HTTPS

Koraci za implementiranje https-a na front-u:

1) Napraviti folder “https” i u njemu skladistiti sve dokumente vezane za https
2) Pozicionirati se u “https” folder i u administratorskom rezimu otvoriti Git Bash
3) Pratiti korake sa sajta: https://blog.didierstevens.com/2008/12/30/howto-make-your-own-cert-with-openssl/:
    a. Ispred svake openssl komande pisati “winpty”
    b. Napraviti rootCA.key i rootCA.crt – root sertifikat kojim ce biti potpisani admin i hospital sertifikati(mora da postoji neki root sertifikat jer ukoliko browser vidi sertifikat koji je self-signed od neke nepoznate kompanije on ce tu vezu smatrati nebezbednom)
    c. Napomena: OBAVEZNO U COMMON NAME DA STOJI LOCALHOST – u common name mora da stoji domen za koji se kreira sertifikat(npr ukoliko kreiramo sertifikat za neki sajt u common name mora da se nalazi url do tog sajta)
    d. Napomena: Prilikom izvrsavanja **openssl x509 -req -days 730 -in ia.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out ia.crt OBAVEZNO dodati** `-sha256 -extfile v3.ext` https://stackoverflow.com/questions/43665243/invalid-self-signed-ssl-cert-subject-alternative-name-missing - v3.ext fajl smo prekopirali sa tog sajta

       Prethodno smo izgenerisali sertifikate bez te komande i rasporedili ih na odgovarajuca mesta i sve ostalo ali smo nakon podizanja fronta uporno dobijali gresku: net::ERR_CERT_COMMON_NAME_INVALID

    e. Napraviti admin.key, admin.crt i admin.p12 za adminsku aplikaciju
    f. Napraviti hospital.key, hospitl.crt i hospital.p12 za bolnicku aplikaciju
4) Potrebno je da malopre kreiran rootCA sertifikat bude u trusted sertifikatima naseg racunara:
    a. Desni klik na windows meni ikonicu
    b. “Run”
    c. Uneti “mmc” i kliknuti Ok
    d. Nakon sto iskoci dijalog kliknuti Yes
    e. Kliknuti file i odabrati ..Console1.msc
    f. Desni klik na trusted root certification authorities -> All tasks -> Import nas rootCA.crt
    g. Nas racunar sada posmatra nas novokreirani rootCa sertifikat kao sertifikat kom moze da veruje iako ga nije izdalo pravo sertifikaciono telo
5) U adminFront u hijerarhiji foldera na nivou gde se nalazi i “src” folder napravili “ssl” folder i u njega staviti admin.crt i admin.key

6) Isto to uraditi za hospitalFront samo sa hospital.key i hospital.crt
7) U angular.json dodati:

```
"serve": {
    "builder": "@angular-devkit/build-angular:dev-server",
    "options": {
      "browserTarget": "adminFront:build",
      "ssl": true,
      "sslCert": "./ssl/admin.crt",
      "sslKey": "./ssl/admin.key"
    }
```

8) Isto to uraditi i u hospitalFront samo sa hospital.key i hospital.crt
9) Svuda u aplikaciji http zameniti sa https

Koraci za implementiranje https-a na back-u:

1) U src -> main -> resources adminApp ubaciti admin.p12
2) U application.properties dodati:

```
#https
security.require-ssl=true

# The format used for the keystore
server.ssl.key-store-type=PKCS12
# The path to the keystore containing the certificate
server.ssl.key-store=src/main/resources/admin.p12
# The password used to generate the certificate
server.ssl.key-store-password=AdminSecretPass – password koji je unet pri kreiranju *.p12
# The alias mapped to the certificate
server.ssl.key-alias=1
```

3) Isto uraditi i u hospitalApp samo sa hospital.p12 i odgovarajucom sifrom
4) Napomena: moguce je da dodje do cross origin validation greske, ukoliko dodje do toga odgovarajuce kontrolere anotirati sa @CrossOrigin(origins = "https://localhost:4201") ili @CrossOrigin(origins = "https://localhost:4200")

Nakon uspesno implementiranih koraka trebalo bi da je omogucena sigurna komunikacija front-back u obe aplikacije. Trebalo bi jos da se implementira sigurna komunikacija izmedju dva back-a.

Koraci za implementiranje https-a za komunikaciju back-back:

1) HospitalApp salje zahteve ka adminApp pa je potrebno implementirati jos neke stvari da bi ta komunikacija bila moguca

2) U pom.xml hospitalApp dodati:

```
<!--Dependecy for http-->
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
            <version>4.5.3</version>
        </dependency>
```

3) U “config” folderu napraviti novu konfiguracionu klasu(anotacija @Configuration) i implementirati sledecu metodu:

```
@Bean
 RestTemplate restTemplate() throws KeyStoreException, IOException,
NoSuchAlgorithmException,
        UnrecoverableKeyException, KeyManagementException, CertificateException {
    KeyStore keyStore = KeyStore.getInstance(KeyStore.getDefaultType());
    keyStore.load(new FileInputStream(new ClassPathResource(keyStoreName).getFile()),
        keyStorePass.toCharArray());
    KeyStore trustStore = KeyStore.getInstance(KeyStore.getDefaultType());
    trustStore.load(new FileInputStream(new
ClassPathResource("truststoreHospital.jks").getFile()),
        trustStorePass.toCharArray());
    SSLConnectionSocketFactory socketFactory = new SSLConnectionSocketFactory(
        new SSLContextBuilder()
            .loadTrustMaterial(trustStore, new TrustSelfSignedStrategy())
            .loadKeyMaterial(keyStore, keyStorePass.toCharArray()).build());
    CloseableHttpClient httpClient =
HttpClients.custom().setSSLSocketFactory(socketFactory).build();
    ClientHttpRequestFactory requestFactory = new
HttpComponentsClientHttpRequestFactory(httpClient);
    return new RestTemplate(requestFactory);
```

}

4) U obe aplikacije u src->main-> resources potrebno je dodati odgovarajuci truststore – u adminApp truststoreAdmin.jks u kom se nalaze svi sertifikati kojima ta aplikacija moze da veruje
5) Sigurna komunikacija/https je omogucen i za adminApp i za hosptalApp i za njihovu medjusobnu komunikaciju!!!

Ostali sajtovi:

https://medium.com/@tbusser/creating-a-browser-trusted-self-signed-ssl-certificate-2709ce43fd15 – generisanje sertifikata

https://stackoverflow.com/questions/39210467/get-angular-cli-to-ng-serve-over-https?fbclid=IwAR0dxfRUPVTGodd9SEg8N__YjONV-RzogZoEqyxOkFZBhdHqVXOoyx04mAo – gde u angular aplikaciji treba da se postave kljuc i sertifiakt da bi komunikacija bila https

https://stackoverflow.com/questions/34156938/openssl-hangs-during-pkcs12-export-with-loading-screen-into-random-state?fbclid=IwAR3y7-iGonsdsUaLOFrMa3SHMMizg0wkj8Xx2XDWjcduwSUPA4Nfsu-rPM8 – dodavalje “winpty”

https://betterprogramming.pub/trusted-self-signed-certificate-and-local-domains-for-testing-7c6e6e3f9548 - stavljanje sertifikata u racunarov certificate truststore

https://medium.com/swlh/how-to-secure-a-spring-boot-application-with-tls-176062895559 - konfiguracija spring boot aplikacije za https

https://stackoverflow.com/questions/26180650/unable-to-find-valid-certification-path-to-requested-target-but-browser-says?fbclid=IwAR05gseoQZmFo3dZLzfr51WzDdeP7GnAAIKA9Hmm3P9jUjy_y8QPxM4XF78 i

https://stackoverflow.com/questions/9619030/resolving-javax-net-ssl-sslhandshakeexception-sun-security-validator-validatore?rq=1&fbclid=IwAR31wl0J8JvjPvEn5UvXxKdM2W6oCH75apVczJLCTksBwMiwS8LU3MIelY0 – dodavanje trusted sertifikata u java konfiguraciju

https://www.educative.io/edpresso/keystore-vs-truststore i https://www.baeldung.com/java-keystore-truststore-difference - upoznavanje sa truststore

https://stackoverflow.com/questions/47434877/how-to-generate-keystore-and-truststore?fbclid=IwAR1O9_Y1uu9wrmDJ49fL8lY4kft7KjCZmOLd0UWzDMwTnXEz6-MIuByfHu0 – generisanje truststore