Planejamento Técnico – Sprint 3

Projeto VirtuaLib

Período: 09/10 → 16/10
Valor da Sprint:
Transformar o VirtuaLib em uma arquitetura **orientada a serviços (SOA)**, desacoplando os módulos de autenticação, catálogo, empréstimos e notificações, mantendo todas as funcionalidades existentes.

Objetivo da Sprint

Reestruturar o sistema em **serviços independentes**, permitindo manutenção e deploy separados, sem quebrar as funcionalidades do front-end e garantindo integração via **API Gateway**.

Arquitetura Alvo

Diagrama Simplificado:

```
None
Front-End
   ↓
API Gateway
 ├── Auth Service
 ├── Catalog Service
 ├── Borrow Service
 ├── Notification Service
 └── Admin Dashboard Service
```

Cada serviço roda em um container isolado (Docker) e se comunica via HTTP interno.

User Stories

US10 – Reestruturação para SOA

> Como desenvolvedor, quero dividir a aplicação em múltiplos serviços para facilitar a escalabilidade e manutenção.

- **RF10:** Cada módulo deve ser um microserviço independente (Auth, Catalog, Borrow, Notification, Dashboard).

- **Exemplo:**
  O antigo `AuthController.php` vira um serviço completo, com `index.php`, `routes.php` e `UserModel.php` próprios.

US11 – Gateway de Integração

Como sistema, quero ter um ponto único de entrada que roteie as requisições entre os serviços.

- **RF11:** Criar um `gateway/` que contenha as rotas principais do front e redirecione via cURL ou HTTP interno.

US12 – Comunicação entre Serviços

Como sistema, quero que os serviços troquem dados de forma segura e performática.

- **RF12:** Comunicação via HTTP interno (Docker network) com autenticação via tokens.

- **Exemplo:**
  O Borrow Service faz requisições ao Auth Service para validar o usuário:

```
PHP
$response =
file_get_contents("http://auth-service:8080/validate?token=$token");
```

US13 – Deploy Modular

Como devops, quero rodar todos os serviços de forma isolada via Docker Compose.

- **RF13:** Cada serviço deve ter seu próprio `Dockerfile`.

- **RNF10:** O `docker-compose.yml` orquestra todos.

- **Exemplo:**

```
None
services:
  gateway:
    build: ./gateway
    ports: ["8080:80"]
  auth-service:
    build: ./services/auth
  catalog-service:
    build: ./services/catalog
  borrow-service:
    build: ./services/borrow
```

Tarefas e Responsáveis

ID	Task	Responsável	Revisor	Deadline
10.1	Criar estrutura SOA e Docker Compose	Cayke	Lucas Yudi	09/10
10.2	Migrar módulo Auth para serviço isolado	Gabryel	Filipe	10/10
10.3	Migrar módulo Books/Borrow	Filipe	Lucas Gabriel	11/10
10.4	Criar Notification Service	Lucas Yudi	Gabryel	12/10
10.5	Implementar API Gateway	Cayke	Lucas Gabriel	13/10
10.6	Integrar Dashboard com Gateway	Lucas Gabriel	Filipe	14/10
10.7	Testes de integração	Todos	-	15/10
10.8	Documentação e Swagger	Gabryel	Cayke	16/10

Exemplo de Estrutura Final (parcial)

```
None
VirtuaLib/
├── gateway/
│   └── index.php
│
├── services/
│   ├── auth/
│   │   ├── index.php
│   │   ├── routes.php
│   │   └── UserModel.php
│   ├── catalog/
│   │   ├── index.php
│   │   ├── BookModel.php
│   └── borrow/
│       ├── index.php
│       └── BorrowModel.php
│
└── docker-compose.yml
```

Critérios de Aceite

- O sistema deve rodar com `docker-compose up` criando todos os serviços.
- O front deve continuar acessando as rotas originais.
- Todos os endpoints antigos devem funcionar via Gateway.
- O histórico e notificações continuam persistentes.

Entregáveis

- Estrutura SOA implantada (Auth, Catalog, Borrow, Notification, Dashboard).
- API Gateway funcional.
- Docker Compose orquestrando todos os serviços.
- Documentação e endpoints documentados com Swagger.