

Database Applications

Building the Application Layer

The *application layer* is what sits between the user and the database – the back end.

- built as an API – any client can talk to it in the same way

- CRUD
- REST conventions
- JDBC fundamentals
- DAO layer, business logic, tiers, JSON, and routing
- application-level authentication and authorization
- error handling

Library

- a user clicks “check out this book”

What happens between that click and the database?

- the request arrives
- get the information from the request: identify the borrower and the book
 - is the request well-formed?
- determine if the book is available
- determine if the borrower is allowed to check out the book
- create the loan
- update the book’s availability
- respond with success or failure
 - all or nothing – all steps must succeed or fail together
 - two borrowers clicking at the same time can’t both get the same book

CRUD

- the four basic operations of persistent storage

operation	what it does	SQL
create	add a new row	INSERT
read	retrieve existing rows	SELECT
update	change existing rows	UPDATE
delete	remove rows	DELETE

Decomposing Features

Check out a book –

- *read* the book – does it exist? is it available?
- *create* a loan
- *update* the book's availability

Return a book –

- *read* the loan – does it exist? is it yours? has the book already been returned?
- *update* the loan – book has been returned
- *update* the book's availability

From CRUD to an API

Why not let clients talk to the database directly?

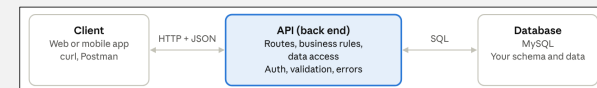
- trust – security and integrity depend on every client
 - database credentials would ship inside every client
 - business rules (“no checking out a book that’s already out”) hold only if every client enforces them correctly
- duplication and fragility – change is painful
 - the same business logic is rewritten in every client
 - every client depends on the database schema – rename a column and they all break

From CRUD to an API

An API provides –

- a contract – clients know what they can ask for, not how it is stored
- a stable external view of the database schema
- a single place for business logic
- a single place for authentication, validation, and error handling
- trust moves to code you control
- fragility is concentrated, not eliminated
- “single place” means the authoritative place
 - clients may repeat some validation for a better user experience, but the system no longer requires every client to get it right

Application Structure



Three tiers –

- client – web or mobile app, curl, Postman, ...
- API / back end – routes, business rules, data access
- database
- this course addresses only the API / back end and database tiers