Exercise for OWASP-**A2** og **A5**

List of Worst and/or Broken Passwords

* **How many lists of Worst Passwords can you find via Google**
  + <https://www.forbes.com/sites/daveywinder/2019/12/14/ranked-the-worlds-100-worst-passwords/#1d5e086869b4>
* **How many lists of possible valid usernames/passwords can you find via Google**
  + <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Default-Credentials/default-passwords.csv>

Preventing bad passwords

* **Why is this not always as easy as it sounds? - which two “project requirements” often draws in two quite different directions?**
  + Brugervenlighed og sikkerhed. Jo mere sikker ens applikation er, jo mindre brugervenlig vil den blive.
* **Implement a simple control (feel free to use predefined packages) to verify passwords, up against a set of rules decided by you (length, required character, illegal words etc.)**

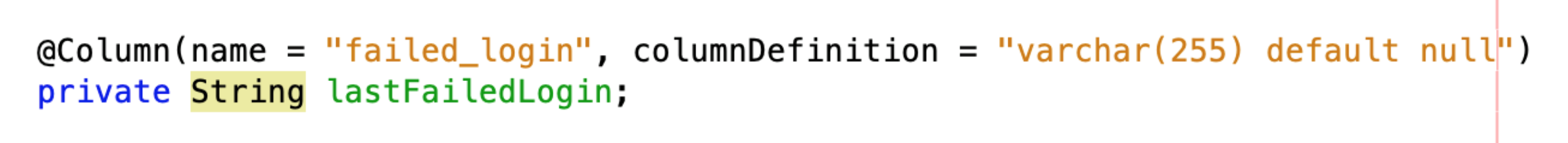
Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Prevent Brute Force Attacks

* **Consider ways to prevent brute force attacks and implement a VERY simple proof of concept to illustrate your ideas (in any language/platform you prefer)**
  + For at undgå brute force angreb, har vi valgt at implementere en metode, som låser en bruger ude i 30 sekunder, hvis der angives en forkert adgangskode til den tilhørende bruger.

I klassen User er denne nedenstående attribut tilføjet:

****

Disse funktioner bruges i klassen User, i forbindelse med den nye attribut:

**Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse**

Herefter bruges de ovenstående funktioner, i **verifyPassword** til at sætte et nyt timestamp på User, hvis det er under 30 sekunder siden de sidst fejlede i at logge ind eller hvis adgangskoden er skrevet forkert.

**Et billede, der indeholder skærmbillede, fugl

Automatisk genereret beskrivelse**

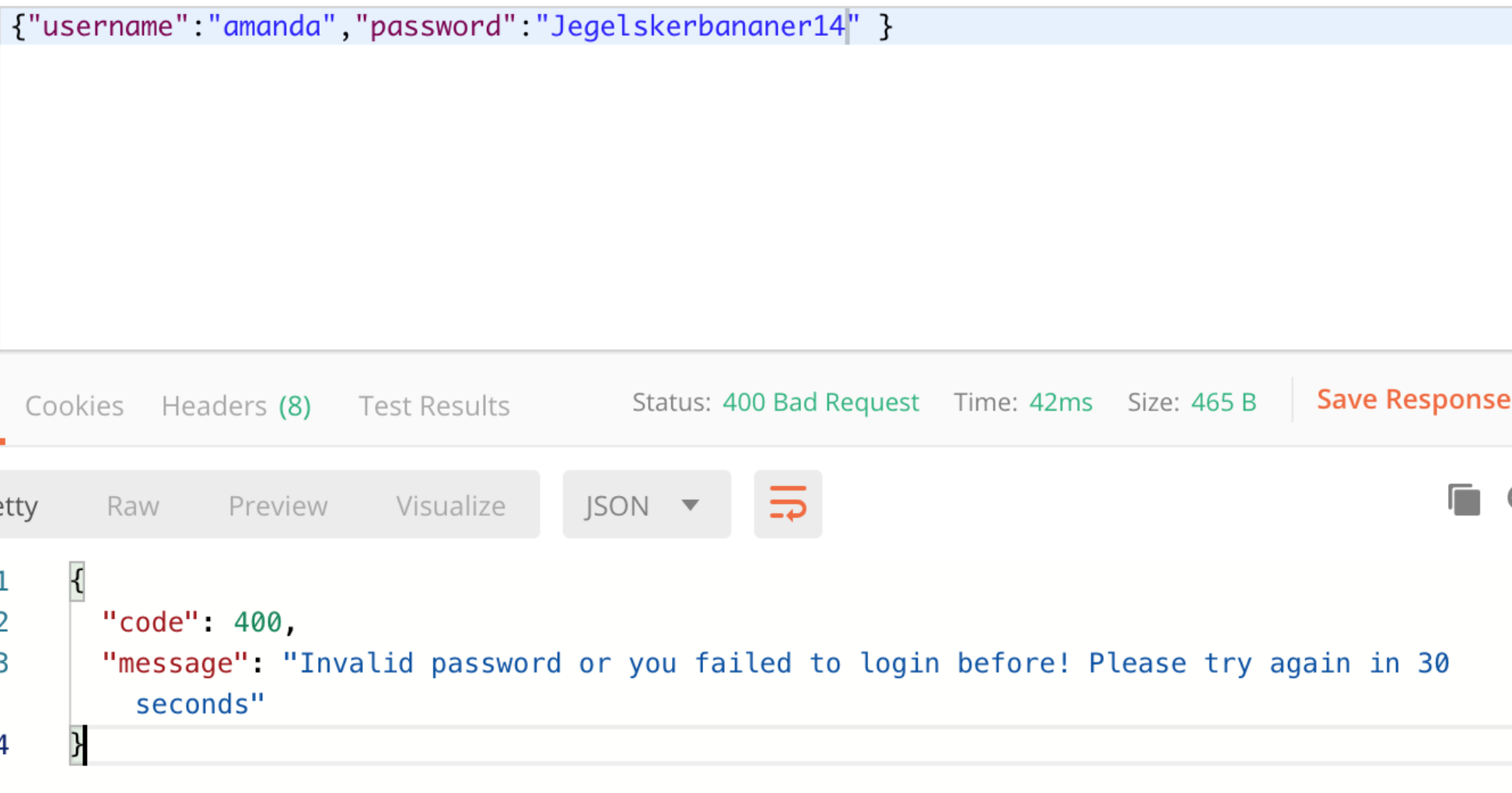
I userFacaden opdateres brugeren med sit nye timestamp i databasen, hvis verificering af brugeren fejler.

**Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse**

**Praktisk eksempel på at den ovenstående kode virker**

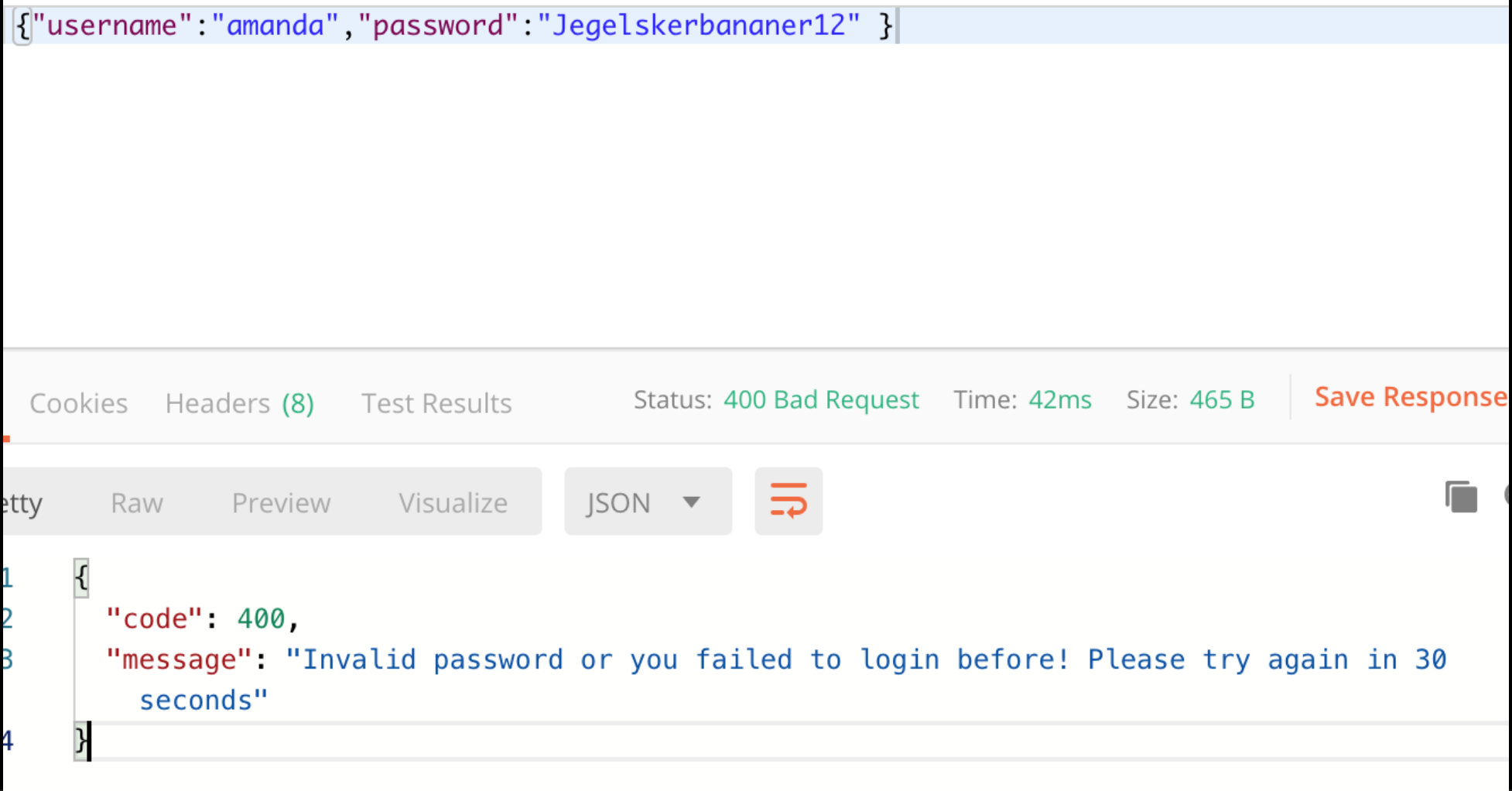
Vi indtaster en forkert adgangskode til en bruger, der findes i databasen. Vi får en kode 400 tilbage med en tilhørende besked.



I databasen ses det, at der kommer et timestamp (17:12:28), for **failedLogin**



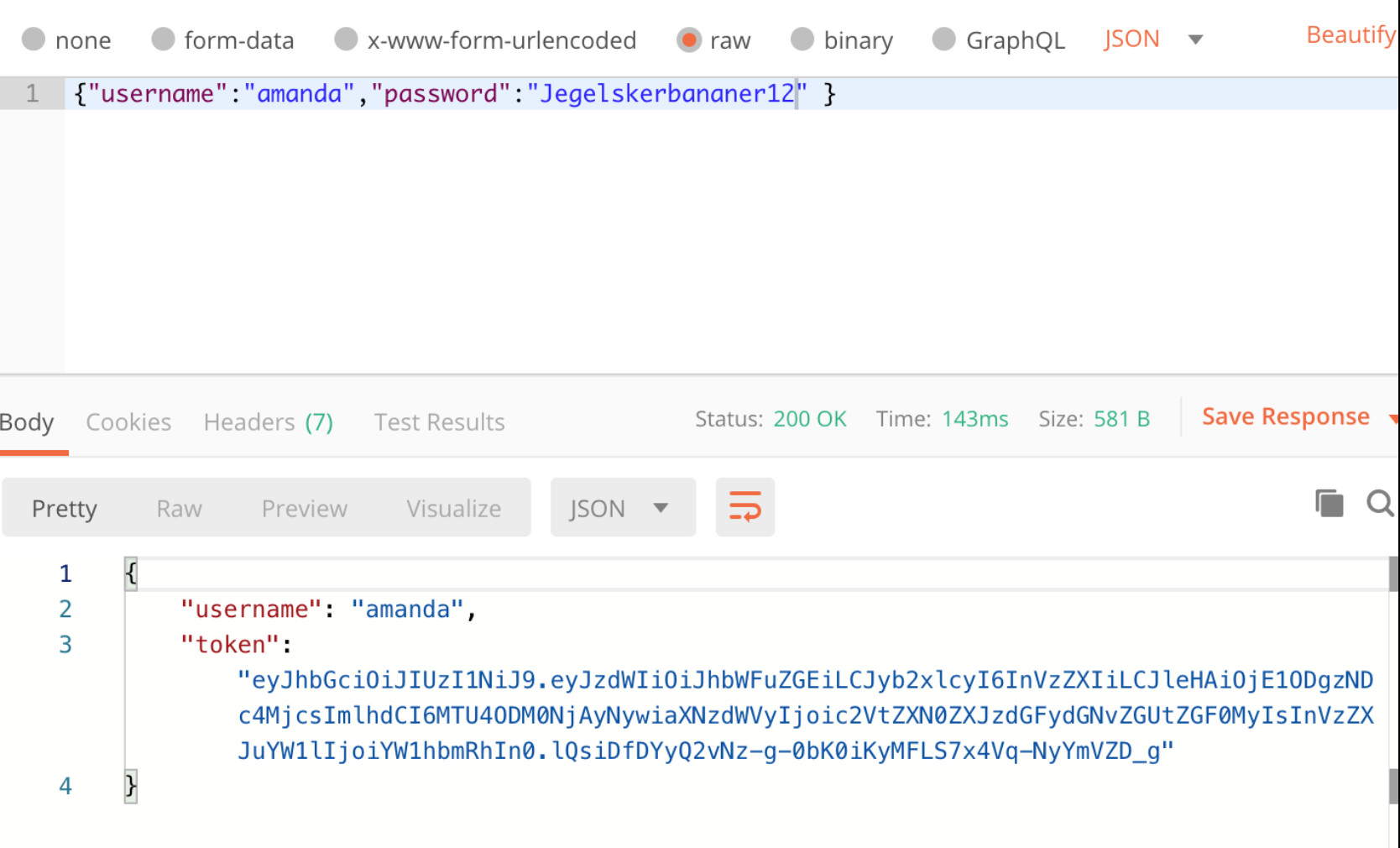
Herefter forsøges det at logge ind med den rigtige kode, men før de 30 sekunder er gået. Og det kan ses på nedenstående billede, at det stadig ikke er muligt at logge ind.



I databasen er der nu kommet et nyt timestamp (17:12:46) **failedLogin**



Når de 30 sekunder er gået, siden det sidste mislykkede forsøg, er det muligt at logge ind med den rigtige adgangskode, og der generes et token.



Broken Access Control - *Practical Hacking*

*The vulnerability you must exploit in this example is, unfortunately, even though you might find it simple once you have discovered the problem, very common and actually had its own number in the previous version of the OWASP list (*[*A4-Insecure Direct Object References*](https://www.owasp.org/index.php/Top_10_2013-A4-Insecure_Direct_Object_References)*). In the new (2017) version A4 and A7 from the 2013 version have been merged into A5, one of the topics for today's lesson, “Broken Access Control”. This vulnerability is one we have seen students “implement” again and again, most often in good faith because they have “protected” the resource.*

* **Brug dette link:** <http://dat-security.dk/a5demo/>

This website uses Basic HTTP Authentication via Java EE’s built-in security mechanisms. Using a built-in mechanism (from a trusted provider) is usually preferred, compared to inventing your own strategy.

A problem with Basic HTTP is that it is “impossible” to log out, so be prepared to close your browser while you are testing to log out or use an incognito window.

1. **Locate the problem in how Access Control is implemented and use this knowledge to “steal” other users’ private info.**

Start med at logge ind med brugeren: ”*a@b.dk*” og adgangskode: ”*test* ”

Herfra er det muligt at få a’s personlige data. Ved at undersøge siden, således:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

På ovenstående billede, kan man se værdien for brugerens e-mail, og ved at ændre denne til en anden e-mail, er det muligt at få en anden brugers personlige data (se nedenstående).

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

1. **Fix the problem**

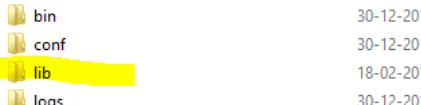
As mentioned earlier this project uses one of JavaEE/Tomcats built-in security mechanisms, more precisely a JDBCRealm (read [here](https://tomcat.apache.org/tomcat-8.0-doc/realm-howto.html) if you want more info), in combination with Tomcats built-in BASIC HTTP auth support. What it means is that your application (the one you will clone in the next step) does not know anything about specific users, that is left for Tomcat to handle. The application will only focus on Roles and what a user can do with a specific role (see the code for details). This requires a simple setup as outlined below:

**Følg nedenstående instruktioner:**

* Clone this project: <https://github.com/SecurityDatFall2018/a5demo.git>
* Create a local database called *a5demo* and run the script initalUserAndRoles in the scripts folder up against this database.
* Create a persistence.xml file with a persistence unit called **pu** (set create-strategy to NONE)
* Since Tomcat, and not you, will handle security we need to tell Tomcat which database contains users and the names of the relevant columns. This is done in the Realm-declaration in the file context.xml (in META-INF). If you have named your database a5demo, the only thing you need to do here is to provide the right username and password for the database
* Finally, again since Tomcat and NOT you will be handling most of the security, you need to copy the MySQL-database driver (*mysql-connector-java-xxxxxxx)*  into Tomcats lib-directory as explained below.
* *In NetBeans Services tab, locate your Tomcat Server, and the path for Catalina Home (see below).*

*Et billede, der indeholder skærmbillede, bord, sidder, sort

Automatisk genereret beskrivelse*

**

*In this folder, you will find a sub-folder* ***lib****. Copy the driver into this folder.* ***IMPORTANT:*** *When/if you deploy to DigitalOcean, you need to install the driver in Tomcat, here as well (in /usr/share/tomcat8/lib).*

**Hint, if you would like to deploy your solution:** Copy the driver up into the right folder on your Droplet

1) On your laptop, in the folder with the driver, type this (change parts in yellow):

scp mysql-connector-java-8.0.17.jar USER@IP:~/opt/tomcat/lib

2) Now the file should be in the root directory of your user (where you have write-access). Next in a shell on your droplet copy the file into Tomcats lib-dir.

sudo cp mysql-connector-java-8.0.17.jar /opt/tomcat/lib

3) Restart Tomcat

*The reason we did the copy as a "two-step operation" is that your non-root user does not have write access to /opt/tomcat/lib*

Deploy manually, using the *gui\_user* (assuming you are using the setup from last semester) and DO NOT push the version of context.xml and persistence.xml that includes your remote credentials.

3)      Change the default name (JSESSIONID) used for the Session Cookie

Out of the box, Tomcat, and JavaEE, in general, uses the string JSESSIONID for the session cookie. This discloses the underlying technologies and language used by the server, and OWASP recommends that we always change this value for a production server. Do that ([here](https://stackoverflow.com/questions/877064/changing-cookie-jsessionid-name) are a number of ways to do it).

Eksempel på

|  |  |  |  |
| --- | --- | --- | --- |
| id | warning | Bitcoin\_Address | email |
| 1 | To recover your lost Database and avoid leaking it: Send us 0.06 Bitcoin (BTC) to our Bitcoin address 1BLYhUDmnmVPVjcTWgc6gFT6DCYwbVieUD and contact us by Email with your Server IP or Domain name and a Proof of Payment. If you are unsure if we have your data, contact us and we will send you a proof. Your Database is downloaded and backed up on our servers. Backups that we have right now: securitydemo1, a2demo . If we dont receive your payment in the next 10 Days, we will make your database public or use them otherwise. | 1BLYhUDmnmVPVjcTWgc6gFT6DCYwbVieUD | contact@sqldb.to |