Автоматизований аналіз рівня використання трафіку мобільними застосунками

Виконав:
студент БП-4 «Комп’ютерні науки»
Кучеренко Д.О.

Науковий керівник:
старший викладач
Франків О.О.

Актуальність теми

Вплив на користувацький досвід

швидкість завантаження, відгук інтерфейсу та загальна стабільність роботи

Обмежений ресурс трафіку

тарифні плани та швидкість мережі, навантаження на сервери та енергоспоживання пристрою

Нестача інструментів розробки

економія часу, ресурсів, покращення якості кінцевого продукту

Антипатерни використання трафіку

Тайм-аути та повторення запитів

час очікування на отримання даних

час очікування на завершення передачі даних

HTTP-заголовки

заголовки кешування

cookies

кешування на стороні клієнта

Керування з'єднаннями

використання мобільного трафіку

тип мережевого сервісу

поведінка URLSession при нестабільному з'єднанні

Доставлення вмісту

max кількість одночасних з'єднань

використання «дорогого» трафіку

Полінг

полінг із таймером

полінг із нескінченними циклами

рекурсивний полінг

Префетчинг та фонові задачі

сповіщення про запуск

правила використання Multipath TCP

HTTP-заголовки

Налаштування cookies

httpCookieAcceptPolicy
httpShouldSetCookies

Кешування даних

reloadIgnoringLocalCacheData
returnCacheDataDontLoad

Безпекові заголовки

Заголовки кешування

- Cache-Control
- Expires
- Content-Encoding
- Last-Modified

```
override func visit(_ node: AssignmentExprSyntax) -> SyntaxVisitorContinueKind {
    if let parentNode = node.parent?.as(ExprListSyntax.self),
       let memberAccessNode = parentNode.first?.as(MemberAccessExprSyntax.self) {
        let property = memberAccessNode.declName.baseName.text
        let location = node.startLocation(converter: SourceLocationConverter(fileName: filePath, tree: node.root))
```

HTTP-заголовки

```
    if properties.contains(property) {
        switch property {
        case "httpShouldSetCookies":
            handleCookieSettingAssignment(parentNode: parentNode, location: location)
        case "httpShouldUsePipelining":
            handlePipeliningAssignment(parentNode: parentNode, location: location)
        case "httpCookieAcceptPolicy":
            handleCookiePolicyAssignment(parentNode: parentNode, location: location)
        case "httpAdditionalHeaders":
            handleAdditionalHeadersAssignment(location: location)
        default:
            break
        }
    }
}
return .visitChildren
```

Тайм-аути та повторення запитів

timeoutIntervalForRequest

timeoutIntervalForResource

```
override func visit(_ node: AssignmentExprSyntax) -> SyntaxVisitorContinueKind {
    if let parentNode = node.parent?.as(ExprListSyntax.self) {
        if let memberAccessNode = parentNode.first?.as(MemberAccessExprSyntax.self) {
            let property = memberAccessNode.declName.baseName.text
            let location = node.startLocation(converter: SourceLocationConverter(fileName: filePath, tree: node.root))

            if properties.contains(property),
               let numberLiteral = parentNode.last?.as(IntegerLiteralExprSyntax.self) {
                let timeoutValue = Int(numberLiteral.literal.text)
```

Тайм-аути та повторення запитів

```
switch property {
   case "timeoutIntervalForRequest":
     if let timeout = timeoutValue, timeout < 30 || timeout > 120 {
         warnings.append(AntipatternWarning(
              filePath: filePath,
              line: location.line,
              column: location.column,
              message: "Consider setting a more appropriate timeout value between 30 and 120 seconds
              ))
       }
    case "timeoutIntervalForResource":
      if let timeout = timeoutValue, timeout < 3600 || timeout > 3600 * 60 * 3 {
          warnings.append(AntipatternWarning(
                   filePath: filePath,
                   line: location.line,
                   column: location.column,
                   message: "Consider setting a more appropriate timeout value between 1 hour and 8
          ))
}
```

Lifecycle operations

applicationDidEnterBackground

applicationWillResignActive

```
private let suspiciousFunctions = [
        "URLSession",
        "dataTask",
        "downloadTask",
        "uploadTask",
        "fetch",
        "download",
        "upload",
        "request",
        "performLongTask",
        "process",
        "calculate",
        "compute",
    ]
```

Lifecycle operations

```
override func visit(_ node: FunctionDeclSyntax) -> SyntaxVisitorContinueKind {
    let functionName = node.name.text
    let location = node.startLocation(converter: SourceLocationConverter(fileName: filePath, tree: node.root))
    switch functionName {
        case "applicationDidEnterBackground":
            handleBackgroundMethod(node: node, location: location)
        case "applicationWillResignActive":
            handleResignActiveMethod(node: node, location: location)
        default:
            break
    }

  return .visitChildren
}
```

Полінг

Полінг з використанням таймера

Рекурсивний полінг

Полінг за допомогою нескінченних циклів із затримкою

```
func checkTimerPolling(_ node: FunctionCallExprSyntax) {
    guard let memberAccess = node.calledExpression.as(MemberAccessExprSyntax.self),
        memberAccess.declName.baseName.text == "scheduledTimer",
        let base = memberAccess.base?.as(DeclReferenceExprSyntax.self),
        base.baseName.text == "Timer" else {
            return
        }
```

Полінг

```
let hasRepeatsTrue = node.arguments.contains { arg in
    if arg.label?.text == "repeats",
        let boolExpr = arg.expression.as(BooleanLiteralExprSyntax.self),
        boolExpr.literal.text == "true" {
            return true
        }
    return false
}

if hasRepeatsTrue {
    let location = node.startLocation(converter: SourceLocationConverter(fileName: filePath, tree: node.root))

    warnings.append(AntipatternWarning(
        filePath: filePath,
        line: location.line,
        column: location.column,
        message: "Consider using a more appropriate polling mechanism or setting a more appropriate timeout value"
    ))
}
```

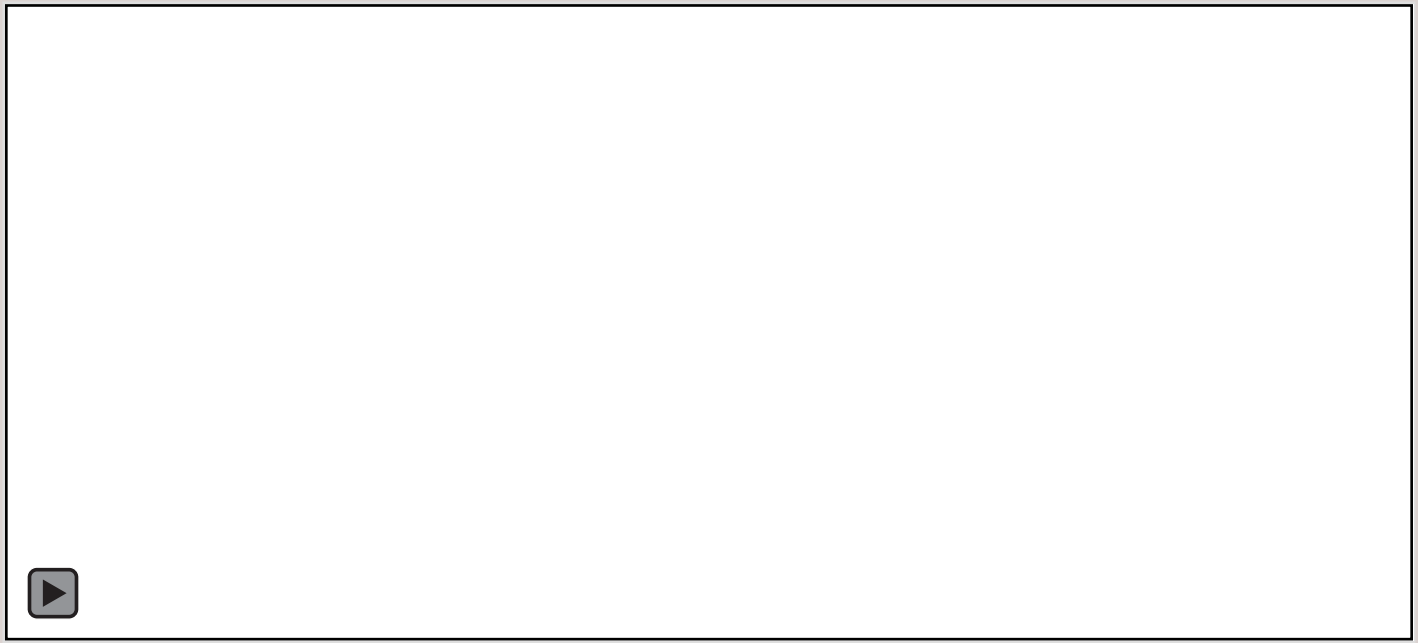
Використані технології та патерни

SwiftSyntax

ArgumentParser

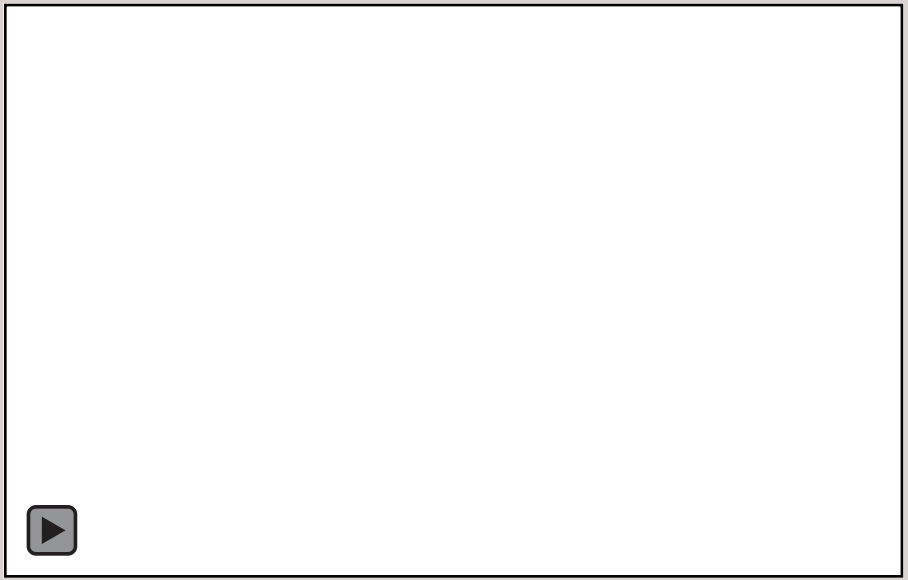
Swift Package Plugins

Принцип роботи

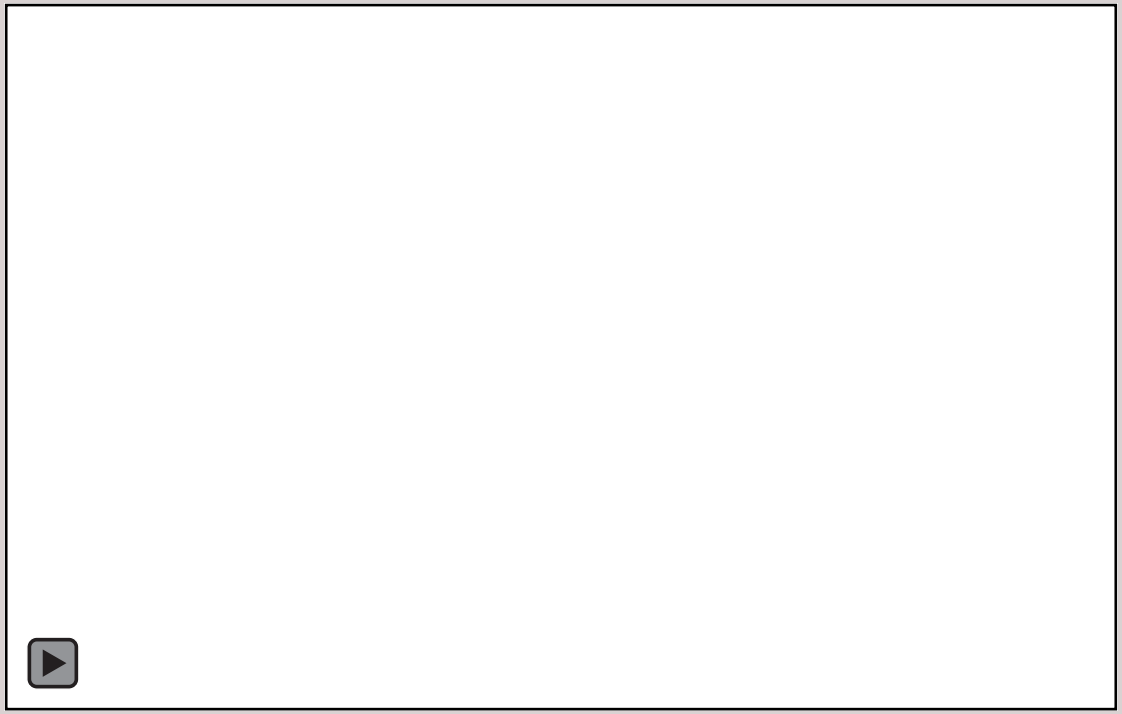


Виклик з терміналу

Інтеграція у середовище розробки



Принцип роботи



Виклик плагіну з середовища розробки

Висновки

Проведено детальний аналіз антипатернів використання мережевого трафіку в мобільних застосунках

Формалізовано правила пошуку антипатернів

Розроблено аналізатор для запобігання виникненню вивчених антипатернів

Дякую за увагу!