Planning application description:

I used Spring Boot to develop the application. I chose Maven as my building system, then H2 in-memory database as data storage and embedded Tomcat server, available via Spring Boot.

To build the application just type "mvn spring-boot:run" in project's main directory. If the project was built properly, you should see two lines on the end of all the messages:

"
[...]
Tomcat started on port(s): 8080 (http)
Started PlanerApplication in 2.582 seconds (JVM running for 2.906).
"

Then go to localhost:8080 to see the index of the application.

Application was made in Model-View-Controller architecture.

I created one model - Meeting. It has fields like id, name, description, date and time, meeting room ID and duration. These values tell me everything about a single meeting. With a model, I created table Meetings in my H2 Database (I used H2 because I found some sample projects and tutorials with in-memory H2 database and it was easy to use it) and after creating a meeting, it was translated via Object-Relational Mapping into a record in database and inserted into the table.

I decided to create three views:
- index.html - you see it only at the beginning
- addMeeting.html - this is the view I see, when I want to create a new meeting. There is a form with submit button and a single button on the top, that allows me to go to viewMeetings.
- viewMeetings.html - this view has a button which allows me to go to creating a new meeting and a table with all the records from database, from table Meetings. Every row in table has special "Delete" button, which, when clicked, will delete this meeting from database.

Then, I implemented two controllers - one - HomeController just returns view index on main page (localhost:8080) - for testing purposes, just to know how a controller (and further, two controllers in one project) works. Second - meetingController - was created to handle all the Posts and Gets about meetings, ie. adding, deleting or viewing the meetings.

At first I wanted to make two tables (and two models) - Meeting and MeetingsRoom, but the second one didn't seem to be useful in current scope of the project. While making potentially bigger system, I would implement two models, because it would be easier to develop them. But in this application, it would only make everything more complicated.

I didn't make documentation of any getters or setters, because they are self-descriptive. Some more complicated functions or classes have javadoc above them to make it easier to read the code.