UNIVERSIDADE DO ESTADO DA BAHIA
Departamento de Ciências Exatas e da Terra, Campus I
Colegiado de Sistemas de Informação / **Semestre:** 2025.2
Disciplina: Linguagem de Programação III / **Professor:** Vagner Fonseca

Avaliação IV
Desenvolvimento de Sistemas em Arquitetura MVC

O trabalho final da disciplina tem como objetivo avaliar a capacidade dos alunos em planejar, implementar um sistema web completo utilizando o padrão arquitetural MVC (Model-View-Controller) com Spring Boot e Angular (ou simlar). O foco está na separação clara entre as camadas de dados (Model), lógica de controle (Controller) e interface do usuário (View), aplicando boas práticas de desenvolvimento de software.

Organização e Formação dos Grupos

- **Data de apresentação:** A definir (durante o horário regular de aula)
- **Composição:** Serão formados até 5 grupos com no máximo 5 alunos cada
- **Escolha do tema:** Cada grupo deverá escolher um dos temas listados abaixo e registrar o nome dos integrantes na planilha compartilhada no TEAMS
- **Regra de formação:** O primeiro aluno a escolher um tema será o responsável por agregar novos membros ao grupo até atingir o limite máximo
- **Importante:** Cada tema só poderá ser escolhido por um único grupo. A escolha é por ordem de chegada.

Sistema de Avaliação dos Grupos

- 50 pontos por grupo, distribuídos conforme barema técnico detalhado disponível na plataforma TEAMS.
- Após a apresentação e correção do trabalho, o grupo deverá realizar a divisão interna dos pontos entre os membros, considerando a contribuição individual de cada um no desenvolvimento do projeto.
- A soma dos pontos individuais deve ser exatamente igual à pontuação total recebida pelo grupo
- A distribuição deve refletir de forma justa o esforço e dedicação de cada integrante
- Um ou mais membros do grupo devem informar ao professor, por escrito, a divisão final dos pontos

Cada grupo deverá selecionar um dos temas abaixo. Todos os temas possuem complexidade equivalente e requisitos funcionais bem definidos:

- **Loja Virtual "E-commerce"** (Cadastro de produtos, usuários e controle de pedidos)
    - Requisitos Funcionais:
        - Cadastro completo de produtos (nome, descrição, preço, estoque, categoria, imagem)
        - Sistema de categorização de produtos com filtros
        - Cadastro e autenticação de usuários (cliente e administrador)
        - Carrinho de compras com adição/remoção de itens
        - Cálculo automático do total da compra (subtotal + frete simulado)

        - Finalização de pedido com validação de estoque
        - Histórico de pedidos por cliente
        - Painel administrativo para gerenciar produtos e pedidos
        - Atualização automática de estoque após confirmação do pedido
        - Sistema de busca de produtos por nome ou categoria
        - Visualização detalhada do produto
        - Gestão de status do pedido (Pendente, Processando, Enviado, Entregue, Cancelado)
    - Requisitos Não Funcionais:
        - Interface responsiva para mobile, tablet e desktop
        - Tempo de resposta máximo de 2 segundos para operações de busca
        - Validação de campos obrigatórios com mensagens claras de erro
        - Senha deve ter no mínimo 8 caracteres
        - Sistema deve suportar até 100 produtos simultâneos em memória
        - Feedback visual para todas as ações do usuário (loading, sucesso, erro)
        - Código organizado seguindo padrão MVC estrito

- **Sistema de Reservas de Eventos** (Gerenciamento de reservas de salas, auditórios ou eventos)
    - Requisitos Funcionais:
        - Cadastro de locais (salas, auditórios) com capacidade e recursos disponíveis
        - Cadastro e autenticação de usuários
        - Criação de reserva com data, horário inicial e final, e local
        - Validação de conflitos de horário (não permitir reservas simultâneas no mesmo local)
        - Listagem de reservas por usuário
        - Listagem de reservas por local
        - Calendário visual mostrando disponibilidade dos locais
        - Cancelamento de reserva
        - Edição de reserva (com revalidação de conflitos)
        - Filtro de reservas por período (dia, semana, mês)
        - Sistema de notificação de conflitos
        - Histórico de todas as reservas realizadas
    - Requisitos Não Funcionais:
        - Algoritmo eficiente de detecção de conflitos (tempo < 1 segundo)
        - Interface intuitiva para visualização do calendário
        - Validação de datas (não permitir reservas em datas passadas)
        - Horários devem seguir intervalos de 30 minutos
        - Sistema deve manter consistência mesmo com múltiplas tentativas simultâneas
        - Mensagens de erro descritivas para conflitos
        - Responsividade em diferentes dispositivos

- **Rede Social para Alunos** (ambiente para criação de perfis, postagens e interações entre usuários)
    - Requisitos Funcionais:
        - Cadastro de perfil com foto, biografia, curso e período
        - Autenticação de usuários
        - Criação de postagens (texto, imagem opcional)
        - Feed dinâmico ordenado por data (mais recentes primeiro)
        - Sistema de curtidas em postagens
        - Sistema de comentários em postagens
        - Edição e exclusão de postagens próprias
        - Visualização de perfil de outros usuários
        - Busca de usuários por nome ou curso
        - Contagem de curtidas e comentários por postagem
        - Filtro de postagens por usuário
        - Timeline pessoal mostrando apenas postagens próprias
    - Requisitos Não Funcionais:
        - Feed deve carregar em menos de 2 segundos
        - Atualização em tempo real das curtidas e comentários
        - Limite de 500 caracteres por postagem
        - Imagens devem ser validadas (formato e tamanho máximo)
        - Interface similar a redes sociais conhecidas (UX intuitiva)
        - Responsividade total (mobile-first)
        - Sistema deve suportar até 50 usuários simultâneos
        - Validação de conteúdo ofensivo (palavras-chave básicas)
- **Sistema de Gestão de Tarefas** (controle de tarefas por status e atribuição a diferentes usuários)
    - Requisitos Funcionais:
        - Cadastro de tarefas com título, descrição, prioridade e prazo
        - Cadastro de usuários com perfis (Administrador, Membro)
        - Atribuição de tarefas a usuários específicos
        - Alteração de status da tarefa (A Fazer, Em Progresso, Concluída, Cancelada)
        - Filtros por status, prioridade e usuário responsável
        - Ordenação de tarefas por prioridade e prazo
        - Dashboard com visão geral das tarefas
        - Histórico de alterações de cada tarefa
        - Sistema de notificação de tarefas próximas do prazo
        - Criação de projetos para agrupar tarefas
        - Marcação de tarefas como urgentes
        - Comentários e atualizações em tarefas
    - Requisitos Não Funcionais:
        - Interface drag-and-drop para mudança de status (estilo Kanban)
        - Código cores por prioridade (Alta=Vermelho, Média=Amarelo, Baixa=Verde)
        - Validação de datas (prazo não pode ser anterior à criação)
        - Dashboard deve carregar todas as tarefas em menos de 1 segundo
        - Histórico deve registrar usuário, data e tipo de alteração

        - Sistema deve alertar tarefas com prazo vencido
        - Responsividade completa
        - Backup automático do estado das tarefas em LocalStorage

- **Plataforma de Troca de Produtos** (sistema de cadastro e troca de itens entre usuários)
    - Requisitos Funcionais:
        - Cadastro de produtos com foto, descrição, categoria e condição (Novo, Seminovo, Usado)
        - Cadastro e autenticação de usuários
        - Listagem de produtos disponíveis para troca
        - Sistema de busca por categoria e palavra-chave
        - Criação de proposta de troca (produto A por produto B)
        - Visualização de propostas recebidas
        - Aceitar ou recusar propostas
        - Atualização automática do status dos produtos (Disponível, Em negociação, Trocado)
        - Histórico de trocas realizadas
        - Avaliação dos usuários após troca concluída
        - Visualização de perfil com histórico de trocas
        - Sistema de favoritos para produtos de interesse
    - Requisitos Não Funcionais:
        - Propostas devem ser processadas com confirmação de ambas as partes
        - Validação de produtos (não permitir troca de produtos já trocados)
        - Interface visual clara mostrando status das negociações
        - Sistema de notificação de novas propostas
        - Fotos devem ter tamanho máximo de 2MB
        - Avaliações devem ser de 1 a 5 estrelas
        - Responsividade em todos os dispositivos
        - Transações devem ser atômicas (ou ambos os produtos mudam de status, ou nenhum)

- **Plataforma de E-learning** (sistema de cursos online com controle de alunos e progresso)
    - Requisitos Funcionais:
        - Cadastro de cursos com título, descrição, instrutor e carga horária
        - Cadastro de módulos dentro de cada curso
        - Cadastro de aulas dentro de cada módulo (título, conteúdo, vídeo/link)
        - Cadastro e autenticação de usuários (Aluno, Instrutor)
        - Matrícula de alunos em cursos
        - Marcação de aulas como concluídas
        - Cálculo automático de progresso do aluno (% de conclusão)
        - Dashboard do aluno com cursos em andamento
        - Certificado de conclusão ao finalizar 100% do curso
        - Sistema de avaliação de cursos pelos alunos

        - Painel do instrutor para gerenciar seus cursos
        - Fórum de dúvidas por curso
    - Requisitos Não Funcionais:
        - Progresso deve ser salvo automaticamente
        - Barra de progresso visual em cada curso
        - Validação de sequência (não pular módulos)
        - Interface amigável e intuitiva para navegação
        - Responsividade total para estudo mobile
        - Sistema deve suportar múltiplos alunos assistindo simultaneamente
        - Certificado deve ser gerado em formato PDF ou imagem
        - Tempo de carregamento de aulas < 2 segundos
        - Sistema de busca eficiente de cursos

Requisitos Técnicos

1. Arquitetura MVC
    a. Estrutura MVC obrigatória, com as três camadas (Model, View e Controller) claramente separadas e organizadas em pacotes/diretórios distintos.
2. Persistência de Dados
    a. Utilização de no mínimo um banco de dados em memória (H2, HSQLDB ou similar) com JPA/Hibernate para mapeamento objeto-relacional ou qualquer outro banco relacional como exemplo MariaDB, MySQL, SQL Server, Oracle.
    b. Implementação de relacionamentos entre entidades quando aplicável (@OneToMany, @ManyToOne, @ManyToMany).
    c. Carga de dados iniciais para testes (mínimo de 5 registros por entidade principal).

3. Operações CRUD
    a. Implementação completa de CRUD (Create, Read, Update, Delete) para no mínimo duas entidades principais do sistema.
    b. Criação de endpoints REST no backend seguindo o padrão RESTful.
    c. Integração completa entre frontend e backend para todas as operações.

4. Camada de Controle
    a. Implementação de Controllers REST no backend responsáveis por receber requisições HTTP e delegar para a camada de Service.
    b. Criação de Services contendo toda a lógica de negócio da aplicação.
    c. Utilização de Repositories apenas para acesso aos dados.
    d. No frontend, implementação de Services Angular ou qualquer outro framework de sua preferência para comunicação com o backend via HTTP.

5. Interface Visual Web
    a. Desenvolvimento de interface web utilizando Angular ou qualquer outro framework de sua preferência com navegação entre páginas através de rotas.

b. Criação de formulários reativos (Reactive Forms) para cadastro e edição de dados.
c. Interface responsiva que se adapte a diferentes tamanhos de tela (mobile, tablet e desktop).
d. Utilização de framework CSS (Bootstrap, Angular Material ou Tailwind CSS) ou do framework de sua escolha.
e. Feedback visual para todas as ações do usuário (loading, mensagens de sucesso e erro).

6. Validação e Tratamento de Erros
    a. Validação de campos obrigatórios e formato de dados tanto no frontend (experiência do usuário) quanto no backend (segurança).
    b. Utilização de Bean Validation no backend (@NotNull, @Size, @Min, @Email, etc.).
    c. Implementação de Exception Handler global no backend para tratamento centralizado de erros.
    d. Mensagens de erro claras e em português exibidas ao usuário.
    e. Tratamento adequado de erros HTTP com códigos de status apropriados (200, 201, 400, 404, 500).
    f. Prevenção de submissão de formulários inválidos no frontend com indicadores visuais nos campos com erro.