SPRING BOOT & REACT TODO APP

Ahmet Hakan Beşel

TECH STACK

- Spring Boot with Maven
- Swagger
- PostgreSQL
- React
- Chakra UI
- Vite
- TypeScript
- Docker
- Postman

TASK CONTROLLER

- Controls the interaction between client and task service.
- Available endpoints:
  - GET /api/tasks/{id}
  - PUT /api/tasks/{id}
  - DELETE /api/tasks/{id}
  - GET /api/tasks
  - POST /api/tasks
  - DELETE /api/tasks/

```java
@Operation(
        summary = "List all tasks",
        description = "Retrieves all tasks"
)
@ApiResponse(
        responseCode = "200",
        description = "200 SUCCESS"
)
@GetMapping
public TaskResponse getAllTasks(
        @RequestParam(value = "completed", required = false) String completed,
        @RequestParam(value = "pageNo", defaultValue = AppConstants.DEFAULT_PAGE_NUMBER, required = false) int pageNo,
        @RequestParam(value = "pageSize", defaultValue = AppConstants.DEFAULT_PAGE_SIZE, required = false) int pageSize,
        @RequestParam(value = "sortBy", defaultValue = AppConstants.DEFAULT_SORT_BY, required = false) String sortBy,
        @RequestParam(value = "sortDir", defaultValue = AppConstants.DEFAULT_SORT_DIRECTION, required = false) String sortDir
) {
    if (completed != null) {
        return taskService.getAllTasksByCompleted(completed.equals("true"), pageNo, pageSize, sortBy, sortDir);
    }
    return taskService.getAllTasks(pageNo, pageSize, sortBy, sortDir);
}

@Operation(
        summary = "Get a task by id",
        description = "Retrieves a task by id"
)
@ApiResponse(
        responseCode = "200",
        description = "200 SUCCESS"
)
@GetMapping("/{id}")
public ResponseEntity<TaskDto> getTaskById(@PathVariable(name = "id") long id) {
    return ResponseEntity.ok(taskService.getTaskById(id));
}

@Operation(
        summary = "Update a task",
        description = "Updates a task with the given details"
)
@ApiResponse(
        responseCode = "200",
        description = "200 SUCCESS"
)
@PutMapping("/{id}")
public ResponseEntity<TaskDto> updateTask(@Valid @RequestBody TaskDto taskDto, @PathVariable(name = "id") long id) {
    TaskDto taskResponse = taskService.updateTask(taskDto, id);
    return new ResponseEntity<>(taskResponse, HttpStatus.OK);
}

@Operation(
        summary = "Delete a task",
        description = "Deletes the corresponding task"
)
@ApiResponse(
        responseCode = "200",
        description = "200 SUCCESS"
)
@DeleteMapping("/{id}")
public ResponseEntity<String> deleteTask(@PathVariable(name = "id") long id) {
    taskService.deleteTaskById(id);
    return new ResponseEntity<>("The task deleted successfully.", HttpStatus.OK);
}
```

TASK SERVICE

- Contains the main business logic.

```java
public interface TaskService {
    TaskDto createTask(TaskDto taskDto);

    TaskResponse getAllTasks(int pageNo, int pageSize, String sortBy, String sortDir);

    TaskDto getTaskById(long id);

    TaskDto updateTask(TaskDto postDto, long id);

    void deleteTaskById(long id);

    void deleteTasksByCompleted(boolean completed);

    TaskResponse getAllTasksByCompleted(boolean completed, int pageNo, int pageSize, String sortBy, String sortDir);
}
```

TASK SERVICE

- Contains the main business logic.

```java
public interface TaskService {
    TaskDto createTask(TaskDto taskDto);

    TaskResponse getAllTasks(int pageNo, int pageSize, String sortBy, String sortDir);

    TaskDto getTaskById(long id);

    TaskDto updateTask(TaskDto postDto, long id);

    void deleteTaskById(long id);

    void deleteTasksByCompleted(boolean completed);

    TaskResponse getAllTasksByCompleted(boolean completed, int pageNo, int pageSize, String sortBy, String sortDir);
}
```

TASK REPOSITORY

- Handles database operations.
- An interface extends JPA Repository.

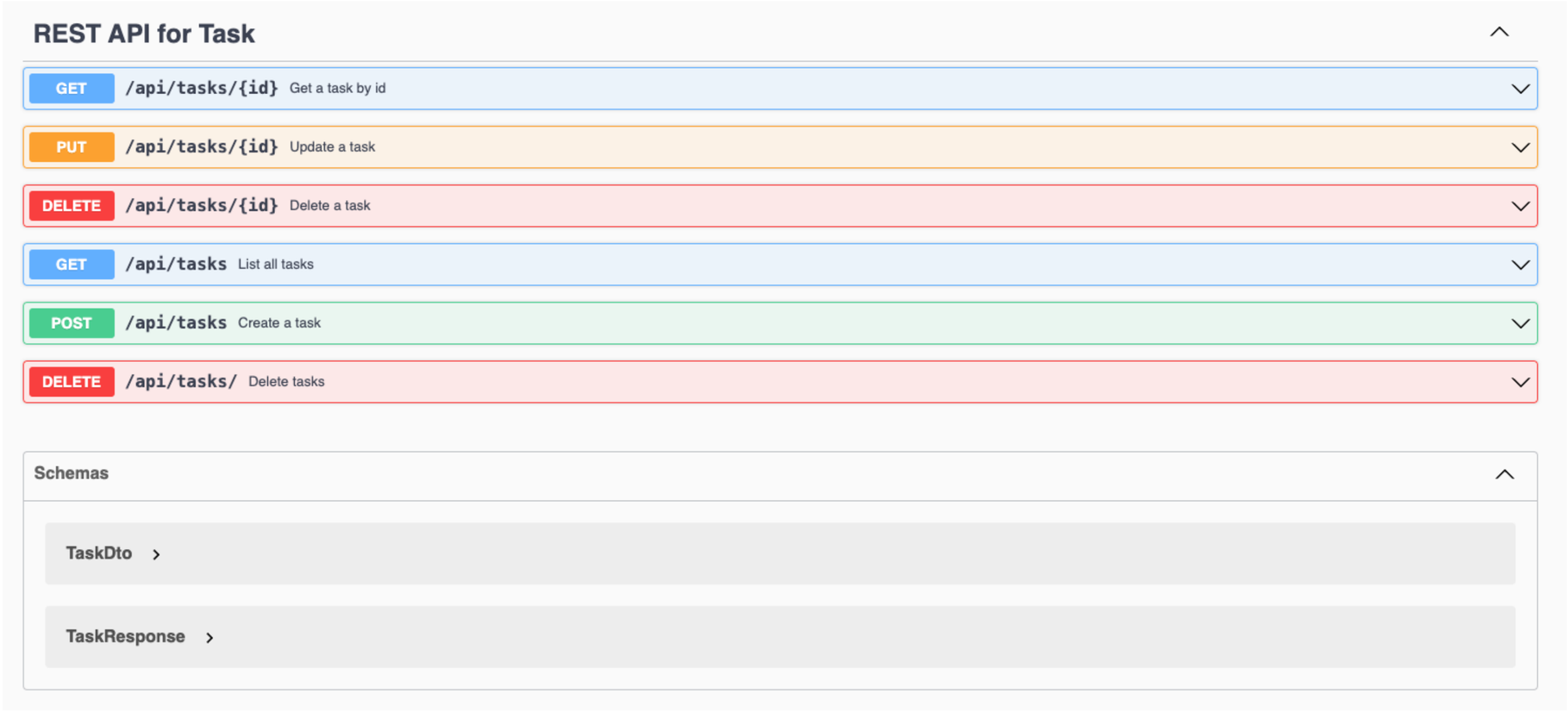
FRONTEND COMPONENTS

- TaskList
  - TaskCard
- NewTaskForm

```
return (
  <>
    <Container centerContent={true}>
      <VStack
        divider={<StackDivider borderColor='gray.200' />}
        spacing={8}
        align='stretch'
        width='100%'
      >
        <Box textAlign='center'>
          <Text fontSize='4xl'>TODO List</Text>
        </Box>
        <Box>
          <NewTaskForm onSave={handleSave}></NewTaskForm>
        </Box>
        <Box>
          <HStack spacing={0} justifyContent='space-between'>
            <Button
              onClick={() => setFilter('all')}
              colorScheme={filter === 'all' ? 'blue' : 'gray'}
              flex='1'
              mr={2}
            >
              All Tasks ({tasks.length})
            </Button>
            <Button
              onClick={() => setFilter('completed')}
              colorScheme={filter === 'completed' ? 'green' : 'gray'}
              flex='1'
              mr={2}
            >
              Completed ({tasks.filter(task => task.completed).length})
            </Button>
            <Button
              onClick={() => setFilter('pending')}
              colorScheme={filter === 'pending' ? 'yellow' : 'gray'}
              flex='1'
            >
              Pending ({tasks.filter(task => !task.completed).length})
            </Button>
          </HStack>
        </Box>
        <Box>
          <TaskList tasks={filteredTasks} onUpdate={handleUpdate} onDelete={handleDelete} />
        </Box>
        <Box>
          <HStack spacing={0} justifyContent='space-between'>
            <Button
              leftIcon={<DeleteIcon />}
              onClick={() => batchDelete('true')}
              colorScheme='red'
              flex='1'
              mr={2}
            >
              Delete completed
            </Button>
            <Button
              leftIcon={<DeleteIcon />}
              onClick={() => batchDelete('all')}
              colorScheme='red'
              flex='1'
              mr={2}
            >
              Delete all
            </Button>
          </HStack>
        </Box>
      </VStack>
    </Container>
  </>
);
};

export default App;
```

SWAGGER UI


REST API for Task
GET /api/tasks/{id} Get a task by id
PUT /api/tasks/{id} Update a task
DELETE /api/tasks/{id} Delete a task
GET /api/tasks List all tasks
POST /api/tasks Create a task
DELETE /api/tasks/ Delete tasks
Schemas
TaskDto
TaskResponse

RUN THE BACKEND

1. Initialize a PostgreSQL database
2. Set Spring app properties
3. ***mvn spring-boot:run***

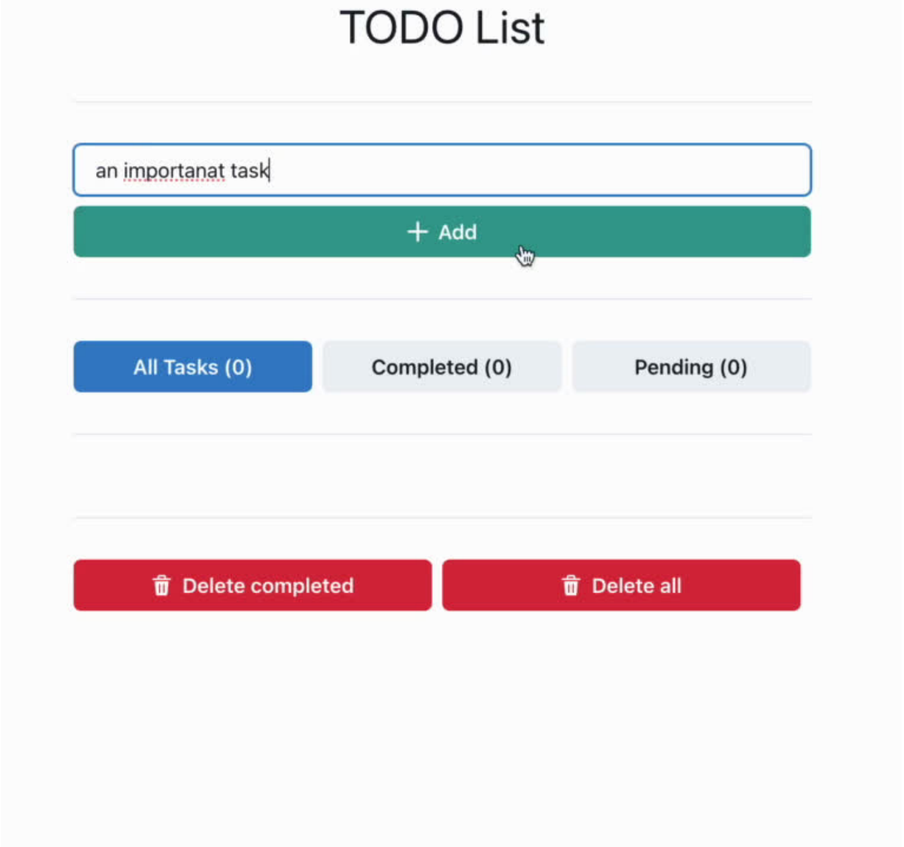
RUN THE FRONTEND

1. *bun install*
2. Set the API URL in `.env`
3. ***bun run dev***

DEPLOYMENT

1. mvn clean install
2. Set variables in `.env`
3. ***docker compose up -d***

DEMO


TODO List
an importanat task
+ Add
All Tasks (0)
Completed (0)
Pending (0)
Delete completed
Delete all

THANK YOU

Have any question?

ahmethakan.besel@gmail.com
github.com/ahmethakanbesel