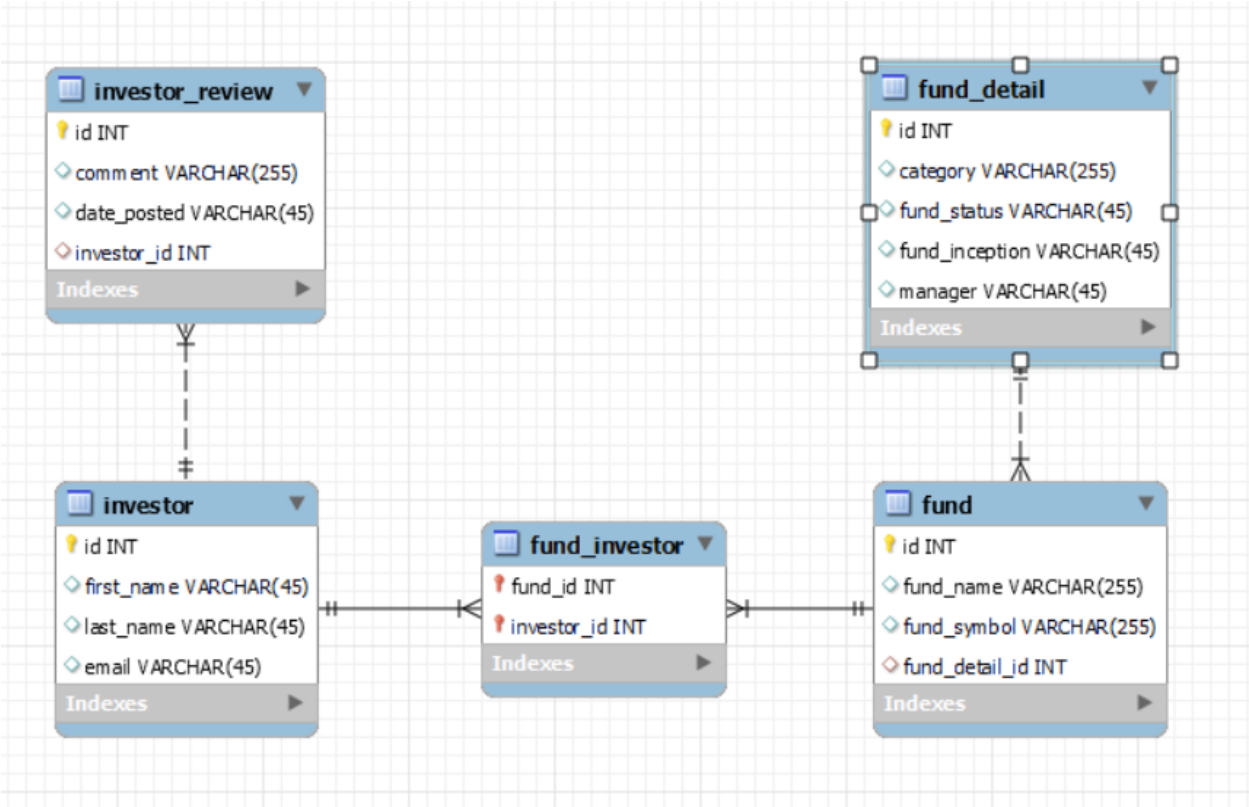
## Database Schema




- The image above description/database schema of the Many-to-Many relationship between Funds and investors. **Investors can have many funds, and funds can have many investors**. A Join table created for this relationship, and associations can be described further via REST requests through different HTTP methods

- I added additional tables (fund_detail and investor_review) to represent possible One-to-Many relationships. Those are less important, but I wanted to play around with different relationships possible between our entities. They don't have any preloaded contents.

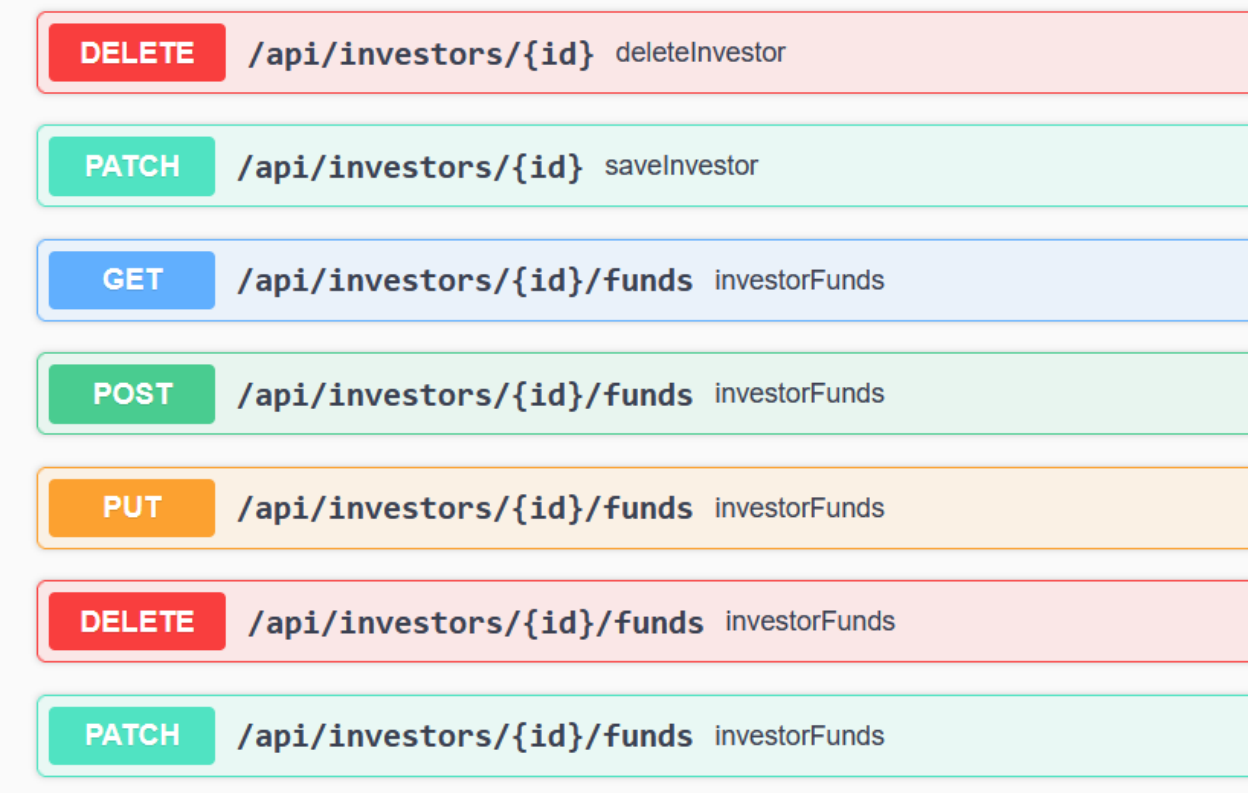
## REST Design

- I employed a few Spring starters to help with this design. Spring-Data-Rest allows us to gain a few REST actions automatically, so I employed those REST actions and decided to spend more time on schema design, instead.

- If you visit **http://localhost:6808/swagger-ui/**, you can see the beginnings of a SwaggerUI REST definition page. It is best practice to define your REST API design first before backend design (Using OpenAPI spec), but I jumped to the backend first for this

demonstration and added the SwaggerUI after. This is definitely a consideration for the future

- There are associations between our entities that can be described. For example, a PUT request to http://localhost:6808/api/Investor/1/Investor-review will allow you to define an association between an investor of id value 1, and a specific review. Additional examples are found below




**Security**

- I was looking to implement basic security, at least some basic auth. The Spring-Starter-Security package would be helpful for this. For future security considerations, I would consider implementing tokens for authentication, basic auth, and enabling SSL/TLS.

**FUTURE CONSIDERATIONS**

- Better define the REST API spec. It would be better to even define the Swagger YAML first, then export the Spring MVC REST API stubs. This can be done on editor.swagger.io

- Further refine security.