Tabelle → classi con @entity e @table per poterle mappare nel database. @id mappa la chiave primaria @GeneratedValue(strategy = GenerationType.Identity) setta l'autoincremento

file application.properties permette di impostare le info per connettersi al db (cambia in base al database). Con la clausola ps.hibernate.ddl-auto si imposta il comportamento all'avvio

Jpa repository<T, S> → interfaccia da estendere da un'altra interfaccia

- classe tabella
- tipo chiave primaria

Service → sono le effettive route che costruiscono l'API

@RestController → gestisce richieste
@Service → identifica un servizio

] messi sulla classe

sfrutta le repository che abbiamo creato prima con una istanza final. Per esempio se abbiamo UserTableRepo extend JpaRepository<> in UserTableService avrà:

- private final UserTableRepo
- costruttore con

Annotation @RestController

Operazioni Crud

Get : find() o simili
return list<tabella>
return tabella
può usare @PathVariable

Delette : delete()
void
usa @PathVariable

Put : misto
1 @pathVariabile + @RequestBody
2 getta e setta
3 return save()

Post : save()
return nuova entry
usa @RequestBody

@PathVariable => prende un parametro dall'url

api/user/{id} → api/user/2
server client

@RequestBody => ottenere informazioni tramite un tipo di dato specifico, come il Json