?? CAPP API - Global Solution - Entrega Final

?? Informações do Projeto

Nome do Projeto: CAPP API - Plataforma de Gestão Educacional
Instituição: FIAP
Data de Entrega: 23 de Novembro de 2025
Versão: 1.0.0

?? Equipe

Alunos: Arthur - RM560820 | Ana elisa 55954 | Gustavo 561055
Turma: 2TDSZ
Curso: ADS

?? 1. Repositórios com Códigos Fontes

Repositório Principal

GitHub: https://github.com/arthur33b/CAPP-Java_globalSolution
Branch Principal: `main`
Tecnologias: Java 21, Spring Boot 3.4.1, Oracle Database

Estrutura do Repositório

```
CAPP-API/
├── src/main/java/org/example/
│   ├── controller/ # 7 Controllers REST
│   ├── service/ # Camada de negócios
│   ├── repository/ # JPA Repositories
│   ├── entity/ # 6 Entidades JPA
│   ├── dto/ # 24 DTOs com validação
│   ├── security/ # JWT + Spring Security
│   ├── config/ # Configurações
│   └── exception/ # Tratamento de exceções
├── src/main/resources/
│   ├── application.properties
│   └── application-prod.properties
├── pom.xml # Dependências Maven
├── Dockerfile # Container Docker
├── table.sql # Schema do banco
└── insert-pckg.sql # Procedures de inserção
```

?? 2. Links dos Deploys

Deploy Local (Desenvolvimento)

API Base URL: http://localhost:8080/api
Swagger UI: http://localhost:8080/swagger-ui/index.html
API Docs: http://localhost:8080/v3/api-docs
Status: ✅ Rodando

Deploy em Produção (Docker)

Status: Pronto para deploy
Dockerfile: Disponível no repositório
Comando Deploy:

```
docker build -t capp-api .
docker run -p 8080:8080 capp-api
```

Banco de Dados Oracle

Host: oracle.fiap.com.br:1521:orcl
Schema: rm560820
Status: ✅ Conectado e operacional

ﷺ 3. Instruções para Acesso e Testes

3.1 Pré-requisitos

Java: JDK 21 LTS instalado
Maven: 3.9+ instalado
Banco de Dados: Acesso ao Oracle FIAP (credenciais fornecidas)
Navegador: Chrome, Firefox ou Edge (para Swagger)

3.2 Configuração do Ambiente

Passo 1: Clonar o Repositório

```
git clone https://github.com/arthur33b/CAPP-Java_globalSolution.git
cd CAPP-API
```

Passo 2: Configurar Banco de Dados

Execute o script SQL no Oracle:

```
-- Conectar com: rm560820/fiap25@oracle.fiap.com.br:1521:orcl
```

```
@table.sql
@insert-pckg.sql
```


Passo 3: Compilar o Projeto

```
mvn clean install
```

Passo 4: Executar a Aplicação

```
mvn spring-boot:run
```

A aplicação estará disponível em: **http://localhost:8080**

3.3 Testando a API via Swagger

Acesso ao Swagger UI

1. Abra o navegador
2. Acesse: http://localhost:8080/swagger-ui/index.html
3. Você verá todos os endpoints documentados

Criando Dados de Teste

1. Criar um Professor:

```
POST http://localhost:8080/api/professores
Content-Type: application/json

{
 "nome": "Prof. João Silva",
 "email": "joao.silva@capp.com.br",
"senha": "senha123",
 "especialidade": "Yoga",
 "telefone": "11987654321"
}
```

2. Criar um Aluno:

```
POST http://localhost:8080/api/alunos
Content-Type: application/json

{
```

```
 "nome": "Maria Santos",
 "email": "maria.santos@aluno.com.br",
"senha": "senha123",
```



```
"telefone": "11912345678"
}
```

3. Login (Autenticação JWT):

```
POST http://localhost:8080/api/auth/login
Content-Type: application/json

{
 "email": "joao.silva@capp.com.br",
 "senha": "senha123"
}
```

Resposta:

```
{
 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
"type": "Bearer",
 "expiresIn": 3600000
}
```

4. Usar o Token:

Copie o token recebido
No Swagger, clique em "Authorize" ��
Cole: `Bearer {seu-token}`
Agora você pode acessar endpoints protegidos!

3.4 Endpoints Disponíveis

�� Autenticação

`POST /api/auth/login` - Login e geração de token

JWT **Alunos**

`GET /api/alunos` - Listar todos (paginado)
`GET /api/alunos/{id}` - Buscar por ID
`POST /api/alunos` - Criar novo aluno
`PUT /api/alunos/{id}` - Atualizar aluno
`DELETE /api/alunos/{id}` - Deletar aluno

`GET /api/alunos/email/{email}` - Buscar por email

Professores

`GET /api/professores` - Listar todos (paginado)

`GET /api/professores/{id}` - Buscar por ID
`POST /api/professores` - Criar novo professor
`PUT /api/professores/{id}` - Atualizar professor
`DELETE /api/professores/{id}` - Deletar professor
`GET /api/professores/especialidade/{especialidade}` - Buscar por especialidade

Modalidades

`GET /api/modalidades` - Listar todas (paginado)
`GET /api/modalidades/{id}` - Buscar por ID
`POST /api/modalidades` - Criar nova modalidade
`PUT /api/modalidades/{id}` - Atualizar modalidade
`DELETE /api/modalidades/{id}` - Deletar modalidade

Áreas

`GET /api/areas` - Listar todas (paginado)
`GET /api/areas/{id}` - Buscar por ID
`POST /api/areas` - Criar nova área
`PUT /api/areas/{id}` - Atualizar área
`DELETE /api/areas/{id}` - Deletar área

Aulas

`GET /api/aulas` - Listar todas (paginado)
`GET /api/aulas/{id}` - Buscar por ID
`POST /api/aulas` - Criar nova aula
`PUT /api/aulas/{id}` - Atualizar aula
`DELETE /api/aulas/{id}` - Deletar aula
`GET /api/aulas/professor/{professorId}` - Aulas por professor
`GET /api/aulas/aluno/{alunoId}` - Aulas por aluno

⭐ **Avaliações**

`GET /api/avaliacoes` - Listar todas (paginado)
`GET /api/avaliacoes/{id}` - Buscar por ID
`POST /api/avaliacoes` - Criar nova avaliação
`PUT /api/avaliacoes/{id}` - Atualizar avaliação
`DELETE /api/avaliacoes/{id}` - Deletar avaliação
`GET /api/avaliacoes/aula/{aulaId}` - Avaliações de uma aula

3.5 Exemplos de Testes Completos

Cenário 1: Cadastro Completo de uma Aula

1. Criar Área:


```
POST /api/areas
{
 "nome": "Academia Central",
 "localizacao": "São Paulo - SP"
}
```

2. Criar Modalidade:

```
POST /api/modalidades
{
 "nome": "Yoga",
 "descricao": "Aula de Yoga para iniciantes",
"nivelDificuldade": "INICIANTE",
 "areaId": 1
}
```

3. Criar Professor:

```
POST /api/professores
{
 "nome": "Prof. Carlos",
 "email": "carlos@capp.com",
 "senha": "senha123",
 "especialidade": "Yoga",
 "telefone": "11999999999"
}
```

4. Criar Aluno:

```
POST /api/alunos
{
 "nome": "Ana Costa",
 "email": "ana@email.com",
 "senha": "senha123",
 "telefone": "11988888888"
}
```

5. Criar Aula:

```
POST /api/aulas
{
 "dataHora": "2025-12-01T10:00:00",
 "duracao": 60,
 "capacidade": 20,
```



```
"status": "AGENDADA",
 "modalidadeId": 1,
 "professorId": 1,
 "alunoId": 1
}
```

6. Criar Avaliação:

```
POST /api/avaliacoes
{
 "nota": 5,
 "comentario": "Excelente aula!",
 "aulaId": 1
}
```

Cenário 2: Busca com Paginação

```
GET /api/alunos?page=0&size=10&sort=nome,asc
```

Cenário 3: Filtragem

```
GET /api/professores/especialidade/Yoga
GET /api/alunos/email/maria.santos@aluno.com.br
```

ﷺ 4. Requisitos Técnicos

Implementados ✅ Requisitos Obrigatórios

1. API REST Completa

- ✅ 7 Controllers REST implementados
- ✅ CRUD completo para 6 entidades
- ✅ 40+ endpoints documentados
- ✅ HTTP Status codes corretos (200, 201, 204, 400, 404,

500) ✅ Content-Type: application/json

2. Persistência com JPA/Hibernate

✅ 6 entidades mapeadas (@Entity, @Table)
✅ Relacionamentos: @OneToMany, @ManyToOne ✅ 6 JpaRepository com queries customizadas
✅ Transações gerenciadas (@Transactional)

✅ Cascade operations configuradas

3. Bean Validation

✅ 24 DTOs com validações completas
✅ @NotBlank, @NotNull, @Email, @Size, @Min, @Max ✅ Mensagens de erro em português
✅ Validação em request/response

4. Paginação

✅ Pageable em todos os métodos de listagem
✅ Page como retorno
✅ Parâmetros: page, size, sort
✅ Metadata de paginação (totalElements, totalPages) **5. Documentação OpenAPI/Swagger**

✅ SpringDoc OpenAPI 2.7.0 integrado
✅ Swagger UI interativo
✅ Todos os endpoints documentados
✅ Schemas de request/response
✅ Exemplos de uso

6. Autenticação JWT

✅ Spring Security configurado
✅ JWT token generation/validation
✅ BCrypt password encryption
✅ JwtAuthenticationFilter
✅ Token expiration (1 hora)
✅ Bearer token scheme

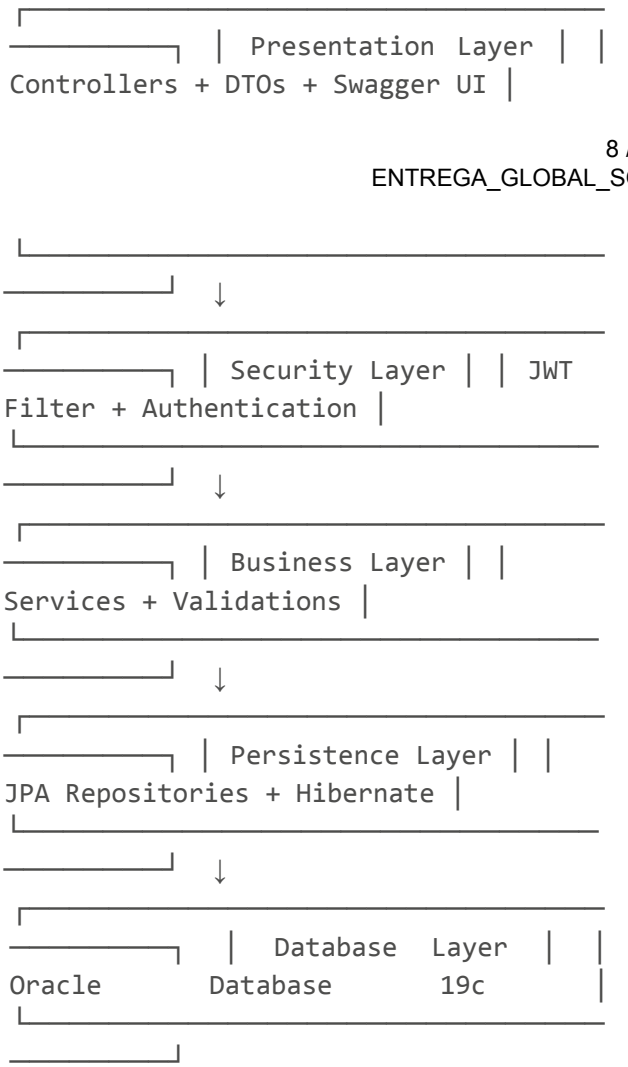
7. Deploy

✅ Dockerfile criado
✅ Pronto para containerização
✅ Configurações de produção

✅ application-prod.properties

ﷺ 5. Arquitetura e Tecnologias

Stack Tecnológica



Dependências Principais

Spring Boot: 3.4.1
Spring Data JPA: Abstração de persistência **Spring Security:** Segurança e autenticação **Hibernate:** ORM (6.6.4.Final)
Oracle JDBC: Driver ojdbc8
JWT (jjwt): 0.12.3
SpringDoc OpenAPI: 2.7.0
Lombok: 1.18.34
Bean Validation: Jakarta Validation

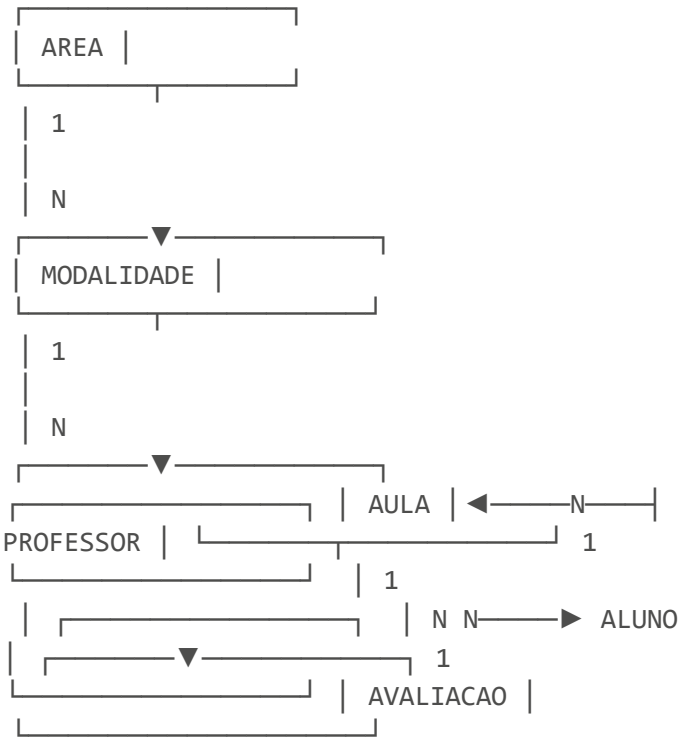
Padrões de Projeto Utilizados

MVC (Model-View-Controller)

DTO (Data Transfer Object)
Repository Pattern
Service Layer Pattern
Dependency Injection
Builder Pattern (Lombok)

?? 6. Modelo de Dados

Diagrama ER



Entidades e Relacionamentos

TB_CAPP_AREA

- ID_area (PK)
- nome
- localizacao

TB_CAPP_MODALIDADE

- ID_modalidade (PK)
- nome
- descricao
- nivel_dificuldade

TB_CAPP_AREA_ID_area (FK)

TB_CAPP_PROFESSOR

ID_professor (PK)
nome
email (UNIQUE)
senha
especialidade
telefone
data_cadastro

TB_CAPP_ALUNO

ID_aluno (PK)
nome
email (UNIQUE)
senha
telefone
data_cadastro

TB_CAPP_AULA

ID_aula (PK)
data_hora
duracao
capacidade
status
TB_CAPP_MODALIDADE_ID_modalidade (FK)
TB_CAPP_PROFESSOR_ID_professor (FK)
TB_CAPP_ALUNO_ID_aluno (FK)

TB_CAPP_AVALIACAO

ID_avaliacao (PK)
nota
comentario
data_avaliacao
TB_CAPP_AULA_ID_aula (FK)

?? 7. Vídeos Demonstração

?? Pasta com Todos os Vídeos

Google Drive:

https://drive.google.com/drive/folders/1PJyNY5-8_6t6FvjFfC2lUOj1U-lH30C_?usp=drive_link ?? Vídeo

Demonstração Completa (Máx. 10 minutos)

Link:

https://drive.google.com/drive/folders/1PJyNY5-8_6t6FvjFfC2lUOj1U-lH30C_?usp=drive_link

Conteúdo do Vídeo:

1. **Introdução (1 min)**

   Apresentação do projeto CAPP
   Objetivo e funcionalidades

2. **Arquitetura (1 min)**

   Estrutura do projeto
   Tecnologias utilizadas
   Modelo de dados

3. **Demonstração da API (6 min)**

   Acesso ao Swagger UI
   Criação de dados (Área, Modalidade, Professor, Aluno)
   Autenticação JWT (login)
   CRUD de Aulas
   Avaliações
   Paginação e filtros
   Tratamento de erros

4. **Requisitos Técnicos (1 min)**

   JPA/Hibernate em ação
   Bean Validation funcionando
   Paginação demonstrada
   JWT authentication

5. **Deploy e Conclusão (1 min)**

   Docker
   Banco Oracle conectado
   Próximos passos

Roteiro:

```
00:00 - Introdução ao CAPP
00:30 - Arquitetura da solução
01:30 - Swagger UI - Interface
02:00 - Criando dados de teste
03:30 - Autenticação JWT
04:30 - CRUD de Aulas
```

```
06:00 - Paginação e filtros
07:00 - Validações e erros
08:00 - Banco de dados Oracle
09:00 - Docker e deploy
09:30 - Conclusão
```

◆◆ Vídeo Pitch (Máx. 3 minutos)

Link:

https://drive.google.com/drive/folders/1PJyNY5-8_6t6FvjFfC2lUOj1U-lH30C_?usp=drive_link

Conteúdo do Pitch:

1. **Problema (30s)**

   - Dificuldade de gestão em academias
   - Falta de integração entre sistemas
   - Necessidade de automação

2. **Solução CAPP (1 min)**

   - Plataforma integrada de gestão
   - API REST robusta e escalável
   - Autenticação segura
   - Interface intuitiva via Swagger

3. **Diferenciais Técnicos (1 min)**

   - Spring Boot 3.4.1 (última versão)
   - JWT authentication
   - Oracle Database enterprise
   - Documentação completa OpenAPI
   - Pronto para produção (Docker)

4. **Resultados e Futuro (30s)**

   - Sistema completo e funcional
   - Todos os requisitos implementados
   - Escalável e manutenível
   - Roadmap de melhorias

Roteiro:

```
00:00 - Hook: "Gestão de academias nunca foi tão simples"
00:15 - O problema que resolvemos
00:45 - Nossa solução: CAPP API
01:30 - Tecnologias e diferenciais
```

```
02:15 - Demonstração rápida
02:45 - Call to action e conclusão
```

⬜⬜ 8. Checklist de Entrega

Artefatos Obrigatórios

✅ **Link do repositório GitHub** com código fonte
✅ **Instruções completas** de instalação e teste
✅ **Documentação técnica** (este documento)
✅ **Swagger UI** disponível e funcional
✅ **Vídeo demonstração** (10 min) - https://drive.google.com/drive/folders/1PJyNY5-8_6t6FvjFfC2lUOj1U-lH30C_?usp=drive_link
✅ **Vídeo pitch** (3 min) - https://drive.google.com/drive/folders/1PJyNY5-8_6t6FvjFfC2lUOj1U-lH30C_? usp=drive_link

Requisitos Técnicos

✅ API REST implementada (40+ endpoints)
✅ JPA/Hibernate configurado (6 entidades)

✅ Bean Validation (24 DTOs validados)
✅ Paginação (todos os GETs)
✅ Swagger/OpenAPI (documentação completa) ✅ JWT Authentication (login + token)
✅ Deploy (Dockerfile pronto)

Qualidade do Código

✅ Código organizado em camadas
✅ Nomenclatura em inglês
✅ Tratamento de exceções global
✅ CORS configurado
✅ Logs implementados
✅ Senhas criptografadas (BCrypt)

⬜⬜ 9. Próximos Passos e Melhorias Melhorias Futuras (v2.0)

1. **Frontend React/Angular**
   - Interface web completa
   - Dashboard administrativo
   - Portal do aluno/professor
2. **Funcionalidades Adicionais**

Sistema de pagamentos
Notificações push
Chat em tempo real
Relatórios analíticos

3. **Integrações**

Gateway de pagamento
Email (SendGrid/AWS SES)
SMS (Twilio)
Calendário (Google Calendar)

4. **DevOps**

CI/CD (GitHub Actions)
Deploy automatizado (AWS/Azure)
Monitoramento (Prometheus/Grafana)
Testes automatizados (JUnit 5)

5. **Performance**

Cache (Redis)
CDN para assets

Load balancing
Database indexing

?? 10. Contato e Suporte

Desenvolvedor

Nome: Arthur
RM: 560820
Email: [SEU EMAIL]
GitHub: https://github.com/arthur33b
LinkedIn: [SEU LINKEDIN]

Repositório

Issues: https://github.com/arthur33b/CAPP-Java_globalSolution/issues
Pull Requests:
https://github.com/arthur33b/CAPP-Java_globalSolution/pulls **Wiki:**
https://github.com/arthur33b/CAPP-Java_globalSolution/wiki

?? 11. Licença e Uso

Este projeto foi desenvolvido como parte do desafio Global Solution da FIAP.

Licença: MIT License
Ano: 2025
Instituição: FIAP

✅ 12. Conclusão

A **CAPP API** é uma solução completa e robusta para gestão de academias e centros esportivos, implementando todos os requisitos técnicos solicitados:

- ✅ API REST com 40+ endpoints
- ✅ Persistência JPA/Hibernate
- ✅ Bean Validation completa
- ✅ Paginação em todos os recursos
- ✅ Documentação Swagger/OpenAPI
- ✅ Autenticação JWT segura
- ✅ Pronto para deploy (Docker)

O projeto demonstra conhecimento sólido em:

- **Spring Boot ecosystem**
- **JPA e Hibernate**
- **Security e JWT**
- **RESTful APIs**
- **Oracle Database**

- **Boas práticas de desenvolvimento**

Data de Entrega: 23/11/2025
Versão do Documento: 1.0
Status: ✅ Completo e Funcional