Introdução

A Visão Computacional é uma área da computação e engenharia que tem se expandido bastante e cada vez mais nos últimos anos. O seu principal objetivo é extrair dados a partir de imagens. Embora para o olho humano seja relativamente trivial fazer isso, para a máquina pode ser bem complicado, visto que uma imagem é, essencialmente, apenas uma matriz de números.

Assim, operações sobre imagens podem desempenhar um papel muito importante em várias aplicações. Uma operação simples e fácil de entender é o recorte de imagem. Ao tirar fotos, por exemplo, é bem comum realizar um recorte para pegar apenas a parte da foto que lhe interessa, não é? Outra operação bem difundida é a aplicação de filtros digitais e a mudança de cores. Quem nunca pegou aquela foto e usou aquele efeito que deixa a foto com jeito de antiga?

Fazendo um paralelo com o nosso dia a dia, quando escrevemos numa folha de papel e tiramos uma foto, praticamente, vemos o texto na cor da caneta e o papel "branco", no geral. Não parece mais fácil ler um texto assim, do que quando há desenhos ou outras coisas misturadas? Isso também é verdade para programas de Visão Computacional. Assim, não seria possível concluir que uma transformação de cores pode ajudar a reconhecer o texto em uma imagem?

Além de serem usadas para visualização, esses tipos de operação também são utilizados como parte do processo de extração de informação em uma imagem. Suponha por exemplo, um programa que tem o objetivo exclusivo de ler o valor monetário de uma cédula ou de um cheque bancário; ou o número do CPF em um documento; ou ainda, o título de uma manchete de jornal. Ele não precisa analisar todas as informações contidas na imagem, concorda? Seria possível obter a informação desejada, usando apenas a região da imagem em que a informação está, não é mesmo? Sendo assim, imagine também, que nosso programa leitor de informação (o qual recebe esse recorte de imagem) só consegue ler se tivermos o texto preto e o papel branco. Dessa forma, considerando que tiramos uma foto colorida, será que um algoritmo de transformação de cores se aplica bem?

Problema

Quando precisamos contar uma quantia em dinheiro, nós, humanos, olhamos para as notas, somamos seus valores e, a partir disso, obtemos o total. Agora, considere um caixa eletrônico (ATM), que conta o dinheiro automaticamente. Analogamente ao ser humano, ele olha para as cédulas (através de uma câmera) e para cada uma delas, precisa calcular o valor e, no fim, ter o valor total. Já temos um código que faz o reconhecimento do valor da cédula. No entanto, ele precisa receber uma imagem em preto e branco que contém apenas o número de valor da nota. Você consegue implementar uma solução que recebe a imagem colorida completa da página e devolve a imagem processada da forma que o nosso leitor de dados precisa?

Desafio

O desafio que propomos é: criar uma aplicação web (*frontend* e *backend*) para lidar com o cenário acima. Ou seja, em termos funcionais o propósito da aplicação é viabilizar o upload da imagem ao *backend*, simulando a entrada da cédula no ATM, através de uma tela; e no *backend* realizar as operações de recorte e binarização (transformação de colorido para preto e branco) nessa imagem. Também deverá ser feita uma tela para visualização (download) do resultado de cada envio, trazendo a imagem de saída para o programa leitor de dados (após as operações realizadas).

Sobre as regras das operações a se aplicar na imagem:

- **Recorte:** a região alvo da imagem de entrada é o canto superior direito da imagem, pegando na horizontal, de 80% da imagem até o final e, na vertical, do início até 30% (que um dos locais onde existe a informação do valor da cédula). Por exemplo: se uma imagem enviada tem 100 de largura por 100 de altura, a imagem de saída virá do recorte na largura da posição 81 a 100, enquanto o recorte na altura, da posição 1 a 30.

- **Binarização:** a forma clássica de se fazer essa conversão é calcular um limiar baseado em alguma propriedade da imagem. Depois, cada ponto é analisado, comparando seu valor em relação ao limiar. Fazendo isso, ao final, temos uma imagem apenas com zeros (para valores menores ou iguais ao limiar) e uns (para valores maiores que o limiar). Adiantamos para você a primeira parte do processo, e um limiar interessante para esse problema é o valor 165. Assim, você deve implementar a binarização de acordo com o processo descrito (que resultará em imagem em preto e branco).

Caso você não se instigue na programação, não fique triste...nem tudo está perdido! Nesse caso, considerando todo o cenário apontado, você deve produzir um documento descrevendo um plano de testes para a aplicação sugerida, seguindo as boas práticas tanto para a cobertura de cenários, como de padronização e de escrita técnica. Atente para contemplar os possíveis cenários de execução das funcionalidades além de formas incorretas de executar os fluxos. Quanto a você, com maior interesse ligado a programação, não deixe de mostrar sua habilidade nesse quesito (de testes) e também tente elaborar o referido documento.

Requisitos funcionais

1) Envio de imagem (*upload*)

- Serão permitidos os formatos jpeg, png. Idealmente, deve ser feita uma validação e notificação com status de erro e mensagem (*feedback*), para envios fora da regra
- No *backend* as imagens enviadas devem ficar salvas num diretório específico para imagens de entrada. Por exemplo, suponha que foram enviadas duas imagens
    - C:/diretorioRaiz/ENVIOS/imagem_teste_01.png

    - C:/diretorioRaiz/ENVIOS/imagem_teste_02.jpeg
- A aplicação deve persistir as seguintes informações:
    - Caminho do arquivo de entrada
    - Caminho do arquivo de saída (resultado)
    - Data do envio
- A requisição deve retornar um identificador único como resposta para cada envio realizado

2) Operações de transformação na imagem

- Para cada envio devem ser realizadas as operações de recorte e binarização na imagem
- As regras das transformações devem ser implementadas conforme mencionado anteriormente, na descrição do desafio
- A imagem de resultado deve ser salva mantendo o mesmo nome da imagem de entrada num diretório dedicado para os resultados. Por exemplo para as duas imagens de entrada mencionadas no requisito funcional 1 os resultados ficariam em:
    - C:/diretorioRaiz/RESULTADOS/imagem_teste_01.png
    - C:/diretorioRaiz/RESULTADOS/imagem_teste_02.jpeg

3) Listagem dos envios e seus respectivos dados

- A listagem dos envios recebidos no *backend* deve ser apresentada de modo ordenado pela data do envio, do mais recente para o mais antigo.
- Para cada envio da lista devem ser exibidos o identificador único da requisição, o nome da imagem de entrada e a data do envio.
- Na tela desta funcionalidade, para cada envio apresentado, deve haver dois links para download da imagem de entrada e da imagem de resultado (vide requisito funcional 4).

4) *Download* de imagem

- Para um dado identificador único de envio, deve-se implementar um serviço para retornar o binário da imagem de saída, mapeada por esse identificador, como download na resposta
    - o identificador usado como chave para busca e download é o mesmo retornado na resposta da requisição de upload
- caso não seja encontrada imagem para o identificador informado, deve-se notificar com status de erro mensagem (*feedback*).

Observações adicionais

- O diretório raiz, onde serão armazenadas as imagens, deve ser configurável, tanto para imagens de entrada quanto de saída.
- Para persistência das informações dos envios (vide requisito 1), pode-se utilizar banco de dados relacional (H2, MSSQL, MySQL, PostgreSQL), ou outra abordagem com persistência direta em arquivos (por exemplo, salvando os dados num arquivo .txt)

- Quanto às tecnologias a utilizar no desenvolvimento: você pode escolher as de sua preferência, **porém projetos elaborados com as tecnologias que usamos em nosso dia a dia terão um diferencial a favor na avaliação**. Para esse cenário de problema o principal conjunto ferramental que utilizamos é:
  - **Frontend**: desenvolvimento web, framework Angular
  - **Backend**: linguagem Java com framework Spring
  - Outras tecnologias que também usamos e tem *match* conosco: C/C++, C# e Python
- Sobre as operações de processamento de imagem: você pode escolher entre implementar você mesmo os algoritmos das operações ou buscar uma biblioteca de terceiros, que tenha tais operações, para integrar e utilizar no sistema. Considere a busca da solução, tanto por uma quanto outra abordagem, como parte da atividade e **saiba que na avaliação iremos favorecer os projetos onde houve a implementação por sua parte, principalmente para quem quiser demonstrar maior interesse em atuar no perfil de pesquisa e desenvolvimento (PED)**

Não se deixe intimidar pelo desafio: achou que vai ficar puxado para seu tempo disponível, conhecimento ou experiência atualmente? O que você acredita que trará mais valor? Ou o que você não gostaria de perder a oportunidade de nos mostrar que sabe? Enfim... priorize, mas não deixe de nos mostrar suas habilidades desenvolvendo. Caso tenha mais interesse ou aptidão com especificações ou testes, também está valendo!

Ao final disponibilize o código ou documentação enviando o link do projeto no Github para nós. Obs. procure não deixar para *commitar* tudo no final.

Recursos de apoio

Para cumprir este desafio indicaremos alguns recursos de apoio para ajudá-lo nesta jornada. Você pode adotá-los como um ponto de partida para estudar e aplicar na resolução durante o tempo disponibilizado até a data de entrega. No entanto não precisa se limitar apenas a este material e no mais lembre-se dos recursos e tutoriais disponibilizados oficialmente nos sites das tecnologias indicadas na seção de requisitos não funcionais.

- Curso Academia Stefanini – Iniciando com Spring Boot
- Curso Academia Stefanini – Iniciando com Angular 7
- Documentação oficial Spring Boot – Quickstart Guide [https://spring.io/quickstart]
- Documentação oficial Spring Boot – Guides [https://spring.io/guides]
- Documentação oficial Angular – Getting started [https://angular.io/start]
- Documentação oficial Angular - Tour of Heroes [https://angular.io/tutorial]
- Documentação oficial Github – Configurar git [https://docs.github.com/pt/github/getting-started-with-github/quickstart/set-up-git]
- Documentação oficial Github – Criar um repositório git [https://docs.github.com/pt/github/getting-started-with-github/quickstart/create-a-repo]

Bom trabalho!