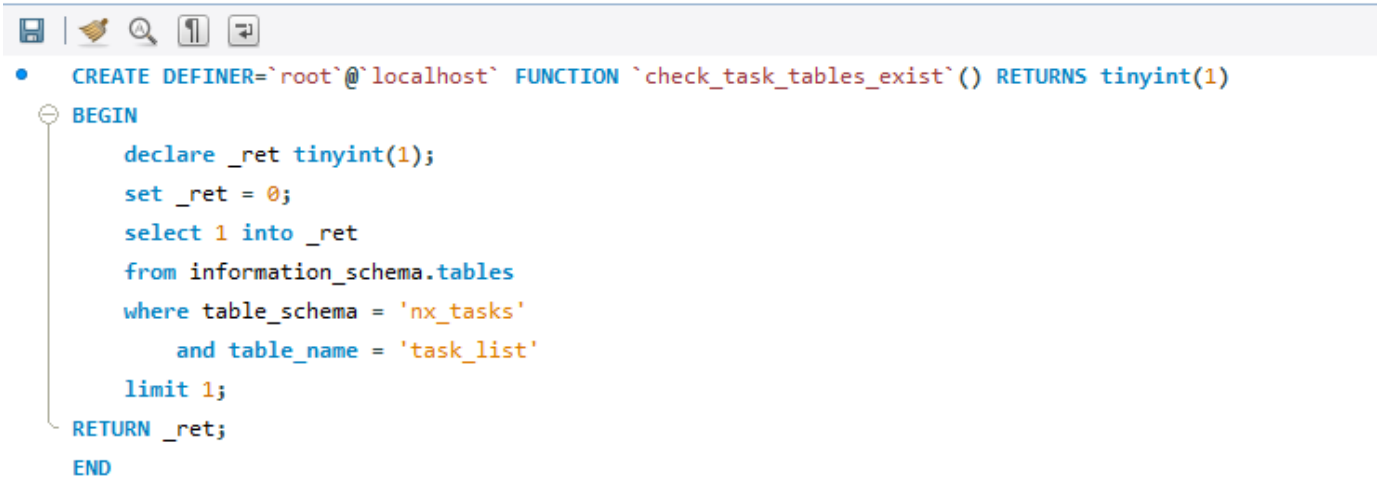
Day 3

Today I wrote the MySQL SPs and wrote the repository code to call them from my API.

For the sake of security and to avoid the potential of SQL injection attacks, I have decided to use Stored Procedures to write information into the DB.

The Stored Procedures will simply take in a set of parameters which will be used to assign the data to each relevant field.

The other advantage of this approach is that the API doesn't need to know the table structure and inner workings of the DB, as long as it can call the Stored Procedure everything will work fine. I have also designed the Stored Procedures so that they will create the relevant tables if the tables don't already exist. This will help when deploying the application to a server because the DB is partly self configuring.



```
CREATE DEFINER=`root`@`localhost` FUNCTION `check_task_tables_exist`() RETURNS tinyint(1)
BEGIN
    declare _ret tinyint(1);
    set _ret = 0;
    select 1 into _ret
    from information_schema.tables
    where table_schema = 'nx_tasks'
        and table_name = 'task_list'
    limit 1;
RETURN _ret;
END
```

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `create_tables`()
BEGIN
    CREATE TABLE `task_list` (
  `id` varchar(36) primary key not null,
  `title` varchar(50) not null
);
commit;

CREATE TABLE `task` (
  `id` varchar(36) primary key not null,
  `taskListId` varchar(36) not null,
  `title` varchar(50) not null,
  `position` integer,
  `notes` varchar(255),
  `due` datetime,
  `completed` datetime,
  `hidden` boolean
);

ALTER TABLE `task` ADD FOREIGN KEY (`taskListId`) REFERENCES `task_list` (`id`);
commit;
END
```

My Java API uses JDBC with the MySQL connector to communicate with MySQL. Each SP has an endpoint on the API, so all of them can be called via postman / fe.

```java
public TaskListDto insertTaskList(TaskListDto taskList) {
    SimpleJdbcCall simpleJdbcCall = new SimpleJdbcCall(jdbcTemplate.getDataSource()).withProcedureName("insert_task_list");
    SqlParameterSource in = new MapSqlParameterSource()
            .addValue( paramName: "pId", taskList.getId())
            .addValue( paramName: "pTitle", taskList.getTitle());
    Map<String, Object> out = simpleJdbcCall.execute(in);
    List<Object> resultsSet = (List<Object>) out.get("#result-set-1");
    if (!resultsSet.isEmpty()) {
        LinkedCaseInsensitiveMap<String> result = (LinkedCaseInsensitiveMap<String>) resultsSet.get(0);
        return new TaskListDto(UUID.fromString(result.get("id")), result.get("title"));
    } else {
        return null;
    }
}
```

Here are to postman collections that I have created to test my DB and API functionality.

The database will obviously, get or update the information in the database and the API will control how the output from the Stored Procedure is presented to the front end, i.e. in a format that it can read and interpret.