

UNIVERSITÀ DEGLI STUDI DI URBINO

**DIPARTIMENTO DI SCIENZE PURE E APPLICATE
FACOLTÀ DI INFORMATICA APPLICATA**

**Progetto di Reti dei Calcolatori
CHAT LOGIN**

Crittografia del dominio tramite protocollo HTTPS/HSTS

di Costa Giulia

Matricola 308558

Anno Accademico 22/23 – Sessione estiva

Docente: Ing. Antonio Della Selva

Sommario

INTRODUZIONE	3
OBIETTIVI DEL PROGETTO.....	3
STRUTTURA DEL PROGETTO	4
ARCHITETTURA DEL SOFTWARE.....	4
CONFIGURAZIONE DEL PROTOCOLLO HTTPS E HSTS.....	5
STRUTTURA APPLICATIVA	7
ENCODING DEL LOGIN.....	9
PROTOCOLLO DI COMUNICAZIONE PER LA MESSAGGISTICA.....	10
CONSIDERAZIONI E CONCLUSIONI	11

INTRODUZIONE

OBBIETTIVI DEL PROGETTO

L'obiettivo del progetto è quello di creare un'applicazione che permetta di effettuare un login a seguito di una registrazione per accedere ad una chat dove è reso possibile lo scambio di messaggi.

L'applicazione è strutturata nei seguenti passi:

1. Accesso al link: <https://localhost/chatmessage/login>
2. Registrazione da parte dell'utente se non ancora registrato, tramite inserimento univoco di username (e-mail) e password.
Nel caso in cui l'utente sia già registrato tramite e-mail, sarà visualizzato un messaggio di Warning, che attesta la presenza dell'account.
3. Login tramite inserimento delle credenziali corrette. Nel caso in cui non vi siano credenziali errate o inesistenti sarà visualizzato un messaggio di errore.
4. Accesso alla chat con possibilità di connessione alla rete ed al relativo servizio di messaggistica istantanea.

Il secondo obiettivo è quello di consentire l'accesso a sole connessioni sicure, mediante il protocollo HTTPS e tramite la politica di sicurezza HSTS.

L'idea è quella di creare connessioni univoche, autorizzate e crittografate sia per la raccolta di dati in fase di login tramite encrypting delle password che per lo scambio di informazioni che avverranno all'interno della chat.

STRUTTURA DEL PROGETTO

ARCHITETTURA DEL SOFTWARE

Di seguito sono riportate le tecnologie utilizzate per lo sviluppo del progetto:

- **PROJECT OBJECT MODEL (APACHE MAVEN)**: strumento open source di gestione di progetti software basati su Java e build automation. Compila ed esporta progetti, esegue task automatizzati e gestisce le dipendenze dei progetti.
- **SERVER WEB (TOMCAT)**: fornisce una piattaforma software per l'esecuzione di applicazioni web sviluppate in lingua Java. Viene usato come contenitore servlet ¹per framework come Spring Boot.
- **FRAMEWORK (SPRING BOOT)**: è un modulo utilizzato per lo sviluppo di applicazioni java online e offline, in particolar modo usato per semplificare la creazione di applicazioni web, sfruttando gli starter, ovvero dipendenze attraverso le quali è possibile aggiungere funzionalità al framework base. Svolge una funzione di Controller nel MVC (pattern model-view-controller), che una volta invocata dal server può decidere quale pagina visualizzare o parte dell'applicazione invocare.

Le dipendenze utilizzate sono:

- **THYMLEAF**: template engine, ovvero un modulo che prende un contenuto testuale dei dati dinamici, li combina e restituisce il nuovo contenuto elaborato.
- **SPRING-BOOT-STARTER-WEB**: librerie e configurazioni per lo sviluppo di servizi web.
- **SPRING-BOOT-STARTER-SECURITY**: framework di autenticazione e controllo per gli accessi. Fornisce *autenticazione*² e *autorizzazione*³ per le applicazioni java. Se si usa un contenitore servlet non è necessario alcun file di configurazione.
- **SPRING-BOOT-DEVTOOLS**: include un set aggiuntivo di strumenti per fornire funzionalità aggiuntive in fase di sviluppo, come il riavvio automatico dell'applicazione, e la semplificazione della gestione delle risorse statiche come HTML, CSS e JavaScript (Classpath Static Resources).
- **SPRING-BOOT-STARTER-WEBSOCKET**: configurazione automatica alle websocket per Tomcat.
- **SPRING-BOOT-STARTER-DATA-JPA**: necessaria per l'implementazione di repository basata su JPA. Si occupa del supporto avanzato per i livelli di accesso ai dati.
- **MYSQL-CONNECTOR-J**: è un driver JDBC utilizzato per connettersi a un database, in questo caso a MySQL.
- **WEBJARS**: ovvero pacchetti Maven che permettono di includere librerie lato client, all'interno di un progetto Java.

Inclusione dei file necessari per il front end e la logica del trasporto dati:

- **LOGIN.HTML**: file contenente il front-end della struttura del login.
- **NEW.HTML**: file contenente il front-end della struttura per la registrazione del nuovo utente.
- **INDEX.HTML**: file contenente il front-end della struttura della chat.
- **APP.JS**: nella quale è contenuta la logica per la gestione della connessione, disconnessione e messaggistica, che si avvale dell'utilizzo dei file JSON.
- **HOME.CSS**: file di supporto per il front-end della chat.

¹ I *servlet* son oggetti scritti in linguaggio java che operano all'interno del server web permettendo la creazione di applicazioni web (elaborazione lato server). I programmi che implementano le specifiche dei servlet possono girare all'interno di qualunque servlet container e non sono vincolati ad un particolare server. Lo standard rientra all'interno di un vasto standard detto Java EE.

² L'*autenticazione* è il processo di verifica dell'identità di un utente o di un sistema. Consiste nel fornire credenziali, come nome utente e password, per dimostrare di essere l'entità dichiarata.

³ L'*autorizzazione* è il processo di attribuzione dei privilegi o dei diritti di accesso a un utente autenticato ed essa viene stabilita in base alle politiche di sicurezza dell'applicazione o del sistema

CONFIGURAZIONE DEL PROTOCOLLO HTTPS E HSTS

Il **protocollo HTTPS** (HyperText Transfer Protocol Secure) è una variante del protocollo HTTP utilizzata per la comunicazione sicura su Internet; i dati vengono crittografati utilizzando un protocollo a chiave pubblica, Secure Sockets Layer (SSL), che creano un canale sicuro tra il client e il server, al fine di garantire la riservatezza delle informazioni.

Quando un client accede a un sito web tramite HTTPS, avviene un processo di “*handshake*”⁴ tra il client e il server, in cui viene stabilita una connessione sicura e vengono scambiate le chiavi di crittografia.

1. GENERAZIONE DEL CERTIFICATO DIGITALE SSL AUTOFIRMATO CON JAVA KEYTOOLS:

SSL/TLS è utilizzato per creare un canale crittografato tra un client (ad esempio, un browser web) e un server, garantendo che i dati trasmessi siano protetti da accessi non autorizzati e manipolazioni durante il trasferimento (crittografia, autenticazione ed integrità).

I siti web che utilizzano SSL/TLS sono riconoscibili dal prefisso "https://" nell'URL.

ABILITAZIONE DI UNA COPPIA DI CHIAVI USANDO L'ALGORITMO RSA KEYSIZE.

```
PS C:\Program Files\Java\jdk-19\bin\SSL\self-signed> keytool -genkeypair -alias local_ssl -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore local-ssl.p12 -validity 365 -ext san=dns:localhost
Enter keystore password:
Re-enter new password:
Enter the distinguished name. Provide a single dot (.) to leave a sub-component empty or press ENTER to use the default value in braces.
What is your first and last name?
  [Unknown]: giulia.costa
What is the name of your organizational unit?
  [Unknown]: localhost
What is the name of your organization?
  [Unknown]: Development
What is the name of your City or Locality?
  [Unknown]: Rimini
What is the name of your State or Province?
  [Unknown]: Italy
What is the two-letter country code for this unit?
  [Unknown]: IT
Is CN=giulia.costa, OU=localhost, O=Development, L=Rimini, ST=Italy, C=IT correct?
  [no]: yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA384withRSA) with a validity of 365 days
for: CN=giulia.costa, OU=localhost, O=Development, L=Rimini, ST=Italy, C=IT
```

2. COPIA DEL CERTIFICATO NEL PROGETTO DI SPRING BOOT (all'interno della cartella KEYSTORE)

3. ABILITAZIONE SSL PER SPRING BOOT:

Configurazione del certificato sulle proprietà del progetto.

⁴ L'handshake, in informatica, si riferisce a una serie di scambi di messaggi tra due entità (come client e server) al fine di stabilire una connessione sicura e stabilita tra di loro. È un processo fondamentale utilizzato in molti protocolli di comunicazione, inclusi quelli utilizzati su Internet come HTTPS, TLS/SSL, SSH e molti altri.

```
server.port=443

server.servlet.context-path= /Chatmessage
# configurazione necessaria per accettare richieste HTTPS
server.ssl.enabled=true
# The alias mapped to the certificate
server.ssl.key-alias=local-ssl
# The path to the keystore containing the certificate
server.ssl.key-store=classpath:keystore/local-ssl.p12
# The format used for the keystore.
server.ssl.key-store-type=PKCS12
# The password used to generate the certificate
server.ssl.key-store-password=password
# password archivio chiavi, sono le stesse
server.ssl.key-password=password
```

4. INSTALLAZIONE DEL CERTIFICATO SELF-SIGNED SUL PC PERCHÉ NON TRACCIATO DAL BROWSER

Prompt (comando per esportare il certificato del file keystore):

```
PS C:\Program Files\Java\jdk-19\bin\SSL\self-signed> keytool -exportcert -keystore local-ssl.p12 -alias local_ssl -file local-cert.cer
Enter keystore password:
Certificate stored in file <local-cert.cer>
```

Grazie all'importazione della dipendenza SPRING-BOOT-STARTER-SECURITY è stato possibile implementare HSTS, ovvero un meccanismo di sicurezza il cui obiettivo è quello di proteggere le comunicazioni utilizzando una connessione HTTPS sicura, escludendo qualsiasi comunicazione tramite protocollo HTTP non crittografato.

La configurazione HSTS prevede l'inserimento di un'intestazione HTTP specifica chiamata "Strict-Transport-Security", che comprende la direttiva obbligatoria max-age e può essere ampliato con le direttive opzionali "includeSubDomains".

Nella direttiva "max-age" viene indicato il tempo per cui un sito deve rimanere crittografato, utilizzando quindi il certificato SSL precedentemente abilitato (31536000 corrisponde ad un anno).

"IncludeSubDomains", segnala invece al browser come l'header HSTS non valga solo per l'host attuale (ad esempio www.example.com), ma anche per tutti i sottodomini del rispettivo dominio (ad esempio anche per blog.example.com o adserver.example.com).

HSTS nasce per prevenire gli attacchi dirottamento degli utenti, noti anche come attacchi "Man-in-the-Middle" (MITM), che possono avvenire tramite reindirizzamenti non sicuri o collegamenti non sicuri.


PROTOCOLLO HSTS installato:

The screenshot shows a web browser window with the address bar displaying 'localhost/chatmessage/login'. The page content includes a 'Refresh' button, a 'Log In' heading, and input fields for 'Username' and 'Password'. Below the password field is a 'Log In' button and a link that says 'Non sei ancora iscritto? Iscriviti'. An 'HTTP Header Spy' tool is open on the right side of the browser, showing the details of a GET request to 'https://localhost/chatmessage/login'. The tool displays the response headers, including 'Strict-Transport-Security: max-age=31536000; includeSubDomains', which confirms that HSTS is installed and configured correctly.

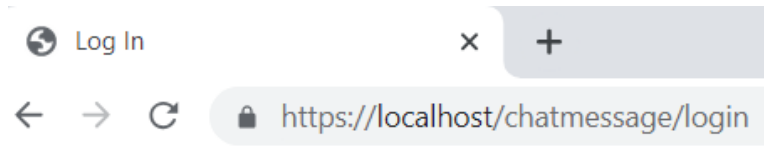
Request	
Method	GET
URL	https://localhost/chatmessage/login
Protocol	HTTP/1.1
Status	200
Time	21 ms
Response	
X-Content-Type-Options	nosniff
X-XSS-Protection	1; mode=block
Cache-Control	no-cache, no-store, max-age=0, must-revalidate
Pragma	no-cache
Expires	0
Strict-Transport-Security	max-age=31536000; includeSubDomains
X-Frame-Options	DENY
Content-Type	text/html; charset=UTF-8
Content-Language	it-IT
Transfer-Encoding	chunked
Date	Tue, 20 Jun 2023 20:29:26 GMT
Keep-Alive	timeout=60
Connection	keep-alive

STRUTTURA APPLICATIVA

Inserendo l'URL senza l'intestazione HTTPS,

 localhost/chatmessage/login

viene comunque caricato l'ambiente HTTPS:



UTENTE UTILIZZATO PER I VARI TEST:

Registrazione

Username

Password

Sei stato registrato. [Inizia](#)

1. ACCESSO CON DATI NON ANCORA REGISTRATI:

Refresh

Log In



Attenzione! Nome utente o password non validi!

Non sei ancora iscritto? [Iscriviti](#)

2. CREAZIONE DI UN UTENTE

✓ Iscrizione avvenuta con successo!

Registrazione

Username

Password

Sei stato registrato. [Inizia](#)

E RELATIVO INSERIMENTO IN DATABASE:

Al fine di consentire la gestione del login, i dati, vengono sincronizzati e salvati all'interno di un database (MySQL); tale operazione consente di memorizzare, garantendo il riconoscimento in caso di nuova login, le credenziali, ereditandone anche le caratteristiche di sicurezza, come l'encrypting della password.

Query 1

Limit to 1000 rows

1 • SELECT * FROM user_entity;

2

3

Result Grid

Filter Rows:

Edit: Export/Import: Wrap Cell Co

	id	email	password
▶	17	giuliacosta93@gmail.com	\$2a\$10\$xcrrfigh3cAQZRwzUBb6xuB.j3/p7k9VEzpB53FFxFKyZuRqnpd32
	15	sara.leoni@hotmail.com	\$2a\$10\$mma22UXlBBVfmh2xr7XreEboRRfOO2ULzG8rp1xrVWHCbKWAXd6G
*		NULL	NULL

3. TENTATIVO DI CREAZIONE NUOVO UTENTE GIA' ESISTENTE

localhost/chatmessage/new/error

! Attenzione! Email già esistente.

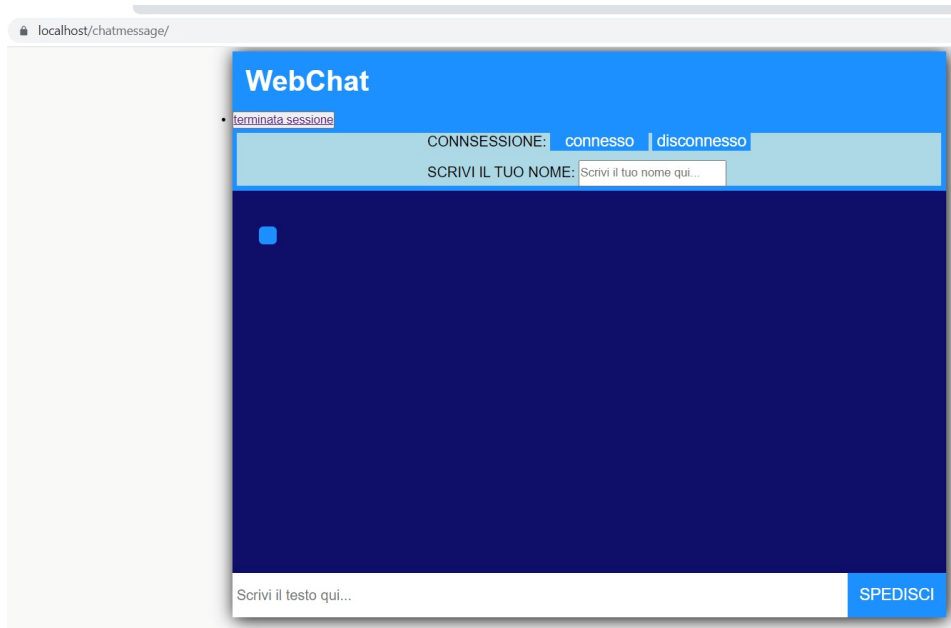
Registrazione

Username

Password

Sei stato registrato. [Inizia](#)

4. LOGIN



ENCODING DEL LOGIN

L'algoritmo di hashing che è stato utilizzato per l'encoding delle password è BCrypt.

Stiamo parlando di un algoritmo di hashing adattativo progettato per la crittografia delle password, basata sull'algoritmo di crittografia Blowfish.

Parliamo quindi dell'utilizzo di un algoritmo di crittografia sicuro e solido, che si avvale di una crittografia a chiave simmetrica, ovvero la stessa chiave viene utilizzata sia per la crittografia che per la decrittografia dei dati.

L'obiettivo è quindi migliorare la sicurezza dell'applicazione, rendendo così più difficile recuperare le password originali memorizzando direttamente sul database quelle criptate.

```
// creo codificatore di password
@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

PROTOCOLLO DI COMUNICAZIONE PER LA MESSAGGISTICA

Per gestire la comunicazione all'interno della chat sono state utilizzate le WebSocket, parliamo di un protocollo utilizzato a livello di trasporto per la comunicazione bidirezionale. Questo consente una connessione interattiva in tempo reale tra un client e un server attraverso il web.

Viene utilizzato un protocollo basato su TCP per stabilire una connessione persistente tra client e server. Una volta stabilita la connessione, sia il client che il server possono inviare messaggi l'uno all'altro in modo asincrono, senza la necessità di una richiesta esplicita da parte del client.

Prima di tutto sarà necessario configurare un broker di messaggi, necessario per l'instradamento dei dati tra i client.

```
// configurazione del broker di messaggi:
// viene abilitato un broker necessario per riportare i messaggi al client
// definendo un prefisso utilizzato come mapping
@Override
public void configureMessageBroker(MessageBrokerRegistry config) {
    config.enableSimpleBroker("/topic");
    config.setApplicationDestinationPrefixes("/app");
}

// viene abilitato l'opzione di fallback(ripiego) di SockJS, per trasporti alternativi
// nel caso in cui le websocket non siano disponibili.
// Ovvero SockJS tenterà di connettersi e utilizzare il miglior trasporto disponibile
@Override
public void registerStompEndpoints(StompEndpointRegistry registry) {
    registry.addEndpoint("/gs-guide-websocket").withSockJS();
}
```

È necessario definire il controller per la gestione dei messaggi, la cui logica sarà gestita direttamente nel file "app.js"

```
@Controller
public class MessageController {

    @RequestMapping("/input")
    @SendTo("/topic/output")
    public MessageOutput messages(Message output) throws Exception {
        Thread.sleep(1000);
        return new MessageOutput(HtmlUtils.htmlEscape(output.getName() + ": " + output.getTextMex()));
    }
}
```

CONSIDERAZIONI E CONCLUSIONI

Nel complesso, pur non avendo effettuato il deploy, si denotano i seguenti aspetti tecnici:

1. Utilizzo di tecnologie moderne: Il progetto fa uso di tecnologie moderne come Spring Boot, Maven e Tomcat, che semplificano lo sviluppo di applicazioni web in Java. Queste tecnologie offrono una vasta gamma di funzionalità e librerie per la gestione delle dipendenze, la sicurezza, la gestione delle risorse statiche e altro ancora.
2. Sicurezza dei dati: L'applicazione si preoccupa di garantire la sicurezza dei dati sensibili, come le password degli utenti. Utilizzando l'algoritmo di hashing BCrypt, le password vengono crittografate prima di essere memorizzate nel database. Questo aumenta la sicurezza, poiché le password originali non sono direttamente accessibili anche in caso di compromissione del database.
3. Connessioni sicure: Il progetto prevede l'utilizzo del protocollo HTTPS per consentire solo connessioni sicure tra il client e il server. Viene generato un certificato SSL autofirmato utilizzando Java Keytool e configurato su Spring Boot per abilitare il supporto SSL/TLS. Ciò garantisce che le comunicazioni tra il client e il server siano crittografate e proteggano la riservatezza delle informazioni.
4. Politica di sicurezza HSTS: Viene implementata la politica di sicurezza HSTS (Strict-Transport-Security) per assicurare che il sito web venga sempre fruito tramite HTTPS. Questo protegge gli utenti da attacchi di dirottamento e garantisce che tutte le comunicazioni avvengano attraverso una connessione crittografata.
5. Utilizzo di WebSockets: Per la gestione della chat, vengono utilizzate le WebSocket, consentendo una comunicazione bidirezionale in tempo reale tra il client e il server. Questo permette agli utenti di inviare e ricevere messaggi istantaneamente senza dover aggiornare la pagina.

Il progetto proposto mira a creare un'applicazione di chat che garantisca la sicurezza delle comunicazioni e la protezione dei dati sensibili. Attraverso l'uso di tecnologie sopra citate viene fornita un'architettura solida per lo sviluppo dell'applicazione.

L'implementazione dei parametri di sicurezza garantiscono la riservatezza delle comunicazioni, contribuendo a garantire un'esperienza sicura e interattiva per gli utenti.

Complessivamente, il progetto si concentra nel garantire una buona attenzione alla sicurezza delle comunicazioni e alla protezione dei dati sensibili, avvalendosi di tecnologie specifiche per fornire un'applicazione affidabile e sicura.