

#### 4<sup>η</sup> εργαστηριακή άσκηση ασφάλεια δικτύων

|                     |         |
|---------------------|---------|
| Δικαία Σωτηροπούλου | 3160172 |
| Μαρία Ελένη Κοκκίνη | 3170070 |

- Προκειμένου να προστατεύσουμε τους κωδικούς στη βάση σε περίπτωση παράνομης εξαγωγής δεδομένων αποφύγαμε να κρατάμε τους κωδικούς σε απλή text μορφή. Χρησιμοποιήσαμε το default module της PostgreSQL, το pgcrypto, για κρυπτογράφηση των κωδικών. Η συνάρτηση που χρησιμοποιεί το module παίρνει δύο παραμέτρους: το text που θέλουμε να κρυπτογραφηθεί και ένα salt (τυχαία τιμή) 'crypt('mypassword', gen\_salt('bf'))'. Για την παραγωγή του τυχαίου αριθμού salt χρησιμοποιήθηκε ο αλγόριθμος Blowfish (παράμετρος 'bf'), θα μπορούσαν να έχουν χρησιμοποιηθεί επίσης άλλοι αλγόριθμοι όπως ο md5, des. Έτσι όταν κάποιος βλέπει τους κωδικούς αντί να βλέπει τον πραγματικό κωδικό θα βλέπει κάποια κρυπτογραφημένη μορφή:

```
GDPR=# SELECT * FROM users;
      username      |      password
-----+-----
admin              | $2a$06$Cir.DDUm9x.mgGLp..18h.οRO0Q/SCZ.t0Ij7ibGqWJPgBCI1GN0q
| this user is the admin
3160172-3170070    | $2a$06$ikrzRR3DtAcib9T2s5tEUuitlYKWFcomxJUy8tDLu7t.BneGGlk62
| these are our school numbers
(2 rows)
```

Όταν ο χρήστης θα εισάγει τον κωδικό του θα χρησιμοποιηθεί η συνάρτηση crypt() ξανά, ώστε να γίνει η σύγκριση μεταξύ των κρυπτοκειμένων.

Εντολές που χρησιμοποιήθηκαν για τη δημιουργία της βάσης και τον μετασχηματισμό του password: (Τα bold αφορούν τον μετασχηματισμό του κωδικού)

1. Κατασκευή πίνακα → CREATE TABLE users( username varchar (50) PRIMARY KEY NOT NULL, password text NOT NULL, description text);
  2. Εγκατάσταση του crypto extension → **create extension pgcrypto;**
  3. Εισαγωγή στοιχείων admin → INSERT INTO users (username, password, description) VALUES ('admin', **crypt('chicken', gen\_salt('bf'))**), 'this user is the admin');
  4. Εισαγωγή στοιχείων 3160172-3170070 → INSERT INTO users (username, password, description) VALUES ('3160172-3170070', **crypt('kotopoulos', gen\_salt('bf'))**), 'these are our school numbers');
- Έλεγχος για σωστό login: (χρησιμοποιήθηκε ο browser elinks)  
Ο χρήστης δίνει ένα username και password και χρησιμοποιούμε queries σε παραμετρική μορφή χρησιμοποιώντας "?" και τη συνάρτηση set, με αποτέλεσμα οι τιμές

να μετατρέπονται σε strings εμποδίζοντας την πραγματοποίηση sql injection (δοκιμάστηκε). → πχ stmt = connection.prepareStatement(query\_mod\_pwd); stmt.setTimestamp(1, sqlTimestamp);

Δίνοντας τα σωστά credentials προκύπτει → Boolean is: true σε κάθε άλλη περίπτωση βγάζει boolean is: false. Ακολουθεί παράδειγμα (έχει ληφθεί υπόψιν η αποφυγή sql injection επίθεσης)

```
2021-06-07 03:02:00.270 INFO 4452 [INFO-0001 EXEC-1] o.s.web.servlet.DispatcherServlet
User from UI = User [name=admin, password=chicken]
before connection
connection ok!
name is : admin pwd is : chicken
Success!
query is: SELECT * FROM users WHERE username=? AND password = crypt(?, password)
boolean is: true
User from UI = User [name=3160172_3170070, password=kotopoulo]
before connection
connection ok!
name is : 3160172_3170070 pwd is : kotopoulo
boolean is: false
User from UI = User [name=3160172-3170070, password=kotopoulo]
before connection
connection ok!
name is : 3160172-3170070 pwd is : kotopoulo
Success!
query is: SELECT * FROM users WHERE username=? AND password = crypt(?, password)
boolean is: true
```

- Για το τρίτο ερώτημα αρχικά ορίσαμε τον μέγιστο αριθμό αποτυχημένων προσπαθειών (5) και κατασκευάσαμε μέσω της Java τον πίνακα logging στη βάση GDPR. Αυτά συμβαίνουν κατά τη σύνδεση του χρήστη στο web application:

```
final int MAXIMUM_LOGIN_ATTEMPTS = 3;

@RequestMapping("/")
public String index(){
    //create table for login attempts --> we need only one
    //db info
    String host = "localhost";
    String port = "5432";
    String db_name = "GDPR";
    String username = "GDPR";
    String password = "";
    Connection connection = null;
    Statement statement = null;
    boolean table_exist = false;
    try{

        Class.forName("org.postgresql.Driver");
        connection = DriverManager.getConnection("jdbc:postgresql://" + host + ":" + port + "/" + db_name + "", "" + username + "", "" + password);
        if(connection != null){
            System.out.println("connection ok!");
            //check if table exists
            DatabaseMetaData databaseMetaData = connection.getMetaData();
            ResultSet resultSet = databaseMetaData.getTables(null, "logging", null, new String[] {"TABLE"});
            if(resultSet.next()){
                table_exist = true;
                System.out.println("table exists: " + table_exist);
            } else{
                //create table
                String query = "CREATE TABLE IF NOT EXISTS logging(username varchar (50) PRIMARY KEY NOT NULL, attempts int, timestamp TIMESTAMP)";
                statement = connection.createStatement();
                statement.executeUpdate(query);
                System.out.println("finished");
            }
        }
    }
}
```

```

    }else{
        System.out.println("connection failed");
    }
}catch(Exception e){
    e.printStackTrace();
}finally{
    try{
        statement.close();
        connection.close();
    }catch(Exception e){
        e.printStackTrace();
    }
}
return "addUser";
}
}

```

- Για να μπορούμε να ελέγχουμε πότε άλλαξε τελευταία φορά ο κωδικός, χρειάστηκε να κάνουμε Update τον πίνακα των users προσθέτοντας μια στήλη που ελέγχει πότε τροποποιήθηκε τελευταία φορά ο κωδικός. Θέσαμε αρχικά μια default τιμή, η οποία θα αλλάζει το πάτημα του κουμπιού change password

```

GDPR=# UPDATE users SET last_modified='2021-06-08 10:28:14.581';
UPDATE 2
GDPR=# SELECT * FROM users;

```

| username        | password  | description                  | last_modified           |
|-----------------|---|------------------------------|-------------------------|
| admin           | \$2a\$06\$Cir.DDum9x.mgGLp..18h.oRO0Q/SCZ.t0Ij7ibGqWJPgBCIIGN0q | this user is the admin       | 2021-06-08 10:28:14.581 |
| 3160172-3170070 | \$2a\$06\$ikrzRR3DtAcib9T2s5tEUuitlYKWFcomxJUy8tDLu7t.BneGGlk62 | these are our school numbers | 2021-06-08 10:28:14.581 |

(2 rows)

Στη συνέχεια ορίσαμε διάστημα 3 μηνών, το οποίο εάν περάσει θα αναγκάζεται ο χρήστης να αλλάξει κωδικό πρόσβασης. Σε αυτή την οθόνη προβαίνουμε μόνο εάν το login ήταν επιτυχές. Αυτό το ρυθμίσαμε στον consoler:

```

ResultSet rs = stmt.executeQuery();
if (rs.next()) {
    // Login Successful if match is found
    success = true;
    System.out.println("Success!");

    //check if you need to change pwd
    //compare timestamps
    String get_timestamp_query = "SELECT last_modified FROM users WHERE username=?";
    stmt_timestamp = connection.prepareStatement(get_timestamp_query);
    stmt_timestamp.setString(1, name);
    ResultSet rs_timestamp = stmt_timestamp.executeQuery();
    while(rs_timestamp.next()){
        Timestamp prev_timestamp = rs_timestamp.getTimestamp("last_modified");
        System.out.println("last modified at: "+prev_timestamp);
        long now = System.currentTimeMillis();
        Timestamp currentTimeStamp = new Timestamp(now);

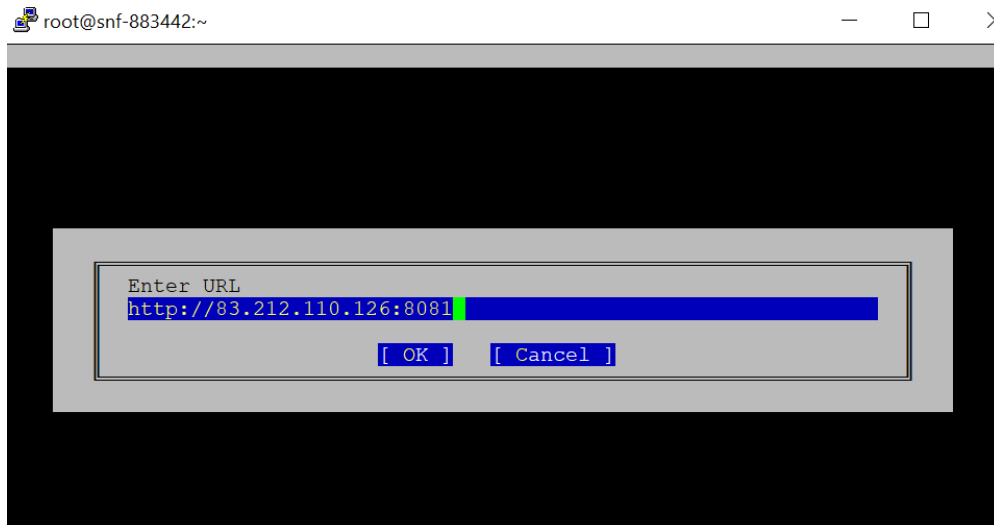
        long diff = getDateDiff(prev_timestamp, currentTimeStamp, TimeUnit.DAYS);
        System.out.println("difference in days is: "+diff);
        if(diff>=DAYS_TO_CHANGE_PWD){
            System.out.println("pwd must be changed");
            //msgChangePwd
            return modelAndView4;
        }
    }
}
}

```

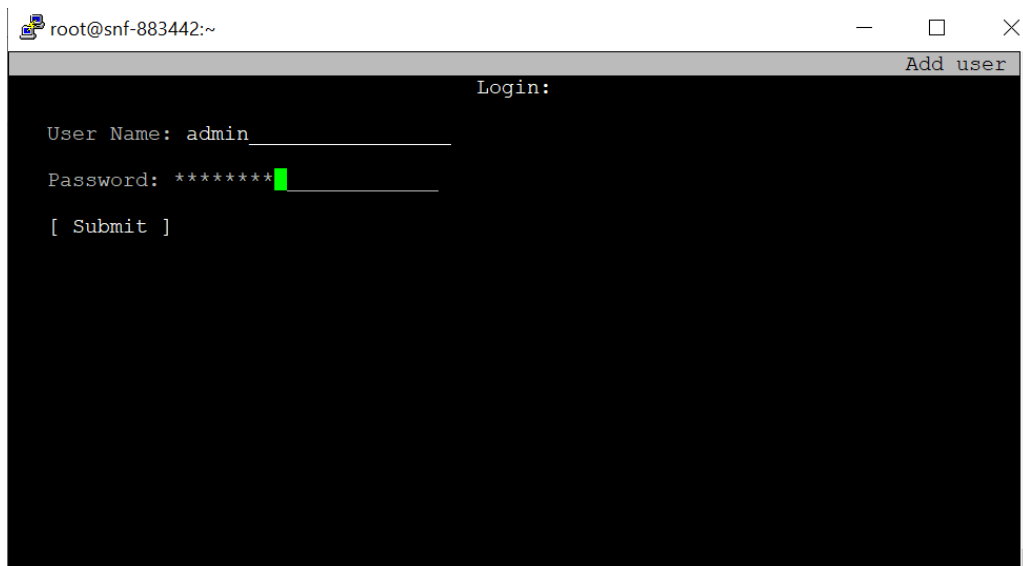
Αν δηλαδή το τρέχον Timestamp ξεπερνάει αυτό που βρίσκεται στη βάση κατά 90 ημέρες, τότε πρέπει να αλλαχθεί ο κωδικός.

Παρακάτω εμφανίζουμε τα use cases:

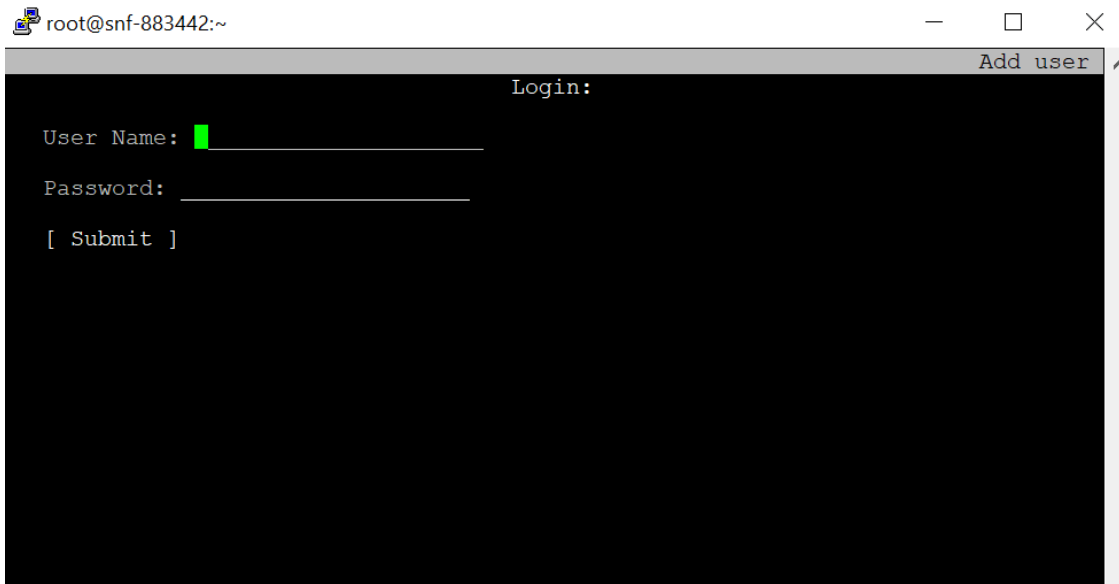
Αρχικά για κάθε περίπτωση απαιτείται η σύνδεση στον διακομιστή (μέσω elinks)



Επίσης σε κάθε περίπτωση δίνουμε username, password:



1. Αποτυχημένο login (φορά <5): ξανά εμφανίζεται η αρχική σελίδα  
Αποτέλεσμα που βλέπει ο χρήστης:

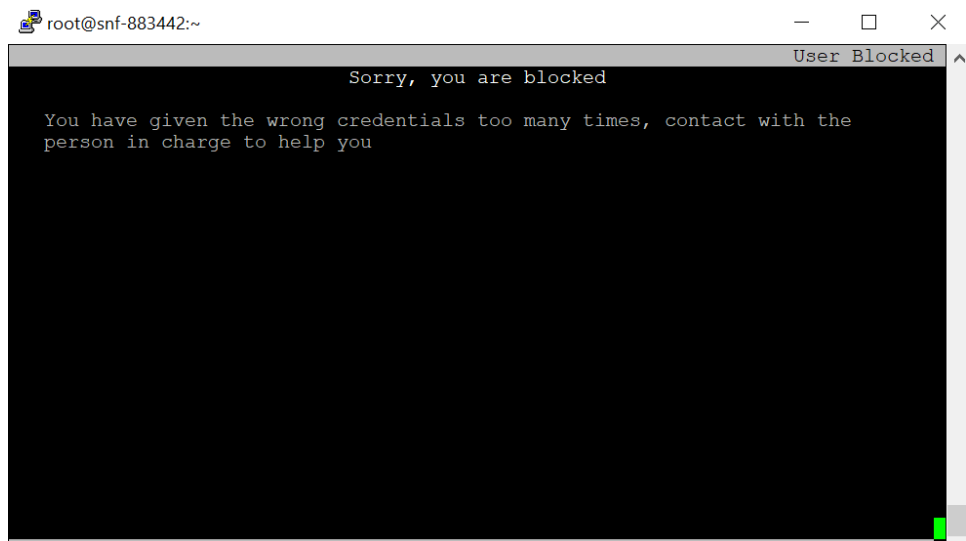


Αυτό που φαίνεται στον σέρβερ:

```
User from UI = User [name=admin, password=wrongpwd]
before connection
connection ok!
name is : admin pwd is : wrongpwd
insert row
Number of admin records in the table: 2
```

Το οποίο δείχνει τον λάθος κωδικό που δόθηκε για τον admin (wrongpwd αντί για chicken). Το Insert row δείχνει ότι προστέθηκε γραμμή στον πίνακα logging με username admin. Το number of admin records είναι 2, γιατί είχαμε ξαναδώσει λάθος κωδικό. Στις 5 λάθος προσπάθειες ο χρήστης κλειδώνεται.

## 2. Αποτυχημένο login (φορά >=5): ο χρήστης κλειδώνεται

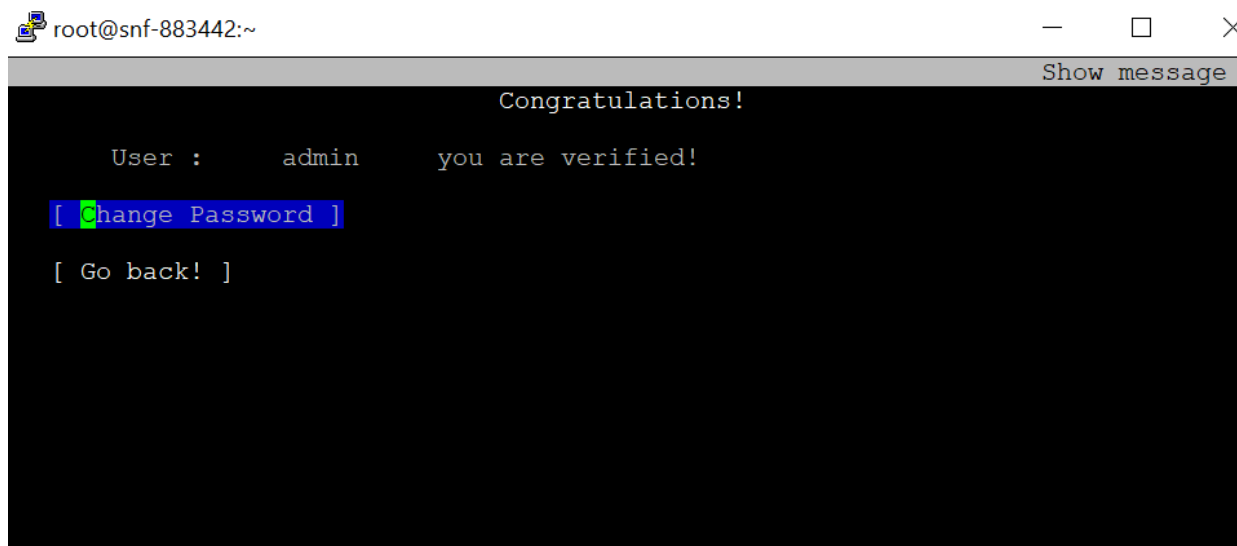


Σε αυτή την περίπτωση ο χρήστης δε μπορεί να κάνει κάτι για να αποτρέψει το συμβάν, θεωρητικά τον προτρέπουμε να επικοινωνήσει με τον υπεύθυνο της ιστοσελίδας, ο οποίος αφού προβεί σε επιπλέον ταυτοποιήσεις θα δεχτεί να ξαναδώσει πρόσβαση στον χρήστη ή όχι.

Από την πλευρά του σέρβερ ως διαχειριστές βλέπουμε ότι ο χρήστης κλειδώθηκε.

```
name is : marilena pwd is : 12
insert row
Number of marilena records in the table: 6
```

### 3. Επιτυχημένο Login (ο κωδικός δε χρειάζεται ακόμη αλλαγή):



Εμφανίζεται το όνομα του χρήστη και επιβεβαιώνουμε ότι απέκτησε πρόσβαση στην σελίδα μας. Επιπλέον υπάρχουν τα κουμπιά Change password και go back. Το 1<sup>ο</sup> δεν έχει όντως λειτουργικότητα αλλαγής κωδικού αλλά δίνει στον σερβερ οδηγία να κρατήσει το παρόν timestamp και να ενημερώσει τον πίνακα users με αυτό. Το 2<sup>ο</sup> απλά γυρίζει στην πίσω σελίδα.


Από πλευράς σέρβερ βλέπουμε τα εξής:

```
name is : admin pwd is : chicken
Success!
last modified at: 2021-06-08 12:57:12.314
difference in days is: 0
```

Οι πληροφορίες που λαμβάνουμε είναι ότι ο χρήστης συνδέθηκε επιτυχώς, βλέπουμε πότε τροποποιήθηκε ο κωδικός τελευταία φορά και βλέπουμε επίσης πόσες μέρες πέρασαν από την τελευταία τροποποίηση του κωδικού. Οι πληροφορίες αυτές δεν είναι διαθέσιμες στον client.

- Εάν πατηθεί το κουμπί αλλαγής κωδικού:

Ενημερώνεται ο χρήστης για αυτή την εικονική αλλαγή

 root@snf-883442:~

```

                                     Password changed!

you will be notified when you must change it again

[ Go back! ]
```


Και από την πλευρά του σέρβερ βλέπουμε τα εξής:

```
current time: 2021-06-08 13:37:37.041
login name: admin
```

Που μας ενημερώνουν πότε συνέβη η τελευταία τροποποίηση και δηλώνεται επίσης και το όνομα ου χρήστη, ο οποίος άλλαξε τον κωδικό του.

#### 4. Επιτυχημένο Login (ο κωδικός χρειάζεται αλλαγή):

Η διαδικασία σύνδεσης γίνεται με τον ίδιο τρόπο, όπως δείξαμε παραπάνω

 root@snf-883442:~

```

Change password
Last password expired! You need to change your password

Change password
```

Με το που συνδεθεί ο χρήστης τυπώνεται το παραπάνω μήνυμα στην οθόνη του. Το κουμπί Change password δεν έχει λειτουργικότητα άλλα εάν είχε θα έπρεπε να αλλαχθεί ο κωδικός και να κρατηθεί το αντίστοιχο Timestamp όπως και παραπάνω.

Η λειτουργικότητα που θέσαμε από πίσω δεν αφορά προφανώς τους 3 μήνες, αλλά θέσαμε δοκιμαστικά την μία ώρα για ελάχιστο διάστημα αλλαγής κωδικού

προκειμένου να δούμε αν λειτουργεί. Επίτηδες δεν υπάρχει κουμπί επιστροφής, έτσι ώστε αν δεν αλλάξει ο χρήστης κωδικό να μη μπορεί να κάνει κάτι άλλο.

```
long diff = getDateDiff(prev_timestamp, currentTimestamp, TimeUnit.HOURS);
//System.out.println("difference in days is: "+diff);
System.out.println("difference in hours is: "+diff);
//if(diff>=DAYS_TO_CHANGE_PWD){
if(diff>=1){
    System.out.println("pwd must be changed");
    //msgChangePwd
    return modelAndView4;
}
```

Από την πλευρά του σέρβερ βλέπουμε τα εξής:

```
name is : 3160172-3170070 pwd is : kotopoulos
Success!
last modified at: 2021-06-08 10:28:14.581
difference in hours is: 3
pwd must be changed
```

Δηλαδή βλέπουμε ότι η ταυτοποίηση έγινε σωστά και ότι ξεπεράστηκε η μία ώρα οπότε πρέπει να αλλαχθεί και ο κωδικός.

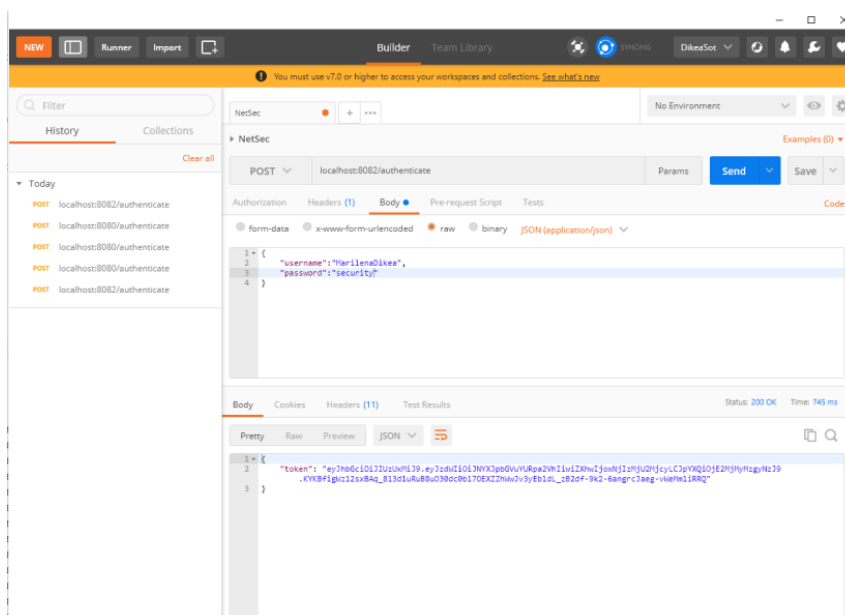
---

Ο κώδικας βρίσκεται στον φάκελο Bonus, κάτω από το root. Για να τρέξει το project, ο server ακούει στο Port 8082

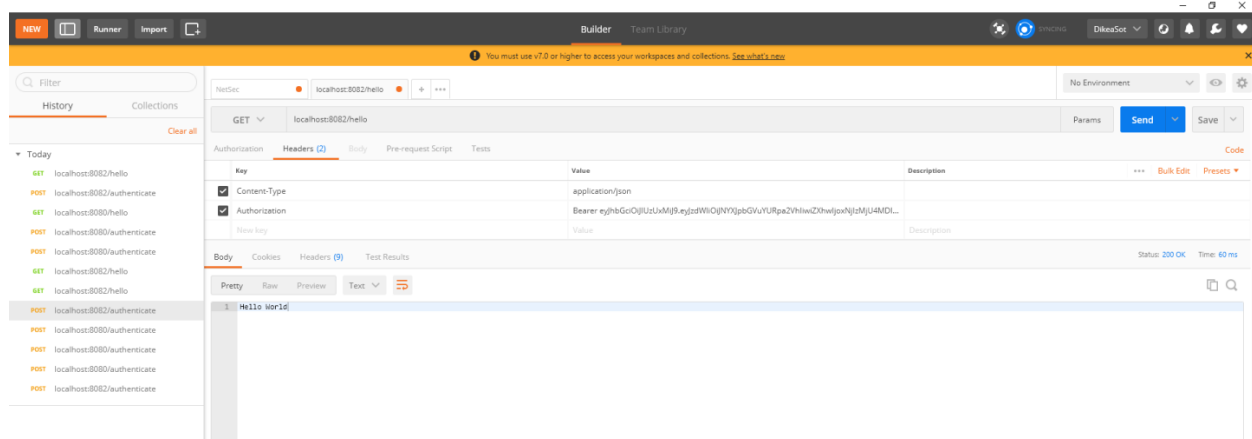
Προσθήκες για το bonus:

- 1) Προσθήκη στα dependencies στο pom.xml → {<dependency>  
<groupId>io.jsonwebtoken</groupId>  
<artifactId>jjwt</artifactId>  
<version>0.9.1</version>  
</dependency>  
<dependency>  
<groupId>javax.xml.bind</groupId>  
<artifactId>jaxb-api</artifactId>  
<version>2.3.0</version>  
</dependency>}
- 2) Προσθήκη end point hello world στον controller → {  
@RequestMapping("/{hello"})  
public String hello(){  
 return "Hello world!";  
}}





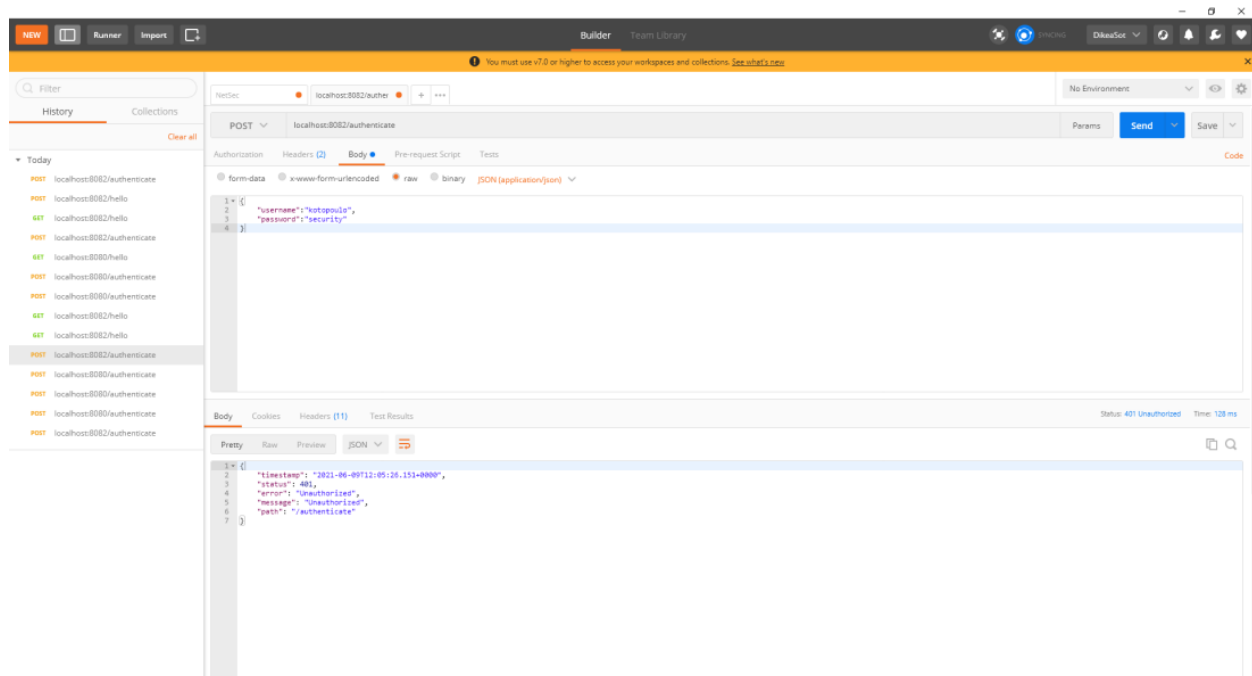
Στη συνέχεια χτυπάμε το hello world και παίρνουμε το αποτέλεσμα που αναμένουμε



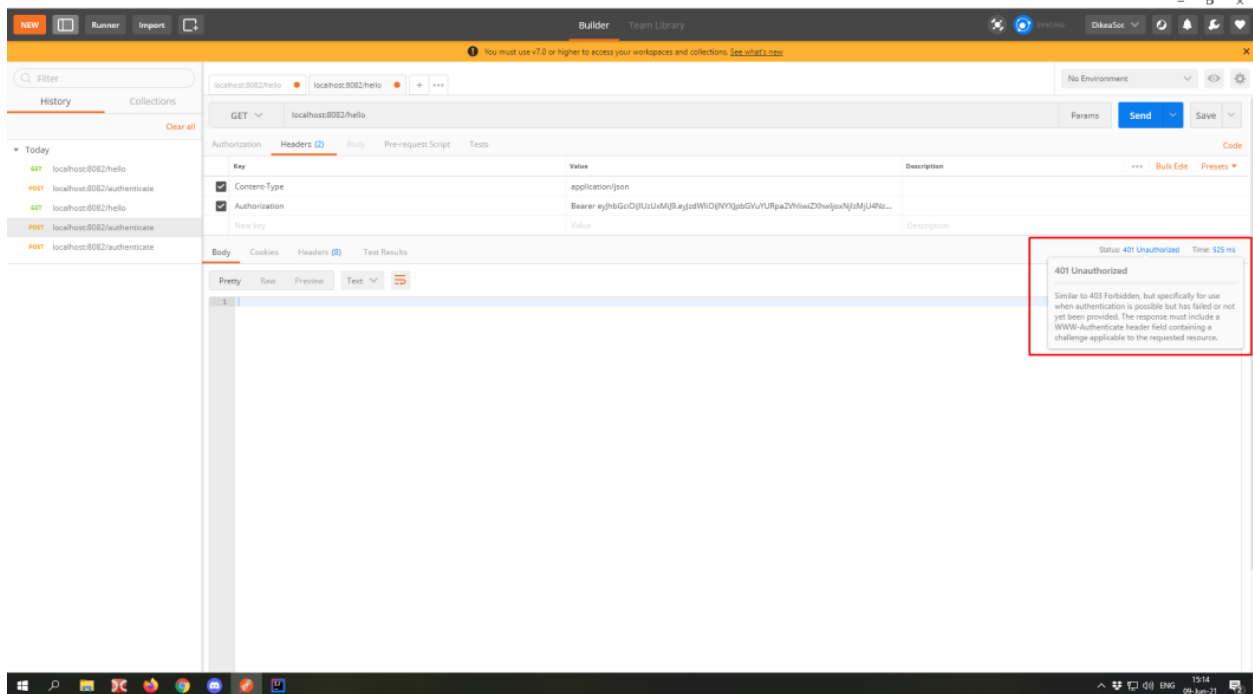
(Τα παραπάνω αρχεία μεταφέρθηκαν στον server μας στο μηχάνημα centos)

2<sup>η</sup> περίπτωση: δόθηκαν λάθος credentials:

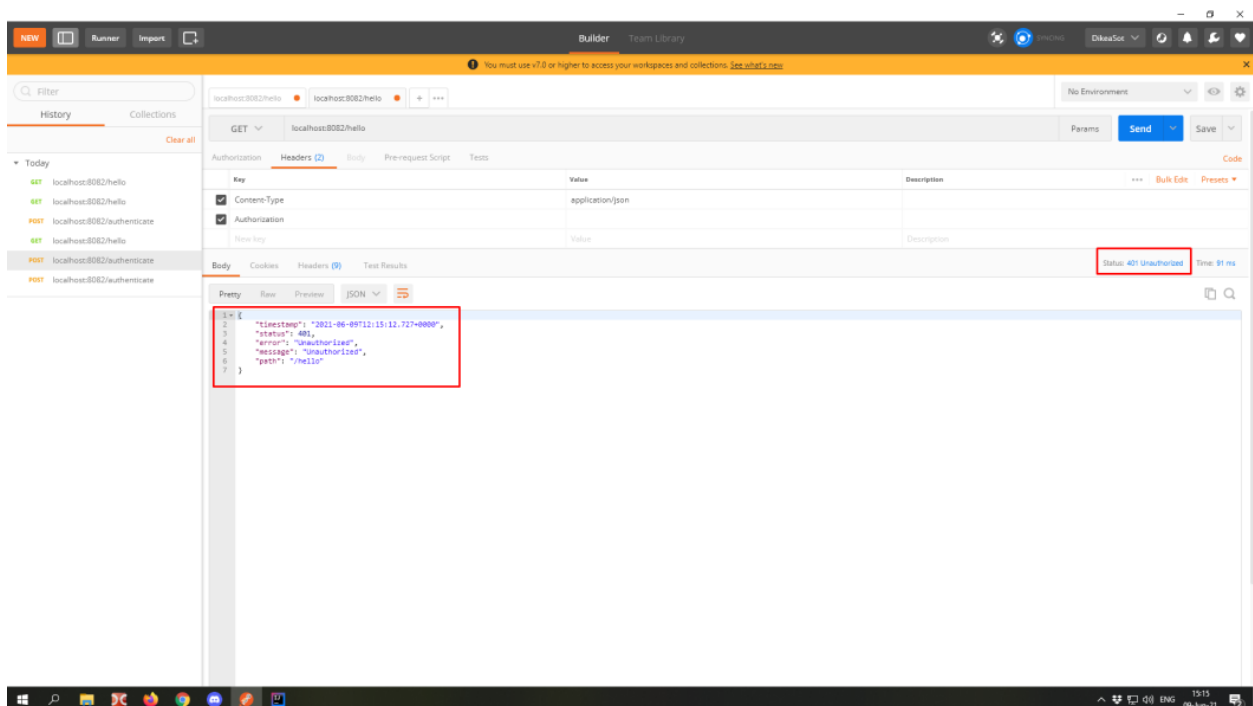
(Αντί για username MarilenaDikea δόθηκε username kotopoulos)



3<sup>η</sup> περίπτωση: έληξε το token:



3η περίπτωση: ο χρήστη προσπάθησε να συνδεθεί χωρίς token:



Δοκιμή τοπικά με τους καταγεγραμμένους χρήστες της βάσης (οι χρήστες ανακτούνται από τη βάση δυναμικά)

Βλέπουμε την ορθή παραγωγή του token για τον admin:

```
[root@snf-883442 ~]# curl -X POST -H "Content-Type: application/json" -d '{"username": "admin", "password": "chicken"}' http://83.212.110.126:8082/authenticate
{"token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pblliImV4cCI6MTYyMzZlOSwiaWF0IjoxNjIzMzE5NTU5fQ.qk4sgttaOeLGFCWMN6bR4IHMMETZZ3f-LMyGzHHxmi7rXLXLSW_lAi0bykp4bxs3n2Ev1VLYlqpdM6pcizlkRg"} [root@snf-883442 ~]#
```

Οπότε πηγαίνοντας τώρα στο hello μπορεί και παίρνει το μήνυμα hello word

```
[root@snf-883442 ~]# curl -i -H "Content-Type: application/json" -H "Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pblliImV4cCI6MTYyMzZlOSwiaWF0IjoxNjIzMzE5NTU5fQ.qk4sgttaOeLGFCWMN6bR4IHMMETZZ3f-LMyGzHHxmi7rXLXLSW_lAi0bykp4bxs3n2Ev1VLYlqpdM6pcizlkRg" -X GET http://83.212.110.126:8082/hello
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: text/plain;charset=UTF-8
Content-Length: 11
Date: Thu, 10 Jun 2021 10:19:44 GMT
Hello World [root@snf-883442 ~]#
```

Οι υπόλοιπες περιπτώσεις είναι αντίστοιχες με αυτές του hardcoded χρήστη.