**Data:** 07/07/2020 **Autor:** Carlos Augusto Ferreira Dias

**Sistema:** OMR

**Links do Código Fonte Estudado:**

* <https://github.com/prefeiturasp/SME-omr-api>
* <https://github.com/prefeiturasp/SME-omr-preprocessor>
* <https://github.com/prefeiturasp/SME-omr-processor>
* <https://github.com/prefeiturasp/SME-omr-file-organizer>
* <https://github.com/prefeiturasp/SME-omr-base>
* <https://github.com/prefeiturasp/SME-Serap-main>

**Visão Geral do Contexto:**

O SERAP é o sistema utilizado por professores e alunos para gerenciamento de provas e avaliação de desempenho dos próprios alunos. O sistema concede uma plataforma para realização de provas online e a possibilidade de importação de folhas de resposta no caso de provas presenciais. Também é de sua responsabilidade comportar as métricas avaliativas de fatores associados ao desempenho do aluno como bullying escolar, nível socioeconômico e engajamento dos pais.

Considerando as avaliações presenciais, as folhas de reposta passam por um fluxo de processamento para a obtenção dos resultados do executor da prova. O fluxo é distribuído entre uma série de aplicações (APIs), onde cada uma é responsável por um ou mais passos desse processamento.

**Objetivo do Estudo:**

Este documento visa mapear as funcionalidades e responsabilidades de cada aplicação presente no processamento das folhas de resposta importadas no SERAP em ambiente de depuração, esperando assim possibilitar o embasamento para manutenção destas aplicações.

**Tecnologias Envolvidas e requisitos funcionais:**

* As aplicações OMR foram produzidas em Node.js.
* O SERAP foi feito em .NET Framework 4.5.
* Banco de dados SQL Server (última versão) – GestaoAvaliacao.
* Banco de dados MongoDB (última versão) – GestaoAvaliacao\_OMR.
* Node.js versão 6.17.
* Python 2.7.
* Microsoft Visual C++ 2010 Redistributable.
* ImageMagick versão 7.0.8.
* Para os projetos Node.js é necessário criar na pasta root uma pasta chama “lib” e copiar a pasta do projeto SME-omr-base para lá.
* Inserir o JSON de usuários na tabela User do MongoDB (Fazer isso apenas quando o banco for criado).
* Copiar da máquina de produção a pasta C:\OMR que contém os JSON config dos projetos node.
* No projeto GestaoAvaliacaoAPI deverá ser configurada uma porta específica e alterada nos json config dos projetos node.
* Nos projetos GestaoAvaliacaoAPI e GestaoAvaliacao deverão ser configurados no WebConfig o usuário e senha contidos nos JSON config projetos node na tag OMRService.
* NewtonSoft em todos os projetos .NET deverão estar na mesma versão (> 11).

**Arquitetura Tecnológica:**

Uma imagem contendo texto

Descrição gerada automaticamente

Figura 1 – Organização e comunicações da arquitetura de processamento de folhas de respostas

**Depuração:**

Uma vez que as bibliotecas das aplicações node.js se encontram em versões descontinuadas, a depuração dos projetos é impossibilitada, visto do uso de DLLs externas (C++) que são acessadas em um meio assíncrono.

Para realizar a depuração das aplicações é necessário atualizar o node-canvas da aplicação que deseja depurar. Fazendo isso algumas sintaxes que são usadas na versão anterior não serão válidas mais, assim será necessário consultar a documentação oficial, disponível em: <https://www.npmjs.com/package/canvas>, e realizar as alterações para que a aplicação continue a funcionar. As principais são:

* **Instanciar a classe Image:** Antes a classe Image era instanciada de forma simples através de seu construtor (var img = new Image(src)), porém na nova versão do canvas, o construtor da classe Image não está mais disponível, sendo necessário utilizar uma Promise para realizar sua instância:  
  const { loadImage } = require(‘canvas’);  
  // ... Code  
  loadImage(src)  
  .then((imr) =>   
  {  
  // ... Code  
  });
* **Instanciar a classe Canvas:** Seguindo a mesma diretiva da classe Image, será necessário utilizar um método para realizar a instância da classe, porém este método não é tratado como Promise:  
  const { createCanvas } = require(‘canvas’);  
  // ... Code  
  var canvas = createCanvas (width, height);

Uma outra ação que facilita a depuração das aplicações é o uso de logs de execução (console.log) e o salvamento da imagem nos processos de manipulação (jobs). Para salvar a imagem, basta importar a biblioteca “fs” para a classe que fará ação e chamar a Promise que irá realizar o salvamento passando o canvas em formato de buffer. Exemplo:

const filePath = Config.FileResource.PATH.BASE + Config.FileResource.DIRECTORY.TESTING + "/" + 'FindClippingPoint' + '.png';

fs.open(filePath, 'w', (error, fd) => {

if (error) throw error;

canvas.toBuffer(function (error, buffer) {

if (error) throw error;

fs.write(fd, buffer, 0, buffer.length, null, (error) => {

if (error) throw error;

fs.close(fd);

})

});

});

**Aplicações e funcionalidades:**

* **SERAP**

Considerando o fluxo de correção de avaliações, o SERAP tem como responsabilidade gerenciar as filas de arquivos importados e notificar o usuário sobre seu estado.

* + **Importar folhas de resposta [SERAP]**
    - No SERAP o usuário tem a opção de importar uma ou várias imagens (em formato ZIP) das folhas de resposta para processamento. Essas folhas serão salvas em uma pasta interna do sistema e um registro na tabela AnswerSheetBatchQueue será adicionado contendo o caminho da imagem. Este registro entra na tabela com a coluna Situation = 1, que significa que está aguardando para ser descompactada.
    - Um registro na tela File também será criado referenciando a AnswerSheetBatchQueue e o caminho do ZIP.
    - Na listagem de arquivos enviados, o registro adicionado aparecerá com a situação “Na fila para identificação”.
  + **Descompactar imagens [SERAP]**
    - Existe um serviço Windows que roda na máquina do SERAP (UnzipAnswerSheetQueue) que de tempos em tempos verifica se existe um registro na tabela AnswerSheetBatchQueue que precisa de descompactação. Ao descompactar, para cada imagem será gerado um registro na tabela AnswerSheetBatchFiles e um registro na tabela File, referenciando as imagens descompactadas. Os registros da AnswerSheetBatchFiles serão vinculados à AnswerSheetBatchQueue criada anteriormente.
    - Após descompactar a AnswerSheetBatchQueue terá a coluna Situation = 2 e os AnswerSheetBatchFiles terão a coluna Situation = 1.
* **SME-file-organizer**

Esta aplicação tem como finalidade migrar as imagens pendentes do SERAP para o ambiente do OMR.

* + **Organização de arquivos [SME-omr-file-organizer]**
    - Após a descompactação é necessário passar os arquivos que serão processados para o diretório onde as aplicações OMR irão realizar suas tarefas. O SME-omr-file-organizer é a aplicação responsável por realizar esta tarefa.
    - Ao executar o SME-omr-file-organizer, será emitida um requisição para o GestaoAvaliacao.Api (que deverá estar em execução) buscando os arquivos que precisa começar o processamento. Os arquivos serão copiados para a pasta “*scanned*” (caminho no JSON copiado da máquina de produção).
    - Será criado, no GestaoAvaliacao\_OMR, um registro na tabela.
* **SME-omr-preprocessor**

Esta aplicação tem como finalidade preparar as imagens das respostas das provas para uma visualização limpa do SME-omr-processor. Cada uma de suas funcionalidades é realizada por um Job que executa de forma paralela aos outros Jobs.

* + **Identificação do QR-Code [SME-omr-preprocessor]**
    - Executando o SME-omr-preprocessor será feita a validação dos QR-Codes de todas as imagens da pasta “*scanned*”. Esta leitura tem como finalidade agrupar as imagens por prova, visto que o ZIP importado pode vir imagens de provas diversas.
    - A leitura é feita utilizando um executável externo (Zxing.exe) que se localiza nas pastas do SME-omr-base inseridas na pasta “lib” da aplicação.

Obs: Este EXE utiliza bibliotecas C++ instaladas no pacote Redistributable 2010, caso a aplicação não consiga iniciá-lo, pode-se validar seu funcionamento no CMD utilizando o comando será mostrado no *output* do VsCode.

* + - Para cada prova localizada nas imagens será criado um registro na tabela AnswerSheetBatch, no banco GestaoAvaliacao, ligando os registros da tabela AnswerSheetFiles a este novo registro. Também será criado um registro na tabela Aggregation do GestaoAvaliacao\_OMR que terá um comportamento de fila para os Jobs a seguir.
  + **Equalização e corte da imagem [SME-omr-preprocessor]**
    - As imagens escaneadas (na pasta scanned) passam por um molde de tamanho da folha configurado no SERAP para reduzir o tamanho da imagem para o esperado caso a imagem venha a possuir um tamanho superior.
    - Também é alterado a escala de cinza dos pixels para que seja possível distinguir com mais facilidade as marcações da folha de resposta das escritas do executor da prova (ImageMagick).
    - Feito este processamento as imagens são movidas para a pasta “equalized”.
  + **Orientação da imagem (FindClippingPonit) [SME-omr-preprocessor]**
    - A partir da imagem original equalizada (pasta equalized), serão validadas as dimensões da imagem. Caso ela esteja em paisagem, será reorientada para retrátil.
    - Será validado também o posicionamento do cabeçalho da prova (marcações superiores), através da biblioteca jsfeat, caso o cabeçalho esteja para o lado de baixo da imagem, será feito a rotação 180º para alinhar a imagem e seguir os Jobs normalmente.
    - Esta validação é algo similar ao processo que ocorre no job de detecção de cantos abaixo, porém com a finalidade de identificação para posicionamento correto da image.
  + **Corte da imagem [SME-omr-preprocessor]**
    - Após equalizada a imagem, o sistema recorta a parte que será validada da imagem (pre-processor: cabeçalho de ausência e processor: respostas). A partir deste recorte os outros Jobs irão executar em sequência e rodar suas respectivas funcionalidades.
  + **Detecção de cantos [SME-omr-preprocessor]**
    - A imagem da folha de resposta é vetorizada e colocada em uma matriz, através da biblioteca jsfeat, para que seja possível realizar a leitura dos padrões dos processos a seguir. A imagem é remontada, alterando os tons de cinza da imagem para preto (255, 255, 255), e comparando as marcações encontradas no recorte da página com os do template.
  + **Alinhamento da imagem [SME-omr-preprocessor]**
    - Utilizando a matriz do Job anterior, este Job verifica os ângulos de cada ponta da matriz, visando ajustá-la caso não esteja linear. O ângulo de alinhamento é calculado em função das duas marcações superiores de presença do aluno, validando o menor ponto X de uma com o menor do outro.
  + **Validação de template [SME-omr-preprocessor]**
    - Antes e depois da execução dos Jobs é executada uma validação de template visando uma conferência da localização de todas as marcações e QR-Code encontrados.
* **SME-omr-processor**Está aplicação visa manipular a imagem da prova para realizar a coleta das respostas definidas pelos alunos. Os mesmo Jobs executados pelo SME-omr-preprocessor são utilizados neste ponto, porém passando os parâmetros referentes ao objetivo da aplicação. Estes parâmetros estão localizados no JSON de configuração que foi resgatado da máquina de produção.

**Pontos de Melhorias, Vulnerabilidades:**

* Possibilidade de melhorar a legibilidade dos termos do código, como nomes de entidades e palavras chave do negócio.
* Criação de um protótipo do SME-omr-preprocessor e SME-omr-processor que executam de forma síncrona os Jobs para prover a depuração de código, na qual hoje não é possível.
* Criação de ambiente de homologação contendo os serviços do OMR.
* Ao atualizar o canvas para possibilitar a depuração, tem a necessidade de migrar o código implementado para a versão anterior, no momento de gerar o Pull Request, visto que o ambiente de produção não está atualizado.
* Possibilidade de desmembrar o SME-omr-preprocessor e SME-omr-processor e outros 2 jobs. Um tendo a responsabilidade apenas de manipular a imagem e fornecê-la de uma forma que própria para o processamento, e outro job realizando apenas o processamento dos dados coletados. Uma vez que as duas aplicações realizam os mesmos procedimentos iniciais de alinhamento e detecção de marcações, isto reduziria o recurso utilizado para processar da prova.

**Impeditivos:**

* Não é possível depurar os Jobs de ajuste da imagem no SME-omr-preprocessor e SME-omr-processor livremente. Devido a forma de execução paralela, os Jobs reservam o uso de DLLs C++ e ao parar em um ponto de parada (*break point*), os outros Jobs não conseguem acessar a DLL e a execução é cancelada.
* Mesmo fazendo os passos para a depuração citados acima, em alguns pontos o processamento é impedido pelo compilador por conta de timeouts e utilização de muitos pacotes para a depuração. Mensagem de erro: Node.js is unresponsive.