

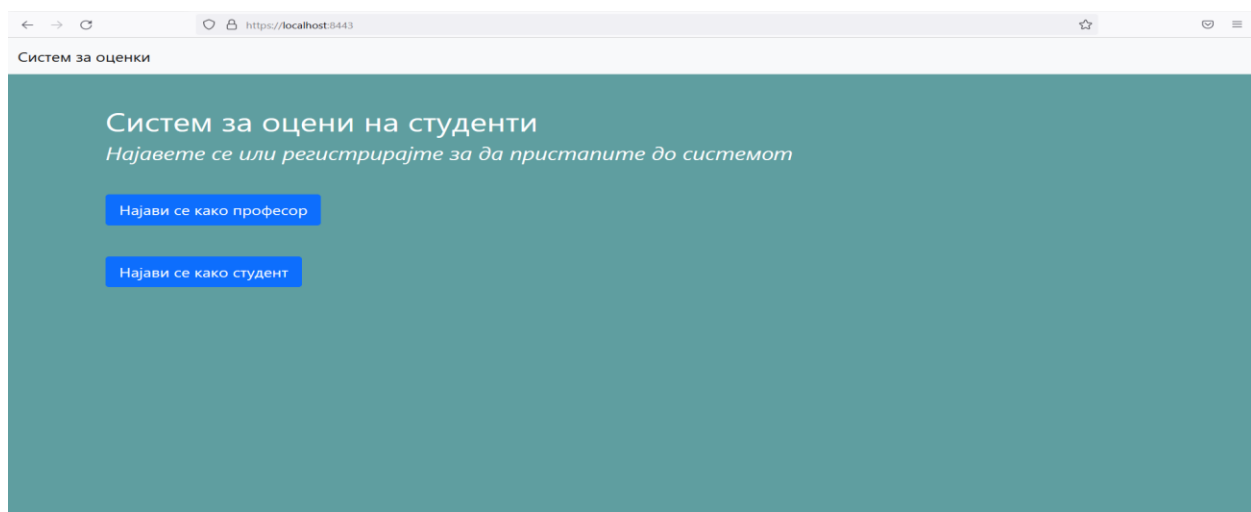
„Систем за оцени на студенти сличен на Iknow“

Проект по предметот Информациска безбедност

1) Објаснување на системот

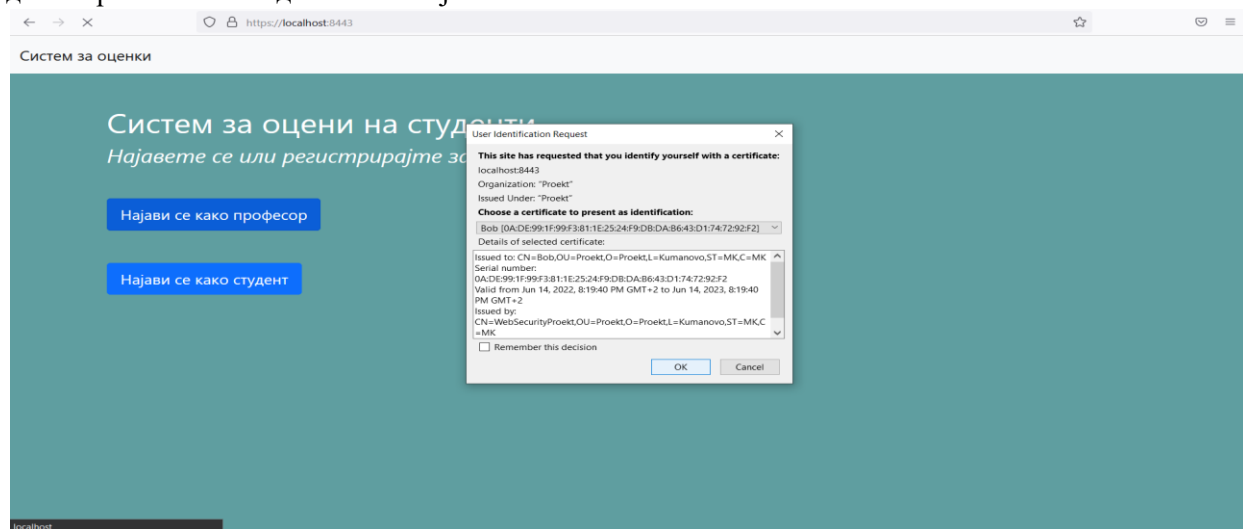
Нашата апликација претставува систем за оцени на студенти сличен на Iknow на која имаме повеќе функционалности кои се користат по воспоставување на безбедна врска со системот. На системот професорот се најавува со сертификат, додека студентите се најавуваат со внесување на корисничко име и лозинка.

Најпрвин на апликацијата може да се најавиме како студент или како професор.



Слика 1: Почетна страна на системот

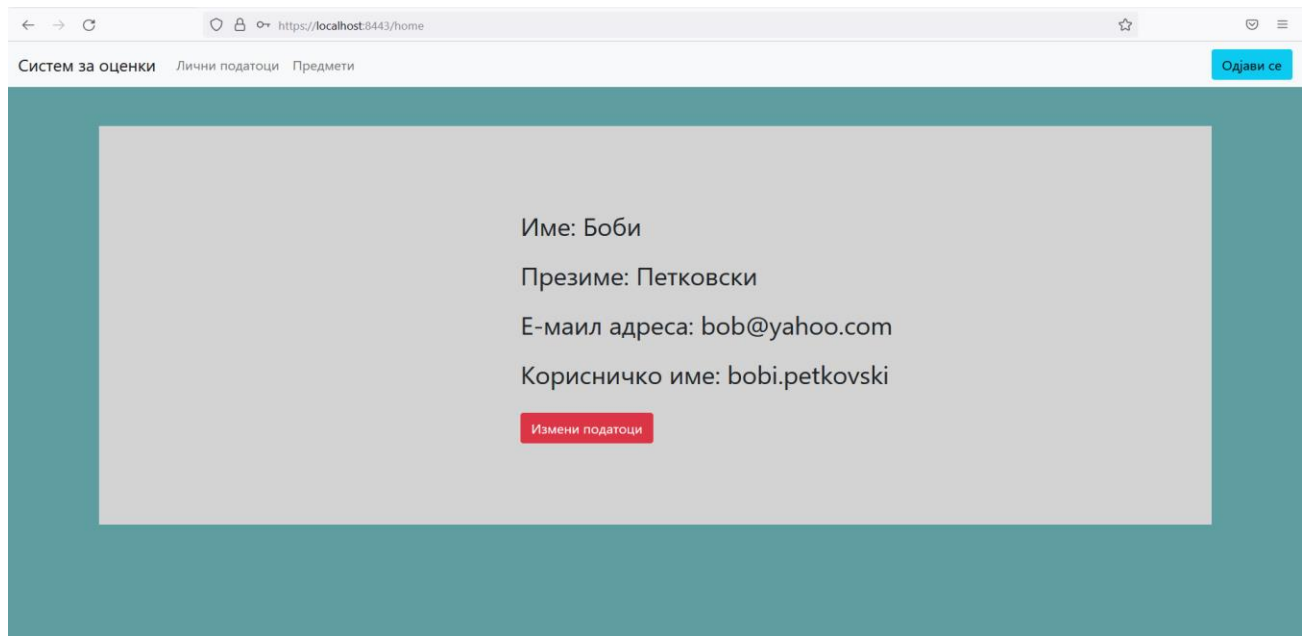
Доколку избереме да се најавиме како професор, треба да се приложи сертификат да се потврди дека корисникот има дозвола за најава.



Слика 2: Приказ на сертификат при најава

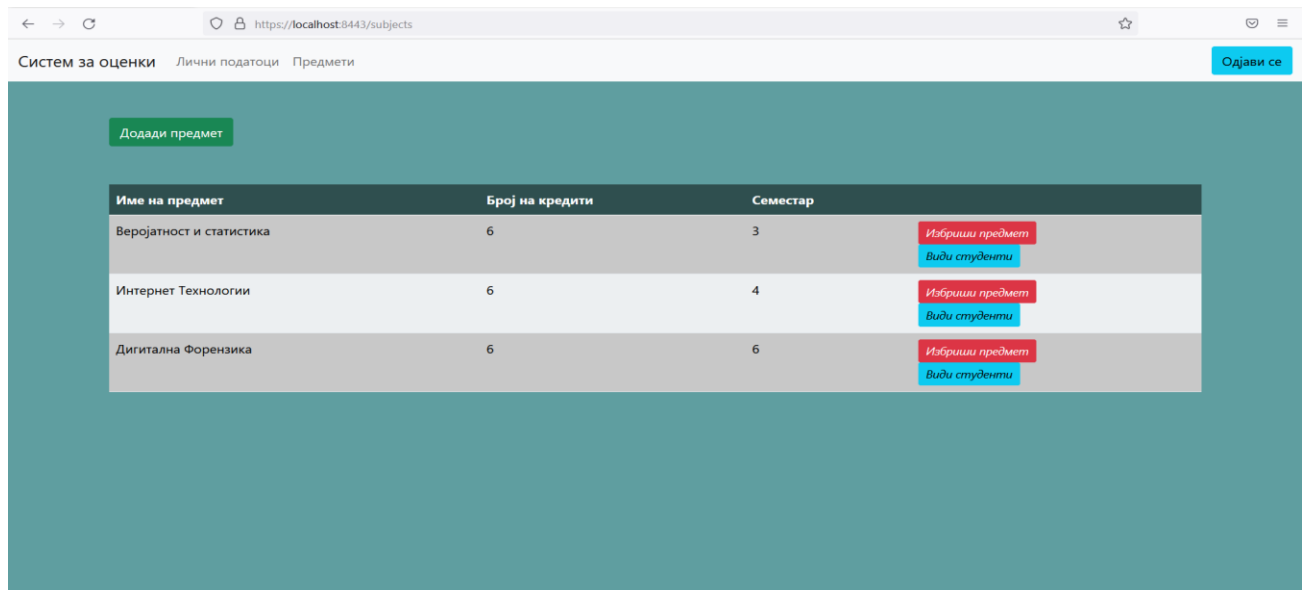
По избирање на сертификатот, корисникот автоматски го потврдува својот идентитет на серверот и на тој начин се најавува.

По најавата, професорот има приказ на своите лични податоци, на кои соодветно може да направи измена доколку има потреба и истата измена ќе се пренесе во базата на податоци.



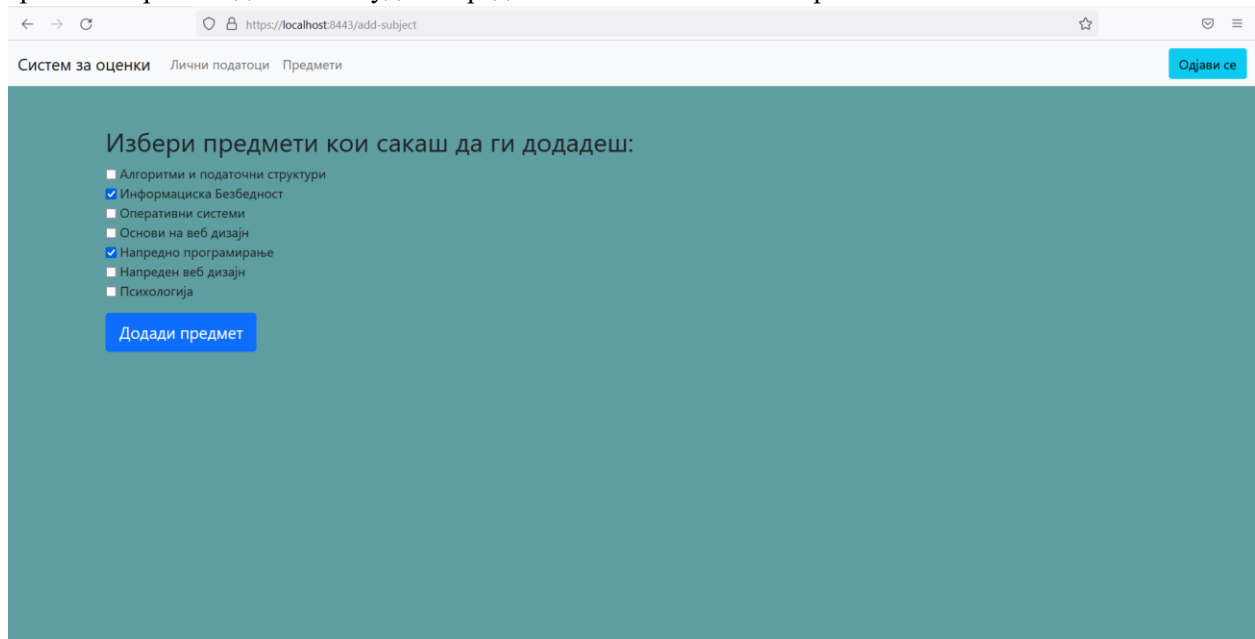
Слика 3: Лични податоци на професор

Професорот исто така има пристап и до предметите на факултетот, преку навигациското мени. На таа страна му се прикажуваат сите предмети на кои тој предава, при тоа има можност да додаде предмет или да острани предмет, но и да ги види студентите по даден предмет.



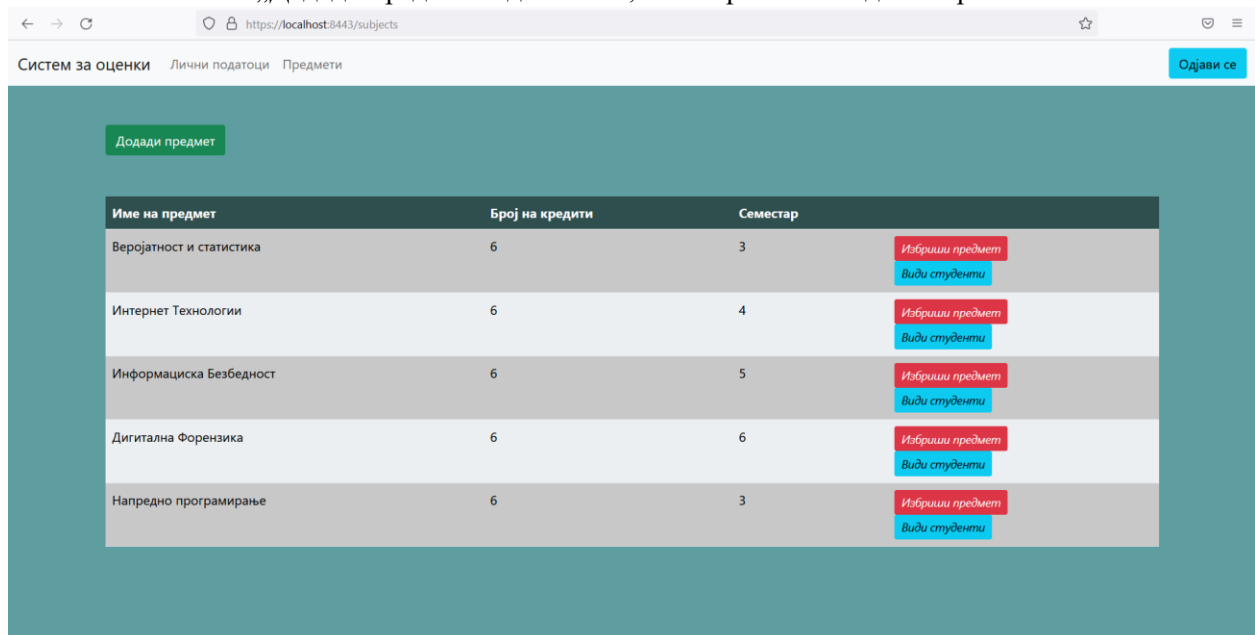
Слика 4: Приказ на сите изберени предмети

Доколку професорот сака да додаде нов предмет, по клик на копчето „Додади предмет“, ќе му се прикаже страна каде има понудени предмети кои не се веќе изберени на Слика 4.



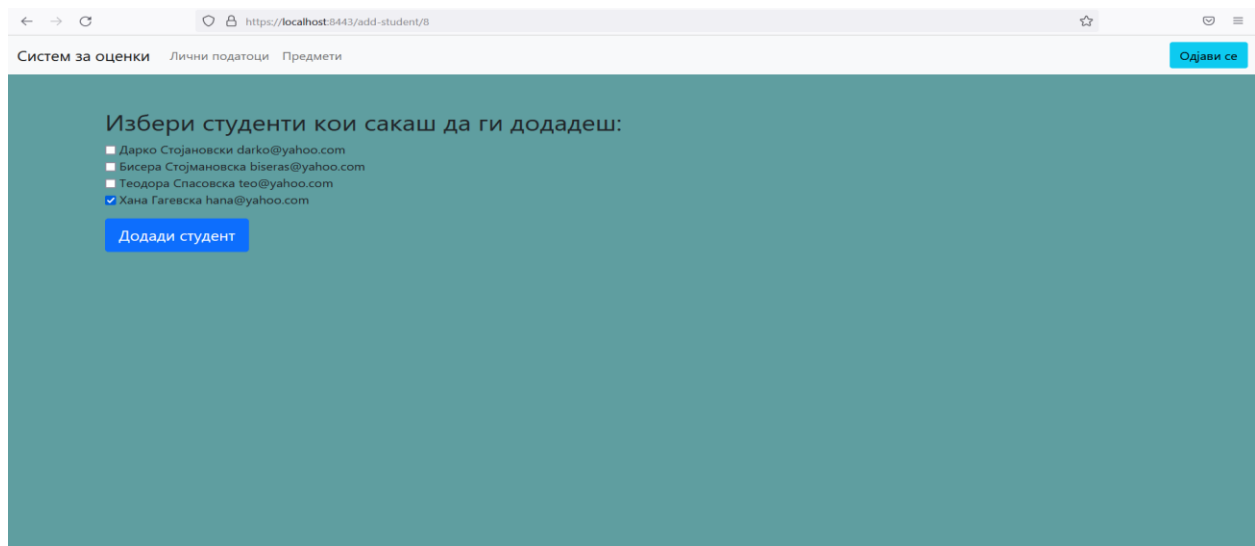
Слика 5: Додавање на предмети

По клик на копчето „Додади предмет“ од Слика 5, ќе се прикаже следниот екран:



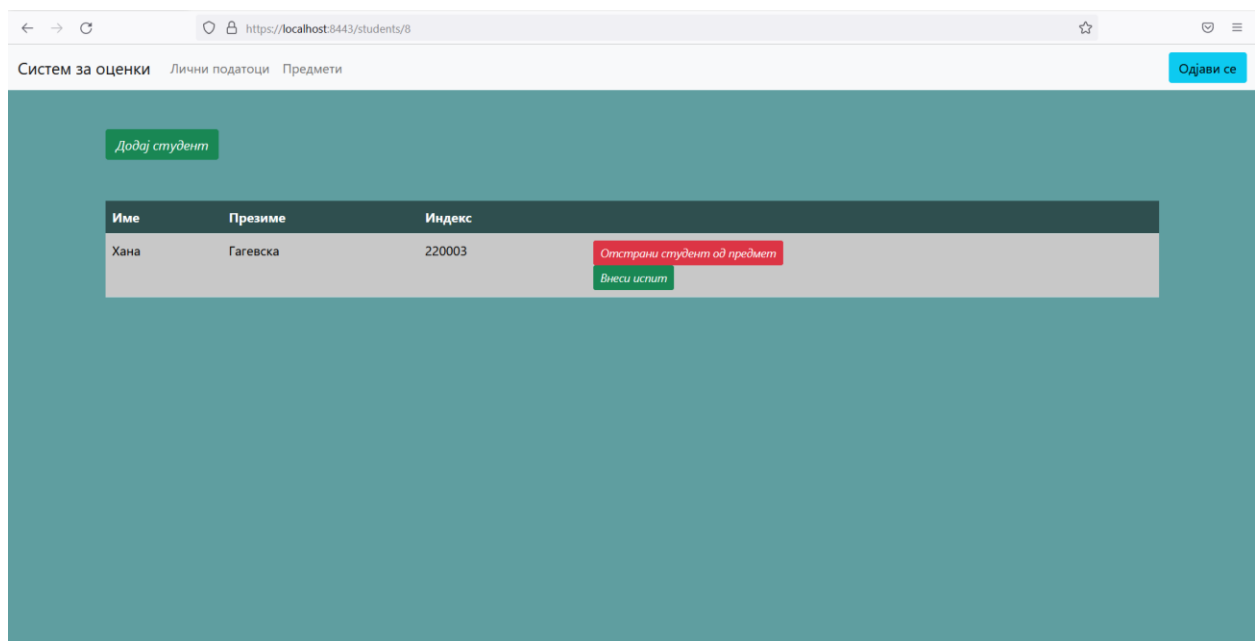
Слика 6: Екран по додавање на нови предмети

По клик на копчето „Види студенти“, за ново додадениот предмет „Напредно програмирање“, ќе може да додаде студенти за новиот предмет.



Слика 7: Додавање на студент по даден предмет

Во овој пример, се додава студентот Хана Гагевска на новиот предмет и се прикажуваат информации за студентот, како и опции за отстранување на студентот од предметот, но и внесување на испит.



Слика 8: Приказ на студенти за даден предмет

По клик на копчето „Внеси испит“ од Слика 8, ќе се прикаже екран каде се податоците за студентот кои се превземени од базата на податоци, но и податоци за испит кои треба да се потполнат како што се оценка, датум на полагањето и сесијата.

Систем за оценки | Лични податоци | Предмети

Одјави се

Внеси испит по предметот Напредно програмирање

Податоци за студентот:

Име: Хана

Презиме: Гагевска

Индекс: 220003

Податоци за испитот:

Оценка: 8

Датум на испитот: 6/27/2022

Сесија: јунска

Внеси испит

Слика 9: Внесување испит на студент по даден предмет

Доколку избереме да се најавиме како студентот Хана Гагевска, на која претходно и беше внесен испит, може да го направиме со избирање на најава како студент. Бидејќи само професорите се најавуваат преку сертификати, студентот се најавува со внесување на корисничко име и лозинка.

https://localhost:8443/login

Please sign in

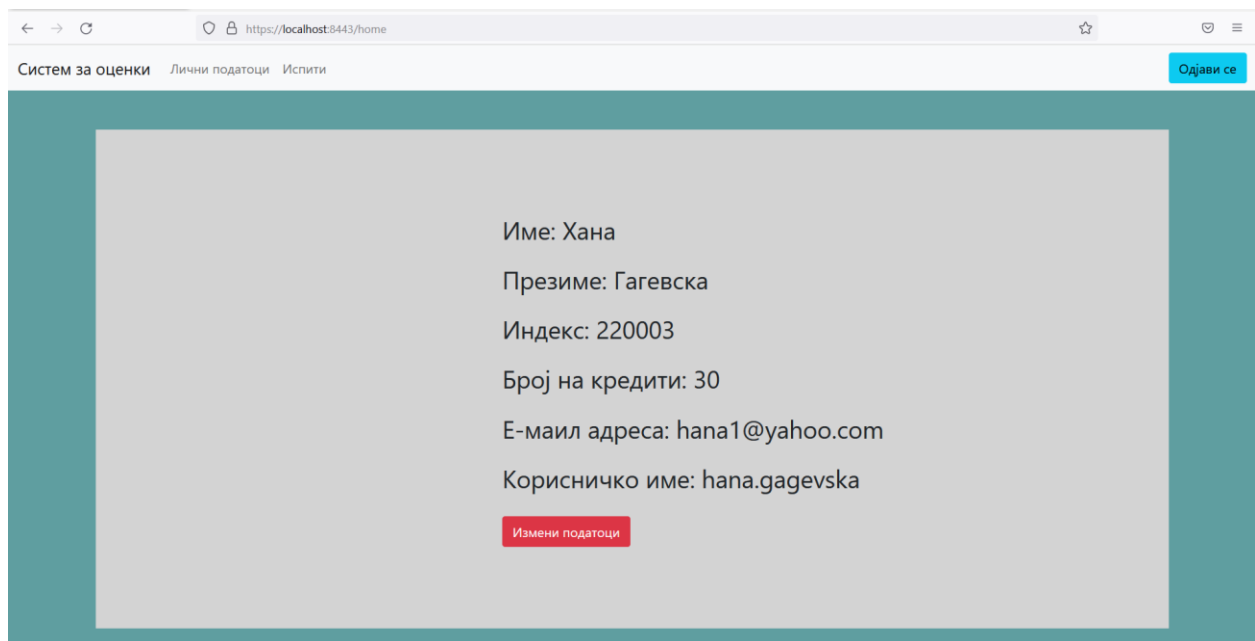
hana.gagevska

••

Sign in

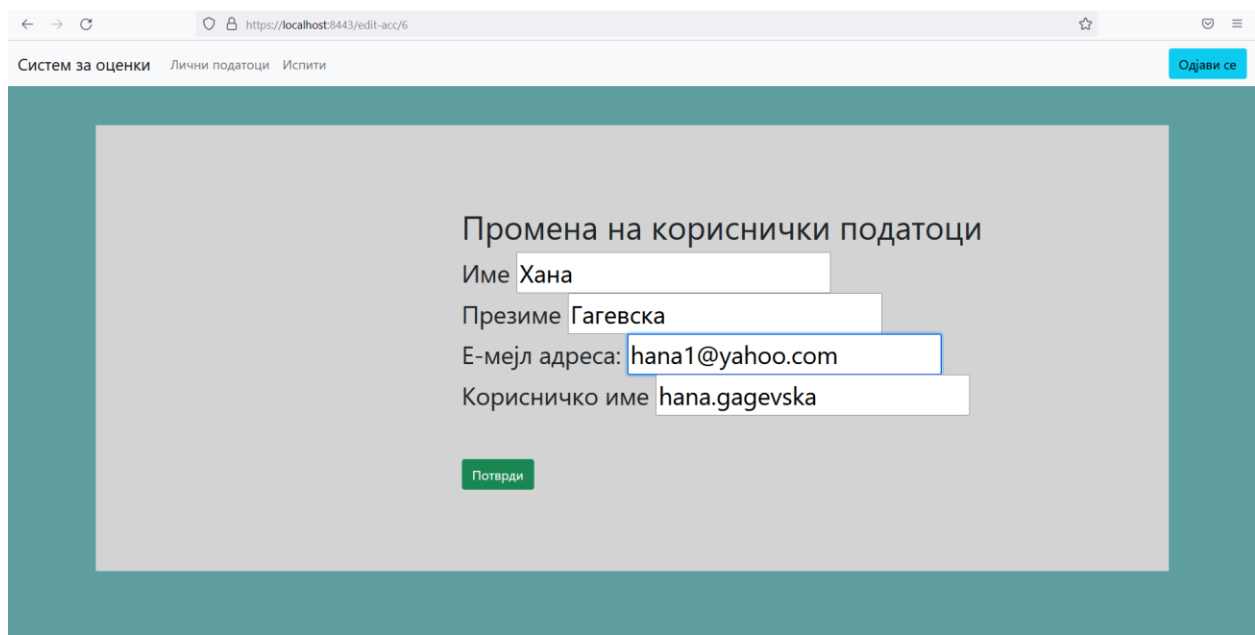
Слика 10: Најава на студент со корисничко име и лозинка

По најавата се прикажува екранот со лични податоци на студентот.



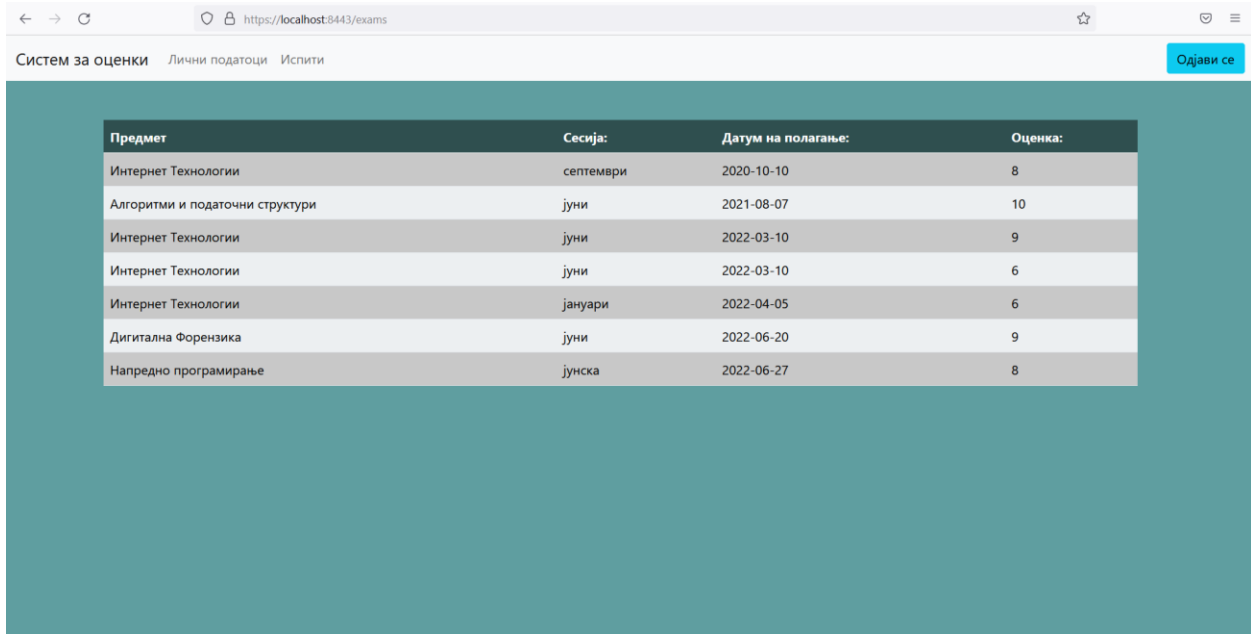
Слика 11: Лични податоци на студент

Тука повторно има можност студентот да направи измена на своите податоци, како што беше можноста и доколку професорот се најави на системот, по што соодветната промена ќе се зачува во базата на податоци и ќе се прикажува како таква кај сите други професори.



Слика 12: Екран за промена на лични податоци на студент

Доколку студентот кликне на „Испити“ од навигациското мени, ќе добие преглед на сите положени испити како и информации за истите. Каде што во последен ред е испитот кој претходно во овој пример професорот го внел.



Предмет	Сесија:	Датум на полагање:	Оценка:
Интернет Технологии	септември	2020-10-10	8
Алгоритми и податочни структури	јуни	2021-08-07	10
Интернет Технологии	јуни	2022-03-10	9
Интернет Технологии	јуни	2022-03-10	6
Интернет Технологии	јануари	2022-04-05	6
Дигитална Форензика	јуни	2022-06-20	9
Напредно програмирање	јунска	2022-06-27	8

Слика 13: Приказ на испити на студент

2) Објаснување на безбедносни аспекти на системот

При најава професорот треба да користи сертификат. Се работи за сертификатот за автентикација X.509 – потврдување на идентитетот при користење на протоколот HTTPS (HTTP преку SSL). Додека е воспоставена безбедна врска клиентот го верифицира серверот според неговиот сертификат (издаден од доверлив орган за сертификати). X.509 во Spring Security може да се користи за да се потврди идентитетот на клиентот од серверот додека се поврзува.

За да можеме да ги потпишеме нашите сертификати од страна на серверот и клиентот, прво треба да создадеме сопствен самопотпишан root CA сертификат. На овој начин ние ќе дејствуваме како сопствен орган за сертификати. За таа цел ја користевме OpenSSL библиотека, која мора да ја инсталираме. Со наредбата `openssl req -x509 -sha256 -days 3650 -newkey rsa:4096 -keyoutrootCA.key -out rootCA.crt` креираме CA сертификат.

По оваа команда мора да дадеме лозинка за нашиот приватен клуч. Keystore е складиште што нашата апликација во Spring Boot ќе го користи за да ги чува приватниот клуч и сертификатот на нашиот сервер. Односно, нашата апликација ќе ја користи keystore за да им го сервира сертификатот на клиентите за време на SSL ракување. Ние користиме Java Key-Store (JKS) формат и алатка за keytool командна линија.

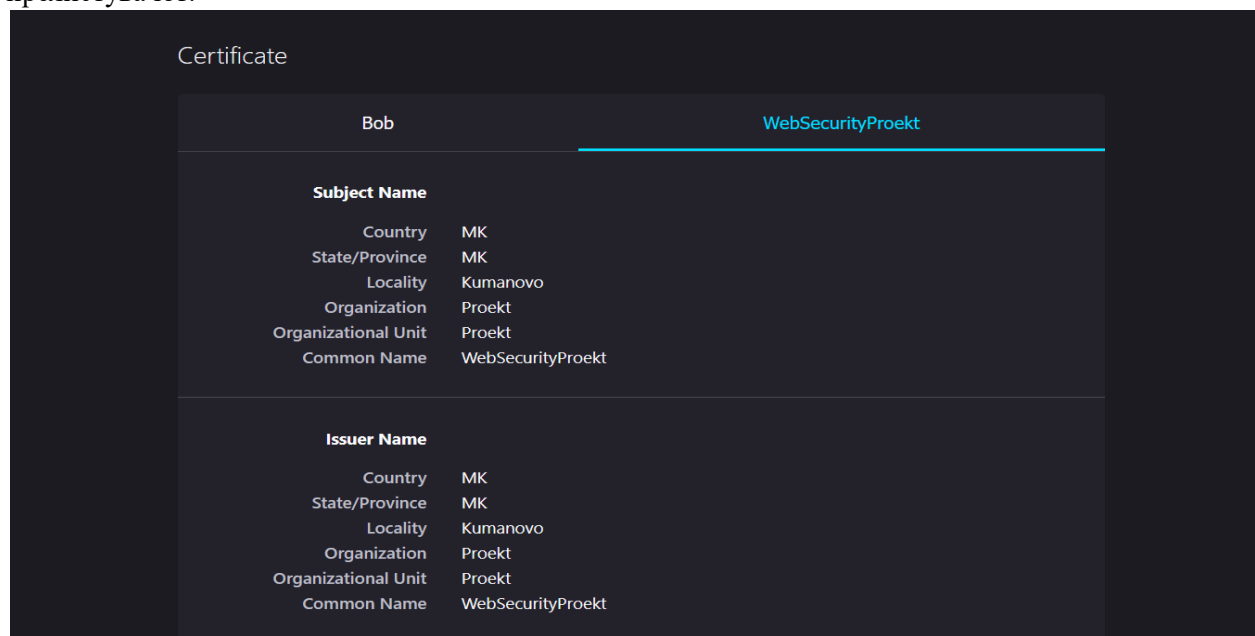
За да ја имплементираме автентикацијата X.509 од страна на серверот во нашата апликација Spring Boot, прво создадовме сертификат од страна на серверот. Барање за потпишување сертификат (CSR): `openssl req -new -newkey rsa:4096 -keyoutlocalhost.key -out localhost.csr`. Ја креиравме конфигурациска датотека – localhost.ext. Таа ќе складира некои дополнителни параметри потребни за време на потпишувањето на сертификатот.

Потоа се потпишува барање со нашиот сертификат и неговиот приватен клуч : `openssl x509 -req -CA rootCA.crt -CAkeyrootCA.key -inlocalhost.csr -out localhost.crt -days 365 -CAcreateserial -extfilelocalhost.ext`. Со ова имаме подготвен за користење localhost.crt сертификат потпишан од нашиот сопствен орган за сертификати. За да ги испечатиме деталите за нашиот сертификат во форма што може да се чита од човек, ја користиме следнава команда: `openssl x509 -in localhost.crt -text`.

Ја користиме архивата PKCS 12 за да го спакуваме приватниот клуч на нашиот сервер заедно со потпишаниот сертификат. Потоа го импортираме во новосозданиот keystore.jks. За да се создаде датотека .p12 се користи наредбата `openssl pkcs12 -export -out localhost.p12 -name "localhost" -inkeylocalhost.key -in localhost.crt`. Сега ги имаме localhost.key и localhost.crt во комплет во единствената датотека localhost.p12, што претставува сертификат на серверот.

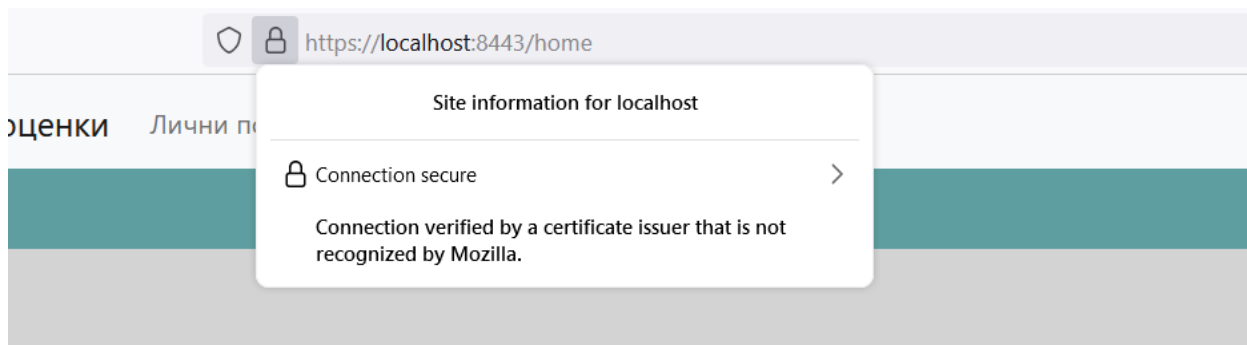
Потоа користиме keytool за да креираме складиште keystore.jks и да ја импортираме датотеката localhost.p12 со командата: `keytool -importkeystore -srckeystore localhost.p12 -srcstoretype PKCS12 -destkeystorekeystore.jks -deststoretype JKS`. Ова е потребно за автентикација на серверот.

Наредно го прикачиме нашиот root certificate authority како доверлив сертификат во прелистувачот.



Слика 14: Приказ на сертификат на серверот прикачен на прелистувач

По пребарување на <https://localhost:8443/>, можеме да го провериме статусот на врската со кликување на симболот „зелена брава“ во лентата за адреси и може да видиме дека врската е безбедна.



Слика 15: Приказ на безбедна конекција воспоставена со серверот

Следно е другиот дел од автентикацијата – автентикација на клиентот. На овој начин, само клиентите со валидни сертификати потпишани од органот на кој што му верува нашиот сервер, можат да пристапат до нашата заштитена веб-страница.

Но, пред да продолжиме, да видиме кои се добрите и лошите страни на користењето на меѓусебната SSL автентикација.

Добри страни се што приватниот клуч на сертификат за клиент X.509 е посилен од која било лозинка дефинирана од корисникот, но тоа мора да се чува во тајност! Со сертификат, идентитетот на клиентот е добро познат и лесен за проверка, но и нема повеќе заборавени лозинки.

Од друга страна лоши страни се што треба да создадеме сертификат за секој нов клиент. Потоа сертификатот на клиентот треба да се инсталира во клиентска апликација. Всушност: автентикацијата на клиентот X.509 зависи од уредот, што го оневозможува користењето на овој вид автентикација на јавни места, на пример во интернет-кафе. Но и мора да ги одржуваме сертификатите на клиентите. Ова лесно може да стане скапо.

Truststore на некој начин е спротивно на keystore. Ги поседува сертификатите на надворешните субјекти на кои им веруваме. Во нашиот случај, доволно е да го чуваме root CA сертификатот во truststore. Еве како се креира датотека truststore.jks и се импортира rootCA.crt со помош на `keytool:keytool -import -trustcacerts -noprompt -alias ca -extsan=dns:localhost,ip:127.0.0.1 -file rootCA.crt -keystore truststore.jks`.

Нашиот проект за заштитен SSL сервер се состои од конфигурации во класата која е означена со `@SpringBootApplication` (што е еден вид `@Configuration`) и додавање потребни детали во конфигурациска датотека `application.properties`.

Импортиравме сопствен CA сертификат и truststore е подготвена за употреба. За да продолжиме, го менуваме нашиот `X509AuthenticationServer` да се наследува од `WebSecurityConfigurerAdapter` и да преоптовари еден од дадените методи за конфигурирање. Овде го конфигурираме механизмот X.509 да го парсира полето Common Name (CN) на сертификатот за извлекување кориснички имиња. За овие извлечени кориснички имиња, Spring Security бапа `UserDetailsService` за соодветните корисници.

Па, ние исто така го имплементираме овој сервисен интерфејс кој содржи еден демо-корисник. Мора да забележиме дека нашата класа ја нотираме со `@EnableWebSecurity` и `@EnableGlobalMethodSecurity` со овозможена пред/пост-авторизација.

```
WebsecurityprojectApplication.java
28 public static void main(String[] args) { SpringApplication.run(WebsecurityprojectApplication.class, args); }
29
30 @Bean
31 PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(10); }
32
33 @Autowired
34 private UserDetailsService userDetailsService;
35
36 @Override
37 protected void configure(HttpSecurity http) throws Exception {
38     http.csrf().disable().authorizeRequests().expressionInAuthorizationConfigurer().expressionInterceptUrlRegistry()
39         .anyRequest().authenticated()
40         .and().httpSecurity()
41         .x509().x509Configurer().httpSecurity()
42         .subjectPrincipalRegex("CN=(.*)?(?:,|$)")
43         .userDetailsService(userDetailsService())
44         .and().httpSecurity()
45         .formLogin().formLoginConfigurer().httpSecurity()
46         .defaultSuccessUrl("/home", alwaysUse: true)
47         .and().httpSecurity()
48         .logout().logoutConfigurer().httpSecurity()
49         .logoutUrl("/logout")
50         .clearAuthentication(true)
51         .invalidateHttpSession(true)
52         .deleteCookies("SESSIONID")
53         .logoutSuccessUrl("/");
54 }
```

Слика 16.1: Приказ на класата WebsecurityprojectApplication.java

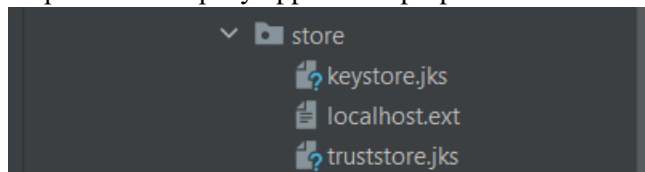
```

35 @Bean
36 UserDetailsService userDetailsService() {
37     return new UserDetailsService() {
38         @Override
39         public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
40             if (username.equals("Bob")) {
41                 return new User(username, password: "", AuthorityUtils.commaSeparatedStringToAuthorityList("ROLE_ADMIN"));
42             }
43             throw new UsernameNotFoundException("User not found!");
44         }
45     };
46 }
47
48 @Override
49 protected void configure(AuthenticationManagerBuilder auth) throws Exception {
50     auth.userDetailsService(this.userDetailsService);
51 }
52
53 |
54 }
```

Слика 16.2: Приказ на класата WebsecurityprojectApplication.java

Тука покрај тоа што ги наведовме потребните работи за да работи X.509 автентикација, додадовме и други делови од Spring Security како што се страна за логирање и за одјавување.

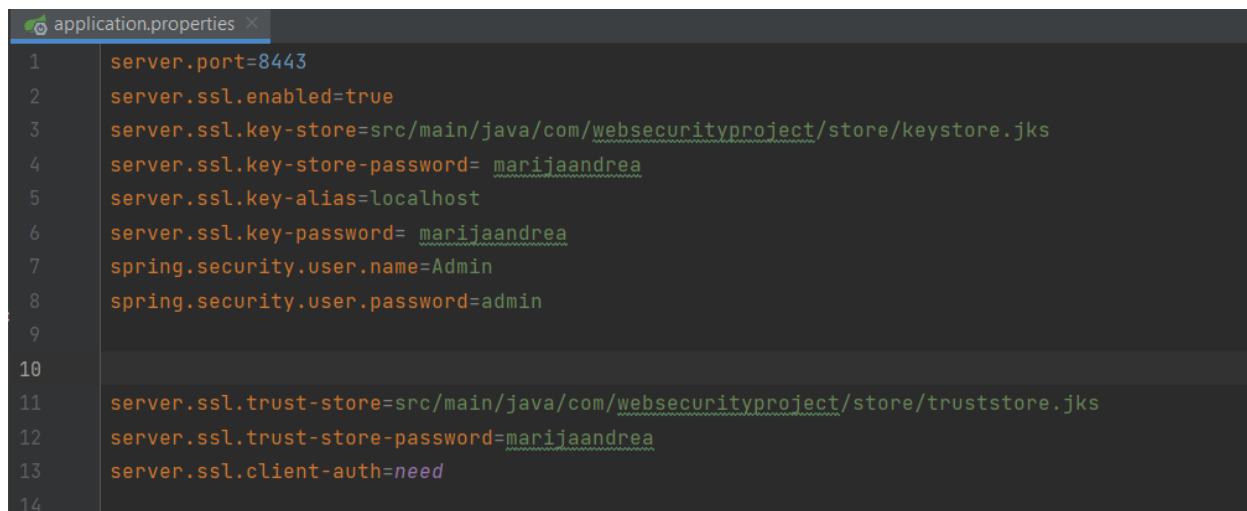
Понатаму, новите генерирани датотеки за автентикација ги сместуваме во пакетот store во нашата апликација со цел да ги пристапиме преку application.properties.



Слика 17: Приказ на пакетот store

Сега, и кажуваме на апликацијата каде да го најдена нашиот keystore.jks и како да пристапи до него. Го поставивме SSL на статус „enabled“ и ја менуваме стандардната порта за слушање за да укажена обезбедена врска. Дополнително, конфигурираме некои кориснички детали за пристап до нашиот сервер преку основна автентикација. Како последен чекор на модификација, треба да и

кажеме на апликацијата каде се наоѓа нашата truststore и дека е неопходна автентикација на SSL клиент (server.ssl.client-auth=need). Во нашата application.properties го ставаме:

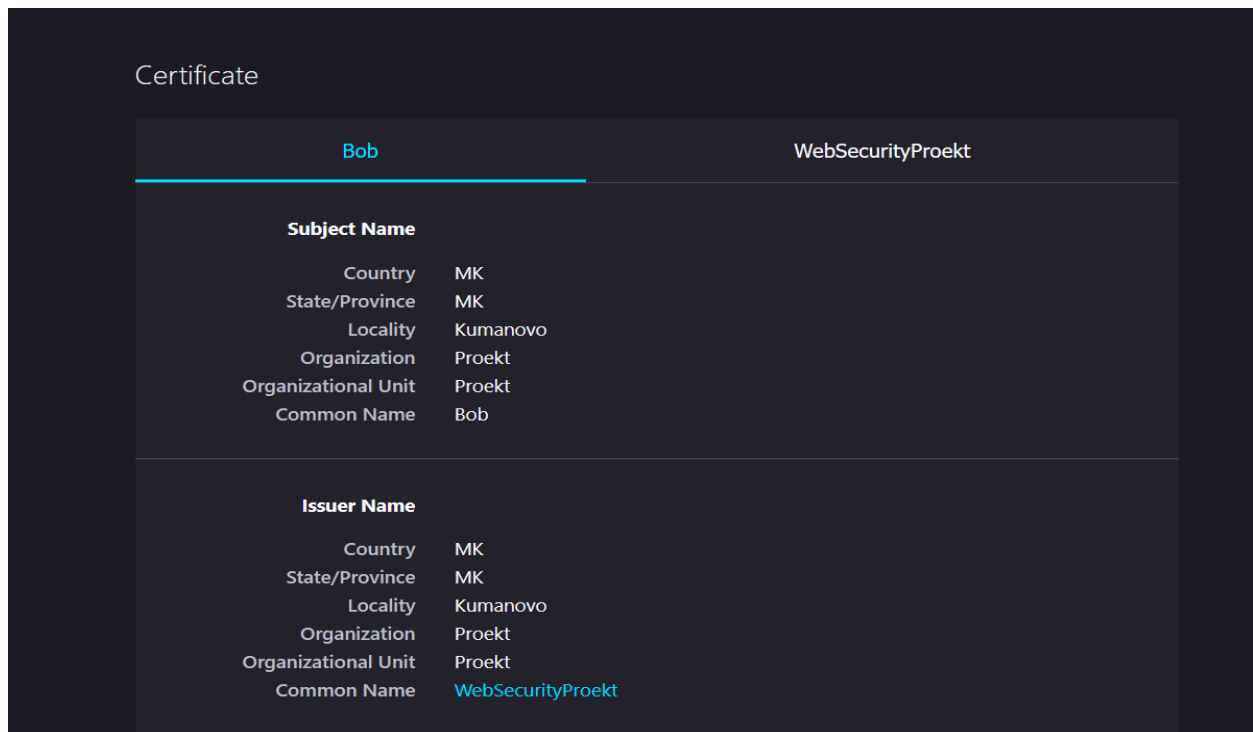


```
1 server.port=8443
2 server.ssl.enabled=true
3 server.ssl.key-store=src/main/java/com/websecurityproject/store/keystore.jks
4 server.ssl.key-store-password= marijaandrea
5 server.ssl.key-alias=localhost
6 server.ssl.key-password= marijaandrea
7 spring.security.user.name=Admin
8 spring.security.user.password=admin
9
10
11 server.ssl.trust-store=src/main/java/com/websecurityproject/store/truststore.jks
12 server.ssl.trust-store-password=marijaandrea
13 server.ssl.client-auth=need
14
```

Слика 18: Приказ на фајлот application.properties

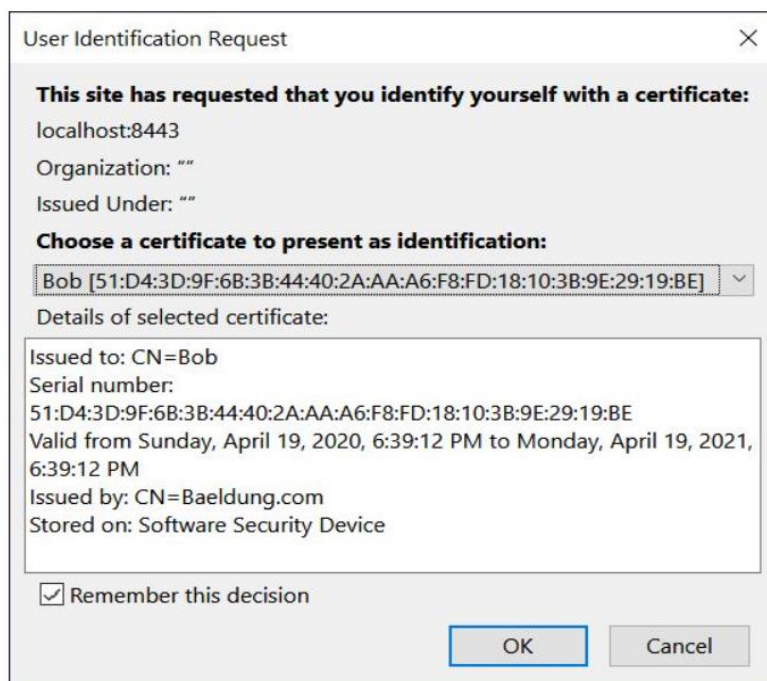
Нареден чекор беше да создадеме сертификат од страна на клиентот. Прво, направивме барање за потпишување сертификат: `openssl req -new -newkey rsa:4096 -nodes -keyoutclientBob.key -out clientBob.csr`. Потоа обезбедвме информации што ќе бидат вградени во сертификатот. Ние го внесовме само заедничкото име (CN) – Боб. Важно е бидејќи го користиме овој запис за време на авторизација и само Боб е препознаен од нашата апликација за примерок.

Го потпишуваме барањето со нашиот CA: `openssl x509 -req -CA rootCA.crt -CAkeyrootCA.key -inclientBob.csr -out clientBob.crt -days 365 -CAcreateserial`. Последниот чекор што треба да го направиме е да го спакуваме потпишаниот сертификат и приватниот клуч во датотеката PKCS: `openssl pkcs12 -export -out clientBob.p12 -name "clientBob" -inkeyclientBob.key -in clientBob.crt`. И како последен чекор го инсталираме сертификатот за клиентот во прелистувачот со цел да може да се користи при најава на системот.



Слика 19: Приказ на сертификат на клиентот прикачен на прелистувач

Сега, кога ќе ја освежиме нашата веб-локација, ќе ни биде побарано да го избереме сертификатот за клиент што сакаме да го користиме:



Слика 19: Приказ на генерираниот сертификат за клиентот Боб

Во нашиот проект креираме самопотпишан СА сертификат и го користиме за потпишување други сертификати. Дополнително, создадовме и сертификати од страна на серверот и од клиентот. Потоа, претставивме како да ги импортираме во keystore и truststore соодветно.

Како дополнителни безбедносни делови користевме Password Encoder и Spring Security форма за логирање на студенти.

PasswordEncoder работи со хеширање, што го решава проблемот со непосреден пристап до базата на податоци на системот со откриени лозинки. Овој интерфејс го дефинира методот encode() за конвертирање на обичната лозинка во кодирана форма и методот matches() за споредба на обичната лозинка со кодираната лозинка. Ние го искористивме при запишување корисници до базата на податоци во сервисен слој, како и при измена на податоци на даден корисник.

Во наредната слика е прикажан метод од сервисен слој на ентитетот Professor, каде се користи PasswordEncoder за енкодирање на лозинката на корисникот.

```
@Override
public Professor add(String name, String lastname, String email, String username, String password, Role role, List<Long> subjects, List<Student> students) {
    List<Subject> subjectsList = this.subjectRepository.findAllById(subjects);
    Professor professor=new Professor(name,lastname,email,username,passwordEncoder.encode(password),role,subjectsList,students);
    return this.professorRepository.save(professor);
}
```

Слика 20: Користење на PasswordEncoder во сервисен слој на проектот

Изработено од:

- Марија Спасовска 191031
- Андреа Стојмановска 191161