Lesson 6.3: API Economy and Platform Architecture – Practice Exercises

Prof. Dr. Joerg Osterrieder

Exercise 1: API Style Selection

For each financial use case, choose the most appropriate API style (REST, GraphQL, or Webhook) and justify your choice:

1. A mobile banking app that displays the current balance of a single checking account
2. A portfolio management dashboard that shows holdings, P&L, and risk metrics from three different custodians in one screen
3. A fraud monitoring system that needs to be notified the instant a suspicious transaction occurs
4. A regulatory reporting system that pulls end-of-day account snapshots for all customers
5. A point-of-sale terminal that initiates a card payment and waits for approval
6. An accounting software that receives a callback when an invoice payment settles

For each, also identify:

- Whether the interaction is synchronous or asynchronous
- The acceptable latency (real-time, near-real-time, batch)

Exercise 2: Open Banking Data Flow

A customer named Alice wants to use a budgeting app ("BudgetWise") to view her transaction history from two banks (Bank A and Bank B) and initiate a transfer from Bank A to Bank B.

Tasks:

1. Draw the complete data flow for the AIS (read) operation, including:
   - Customer consent step and SCA
   - OAuth 2.0 authorization code exchange
   - API call from BudgetWise to each bank
   - Data returned to the customer
2. Draw the data flow for the PIS (write) operation to initiate the transfer
3. Identify which party (BudgetWise, Bank A, Bank B, or the customer) is responsible for:
   - Authenticating the customer
   - Verifying BudgetWise's license
   - Executing the transfer
   - Handling a failed payment
4. What happens if Bank A's API is down? Who is liable under PSD2?

Exercise 3: API Gateway Design

You are the lead architect at a mid-size bank. The bank has 15 internal microservices (accounts, payments, cards, loans, KYC, etc.) and wants to expose 5 external APIs to third-party fintechs.

Design the API gateway layer:

1. List the 5 cross-cutting concerns the gateway must handle. For each, explain *why* it cannot be delegated to individual microservices.
2. Define rate limiting rules for three tiers of API consumers:
   - Free tier (sandbox/testing)
   - Standard tier (licensed fintechs)
   - Premium tier (strategic partners)
3. The payments microservice has a newer version (v2) alongside the legacy version (v1). Describe your API versioning strategy: URL-based, header-based, or query parameter? Justify.
4. How would you handle a scenario where the KYC microservice is down but all other services are operational? Design the circuit breaker behavior at the gateway level.

Exercise 4: OAuth 2.0 Security Analysis

A fintech app ("PayQuick") uses OAuth 2.0 Authorization Code flow to access a bank's payment API. Analyze the following security scenarios:

1. An attacker intercepts the authorization code during the redirect. What prevents them from exchanging it for an access token? (Hint: PKCE)
2. PayQuick's access token is stolen from their server logs. What limits the damage? Specify at least three mitigating controls.
3. A rogue app registers a similar name ("PayQu1ck") and tricks users into authorizing it. What controls prevent this under FAPI 2.0?
4. The bank wants to allow PayQuick to initiate payments up to €500 but only read account data for balances. How would you implement this using OAuth scopes?

Bonus: Explain the difference between OAuth 2.0 (authorization) and OpenID Connect (authentication). Why does open banking need both?

Exercise 5: BaaS Provider Risk Assessment

Your fintech startup uses a single BaaS provider to offer accounts, cards, and lending. After studying the Synapse collapse, your board demands a risk assessment.

Complete the following:

1. Identify five specific risks of depending on a single BaaS provider. Categorize each as operational, financial, regulatory, or reputational.
2. For each risk, propose a concrete mitigation strategy.
3. Design a multi-provider architecture where:
   - Provider A handles accounts and cards
   - Provider B handles lending
   - Your own system handles the customer-facing layer and reconciliation

   Draw the architecture diagram and identify the reconciliation points.
4. The FDIC issued guidance on BaaS risks in 2024. What is the key principle regarding who is responsible for customer deposits in a BaaS chain?

Exercise 6: Embedded Finance Business Case

An online marketplace ("ShopNow") with 500,000 active merchants wants to offer embedded lending (merchant cash advances) and embedded insurance (shipping insurance at checkout).

Tasks:

1. Map the value chain: for each product (lending and insurance), identify the licensed provider, the API middleware, and the distribution channel. Who earns revenue at each layer?
2. Calculate the embedded lending economics:
   - Average loan size: €15,000, term: 6 months, interest rate: 12% APR
   - ShopNow's revenue share: 20% of interest income
   - Expected default rate: 4%
   - If 5% of merchants take a loan annually, what is ShopNow's annual revenue from embedded lending?
3. Why does ShopNow have a *data advantage* over a traditional bank in underwriting these loans? Identify three data signals ShopNow has that banks do not.
4. What regulatory risks does ShopNow face by offering embedded financial products? Consider: licensing, consumer protection, and data privacy.

Exercise 7: Developer Experience Audit

You are evaluating three payment API providers for your startup. Conduct a DX audit:

DX Criterion	Provider A	Provider B	Provider C
Time to first API call	3 minutes (self-serve)	2 weeks (sales call)	30 minutes (manual key)
Documentation	Interactive (Swagger)	PDF manual	Static HTML
Sandbox	Full sandbox, test cards	No sandbox	Limited sandbox
SDKs	Python, JS, Go, Java, Ruby	Java only	Python, JS
Error messages	Machine-readable codes + human text	HTTP 500 only	Codes, no text
Versioning	URL-based, 12-month deprecation	No versioning	Header-based

Tasks:

1. Rank the providers and justify your choice
2. For the worst provider, propose three specific improvements
3. Estimate the cost of choosing Provider B if your team has 4 developers at €80/hour. How many extra hours will poor DX cost over 6 months?

Exercise 8: API Economy Strategy

A traditional European bank (200 years old, 5 million customers, full banking license) must respond to PSD3 and the rise of embedded finance.

Tasks:

1. Evaluate three strategic options: (a) comply minimally with PSD3, (b) launch a BaaS offering, (c) become a platform with its own developer ecosystem. For each, identify pros, cons, required investment, and 5-year revenue impact.
2. The bank's CTO proposes building an API developer portal. Design the portal:
   - List the 6 most important APIs to expose first and justify the sequence
   - Define the onboarding funnel: sign-up $\rightarrow$ sandbox $\rightarrow$ production. What conversion rate targets would you set?
   - Propose a pricing model (freemium, per-call, revenue share, or tiered)
3. Identify the bank's competitive advantages over pure BaaS providers (e.g., Solarisbank). What can a 200-year-old bank offer that a startup cannot?
4. What is the biggest risk if the bank does nothing? Quantify using the concept of "death by a thousand APIs."

Answer Key (1/3)

Exercise 1: 1) REST – synchronous, single resource, real-time (<200ms). 2) GraphQL – synchronous, multi-entity flexible query, near-real-time (<1s). 3) Webhook – asynchronous, event-driven push, real-time (immediate). 4) REST – synchronous batch pull, batch (acceptable delay minutes/hours). 5) REST – synchronous, single resource request/response, real-time (<500ms). 6) Webhook – asynchronous, server pushes on settlement event, near-real-time.

Exercise 2: 1) AIS flow: Customer → opens BudgetWise → BudgetWise redirects to Bank A/B login → Customer completes SCA → Bank issues authorization code → BudgetWise exchanges for access token → BudgetWise calls GET /accounts/transactions → Data displayed. 2) PIS flow: Customer selects transfer in BudgetWise → BudgetWise sends POST /payments to Bank A → Bank A triggers SCA → Customer confirms → Bank A executes transfer → Webhook confirms settlement. 3) Authentication: Bank A/B. License verification: Bank A/B (checks AISP/PISP register). Execution: Bank A. Failed payment: Bank A (originator) + BudgetWise (inform customer). 4) Bank A is liable for API availability under PSD2/PSD3 SLAs. BudgetWise should implement graceful degradation.

Answer Key (2/3)

Exercise 3: 1) Five cross-cutting concerns: (i) Authentication – centralized identity verification avoids duplicating security logic. (ii) Rate limiting – per-consumer fairness requires global view. (iii) Logging/audit – regulatory audit trail must be consistent. (iv) Request routing – consumers see one URL, gateway routes internally. (v) Format transformation – legacy XML services behind JSON facade. 2) Rate limits: Free: 100 req/min, 1,000/day. Standard: 1,000 req/min, 100,000/day. Premium: 10,000 req/min, unlimited/day. 3) URL-based versioning (/v1/payments, /v2/payments) because: clearest for consumers, easiest to route at gateway, simple cache invalidation. Deprecation: 12-month notice, sunset header. 4) Circuit breaker: gateway returns HTTP 503 for KYC endpoints, health-check every 30s, auto-recover when healthy. Other services remain operational. Consumer receives clear error: "KYC service temporarily unavailable, other services unaffected."

Exercise 4: 1) PKCE (Proof Key for Code Exchange): the client generates a random code_verifier, sends hash (code_challenge) in auth request. Only the client that generated the verifier can exchange the code. 2) Three mitigations: (i) short-lived tokens (5 min), (ii) scope limitation (read-only), (iii) IP binding or client certificate-bound tokens (FAPI). Also: refresh token rotation, anomaly detection. 3) FAPI 2.0 requires: pushed authorization requests (PAR) preventing redirect manipulation, client certificates tied to regulated entity, dynamic client registration restricted to licensed TPPs. 4) Scopes: `accounts:balance:read` for balance access, `payments:initiate` with amount limit €500. Token issued with both scopes; payment API enforces amount ceiling. Bonus: OAuth = authorization (what you can access). OIDC = authentication (who you are). Open banking needs both: verify identity (OIDC) and grant scoped access (OAuth).

Answer Key (3/3)

Exercise 5: 1) Risks: (i) Operational: provider outage = total outage. (ii) Financial: provider bankruptcy (Synapse). (iii) Regulatory: license revocation of provider. (iv) Reputational: provider data breach harms your brand. (v) Financial: price increases with no alternative. 2) Mitigations: (i) Multi-provider architecture. (ii) Escrow/custody agreements. (iii) Direct relationship with underlying bank. (iv) Own reconciliation engine. (v) Contractual price caps + exit clauses. 3) Architecture: customer layer (yours) $\rightarrow$ reconciliation engine (yours) $\rightarrow$ Provider A (accounts/cards) + Provider B (lending). Reconciliation at: daily balance verification, real-time transaction matching, monthly statement comparison. 4) FDIC principle: the chartered bank is ultimately responsible for all depositor funds, regardless of middleware layers.

Exercise 6: 1) Lending chain: Licensed lender $\rightarrow$ BaaS API middleware $\rightarrow$ ShopNow (distribution). Revenue split at each layer. 2) Merchants taking loans: $500{,}000 \times 5\% = 25{,}000$ loans. Interest per loan: $€15{,}000 \times 12\% \times 0.5 = €900$. Gross interest: $25{,}000 \times €900 = €22.5\text{M}$. Defaults: $25{,}000 \times 4\% \times €15{,}000 = €15\text{M}$. Net interest: €7.5M. ShopNow's 20% share: **€1.5M/year**. 3) Data advantages: (i) real-time sales volume and trends, (ii) customer return/chargeback rates, (iii) seasonal patterns and growth trajectory. 4) Risks: (i) consumer credit regulation (may need license), (ii) GDPR: using sales data for underwriting requires explicit consent, (iii) consumer protection: duty of care for financial product recommendations.

Exercise 7: 1) Provider A (best DX), C (acceptable), B (worst). 2) Provider B improvements: self-serve sign-up, sandbox, multi-language SDKs. 3) Extra hours: assume 40 hours/dev wasted on poor docs, no sandbox, manual testing over 6 months $= 4 \times 40 \times €80 = €12{,}800$ minimum.

Exercise 8: Strategy (c) is highest-risk/highest-reward. Bank advantages: trust, existing customer base, full license, deposit base. "Death by 1,000 APIs": each fintech captures a thin slice of the value chain; the bank retains the license cost but loses the revenue.