Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Verificação, Validação e Teste de Software (CK0241)

TB03 - Relatório de Análise Estática

Video Splitter

Fernanda Costa de Sousa - 485404
José Douglas Gondim Soares - 485347

Histórico de versões

Versão	Data	Autor	Descrição
1.0	14/10/2022	Fernanda Costa e José Douglas	Criação da versão inicial do relatório de análise estática
1.1	15/10/2022	José Douglas	Atualização do documento: inclusão das seção 2. Resultados gerais e 2.1. Lista de problemas analisados
2.0	15/10/2022	Fernanda Costa	Atualização do documento: inclusão das Seções 1. Introdução, 1.1. aplicação e código fonte e 1.2. Descrição da(s) ferramenta(s) de Análise Estática
3.0	16/10/2022	Fernanda Costa e José Douglas	Atualização do documento: inclusão na seção 2.1. Lista de problemas analisados, adição do E01 ao E13
4.0	17/10/2022	Fernanda Costa e José Douglas	Atualização do documento: inclusão na seção 2.1. Lista de problemas analisados, adição do E14 e E15
5.0	17/10/2022	Fernanda Costa	Criação das Seções: 5. Referência e 6. Glossário
6.0	18/10/2022	José Douglas	Criação da Seção 3. Discussão
7.0	19/10/2022	Fernanda Costa e José Douglas	Criação da Seção 4. Conclusão

Sumário

1. Introdução

Existem muitos fatores que podem afetar a qualidade do software durante o desenvolvimento, sobretudo as falhas que podem se acumular e causar prejuízos ao projeto como um todo, pois podem gerar problemas e vulnerabilidades nos sistemas e aplicativos. Dentre as estratégias existentes uma alternativa para reduzir essas falhas é a análise estática de software.

A análise estática consiste em uma série de verificações automatizadas realizadas no código-fonte. Uma ferramenta de análise estática verifica o código em busca de erros e vulnerabilidades comuns, como vazamentos de memória ou estouros de buffer. A análise também pode impor padrões de codificação.[1]

Este relatório tem como objetivo apresentar os resultados da análise estática realizada no projeto do aplicativo Video Splitter, apresentando possíveis soluções para os problemas encontrados.

1.1. Aplicação e código fonte

A aplicação Video Splitter é um software de manipulação de vídeos que permite ao usuário cortar um vídeo qualquer em n partes de tamanhos iguais, para então utilizá-las de diversas formas criativas.

A aplicação não é open source e foi desenvolvida utilizando a linguagem de programação Swift da Apple com algum auxílio do codex FFmpeg. O seu escopo é limitado apenas a dispositivos que rodam o sistema operacional iOS. Todos os códigos fonte do projeto estão disponíveis em: https://drive.google.com/file/d/1O7PVoGmku8prmgkYJs86lSUSGtjDLixZ/view?usp=sharing

1.2. Descrição da(s) ferramenta(s) de Análise Estática

Foi utilizada a ferramenta SwiftLint, que é uma ferramenta de código aberto para impor o estilo e convenções. No SwiftLint é possível definir as regras de estilo de codificação e forçá-las durante o desenvolvimento.[2]

São utilizadas diferentes regras e mais de duzentas delas estão incluídas no SwiftLint, que inclusive podem ser encontradas aqui neste endereço: https://realm.github.io/SwiftLint/rule-directory.html e as suas implementações aqui: source/Swift Lint Framework/Rules. Vale salientar que é possível customizar regras no SwiftLint e além disso, também estão disponíveis opt_in_rules (regras opcionais), que são desabilitadas por padrão (ou seja, você deve habilitá-las explicitamente em um arquivo de configuração). As diretrizes sobre quando marcar uma regra como opt-in são:

- Uma regra que pode ter muitos falsos positivos (por exemplo, empty_count);
- Uma regra que é muito lenta;

- Uma regra que não é consenso geral ou é útil apenas em alguns casos (por exemplo, force_unwrapping);

Os problemas encontrados são do tipo:

- Style: relacionado ao estilo do código, comprimento da linha, tabulações e espaços;
- Idiomatic: regras que validam práticas comuns da comunidade swift;
- Lint: regras que validam as convenções;
- Metrics: regras que validam medidas e magnitudes;
- Performance: regras que evitam padrões de código que causam baixa performance.

Cada problema possui uma severidade. São elas: Error e Warning.

O SwiftLint se integra ao Xcode e lista os erros encontrados na mesma aba de erros que o Xcode identifica.

A ferramenta Codacy também foi utilizada. Ela automatiza revisões de código e monitora a qualidade do código em cada solicitação de confirmação e por pull. Ela relata o impacto de cada solicitação de commit ou pull em novos problemas relacionados ao estilo de código, melhores práticas, segurança e muitos outros. Ela monitora as alterações na cobertura de código, duplicação de código e complexidade de código. [3] As suas categorias são: Segurança, Propensão a erros, Estilo, Compatibilidade, Código não utilizado e desempenho. Vale observar que esta ferramenta somente foi utilizada para uma melhor visualização dos problemas encontrados no projeto e possibilitar uma comparação entre os resultados obtidos com a ferramenta SwiftLint.

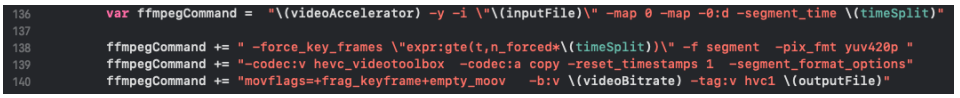
2. Resultados gerais

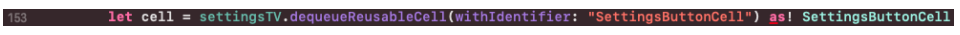
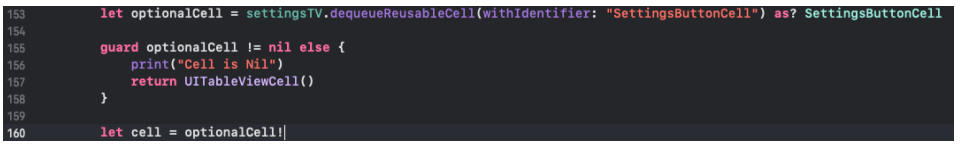
A ferramenta foi utilizada em todo escopo da aplicação. É esperado que ela indique uma quantidade significativa de erros considerando que a análise será feita de maneira global. Logo, muitos problemas relacionados a Métricas e Estilos são esperados.

2.1. Lista de problemas analisados

As tabelas abaixo indicam os problemas encontrados e possuem os seguintes apontamentos: Identificador (Um rótulo para o problema encontrado), Ferramentas envolvidas, categoria, a localização do problema, a mensagem exibida, o trecho de código, sua proposta de solução e um campo para comentário.

Identificador	E01
Ferramentas envolvidas	SwiftLint

Categoria	Metric / Level: Error
Localização	VideoSplitter/Controller/ExportController.swift - Linha 136
Mensagem	Line Length Violation: Line should be 200 characters or less: currently 373 characters (line_length)
Trecho do código	
Proposta de solução	
Comentários	A linha de código estava muito longa então ela precisou ser dividida em partes.

Identificador	E02
Ferramentas envolvidas	SwiftLint
Categoria	Idiomatic / Level: Error
Localização	VideoSplitter/Controller/SettingsController.swift - Linha 153
Mensagem	Force Cast Violation: Force casts should be avoided. (force_cast)
Trecho do código	
Proposta de solução	
Comentários	O casting de UITableViewCell? para SettingsButtonCell estava sendo forçado e a solução foi colocar uma guarda para caso o retorno de dequeueReusableCell() seja Nil, imprimir mensagem de erro e retornar uma UITableViewCell vazia.

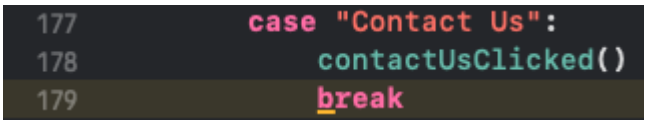
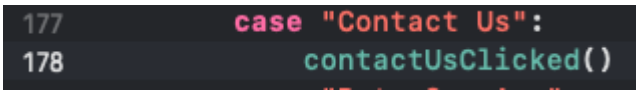
Identificador	E03
Ferramentas envolvidas	SwiftLint
Categoria	Style / Level: Warning

Localização	VideoSplitter/Controller/SplitterController.swift - Linha 522
Mensagem	Control Statement Violation: `if`, `for`, `guard`, `switch`, `while`, and `catch` statements shouldn't unnecessarily wrap their conditionals or arguments in parentheses. (control_statement)
Trecho do código	
Proposta de solução	
Comentários	Em Swift não é necessário colocar parênteses numa condição simples. A solução foi remover os parênteses da solução.

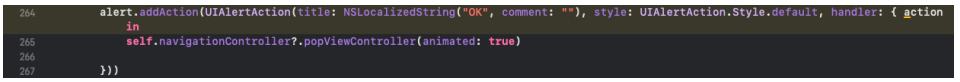
Identificador	E04
Ferramentas envolvidas	SwiftLint
Categoria	Style / Level: Warning
Localização	VideoSplitter/Controller/SplitterController.swift - Linha 575
Mensagem	Opening Brace Spacing Violation: Opening braces should be preceded by a single space and on the same line as the declaration. (opening_brace)
Trecho do código	
Proposta de solução	
Comentários	A solução foi adicionar um espaço após a condição do if.

Identificador	E05
Ferramentas envolvidas	SwiftLint
Categoria	Lint / Level: Warning
Localização	VideoSplitter/Controller/SplitterView.swift - Linha 212
Mensagem	Comment Spacing Violation: Prefer at least one space after slashes for comments. (comment_spacing)
Trecho do código	

Proposta de solução	`212 // Function that sets up progress bars`
Comentários	É uma convenção do Swift colocar um espaço após o “//” de um comentário.

Identificador	E06
Ferramentas envolvidas	SwiftLint
Categoria	Idiomatic / Level: Warning
Localização	VideoSplitter/Controller/SettingsController.swift - Linha 179
Mensagem	Unneeded Break in Switch Violation: Avoid using unneeded break statements. (unneeded_break_in_switch)
Trecho do código	
Proposta de solução	
Comentários	O “break” dentro do switch case era desnecessário e podia ser removido.

Identificador	E07
Ferramentas envolvidas	SwiftLint
Categoria	Idiomatic / Level: Warning
Localização	VideoSplitter/Controller/SplitterController.swift - Linha 30
Mensagem	Redundant Optional Initialization Violation: Initializing an optional variable with nil is redundant. (redundant_optional_initialization)
Trecho do código	
Proposta de solução	
Comentários	É redundante atribuir “nil” a inicialização de uma variável opcional.

Identificador	E08
Ferramentas envolvidas	SwiftLint
Categoria	Lint / Level: Warning
Localização	VideoSplitter/Controller/ExportController.swift - Linha 136
Mensagem	Unused Closure Parameter Violation: Unused parameter "action" in a closure should be replaced with _. (unused_closure_parameter)
Trecho do código	
Proposta de solução	
Comentários	Como a variável “action” do closure não estava sendo utilizada, podemos substituí-la por um “_”.

Identificador	E09
Ferramentas envolvidas	SwiftLint
Categoria	Idiomatic / Level: Warning
Localização	VideoSplitter/Controller/ExportController.swift - Linha 14
Mensagem	Explicit Type Interface Violation: Properties should have a type interface (explicit_type_interface)
Trecho do código	
Proposta de solução	
Comentários	O tipo da variável deve ser declarado explicitamente.

Identificador	E10
Ferramentas envolvidas	SwiftLint

Categoria	Lint / Level: Warning
Localização	VideoSplitter/Controller/ExportController.swift
Mensagem	Required Deinit Violation: Classes should have an explicit deinit method. (required_deinit)
Trecho do código	-
Proposta de solução	
Comentários	Toda classe precisa ter seu método destrutor declarado. A solução foi declarar um destrutor vazio.

Identificador	E11
Ferramentas envolvidas	SwiftLint
Categoria	Idiomatic / Level: Warning
Localização	VideoSplitter/Controller/SplitterController.swift - Linha 475
Mensagem	Explicit ACL Violation: All declarations should specify Access Control Level keywords explicitly. (explicit_acl)
Trecho do código	
Proposta de solução	
Comentários	Todas as declarações de variáveis e funções precisam ter seu nível de acesso declarado (public, private, etc.).

Identificador	E12
Ferramentas envolvidas	SwiftLint
Categoria	Lint / Level: Warning
Localização	VideoSplitter/Controller/ExportView.swift - Linha 11
Mensagem	Weak Delegate Violation: Delegates should be weak to avoid reference cycles. (weak_delegate)

Trecho do código	
Proposta de solução	
Comentários	No padrão de delegação, é importante declarar a variável delegate como weak var para evitar problemas de ciclos de referência (loop).

Identificador	E13
Ferramentas envolvidas	SwiftLint
Categoria	Idioamtic / Level: Warning
Localização	VideoSplitter/Controller/SplitterController.swift - Linha 475
Mensagem	Explicit ACL Violation: All declarations should specify Access Control Level keywords explicitly. (explicit_acl)
Trecho do código	
Proposta de solução	
Comentários	Todas as declarações de variáveis e funções precisam ter seu nível de acesso declarado (public, private, etc.).

Identificador	E14
Ferramentas envolvidas	SwiftLint
Categoria	Performance / Level: Warning
Localização	VideoSplitter/Controller/HomeController.swift - Linha 80
Mensagem	Empty String Violation: Prefer checking `isEmpty` over comparing `string` to an empty string literal. (empty_string)
Trecho do código	
Proposta de solução	
Comentários	Para saber se uma string é vazia o correto é usar a função isEmpty. Comparar uma string com outra string vazia é um problema de performance.

Identificador	E15
Ferramentas envolvidas	SwiftLint
Categoria	Lint / Level: Warning
Localização	VideoSplitter/Controller/SplitterController.swift - Linha 384
Mensagem	Unused Enumerated Violation: When the item is not used, `.indices` should be used instead of `.enumerated()`. (unused_enumerated)
Trecho do código	`384 for (idx, _) in progressBars.enumerated(){`
Proposta de solução	`384 for (idx) in progressBars.indices{`
Comentários	Como o ítem não estava sendo usado no laço de repetição, o enumerated era desnecessário e podemos invocar somente os índices.

3. Discussão

Como apresentado em tópicos anteriores, a ferramenta SwiftLint foi utilizada em todo escopo da aplicação. Porém como não existe uma funcionalidade com processamento de dados em formato de imagem, para uma melhor visualização optamos por utilizar também a ferramenta Codacy para uma melhor visualização e análise dos problemas encontrados.

No SwiftLint, 1415 problemas foram encontrados, sendo 8 erros e os demais warnings que podem vir a gerar erros. Enquanto na ferramenta Codacy foram encontrados 440 problemas distribuídos em diferentes categorias, como podemos ver na imagem abaixo:

Figura 01 - Captura de tela dos problemas encontrados pela ferramenta Codacy.

A imagem acima mostra a classificação dos problemas encontrados com a ferramenta Codacy. Poucas variações nas categorias foram observadas nas duas ferramentas.

O SwiftLint possui integração com Xcode o que facilitou bastante seu uso. A ferramenta Codacy faz análise a partir dos commits de código no github e possui interface muito intuitiva, foi utilizada a partir da leitura do código no Github.

4. Conclusão

Como apresentado nos tópicos anteriores, o documento aborda a análise estática da aplicação Video Splitter, realizada a partir da ferramenta SwiftLint, onde foi possível analisar os problemas do código. Na ferramenta foi possível definir regras customizadas, além de ativar e desativar determinadas regras.

Neste documento foram ilustrados os diferentes problemas encontrados, trechos de código e suas localizações, além de uma sugestão de solução para cada problema. Após a análise, concluímos que o código possuía muitos erros e de diferentes categorias, sendo os do tipo idiomatic e Lint os com o maior número de ocorrências.

5. Referências

1. O que é a análise de código estático? Disponível em: <https://www.jetbrains.com/pt-br/teamcity/ci-cd-guide/concepts/static-code-analysis/>. Acesso em: 16 de outubro de 2022.

2. SwiftLint documentation. Disponível em: <https://github.com/realm/SwiftLint>. Acesso em: 16 de outubro de 2022.

3. Ferramentas para Testes Automatizados, Codacy. Disponível em: <https://www.capterra.com.br/software/144689/codacy>. Acesso em: 18 de outubro de 2022.

Glossário

Termo	Definição
Buffer	Processo de pré-carregamento de dados na memória do computador.
Swift	Swift é uma linguagem de programação desenvolvida pela Apple para desenvolvimento no iOS.
FFmpeg	FFmpeg é um programa em linha de comando que é composto de uma coleção de software livre e bibliotecas de código aberto.
SwiftLint	O Swift Lint é uma ferramenta para impor o estilo e as convenções do Swift.

opt_in_rules	Regras opcionais da ferramenta Swift Lint.