Ministério da Justiça

Caio Rocha Bessa

PARECER TÉCNICO EDUCATIO

Brasília
2017

Sumário

1. Introdução

Esse documento visa mostrar a auditoria realizada no sistema educatio, usando como parâmetro os requisitos não funcionais de qualidade. Em cada sessão será demonstrado o resultado de cada requisito não funcional e uma conclusão se o sistema poderá continuar sendo utilizado em produção.

2. Avaliação dos requisitos não funcionais de qualidade

Requisitos não-funcionais são os requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas.

Essa sessão está dividida em subseções que descreverá os requisitos não funcionais que não foram atingidos no projeto educatio.

2.1 Manutenibilidade, Flexibilidade e Extensibilidade

O front-end da aplicação foi escrito utilizando a linguagem java script com a biblioteca Jquery, que são bastantes utilizadas pelo mercado, porém não foi utilizado nenhum framework. Um framework é utilizado como um esqueleto para a construção da arquitetura de um projeto, que possui várias funcionalidades e implementam padrões de projetos, isso proporciona melhor flexibilidade e manutenibilidade.

Como não foi utilizado nenhum framework a arquitetura do projeto não ficou bem estruturada, a qualidade da aplicação foi fortemente afetada, pois o frontend ficou um código sem coesão, pois está misturado código de estrutura, com de negócio e html. Abaixo segue evidência na Figura 1:

```
    openNewPagePost(url+"/bServerSide/xls", montarParametrosServidor(id, colunas, table, where, argumento, tipoArgumento, order, "s
});

var aoColumns = [
                {"sTitle": "-", "mDataProp": "htmlControl","bSortable": false, "sWidth": "20px", "bVisible": true, "sDefaultContent
                    "fnRender": function ( oObj ) {
                        if(  $("#Arquivo_listar_popUp").attr('innerObject') === 'true' ){
                            return "<button type=\"button\" id=\"checkBtn_"+oObj.aData.id+"\" class=\"checkBtn"+id+" ui-button ui-v
                        }else{
                        return "<button type=\"button\" id=\"shareBtn_"+oObj.aData.id+"\" class=\"shareBtn"+id+" ui-button ui-widget
                        "<button type=\"button\" id=\"editarBtn_"+oObj.aData.id+"\" class=\"editarBtn"+id+" ui-button ui-widget ui-s
                            "<a id=\"downloadBtn_"+oObj.aData.id+"\"  class=\"downloadBtn"+id+"\" href=\""+url+"/download/"+oObj.aDa
                        }
                    }
                },
                {"sTitle": "Id", "mDataProp": "id", "bVisible": false, "sType": "numeric" },
                {"sTitle": "Data", "mDataProp": "dataImagem", "bVisible": true, "sDefaultContent":"", "sWidth": "45px",
                    "mRender": function ( dataObject, type, full) {
                        return  full.data_imagem;
                    }
                },
                {"sTitle": "Descrição", "mDataProp": "descricao", "bVisible": true, "sDefaultContent":"", "sWidth": "45px"},
                {"sTitle": "Nome", "mDataProp": "nome", "bVisible": true, "sDefaultContent":"", "sWidth": "45px"},
                {"sTitle": "Content Type", "mDataProp": "contentType", "bVisible": true, "sDefaultContent":"", "sWidth": "45px",
                    "mRender": function ( dataObject, type, full) {
                        return  full.content_type;
                    }
                },
                {"sTitle": "Tamanho", "mDataProp": "tamanho", "bVisible": true, "sDefaultContent":"", "sWidth": "45px"},

                ];

var colunas =   JSON.stringify(aoColumns, function(key, val) {if (typeof val === 'function') { return val + '';} return val;});
listServerSide(id, colunas, sInfoComplemento, url, table, where, argumento, tipoArgumento, order, "${encoded}");
```

Figura 1: Código frontend

No código as mensagens estão escritas dentro do condigo fonte (hardcoded) tornando impossível a aplicação funcionar com tradução de linguística. Segue o exemplo abaixo na Figura 2.

```
public JSONObject validacaoDocente(Long idDocente, Long idDisciplina, Long idExecucaoCurso, Date dataTurma, String atividade, Date
    JSONObject retorno = new JSONObject();
    Long horasComputadas = new Long(findHoursClassesDocentes(idDocente, idExecucaoCurso, dataTurma, atividade, fim));
    Long cargaHoraria = new Long(0);
    if((atividade.equals(AtividadesCurso.Coordenacao.toString()) || atividade.equals(AtividadesCurso.Supervisao.toString()) || ativ
        ExecucaoCurso execucaoCurso = execucaoCursoPersistenciaBusiness.find(idExecucaoCurso);
        cargaHoraria = new Long(execucaoCurso.getCurso().getCargaHoraria());
    }else{
        Disciplina disciplina = disciplinaPersistenciaBusiness.find(idDisciplina);
        cargaHoraria = new Long(disciplina.getCargaHoraria());
    }
    if(horaAulaBase < cargaHoraria){
        cargaHoraria = horaAulaBase;
    }
    Long horasComputadasDisciplina = horasComputadas+cargaHoraria;
    retorno.put("horasComputadasDisciplina", horasComputadasDisciplina);
    retorno.put("horasComputadas", horasComputadas);
    if(horasComputadasDisciplina <= new Long(240)){
        retorno.put("extrapolaLimiteExcedente", false);
        if(horasComputadasDisciplina > new Long(120)){
            retorno.put("extrapolaLimite", true);
            Long extrapola = horasComputadasDisciplina - new Long(120);
            if(extrapola > 0){
                retorno.put("mensagem", "Este docente extrapola limite anual de <b>120 h/a</b> permitido, em <b>"+extrapola.toStri
            }else{
                retorno.put("mensagem", "Este docente extrapola limite anual, favor incluir número do memorando de autorização ou
            }
        }else{
            retorno.put("extrapolaLimite", false);
            retorno.put("mensagem", "Esta pessoa tem "+ horasComputadas +"h/a executadas, e a disciplina escolhida tem "+cargaHorar
        }
    }else{
        retorno.put("extrapolaLimiteExcedente", true);
```

Figura 2: Código hardcoded

No código existem também

```
String habilitadoMensagem ="";

if(edital.getEditalFormacaoAcademicaHabilitacao()!=null && edital.getEditalFormacaoAcademicaHabilitacao().size()>0){
    validarFormacaoAcademica=true;
    for (EditalFormacaoAcademicaHabilitacao editalFormacaoAcademicaHabilitacao : edital.getEditalFormacaoAcademicaHabilitacac
        if(pessoa.getFormacoesAcademicas()!=null && pessoa.getFormacoesAcademicas().size()>0){
            for(FormacaoAcademica formacaoAcademica : pessoa.getFormacoesAcademicas()){
                if(formacaoAcademica.getTipoFormacaoAcademica().name().equals(editalFormacaoAcademicaHabilitacao.getTipoForma
                    if(editalFormacaoAcademicaHabilitacao.getAreaConhecimento()!=null && editalFormacaoAcademicaHabilitacao.g
                        if(formacaoAcademica.getAreaConhecimentos()!=null && formacaoAcademica.getAreaConhecimentos().size()>
                            for(AreaConhecimento areaConhecimento : formacaoAcademica.getAreaConhecimentos()){
                                if(areaConhecimento.getArea().equals(editalFormacaoAcademicaHabilitacao.getAreaConhecimento()
                                    if(editalFormacaoAcademicaHabilitacao.getNomeCurso()!=null && !editalFormacaoAcademicaHab
                                        String[] tags = editalFormacaoAcademicaHabilitacao.getNomeCurso().split(";");
                                        for(String tag: tags){
                                            tag=tag.trim().toLowerCase().trim().replaceAll(" +", " ").replaceAll("\\s",".*");
                                            if(formacaoAcademica.getCurso().toLowerCase().matches(tag)){
                                                habilitadoFormacaoAcademica=true;
                                                break;
                                            }
                                        }
                                    }else{
                                        habilitadoFormacaoAcademica=true;
                                        break;
                                    }
                                }
                            }
                        }
                    }else{
                        if(editalFormacaoAcademicaHabilitacao.getNomeCurso()!=null && !editalFormacaoAcademicaHabilitacao.get
                            String[] tags = editalFormacaoAcademicaHabilitacao.getNomeCurso().split(";");
                            for(String tag: tags){
                                tag=tag.trim().toLowerCase().trim().replaceAll(" +", " ").replaceAll("\\s",".*");
                                if(formacaoAcademica.getCurso().toLowerCase().matches(tag)){
                                    habilitadoFormacaoAcademica=true;
                                    break;
                                }
```

Figura 3: Código frontend

Em analise estática utilizando “sonarqube” o código obteve um resultado negativo de qualidade e código. A ilustração abaixo na Figura 4 contem o relatório o resultado da análise:

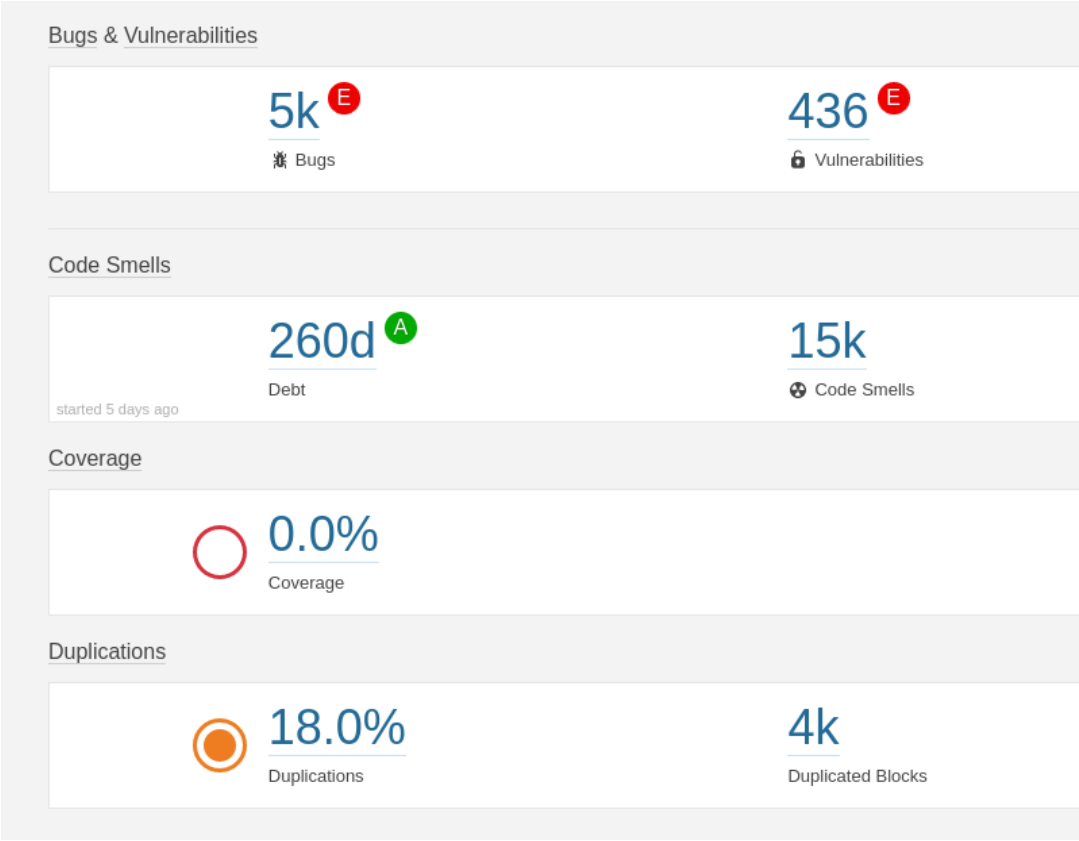

Figura 4: Relatório Sonar

No relatório da Figura 4 é observado que a aplicação possui milhares de não conformidades sendo elas: erro de sintaxe, falta de coesão, vulnerabilidades e códigos duplicados e sem cobertura de testes unitários. Nas tabelas abaixo serão apresentados os arquivos de forma sumarizada que contém o maior número de não conformidades.

Na Tabela 1 contém os dados dos arquivos que possuem mais violações de regras de boas praticas de programação e sintaxe.

Arquivos com mais violações de regras	Número de violações
InstituicaoEnsinoTableInfo.java	417
MyJdbcMutableAclService.java	302
CurriculoTableInfo.java	302
PessoaTableInfo.java	247
CurriculoPersistenciaBusiness.java	234

Tabela 1: Relatório Sonar violações

Complexidade ciclomática, é uma métrica de software usada para indicar a complexidade de um programa de computador, ela mede a quantidade de caminhos de execução independentes a partir de um código fonte.

Na Tabela 2 contém os dados dos arquivos que contém o código com maior complexidade ciclomática.

Arquivos com código mais complexo	Número de complexidades encontradas
ExtratoBusiness.java	472
MyJdbcMutableAclService.java	229
InstituicaoEnsinoTableInfo.java	205
EditalBusiness.java	157
CurriculoBusiness.java	157

Tabela 2: Relatório Sonar violações

A falta de utilização das melhores praticas de desenvolvimento, resulta em um código fonte com auto índice de duplicação. Na tabela 3 contém os dados das classes com maior índice de duplicação.

Arquivos com maior duplicação de código	Número de duplicações entradas
ExtratoBusiness.java	528
PessoaTable.java	518
CurriculoTable.java	518
Pessoa.java	493
Curriculo.java	493

Tabela 3: Relatório Sonar violações

2.2 Confiabilidade

No back end não está sendo utilizado o conceito ACID de transação (Atomicidade, Consistência, Isolamento, Durabilidade). A aplicação está utilizando o framework de persistência “spring-data-jdbc-repository” (https://github.com/nurkiewicz/spring-data-jdbc-repository) esse framework não implementa os conceitos de ORM e utiliza JDBC puro, fazendo assim as transações do banco de dados não serem atômicas, o que é um problema sério, pois caso existe algum erro na transação os dados que foram salvos durante a transação serão persistidos, resultando em dados não consistentes. Abaixo segue as evidências.

Na Figura 5 e ilustrado um método de serviço que tem pro objetivo cadastrar um evento.

```
@Transactional(rollbackFor=Throwable.class)
@RequestMapping(method = RequestMethod.POST, headers = "Accept=application/json")
public ResponseEntity<String> createFromJson(@RequestBody String json, HttpServletRequest httpServletRequest, Principal principla)

    Usuario currentUser = AclUtils.obterUsuario();
    eventoLogBusiness.gerarLog(json, httpServletRequest, currentUser);
    Evento evento = salvarEventoBusiness.salvarEvento(json);
    eventoAclBusiness.salvarPermissoesBasicasEvento(evento);
    HttpHeaders headers = new HttpHeaders();
    headers.add("Content-Type", "application/json; charset=utf-8");
    return new ResponseEntity<String>(headers, HttpStatus.CREATED);
}

@RequestMapping(method = RequestMethod.GET, value = "/listar",produces = "text/html")
```

Figura 5: Código evento serviço

Ao tentar realizar o cadastro de um evento via tela, não informando o campo de município, foi exibido o seguinte erro:

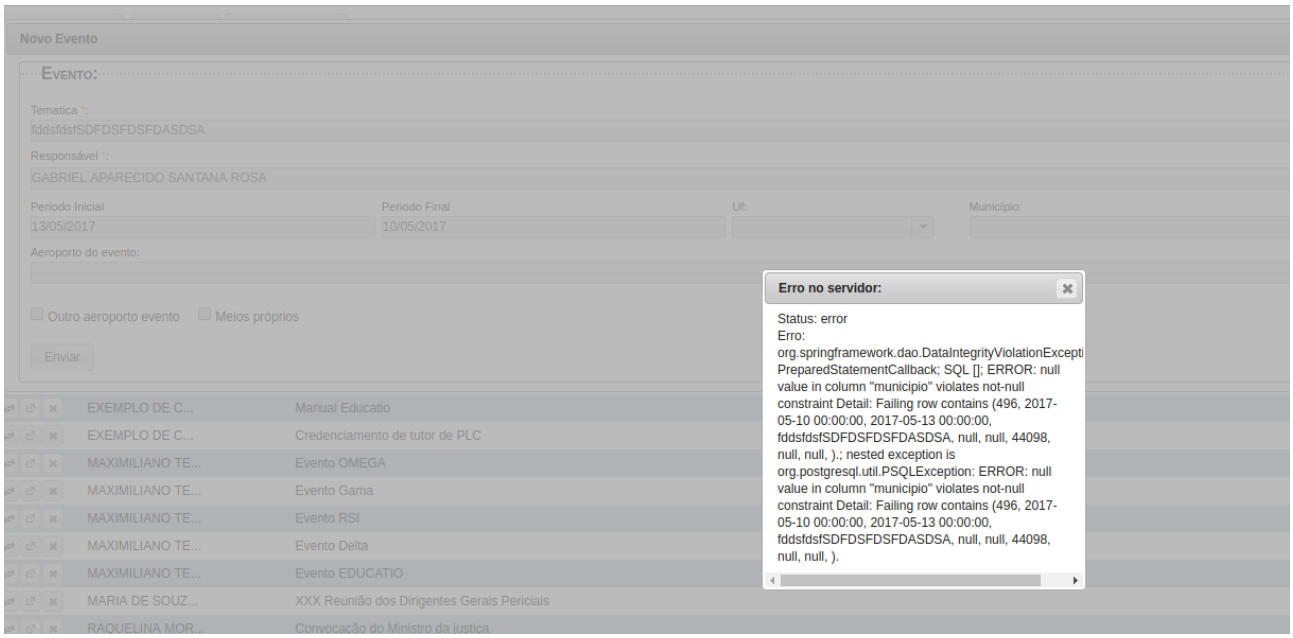

Figura 6: Erro inserir Log

Como não existe controle transacional alguns itens foram salvos mesmo ocorrendo um erro na transação, Segue a evidencia abaixo Figura 7:

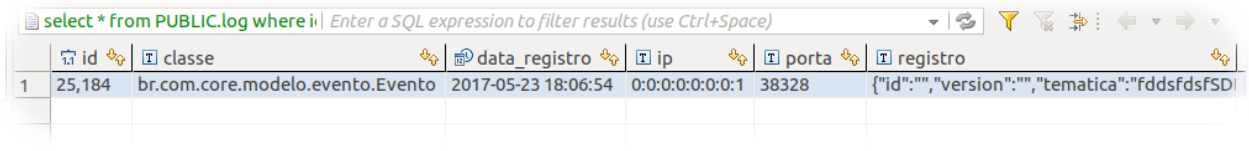

Figura 7: Insert Log no banco de dados

Esse erro ocorre por causa da escolha do framework de persistência. A arquitetura do sistema e criada em cima de um framework, que se tornará base estrutural aonde todo projeto será construído. Como a aplicação foi construída em cima da arquitetura com esse framework de persistência o problema ocorre em toda a aplicação, onde existe 653 classes (230 de negócio, 199 de repositório e 224 entidades) que serão necessárias serem refeitas.

2.3 Segurança

As regras de negócio de validação estão apenas no frontend caso o usuário utilize diretamente o endpoint conseguira inserir registros não obedecendo às regras.

No frontend existem códigos de sql que são passados para o backend, permitindo assim realizar SQL Injection. Todas as pesquisas do sistema estão passando as instruções de select para o backend. Com essa falha é possível utilizar um cliente HTTP e mudar as instruções de select, para saber dados de outro usuário. Um exemplo utilizado foi saber os idiomas de outro usuário:

Ao abrir a tela não obteve nenhum resultado, pois o aluno logado não possui nenhum idioma cadastrado.

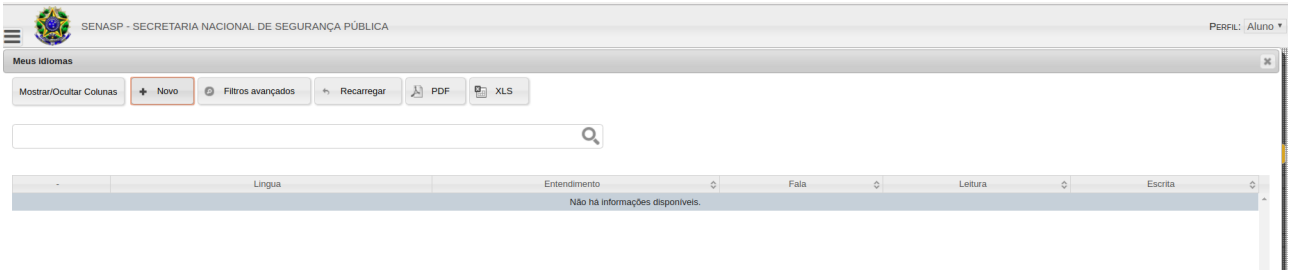

Figura 8: Resultado idiomas

Ao abrir o console do navegador crome na aba “rede” e possível obter os dados do protocolo HTTP, conforme ilustrado na Figura 9.

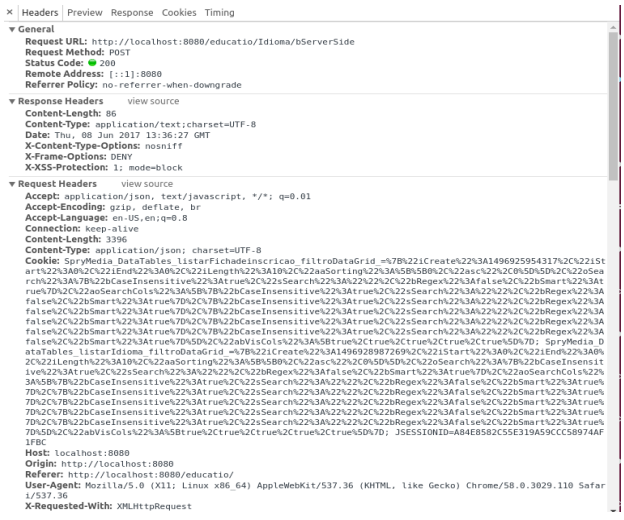

Figura 9: Protocolo HTTP original

Ao copiar a requisição e utilizar um REST cliente, e alterar o id do usuário logado será possível recuperar o Idioma de outro usuário. Como por exemplo o id do usuário logado é 30205, ao alterar o id para outro usuário 31678 e possível recuperar o idioma de outro usuário, conforme ilustrado na Figura 10.

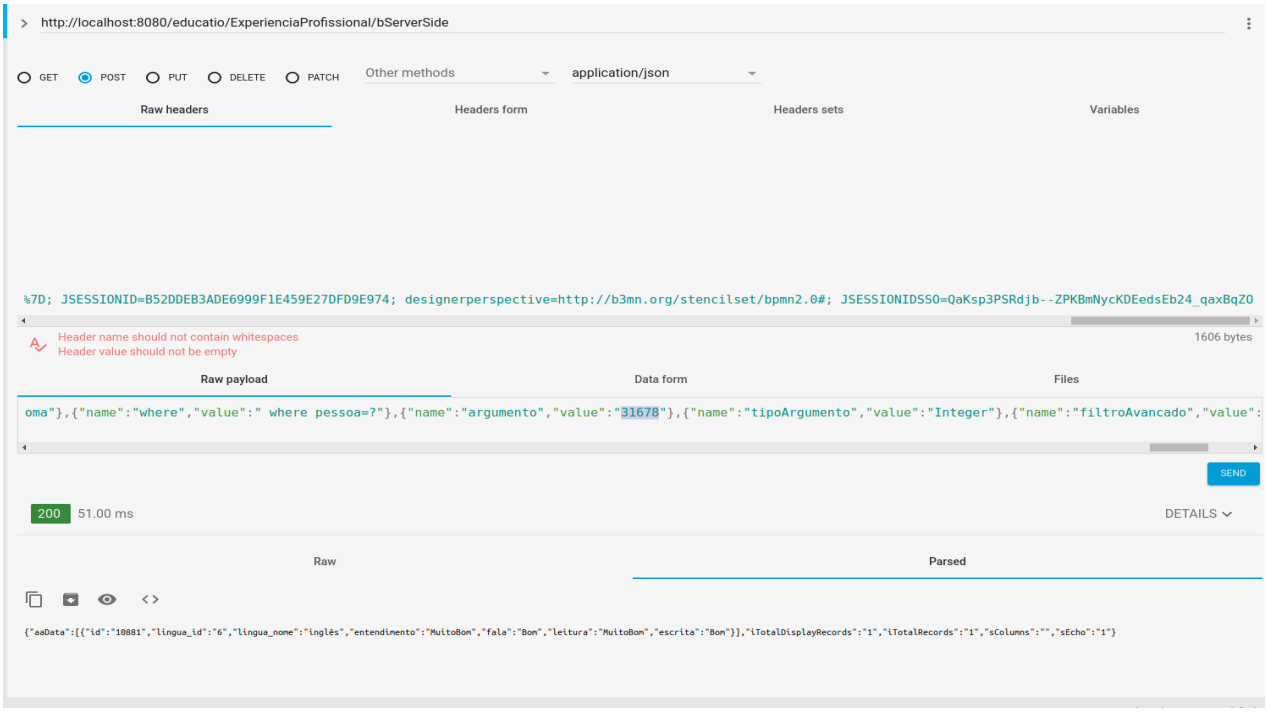

Figura 10: Protocolo HTTP modificado

Todas as consultas do sistema estão com esse falha de segurança. Possibilitando então interagir com o banco, para recuperar dados não permitidos, utilizando um simples cliente http, como demostrando na Figura 10.

Outra falha de segurança encontrada, foi ao executar a funcionalidade “recuperar senha” a senha é exibida em texto plano no log da aplicação, assim expondo dados sensíveis no log. Abaixo segue a figura:

```
Enviando senha nova para 989.609.465-91
achou usuario
gerou senha
I9Y95oxQ
setou senha codificada
enviou senha nova por email com os links
http://localhost:8080/educatio/Usuario/passW?a1=46114696ddd5f5508f04bd0798c68561&add34b=30553&cc6752=localhost:8080
atualizou a senha
setou msg no http
redirecionou
Executing task using Thread:getAsyncExecutor-1
Enviando email para nadjanegs@hotmail.com
org.springframework.mail.MailSendException: Failed messages: javax.mail.SendFailedException: Invalid Addresses;
```

Figura 11: Senha exibida em texto plano

2.4 Performance

Na persistência é utilizado JDBC puro. O que gera uma grande perda de performance pois a aplicação não fará cache de objeto, fazendo assim várias requisições ao banco de dados gerando tráfico desnecessário. Evidenciado na Figura 12.

```
public void deleteUsuario(Long usuario)
{
    this.getJdbcOperations ().update("delete from  " + UsuarioPerfilTableInfo.getTableName() + " where " + UsuarioPerfilTableInfo.CC
}
```

Figura 12: Código SQL

Existem 543 web services disponibilizados no sistema, que possuem o mesmo erro de implementação, que é, sempre recebe uma string e converte para objeto, executa as operações e transforma o objeto em string novamente. Na Figura 13 é evidenciado.

```
@Transactional(rollbackFor=Throwable.class)
@RequestMapping(method = RequestMethod.POST, headers = "Accept=application/json")
public ResponseEntity<String> createFromJson(@RequestBody String json, HttpServletRequest httpServletRequest, Principal principla)

    Usuario currentUser = AclUtils.obterUsuario();
    eventoLogBusiness.gerarLog(json, httpServletRequest, currentUser);
    Evento evento = salvarEventoBusiness.salvarEvento(json);
    eventoAclBusiness.salvarPermissoesBasicasEvento(evento);
    HttpHeaders headers = new HttpHeaders();
    headers.add("Content-Type", "application/json; charset=utf-8");
    return new ResponseEntity<String>(headers, HttpStatus.CREATED);
}

@RequestMapping(method = RequestMethod.GET, value = "/listar",produces = "text/html")
```

Figura 13: End Point convertendo String

Como uma string e imutável essa conversão ocupa muita memória no HEAP space da JVM. Na aplicação existe 543 web services com o mesmo padrão ilustrado na imagem acima.

Abaixo será demostrado a grande quantidade de objetos chars[] e Strings criadas na memória por esta utilizando a conversão dos objetos no webservice.

Nas imagens abaixo evidencia o uso de recursos computacionais de um tomcat executando sem nenhum sistema implantado.

Uso de Memória secundaria (RAM) Figura 14:

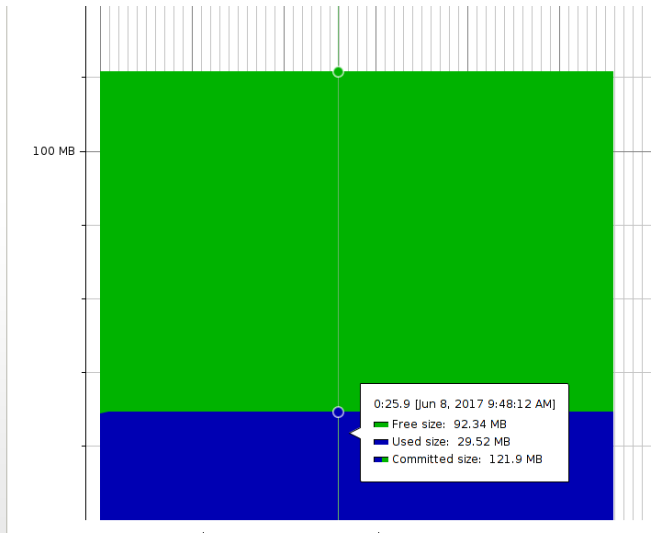

Figura 14: Uso de memória padrão tomcat

Objetos em memória Figura 15:

Name	Instance count
char[ ]	88,812
java.lang.String	55,200
byte[ ]	17,697
java.util.HashMap$Node	13,323
java.lang.StringBuilder	12,435
java.lang.Object[ ]	7,603
java.lang.Class[ ]	6,531
java.lang.reflect.Method	4,880
java.util.concurrent.ConcurrentHashMap$Node	3,843
int[ ]	3,730
java.lang.Object	3,717
java.lang.Class	3,159
sun.reflect.ConstantPool	2,715
java.util.Hashtable$Entry	2,658
java.lang.Class$AnnotationData	2,550
java.util.LinkedHashMap$Entry	1,846
java.lang.ref.Finalizer	1,624
sun.misc.ProxyGenerator$ConstantPool$IndirectEntry	1,480
java.util.ArrayList	1,435
java.util.HashMap	1,419
java.lang.String[ ]	1,380
java.util.HashMap$Node[ ]	1,346
java.util.TreeMap$Entry	1,231
short[ ]	1,221
java.net.URL	989
java.lang.Short	892

Figura 15: Objetos em memória padrão tomcat

Após realizar o “deploy” do educatio no “tomcat”, e realizar o login no sistema, e depois acessar a interface disponível no seguinte menu, “Meu currículo > Dados Pessoais”, conforme ilustrado na Figura 16.

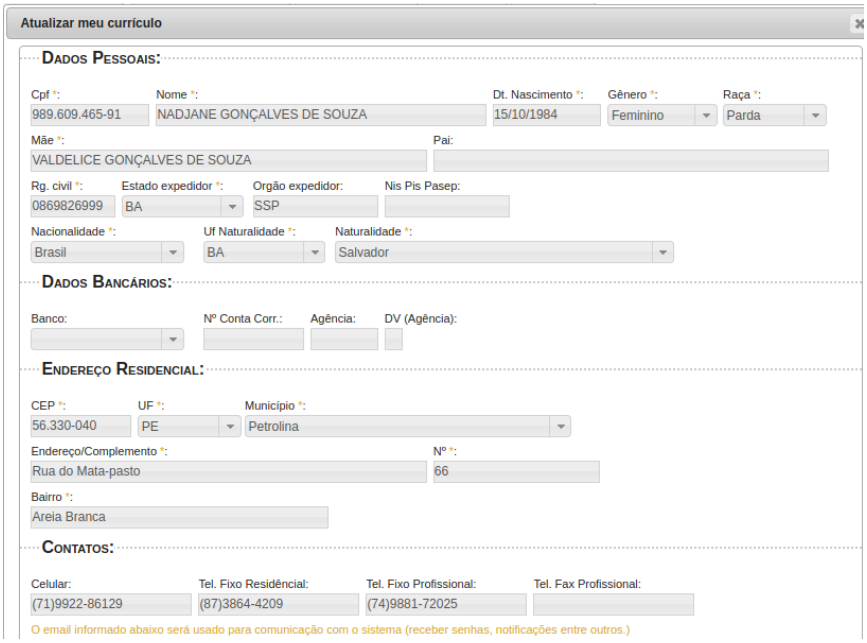

Figura 16: Tela dados pessoais

O sistema utilizou os recursos computacionais da máquina de forma exorbitante apenas acessando a funcionalidade da Figura 16. Abaixo segue as evidências:

Uso da memória com o sistema educatio ilustrado na Figura 17:

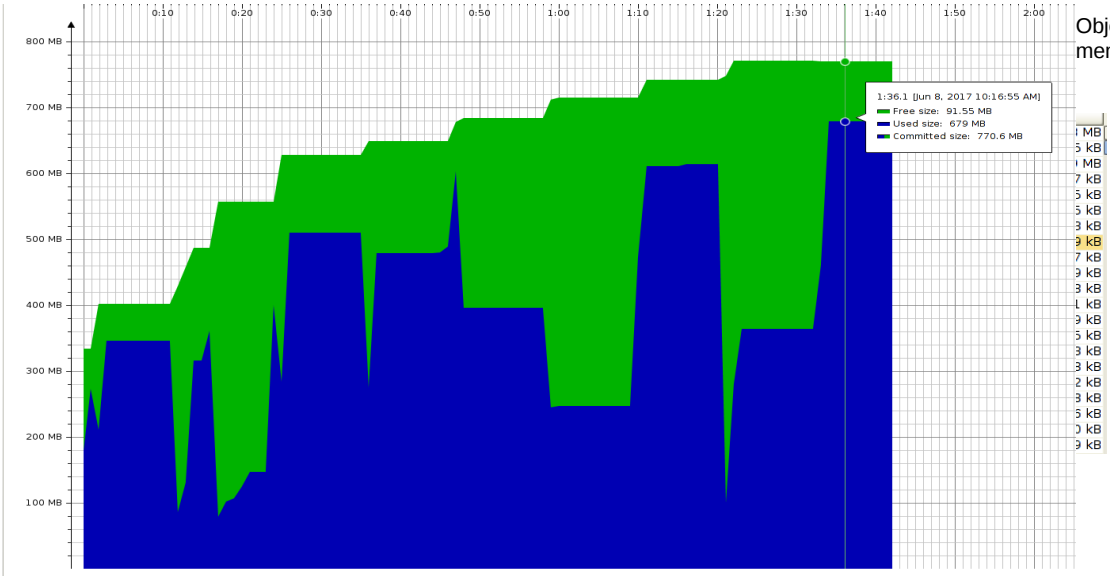

Figura 17: Uso de mé moria com o sistema educatio

Objetos na memória com o sistema educatio, ilustrado na Figura 18:

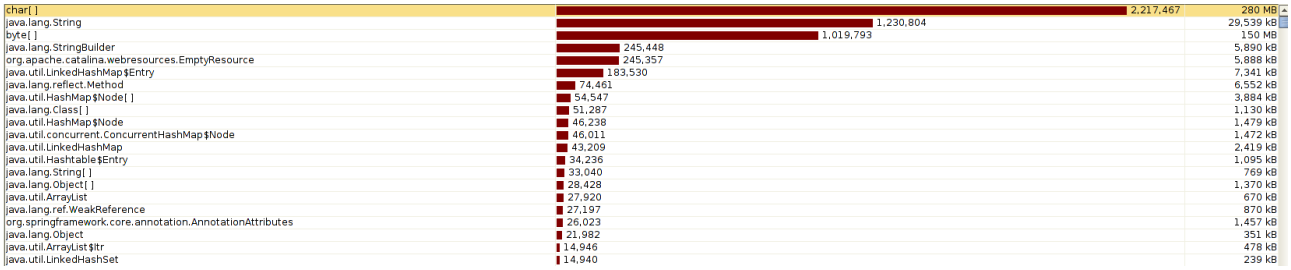

Figura 18: Uso de mé moria com o sistema educatio

Conforme ilustrado nas Figuras 18 e 17 é possível verificar que o número de objetos chars[] aumentaram de 88,812 para 2.214.467, e de “String” de **55.200** para 1.230.804 com apenas um usuário logado.

2.4 Escalabilidade

Foram utilizados serviços do spring que são altamente performáticos para lidarem com a regra de negócio, porém na persistência são utilizados objetos que contem variáveis de instância não gerenciadas pelo contêiner, que resultara em vários comportamentos inesperados, pois variáveis não gerenciadas não são seguras com concorrência de threads.

Abaixo na Figura 10 e ilustrada a classe do framework de persistência que não e thread safe.

```
public abstract class JdbcRepository<T extends Persistable<ID>, ID extends Serializable> implements PagingAndSortingRepository<T

        public static Object[] pk(Object... idValues) {
                return idValues;
        }

        private final TableDescription table;

        private final RowMapper<T> rowMapper;
        private final RowUnmapper<T> rowUnmapper;

        private SqlGenerator sqlGenerator = new SqlGenerator();
        private BeanFactory beanFactory;
        private JdbcOperations jdbcOperations;

        public JdbcRepository(RowMapper<T> rowMapper, RowUnmapper<T> rowUnmapper, SqlGenerator sqlGenerator, TableDescription ta
                Assert.notNull(rowMapper);
                Assert.notNull(rowUnmapper);
                Assert.notNull(table);

                this.rowUnmapper = rowUnmapper;
                this.rowMapper = rowMapper;
                this.sqlGenerator = sqlGenerator;
                this.table = table;
        }
```

Figura 19: Código framework persistência

3 Parecer técnico

O sistema não utilizou uma arquitetura adequada, e também não foi desenvolvido utilizando as melhores praticas de orientação a objeto. Resultando em uma aplicação com um código extremamente difícil de realizar manutenção.

A aplicação utiliza um framework de persistência que está descontinuado e que não é feito para trabalhar com muitas requisições, e também não existe controle transacional e falhas de segurança, o que gerará dados inconsistentes, tornando o sistema sem confiabilidade.

Visto todos os problemas descritos nas sessões dos requisitos não funcionais, a aplicação necessita ser refeita, com uma nova arquitetura que atingirá os requisitos não funcionais, sendo passível de evoluções e mudanças que acontecerão no negócio.

Concluindo assim que a aplicação com a arquitetura atual não pode ser utilizada em produção.