**BLOCO I**

Escolhas: Quinta (5) e Nona (9) questões

5 – Alguns dos principais desafios que poderemos verificar durante a migração de sistemas legados para uma nova plataforma são:

- Capacitação para uso de novo recurso, desenvolvendo um plano de treinamento sobre os novos recursos e fluxos de atuação. Sendo essencial a produção de uma documentação acessível à todos envolvidos;

- Gerenciamento de riscos e planejamento de contingência, sendo necessário identificar riscos potenciais antecipadamente e desenvolver planos para cada risco identificado. Sendo fundamental, também, definir planos de backup e retorno (“rollback”), para reduzir o tempo de inatividade, inclusive com a realização de testes antes da implantação definitiva (entrar em “produção”). Sendo importante em todas as etapas da mudança atualizar os atores envolvidos.

- Monitoração do consumo de recursos, deveremos estabelecer tetos para uso de recursos, com o intuito de evitar gastos desproporcionais para que a mudança seja realizada;

- Migração de dados, sendo necessário observar a integridade para incorporação dos dados em novo cenário. Deveremos analisar uma estratégia de migração de dados que incluam processos de limpeza, transformação e validação de dados;

- Problemas de compatibilidade, pois sistemas legados normalmente dependem de tecnologias desatualizadas. Sendo necessário refatorar o código, com o intuito de identificar possíveis necessidades de adaptações.

9 – Com o intuito de promover melhoria na comunicação e colaboração de um time, podemos adotar as seguintes medidas:

- Promover os princípios da metodologia Ágil no time, como o Scrum ou Kanban (ou ambos), para que possamos reduzir retrabalho e utilizar o tempo do time de maneira mais eficiente;

- Criar um ambiente colaborativo, para que ocorra um repasse de conhecimento natural entre o time, ocasionando, também, uma melhora da empatia de cada um sobre as dificuldades e facilidades enfrentadas pelo colega no dia a dia.

- Investir em atividades e treinamentos de equipe de forma contínua, com o intuito de aumentar a sinergia.

- Melhorar o processo de desenvolvimento, garantindo que os requisitos sejam documentados de forma clara e completa, evitando interpretações erradas e ambiguidades.

- Nutrir na equipe a ideia de pertencimento, para que seja possível todos se sentirem donos e responsáveis pelo processo ou produto.

- Estabelecer formas de monitoramento e medições tanto individuais, quanto coletivas. Para que todos enxerguem que apenas juntos serão capazes de obter os resultados esperados, ou superá-los.

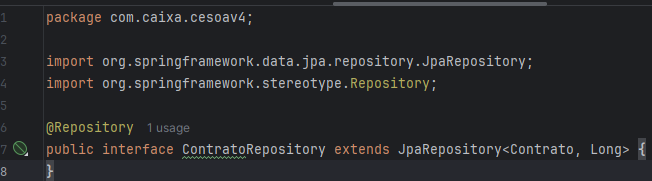
**BLOCO II**

Escolha: Segunda (2) questão

Uma das formas para desenvolvermos um microsserviço em Java usando Spring Boot, para armazenar e recuperar informações sobre contratos de crédito imobiliário, incluindo todas as operações CRUD (Create, Read, Update, Delete) é a seguinte:

I – Deveremos definir a classe de Entidade, que nesse caso, poderá ser através da criação da classe “Contrato.java”, para que possamos estabelecer a estrutura de um contrato de crédito imobiliário. Essa classe será mapeada para uma tabela no banco de dados e, também, incluirá os Getters, Setters, Construtores e Método toString. Devido ao tamanho do código e a limitação de páginas da produção temática, armazenei o código Contrato.java em links apontados na conclusão abaixo.

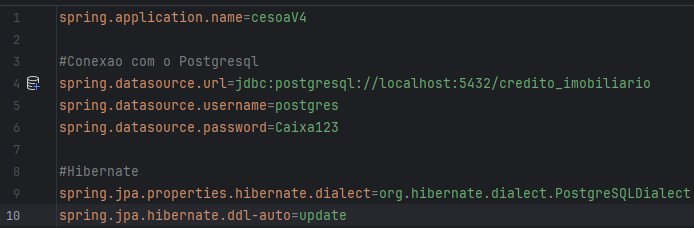
II – Criaremos uma interface de repositório (ou DAO - Data Access Object) Spring Data JPA que gerenciará as operações CRUD para a entidade “Contrato”. Onde denominaremos como ContratoRepository.java, ela estenderá o JpaRepository. O Spring Data JPA irá gerar automaticamente os métodos CRUD necessários com base nas assinaturas de método definidas aqui. Segue:



III – Iremos criar uma camada de serviço, chamada “ContratoService.java”, que encapsulará a lógica de negócio relacionada aos contratos ou orquestrar chamadas a múltiplos repositórios. Devido ao tamanho do código e a limitação de páginas da produção temática, armazenei o código ContratoService.java nos links:

IV – Produziremos o controlador REST que definirá os endpoints para as operações CRUD (GET, POST, PUT, DELETE) e lidará com as requisições HTTP, para manipulação dos contratos via operações CRUD. Sendo aqui a classe “ContratoController.java”. Devido ao tamanho do código e a limitação de páginas da produção temática, armazenei o código ContratoController.java nos links:

V – Será necessário, também, editar as configurações para ligação com o banco de dados, nesse caso, utilizaremos via PostgreSQL, com um projeto em Maven com Spring Boot em Java. As alterações seria realizadas no arquivo application.properties, localizado em “src/main/resources”:



VI – Com a execução da aplicação O Spring Boot irá criar automaticamente as tabelas necessárias no banco de dados com base nos mapeamentos de entidade e na propriedade “ddl-auto” especificada no “application.properties” (etapa V, acima).

Com essas etapas temos um exemplo de uma estrutura básica para criar uma um microsserviço em Java usando Spring Boot para gerenciar contratos de crédito imobiliário em um banco de dado. Podemos verificar os arquivos criados e alterados citados nos links de compartilhamento <https://github.com/caiormaximo/contratos.git> e ., caso fossem colocados neste texto excederia o limite do enunciado (duas páginas).