— ACT III | INTEGRATION -- WE TURN THE CORNER

The server is built. Now we build the app that consumes it.

For seven chapters we lived inside HAPI -- architecture, profiles, interceptors, search, partitions. From here we build the application layer, starting with a doorway into FHIR -- the Backend-for-Frontend.

- DTO map
- Retry
- Idempotency
- Error translate

— CONCEPT | BACKEND-FOR-FRONTEND -- ONE BACKEND PER USER EXPERIENCE

The browser never calls HAPI. ddx-api does -- and only ddx-api.

The single legal path to FHIR

A server-side facade built for one frontend's needs (Sam Newman)

↓

↓

3 **HAPI FHIR | :18080**
The only party ddx-api talks to downstream

— FORBIDDEN EDGE

Browser --[X]--> HAPI :18080

A direct browser-to-HAPI arrow is drawn struck through. That edge never exists. We meet the UI side of this same rule next chapter.

— DISCIPLINE ONE | FHIR → DTO -- DON'T LEAK FHIR TO THE UI

A full FHIR Patient goes in. Four screen fields come out.

— THE PROBLEM | RAW FHIR PATIENT

{ "resourceType": "Patient", "meta": { ... }, "identifier": [{ "system": "...", "value": "MRN-8842" }], "name": [{ "use": "official", "family": "...", "given": [...] }], "extension": [...] }
Nested CodeableConcept and Reference, a meta block, extensions, polymorphic value-x. Verbose, tenant-revealing, version-coupled.

— THE DISCIPLINE | PATIENTSUMMARYDTO

class PatientSummaryDto { id: string; displayName: string; birthDate: string; mrn: string; }
class-transformer and class-validator behind a ValidationPipe. One-way, explicit. When a FHIR version or profile changes, the blast radius is one file: the mapper.

Ava

Never. That is discipline one -- map it. The UI never depends on the FHIR wire shape.

Marcus | expert

Containment. Profiles and versions can shift downstream -- nothing upstream in the UI moves.

— DISCIPLINE TWO | RETRY WITH BACKOFF -- BOUNDED, NEVER BLIND

5xx is transient -- retry it. 4xx is your fault -- never retry it.

Bounded retry on HAPI 5xx

RxJS retry({ count, delay }) on the @nestjs/axios HttpService

- GET / PUT / DELETE -- idempotent, retry freely
- POST -- retry ONLY when guarded by an idempotency key

— DISCIPLINE THREE | IDEMPOTENT CREATE -- THE DESIGN SPACE, TWO WAYS

A naive POST retry creates two Patients. Two ways to stop that.

— OPTION A | CLIENT IDEMPOTENCY-KEY

POST /api/v1/patients Idempotency-Key: 7f3c-...
→ BFF stores (key → resource id) → replay returns the ORIGINAL result
App-level. An IETF draft standardizes the header. The BFF dedups before HAPI ever sees a duplicate.

— OPTION B | FHIR-NATIVE IF-NONE-EXIST

POST [base]/Patient If-None-Exist: identifier=mrn|123 → HAPI conditional create → match exists? no-op, returns 200 not 201
Server-level. HAPI itself enforces the dedup as part of R4 conditional create semantics.

Ava

That you confirm in the code -- it is verified against the live server at build time, not asserted on a slide.

Marcus | expert

The BFF exposes one idempotent create to the UI -- and translates it to one of these downstream.

— DISCIPLINE FOUR | OPERATIONOUTCOME → TYPED ERROR -- TRANSLATE, DON'T FORWARD

HAPI says no in FHIR. The UI hears it in clean HTTP.

— PARSE VIA EXCEPTION FILTER → CLEAN BODY

{ "statusCode": 404, "message": "Patient not found", "code": "not-found" }

An OperationOutcome is a list of issues -- each with a severity, a code, and diagnostics. The BFF never forwards that verbose, internal-leaking payload. It throws a typed NestJS exception through a filter.

- NOT-FOUND
  404 NotFound
  Resource id absent
- INVALID
  400 / 422
  Bad or Unprocessable -- exact status verified live
- SECURITY
  403 Forbidden
  Auth / scope denial
- PROCESSING
  502 BadGateway
  Upstream failure

The typical mapping above is the discipline. The exact issue.code → HTTP status HAPI emits is implementation-specific -- whether a failed validate returns 422 or 400 you confirm against the live :18080 at build time. The discipline is fixed; the table you verify.

— PROPAGATION | WHO THE BFF IS WHEN IT CALLS HAPI

Two headers, injected once -- never plumbed call by call.

Central injection on the shared HttpService

An axios request interceptor reads from request-scoped context

↓

↓

3 **Authorization: Bearer <service token>**
The BFF's own downstream credential

4 **X-Clinic-ID: <tenant slug>**
Drives HAPI's partition interceptor -- slug → partitionId, from the multi-tenant chapter

— THE TRUST RULE

Never trust a client-supplied X-Clinic-ID

Derive and validate the tenant from the verified token. The end-user login itself -- SMART and OIDC -- is a later chapter; here we carry only the downstream service identity.

— CODE | THE WRAPPER WHERE ALL FOUR DISCIPLINES LIVE

One seam every HAPI call flows through. The FHIR mess stops here.

— ILLUSTRATIVE SKELETON -- NOT PRODUCTION SOURCE

```
@Injectable() class FhirGateway {  constructor(private http: HttpService) {}  async getPatient(id:
string): Promise<PatientSummaryDto> {    return firstValueFrom(
    this.http.get(`/Patient/${id}`).pipe(      retry({ count: 3, delay: backoffWithJitter }),
      map(r => toPatientSummaryDto(r.data)),    ),    );  } }
```

- Interceptor stamps the two headers -- on the way out
- retry guards transient 5xx -- on the way out
- map → DTO -- on the way back
- filter translates OperationOutcome -- on the way back

Every controller in ddx-api just calls a clean, typed method. You write the seam once.

— SYNTHESIS | TRACE ONE REAL CALL -- THE WHOLE BFF IN ONE PICTURE

web → ddx-api → HAPI → back -- four disciplines, one rail.

GET /api/v1/patients/:id

Where each discipline fires, annotated on the lifecycle

4 **Resource branch → DTO map**
PatientSummaryDto returned as clean JSON

5 **OperationOutcome branch → error filter**
Typed exception -- clean { statusCode, message, code }

Headers · Retry · DTO map · Error translate

— RECAP & BRIDGE | THE BOUNDARY DISCIPLINE, IN YOUR EDITOR

Sketch the wrapper. Trace one call. Then cross the boundary.

— TRY IT YOURSELF

Build the seam, then trace a GET

Sketch the HttpService wrapper -- inject the two headers from request context, retry 5xx three times with backoff and jitter, map one OperationOutcome code to a typed exception. Then trace a single GET Patient call: web → ddx-api → HAPI.

BFF · DTO · Retry · Idempotency · Errors

— NEXT CHAPTER

Ch 14 -- Next.js 16 rendering FHIR

We cross to the other side of this same boundary -- the BFF rule seen from the UI. Same rule, new vantage point.