API Design & Delivery Checklist

A pragmatic, spec-first checklist to take an API from idea → production with confidence.

1) Vision & Requirements

• Define primary user journeys (mobile app, internal tools, 3rd-party).

• Agree on success metrics & SLOs (availability, p95 latency, error budget).

• Choose style: REST with OpenAPI (recommended) or GraphQL (if client tailoring is critical).

• Pick ID strategy (UUIDv7), timestamps (ISO-8601 UTC), and currencies (minor units).

2) Spec-First (OpenAPI)

• Create a minimal OpenAPI with tags, servers, components, and shared schemas.

• For each resource (e.g., lists, items, plans): define endpoints, request/response bodies, and examples; use consistent verbs (GET/POST/PATCH/DELETE); standardize pagination, filtering, and sorting; document one error schema and reuse.

• Lint with Spectral; publish interactive docs (Insomnia/Swagger UI).

• Auto-generate client SDKs (Swift, TypeScript) to reduce hand-written code.

3) AuthN & AuthZ

• Decide on OAuth2/OIDC or email-link; use short-lived access + refresh tokens.

• Enforce per-user/workspace isolation; role scopes for shared lists.

• Transport security (HTTPS only), CORS policy, cookie vs bearer tokens.

4) Data Modeling & Migrations

• ER diagram for core entities (User, Plan, Recipe, List, Item).

• Add constraints (FKs, uniqueness, cascade rules).

• Plan migrations (Alembic/Prisma) and rollback procedure.

• Seed data and anonymized fixtures for dev/staging.

5) Write Semantics

• Idempotency for unsafe operations (`Idempotency-Key`).

• Optimistic concurrency via version fields or ETags (409 on conflict).

• Soft deletes vs hard deletes (with retention policy).

6) Query Semantics

• Stable pagination cursors and a default sort order.

• Consistent filtering syntax (e.g., `?status=active&after=...`).

• Sparse fieldsets / includes when shaping responses is needed.

7) Error Handling

• Single error envelope: `code`, `message`, `details`, `requestId`.

• Map all exceptions to documented HTTP statuses; never leak internals.

• Human-readable messages + machine-readable `code`s.

8) Performance & Caching

• Add indexes for high-cardinality lookups; avoid N+1 with joins/loader.

• Use CDN/HTTP caching for GETs (ETag/Last-Modified).

• Consider app-level cache (Redis) for hot reads; set TTLs thoughtfully.

9) Observability

• Structured logs with request ID, user ID, route, status, duration.

• Metrics: request rate, error rate, latency (p50/p95/p99), DB timings.

• Tracing across services; surface slow endpoints in dashboards.

10) Security

• Validate inputs (size/shape/range); sanitize outputs.

• Rate limiting & abuse detection; per-IP and per-user.

• Secret management (env vars, vault), dependency scanning, SBOM.

11) Testing Strategy

- Contract tests generated from OpenAPI examples.
- Unit tests; integration tests against a real DB.
- End-to-end happy paths; load tests for critical endpoints.
- Negative tests for auth, limits, and malformed input.

12) CI/CD & Environments

- CI: spec lint, code lint/format, tests, migrations check.
- Preview environments per PR; seed with fixtures.
- Blue/green or canary deploy; automatic rollback on health checks.

13) Backward Compatibility & Versioning

- Prefer additive changes; never break fields without a deprecation window.
- Support parallel versions when removing/renaming fields.
- Communicate changes via changelog and API-deprecation headers.

14) Documentation & DX

- Publish OpenAPI, quickstart, and copy-pasteable curl/HTTPie examples.
- Provide an Insomnia/Thunder Client collection synced to the spec.
- Add a sandbox user and sample data for new developers.

15) Runbooks & Ops

- Incident playbook (on-call, severity levels, escalation).
- Backup/restore procedures; DR objectives (RPO/RTO).
- Data retention and GDPR/DSAR handling (export/delete APIs).

API Design & Delivery Checklist

A pragmatic, spec-first checklist to take an API from idea → production with confidence.

How to Use This Checklist

- Draft your OpenAPI first (resources, auth, errors, examples).
- Generate clients/stubs and wire a thin service layer.
- Stand up migrations and seed data.
- Add idempotency, pagination, and error mapping before the app hits the API.
- Add logging/metrics, then load test core flows.
- Ship to staging with preview envs and contract tests gating deploys.
- Release, monitor SLOs, iterate.

Tooling Quickstart

- OpenAPI authoring: Stoplight Studio or VS Code + OpenAPI extension.
- Linting: Spectral; fail CI on breaking changes.
- Client generation: openapi-generator (Swift, TypeScript).
- Manual testing: Insomnia/Thunder Client; import the OpenAPI.
- Docs: Swagger UI or Redoc; host alongside your API.