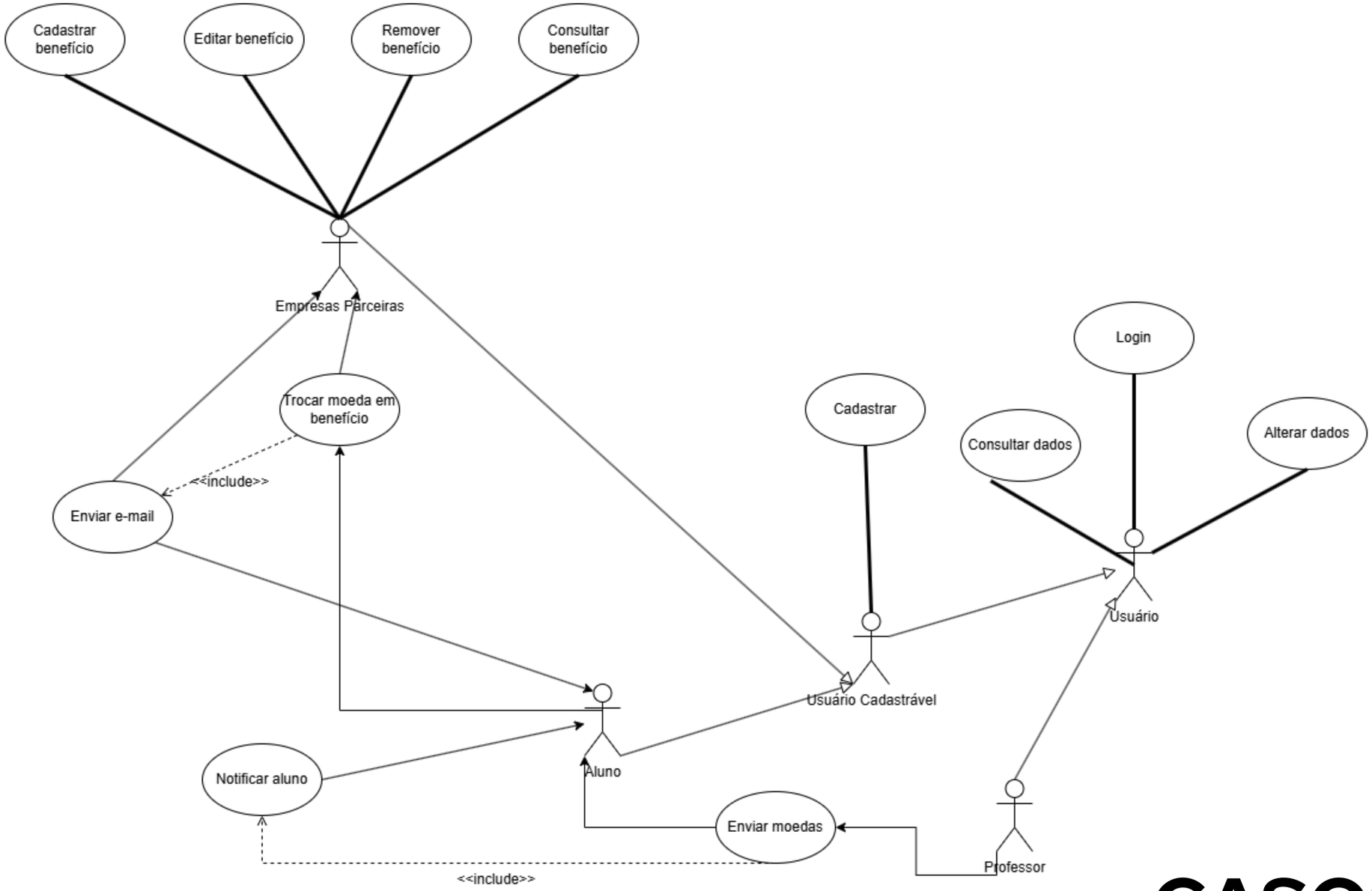
PUC MINAS - LAB DESENVOLVIMENTO SOFTWARE

SISTEMA DE MOEDA ESTUDANTIL

GRUPO : ATHOS FONSECA, MATEUS ARAUJO, RAFAEL GANASCINI


Cadastrar benefício
Editar benefício
Remover benefício
Consultar benefício
Empresas Parceiras
Trocar moeda em benefício
<<include>>
Enviar e-mail
Login
Cadastrar
Consultar dados
Alterar dados
Usuário
Usuário Cadastrável
Aluno
Notificar aluno
Enviar moedas
Professor
<<include>>

CASOS DE USO

PUC MINAS

TECNOLOGIAS UTILIZADAS

O conjunto foi escolhido por oferecer tecnologias já conhecidas, robustez, modularidade e integração facilitada para APIs REST modernas.

INTERFACE WEB

SWAGGER UI – HTML/CSS/JS (A ser implementado)

BACK END

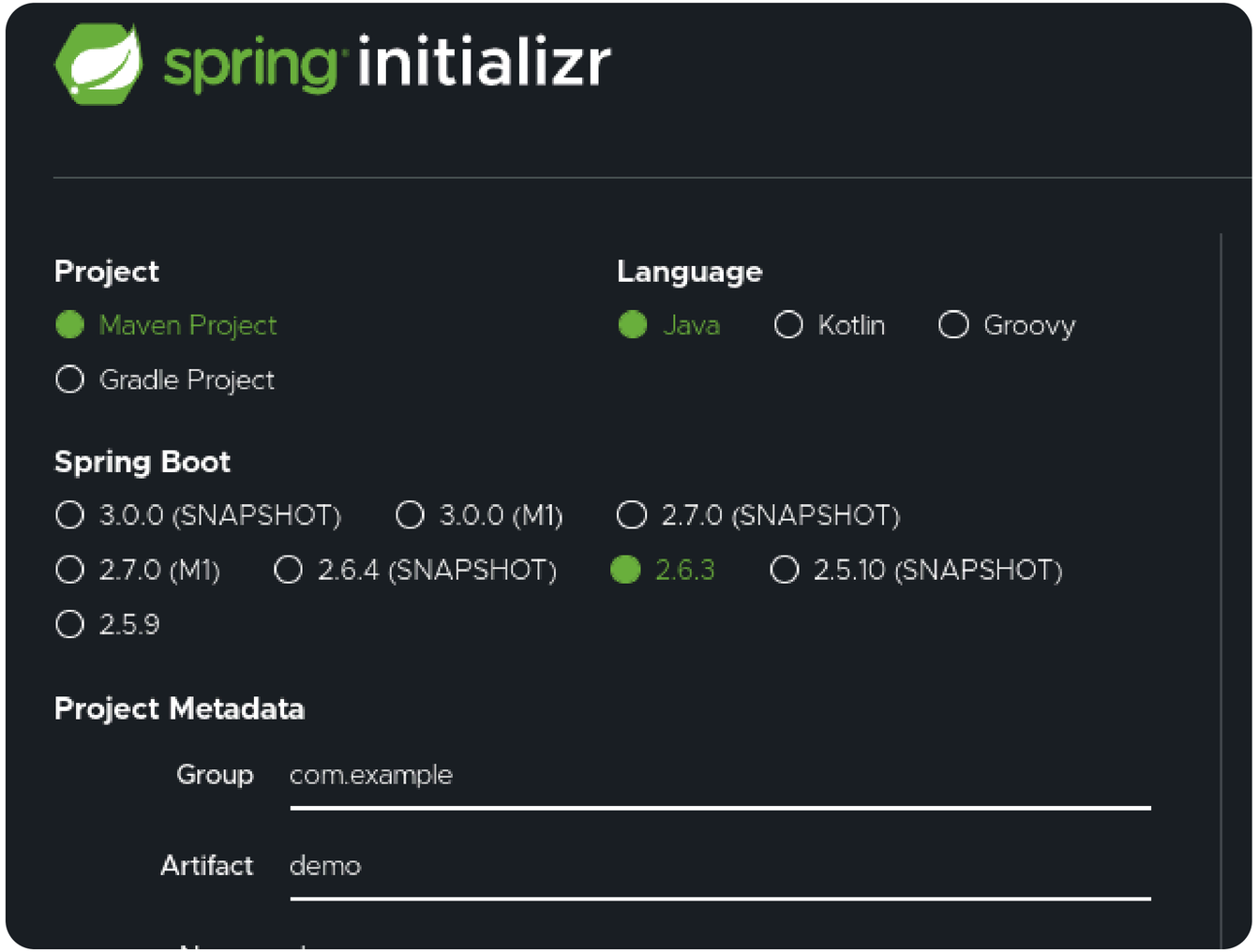
Java - Spring Boot + Lombok

BANCO DE DADOS

MySQL

ORM + REPOSITORY

JPA/Hibernate



SPRING BOOT

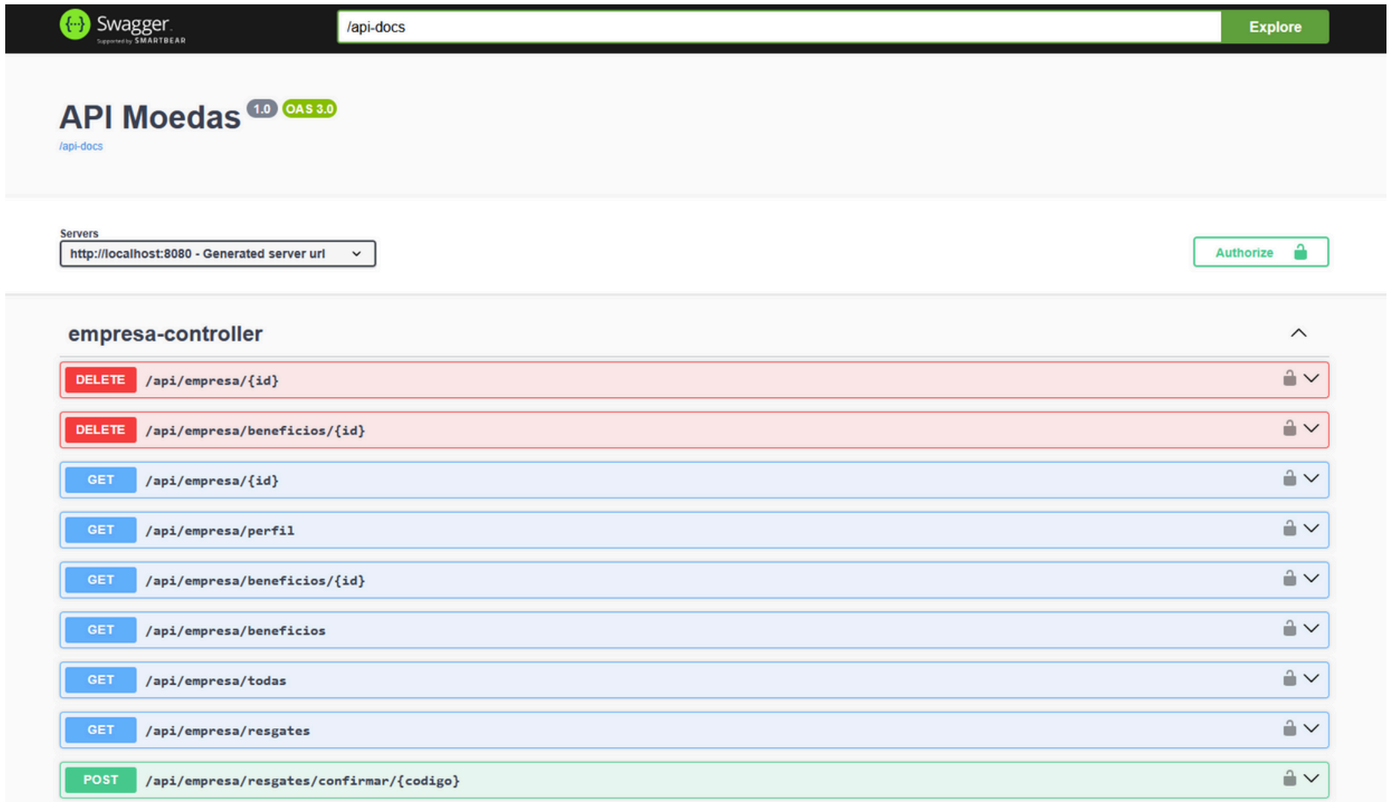
Por que escolhemos:

- Integra nativamente com Spring Data, Security, JPA e Swagger.
- Na faculdade utilizamos Java constantemente, é uma linguagem familiar.
- O Spring Boot é o principal framework moderno Java para APIs REST.
- Já vem com servidor embutido (Tomcat).

SWAGGER

Por que escolhemos:

- Gera documentação automática e interativa da API.
- Atua como a View da arquitetura
- Facilita testes sem precisar de Postman.
- Ajuda na validação da arquitetura durante o desenvolvimento.



MYSQL

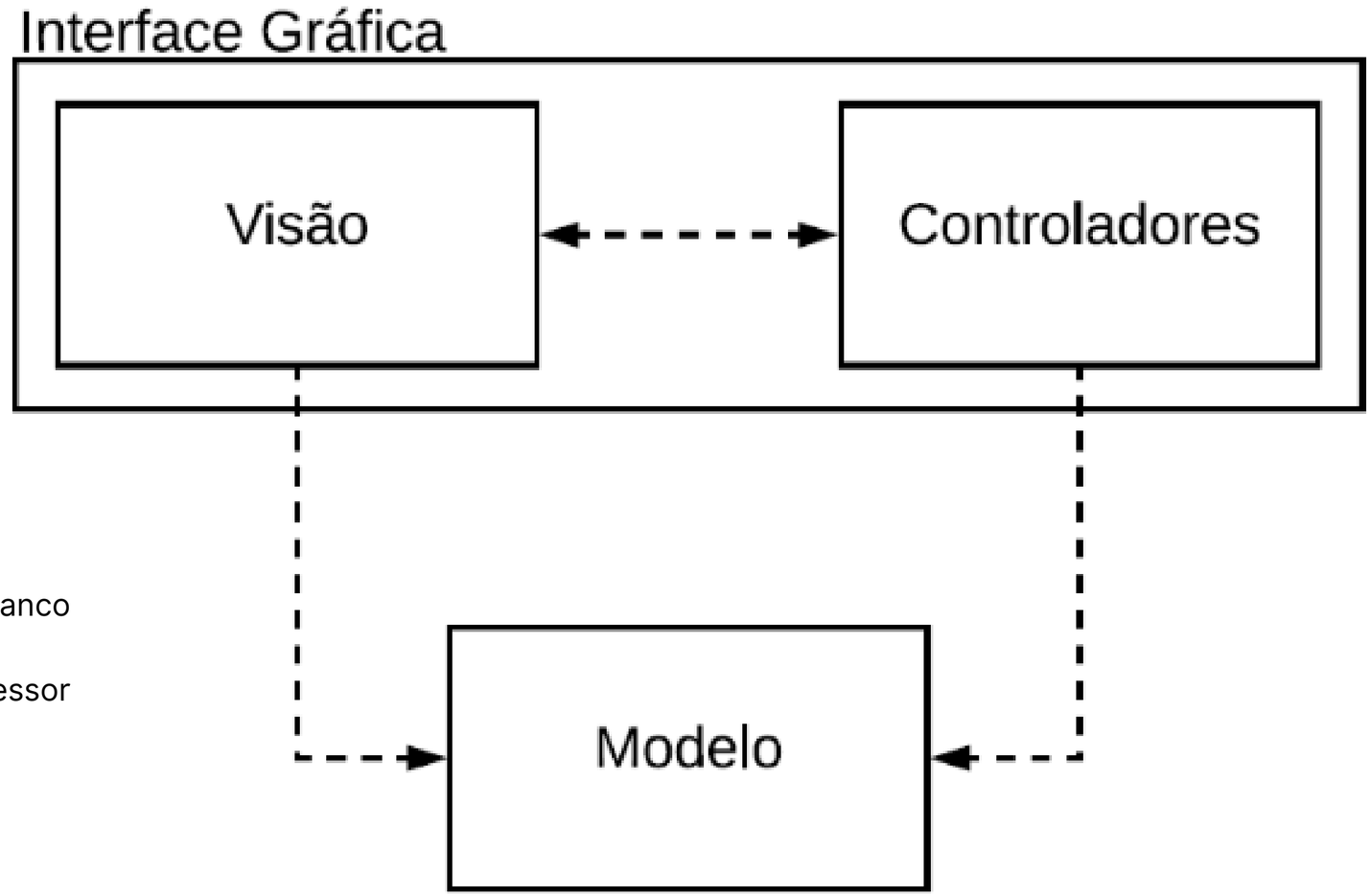
Por que escolhemos:

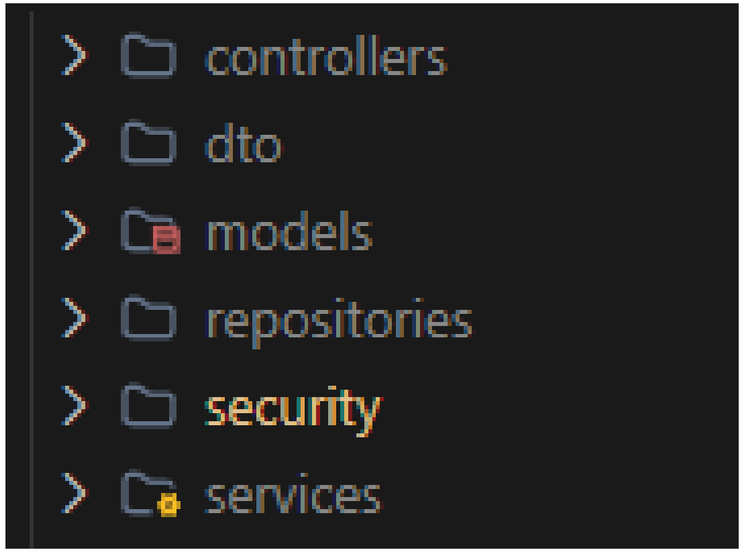
- Banco relacional popular e gratuito.
- Fácil integração com o Spring Boot.
- Compatível com o Hibernate.
- Já utilizado em outras disciplinas da faculdade.

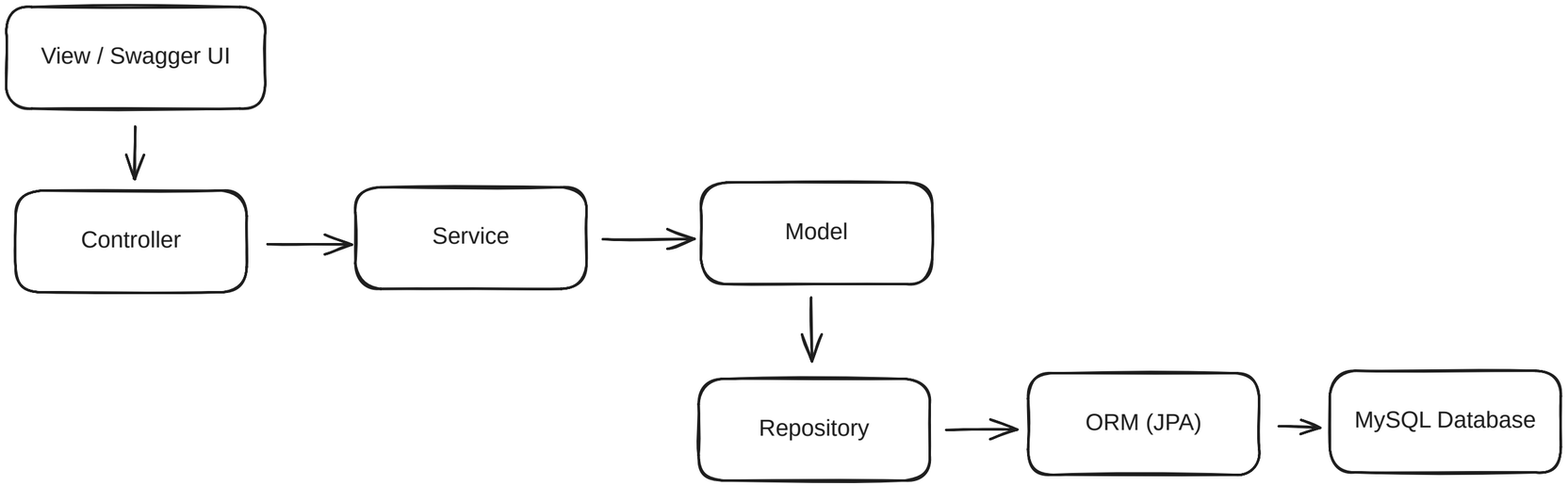
ARQUITETURA E PADRÃO MVC

Camadas Principais:

- **Controller:** recebe requisições HTTP (endpoints REST).
- **Service:** contém as regras de negócio e validações.
- **Repository:** realiza acesso e manipulação de dados no banco (Spring Data JPA).
- **Model:** representa as entidades do sistema (Aluno, Professor etc)







ARQUITETURA (VISUAL)

- **Model**: Entidades JPA (Aluno, Professor, Empresa, Transação, etc).
- **View:** Swagger UI — interface de teste/documentação da API.
- **Controller:** Camada que recebe requisições REST.
- **Service:** Contém regras de negócio.

PUC MINAS

CAMADA DE PERSISTÊNCIA

REPOSITORY + ORM + BANCO

- **Repository:** interfaces que estendem JpaRepository.
- **ORM (Hibernate):** converte objetos Java em SQL.
- **Banco MySQL:** armazena os dados persistidos.


```java
public interface AlunoRepository extends JpaRepository<Aluno, Long> {
  Optional<Aluno> findByEmail(String email);

  Optional<Aluno> findByCpf(String cpf);

  default Aluno saveAluno(Aluno aluno) {
    return save(aluno);
  }

  default Aluno findAlunoById(Long id) {
    return findById(id).orElse(null);
  }

  default void deleteAlunoById(Long id) {
    deleteById(id);
  }

  default List<Aluno> findAllAlunos() {
    return findAll();
  }
}
```

PUC MINAS

POR QUE REPOSITORY E NÃO “DAO”?

Escolha da Camada de Persistência: Repository Pattern

- **O Repository** é uma abstração mais moderna e integrada ao Spring Boot.
- **DAO (Data Access Object)** exige mais código manual (queries, conexões, etc.).
- **O Repository:**
- Reduz boilerplate com métodos prontos (findAll, save, deleteById);
- Integra facilmente com o Spring Data JPA e o ORM Hibernate;
- Permite consultas automáticas com base em nomes de métodos (findByEmail, findByNome);
- É testável e facilita manutenção.