1) Chno drna f TP REST (l’ancien) ?

F TP REST drna :

✔ Controller → kaysemma b7al l-portier

kaysift GET/POST/PUT/DELETE.

✔ Service → l-logique métier

kayverifi :

- wach ID kayn
- wach données sahh
- kayconverti BO ↔ DTO
- kaydir modification w kayrja3 réponse

✔ DAO → pseudo-base de données

wane liste statique.

✔ DTO / BO / Converter

bach ma n3tich data interne direct.

✔ ExceptionHandler

kaydwi m3a client f cas d’erreur.

✔ Tests Postman

kankhayrou GET / POST / PUT / DELETE.

✔ Tests JUnit

MockMvc + RestTemplate.

2) Chno drna f TP GraphQL (jadid)

F TP GraphQL drna système bancaire : users, clients, comptes, virements.

✔ Au lieu dyal bzzaf dyal endpoints REST...

GraphQL عندو *endpoint wa7ed* :

```
POST /graphql
```

✔ Au lieu dyal GET/POST/PUT/DELETE...

GraphQL عندo :

- **Query** → bach tjib data
- **Mutation** → bach tgayer data

✔ Au lieu dyal controller GET/POST...

f GraphQL kayn **resolver** (QueryMapping, MutationMapping).

✔ Au lieu dyal endpoints kayhenw...

f GraphQL nta katsift **requête exacte** li bghiti.

✔ Schema.graphqls

Hadchi li kaybni l'application :
kol types, kol inputs, kol queries, kol mutations.

✔ ModelMapper

yconverti BO ↔ DTO automatiquement.

✔ Exceptions GraphQL

kayn handler spécial kayrja3 erreur format GraphQL.

✔ GraphiQL

Interface interactive katéxécuti query o mutation.

3) Fark kbiiiiir ben TP REST o TP GraphQL (le plus important)

🔘 REST = routes fixes

Example :
GET /customers
GET /customers/1
POST /customers
DELETE /customers/5

Le client khaso yemchi mn route l route.

Darija : route b route, mlli bghiti data kteb endpoint.

🔘 GraphQL = une seule route + nta katsift exactement chno bghiti

Exemple :

```
query {
  customers {
    firstname
    lastname
  }
}
```

Machi b7al REST li kayrja3 kolshi, m3a GraphQL :

➡ **nta kat9elleb o katgoul b'dabt : "3tini ghiiiiir firstname w lastname"**.

Darija :
CLIENT howa li kaymchi direct l data li bghaha b dabbt, ma kaystnach server ydir des routes.

🔘 REST → over-fetching / under-fetching

Exemple :
GET /customer/1
→ kayrja3 bzzaf dyal data li ma bghitsh.

GraphQL **ma kayrja3 ghiiiir data li tlebt** :

```
query {
  customerByIdentity(identity:"A100") {
    firstname
```

```
  }
}
```

Darija :
GraphQL b7al had lcafena li katsawl “bghiti sucre wla bla sucre ?”
nta b’dabt kat9elleb l data.

🔘 REST = Controller

🔘 GraphQL = Resolver (Query/Mutation)

Controller REST kaydir :

```
@GetMapping("/customers")
```

GraphQL kaydir :

```
@QueryMapping
public List<CustomerDto> customers() { ... }
```

Darija :
GraphQL kayb9a f logique dial "chno bghiti ?" machi "had endpoint shno kaydir".

🔘 REST retourne JSON normal

🔘 GraphQL retourne JSON formaté suivant ta requête

4) Workflow d’une requête REST vs une requête GraphQL

Workflow REST (ancien TP)

Client
↓ GET /articles
Controller
↓ service.findAll()
Service
↓ dao.findAll()
DAO
↓ liste articles
Service
↓ convert DTO
Controller
↓ JSON
Client

Darija :
route → controller → service → dao → service → controller → json.

Workflow GraphQL (TP 4)

Client (GraphiQL)
↓ *POST /graphql*
↓ *Requête écrite par le client*
↓ **Schema.graphqls**
↓ matching du champ demandé
↓ **Resolver (QueryMapping/Mutation)**
↓ Service métier
↓ Repository
↓ Service
↓ DTO
↓ **Résultat EXACT demandé**
Client

Darija :
nta katsift l script dial GraphQL → schema kayfham chno bghiti → resolver kayexécuti → service → dao → service → t7ssb → tkonverti → resultat EXACT.

5) Les points ESSENTIELS que tu dois retenir (super important)

✔ 1) GraphQL utilise un seul endpoint

Le client est plus libre.

✔ 2) Le client décide quelle data et combien ybgha.

REST → serveur li kay9elleb
GraphQL → client li kay9elleb

✔ 3) REST → plusieurs endpoints

GraphQL → Query + Mutation

✔ 4) REST → JSON fixe

GraphQL → JSON personnalisé

✔ 5) REST → architecture simple mais parfois lourde

GraphQL → architecture flexible w performant

✔ 6) Workflow très différent

GraphQL kaymchi mn schema → resolver → service

✔ 7) GraphQL parfait pour :

- mobile apps
- front-end li bghaw data exacte
- systèmes complexes

REST parfait pour :

- services classiques
- microservices
- API standard

6) Résumé en darija très simple bach tsift l idée d’un coup :

👉 F TP REST

- kayn bzzaf dyal endpoints
- kol endpoint kayrja3 data
- nta katkhsi tchd li bghiti
- logique dial client s7i7a

👉 F TP GraphQL

- kayn endpoint wa7ed
- nta kat9elleb b’dabt chno bghiti
- serveur kayrja3 ghiiir dakshi
- workflow kaymchi mn schema → resolver

👉 **REST = serveur kay9elleb chno kayn.**
👉 **GraphQL = client kay9elleb chno bgha**