------ΑΣΦΑΛΕΙΑ ΔΙΚΤΥΩΝ-------Εργαστηριακή Άσκηση 2--------

Ονοματεπώνυμο: Ελένη Φουρτούνη

Αριθμός Μητρώου: 3180196

1) Δημιουργία βάσης και αρχικοποίησή της

Βήματα:

i) PostgreSQL installation and Configuration

a) Προσθήκη των γραμμών exclude=postgresql*

```
[root@snf-885687 ~]# sudo vi /etc/yum.repos.d/CentOS-Base.repo
#baseurl=http://mirror.centos.org/centos/$releasever/os/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7

exclude=postgresql*

#released updates
[updates]
name=CentOS-$releasever - Updates
mirrorlist=http://mirrorlist.centos.org/?release=$releasever&arch=$basearch&repo
=updates&infra=$infra
#baseurl=http://mirror.centos.org/centos/$releasever/updates/$basearch/
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
exclude=postgresql*
```

b)Start postgresql service:

sudo systemctl enable postgresql

sudo systemctl start postgresql

c)Conect to database server:

sudo -u postgres psql

CREATE DATABASE GDPR;

\c gdpr;

d) Create table users:

```
gdpr=# CREATE TABLE users(
   id     SERIAL PRIMARY KEY,
   username   VARCHAR(50) NOT NULL,
   password   VARCHAR(200) NOT NULL,
   description VARCHAR(250),
   UNIQUE(username)
);
NOTICE:  CREATE TABLE will create implicit sequence "users_id_seq"
 for serial column "users.id"
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "us
ers_pkey" for table "users"
NOTICE:  CREATE TABLE / UNIQUE will create implicit index "users_u
sername_key" for table "users"
CREATE TABLE
```

Επιλέχτηκε ένα serial id ως μοναδικό αναγνωριστικό κάθε χρήστη που αποθηκεύουμε στη βάση μας. Επίσης, θεωρήθηκε ορθό τα πεδία username password να μην λαμβάνουν την τιμή null, καθώς είναι απαραίτητα για την είσοδο των χρηστών στη βάση μας. Τέλος, θεωρήθηκε δεδομένο πως δεν υπάρχουν δύο εγγεγραμμένοι χρήστες με το ίδιο username. Ως αποτέλεσμα, τα περιχόμενα της βάσης GDRR(gdpr) έχουν την παρακάνω εικόνα:

```
gdpr=# \d
             List of relations
 Schema |     Name      |   Type   |  Owner
--------+---------------+----------+----------
 public | users         | table    | postgres
 public | users_id_seq  | sequence | postgres
(2 rows)
```

e) Insert requested values:

```
gdpr=# INSERT INTO users (username, password, description)
gdpr-# VALUES ('admin', 'atyjokjAdssUtg', 'system administrator');
INSERT 0 1
gdpr=# INSERT INTO users (username, password, description)
VALUES ('3180196', 'dnfkerNjTmbP', 'application user');
INSERT 0 1
```

```
gdpr=# SELECT * FROM users;
 id | username |    password     |     description
----+----------+-----------------+----------------------
  1 | admin    | atyjokjAdssUtg | system administrator
  2 | 3180196  | dnfkerNjTmbP   | application user
(2 rows)
```

2) Προκειμένου να διασφαλίσουμε την ιδιωτικότητα των χρηστών, σε περίπτωση παράνομης εξαγωγής δεδομένων από τη βάση, οφείλουμε να αποκωδικοποιήσουμε τον τρόπο

αποθήκευσης του κωδικού πρόσβασής τους στη βάση μας, έτσι ώστε πλέον να μην φαίνεται ευκρινώς το ποιος είναι, αλλά να απαιτείται η διαδικασία αποκρυπτογράφησης, με τη χρήση ασφαλώς κρυμμένου ιδιωτικού κλειδιού, έτσι ώστε να αποσπάσουμε τη δοθείσα τιμή του. Θέλουμε να χρησιμοποιήσουμε μία μέθοδο η οποία χρησιμοποιεί κάποιο salt, έτσι ώστε να παράγει διαφορετικό αποτέλεσμα κάθε φορά που καλείται με είσοδο τον ίδιο κωδικό πρόσβασης και διαφορετικό salt. Προκειμένου κάποιος επιτήδειος να ανακαλύψει τον κωδικό-πηγή από κάποιον κρυπτογραφημένο κωδικό, οφείλει να χρησιμοποιήσει τον κρυπτογραφημένο κωδικό όπως βρίσκεται αποθηκευμένος πλέον στη βάση μας και τον αρχικό μας κωδικό. Επίσης, η τιμή αυτή είναι καλό να παράγεται από τυχαίες γεννήτριες παραγωγής αριθμών και να αποτελείται από αρκετά ψηφία, για μεγαλύτερη ασφάλεια.
Βήματα:

a) Enable pgcrypto

```
gdpr=# CREATE EXTENSION pgcrypto;
CREATE EXTENSION
```

b) Update already entered passwords

```
gdpr=# UPDATE users
SET
    password = crypt('atyjokjAdssUtg', gen_salt('bf'))
WHERE
    username = 'admin';
UPDATE 1
```

```
gdpr=# UPDATE users
SET
    password = crypt('dnfkerNjTmbP', gen_salt('bf'))
WHERE
    username = '3180196';
UPDATE 1
```

c) Θα φτιάξουμε ένα Trigger Function, έτσι ώστε η διαδικασία αυτή να επαναλαμβάνεται αυτόματα, κάθε φορά που εισάγουμε στη βάση μας ένα νέο χρήστη. Έτσι, θα προστατέψουμε τα δεδομένα των χρηστών και τη βάση μας από επιθέσεις.

```
gdpr=# CREATE FUNCTION filterpasswords() RETURNS TRIGGER AS
$$
BEGIN
    NEW.password = crypt(NEW.password, gen_salt('bf'));
RETURN NEW;
 END;
$$
LANGUAGE 'plpgsql';
```

```
gdpr=# CREATE TRIGGER table_replace
gdpr-#                BEFORE INSERT
gdpr-# ON users
gdpr-# FOR EACH ROW
gdpr-# EXECUTE PROCEDURE filterpasswords();
```

```
gdpr=# CREATE TRIGGER table_replace_update
                BEFORE UPDATE
ON users
FOR EACH ROW
EXECUTE PROCEDURE filterpasswords();
```

Παράδειγμα εισαγωγής χρήστη:

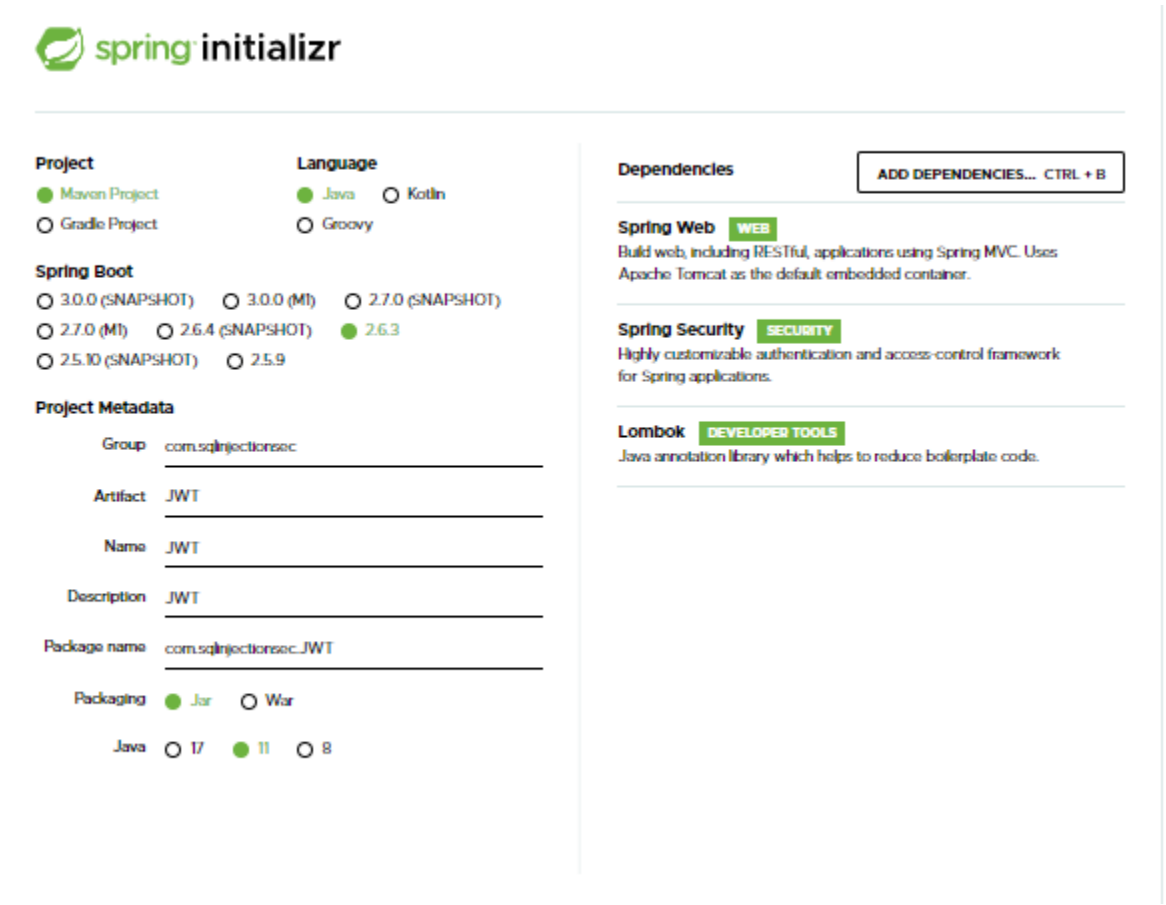
```
gdpr=# INSERT INTO users (username, password, description)
VALUES ('mbk', 'dnfkerNjTkllhjkbP', 'application user');
INSERT 0 1
gdpr=# SELECT * FROM users;
 id | username |                          password                          |     descrip
----+----------+----------------------------------------------------------------+------------
  1 | admin    | $2a$06$sPwthnCLkwcSAa/qfYBQgelkKiSF7FPvVmfo8a.GUq2yHWKGIN/uK | system admi
  2 | 3180196  | $2a$06$btK0DVfeHYTWvol2A9ARYO4fkWC7sOwWFdn6XkE4ZOFMCnSxqkPZ6 | application
 18 | mbk      | $2a$06$4MhSUTdqzXGtcenl5QDQLO/oXyqJf4yT8gY5t.xInYEqj9sefuGLa | application
(3 rows)
```

Τελευταίο βήμα: Διαγράφουμε το νέο χρήστη, καθώς ζητούμενο της εργασίας είναι να υπάρχουν στη βάση, μονο οι χρήστες: 3180196, admin

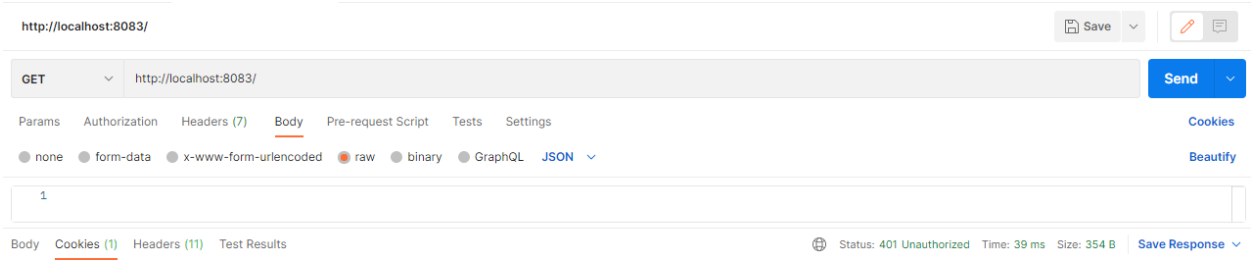
```
gdpr=# DELETE from users WHERE users.username='mbk';
DELETE 1
```

3) Χρησιμοποιήσαμε το framework Spring Boot(main dependencies: Spring WEB, Spring Security) με Java 11.

STEP 0:



Problem1:

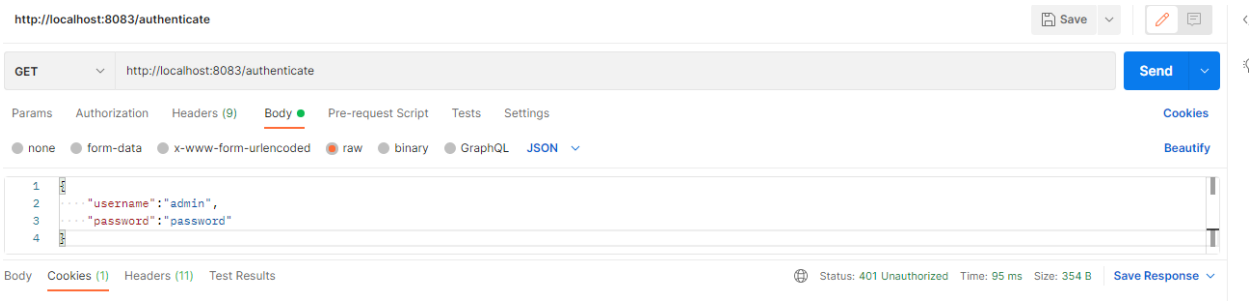


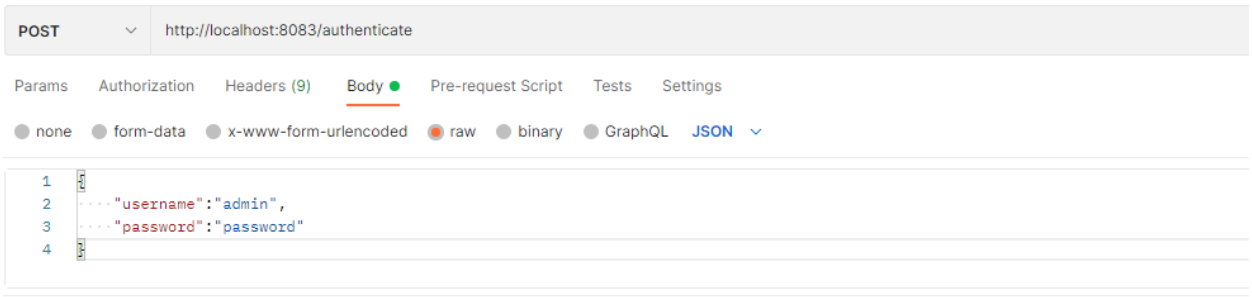
Solution1

SecurityConfiguration.java

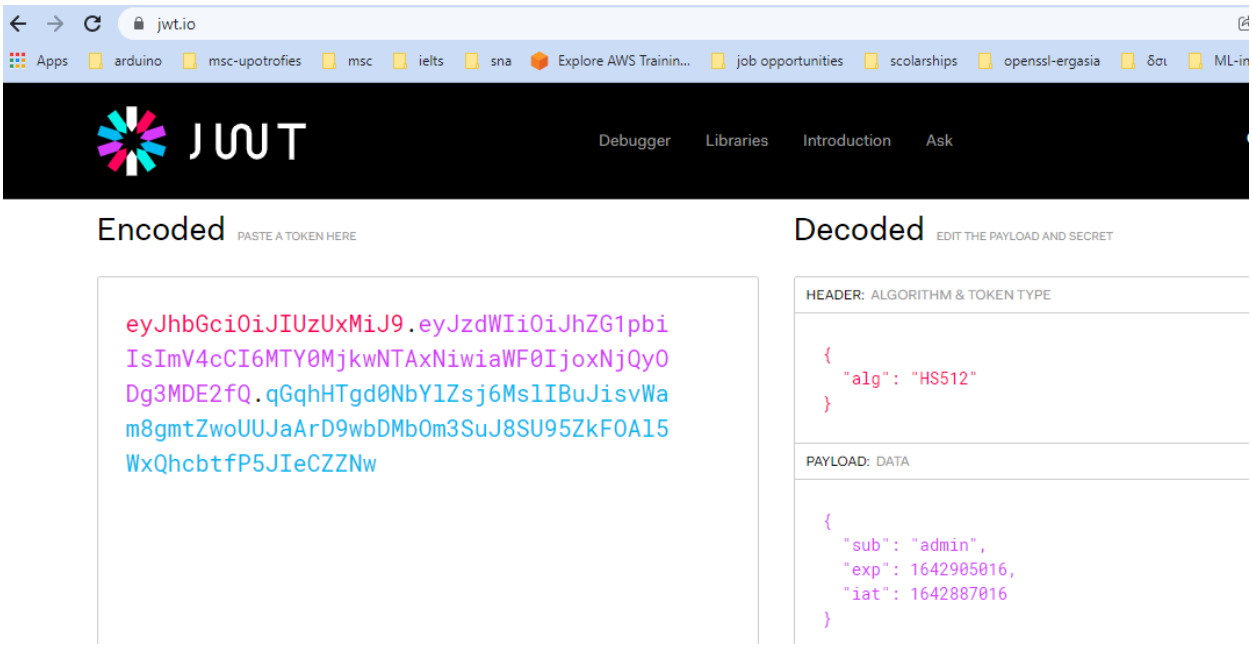
```
protected void configure(AuthenticationManagerBuilder auth)
    auth.userDetailsService(userService);
}

@Override
@Bean
public AuthenticationManager authenticationManagerBean() throws Ex
    return super.authenticationManagerBean();
}

@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf() CsrfConfigurer<HttpSecurity>
            .disable() HttpSecurity
            .authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.Express
            .antMatchers( ...antPatterns: "/authenticate") ExpressionUrlAuthor
            .permitAll() ExpressionUrlAuthorizationConfigurer<...>.ExpressionIntercept
            .anyRequest() ExpressionUrlAuthorizationConfigurer<...>.AuthorizedUrl
            .authenticated();
}
```

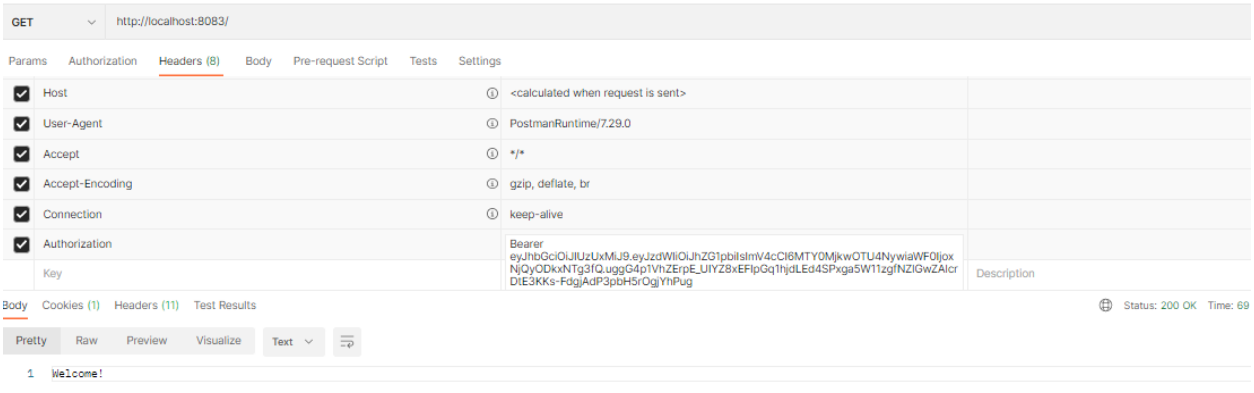
Problem 2:

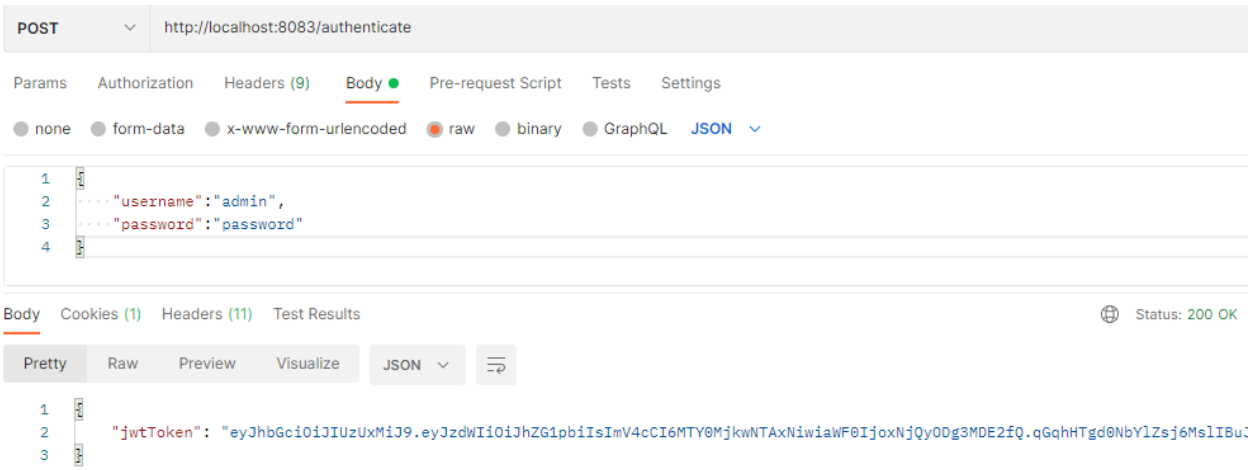



POST
http://localhost:8083/authenticate
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"username":"admin",
"password":"password"
}


Body
Cookies (1)
Headers (11)
Test Results
Status: 200 OK
Pretty
Raw
Preview
Visualize
JSON
{
"jwtToken": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImV4cCI6MTY0MjkwNTAxNiwiaWF0IjoxNjQyODg3MDE2fQ.qGqhHTgd0NbYlZsj6MslIBuJ
}


jwt.io
Apps
arduino
msc-upotrofies
msc
ielts
sna
Explore AWS Trainin...
job opportunities
scolarships
openssl-ergasia
δσι
JWT
Debugger
Libraries
Introduction
Ask
Encoded
PASTE A TOKEN HERE
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImV4cCI6MTY0MjkwNTAxNiwiaWF0IjoxNjQyODg3MDE2fQ.qGqhHTgd0NbYlZsj6MslIBuJisvWam8gmtZwoUUJaArD9wbDMbOm3SuJ8SU95ZkFOA15WxQhcbtfP5JIeCZZNw
Decoded
EDIT THE PAYLOAD AND SECRET
HEADER: ALGORITHM & TOKEN TYPE
{
"alg": "HS512"
}
PAYLOAD: DATA
{
"sub": "admin",
"exp": 1642905016,
"iat": 1642887016
}


GET
http://localhost:8083/
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Host
<calculated when request is sent>
User-Agent
PostmanRuntime/7.29.0
Accept
/
Accept-Encoding
gzip, deflate, br
Connection
keep-alive
Authorization
Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImV4cCI6MTY0MjkwOTU4NywiaWF0IjoxNjQyODkxNTg3fQ.uggG4p1VhZErpE_UIYZ8xEFIpGq1hjdLEd4SPxga5W11zgfNZIGwZAlcrDtE3KKs-FdgjAdP3pbH5rOgjYhPug
Key
Description
Body
Cookies (1)
Headers (11)
Test Results
Status: 200 OK
Time: 69
Pretty
Raw
Preview
Visualize
Text
Welcome!



4. α) Εισαγωγή πίνακα logging

```
gdpr=#  CREATE TABLE logging(
     logging_id SERIAL PRIMARY KEY,
     username VARCHAR(50),
     time TIMESTAMP,
     success BOOLEAN);
```

β) SQL Procedures: (Χρειάστηκε να κάνουμε αναβάθμιση σε Postgresql 11)

```
postgres=# CREATE PROCEDURE count_logging_attemps(un varchar(50),password varchar(200), inout success smallint)
postgres-#  LANGUAGE plpgsql
postgres-# AS $$
postgres$# DECLARE
postgres$# decoded_password varchar;
postgres$# encoded_password varchar;
postgres$#
postgres$# BEGIN
postgres$# SELECT COUNT(*) AS failed_attempts
postgres$# FROM logging
postgres$# WHERE username=un AND success=false;
postgres$# If failed_attempts<3 then
postgres$# SELECT password AS given_pass
postgres$# FROM users WHERE username = un;
postgres$# END IF;
postgres$# decoded_password:=crypt(password, given_pass);
postgres$# encoded_password:=crypt(password, gen_salt('bf'));
postgres$# IF decoded_password=encoded_password THEN
postgres$# set success=1;
postgres$# ELSE
postgres$# set success=0;
postgres$# END IF;
postgres$# INSERT INTO logging VALUES (un, now(), success);
postgres$# select success;
postgres$# END $$;
CREATE PROCEDURE
```

Στην παραπάνω διαδικασία, ελέγχουμε πόσες φορές θα επιτρέπεται σε έναν χρήστη να κάνει login attempt(3) μέχρι να τον "κλειδώσουμε".

```
postgres=# CREATE PROCEDURE expired_certifications(un varchar, date timestamp, inout expired smallint)
LANGUAGE plpgsql
AS $$
        BEGIN
SELECT time AS most_recent FROM logging  WHERE logging.username=un ORDER BY time DESC LIMIT 1;
SELECT time AS second_most_recent FROM (
SELECT time FROM logging WHERE logging.username=un
ORDER BY time DESC LIMIT 2
) AS FOO WHERE logging.username=un ORDER BY time ASC LIMIT 1;
If(most_recent - most_recent > '40 days 00:00:01') THEN
SET expired=1;
ELSE
SET expired=0;
END IF;
SELECT expired;
        END $$;
CREATE PROCEDURE
```

Στην παραπάνω διαδικασία, ελέγχουμε αν έχουν περάσει 40 ημέρες από την τελευταία φορά που αλλάξαμε τον κωδικό πρόσβασής μας, έτσι ώστε αυτός να χαρακτηριστεί ως ληγμένος(expired).