**ONDE RODA:**

- Roda localmente na porta (8080)
- Roda em docker na porta (5432)
- Roda em kubernetes na porta (8081)
- Roda na nuvem da Azure na porta (8082)

**COMO RODA:**

- Instale o docker ou dê **make up**, depois de subir os containers **make automatiza** para preencher o banco de dados
- Em kubernetes dê **minikube start** depois se não der refaça os .yamls com o **koompose**
- Na nuvem da azure acesse **localhost:8082/**

**ONDE FICA A CRIPTOGRAFIA EM HASH DA SENHA (BYCRIPT):**

Em **UserController.java** e pega a string da senha e passa a função do bycript, com o fator de cálculo de hash. Quanto maior, mais demorado e mais difícil de quebrar.

Tem que chamar o Bycript nas dependências do **pom.xml** também.

**ONDE CONECTA O BACK-END COM O FRONT-END:**

O **front-end** usa **fetch** para fazer requisições HTTP, o **back-end** usa as rotas definidas nas controllers pelo Spark

**O QUE O SERVICE FAZ:**

Fica entre o Controller e a DAO, onde contém as **regras de negócio.** Trata lógica, valida antes de chamar a DAO.

**O QUE O CONTROLLER FAZ:**

**Define as rotas** da aplicação como se fosse um túnel que conecta o front com o banco.
E executa Queries SQL usando o preparementstetamen.

**O QUE A DAO FAZ:**

**Define e manipula** as classes de **dados** (as entidades) que serão tratadas no banco.

**O QUE O MODEL FAZ:**

Define as **estruturas de dados** que trafegam entre as camadas da aplicação.

**O QUE A CONFIG FAZ:**

**Configurações de ambiente** e do banco de dados, pode conter as senhas do banco e a API do sistema inteligente.

**O QUE O APP.JAVA FAZ:**

Inicializa o spark, rotas e configurações. Equivalente a um Main.

**O QUE É UM POM.XML:**

Usado pelo Maven para **gerenciar o projeto**, define dependências, plugins, versões, build.

**O QUE É UM PREPAREMENTSTETAMEN:**
Executa comandos SQL de forma **segura**, evitando SQL Injection.

**ONDE ESTÁ O PREPAREMENTSTETAMENT:**
Em todas as **DAO.**

**COMO O PREPAREMENTSTETAMENT FUNCIONA:**
Utiliza **posição - valor**, ao invés de concatenação (+) de Strings.

**ONDE ESTÁ O SISTEMA INTELIGENTE:**
Em **OCRService.java** e em todos os arquivos que começam com OCR para salvar no banco de dados.

**ONDE CHAMA A API DO SISTEMA INTELIGENTE:**
Em **VisionClienfactory** fazendo requisições no front-end com o js em **criar-tarefa.html**

**ONDE NO FRONT-END CONECTA COM O BACK-END:**

- Em arquivos .js (como app.js), usando **fetch**("http://localhost:**8082/rota**")

- As URLs batem nas rotas definidas no **Controlle**r do back-end.