1. **Introduction**

The technical requirements state the following:

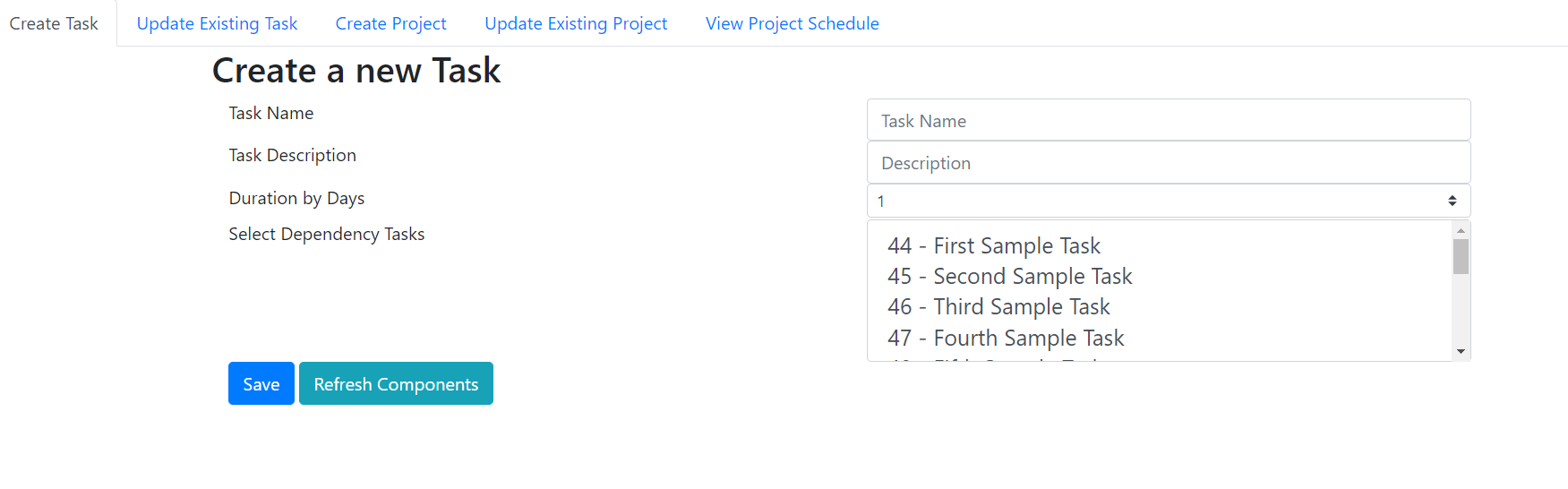
1. We need to calculate calendar schedules for project plans
2. Each project plan consists of tasks. Every task has a certain duration.
3. A task can depend on zero or more other tasks. If a task depends on some other tasks, it can only be started after these tasks are completed
4. So, for a set of tasks (with durations and dependencies), the solution for the challenge should generate a schedule, i.e. assign Start and End Dates for every task
5. It is ok to have a console app

For the solution, I have created three main components: spring boot microservice application (**projectscheduler**), mysql database (**project-scheduler-db**) to store the project info, project tasks and task infos and dependencies, and frontend UI using react-js (**project-scheduler-app**).

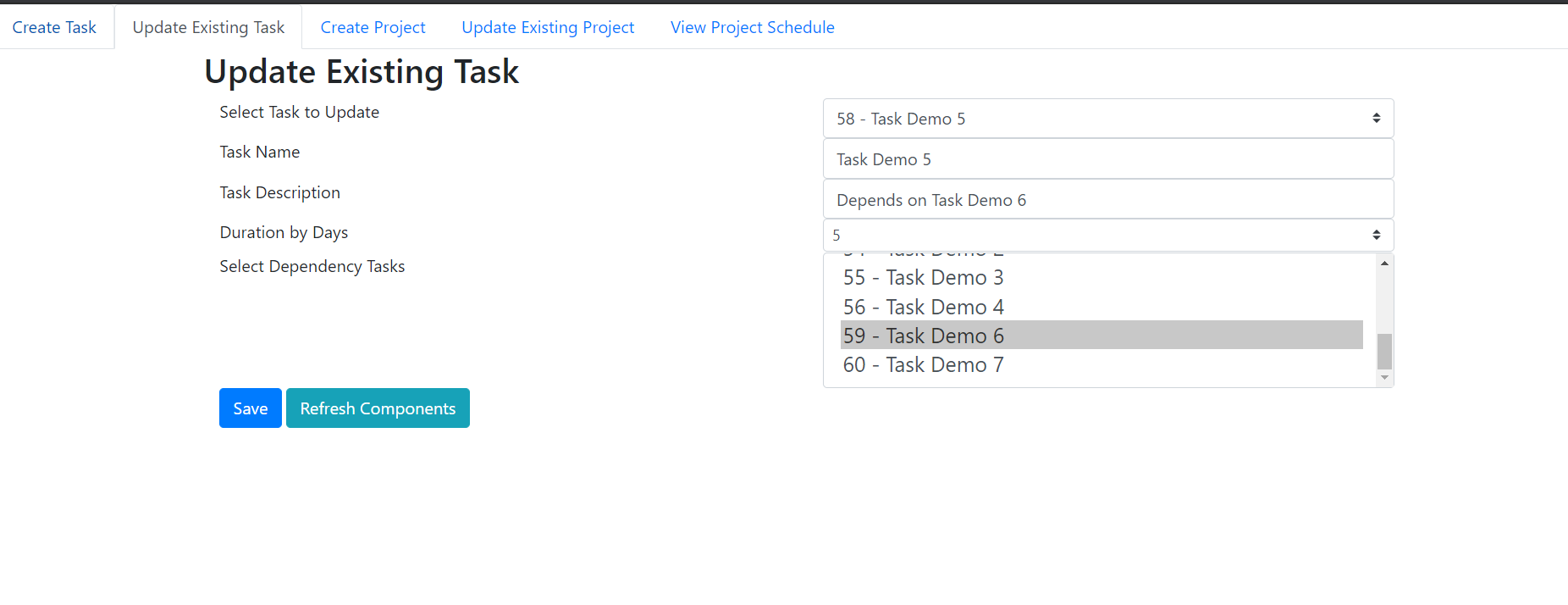
In order to have a working application running in the internet, I have deployed project-scheduler-app into github pages, projectscheduler and project-scheduler-db into my Google Kubernetes Engine wrapped into separate docker containers, and their respective docker images are hosted into my DockerHub account so that they can be fetched into my GKE.

1. **project-scheduler-app (Frontend UI)**
   1. **Screens**

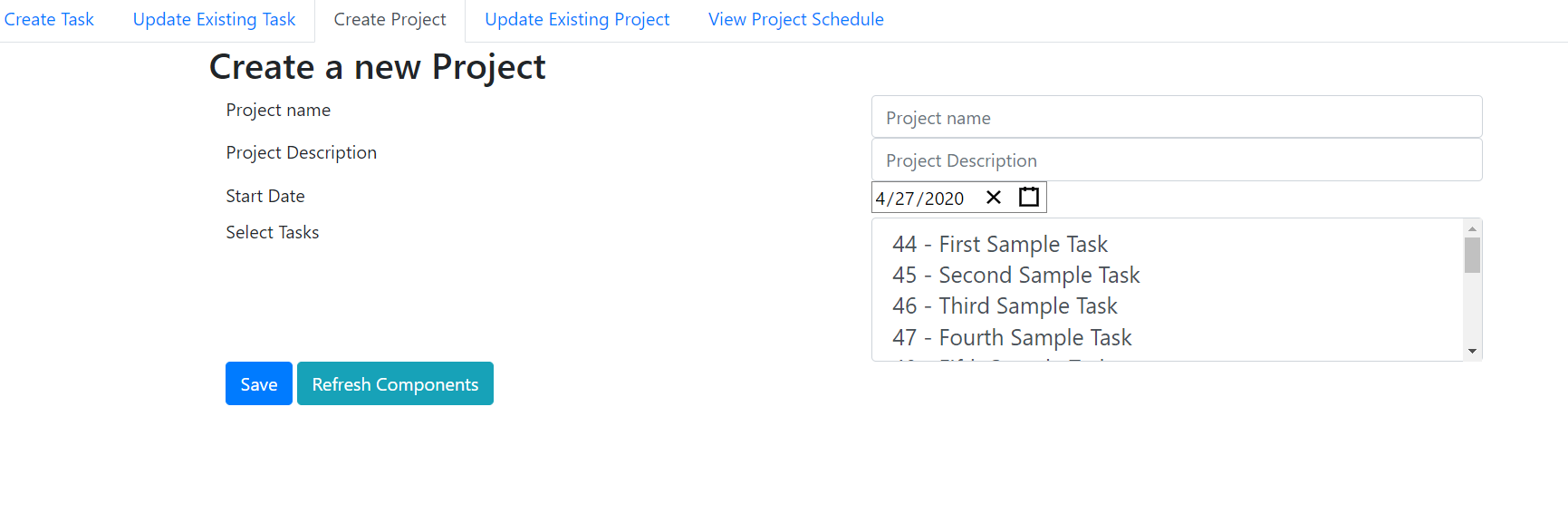
The screens are the following:



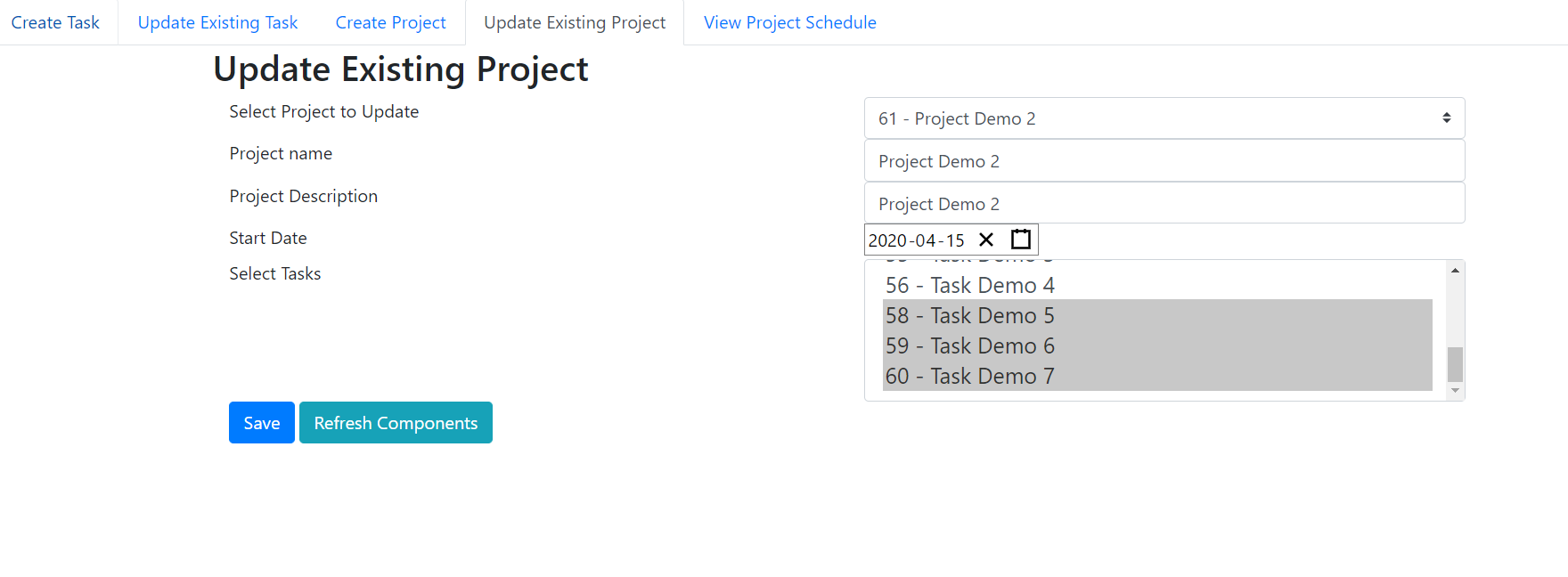
This is for adding new tasks, also you can select available tasks for its dependency.



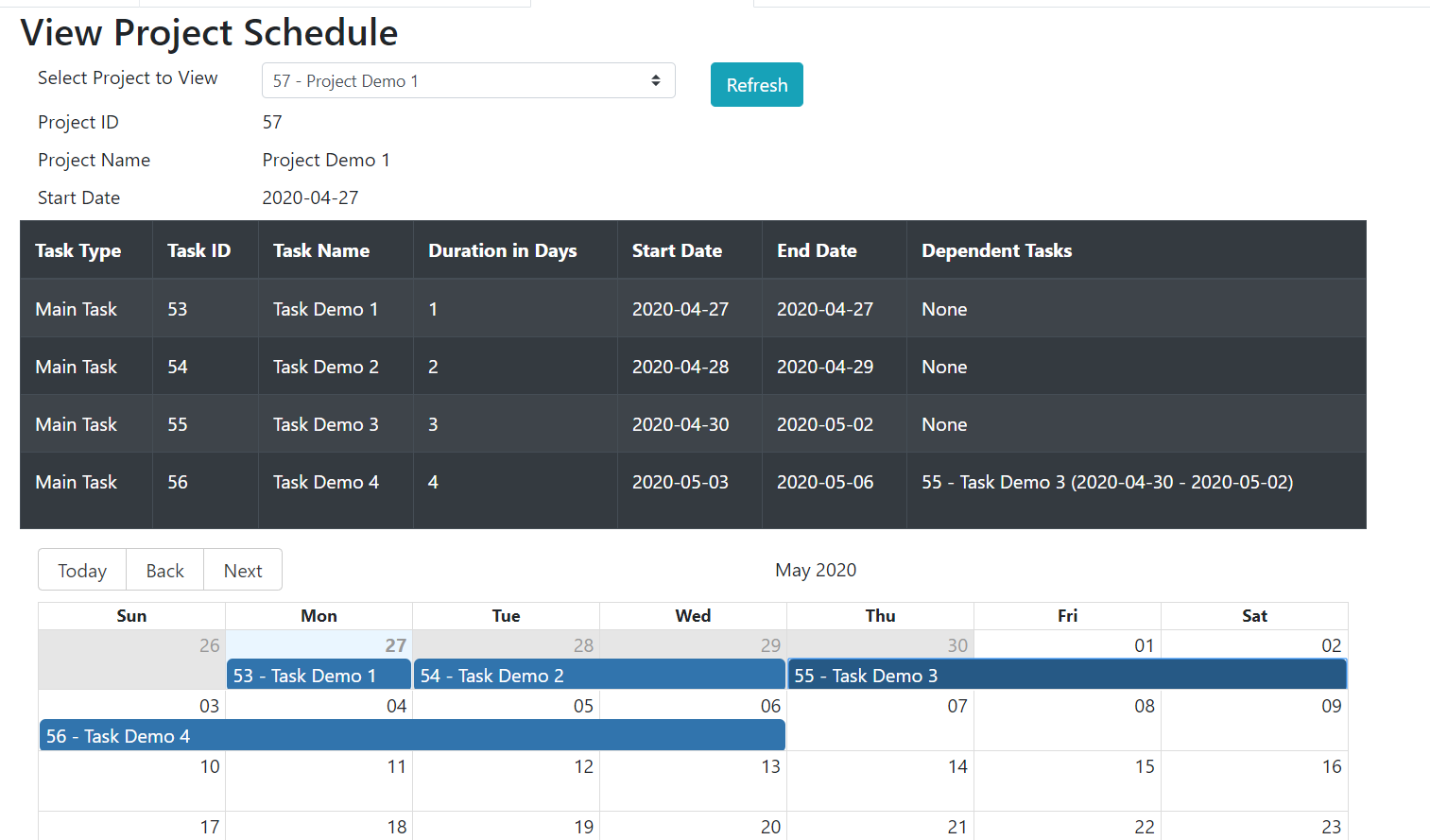
This is for updating existing tasks, if we need to change the information like task name, description, duration and even the dependency tasks.



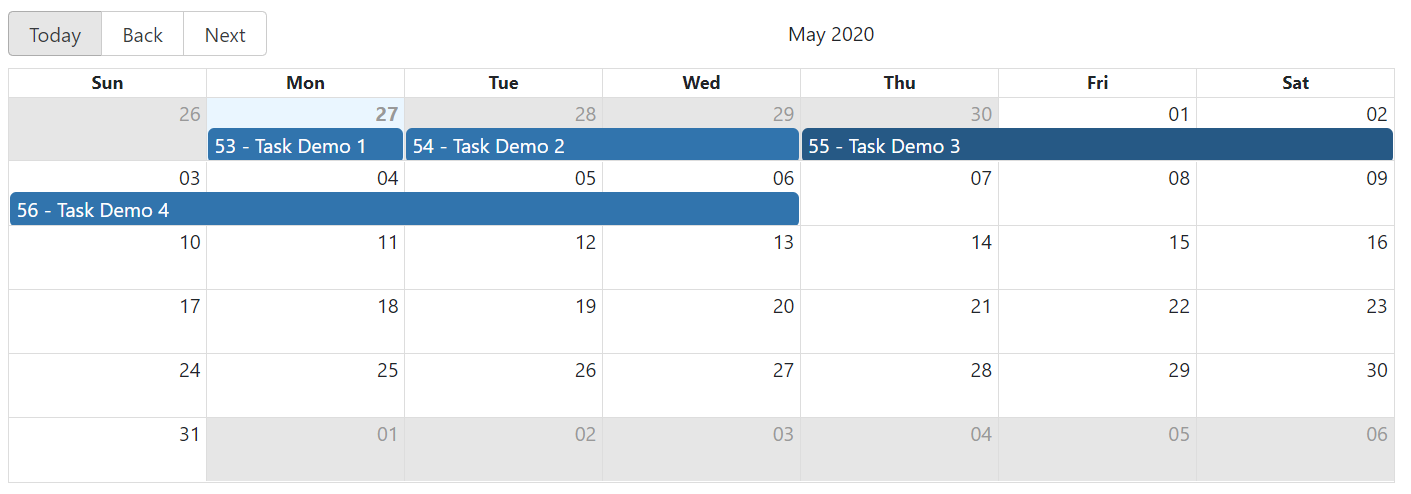
This screen is to create a new project, we can set the project name, description, start date (mandatory) and the dependent tasks of the project.



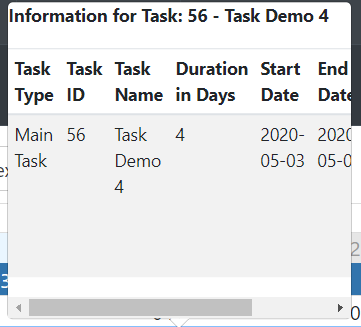
This screen is to update an existing project, we can update the name, description, start date and even the dependency tasks.



This is the main logic screen, where we can view the schedule based on the duration and dependency tasks for each task.



We have a calendar view where we can view the schedule in a big calendar. If youll click the tasks in the calendar it will pop over a small pane that containing the information:



**IF YOU WANT TO PLAY ON THE APPLICATION, PLEASE FOLLOW THIS LINK:** <https://jeremypunsalaniscalevlocity.github.io/project-scheduler-app/>

* 1. **Components used**

1. ReactJS
   1. create-react-app
   2. react-bootstrap
   3. react-bootstrap-date-picker
   4. react-big-calendar
   5. gh-pages (to host the app)
   6. **Github Repo**

<https://github.com/jeremypunsalaniscalevlocity/project-scheduler-app>

1. **projectscheduler (Mircroservice backend)**
   1. **REST APIs**
      1. **/rest/project/save**

This is used to create or update a project.

RequestBody:

projectId (number): project id (this is optional if we will create the project as the project id is incremented by jpa / mysql

projectName (string): project name

projectDescription (string): description

startDate (date): start date of the project (this is mandatory field as this will be used to calculate the start and end dates of tasks)

taskViews (list of objects): list of dependent tasks of the project

* + 1. **/rest/project/id/{id}**

This is used for retrieving project information and tasks under the project.

Parameters:

{id} (integer): the project id to be retrieved

* + 1. **/rest/project/all**

This is used to get all projects without their task dependencies.

* + 1. **/rest/project/all/withtasks**

This is used to get all projects with their task dependencies.

* + 1. **/rest/task/save**

This is used to create or update a task information.

RequestBody:

taskId (integer): task id (this is optional if creating a task as the id is incremented by jpa / mysql)

taskName (string): task name

description (string): task description

duration (integer): task duration in days

startDate (date): start date of the task

endDate (date): end date of the task

dependencies (list of Objects): list of task objects that are dependent to the given task

* + 1. **/rest/task/id/{id}**

This is used to retrieve a task by its id.

Parameters:

{id} (integer): the task id to be retrieved

* + 1. **/rest/task/all**

Retrieves all tasks with their ids and task names.

* + 1. **/rest/task/eligible/{id}**

Retrieves the eligible tasks that can be set as dependent tasks to the given task id, this is used when setting the dependent tasks (only applicable on task creation).

There are two simple rules: (a) Cannot set self and (b) cannot set any other tasks that are dependent to given task (ie if task 1 depends on task 2, task 2 cannot set task 1 as dependent task).

Parameters:

{id} (integer): task id

* + 1. **/rest/scheduler/view/{id}**

**This is the main logic of the application.** It shows the project with its tasks and dependency tasks and their starting date and ending dates.

Parameters:

{id} (integer): project id to be set

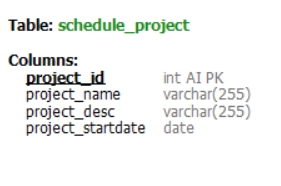
* + 1. **/rest/scheduler/view/all**

Retrieves all project schedule. This is by extension of /rest/scheduler/view/{id}, we only iterate all projects with dependent tasks.

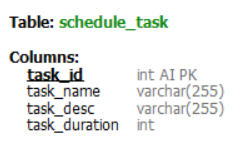
* 1. **Components used** 
     1. **Spring Boot**
        1. **spring-boot-starter-parent**
        2. **spring-boot-starter-data-jpa**
        3. **spring-boot-starter-web**
        4. **junit**
        5. **mysql-connector-java**
        6. **spring-boot-maven-plugin**
        7. **maven-surefire-plugin**
  2. **Github Repo**

<https://github.com/jeremypunsalaniscalevlocity/projectscheduler>

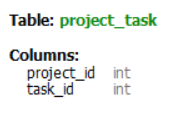
1. **project-scheduler-db (MySQL DB)**
   1. **Tables**



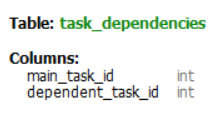
This defines the Project entity.



This defines the Task entity. Take note that we don’t save the startDate and endDate as this needs to be calculated when we want to view the project. Also we can change the project start date anytime so it is better to compute it on the fly.



This is a composite table where it defines the dependency tasks for each project.



This is a composite table where it defines the dependency tasks of each task.

* 1. **Spring JPA Entity Mapping to Tables**

Please refer to the github repo of projectscheduler in the package **com.iscalesolutions.vlocity.projectscheduler.entities**

* 1. **Setup script and Dockerizing MySQL (with Github Repo)**

You can find the setup script and the docker file to build a mysql into an image and deploy/use as container in the github repo:

<https://github.com/jeremypunsalaniscalevlocity/project-scheduler-db>

I have decided to dockerize mysql (For actual application it is not adviced to deploy stateful components as it is not a good practice, but rather create a proxy container to connect to a data source) so that you can test the application in local (either you pull the docker image and use it as the db of this application or execute the setup script into your local instance of MySQL).

1. **Installation**
   1. **projectscheduler**

By cloning code into your local

1. Git clone (<https://github.com/jeremypunsalaniscalevlocity/projectscheduler.git>)

2. Import project as maven project

3. run mvn-clean-install

4. run java -jar /target/projectscheduler.jar

By pulling docker image into your local instance of docker (you also need to pull the project-scheduler-db image together)

1. Pull projectscheduler image from dockerhub (command: docker pull jeremypunsalaniscalevlocity/projectscheduler)

2. Just run the container from your local docker instance

TAKE NOTE THAT YOU NEED TO SPECIFY THE LOCATION OF THE DB IN APPLICATION.YML BEFORE RUNNING OR BUILDING THE APP.

* 1. **project-scheduler-db**

By running setup script

1. Git clone (<https://github.com/jeremypunsalaniscalevlocity/project-scheduler-db.git>)

2. Run setup.sql

By pulling docker image into your local instance of docker (you also need to pull the projectscheduler image together)

1.Pull project-scheduler-db image from dockerhub (command: docker pull jeremypunsalaniscalevlocity/project-scheduler-db)

2.Just run the container from your local docker instance

* 1. **project-scheduler-app**

You can only run this app by cloning the code from github:

<https://github.com/jeremypunsalaniscalevlocity/project-scheduler-app.git> (make sure you have npm installed in your local and update .env file to point the spring boot app to your local instance)

1. **Misc Information**

**A. URLs**

**GitHub:**

<https://github.com/jeremypunsalaniscalevlocity/projectscheduler>

<https://github.com/jeremypunsalaniscalevlocity/project-scheduler-db>

<https://github.com/jeremypunsalaniscalevlocity/project-scheduler-app>

**DockerHub:**

<https://hub.docker.com/repository/docker/jeremypunsalaniscalevlocity/projectscheduler>

<https://hub.docker.com/repository/docker/jeremypunsalaniscalevlocity/project-scheduler-db>

**projectscheduler container deployed into GKE:**

[http://104.198.53.213:8080/{rest-api}](http://104.198.53.213:8080/%7brest-api%7d)

**Live projectscheduler served in HTTPS (used NGROK in a GCP instance as tunnel to serve the projectscheduler into https):**

[https://cd3236e9.ngrok.io/{rest-api}](https://cd3236e9.ngrok.io/%7brest-api%7d)

**where {rest-api} is the rest api defined in III.a**

**Live project-scheduler-app deployed to github pages (connected to projectscheduler microservice in GKE and project-scheduler-db container):**

<https://jeremypunsalaniscalevlocity.github.io/project-scheduler-app/>

**B. DockerHub**

1. **I used dockerhub to store my containers so that I can deploy my containers into GKE.**
2. **Also take note that I connected my github repo to projectscheduler hub so that when I check in, it will automatically build the container. Although the building of the app still needs to be done locally, we just need to deploy the jar output altogether.**

**C. GCP and GKE**

1. **I created a cluster to host the containers (project-scheduler-db and projectscheduler). These containers point to my dockerhub repo.**
2. **I also created one GCP instance to host my ngrok. I used NGROK to serve the exposed service of projectscheduler as HTTPS (I had some issues on using the exposed IP as http when my project-scheduler-app is rendered on github pages as it is using HTTPS and my microservice uses HTTP).**

**If you have questions regarding the app, you may reach me at** [**jem.punsalan@yahoo.com**](mailto:jem.punsalan@yahoo.com) **or thru skype at jem.punsalan**

**Thank you!**