- Duplicate yesterday's project, and continue here
- Your goal is to implement the whole authentication, authorization functionality
- *Hint*: begin by creating 3 database tables: `users`, `roles` and `users_roles`. Create many-to-many relationship
  - After that, start working on your Spring Boot project. You may start by implementing the `User` entity
- End result:
  - Use DTOs for all of the endpoints
  - Non-authenticated people must be allowed to `POST` to `/api/users`, in order to create a user
    * Once a user is created, return everything but the password back to the client. You will want to create a separate DTO to achieve this
  - Only admins may be allowed to query the `GET` endpoint `/api/users`
  - Only admins may be allowed to query the `GET` endpoint `/api/users/{id}`
  - Only admins may be allowed to use `PUT` on the endpoint `/api/users/{id}`
  - Only admins may be allowed to call `DELETE` on the endpoint `/api/users/{id}`
  - Test everything out, and make sure it all works
    * If client is not authenticated, they shoud expect to see `401 Unauthorized`. If client is not able to use the resource because of lack of authority, they should expect to see `403 Forbidden`.
      · Of course, all the other bean validation should work too, such as `@NotNull` and so; should return `400 Bad Request` when fails, as well as a nicely formatted message. Decide by yourself, what validation makes most sense for `UserDTO`.