UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

ESCOLA AGRÍCOLA DE JUNDIAÍ

Disciplina: Programação Web
Professor: Taniro Rodrigues
Estudante: _____
Data: 25 de julho 2024 Nota: _____ de 10,0 pontos

Segunda Avaliação

INSTRUÇÕES:

a) A avaliação em duplas.
*b) **Escolha o tema** da sua prova através do Link abaixo. Os temas devem ser únicos entre os estudantes. Coloque seu nome ao lado do tema que você escolheu para que os colegas saibam quais temas estão disponíveis.*
https://docs.google.com/spreadsheets/d/1i_tyxHoB491KcCIBAELYCqRHdM8N5KToz0PBNZw8Pak/edit?usp=sharing
*c) **Defina os atributos** do seu tema. Cada tema deve ser especificado como uma **classe com pelo menos 7 atributos**, alguns atributos serão: ID (Long), isDeleted (Date), ImageUri(String)*
*d) Crie um projeto no github (ou similar) para enviar o seu código de resposta. **Se necessário** adicione comentários no código para explicar ou justificar sua resposta.*
*e) **Grave um vídeo apresentando cada uma das suas respostas**. A gravação do vídeo é **obrigatória** e a correção da questão está condicionada ao conteúdo do vídeo. **NÃO É NECESSÁRIO EXPLICAR SUAS RESPOSTAS NO VÍDEO. BASTA UMA DEMONSTRAÇÃO DE QUE FUNCIONOU.***
*f) O vídeo deve ter, no máximo, **3 minutos**. Lembre-se **é para mostrar que cada questão funcionou**. **Não precisa explicar código**. Deixa que eu leio :-)*
*d) Envie o **vídeo, o link do projeto no github** para a atividade do **SIGAA**.*
e) Atenção, é mandatório que o projeto seja executado e demonstrado em vídeo. Sua prova só será corrigida se esse item for atingido.
f) A prova vale 10,0 e deve ser enviada até 27/06/2024 às 23:59

QUESTÕES:

1) Crie um projeto com Spring Initializr incluindo Spring Boot Dev Tools, Lombok, Spring Web, Thymeleaf, Spring Data JPA, PostgreSQL Driver e Spring Validator. Crie a classe do modelo conforme o tema escolhido, lembre-se que você precisa adicionar pelo menos 7 atributos (ID, isDeleted, ImageUri). Adicione as restrições (validações) do modelo, por exemplo, não aceitar string em branco, não aceitar campo null, etc. (0,0 ponto)

2) Crie a classe do usuário (id, username, password, isAdmin, etc) que deve implementar a interface UserDetails. Na aplicação os usuários poderão assumir 2 papeis (roles) “ROLE_ADMIN” e “ROLE_USER”. (0,5)

3) Prepare seus templates Thymeleaf para uso de bootstrap. Simplifique suas páginas utilizando fragments para separar, no mínimo, as partes de cabeçalho, principal e rodapé. Você deve utilizar como base o template HTML disponível no link (https://themewagon.com/themes/free-bootstrap-ecommerce-template-electro/) (0,5 ponto)

4) Implemente a rota de (“/index”) para, a partir de uma solicitação do tipo GET, gerar uma resposta contendo no corpo um HTML que contém uma tabela ou similar de todos os itens (linhas) que estão presentes no banco de dados e que não estão deletados (isDeleted == null). Note que a aplicação deve utilizar uma técnica de soft-delete, ou seja, os itens jamais serão deletados de verdade do banco. Os itens são deletados de maneira lógica quando deleted recebe a data atual. Você deve exibir a imagem de cada um dos itens da lista. Para cada item listado adicione um link para a rota “/adicionarCarrinho” passando como parâmetro para tal rota o ID do item escolhido. Por fim, adicione na página gerada pela rota “/index” um link para a rota “/verCarrinho”. Adicione um cookie na resposta chamado “visita” com a data e hora do acesso ao site. Esse cookie deve ser permanente e durar 24hs. (1,0 ponto)

5) Implemente a rota de (“/admin”) para, a partir de uma solicitação do tipo GET, gerar uma resposta contendo no corpo um HTML que contém uma tabela de todos os itens (linhas) que estão presentes no banco de dados e que não estão deletados (isDeleted == null). Para cada item listado adicione um

link para a rota “/editar” e “/deletar” passando como parâmetro para tal rota o ID do item escolhido. Por fim, adicione na página gerada pela rota “/admin” um link para a rota “/cadastro”. (1,0 pontos)

6) Implemente a rota de (“/cadastro”) para, a partir de uma solicitação do tipo GET, gerar uma resposta contendo no corpo um formulário HTML para cadastro de um item do seu tema. O formulário deve conter um input de envio de arquivos para envio da imagem. O formulário deve conter tags para tratamento de erros utilizando o Thymeleaf. O formulário deve enviar os dados da solicitação através do método POST para a rota “/salvar”. (1,0 ponto)

7) Implemente a rota de (“/editar”) para, a partir de uma solicitação do tipo GET, gerar uma resposta contendo no corpo um formulário HTML para edição de um item do seu tema. Os dados do formulário devem estar preenchidos com os dados daquele item no banco. O formulário deve conter tag para tratamento de erros utilizando o Thymeleaf. O formulário deve enviar os dados da solicitação através do método POST para a rota “/salvar”. Ao final do processo, a solicitação deve ser redirecionada para “/admin” enviando uma mensagem de que a atualização ocorreu com sucesso. (1,0 ponto)

8) Implemente a rota de (“/salvar”) que deve receber dados através de método POST e cadastrar ou atualizar um novo item no banco de dados. O método salvar deve validar os atributos do modelo. O método salvar deve criar um nome único para a imagem enviada. Trate a questão de ter apenas uma rota para salvar tanto para o cadastro quanto para edição. Caso ocorra algum erro, a resposta deve ser cancelada e o erro informado no formulário. Ao final do processo, a solicitação deve ser redirecionada para “/admin” enviando uma mensagem de que a atualização ocorreu com sucesso. (0,5 ponto)

9) Implemente a rota de (“/deletar”) que deve receber dados através de método GET e atualizar um item no banco de dados para que o atributo isDeleted contenha a data (Long) atual. Dessa forma, a operação de remoção de um registro será feita através de um *soft delete*, onde o registro do banco de dados não será apagado de fato. Ao final do processo, a solicitação deve ser redirecionada para “/index” enviando uma mensagem de que a remoção ocorreu com sucesso. (0,5 ponto)

10) Implemente a rota de (“/adicionarCarrinho”) que recebe uma solicitação do tipo GET contendo como parâmetro o id de um dos itens. Realize uma busca no banco de dados pelo item a partir do ID e adicione o objeto encontrado em uma Sessão HTTP no atributo “carrinho” que deve conter um ArrayList de itens. Encaminhe a resposta para a rota “/index”. Atualize a página “index” para que seja exibido a quantidade de itens no carrinho. (0,5 ponto)

11) Implemente a rota de (“/verCarrinho”) que ao receber uma solicitação do tipo GET lista todos os itens que estão no atributo “carrinho” da Sessão HTTP. Se o carrinho estiver vazio, redirecione a resposta para “/index” enviando a mensagem de que não existem itens no carrinho. Por fim, adicione um link para a rota “/finalizarCompra”. (0,5 ponto)

12) Implemente a rota de (“/finalizarCompra”) que ao receber uma solicitação do tipo GET invalida a Sessão existente e redireciona a resposta para “index”. (0,5 ponto)

13) Adicione o Spring Security ao seu projeto e implemente o UserDetails Service. Crie uma rota “/cadusuario” para cadastrar usuários na aplicação. O cadastro deve possuir um checkbox para indicar se o usuário é admin ou não. Implemente a rota para salvar o usuário cadastrado. Configure o Spring Security para realizar a autenticação com base no seu UserDetails Service (que busca no banco de dados). Crie páginas personalizadas de login e logout (ou seja, não use as páginas default do Spring security). Utilize BCrypt para codificar a senha. Configure as rotas para que um usuário não logado possa acessar a página de “/login” “/index”. Apenas um usuário com o papel “ROLE_ADMIN” poderá acessar as páginas “/admin” “/cadastro” “/salvar” “/editar” e “/deletar”. Apenas um usuário com o papel “ROLE_USER” poderá acessar as páginas “/vercarrinho”, “/adicionarcarrinho”, “/finalizarcompra”. Adicione o username do usuário logado no cabeçalho da página. Adicione um botão para “logout” no cabeçalho da página. (2,5 pontos)