.1 Level 1: Context diagram	4
4.2 Level 2: Container Diagram.....	5
4.3 Level 3: Component diagram	7
4.3.1 <i>Front-end Angular Application</i>	7
5. ERD	10
6. Ci/CD Integration test report Screenshot	11
7. CI/CD integration Diagram	12
8. SonarQube test report screenshot	13
I. Reference	15

1. JavaScript framework selection

Now a day a lot of frameworks are on the market and these all framework has their own pros and cons. Choosing the right framework is sometimes confusing and it needs deep research. To have the right framework is very important and that will cut unnecessary costs and speed up the development process.

First, I researched and analyzed different web technologies in different perspectives and in terms of building a simple ToDo List web application. Finally, I decided to use Angular for my ToDo list front end application and I listed out the reasons below why I should be using Angular: -

1.1 Angular Advantages over React and Vue

- In 2018 Stack Overflow survey, it is ranked the second most used technologies.
- Boasted in the detailed documentation.
- Supported by Google
- Filters – contains multiple filters to format data of different data types.
- Intuitive and declarative interface
- Component-based architecture
- CLI – automates the whole development process
- Dependency injection – assists the developer in creating components, resolving their dependencies and providing them to the other components as required
- Popularity – apps built with Angular – YoutubeTV, Google Cloud, Netflix and Udacity,
- Directives – allow the developer to build custom HTML tags

1.2 Downsides or limitations of React framework

- Angular offer three 3 files for component HTML, CSS and JS, while React offers only two files for components.
- Angular offers the whole thing from routing to the template, unlike Angular React does not

Provide everything in the official library.

- React doesn't have a form validation and handling like Angular. In Angular, I can easily define a customer validators and use built Angular validators, for example, min/max, email, pattern etc.
- JSX React's documentation is disliked by many developers.

1.3 Downsides or limitations of Vue frameworks

- Vue has only one file for the component, in that reason I prefer to use Angular to work on three components.
- Limited resources and Fewer tutorials, according to me, in order to do one project in a fast and concurrent way, first we need to check if this technology has enough resources Or not. According to many developers and my finding, Vue doesn't have enough resources.
- Lack of experienced developers - Always better to have experienced people in some topics or technology area to get advice and feedback. In the case of Vue, don't get a lot of experienced developers.
- Language barriers, I found most of the codes and examples in Chinese language.

In conclusion, there is no better framework here, all framework has its pros and cons. Three of them are very fast frameworks. But I have to choose only one framework for my project. In that reason, I choice Angular because of the above reasons. And, Angular framework is preferable to access information and data security than Vue and React.

2. Backend technologies

2.1 Why Spring Boot?

- It provides a powerful back to manage REST endpoints.
- In Spring boot, everything is auto configured; no manual configurations are needed.
- It reduces overall development time and increase efficiency by having a default configuration for unit test.
- It provides a very good support to create a DataSource for Database.
 - ✓ Just adding the dependencies and doing the configuration details is enough to create a DataSource and connect the Database.

3. Database Technologies

3.1 Why MySQL database?

- It is a free and open source relational database management system.
- It is highly extensible, reliable, compatible with all major hosting providers and easy to manage.
- Connectivity and security, it is fully networked, and I can share my data with anyone, anywhere. And, it supports encrypted connections using the secure Sockets Layer protocol.
- Open distribution and source code, it is easily obtainable; simply use my browser if I don't understand how something works.
- In addition, fonts provide free MySQL access for its students.

4. Software architecture

4.1 Level 1: Context diagram

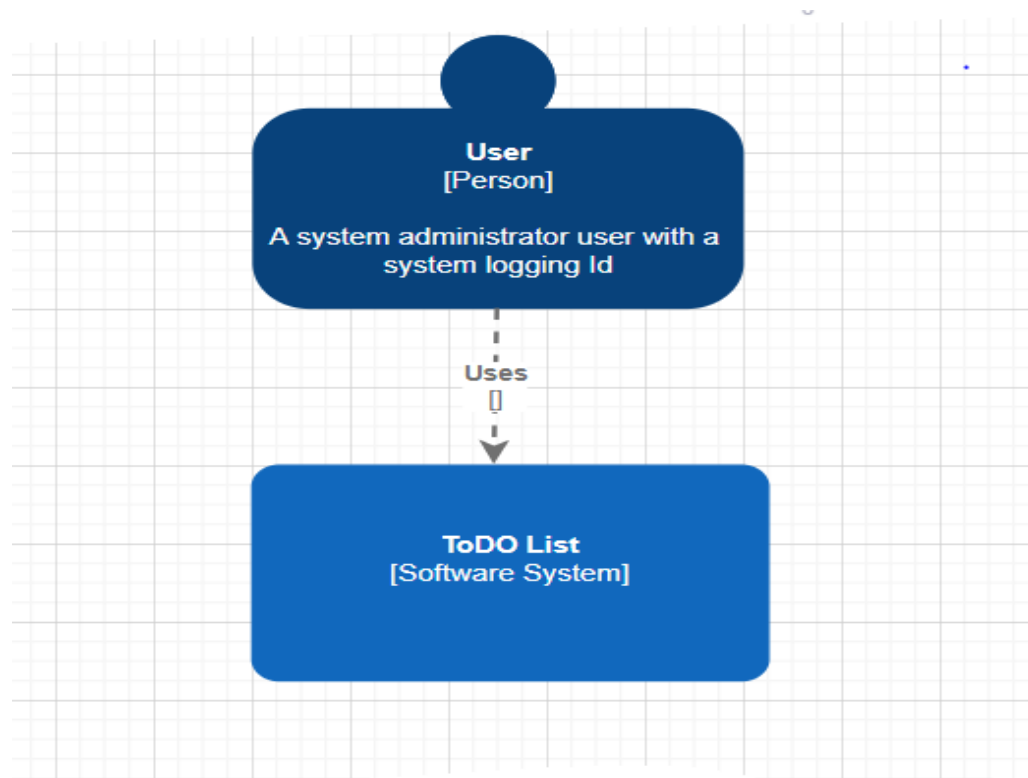


Diagram 1 - Context Diagram

The above diagram is according to the user view and, it shows how the system fits into the world.

The admin/manager of the company uses ToDo List system to do CRUD functionalities, for assigning roles and tasks to employees, and categorize tasks within departments.

4.2 Level 2: Container Diagram

The diagram below shows a container which zooms into the software system and shows the container which creates the software system. Moreover, in this diagram, I have attempted to demonstrate the technological decisions that will be used to create the system.

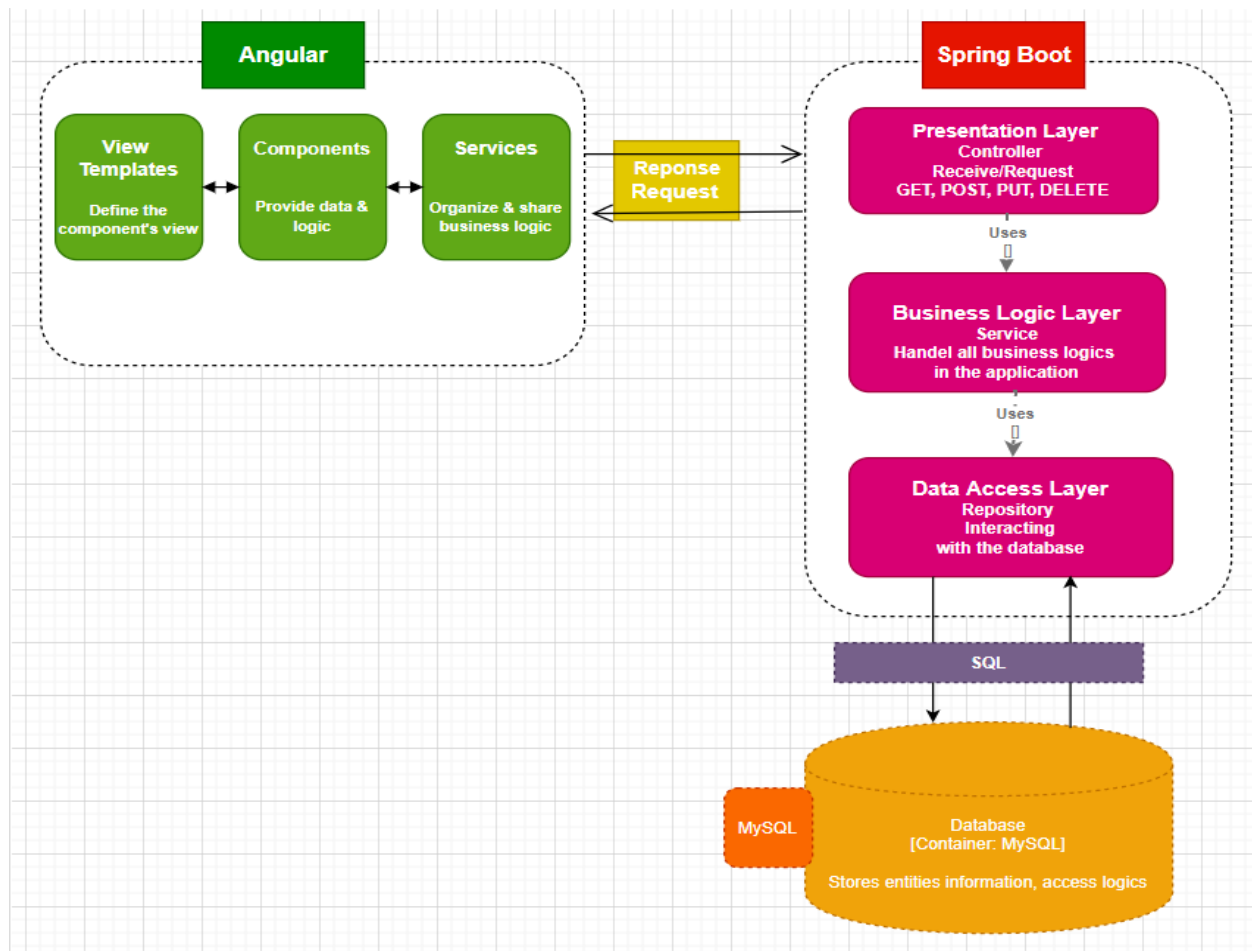


Diagram 2 – Container diagram

The backend application has three different layers: **A presentation layer, a Business logic layer, and a Data access layer/Persistence layer.**

The **Presentation layer** has controller classes and, these controller classes receive requests from the client and maps that request and handle it. It means that the client requests are handled in these controller classes. After that, it calls the service logic if required. These classes are put in a controller package in my project.

The Business logic layer has service classes and, in these service classes, all the business logic (data process, data transformations, and cross-record validations) performs. After performing the business logic uses the data access layer to perform the logic on the data that is mapped to JPA with model classes. And, these classes are put in the Service package in my project.

The Data Access layer/ Persistence layer contains all the storage logic (database queries) and translates business objects from and to database rows. It is mainly responsible to interact with the database. This layer has repository classes and, this layer is responsible for CRUD operations on a data source, which is a relational database. It is implemented using Spring Data JPA. Repository classes are put in a repository package in my project.

Technologies:

The frontend application is an Angular JavaScript framework application that works in a web browser and, it is delivering and presents all features of the ToDo List backend app. The backend application is a Java, Spring Boot application. And, to organize and persist data, the system uses MySQL database technology.

Remarks: Front side tools and technologies used: Angular 9, TypeScript, NodeJS and NPM, IDEA, Angular CLI, and bootstrap 4+.

1. Why NodeJS in Angular, it allows me to spin up a lightweight webserver to host the application locally on my system.
2. Why NPM, it gives me angular CLI (angular command-line interface)

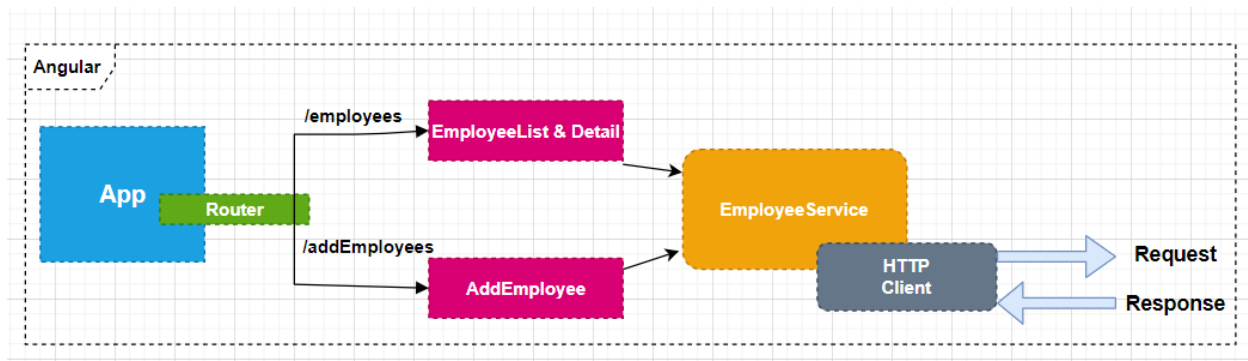
Remarks: The backend technologies used: Spring Boot 2.3.4 +, Spring Data JPA (Hibernate ORM core version 5.4.21), Maven 3.2 + JDK 1.11, Embedded Tomcat 9.0.38 +,

Remarks: Database technologies used: MySQL

4.3 Level 3: Component diagram

4.3.1 Front-end Angular Application

The below diagram shows only employee components within the Angular 9 front-end application. Other components are also following the same process as the Employee component.



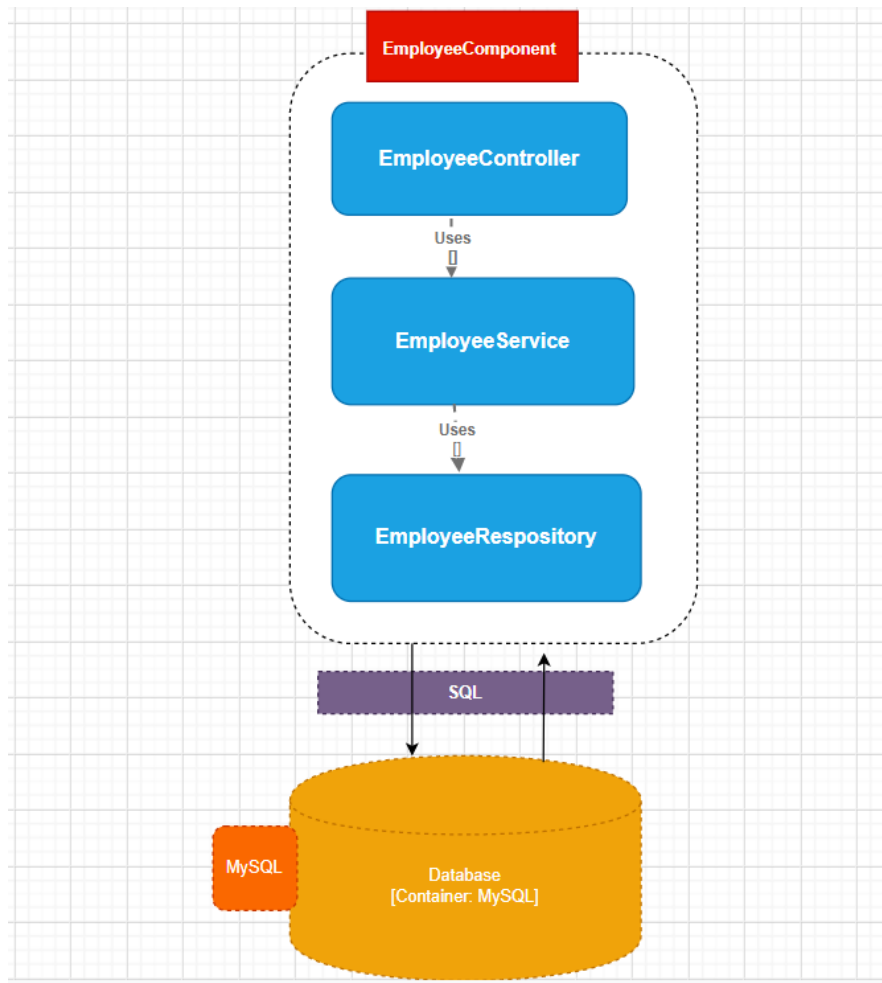
The App component is a container with a router-outlet. It has a navigation bar that links routes through the router link.

- EmployeeList component gets and displays the employee.
- EmployeeDetail component is for editing Employee's detail based on :id.
- AddEmployee component for adding/submission a new Employee.

These Components call EmployeeService methods which use Angular HttpClient to make HTTP requests and receive responses. EmployeeService layer is responsible to handle all business logics.

4.3.2 Backend Application

In below diagram, I will try to demonstrate employee component in different layers. All other components are also having the same diagram as employee component.



In the Employee component presentation layer, there is an **EmployeeController** class that provides APIs for creating, retrieving, updating and, deleting Employees.

In the business logic layer, there is an **EmployeeService** class and, in this service class, all the business logic (data processing, data transformations, and cross-record validations) performs.

Data Access layer/ Persistence layer at the Employee component has an **EmployeeRepository** interface to interact with Employees from the database. This **EmployeeRepository** interface extends **JpaRepository** to use **JpaRepository** methods like (**save()**, **findById()**, **findAll()**, **delete()**, **deleteById()**). **EmployeeRepository** interface is responsible for CRUD operations on a data source, which is a relational database. It is implemented using Spring Data JPA.

Back-end overview

Spring Boot exports REST APIs using Spring Web MVC & interacts with MySQL Database using Spring Data JPA. Below I would like to show only Employee APIs that Spring App will export. And, all other entities use the same overview as Employee

Methods	URLs	Actions
POST	/employees	create new Employee
GET	/employees	retrieve all Employees
GET	/employees/:id	retrieve an Employee by :id
PUT	/employees/:id	update an Employee by: id
DELETE	/employees/:id	delete an Employee by :id

5. ERD

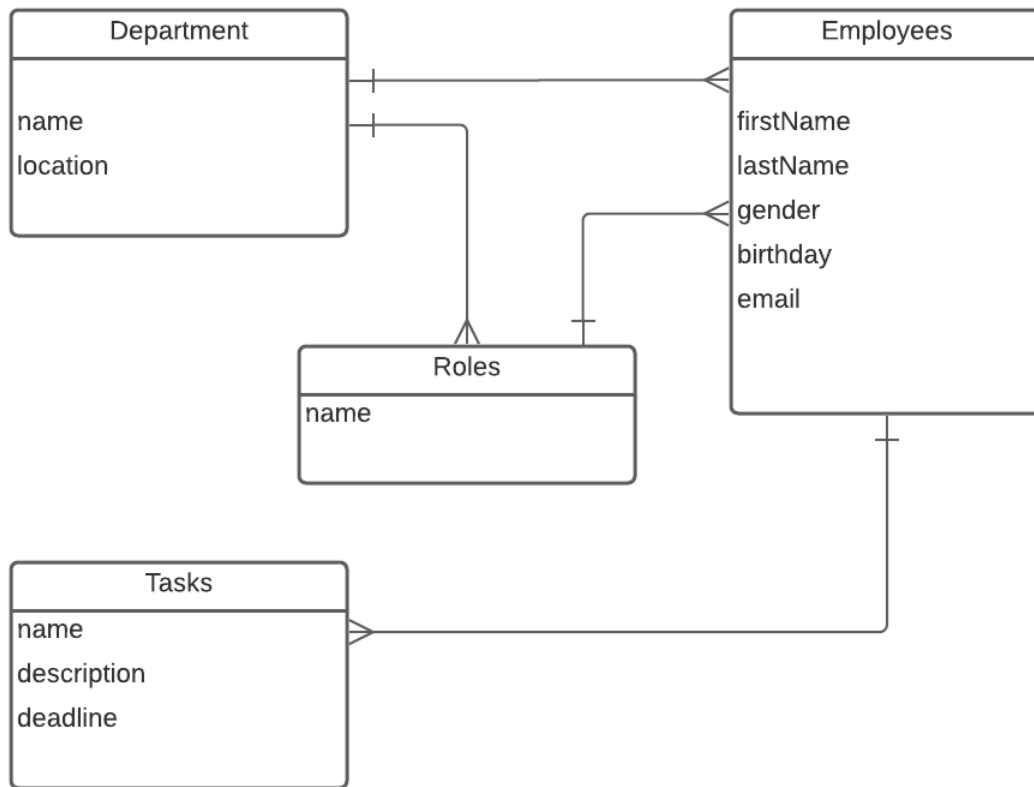


Diagram 5 - ERD

6. Ci/CD Integration test report Screenshot

The image displays three screenshots of a CI/CD integration test report, likely from a tool like GitLab CI/CD. Each screenshot shows a sidebar with navigation options (Project overview, Repository, Issues, Merge Requests, CI / CD, Pipelines, Jobs, Schedules, Operations, Packages & Registries, Analytics) and a main content area with a terminal output and a job status summary.

Top Screenshot: The terminal output shows the build process starting with "Running with gitlab-runner 13.4.0 (4e1f20da)" and "Preparing the 'shell' executor". It lists various steps like "Preparing environment", "Getting source from Git repository", "Fetching changes with git depth set to 50...", "Checking out 9d2421c3 as master...", "Removing target/classes/wondl/", "Removing target/generated-sources/", "Removing target/generated-test-sources/", "Removing target/maven-status/", "Removing target/surefire-reports/", "Removing target/test-classes/", "git-lfs/2.8.0 (GitHub; windows amd64; go 1.12.2; git 30af66bb)", "Skipping Git submodules setup", "Restoring cache", "Version: 13.4.0", "Git revision: 4e1f20da", "Git branch: 13-4-stable", "GO version: go1.13.8", and "Built: 2020-09-18T11:15:49+0000". The job status summary on the right indicates a duration of 20 seconds, a timeout of 7d (from project), and a runner of LAPTOP-1U2VTH48 (#2212). It also shows the commit 9d2421c3 and the job name "update app. properties".

Middle Screenshot: This screenshot is identical to the top one, showing the same build process and job status summary.

Bottom Screenshot: The terminal output shows the build process continuing with "48 [INFO] Changes detected - recompiling the module!", "49 [INFO] compiling 21 source files to C:\GitLab-Runner\builds\Qd5Ty2k\0\1426139\my_todolist\target\classes", "50 [INFO] -----", "51 [INFO] BUILD SUCCESS", "52 [INFO] -----", "53 [INFO] Total time: 7.362 s", "54 [INFO] Finished at: 2020-11-26T11:09:03+01:00", "55 [INFO] -----", "57 Saving cache", "58 Version: 13.4.0", "59 Git revision: 4e1f20da", "60 Git branch: 13-4-stable", "61 GO version: go1.13.8", "62 Built: 2020-09-18T11:15:49+0000", "63 OS/Arch: windows/386", "64 Creating cache default-2...", "65 Runtime platform arch=386 os=windows pid=13000 revision=4e1f20da version=13.4.0", "66 WARNING: .m2/repository/: no matching files", "67 target/: found 44 matching files and directories", "68 No URL provided, cache will be not uploaded to shared cache server. Cache will be stored only locally.", "69 Created cache", and "71 Job succeeded". The job status summary on the right indicates a duration of 20 seconds, a timeout of 7d (from project), and a runner of LAPTOP-1U2VTH48 (#2212). It also shows the commit 9d2421c3 and the job name "update app. properties". The pipeline status is "Pipeline #52834 for master" and the job status is "build - passed".

Diagram 6 – CI/CD integration test result

7. CI/CD integration Diagram

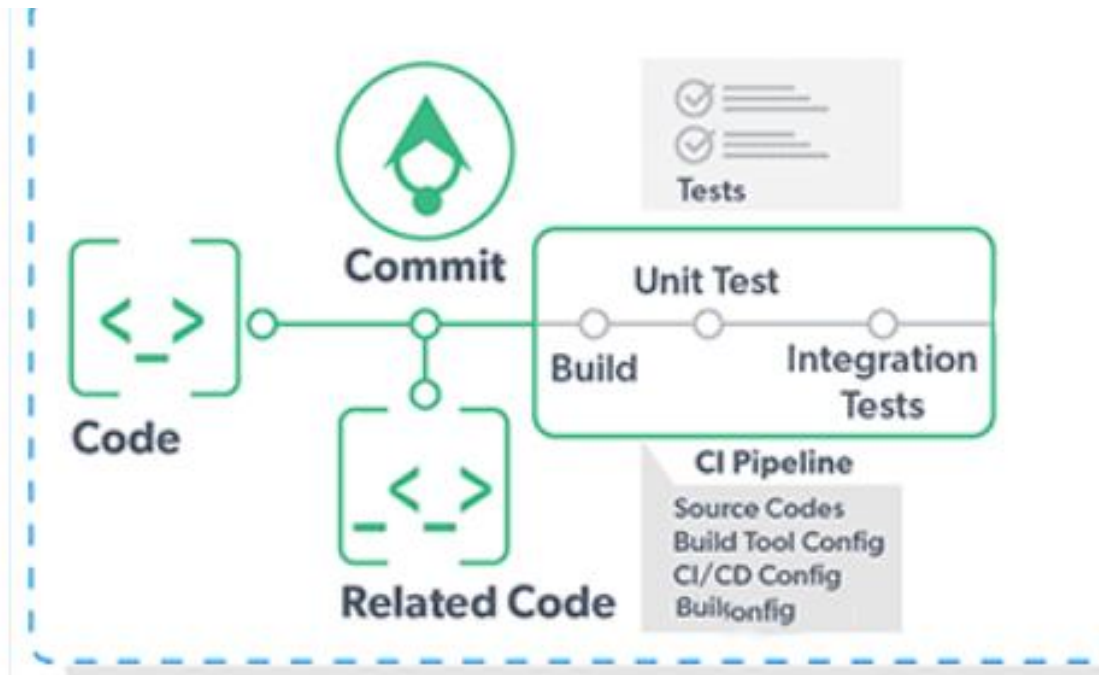


Diagram 7 – Ci/CD integration Diagram

8. SonarQube test report screenshot

The top screenshot shows the SonarQube dashboard for the 'ToDoList' project. The 'Measures' tab is active, displaying a 'Passed' status and various quality metrics. The metrics are as follows:

Metric	Value	Quality Gate
Bugs	0	Reliability A
Vulnerabilities	0	Security A
Security Hotspots	0	Security Review A
Code Smells	56	Maintainability A
Debt	7h 18min	
Coverage	0.0%	
Duplications	0.0%	

The bottom screenshot shows the SonarQube dashboard for the 'ToDoList' project, displaying a table of code quality metrics for the 'src' directory and 'pom.xml' file.

File	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
src	1,155	0	0	56	0	0.0%	0.0%
pom.xml	120	0	0	0	0	—	0.0%

2 of 2 shown

Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA.

```
MINGW64/c:/Users/wonda/Documents/Individual_project/my_todoist

$ mvn sonar:sonar \
> -Dsonar.projectKey=todoist \
> -Dsonar.host.url=http://localhost:9000 \
> -Dsonar.login=10d757e07b507eeb837809686c6508c769b421a8
[INFO] Scanning for projects...
[INFO] -----< wondl.example:Todoist >-----
[INFO] Building Todoist 0.0.1-SNAPSHOT
[INFO] [ jar ]-----
[INFO]
[INFO] --- sonar-maven-plugin:3.7.0.1746:sonar (default-cli) @ Todoist ---
[INFO] User cache: C:\Users\wonda\.sonar\cache
[INFO] SonarQube version: 8.5.1
[INFO] Default locale: "en_US", source code encoding: "UTF-8"
[INFO] Load global settings
[INFO] Load global settings (done) | time=156ms
[INFO] Server id: BF41A1F2-AXYZbIN3LKOYtewdRAaj
[INFO] User cache: C:\Users\wonda\.sonar\cache
[INFO] Load/download plugins
[INFO] Load plugins index
[INFO] Load plugins index (done) | time=103ms
[INFO] Load/download plugins (done) | time=249ms
[INFO] Process project properties
[INFO] Process project properties (done) | time=18ms
[INFO] Execute project builders
[INFO] Execute project builders (done) | time=2ms
[INFO] Project key: todoist
[INFO] Base dir: C:\Users\wonda\Documents\Individual_project\my_todoist
[INFO] Working dir: C:\Users\wonda\Documents\Individual_project\my_todoist\target\sonar
[INFO] Load project settings for component key: 'todoist'
[INFO] Load project settings for component key: 'todoist' (done) | time=30ms
[INFO] Load quality profiles
[INFO] Load quality profiles (done) | time=91ms
[INFO] Load active rules
[INFO] Load active rules (done) | time=1626ms
[INFO] Indexing files...
[INFO] Project configuration:
[INFO] 46 files indexed
[INFO] 0 files ignored because of scm ignore settings
[INFO] Quality profile for java: sonar way
[INFO] Quality profile for xml: sonar way
[INFO] ----- Run sensors on module Todoist
[INFO] Load metrics repository
[INFO] Load metrics repository (done) | time=30ms
[INFO] Sensor JavaSquidSensor [java]
[INFO] Configured Java source version (sonar.java.source): 11
[INFO] JavaClasspath initialization
[INFO] JavaClasspath initialization (done) | time=36ms
[INFO] JavaTestClasspath initialization
[INFO] JavaTestClasspath initialization (done) | time=20ms
[INFO] Java Main Files AST scan
[INFO] 36 source files to be analyzed
[INFO] Load project repositories
[INFO] Load project repositories (done) | time=20ms
[INFO] 36/36 source files have been analyzed
[INFO] Java Main Files AST scan (done) | time=7935ms
[INFO] Java Test Files AST scan
[INFO] 9 source files to be analyzed
[INFO] Java Test Files AST scan (done) | time=1332ms
[INFO] 9/9 source files have been analyzed
[INFO] Java Generated Files AST scan
[INFO] 0 source files to be analyzed
[INFO] Java Generated Files AST scan (done) | time=2ms
[INFO] Sensor JavaSquidSensor [java] (done) | time=9570ms
[INFO] Sensor CSS Rules [cssfamily]
[INFO] No CSS, PHP, HTML or VueJS files are found in the project. CSS analysis is skipped.
[INFO] Sensor JaCoCo XML Report importer [jacoco]
[INFO] 'sonar.coverage.jacoco.xmlReportPaths' is not defined. Using default locations: target/site/jacoco/jacoco.xml,target/site/jacoco-it/jacoco.xml,build/reports/jacoco/test/jacocoTestReport.xml
[INFO] No report imported, no coverage information will be imported by JaCoCo XML Report Importer
[INFO] Sensor JaCoCo XML Report Importer [jacoco] (done) | time=10ms
[INFO] 0/0 source files have been analyzed
[INFO] Sensor C# Properties [csharp]
[INFO] Sensor C# Properties [csharp] (done) | time=3ms
[INFO] Sensor SureFireSensor [java]
[INFO] parsing [C:\Users\wonda\Documents\Individual_project\my_todoist\target\surefire-reports]
[INFO] Sensor SureFireSensor [java] (done) | time=192ms
[INFO] Sensor JavaXmlSensor [java]
[INFO] 1/1 source files to be analyzed
[INFO] Sensor JavaXmlSensor [java] (done) | time=324ms
[INFO] 1/1 source files have been analyzed
[INFO] Sensor HTML [web]
[INFO] Sensor HTML [web] (done) | time=6ms
[INFO] Sensor XML Sensor [xml]
[INFO] 1 source files to be analyzed
[INFO] Sensor XML Sensor [xml] (done) | time=202ms
[INFO] 1/1 source files have been analyzed
[INFO] Sensor VB.NET Properties [vbnet]
[INFO] Sensor VB.NET Properties [vbnet] (done) | time=2ms
[INFO] ----- Run sensors on project
[INFO] Sensor Zero Coverage Sensor
[INFO] Sensor Zero Coverage Sensor (done) | time=136ms
[INFO] Sensor Java CPD Block Indexer
[INFO] Sensor Java CPD Block Indexer (done) | time=213ms
[INFO] SCM Publisher: SCM provider for this project is: git
[INFO] SCM Publisher 46 source files to be analyzed
[INFO] SCM Publisher 46/46 source files have been analyzed (done) | time=3189ms
[INFO] CPD Executor 13 files had no CPD blocks
[INFO] CPD Executor Calculating CPD for 23 files
[INFO] CPD Executor CPD calculation finished (done) | time=32ms
[INFO] Analysis report generated in 230ms, dir size=228 KB
[INFO] Analysis report compressed in 505ms, zip size=119 KB
[INFO] Analysis report uploaded in 52ms
[INFO] ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard?id=todoist
[INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
[INFO] More about the report processing at http://localhost:9000/ap/ce/task?id=Axau08EwlyR731wb_bh3
[INFO] Analysis total time: 20.392 s
[INFO] -----
[INFO] BUILD SUCCESS
```

Diagram 8 – SonarQube report screen shot

I. Reference

- *Angular*. (n.d.). Angular Tour of Hero. Retrieved March 9, 2020, from <https://angular.io/tutorial>
- *MySQL Tutorial - Learn MySQL Fast, Easy and Fun*. (2020, March 30). MySQL Tutorial. <https://www.mysqltutorial.org/>
- *Spring / Guides*. (n.d.). Spring. Retrieved October 1, 2020, from <https://spring.io/guides#tutorials>
- *GitLab CI/CD*. (n.d.-b). GitLab. Retrieved September 8, 2020, from <https://docs.gitlab.com/ee/ci/>