Luồng xử lý API Backend (API Processing Flow)

1.1. Mục tiêu chương.

Mô tả luồng xử lý chuẩn của một API backend từ khi nhận request đến khi deploy production nhằm:

Thống nhất cách xây dựng API giữa các team

Đảm bảo tuân thủ đầy đủ các quy định kiến trúc

Dễ trace lỗi và audit

Liên kết các chuẩn đã quy định ở các chương trước

1.2. Phạm vi áp dụng:

Áp dụng cho:

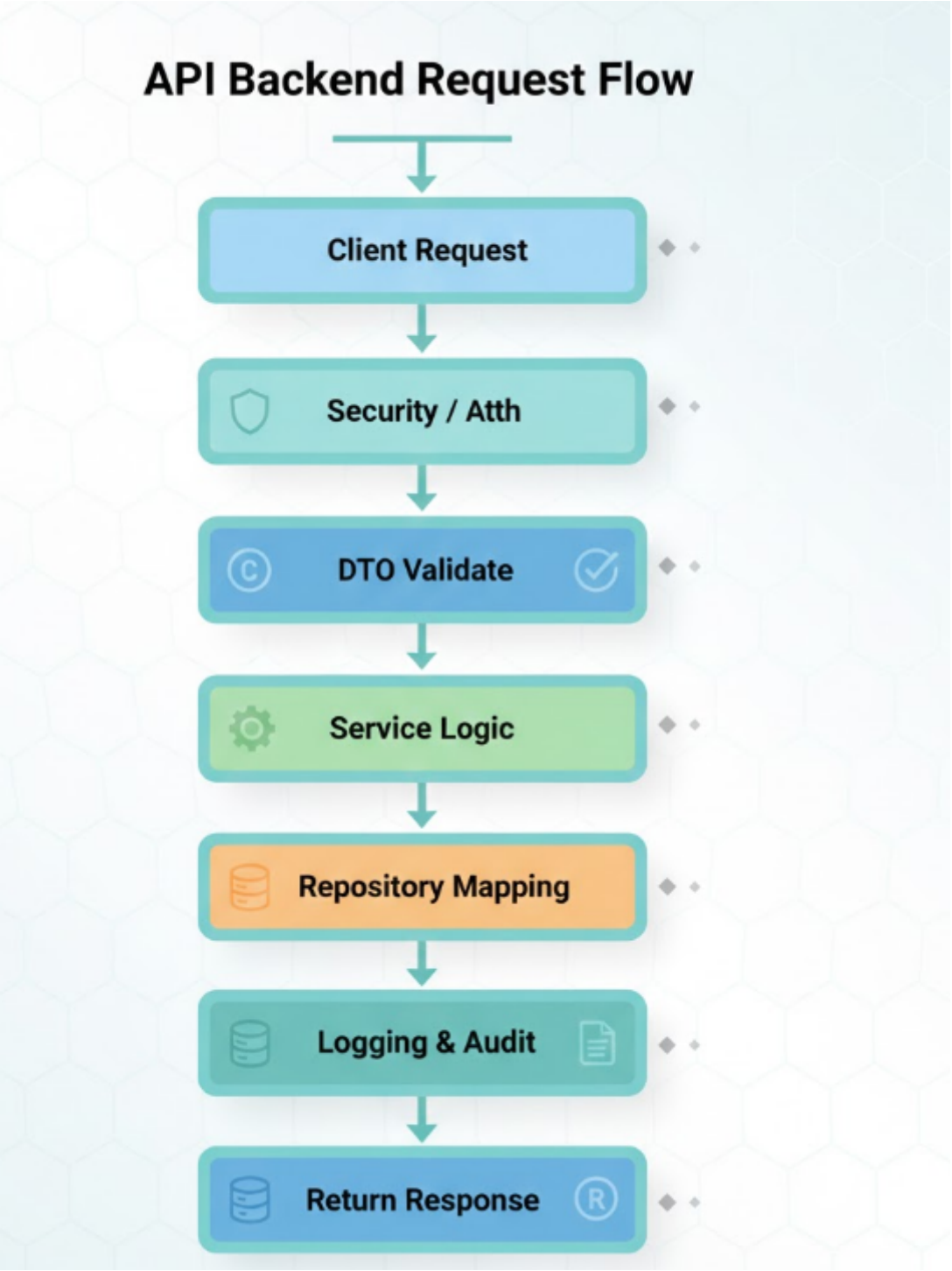
Tất cả REST API trong modules/**

Core Team và Partner Team

Tất cả service Spring Boot

1.3. Tổng quan luồng xử lý API

Luồng chuẩn:



1.4. Các bước xử lý chi tiết

Bước 1. Client gọi API

Ví dụ:

POST /api/v1/users
Payload:

```
{
  "username": "test",
  "password": "123456"
}
```

Quy định liên quan:

□□ Chương 7 — API Design & Response Standard

Bước 2. Security / Authentication

Hệ thống kiểm tra:

JWT / SSO token

Permission

Role

Nếu fail:

401 Unauthorized
403 Forbidden
Quy định liên quan:

□□ Chương 8 — Cơ chế xác thực & SSO (WSO2)

Bước 3. Controller nhận request

```
@PostMapping
public ApiResponse<UserResponse> create(
        @Valid @RequestBody UserCreateRequest request) {
    return ApiResponse.ok(userService.create(request));
}
```

Quy định:

Name Controller

Không chứa business logic

□□ Chương 3 — Coding Convention
□□ Chương 7 — API Design

Bước 4. DTO Validation

DTO:

```
public class UserCreateRequest {

    @NotBlank
    @Size(max = 100)
    private String username;

    @NotBlank
    private String password;
}
```

Spring tự validate trước khi vào service.

Nếu lỗi:

400 Bad Request
Response chuẩn:

```
{
  "code": "VALIDATION_ERROR",
  "message": "username is required"
}
```

Quy định:

□□ Chương 3 — Coding Convention (Validation)
□□ Chương 7 — Response Standard

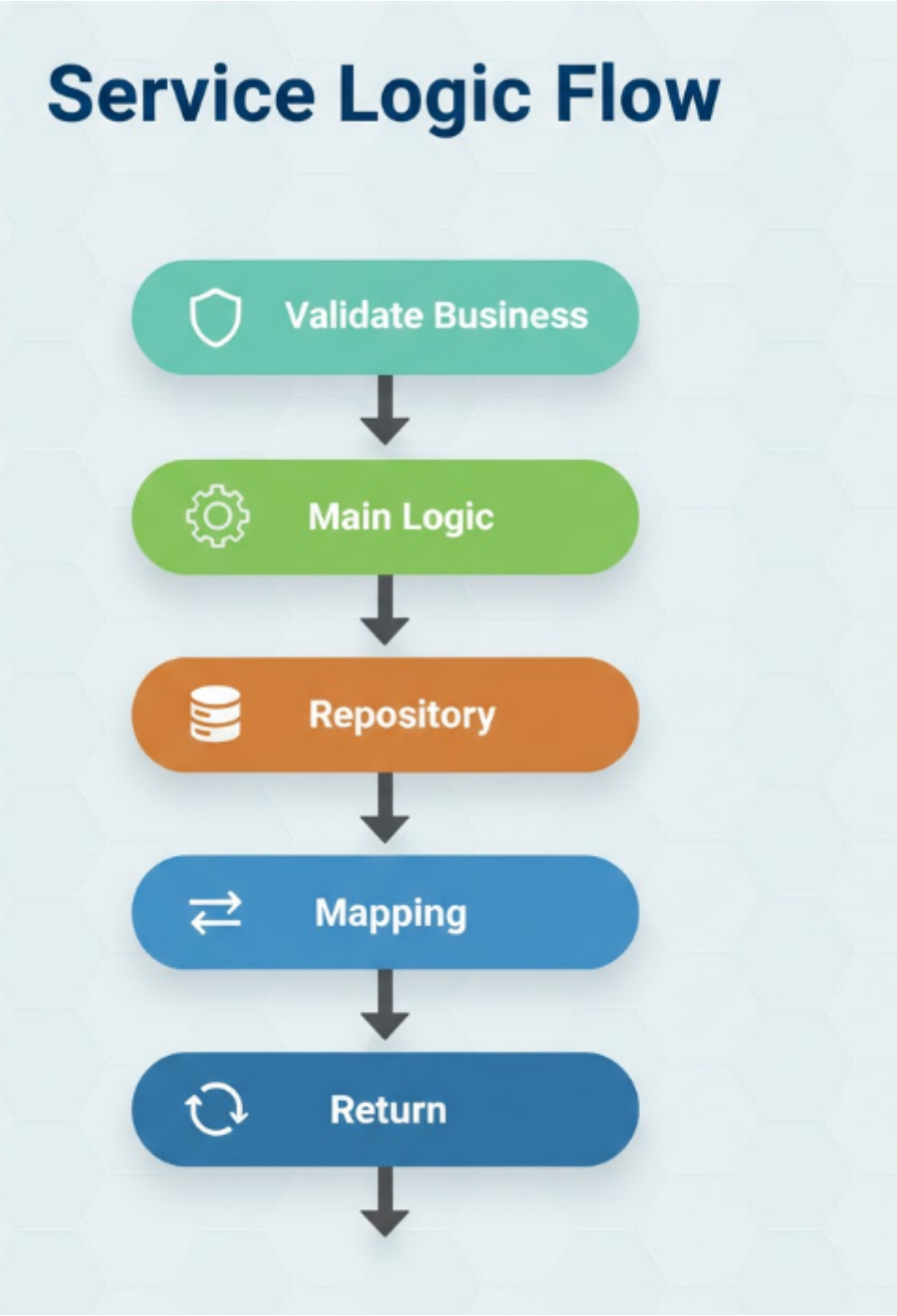
Bước 5. Service xử lý logic

```
public UserResponse create(UserCreateRequest req) {
    log.info("Create user {}", req.getUsername());

    UserEntity e = mapper.toEntity(req);
    repo.save(e);

    return mapper.toResponse(e);
}
```

Cấu trúc chuẩn 1 hàm service:



Quy định:

□□ Chương 3 — Coding Convention
□□ Chương 4 — Entity & Database

Bước 6. Repiository & Database

userRepository.save(entity);
Nếu thay đổi schema:

update entity

migration script

Quy định:

□□ Chương 4 — Thay đổi Entity & DB
□□ Chương 5 — Schema & Migration

Bước 7. Mapping Response

```
return UserResponse.builder()
        .id(e.getId())
        .username(e.getUsername())
        .build();
```

Response chuẩn:

```
{
  "code": "SUCCESS",
  "data": {
    "id": 1,
    "username": "test"
  }
}
```

Quy định:

□□ Chương 7 — Response Standard

Bước 8. Logging & Audit

Trong service:

```
log.info("User created id={}", e.getId());
```

Nếu nghiệp vụ quan trọng:

ghi audit_log

Quy định:

□□ Chương 3 — Logging
□□ Chương 6 — Audit Log

Bước 9. Exception Handling

Ví dụ:

```
if (exists) {
    throw new BusinessException("USER_EXISTS");
}
```

Global handler:

```
@ExceptionHandler(BusinessException.class)
```

Response:

```
{
  "code": "USER_EXISTS",
  "message": "User already exists"
}
```

Quy định:

□□ Chương 7 — Response & Error

1.5. Luồng phát triển & release API

Sau khi code xong:

```
Dev → Commit → PR → Review → Merge → CI Build → Deploy
```

Bước 10. Commit & PR

Quy định:

□□ Chương 9 — Git Workflow & Pull Request

Bước 11. Checklist trước merge

Quy định:

□□ Chương 10 — Checklist merge

Bước 12. Deploy & Update DB

Nếu có migration:

chạy script

deploy service

Quy định:

□□ Chương 5 — Migration
□□ Chương 12 — Connect DB

1.6. Sơ đồ tổng thể API lifecycle

https://docs.lifetex.vn/link/333#bkmrk-request-%E2%86%93-auth-%28ch8%29

Request
  ↓
Auth (Ch8)
  ↓
Controller (Ch3,7)
  ↓
DTO Validate (Ch3)
  ↓
Service Logic (Ch3)
  ↓
Entity/DB (Ch4,5)
  ↓
Mapping (Ch7)
  ↓
Logging/Audit (Ch3,6)
  ↓
Response (Ch7)
  ↓
Git/PR (Ch9,10)
  ↓
Deploy (Ch5,12)

1.7. Transaction trong xử lý API

1.7.1. Nguyên tắc transaction

Mọi thao tác thay đổi dữ liệu nghiệp vụ trong API phải được thực hiện trong transaction nhằm đảm bảo:

- Tính toàn vẹn dữ liệu
- Tính nhất quán hệ thống
- Khả năng rollback khi lỗi
- Đồng bộ dữ liệu và audit log

Transaction phải được đặt tại **Service layer**.

1.7.2. Quy định transaction

Transaction:

- đặt tại Service
- bao phủ toàn bộ nghiệp vụ
- bao gồm repository và audit
- rollback khi exception

Không được đặt transaction tại:

- Controller
- Repository
- Mapper

1.7.3. Ví dụ transaction chuẩn

```
@Transactional
public UserResponse create(UserCreateRequest req) {

    if (repo.existsByUsername(req.getUsername())) {
        throw new BusinessException("USER_EXISTS");
    }

    UserEntity e = mapper.toEntity(req);
    repo.save(e);

    auditService.logCreate("USER", e.getId());

    return mapper.toResponse(e);
```

```
}
```

1.7.4. Quy tắc rollback

Transaction phải rollback khi:

- BusinessException
- RuntimeException
- Database error
- Validation fail trong service

Không được:

- ghi audit trước khi save dữ liệu
- commit từng phần nghiệp vụ
- xử lý lỗi nhưng vẫn commit

1.8. Security Context trong Service

1.8.1. Khái niệm

Security Context là thông tin người dùng hiện tại sau khi xác thực (JWT / SSO).

Thông tin thường có:

- userId
- username
- role
- permission
- tenant
- organization

1.8.2. Sử dụng trong Service

Service được phép truy cập Security Context để:

- xác định user thao tác
- kiểm tra permission
- ghi audit
- filter dữ liệu theo user

Ví dụ:

```
Long userId = SecurityUtils.getCurrentUserId();
String username = SecurityUtils.getCurrentUsername();
```

1.8.3. Quy định sử dụng

Không được:

- tin userId từ request
- truyền userId từ Controller nếu đã có context
- bypass permission check

Permission check phải tại Service layer.

1.8.4. Ví dụ kiểm tra quyền

```
public void updateUser(Long id, UserUpdateRequest req) {

    Long currentUser = SecurityUtils.getCurrentUserId();

    if (!permissionService.canUpdateUser(currentUser, id)) {
        throw new AccessDeniedException("NO_PERMISSION");
    }

    ...
}
```

1.9. Xử lý API danh sách (List API)

1.9.1. Nguyên tắc List API

API trả danh sách phải hỗ trợ:

- pagination
- sorting
- filter

Mục tiêu:

- tránh trả dữ liệu lớn
- đảm bảo hiệu năng DB
- hỗ trợ UI paging

1.9.2. Chuẩn request List API

Ví dụ:

```
GET /api/v1/users?page=0&size=20&sort=createdAt,desc
```

Tham số chuẩn:

- page (bắt đầu từ 0)
- size
- sort
- filter

1.9.3. Chuẩn response List API

```
{
  "code": "SUCCESS",
  "data": {
    "content": [],
    "page": 0,
    "size": 20,
    "totalElements": 100,
    "totalPages": 5
  }
}
```

1.9.4. Quy định bắt buộc

- Không trả toàn bộ bảng
- Pagination bắt buộc với list lớn
- Repository phải dùng paging query
- Không load toàn bộ relation

1.10. Gọi hệ thống ngoài trong API

1.10.1. Phạm vi

API có thể cần gọi:

- SSO
- External API
- Message Queue

- Cache
- File Storage

1.10.2. Kiến trúc gọi ngoài

Luồng chuẩn:

Service → Integration Client → External System

Ví dụ:

```
UserInfo info = ssoClient.getUser(token);
```

1.10.3. Quy định

Không được gọi external tại:

- Controller
- Repository

Phải:

- gọi trong Service
- handle lỗi external
- timeout / retry nếu cần

1.10.4. Xử lý lỗi external

Nếu external fail:

- map thành BusinessException hoặc SystemException
- không expose lỗi raw
- ghi log lỗi

1.11. Monitoring & Metrics trong API

1.11.1. Mục tiêu

Ngoài Logging & Audit, API production cần hỗ trợ:

- theo dõi hiệu năng

- theo dõi lỗi
- theo dõi tải hệ thống

1.11.2. Metrics cần có

- request count
- error rate
- response time
- DB query time
- external call time

1.11.3. Trace request

Mỗi request phải có:

- traceId
- duration
- status

TraceId phải liên kết:

- log
- audit
- exception

1.12. Versioning API

1.12.1. Chuẩn version

API phải có version trong URL:

`/api/v1/...`

1.12.2. Nguyên tắc version

- Không thay đổi breaking trong cùng version
- API cũ phải giữ backward compatibility
- API mới → tăng version
- Không reuse version

1.12.3. Ví dụ

`/api/v1/users`
`/api/v2/users`

1.13. Testing trước merge

1.13.1. Mục tiêu

Đảm bảo API hoạt động đúng trước khi merge và deploy.

1.13.2. Loại test bắt buộc

- Unit test Service
- Validation test
- Error case
- Integration test API

1.13.3. Quy định

API mới hoặc thay đổi logic phải có test:

- thành công
- validation fail
- business fail
- permission fail

1.14. Hiệu năng & Database

1.14.1. Nguyên tắc thiết kế DB trong API

API phải đảm bảo:

- query tối ưu
- index phù hợp
- không N+1 query
- pagination cho list

1.14.2. Các lỗi cần tránh

- load toàn bộ relation
- query trong loop
- select * bảng lớn
- thiếu index filter

1.14.3. Kiểm tra khi review

Khi review API cần kiểm tra:

- query có index không
- list có paging không
- join có cần thiết không
- có N+1 không

Phiên bản #3
Được tạo 2026-02-24 04:52:00 UTC bởi Tuấn Phùng
Được cập nhật 2026-02-24 06:51:18 UTC bởi admin_lifetex