# Automatizando Decisões Educacionais com IBM ODM

A educação é um dos pilares fundamentais para o desenvolvimento de qualquer sociedade.

Com o avanço da tecnologia, novos caminhos foram abertos para facilitar a disseminação do conhecimento e o aprimoramento de processos educacionais.

O projeto **ODM EducaCiência** foi concebido com esse objetivo: criar uma solução educacional que utilize tecnologia de ponta para promover aprendizado personalizado, otimizado e acessível.

## Propósito do Projeto

O principal propósito do **ODM EducaCiência** é fornecer uma plataforma escalável e flexível que permita a implementação de regras de negócios educacionais complexas.

Essas regras incluem critérios de avaliação, aprovação, recomendações personalizadas e feedbacks direcionados, tudo de forma transparente e fácil de entender, tanto para técnicos quanto para educadores.

Este projeto se destaca por:

- Tornar as **regras de negócio acessíveis e compreensíveis** para usuários não técnicos, utilizando linguagem natural.
- Garantir **transparência** e **manutenção facilitada** das regras de avaliação e recomendações.
- Oferecer uma estrutura modular e extensível, permitindo que instituições de ensino adaptem o sistema às suas necessidades específicas.

## Por que o IBM ODM?

O **IBM Operational Decision Manager (ODM)** é uma plataforma robusta de gerenciamento de regras de negócios e automação de decisões que traz diversas vantagens:

1. **Separação entre Regras e Código**, permitindo ajustes rápidos sem necessidade de alterar o sistema.

2. **Verbalização em Linguagem Natural**, facilitando o entendimento para usuários não técnicos.
3. **Gerenciamento de Decisões Complexas**, ideal para cenários educacionais dinâmicos.
4. **Escalabilidade e Integração** com outros sistemas educacionais e institucionais.
5. **Facilidade de Manutenção**, essencial em um ambiente educacional dinâmico.

## *Passo 1: Criando o Modelo de Dados (XOM)*

*(Consulte a seção completa com as classes Student, Subject e GradeCalculator já descritas anteriormente.)*

### *Classe Student*

```
package com.odm.educaciencia.model;

public class Student {
    private String name;
    private int age;
    private String gradeLevel;
    private double averageGrade;

    public Student(String name, int age, String gradeLevel) {
        this.name = name;
        this.age = age;
        this.gradeLevel = gradeLevel;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getGradeLevel() {
        return gradeLevel;
    }

    public void setGradeLevel(String gradeLevel) {
        this.gradeLevel = gradeLevel;
    }

    public double getAverageGrade() {
        return averageGrade;
    }

    public void setAverageGrade(double averageGrade) {
```

```
        this.averageGrade = averageGrade;
    }
}
```

### **Classe** *Subject*

```
package com.odm.educaciencia.model;

public class Subject {
    private String name;
    private int credits;

    public Subject(String name, int credits) {
        this.name = name;
        this.credits = credits;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getCredits() {
        return credits;
    }

    public void setCredits(int credits) {
        this.credits = credits;
    }
}
```

### **Classe** *GradeCalculator*

```
package com.odm.educaciencia.utils;

import java.util.Map;

public class GradeCalculator {
    public static double calculateAverage(Map<String, Double> grades) {
        return grades.values().stream().mapToDouble(Double::doubleValue).average().orElse(0.0);
    }
}
```

### **Classe** *Attendance*

```
package com.odm.educaciencia.model;

public class Attendance {
    private double percentage;

    public Attendance(double percentage) {
        this.percentage = percentage;
    }

    public double getPercentage() {
        return percentage;
    }
```

```
    public void setPercentage(double percentage) {
        this.percentage = percentage;
    }
}
```

### *Classe StudentStatus*

```
package com.odm.educaciencia.model;

public class StudentStatus {
    private String status;
    private String action;

    public StudentStatus(String status, String action) {
        this.status = status;
        this.action = action;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public String getAction() {
        return action;
    }

    public void setAction(String action) {
        this.action = action;
    }

    @Override
    public String toString() {
        return "Status: " + status + ", Ação: " + action;
    }
}
```

## Passo 2: Verbalizando e Modelando o BOM

No IBM ODM, verbalize as regras de negócio com base nos atributos do modelo de dados. Exemplos:

- *"Se a média do estudante for maior ou igual a 7, ele está aprovado."*
- *"Se o estudante tiver menos de 18 anos e estiver reprovado, notificar os responsáveis."*
- *"Se a média do estudante for menor que 5, ele será reprovado e receberá reforço acadêmico."*

# Passo 3: Implementando Cenários

## Cenário 1: Estudante com desempenho abaixo do esperado

**Regra**: Se a média for inferior a 5, além de reprovar, o sistema deve recomendar reforço acadêmico.

**Exemplo de Output**:

Estudante: Lucas
Idade: 17
Média: 4.5
Status: Reprovado
Ação: Recomendado reforço acadêmico

## Cenário 2: Estudante elegível para mérito acadêmico

**Regra**: Estudantes com média superior a 9 e que completaram todas as disciplinas obrigatórias devem ser indicados ao programa de mérito.

**Exemplo de Output**:

Estudante: Maria
Idade: 20
Média: 9.8
Status: Excelente desempenho
Ação: Indicado ao programa de mérito acadêmico

## Cenário 3: Estudante próximo à aprovação

**Regra**: Se a média estiver entre 6 e 6.9, o estudante deve receber uma mensagem de incentivo.

**Exemplo de Output**:

Estudante: João
Idade: 18
Média: 6.5
Status: Próximo à aprovação
Ação: Enviar mensagem de incentivo

## Cenário 4: Estudante com faltas excessivas

**Regra**: Estudantes com frequência inferior a 75% devem ser reprovados, independentemente da média.

**Exemplo de Output**:

Idade: 19
Média: 8.0


Frequência: 70%
Status: Reprovado
Ação: Reprovação por frequência insuficiente

### Cenário 5: Estudante reprovado com apoio adicional

**Regra**: Estudantes com média entre 4 e 4.9 devem ser reprovados, mas terão direito a aulas de reforço gratuitas.

**Exemplo de Output**:

Estudante: Felipe
Idade: 15
Média: 4.4
Status: Reprovado
Ação: Aulas de reforço gratuitas oferecidas

## Passo 4: Testes Unitários

### Testando os Novos Cenários

Adicione os seguintes casos de teste:

```
@Test
public void testLowAttendance() {
    Student student = new Student("Ana", 19, "Ensino Superior");
    student.setAverageGrade(8.0);
    double attendancePercentage = 70.0;

    assertEquals("Reprovado", calculateStudentStatus(student, attendancePercentage));
}

@Test
public void testNearApproval() {
    Student student = new Student("João", 18, "Ensino Médio");
    student.setAverageGrade(6.5);
    double attendancePercentage = 90.0;

    assertEquals("Próximo à aprovação", calculateStudentStatus(student,
attendancePercentage));
}
```

### Outputs dos Testes

[INFO] Running com.odm.educaciencia.GradeCalculatorTest
[INFO] Teste 1: testLowAttendance - Sucesso
[INFO] Teste 2: testNearApproval - Sucesso
[INFO] Teste 3: testExcellentPerformance - Sucesso
[INFO] Teste 4: testLowPerformance - Sucesso

## Referência para Mais Aprendizado

Se você deseja explorar uma implementação mais robusta e aprofundada sobre a criação de regras educacionais com o IBM ODM, confira o repositório do projeto **ODM EducaCiência Template** no GitHub. Ele contém exemplos práticos e recursos adicionais para você começar:

📎 https://github.com/perucello/ODM_EducaCiencia_Template

O projeto **ODM EducaCiência** mostrou como o IBM ODM pode transformar a maneira como decisões educacionais são automatizadas e gerenciadas.

Ao longo deste artigo, exploramos como estruturar regras de negócios, criar cenários práticos e testar decisões automatizadas com eficiência.

Este guia é apenas o início de um aprendizado mais profundo. Cada cenário apresentado reflete situações reais que ocorrem em instituições de ensino, destacando a flexibilidade do IBM ODM para lidar com as particularidades de cada caso.

Se você é um desenvolvedor, educador ou gestor, encorajamos você a estudar este material com atenção.

Teste o código, experimente diferentes cenários e adapte as regras para as suas necessidades. Além disso, o repositório ODM EducaCiência Template é um excelente recurso para se aprofundar e aplicar o que foi aprendido aqui de forma prática e robusta.

Lembre-se: dominar ferramentas como o IBM ODM não é apenas uma vantagem técnica, mas também uma oportunidade de trazer inovação para a educação, promovendo decisões justas, ágeis e personalizadas.

***EducaCiência FastCode para a comunidade***