

Universidade Federal Rural de Pernambuco

Henrique César

Ricardo Henrique

Thales Gabriel

Análise estática do código-fonte do projeto Atox

Sumário

1 – Visão Geral

2 – Métricas

2.1 – Resultados da Análise Estática e Evolução do Código

2.2 – Code Smells

2.3 – Complexidade Ciclomática

2.3.1 – Complexidade Ciclomática por Classe

3 – Computação Trainee

1 - Visão Geral

Através do uso da Análise Estática, buscamos identificar no código erros, mudanças e características que constatem evolução do código durante a avaliação baseada em três versões do código.

2.1 – Resultados da Análise Estática e Evolução do Código

	Versão 1	Versão 2	Versão 3
Bug	5	17	0
Vulnerabilidades	4	4	0
Code Smells	90	164	178
Complexidade Ciclômática	287	422	476
Linhas Duplicadas	0%	2.6%	2.3%

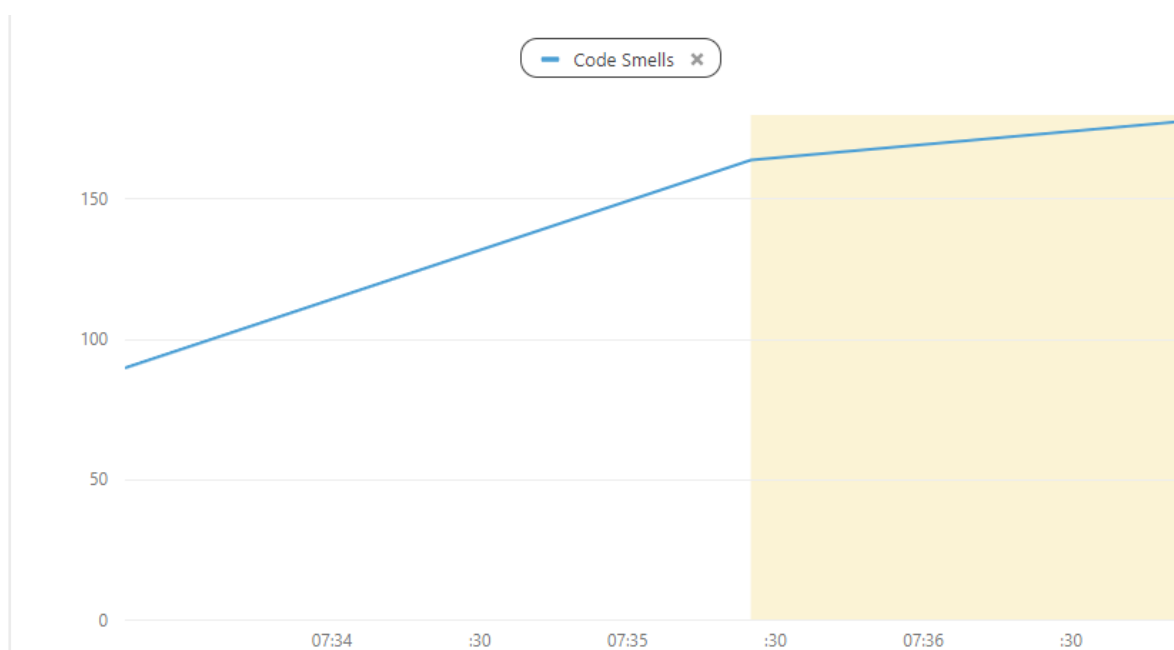
Evolução do Código

Como foi constatado através da análise estática, entre a versão 1 e a versão 2 a equipe acabou por envolver no projeto, tendo aumentado consideravelmente o número de bug's e de code smells.

Mas, da versão dois para a versão três, houve um grande progresso e o projeto se encontrou sem bugs ou vulnerabilidades e uma diminuição percentual nos demais erros, por exemplo, nos Code Smells, a equipe aumentou em 82,22% da versão 1 para a versão 2, mas da versão dois para a versão três houve um aumento de apenas 8,54%.

2.2 Code Smells

Gráfico de evolução dos Code Smells



A equipe mostrou uma evolução na quantidade de code smells, tendo muito deles ocorridos na versão 2 e acabaram por não serem corrigidos e voltaram a ocasionar na versão 3, como alguns dos erros abaixo:

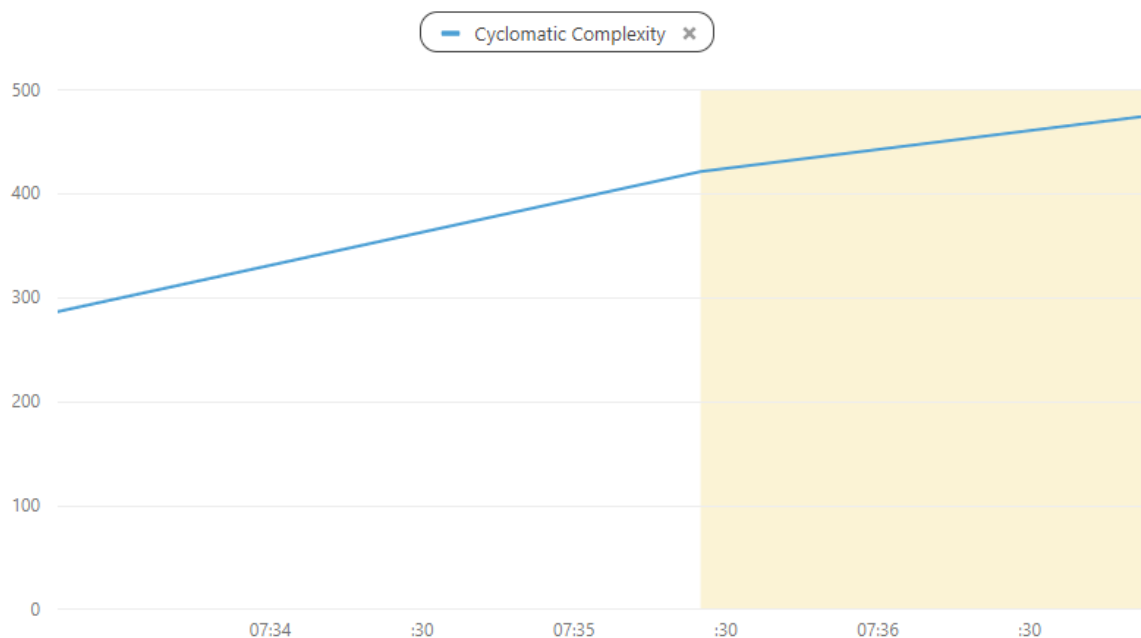
network/negocio/ProdutorNegocio.java	
<input type="checkbox"/> Remove this unused "produtor" private field. *** Code Smell Major Open Not assigned 5min effort Comment	21 minutes ago L16 unused
<input type="checkbox"/> Remove this useless assignment to local variable "novoLog". *** Code Smell Major Open Not assigned 15min effort Comment	21 minutes ago L31 cert, cwe, unused
<input type="checkbox"/> Remove this unused "novoLog" local variable. *** Code Smell Minor Open Not assigned 5min effort Comment	21 minutes ago L31 unused
<input type="checkbox"/> Remove this useless assignment to local variable "novoLog". *** Code Smell Major Open Not assigned 15min effort Comment	21 minutes ago L33 cert, cwe, unused
<input type="checkbox"/> Remove this unused "novoLog" local variable. *** Code Smell Minor Open Not assigned 5min effort Comment	21 minutes ago L33 unused
<input type="checkbox"/> Remove this useless assignment to local variable "novoLogo". *** Code Smell Major Open Not assigned 15min effort Comment	21 minutes ago L46 cert, cwe, unused
<input type="checkbox"/> Remove this unused "novoLogo" local variable. *** Code Smell Minor Open Not assigned 5min effort Comment	21 minutes ago L46 unused
<input type="checkbox"/> Remove this useless assignment to local variable "novoLogo". *** Code Smell Major Open Not assigned 15min effort Comment	21 minutes ago L48 cert, cwe, unused

network/negocio/ProdutorNegocio.java	
<input type="checkbox"/> Remove this unused "produtor" private field. *** Code Smell Major Open Not assigned 5min effort Comment	10 minutes ago L16 unused
<input type="checkbox"/> Remove this useless assignment to local variable "novoLog". *** Code Smell Major Open Not assigned 15min effort Comment	10 minutes ago L31 cert, cwe, unused
<input type="checkbox"/> Remove this unused "novoLog" local variable. *** Code Smell Minor Open Not assigned 5min effort Comment	10 minutes ago L31 unused
<input type="checkbox"/> Remove this useless assignment to local variable "novoLog". *** Code Smell Major Open Not assigned 15min effort Comment	10 minutes ago L34 cert, cwe, unused
<input type="checkbox"/> Remove this unused "novoLog" local variable. *** Code Smell Minor Open Not assigned 5min effort Comment	10 minutes ago L34 unused
<input type="checkbox"/> Remove this useless assignment to local variable "novoLogo". *** Code Smell Major Open Not assigned 15min effort Comment	10 minutes ago L48 cert, cwe, unused

89 dos 178 Code Smells são ocasionados por variáveis locais que não estão sendo utilizadas no código.

2.3 Complexidade Ciclomática

Gráfico Complexidade Ciclomática



Como podemos observar no gráfico, há um aumento constante da complexidade ciclomática entre a versão 1 e a versão três, os principais ocasionadores são:

1 – Métodos com altos níveis de controle

```
for(AtoxLog log : this.logs) {
    if(log.getTipo() == 0) {
        this.mostrarLog(log);
    } else {
        this.mostrarLogDeErro(log);
    }
}

}

public void empurraRegistrosPraFila() {
    if(!this.logs.isEmpty()) {
        List<AtoxLog> logsParaRemover = new ArrayList<>();
        for (AtoxLog log : this.logs) {
            logsParaRemover.add(log);
            this.filaDeLogs.add(log);
        }
        this.logs.removeAll(logsParaRemover);
    }
}

public void salvaRegistrosNoBancoDeDados(FragmentActivity fragmentActivity) {
    if(!this.filaDeLogs.isEmpty()) {
        atoxLogNegocio = new AtoxLogNegocio(fragmentActivity);
        for (int i = 0; i < this.filaDeLogs.size(); i++) {
            AtoxLog logParaRemover = this.filaDeLogs.peek();
            AtoxLog logRemovido = null;
            Long resultadoDaInsercao = null;
            if (logParaRemover != null) {
                resultadoDaInsercao = atoxLogNegocio.cadastrar(logParaRemover);
            }
            if (resultadoDaInsercao != null) {
                logRemovido = this.filaDeLogs.poll();
            }
        }
    }

    public Bitmap carregar() {
        FileInputStream inputStream = null;
        try {
            inputStream = new FileInputStream(criarArquivo());
            return BitmapFactory.decodeStream(inputStream);
        } catch (Exception e) {
            atoxLog = new AtoxLog();
        } finally {
            try {
                if (inputStream != null) {
                    inputStream.close();
                }
            } catch (IOException e) {
                atoxLog = new AtoxLog();
            }
        }
        return null;
    }
}
```

2 – Métodos que podem ser divididos em dois

Foram encontrados alguns métodos que podem ser divididos em dois métodos para melhor manutenção do código, como por exemplo, o método abaixo:

```
public void salvar(Pessoa pessoa, Usuario usuario, String caminhoAvatar) {
















    Boolean valido = true;
    String nome = mName.getText().toString();
    String telefone = mTelefone.getText().toString();
    String dataNascimento = mData.getText().toString();
    String email = mEmail.getText().toString();
    String senha = mSenha.getText().toString();
    String confirmSenha = mSenhaConfirm.getText().toString();
    ValidaCadastro validaCadastro = new ValidaCadastro();
    if (validaCadastro.isCampoVazio(nome)) {
        mName.requestFocus();
        mName.setError(getString(R.string.error_invalid_name));
        valido = false;
    }
    if (validaCadastro.isCampoVazio(telefone)) {
        mTelefone.requestFocus();
        mTelefone.setError(getString(R.string.error_invalid_tel));
        valido = false;
    } else if (!validaCadastro.isDataNascimento(dataNascimento)) {
        mData.requestFocus();
        mData.setError(getString(R.string.error_invalid_date));
        valido = false;
    } else if (!validaCadastro.isEmail(email)) {
        mEmail.requestFocus();
        mEmail.setError(getString(R.string.error_invalid_email));
        valido = false;
    } else if (!validaCadastro.isSenhaValida(senha)) {
        mSenha.requestFocus();
        mSenha.setError(getString(R.string.error_invalid_password));
        valido = false;
    }
}
```



















```

    } else if (!(senha.equals(confirmSenha))) {
        mSenhaConfirm.requestFocus();
        mSenhaConfirm.setError(getString(R.string.error_invalid_password));
        valido = false;
    }
    Long resultadoDaAtualizacao = null;
    if (valido) {
        String realSenha = Criptografia.encryptPassword(senha);
        pessoa.setNome(nome);
        pessoa.setTelefone(telefone);
        pessoa.setDataNascimento(new Date(dataNascimento));
        pessoa.setCaminhoDoAvatar(caminhoAvatar);
        usuario.setEmail(email);
        usuario.setSenha(realSenha);
        pessoaNegocio.setPessoa(pessoa);
        resultadoDaAtualizacao = pessoaNegocio.atualizar();
    }
    if(resultadoDaAtualizacao != null) {
        sessaoUsuario.setPessoaLogada(pessoa);
        sessaoUsuario.setUsuarioLogado(usuario);
        goToHomeScreen();
    } else {
        alert("Ocorreu um erro na atualização. Verifique se os campos foram preenchidos corretamente.");
    }
}
}

```

2.3.1 Complexidade Ciclomática por Classe

 atoxlogs/AtoxLog.java	34
 usuario/persistencia/dao/PessoaDao.java	27
 usuario/gui/EditarPerfilActivity.java	23
 infra/persistencia/FormataData.java	21
 usuario/ dominio/Endereco.java	20
 navegacao/fragments/InicioFragment.java	19
 receitas/persistencia/ReceitaDao.java	19
 infra/negocio/ArmazenarImagem.java	18
 usuario/ dominio/Pessoa.java	18
 usuario/negocio/PessoaNegocio.java	18
 usuario/gui/RegistroActivity.java	16
 usuario/persistencia/dao/SessaoUsuarioDao.java	16
 usuario/gui/LoginActivity.java	15
 network/ dominio/Produtor.java	15
 network/ dominio/Feirinha.java	14
 atoxlogs/persistencia/dao/AtoxLogDao.java	13

 infra/persistencia/Mascara.java	12
 navegacao/fragments/PerfilFragment.java	12
 usuario/ dominio/SessaoUsuario.java	11
 usuario/gui/EnderecoActivity.java	10
 usuario/ dominio/Usuario.java	10
 infra/negocio/ValidaCadastro.java	9
 receitas/ dominio/SecaoReceita.java	8
 atoxlogs/negocio/AtoxLogNegocio.java	7
 navegacao/activities/MenuActivity.java	7
 navegacao/adapters/PlacesRecyclerViewAdapter.java	7
 network/persistencia/dao/ProdutorDao.java	7
 receitas/ dominio/Receita.java	6
 receitas/negocio/ReceitaNegocio.java	6
 usuario/negocio/SessaoNegocio.java	6
 receitas/ dominio/UsuarioReceita.java	6
 infra/negocio/Criptografia.java	5
 network/gui/FeirinhaCustomAdapter.java	5
 network/gui/ProdutorCustomAdapter.java	5

3 – Computação Trainee

A computação escolhida pela equipe foi a de recomendação, baseada na premissa de uma filtragem colaborativa. O algoritmo escolhido foi o de Slope One.

O método de utilização do SlopeOne é baseado numa nota geral de cada vaga, que é calculada através da média geral das avaliações dela e depois disso, é utilizado as predições do usuário em questão baseado nas vagas que ele já avaliou, caso o valor da predição para a vaga seja maior ou igual a 3.0, ela é recomendada.

Segue casos de testes e resultados abaixo:

		VAGAS							
		Analista de SW	Cientista de Dados	Desenvolvedor Python	Desenvolvedor Front-End	Auxiliar Administrativo	Auxiliar em RH	Estágio em Logística	Estágio em Contabilidade
ESTÁGIARIOS	André	2,0	5,0	5,0	4,0	2,0	1,0	2,0	1,0
	Évlla	2,0	1,0	2,0	1,0	3,0	3,0	3,0	5,0
	Henrique	5,0	4,0	5,0	5,0	1,0	2,0	1,0	1,0
	Julia	2,0	1,0	1,0	1,0	5,0	3,0	5,0	4,0
	Laura	1,0	1,0	1,0	2,0	5,0	5,0	5,0	5,0
	Thales	3,0	5,0	5,0	5,0	2,0	1,0	1,0	2,0
	Willian	2,0	2,0	2,0	1,0	3,0	5,0	4,0	5,0
	Yasmmin	3,0	3,0	4,0	4,0	1,0	2,0	1,0	2,0

CASOS DE TESTE									
Ricardo	2,63	2,88	3,25	3,00	2,88	3,00	2,88	3,25	
Ricardo	2,58	2,83	3,21	3,00	4,00	2,96	2,83	2,00	
Ricardo	2,63	2,88	3,25	3,00	4,00	3,00	3,00	2,00	

	Vaga Avaliada
	Vaga não recomendada
	Vaga recomendada