1. Who and how reads data from your API?

- Your **API is a middleman** — it sits between your database (DB) and whoever wants data (clients).
- Clients could be:
    - A **web frontend** (like a React or Angular app in a browser)
    - A **mobile app** (iOS, Android)
    - Another **server or backend service**
    - Even **third-party systems**

How it works:

- The client sends a request to your API endpoint (e.g., `GET /users` or `POST /login`).
- Your API runs some code, queries the DB, transforms the data (models → DTOs), and sends back JSON (or XML, etc.).
- The client reads that response and shows data or acts accordingly.

2. Is the API an alternative to your DB?

- **Sort of, yes.**
- Think of the API as a *controlled gateway* or *driver* that accesses your DB on behalf of clients.
- But clients **never connect directly to the DB** — that would be insecure and fragile.
- API **controls what data goes out and what comes in**, enforces security, business rules, validation, logging, etc.

3. Why shouldn't the API send the password hash to clients?

- Password hashes are **sensitive info**, just like the original password.
- If someone intercepts that data or finds a way to read the app data, they get your hash, which they can try to crack offline.
- Even if you hash passwords, hashes **should stay private in your backend** — only the API backend needs them.

4. How does login work if the app never sees the hash?

- When a user logs in, the **mobile app collects the user's input password** (plain text).
- The app sends the **username/email and the plain password** to your API's login endpoint over HTTPS (encrypted).
- The API backend:
    - Retrieves the **stored hashed password** from the DB (hidden from clients).
    - **Hashes the password sent by the user** (using the same hashing method + salt).
    - Compares the newly hashed password with the stored hash.
- If they match, login succeeds and the API returns an auth token (like JWT), which the app stores and sends with future requests.

5. What you should never do

- **Never send the password hash to the client.**
- **Never let the client handle password hashing or validation.**
  Let the backend do that because it's trusted and secure.
- **Never expose your DB directly to clients.**

Recap:

- **DTOs:** safe subset of data API sends out.
- **API:** secure middleman, controls DB access.
- **Clients:** web/mobile apps talk only to API, not DB.
- **Passwords:** hash stored privately on backend; client sends plain password securely; backend compares hashes.