1. Allege o tema de care esti mandru si vorbeste despre ea.

Aplicatia java pentru un group chat, facuta cu spring boot, security, jpa, jsp, queryuri.

Are register, login, vizualizarea utilizatorilor, criptarea parolelor, crearea de camera de chat, trimiterea de mesaje in camera.

Am facut configurarile doar prin cod java ca sa evit xml ul, parea mai greu.

Intr o clasa de configurare cream entitymanager ul pentru proiect, pe care l folosesc prin layer ul de service-uri ca sa fac operatii pe baza de date. Tot in clasa asta specificam si sursa bazei de date adica url-ul jdbc, userul si parola si driverul adica postgresql.

Am folosit bcrypt pentru encodat parole si am implementat un basic authentication.

Pentru partea de security aveam o clasa de configurare care extinde websecurityconfigureradapter, in asta specificam pt diverse endpoint-uri daca pot fi accesate de oricine sau daca trebuie sa fie logat si cu un anumit rol. Tot acolo specificam si pagina de login, logout [endpoint-urile pe care ar trebui sa se duca].

Am incercat sa fac o arhitectura MVC, aveam controllerele intr-un package si acolo tratam endpoint-urile, view-urile separate ca jsp-uri si modelele in alt package ca niste clase pojo cu datele in sine/ entitatile/ tabelele.

Am avut unele controllere normale, care se ocupau de datele afisate in jsp, si unele tip RestController care doar returnau JSON sau faceau operatii pe baza de date si ma informau daca au avut success acele operatii.

Am implementat metode tip GetMapping, PostMapping mai mult, nu am insistat pe Put care ar fi pt update si DeleteMapping.

2. Ceva citit

m-am documentat recent legat de thymeleaf, am mai avut o tentative sa-l integrez intr-o aplicatie dar era prea mare presiunea de timp pt predarea proiectului. Pare mai clean decat jsp.

3. Baza de date

Baza de date avea tabele pentru user, room, message la group chat.

La munca folosim baze de date mai mari, mai multe tabele si cu mai multe coloane. Trebuie sa fac scripturi pe ele , incerc sa ma axez pe rezolvarea din sql daca nu este necesar sa ma complic si cu prelucrari in java.

4. Impotmolit la o problema

Cand incercam sa configurez spring security a durat mai mult.

Am inceput sa fac research, sa caut tutoriale vreo 2 zile, sa le combin. Am cautat unele erori sau buguri pe stackoverflow, daca ajungeam la un bug il rezolvam intr-o seara adica vreo 4h maxim. Daca tot nu, mai ceream parerea unor colegi si imi mai dadeau niste directii.

5. Provocari

- Am avut o problema cu niste id-uri generate din clasa model, am gasit alternativa de a face interogare de insert separat care sa preia ultimul id din tabela si sa foloseasca urmatorul id. O executam cu un criteria query la care dadeam argumente pentru a seta valorile campurilor cu date pt fiecare coloana.
- Incercam sa compar niste parole criptate cu bcrypt dar am realizat ca mai multe criptari nu dau ca rezultat acelasi string si a trebuit sa folosec o metoda din bcrypt, matches() care facea verificarea asta pt mine. Specificam parola pe care o asteptam, criptata si o tentativa de parola
- Configurarile sunt uneori greoaie, mai ales ca daca faci niste research pe spring gasesti multe versiuni diferite, dar daca fac din cod java partea de configurare devine mai simplu, sunt toate mai adunate
- Voiam sa fac niste celule dintr-un tabel sa devina editabile la click pe un buton. Am pus span si input in acel loc si faceam toggle care sa devina vizibil din js.

6. SQL

a. Diferenta delete, truncate, drop

Delete sterge randuri din tabela, truncate sterge toate randurile, drop sterge tabelul din baza de date

b. Cum implementezi many to many

Folosesc o tabela intermediara ca sa sparg relatia many to many in 2 relatii one to many. Tabela intermediara are combinatii de iduri din cele 2 tabele initiale care aveau many to many intre ele.

c. Inner, left, right join

> `INNER` join compares two tables and only returns results where a match exists. Records from the 1st table are duplicated when they match multiple results in the 2nd. INNER joins tend to make result sets smaller, but because records can be duplicated this isn't guaranteed.
> `CROSS` join compares two tables and return every possible combination of rows from both tables. You can get a lot of results from this kind of join that might not even be meaningful, so use with caution.
> `OUTER` join compares two tables and returns data when a match is available or NULL values otherwise. Like with INNER join, this will duplicate rows in the one table when it matches multiple records in the other table. OUTER joins tend to make result sets larger, because they won't by themselves remove any records from the set. You must also qualify an OUTER join to determine when and where to add the NULL values:
>
> > `LEFT` means keep all records from the 1st table no matter what and insert NULL values when the 2nd table doesn't match.

`RIGHT` means the opposite: keep all records from the 2nd table no matter what and insert NULL values whent he 1st table doesn't match.
`FULL` means keep all records from both tables, and insert a NULL value in either table if there is no match.

d. Sters liniile duplicate din tabela

```
   delete from EmpDup where EmpID in(select EmpID from EmpDup group by EmpId
having      count(*) >1)
```

e. Diferenta where, having

- Where se poate folosi in update si delete, having doar cu select
- Where filtreaza randuri, having filtreaza grupuri
- Where se scrie inainte de group by , having e dupa group by

7. Functii de agregare

- Count, avg, min, max, sum

8. View

- Tabel virtual , bazat pe rezultatele de la o interogare, are campuri din una sau mai multe tabele ale bazei de date

9. Ce e o baza de date

- Un set de date cu o structura