FIAP – Faculdade de Informática e Administração Paulista
Curso de Tecnologia em Análise em Desenvolvimento de Sistemas (TDS)
Professor: Dr. Marcel Stefan Wagner

Checkpoint 5

Parte 1 (API e Criptografia) – Parte 2 (*Spring Security*)

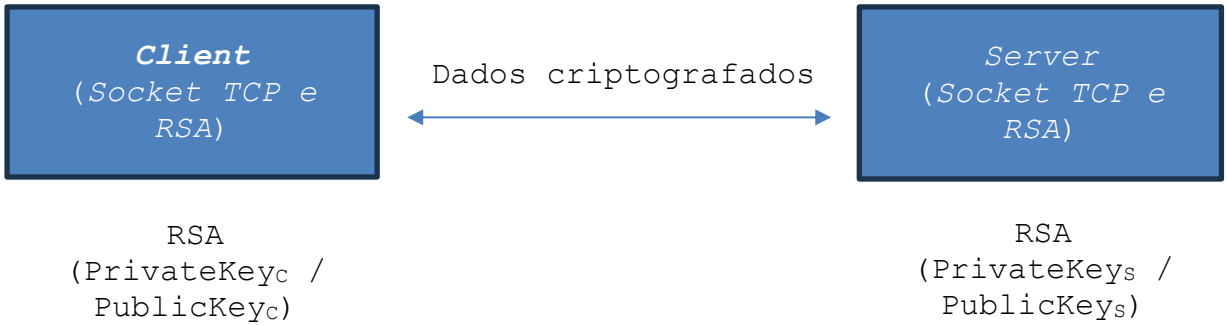
Parte 1 – API *Socket* com Criptografia

Socket TCP com Criptografia RSA

- A entrega deverá ser feita via **Teams** com o envio do arquivo diretamente ao professor (*chat* direto), **não** via *chat* da disciplina, informando a sua turma e curso;
- Deve-se entregar o Projeto em formato **.zip**, somente **um(a) integrante por grupo**, contendo:
    - Um arquivo **.txt** com o **nome e RM de todos(as) os(as) integrantes do grupo**, **o *link* do *GitHub* que possua um *ReadMe* contendo toda a descrição do Projeto** (incluindo imagens e explicações, incluindo exemplos) **e indicar qual o IDE utilizado para elaboração do projeto** (IntelliJ, Eclipse ou NetBeans);
    - A **pasta toda do Projeto**, contendo todas as pastas, subpastas e arquivos.
    - A **planilha Excel (*Dados RSA.xlsx*) contendo toda a configuração que o grupo escolheu para as chaves Pública e Privada** do algoritmo de criptografia RSA.

Para tanto, desenvolva o seguinte aplicativo: Faça um programa aplicativo na linguagem Java que trabalhe na configuração *Client-Server* para envio de mensagens criptografadas entre o Cliente e o Servidor nas duas direções (*Client* ←→ *Server*). O programa deve ter uma classe de Conexão para realizar a transmissão e recepção de dados utilizando o padrão TCP/IP com base em *Sockets* TCP (*Transmission Control Protocol*). A criptografia deve ser feita usando o algoritmo RSA (*Rivest-Shamir-Adleman*) com base em valores de números primos *p* e *q* escolhidos pelo grupo (**conversem com os(as) colegas de sala, pois não poderão ter**

valores iguais de *p* e/ou *q* entre os grupos da turma). Para facilitar, vocês poderão utilizar a planilha Excel *Dados RSA.xlsx* mostrada em aula para determinar os valores do *módulo*, *função totiente*, *expoente* e *inverso multiplicativo*, que deverão obrigatoriamente fazer parte da camada de criptografia do seu programa para realizar a criptografia e descriptografia dos dados. Deve-se mostrar as etapas de conexão entre o Cliente e o Servidor, a etapa de geração de chaves e troca de chaves, a etapa de comunicação de dados e de desconexão. Mostrar também com uma imagem no *GitHub* o teste de validação da saída do seu programa com o simulador da *Drexel University* usado em aula (*RSA Express Encryption-Decryption Calculator*) para criptografar e descriptografar textos.



Parte 2 – *Spring Security*

Spring Security MVC, Deploy e *PostgreSQL*

- A entrega deverá ser feita via **Teams** com o envio do arquivo diretamente ao professor (*chat* direto), **não** via *chat* da disciplina, informando a sua turma e curso;
- Deve-se entregar, somente **um(a) integrante por grupo**, contendo:
    - Um arquivo **.txt** com o **nome e RM de todos(as) os(as) integrantes do grupo**, **fornecer o *link* do *GitHub* que possua um *ReadMe* contendo toda a descrição do Projeto** (incluindo imagens e explicações, principalmente do desenvolvimento do aplicativo e **implementação da tela do login**, incluindo exemplos) e **indicar qual o IDE utilizado para elaboração do projeto** (IntelliJ, Eclipse ou NetBeans);

- No mesmo .txt além do *link* do *GitHub* do Projeto *Spring Security MVC*, **deve-se ter o *link* do *Deploy* e indicação de qual o sistema que foi utilizado para o *Deploy***.
- Um ***print* da tela com a configuração final do *Spring Initializr*** e respectivas dependências em **.jpg**, **.jpeg** ou **.png**.

Para tanto, desenvolva o seguinte aplicativo: Dentro do mesmo tema do último CP, faça um Programa para uma empresa do tipo **mercado *express*** (por exemplo: meias, produtos de limpeza, frutas, etc.) com base no *Spring Framework MVC* configurado para o tipo *Maven* em linguagem Java, com as respectivas dependências, de preferência incluindo o *Lombok*. Para tanto, implemente a interface *Web* e os respectivos *endpoints*, contemplando todos os aspectos básicos *Create*, *Read*, *Update* e *Delete* de CRUD, que devem aparecer na interface *Web* como *links* e/ou botões. Toda esta parte deve estar documentada no *ReadMe* do *GitHub* com *prints* de tela e respectivas explicações. Por exemplo, ao se consultar **um determinado produto do mercado**, o programa deve consultar uma Tabela (por exemplo: **TDS_Sec_MVC_TB_Mercado**) no banco de dados *PostgreSQL* (com uma configuração básica em um arquivo *application.yml* ou *application.yaml*) para então, retornar o resultado da consulta com **as informações do produto do mercado solicitado** na tela do navegador com o *Deploy* realizado. Deve-se implementar uma tela de *login* diferente da fornecida pelo *framework Spring Security* e caso o usuário não possua um *login*, deve-se redirecioná-lo para uma tela de *Sign In* (*Sign Up*) para que possa se inscrever e salvar seu perfil em uma tabela (por exemplo: **TDS_Users_Mercado**), para então poder fazer *login* e acessar as rotas que vocês especificarem. Com base nesse contexto, deve-se considerar:

- As colunas na Tabela do BD ficam abertas para personalização conforme o grupo desejar.
- Para o CREATE, READ, UPDATE e DELETE, sugere-se usar o BD PostgreSQL (mas pode ser utilizado o SQL Developer, caso queira) para o *Commit* com o BD.
- Pode-se utilizar o *Lombok*.
- Fique à vontade para criar os respectivos *endpoints*.
- Faça o *Deploy* em alguma plataforma que você queira (por exemplo: *GitHub pages*, ***Render***, *Fly.io*, entre outros) e disponibilize o *link* de produção via *GitHub*.

- Após o *login*, o usuário deve ser redirecionado para a respectiva tela com acesso restrito;
- Implementar pelo menos 2 tipos de usuários e respectivos acessos;
- A página base como *landing page* deve ser a *index.html*.

Tranquilo!

Bons estudos!

Atenção

- **Data de entrega do CP5: 19/10/2025 (domingo) até 23h59.**
- **Entregas atrasadas até 2 dias terão desconto de 50% na nota final. Após esse prazo, nota zero será atribuída à atividade.**