Università di Camerino

"CHEAT" - APPLICAZIONI WEB, MOBILE E CLOUD

Indice

Frontini Alessandro | Mat. 118532 | A.A. 2023/2024

1. Introduzione

Cheat è un'applicazione web responsive sviluppata in Angular 18 e Spring Boot 3.3.3, avente un DB MySQL su cui salvare i dati.

In termini di utilizzo, Cheat è un ricettario online in cui è possibile cercare ricette in base al budget e tempo a disposizione; se si effettua l'accesso alla piattaforma è possibile aggiungere nuove ricette ed eliminare quelle inserite in precedenza.

2. Backend

Il backend è stato realizzato utilizzando Java 17 LTS e Spring Boot 3.3.3, e si compone di: Repository Spring per le operazioni sul database, Servizi in grado di comunicare con i repository contenenti dei metodi di appoggio, Controller per l'intercettazione delle richieste comunicanti con i servizi, un Componente Spring contenente la Content Security Policy e le classi rappresentanti il Model dell'applicazione.

2.1. Protezione dalle SQLi

La protezione contro le SQL Injection è fornita automaticamente grazie a Spring Boot e al modo in cui vengono strutturate le chiamate nei Repository.

Le classi presentano i decoratori "@Entity" e "@Table(name = "<>")", i quali collegano le classi al nome della tabella inserito come argomento del decoratore "@Table", in modo da creare un'associazione. Allo stesso modo, sono stati mappati anche i singoli attributi della classe, attraverso il decoratore "@Column(name="<>")", i quali garantiscono una corrispondenza tra il nome dell'attributo della classe e il nome della colonna della tabella precedentemente specificata; in caso la corrispondenza non fosse presente, a runtime viene creata la colonna con il nome deciso all'interno della classe.

I Repository creati estendono l'interfaccia "JpaRepository", in cui è necessario specificare la classe di riferimento e il tipo di variabile della chiave primaria; nel caso della classe "Ricetta", il repository "RicetteRepo" presenta questa dicitura.

```
@Repository
public interface RicetteRepo extends JpaRepository<Ricetta, Long> {
```

Grazie a questa inizializzazione è stato possibile definire dei metodi in grado di effettuare ricerche nel DB senza specificare la query: attraverso la semantica generale "get<Classe>By<Attributi>" metodi come il seguente non hanno

richiesto implementazione. In questo caso la semantica prevede di utilizzare il plurale inglese aggiungendo la “s” nel caso in cui siano previste liste di oggetti.

```
Optional<List<Ricetta>> getRicettasByIdUtente(long idUtente);
```

2.2. JWT e Login

La gestione del JWT è stata delegata ad un servizio, in grado di effettuare operazioni di creazione, validazione ed estrazione dei dati. Il secret e il tempo di expiration del JWT sono stati salvati nel file “application.properties”, permettendone la modifica più facilmente. Il Payload del JWT contiene l’ID dell’utente, la data di emissione e la data di scadenza.

Il login viene effettuato tramite una richiesta POST in cui, a seguito di un Sanitize sui valori ricevuti, viene effettuata la richiesta al DB tramite Repository. In caso di esito positivo viene creato il JWT e viene salvato in un cookie che viene trasmesso nella risposta; in caso di esito negativo viene generato un errore 401.

Il cookie in questione presenta le diciture “httpOnly(true)” per renderlo accessibile soltanto tramite HTTP e HTTPS, aumentando la prevenzione contro gli attacchi XSS, e “sameSite(“strict”)”, che permette l’invio del cookie soltanto dal sito che l’ha impostato; in questo modo si prevengono gli attacchi CSRF, in quanto il JWT non verrebbe validato.

2.3. Registrazione e Salt

All’interno del servizio di Login sono presenti un metodo di registrazione, un metodo di generazione di un salt e un metodo di hashing della password. Quando un nuovo utente si registra nella piattaforma, un nuovo salt viene generato, successivamente viene concatenato alla password e hashato in MD5. Infine, la password hashata viene codificata in Base64 per una maggior facilità di salvataggio nel database. In seguito a questi passaggi, l’utente viene salvato nel DB.

2.4. Protezione da XSS

La protezione da XSS avviene tramite un servizio di sanificazione degli input, che rimuove tag HTML in grado di generare script inline indesiderati; inoltre, Angular stesso tratta le risposte del backend come testo semplice; quindi, eventuali tag HTML non verrebbero processati, ma semplicemente mostrati.

Inoltre, è presente una Content Security Policy sottoforma di un Componente Spring che non permette l’esecuzione di script da fonti esterne al dominio dell’applicazione e Bootstrap, bloccando di default l’esecuzione di script inline.

2.5. CORS

Di default i CORS sono disabilitati, ma, in caso si presentasse la necessità di effettuare l'embed di Cheat in un altro sito, sarà possibile accedere soltanto alla ricerca delle ricette, la cui richiesta non richiede la validazione del JWT. Pertanto, all'interno del controller è presente il decoratore "@Crossorigin" solo per quel metodo.

3. Frontend

Il frontend è realizzato in Angular 18, ed è responsive grazie alla presenza di Bootstrap.

Il progetto frontend si compone di componenti Angular, Servizi per le chiamate al backend e classi. Tramite un Proxy, l'applicazione presenta un punto di accesso comune per le richieste, evitando problemi di CORS; inoltre, tramite gli Environment, l'applicazione presenta la possibilità di cambiare configurazioni nel passaggio da "sviluppo" a "produzione".

L'applicazione si compone di una navbar in cui è possibile raggiungere le varie rotte; alcune funzionalità richiedono il login, pertanto, in caso l'utente non fosse autenticato, viene reindirizzato alla schermata di login o registrazione.

L'applicazione è fruibile dai dispositivi mobili grazie all'utilizzo di Bootstrap; sfruttando il layout a griglia, le schermate hanno una larghezza di 12 colonne in modalità desktop e 6 colonne in modalità mobile.