Impact Analysis

Как мы ускоряли Unit-тесты

Половкова Алёна

@polovkovaaa

alena.polovkova@mail.ru

А я?
Как же я!

Содержание

Unit-тестирование – это …

Из Википедии

Проверка корректности отдельных модулей исходного кода программы

- Для нетривиальных методов/функций
- Поиск ошибок, появившихся при изменении ранее протестированного кода

Impact Analysis
— это …

Дословно

“Анализ влияния”

В нашем случае

Способ определить, на какие тесты повлияли изменения в коде.

Unit-тестирование
и варианты его ускорения

Варианты запуска Unit-тестов

Запуск **всех тестов** в проекте

Запуск тестов с анализом зависимостей
между модулями

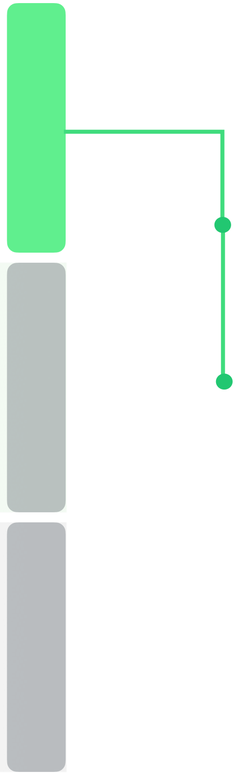
Запуск тестов с анализом зависимостей
между классами

Варианты запуска Unit-тестов

Запуск **всех тестов** в проекте

Запуск тестов с анализом зависимостей **между модулями**

Запуск тестов с анализом зависимостей **между классами**



Самый простой способ

Самые большие затраты времени и ресурсов

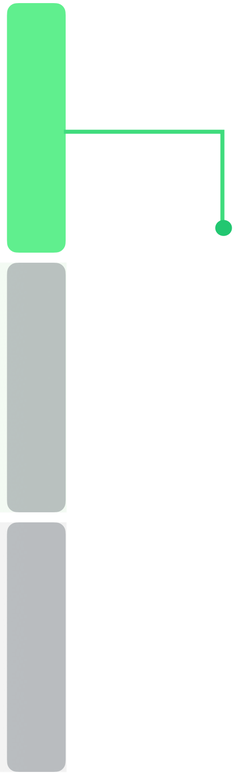
Варианты запуска Unit-тестов

Критерии выбора

Запуск **всех тестов** в проекте

Запуск тестов с анализом зависимостей **между модулями**

Запуск тестов с анализом зависимостей **между классами**



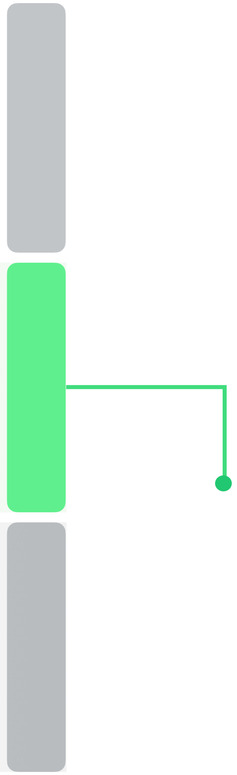
Время прогона тестов в комфортных рамках

Варианты запуска Unit-тестов

Запуск **всех тестов** в проекте

Запуск тестов с анализом зависимостей
между модулями

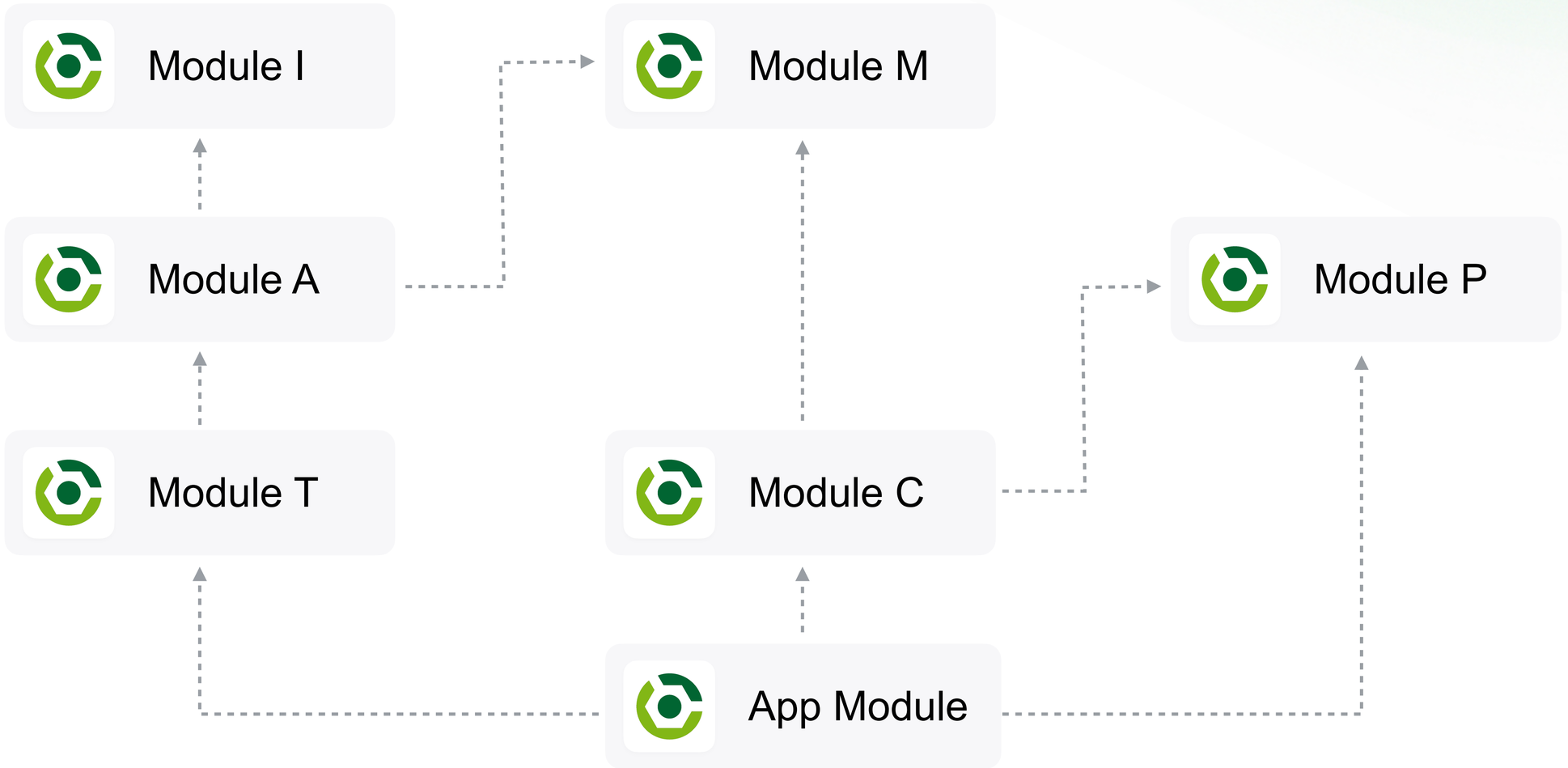
Запуск тестов с анализом зависимостей
между классами



Запуск тестов с Impact Analysis

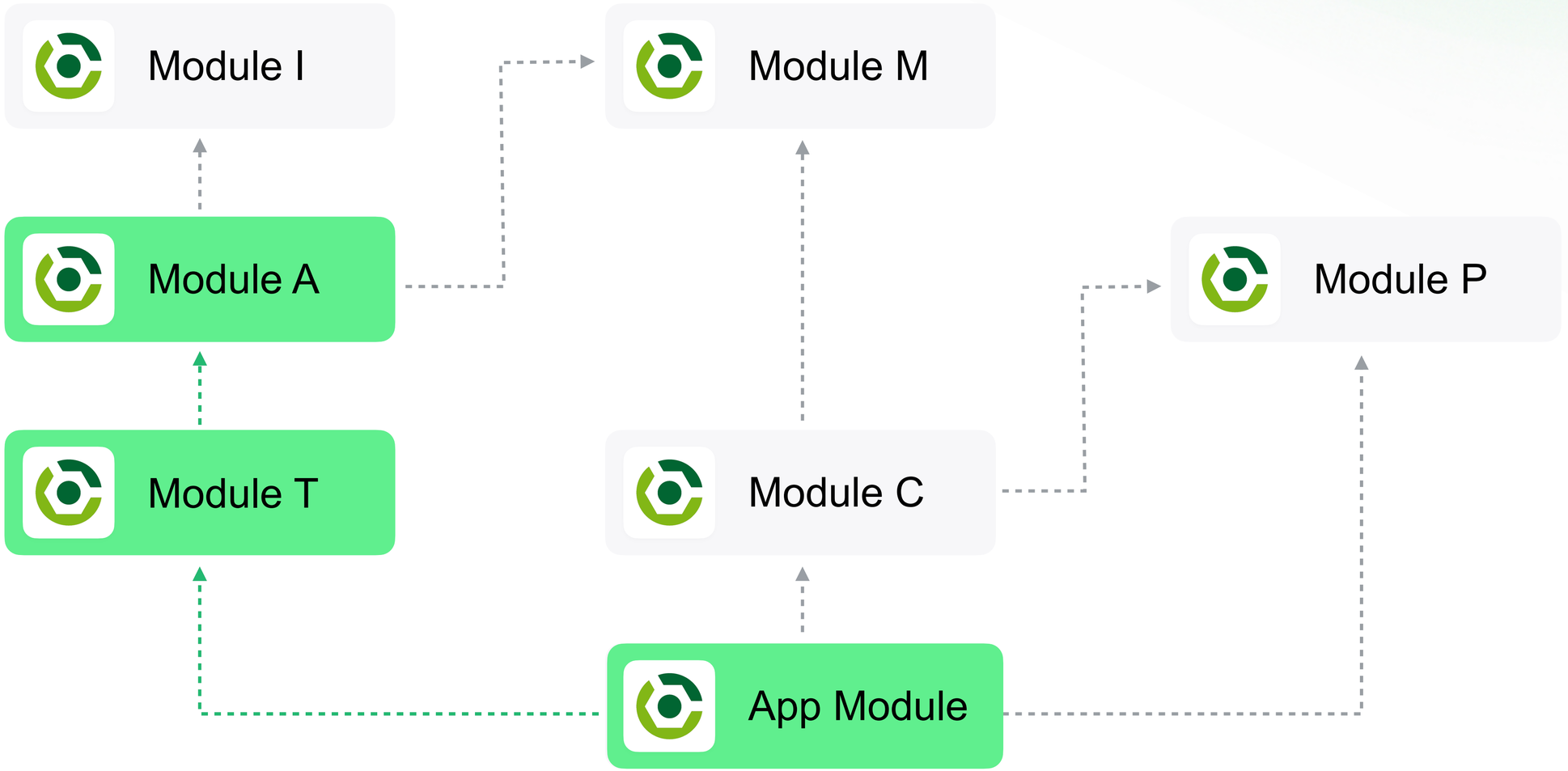
Запуск тестов с анализом зависимостей

между модулями



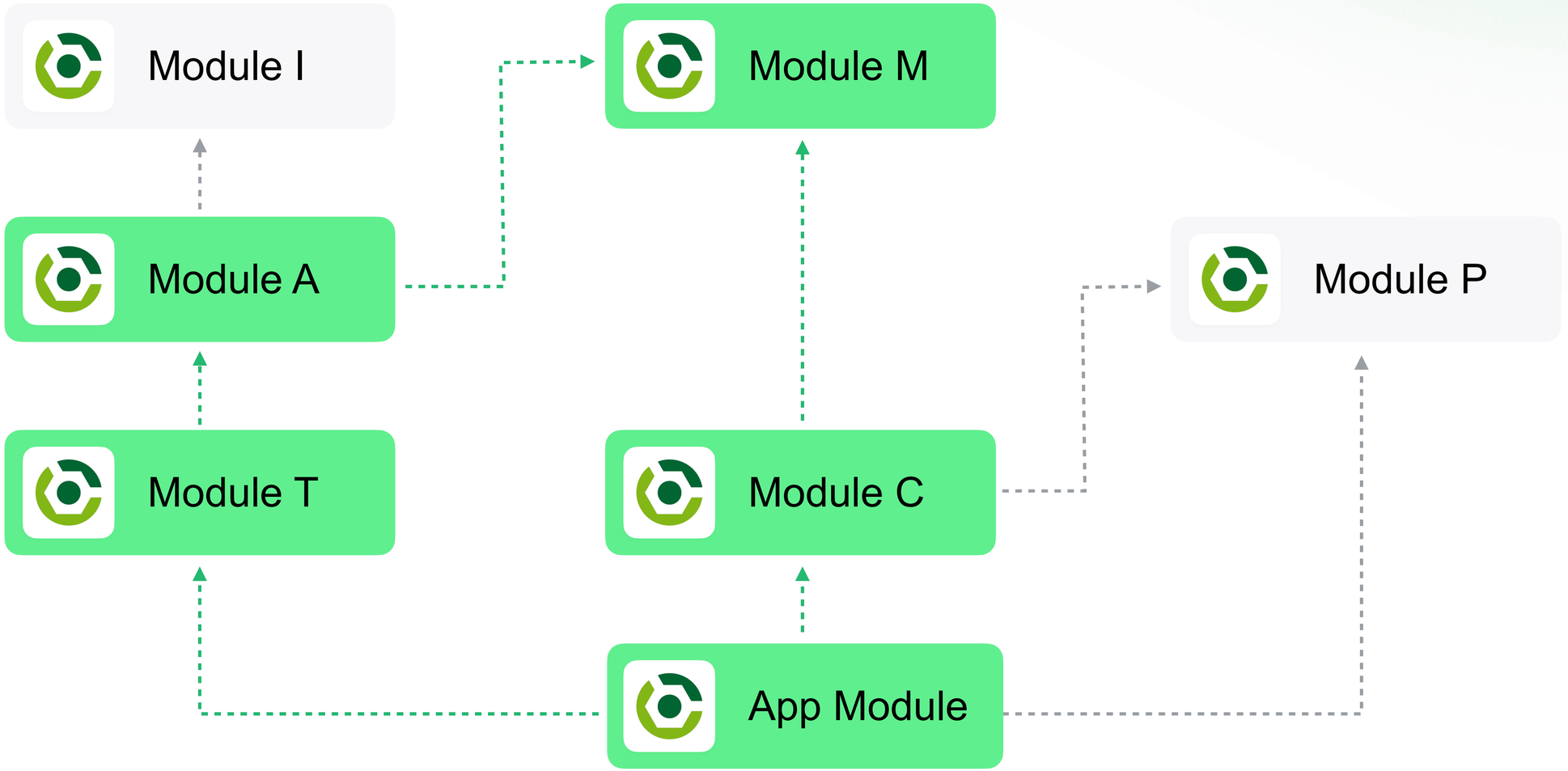
Запуск тестов с анализом зависимостей

между модулями



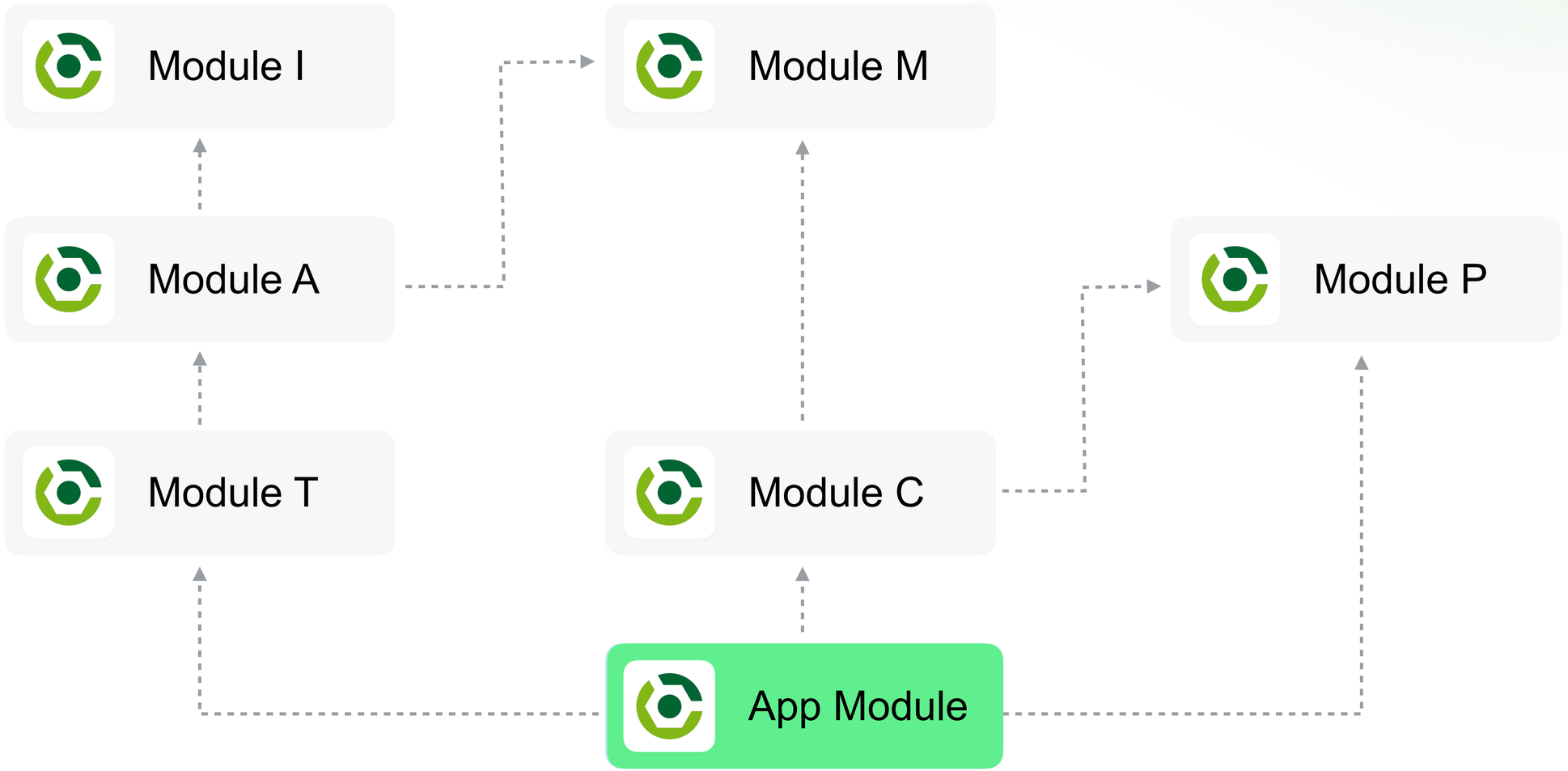
Запуск тестов с анализом зависимостей

между модулями



Запуск тестов с анализом зависимостей

между модулями



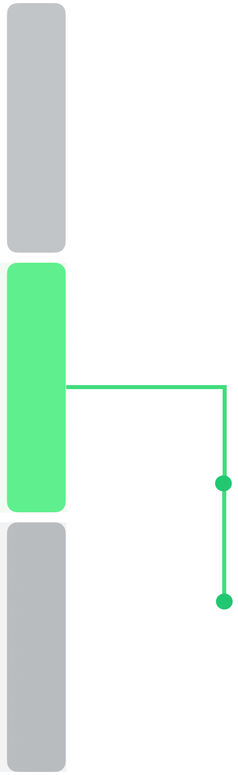
Варианты запуска Unit-тестов

Критерии выбора

Запуск **всех тестов** в проекте

Запуск тестов с анализом зависимостей **между модулями**

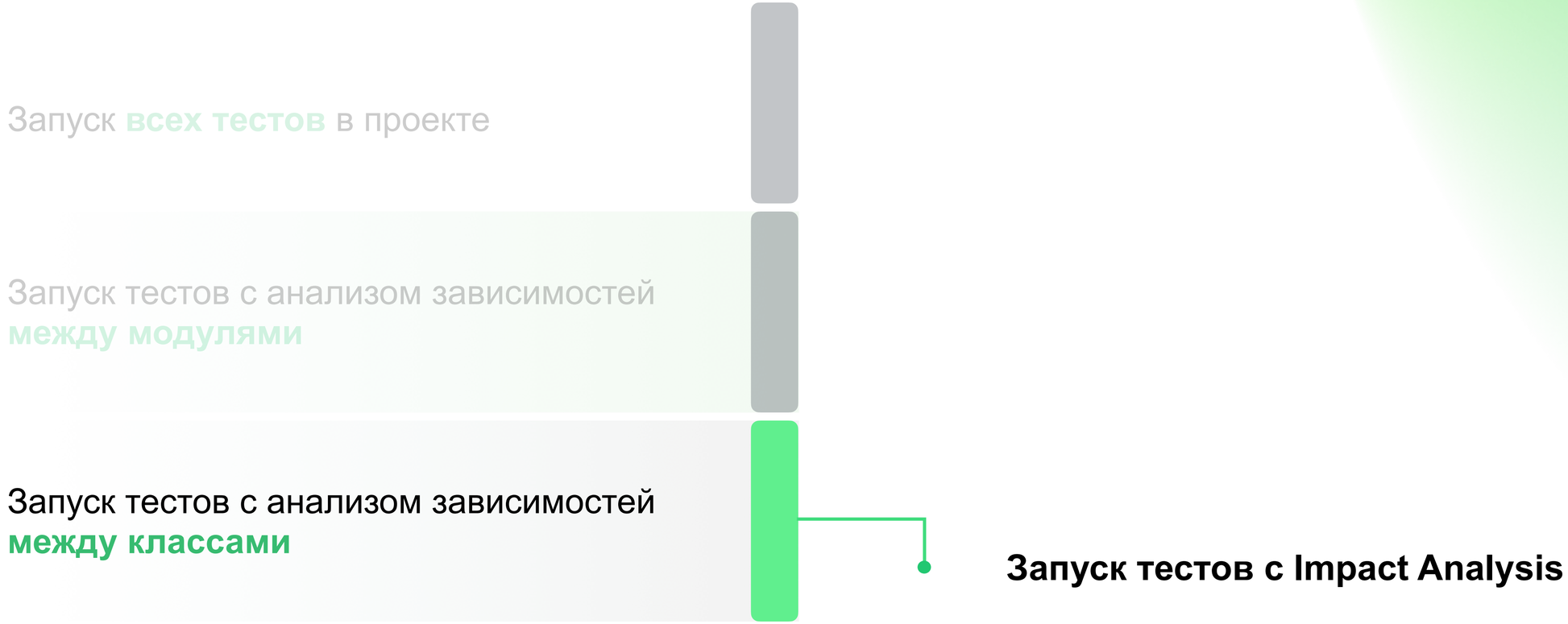
Запуск тестов с анализом зависимостей **между классами**



Разделение модулей на Api-Impl

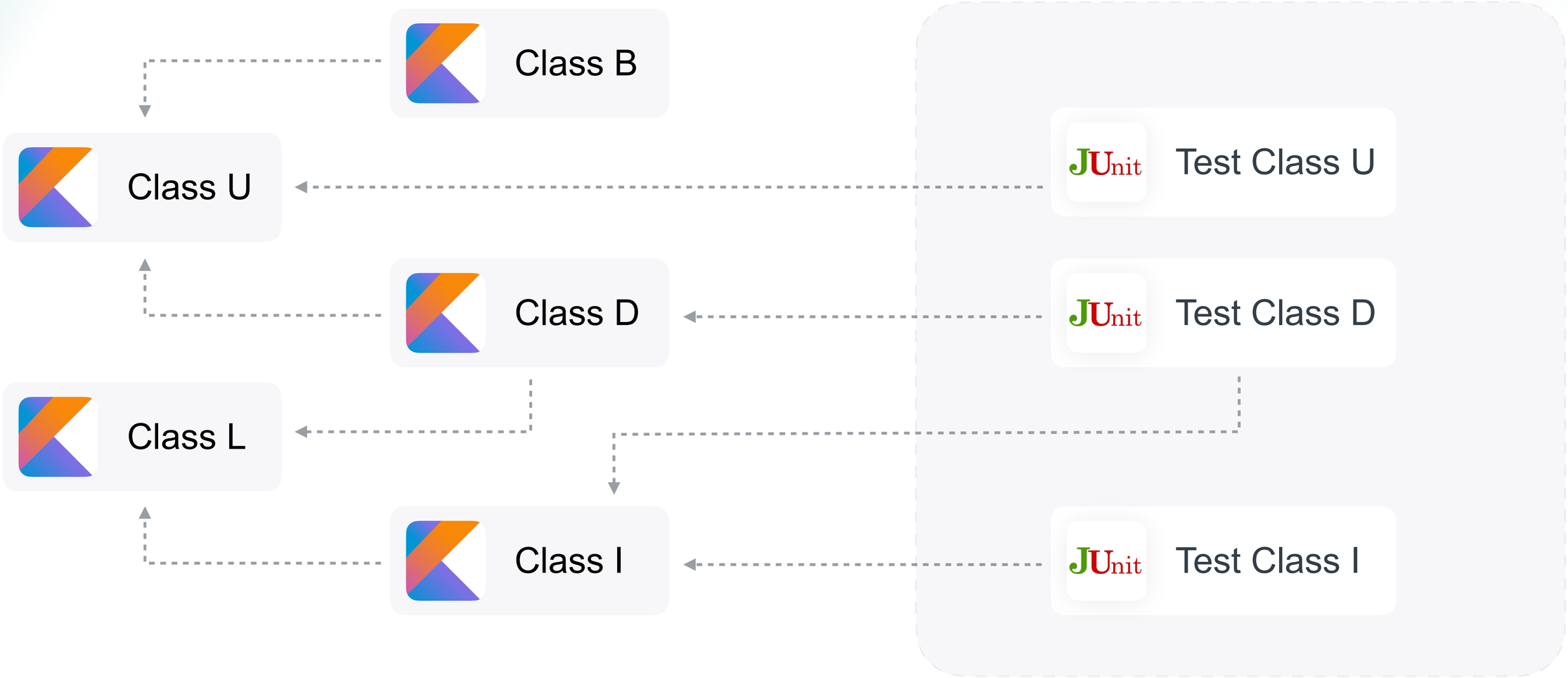
Изменения в Impl-модулях

Варианты запуска Unit-тестов



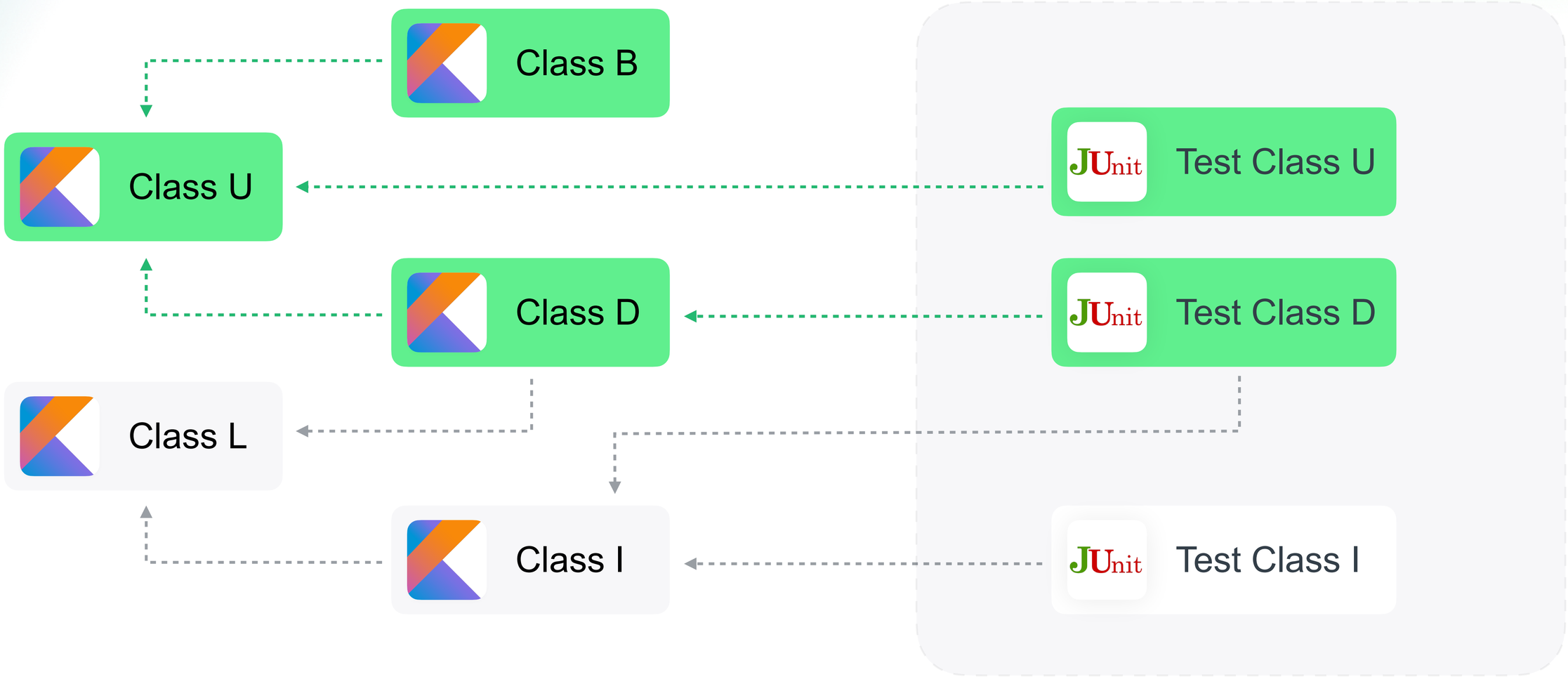
Запуск тестов с анализом зависимостей

между классами



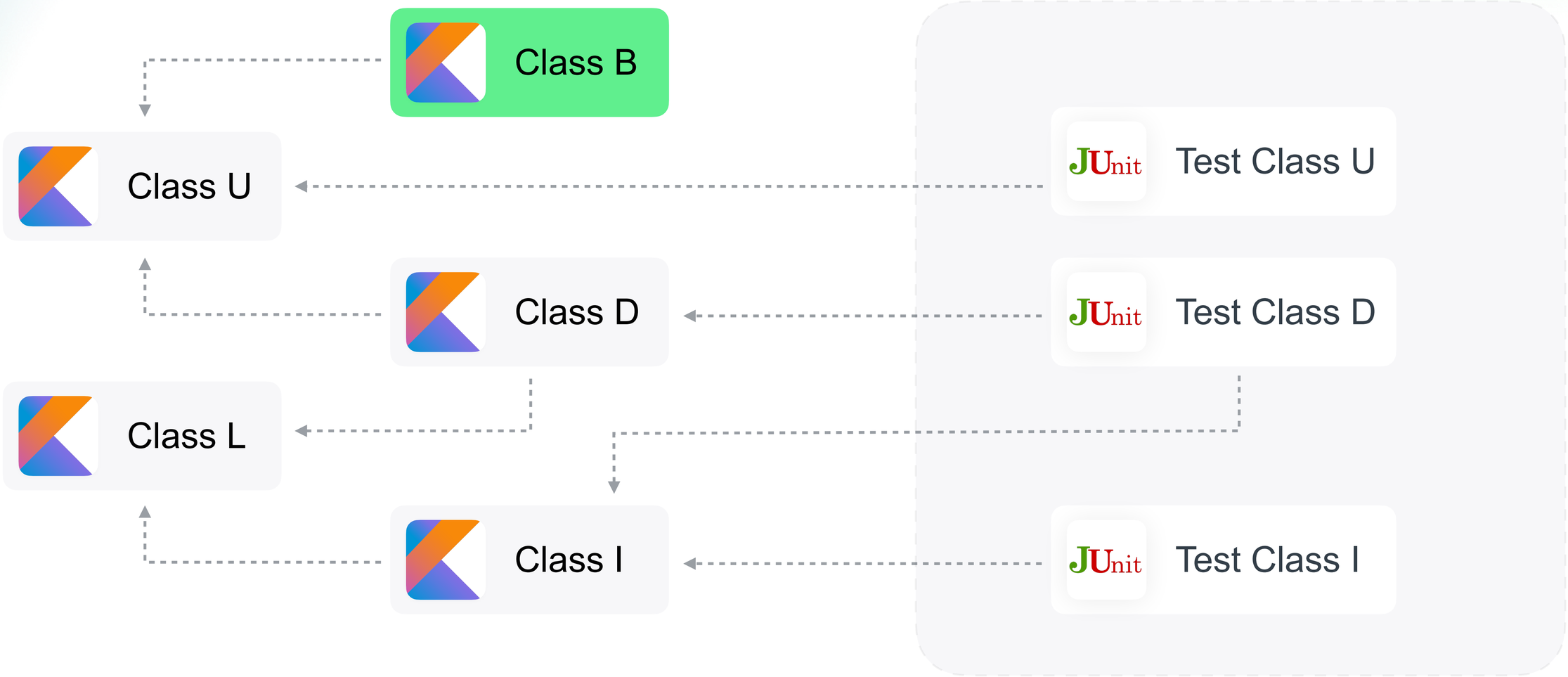
Запуск тестов с анализом зависимостей

между классами



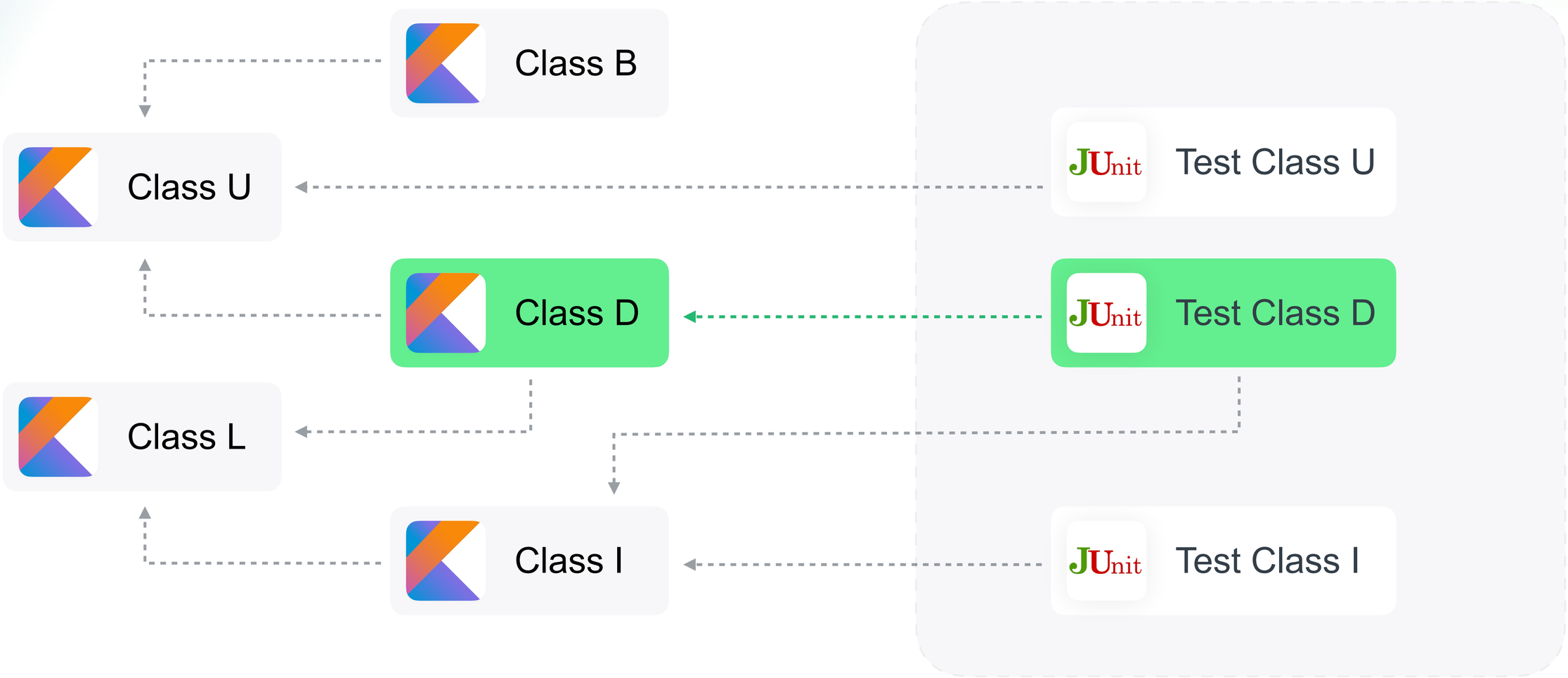
Запуск тестов с анализом зависимостей

между классами



Запуск тестов с анализом зависимостей

между классами



Варианты запуска Unit-тестов

Критерии выбора

Запуск **всех тестов** в проекте

Запуск тестов с анализом зависимостей
между модулями

Запуск тестов с анализом зависимостей
между классами

Изменения в Api-модулях

Разделение модулей на Api-Impl неполное

God-модули, Legacy

Варианты запуска Unit-тестов

Запуск тестов с анализом зависимостей **между методами**

Использование инструментов, вычисляющих **покрытие кода тестами**

Запуск тестов с использованием **прогнозирования нахождения ошибок** тестами

Учитывается **вероятность** того, что тест **найдет ошибку** в измененном коде.

О проекте

Проект

Сбербанк Онлайн Android

3.6 млн.
Количество строк кода

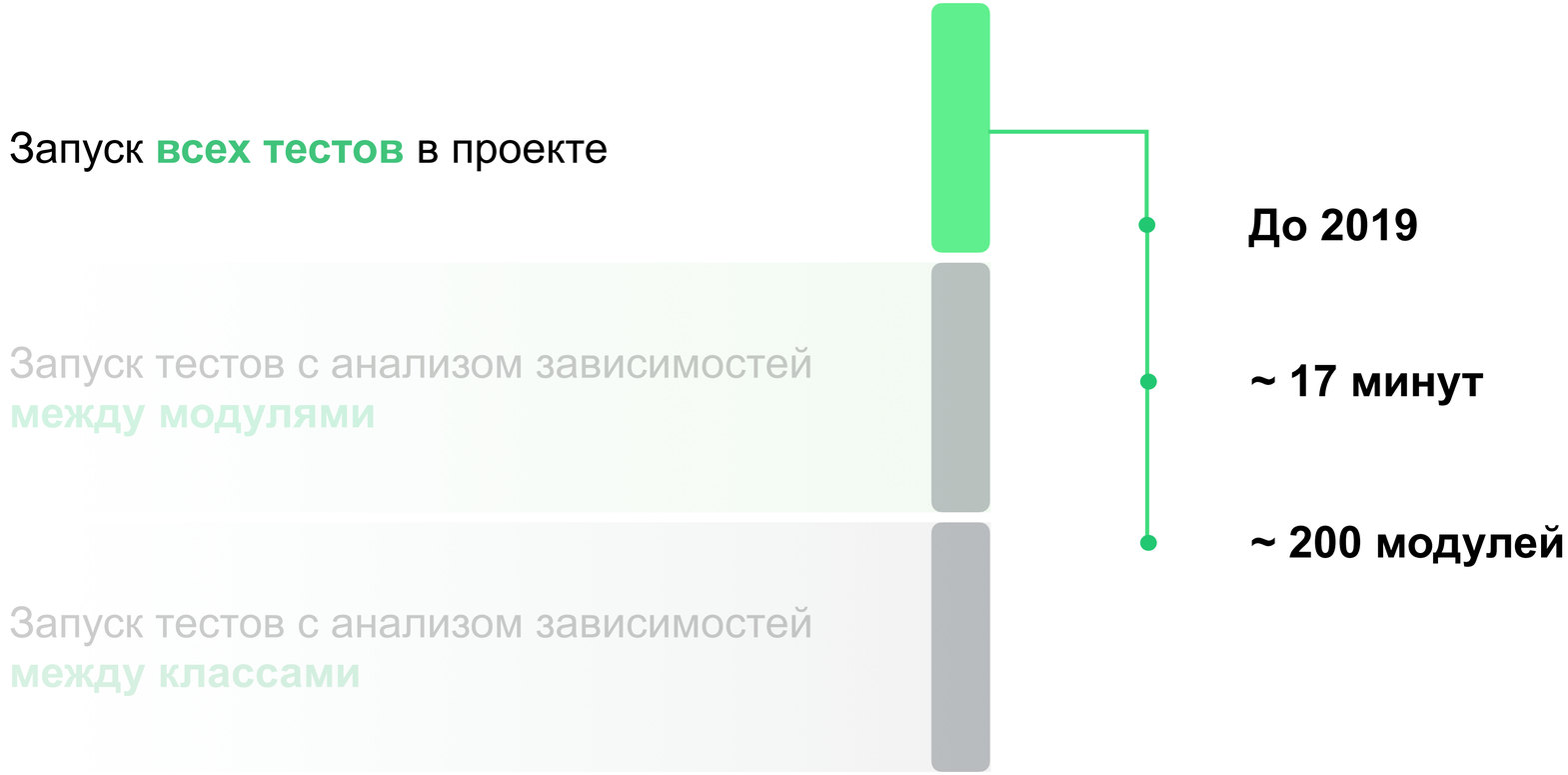
1.1 тыс.
Количество модулей

116 тыс.
Количество тестов

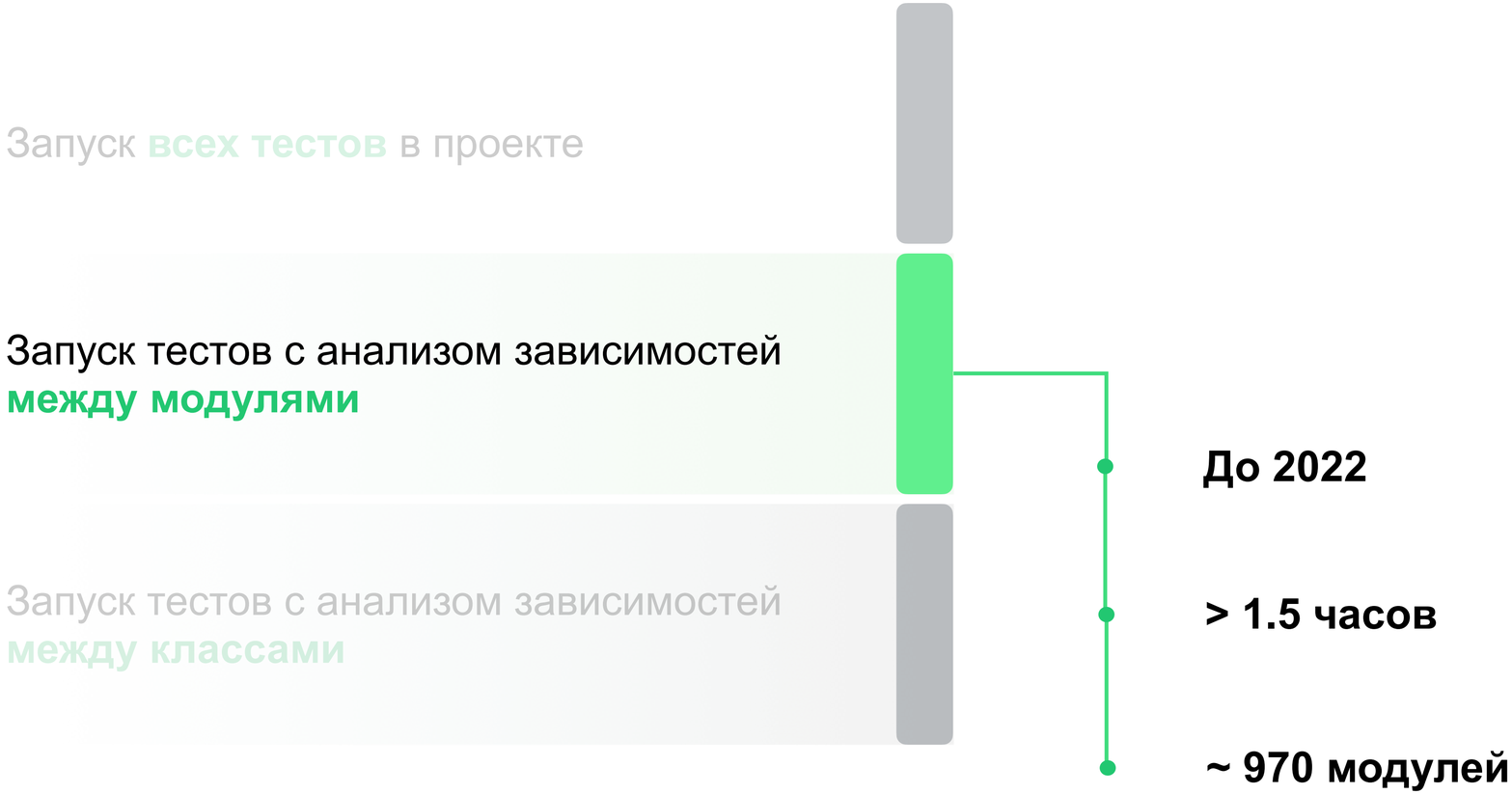
230
Количество разработчиков

80
Среднее количество новых PR в день

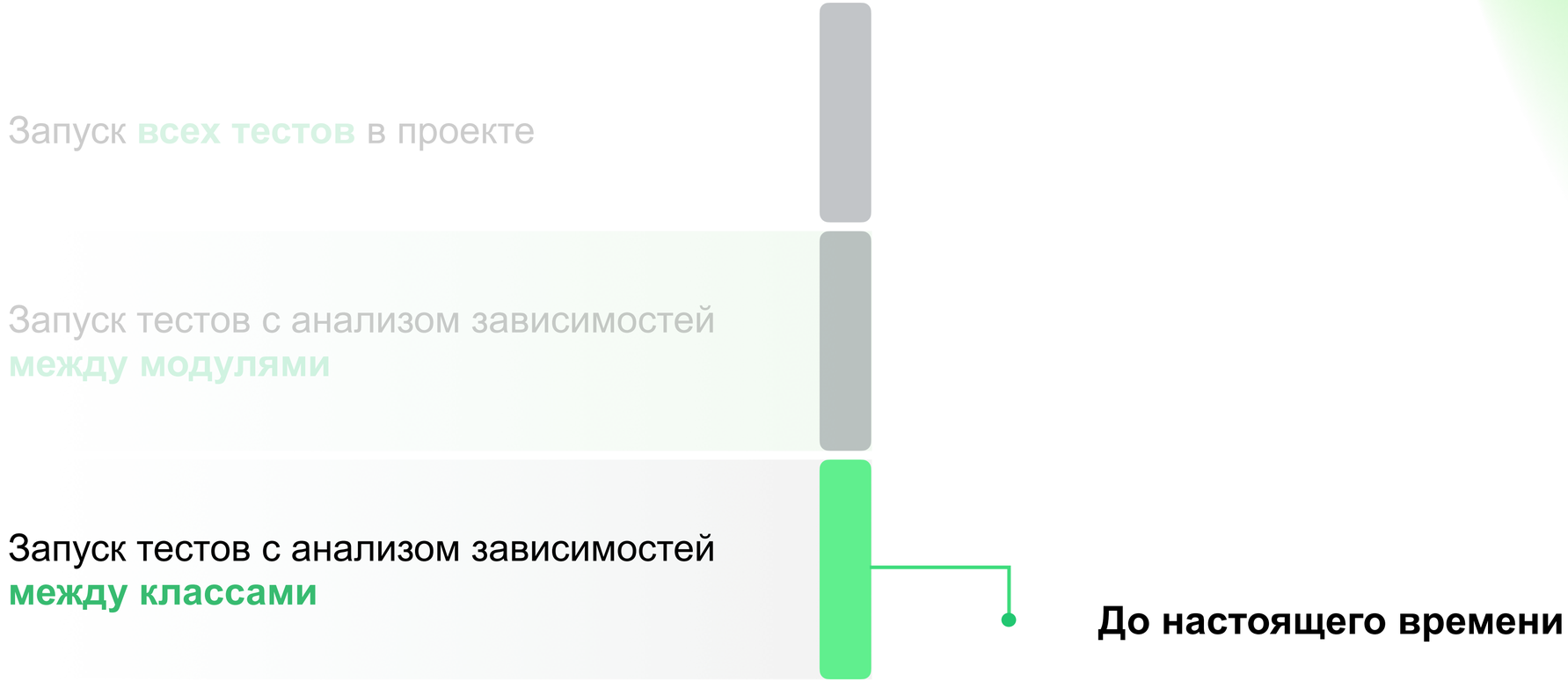
Запуск тестов в Сбербанк Онлайн (Android)



Запуск тестов в Сбербанк Онлайн (Android)



Запуск тестов в Сбербанк Онлайн (Android)

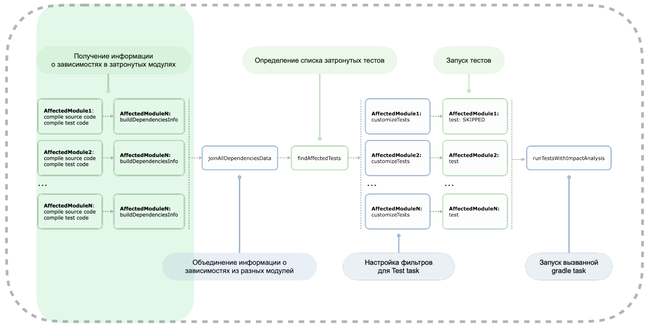


Реализация

Алгоритм работы

Алгоритм работы

Получение информации о зависимостях в модулях



AffectedModule1:
compile source code
compile test code

AffectedModuleN: buildDependenciesInfo
(строим дерево зависимостей между классами и тестами в рамках модуля)

AffectedModule2:
compile source code
compile test code

AffectedModuleN: buildDependenciesInfo

…

…

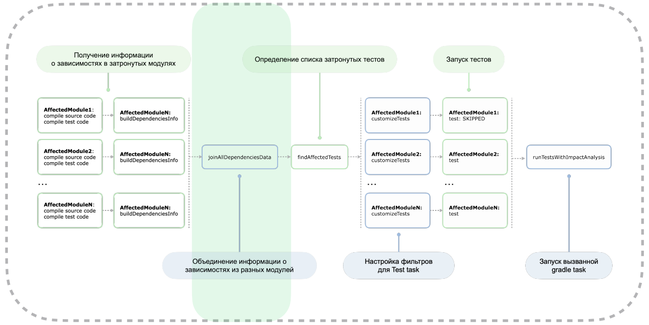
AffectedModuleN:
compile source code
compile test code

AffectedModuleN: buildDependenciesInfo

jo

Алгоритм работы

Объединение информации о зависимостях из разных модулей



joinAllDependenciesData

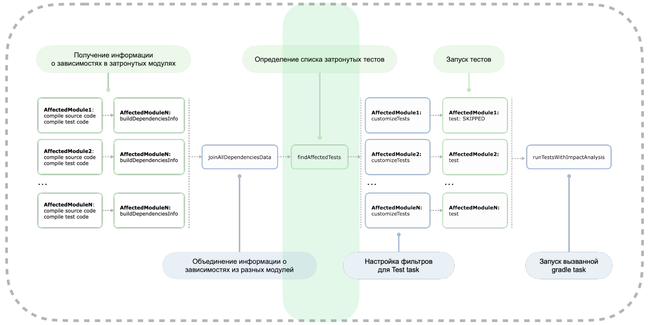
Читает файлы
с данными из модулей

Объединяет
информацию

Записывает всё
в общий файл

Алгоритм работы

Определение списка затронутых тестов

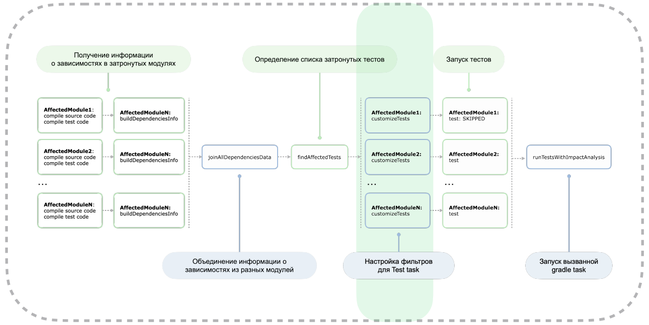


findAffectedTests

По списку измененных классов
определяется список затронутых тестов

Алгоритм работы

Настройка фильтров для Test task



AffectedModule1:
customizeTests

AffectedModule2:
customizeTests

...

AffectedModuleN:
customizeTests

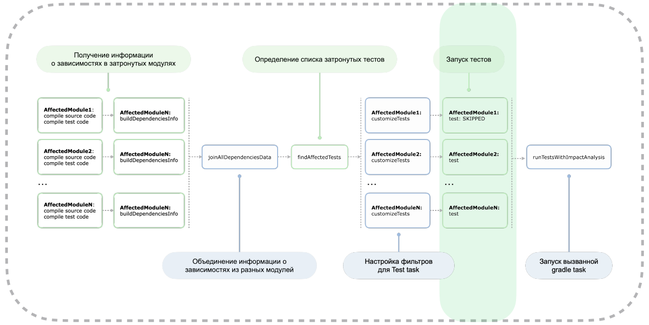
Настраиваем Test task:

— Устанавливаем **фильтр тестов**, если список тестов для прогона не пустой (для примера - Module2 и ModuleN)

— Ставим **статус SKIPPED**, если в модуле прогонять тесты не нужно

Алгоритм работы

Запуск тестов



AffectedModule1: test: SKIPPED

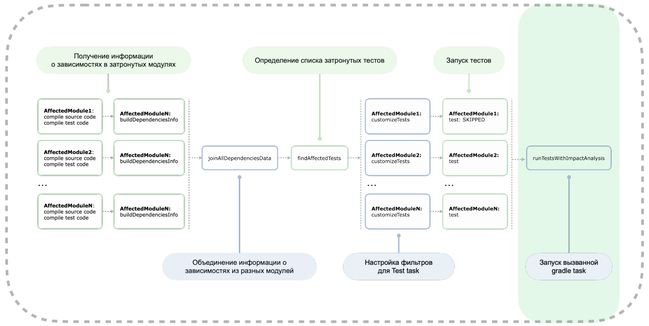
AffectedModule2: test

...

AffectedModuleN: test

Алгоритм работы

Запуск вызванной Gradle task



runTestWithImpactAnalysis

Алгоритм работы

Конфигурация

- Из списка измененных файлов убираются файлы, соответствующие исключениям
- Анализируется оставшийся список измененных файлов
- Настраиваются связи между
  - сбором и анализом зависимостей между классами
  - компиляцией
  - настройкой фильтров для тестов
  - запуском тестов

Алгоритм работы

Конфигурация

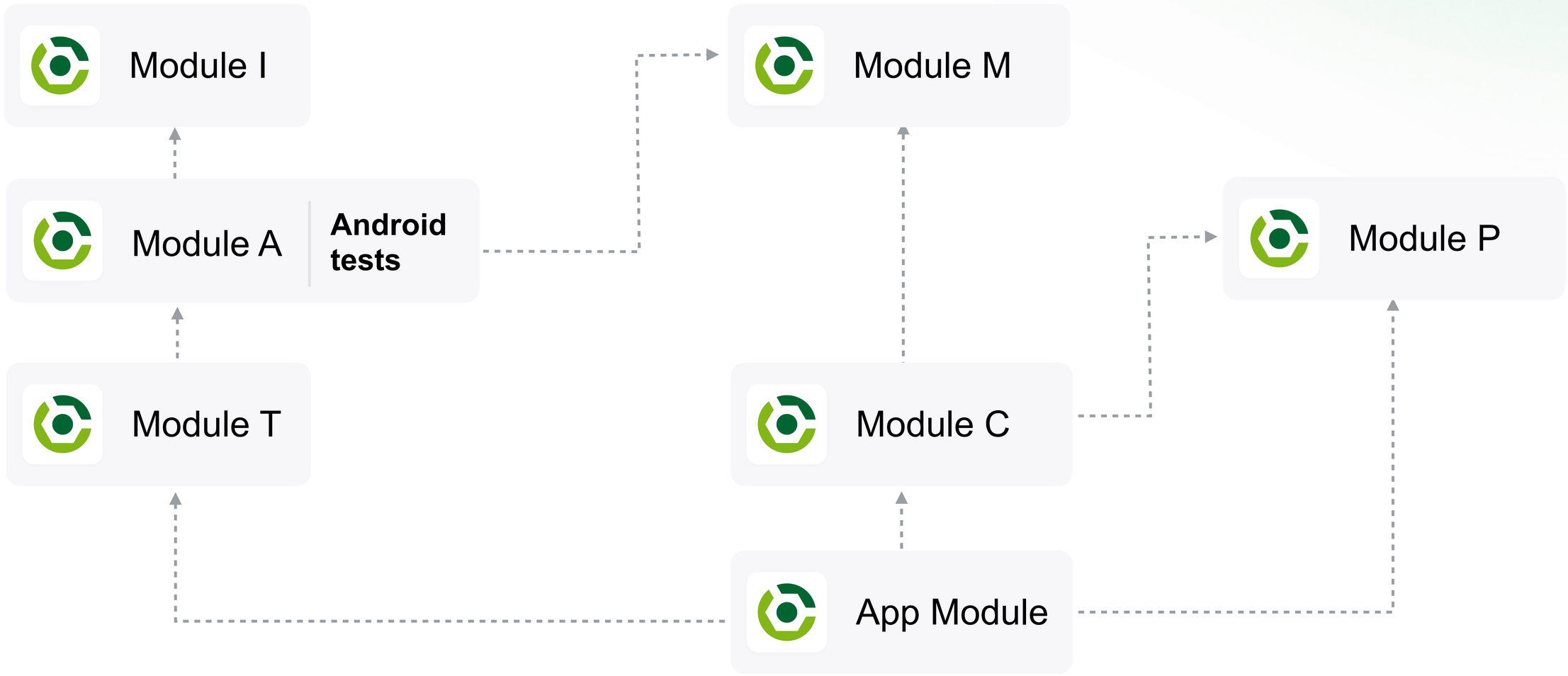
Алгоритм работы

Анализ измененных файлов

Тип файлов	Как запускаются тесты
Файлы в sourceSet для инструментальных тестов	Такие изменения игнорируются, тесты запущены **не будут**.
Исходный код + Android-ресурсы в модуле	Затронутыми модулями считаются **текущий модуль и все, каскадно от него зависимые**. Тесты запускаются в соответствии с **анализом зависимостей** между классами.
Код в sourceSet для Unit-тестов (тесты + вспомогательные классы + android ресурсы)	Затронутым считается только **текущий модуль**. Тесты запускаются в соответствии с **анализом зависимостей** между классами
Остальные файлы в sourceSet для Unit-тестов (например, java-ресурсы)	Затронутым считается только **текущий модуль**. Тесты запускаются в модуле **полностью**, но модуль добавляется в список для **анализа зависимостей**, тк может быть промежуточным звеном в дереве зависимостей.
Остальные файлы в модуле (например, build.gradle)	Затронутыми модулями считаются **текущий модуль и все, каскадно от него зависимые**. Тесты в модулях запускаются **полностью**, анализ зависимостей классов не выполняется.
Файлы в проекте вне модулей (например, settings.gradle)	Затронутыми считаются **все модули** проекта. Тесты по всему проекту запускаются **полностью**, анализ зависимостей не выполняется.

Анализ изменённых файлов

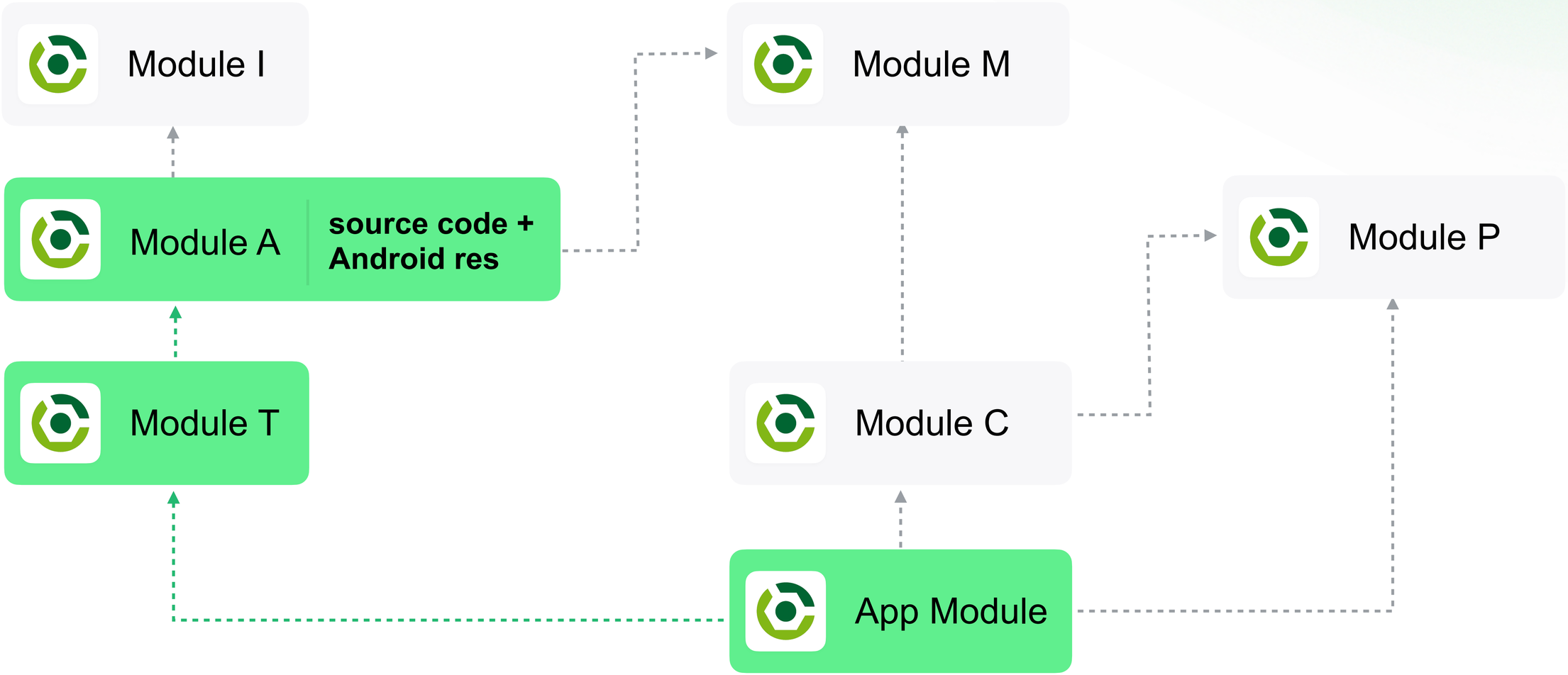
SourceSet инструментальных тестов




НЕПЛОХО

Анализ изменённых файлов

Исходный код + Android-ресурсы

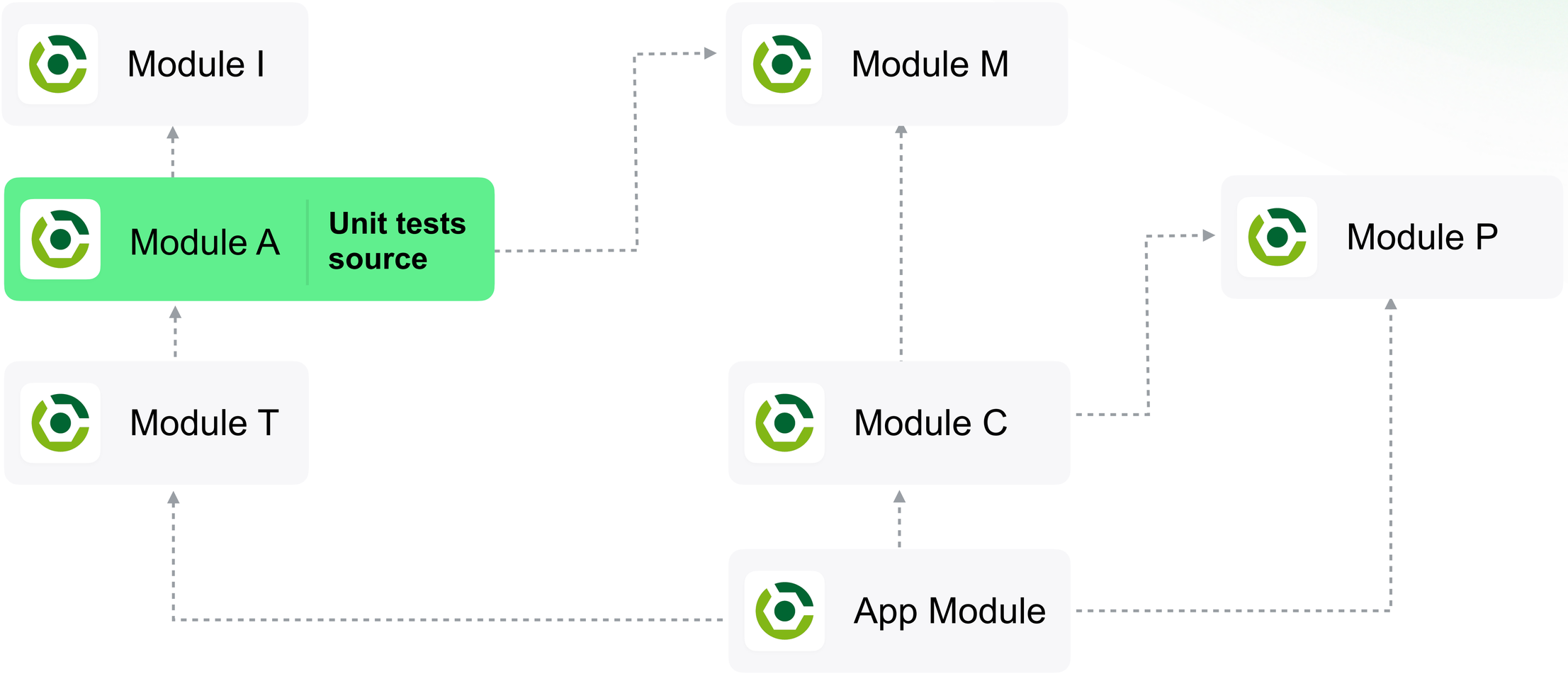


Запуск тестов с анализом зависимостей между классами

Полный запуск тестов в модуле

Анализ изменённых файлов

Исходный код в Unit-тестах



Запуск тестов с анализом зависимостей между классами

Полный запуск тестов в модуле

Анализ изменённых файлов

Анализ измененных файлов и запуск тестов

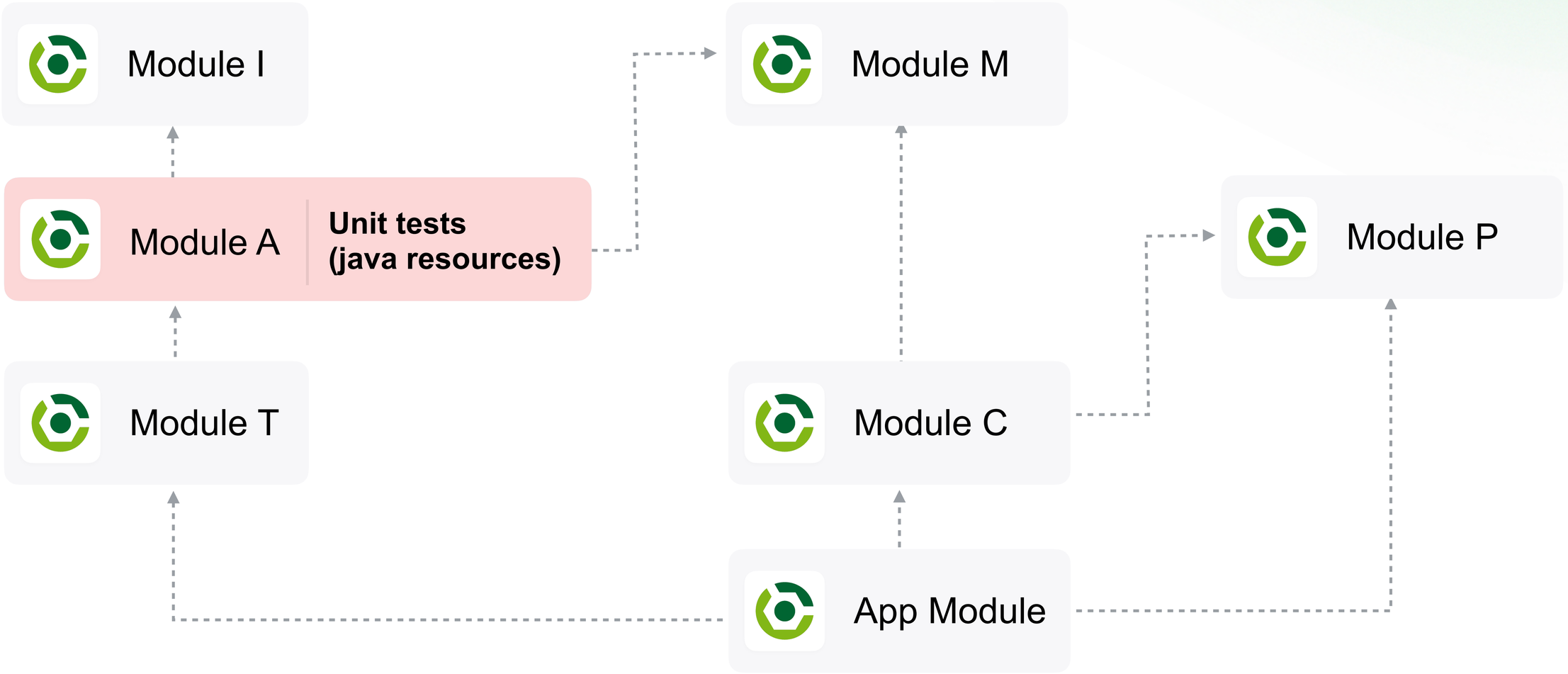
Модули с анализом зависимостей между классами

Алгоритм работы

Конфигурация

Анализ изменённых файлов

Остальные файлы в sourceSet Unit-тестов

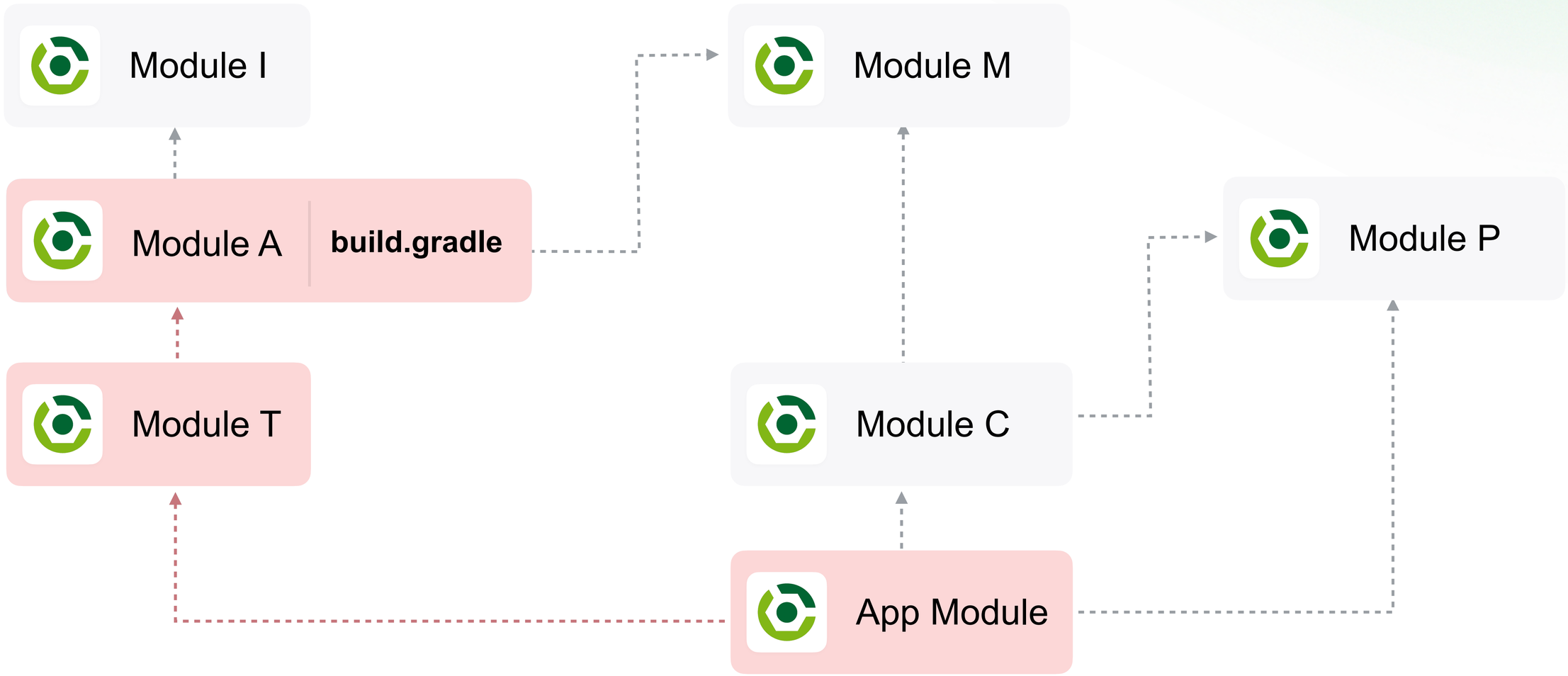


Запуск тестов с анализом зависимостей между классами

Полный запуск тестов в модуле

Анализ изменённых файлов

Остальные файлы в модуле

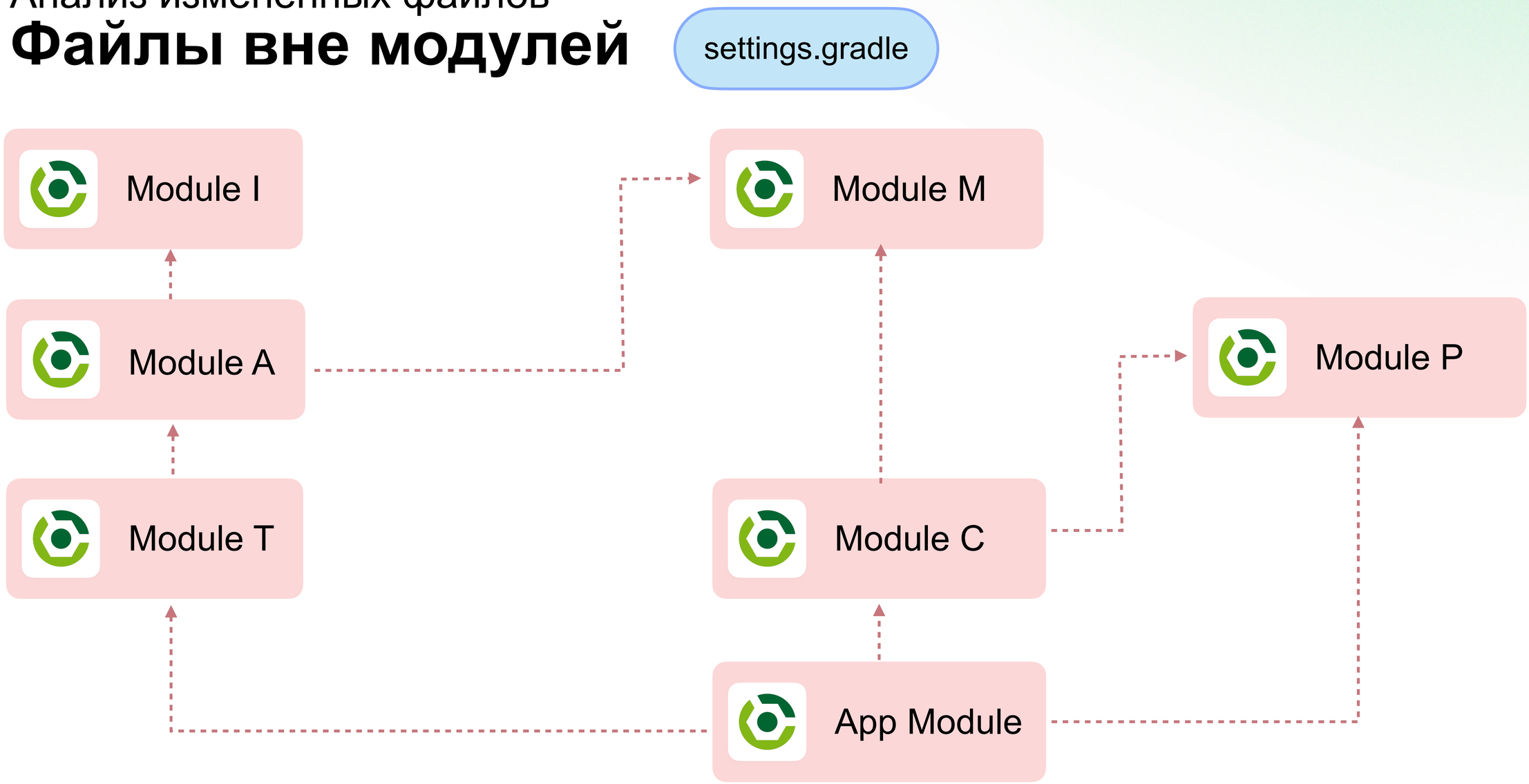


Запуск тестов с анализом зависимостей между классами

Полный запуск тестов в модуле

Анализ изменённых файлов

Файлы вне модулей



Анализ изменённых файлов

Анализ измененных файлов и запуск тестов

Модули с анализом зависимостей между классами

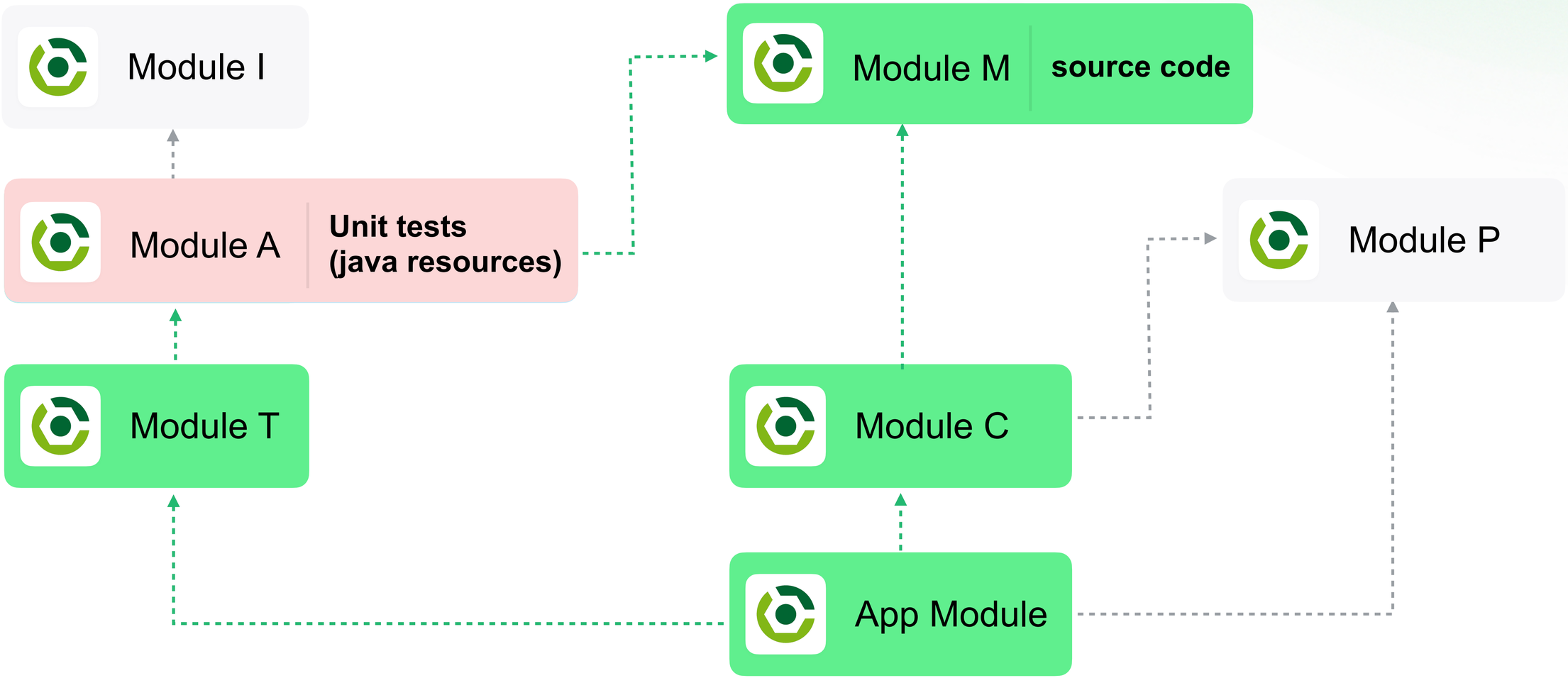
Модули с полным запуском тестов

Алгоритм работы

Конфигурация

Анализ изменённых файлов

Анализ измененных файлов и запуск тестов



Запуск тестов с анализом зависимостей между классами

Полный запуск тестов в модуле

Анализ изменённых файлов

Анализ измененных файлов и запуск тестов

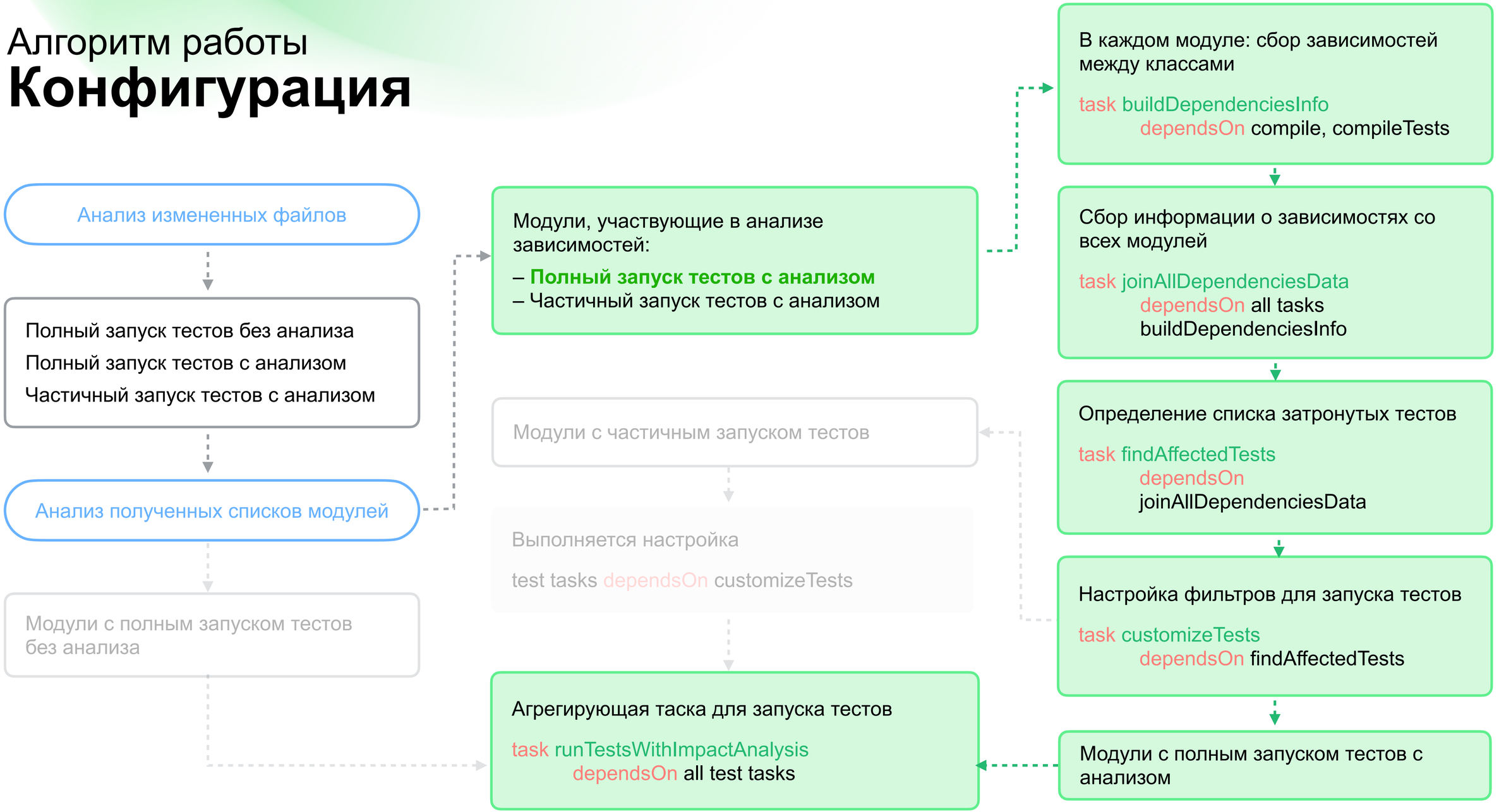
Модули с полным запуском тестов

Модули, в которых нужно включать анализ, но тесты запускать полностью

Модули с анализом зависимостей между классами

Алгоритм работы

Конфигурация




AAAAAA

Реализация

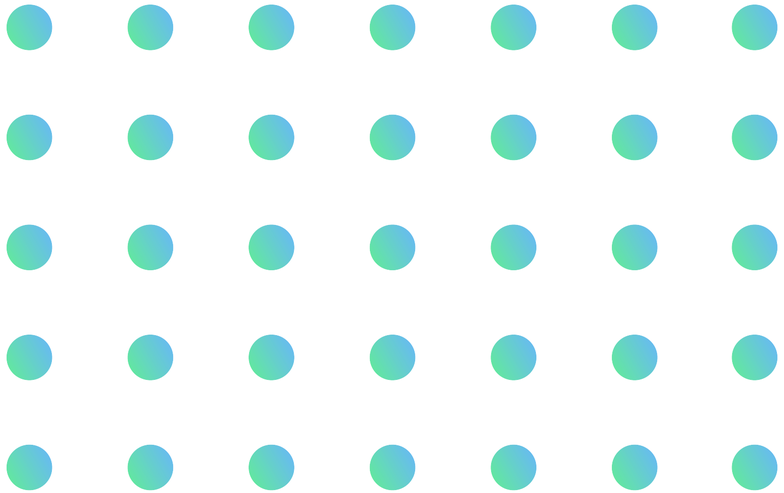
Анализ зависимостей между классами

Файлы

Тесты

Реализация

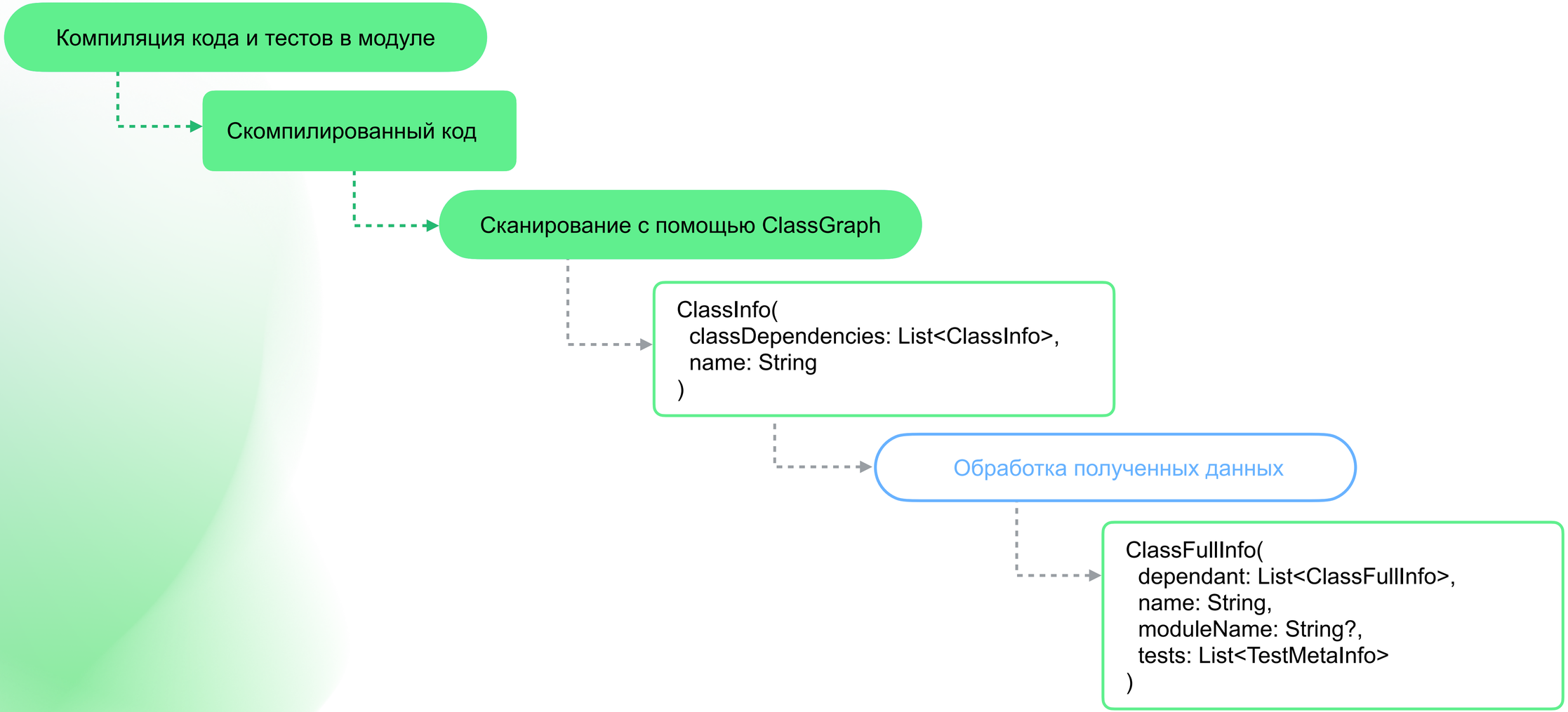
Анализ зависимостей между классами



Классы

Реализация

Анализ зависимостей между классами



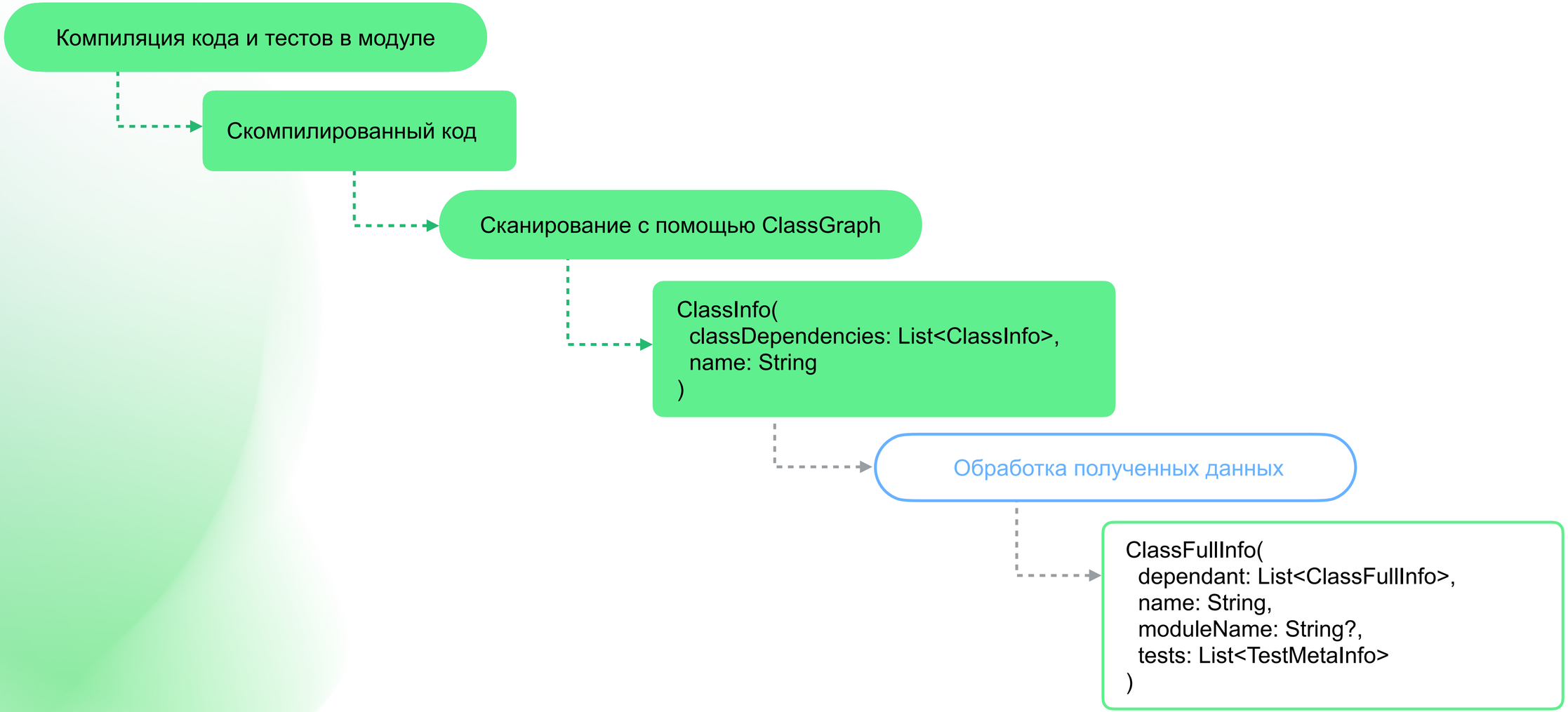
Реализация

ClassGraph

```
ClassGraph()
    .overrideClasspath(scanDirs/*скомпилированный код*/)
    .enableInterClassDependencies()
    .enableAllInfo()
    .enableExternalClasses()
    .scan()
    .allClasses
```

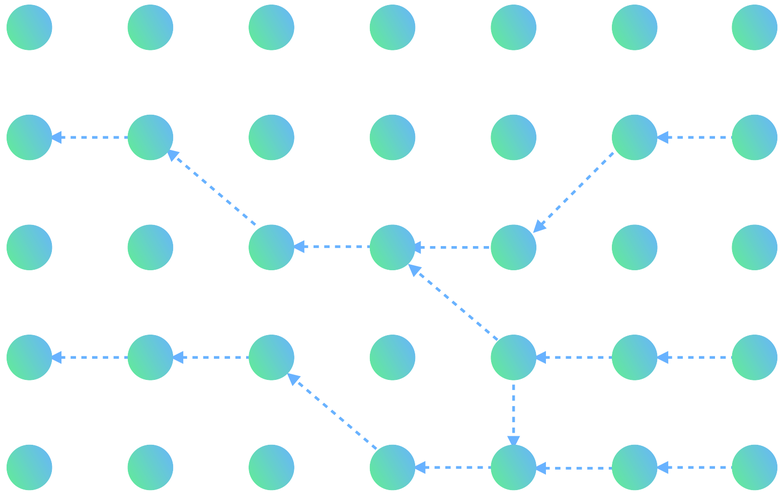
Реализация

Анализ зависимостей между классами



Реализация

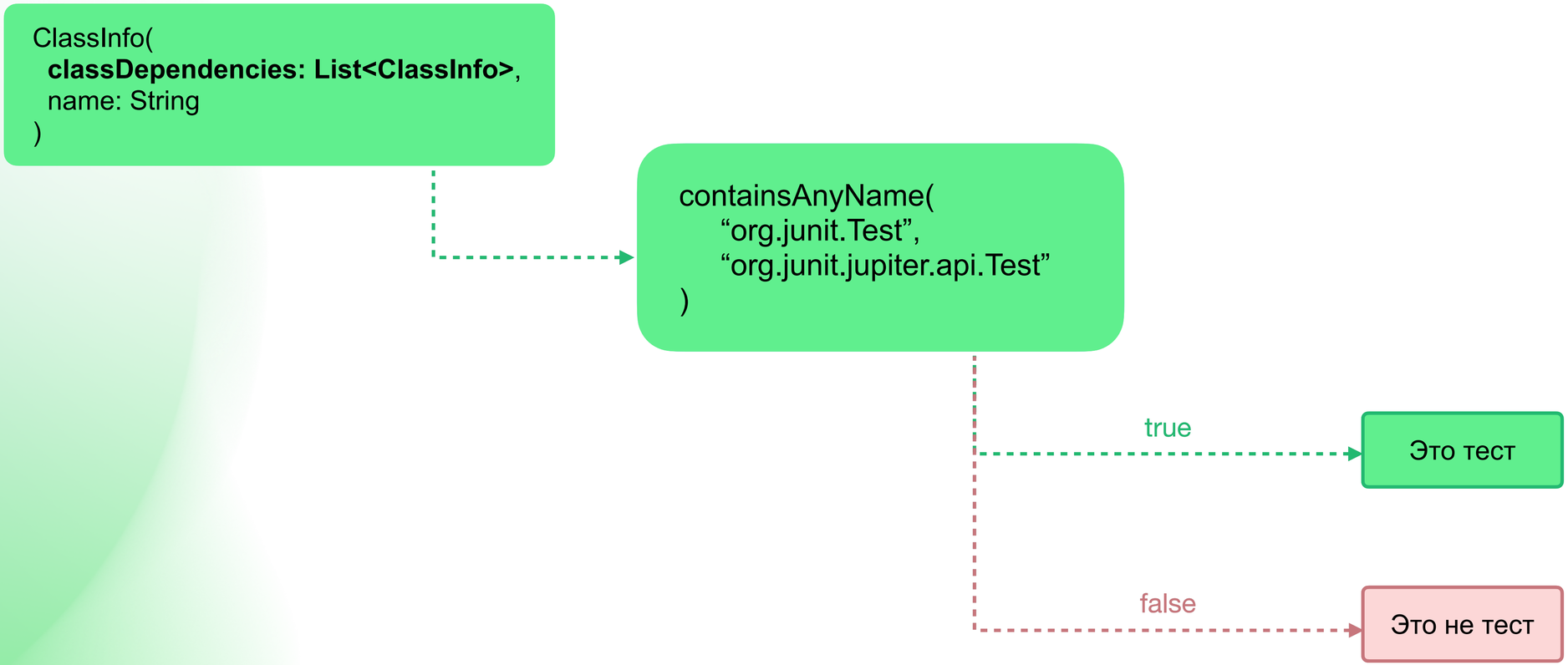
Анализ зависимостей между классами



Класс

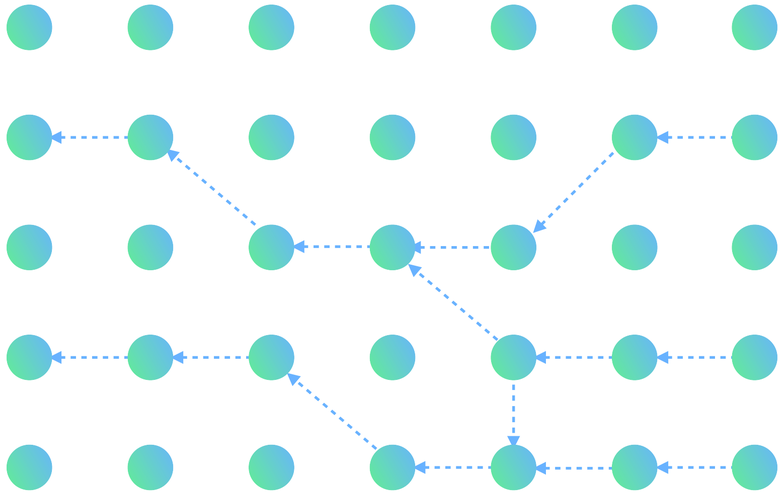
Реализация

Определение тестового класса



Реализация

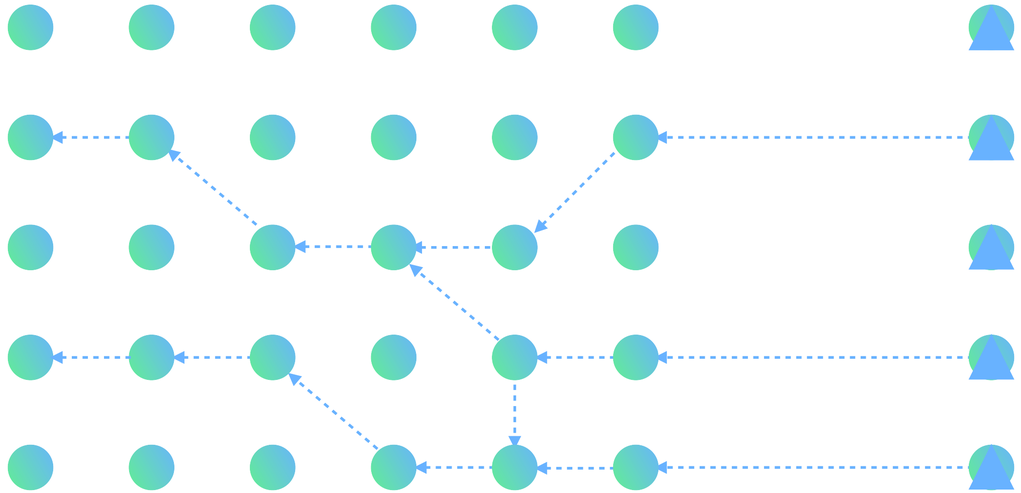
Анализ зависимостей между классами



Класс

Реализация

Анализ зависимостей между классами

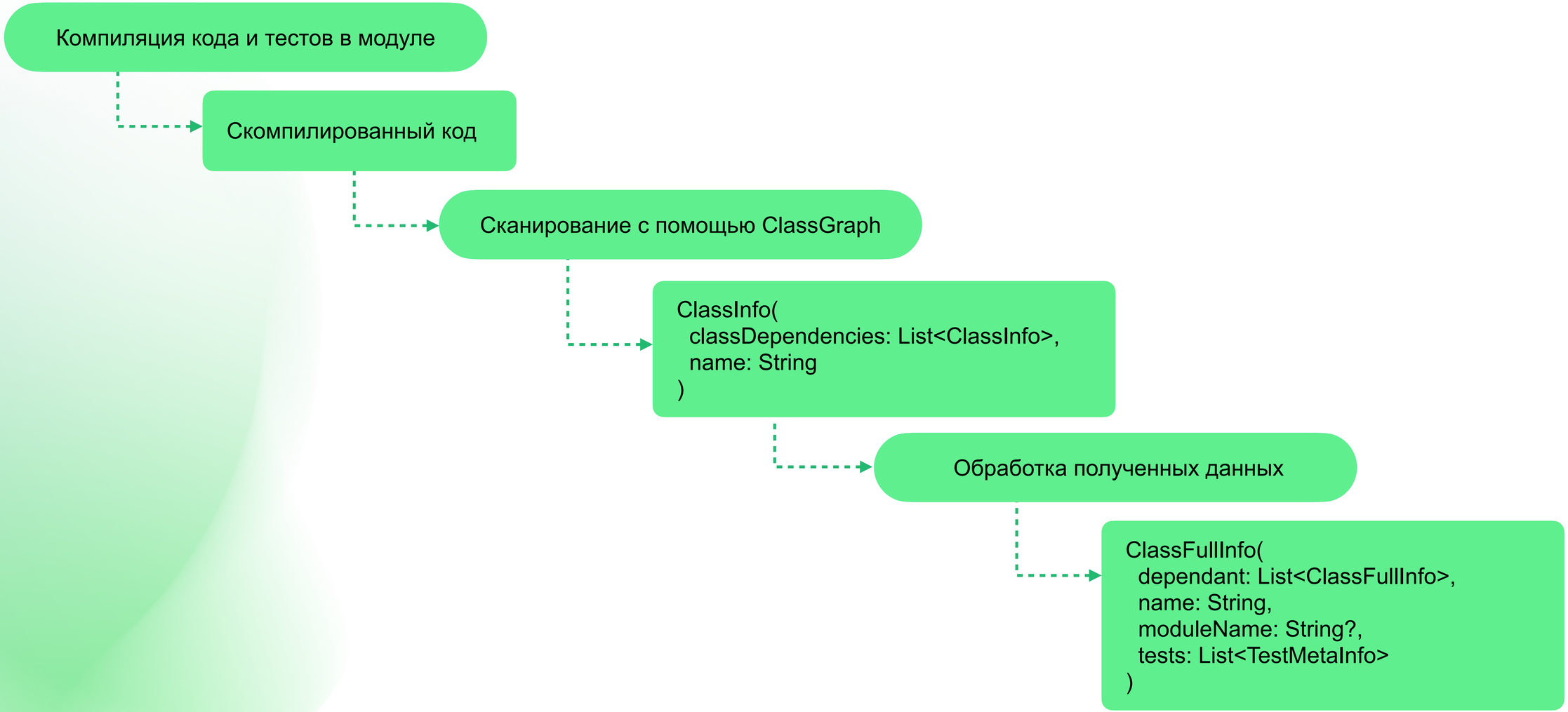


Тест

Класс

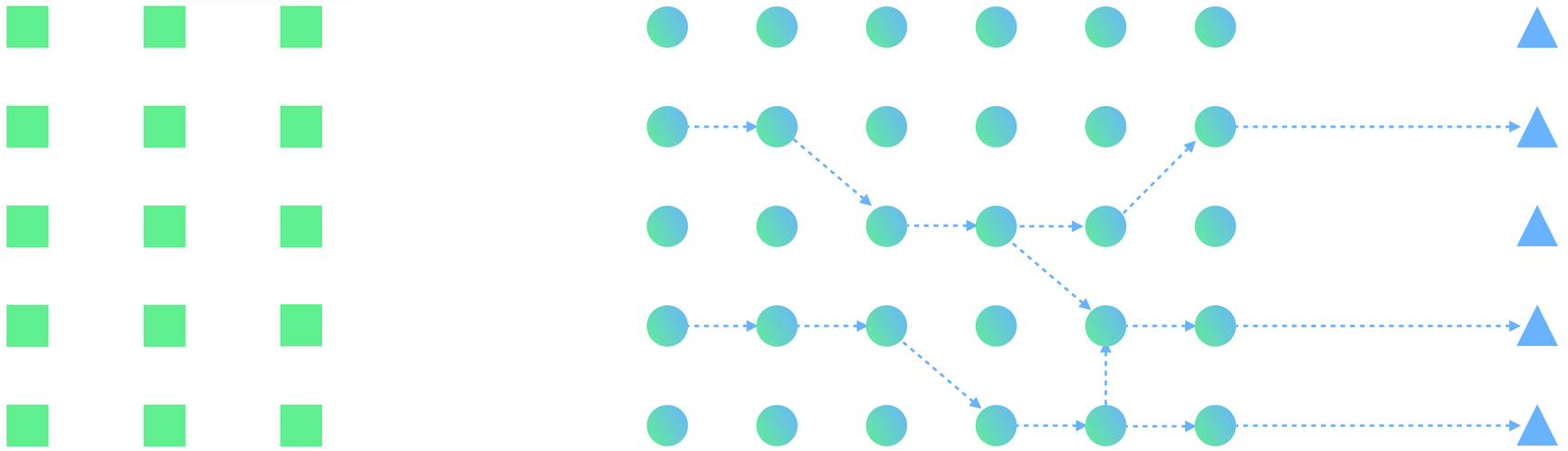
Реализация

Анализ зависимостей между классами



Реализация

Анализ зависимостей между классами



Файл Тест Класс

Реализация

Получение списка классов из списка файлов

Kotlin-файлы

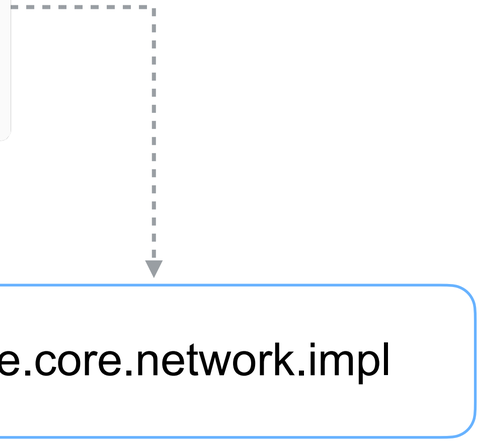
Java-файлы

Android-ресурсы

Реализация

Получение классов из Android-ресурсов

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest package="sample.core.network.impl"/>
```

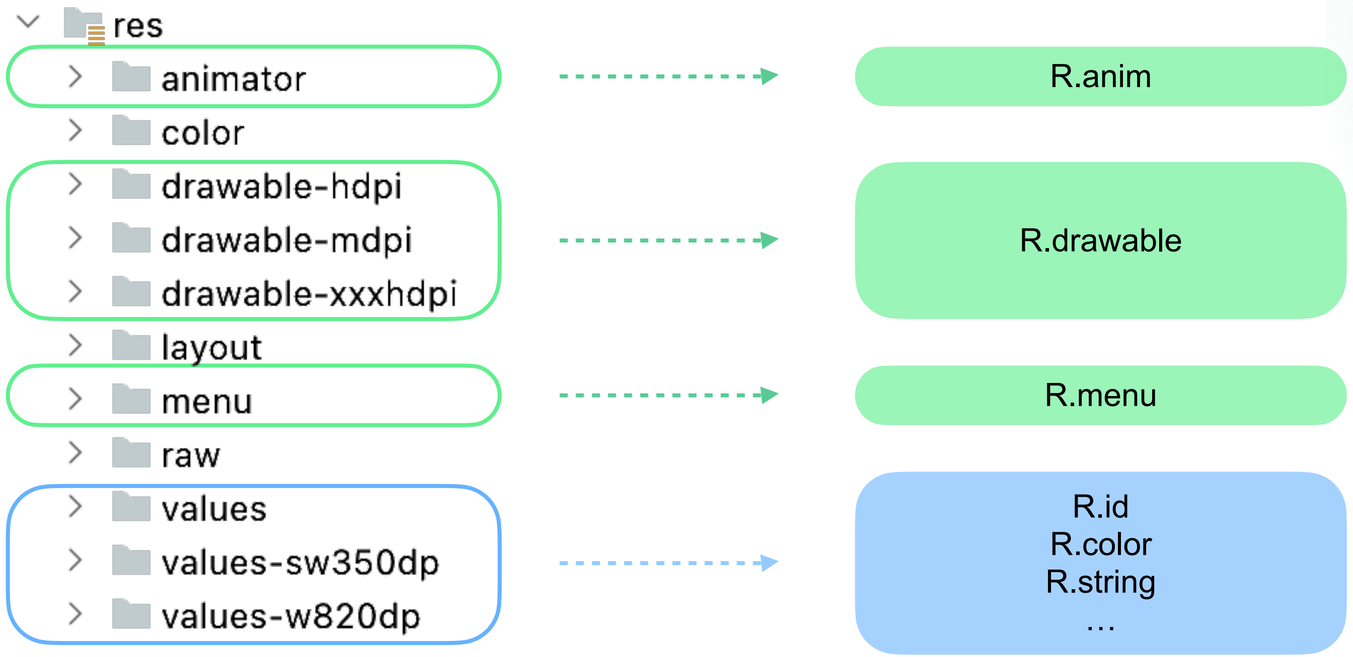


Реализация

Получение классов из Android-ресурсов

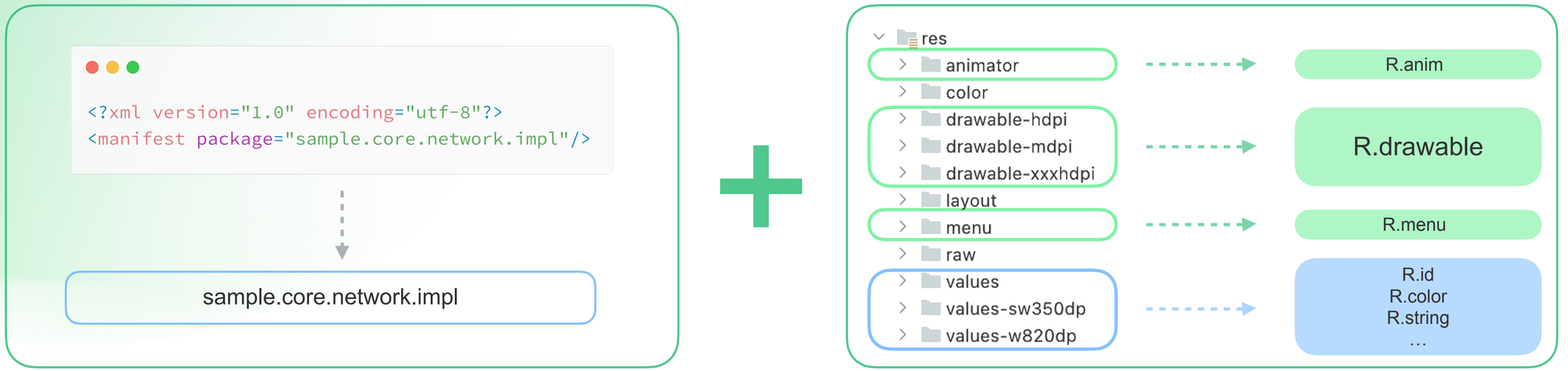
```
<?xml version="1.0" encoding="utf-8"?>
<manifest package="sample.core.network.impl"/>
```

sample.core.network.impl



Реализация

Получение классов из Android-ресурсов



sample.core.network.impl.R.anim

sample.core.network.impl.R.drawable

sample.core.network.impl.R.menu

sample.core.network.impl.R.id

sample.core.network.impl.R.color

sample.core.network.impl.R.string

Реализация

Получение классов из Kotlin-файлов

SampleFile.kt

```
package sample.app

interface PublicInterface {}
class PublicClass {}
private class PrivateClass {}
```

sample.app.PublicInterface

sample.app.PublicClass

ZeroString.kt

```
package sample.app
@file:JvmName("ZeroClass")

fun getZeroString() = "0"
```

sample.app.ZeroClass

Zero.kt

```
package sample.app

fun getZero() = 0
```

sample.app.ZeroKt

Реализация

Получение классов из Java-файлов

TestClass.java

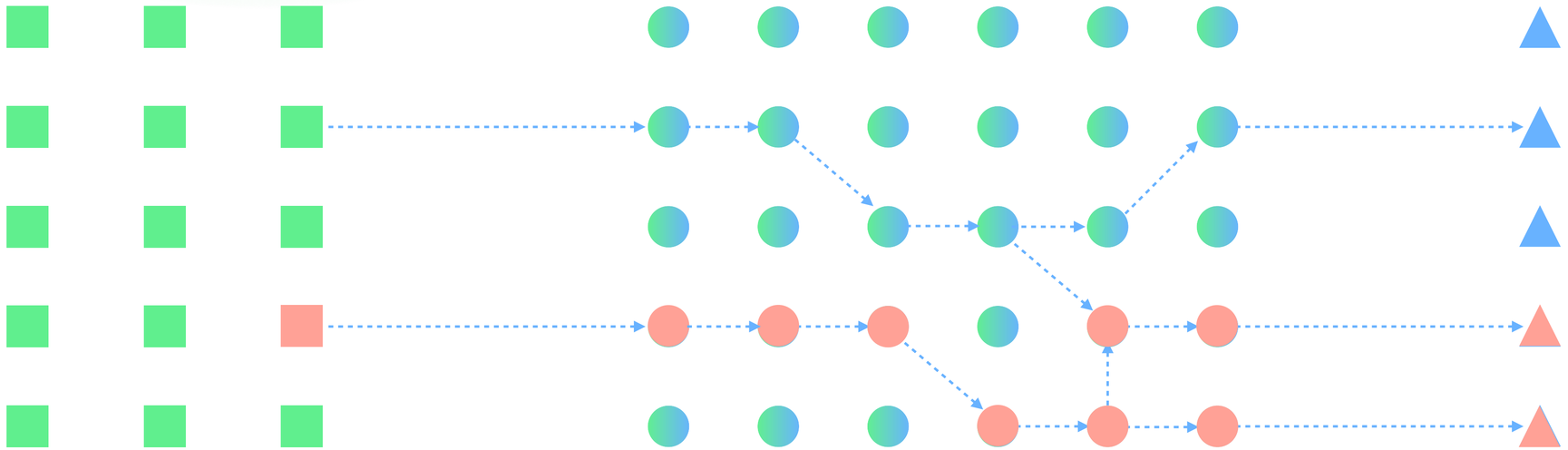
```java
package sample.app;

public class TestClass {
    class NestedClass {}
}
```

sample.app.TestClass

Реализация

Анализ зависимостей между классами



Файл Тест Класс

Реализация

Настройка фильтров для тестов

```
public abstract class AbstractTestTask {
    private final DefaultTestFilter filter;

    ...
}
```

Реализация

Настройка фильтров для тестов

```
// если список тестов для модуля не пустой
testTask.filter.setIncludePatterns(*currentTests.toTypedArray())
```

Реализация

Настройка фильтров для тестов

```
// если тестов для запуска нет
testTask.enabled = false
prepareTestTasks.forEach { prepareTestTask ->
    prepareTestTask.enabled = false
}
```

Ho … **Kotlin**

Встреченные сложности
Kotlin

```kotlin
package ru.sber.utils

const val CONST_VAL = "This is const val"
```

```kotlin
package ru.sber
import ru.sber.utils.CONST_VAL

class SampleClass {
    fun getConstVal() = CONST_VAL
}
```

Декомпиляция в Java

```java
package ru.sber.utils;
import org.jetbrains.annotations.NotNull;

public final class SampleConstKt {
  @NotNull
  public static final String CONST_VAL = "This is const val";
}
```

```java
package ru.sber;
import org.jetbrains.annotations.NotNull;

public final class SampleClass {
  @NotNull
  public final String getConstVal() {
     return "This is const val";
  }
}
```

Встреченные сложности

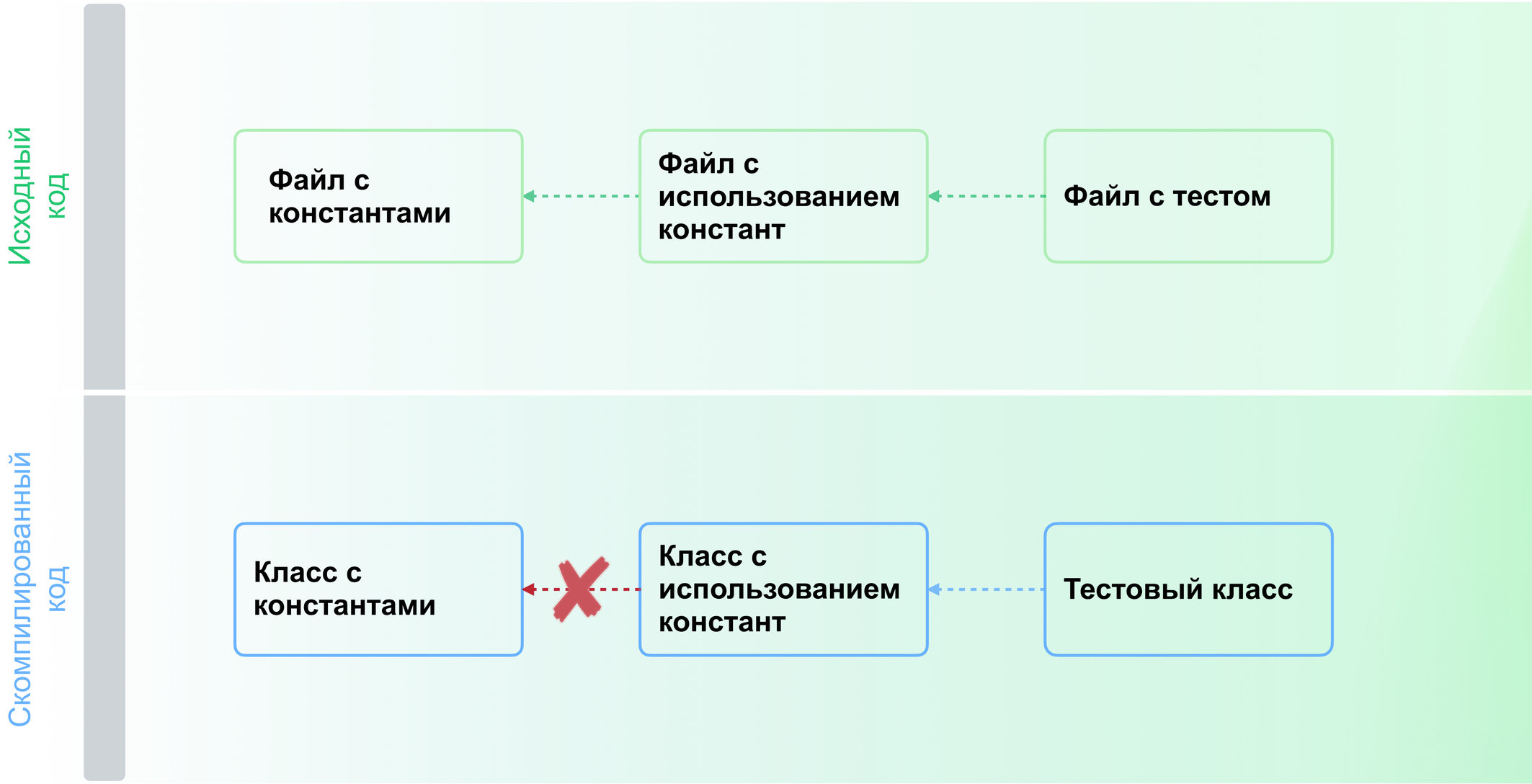
Kotlin

```java
package ru.sber;
import org.jetbrains.annotations.NotNull;

public final class SampleClass {
   @NotNull
   public final String getConstVal() {
      return "This is const val";
   }
}
```

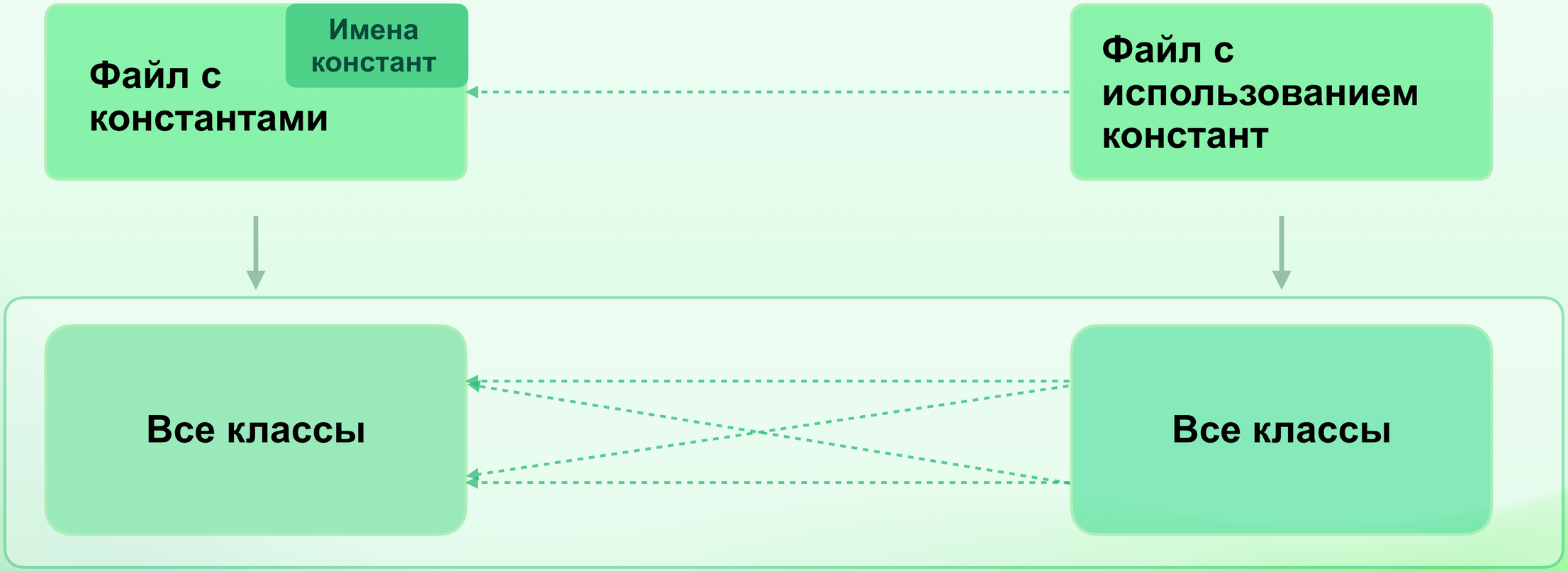
Встреченные сложности

Kotlin



Встреченные сложности

Kotlin

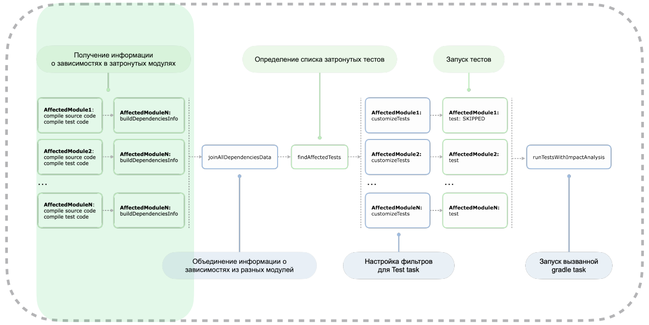


Статический анализ

Алгоритм работы

Алгоритм работы

Получение информации о зависимостях в модулях



AffectedModule1:
compile code

AffectedModule1: buildDependenciesInfo

AffectedModule1: buildInlineDependenciesInfo

AffectedModule2:
compile code

AffectedModule2: buildDependenciesInfo

…

…

AffectedModuleN:
compile code

AffectedModuleN: buildDependenciesInfo

AffectedModuleN: buildInlineDependenciesInfo

jo

Impact Analysis Gradle Plugin

Подключение и настройка

Составные части

Impact Analysis Gradle Plugin

Root-плагин

Используется для общих настроек на весь проект

App-плагин

Используется для настроек конкретного Application-а

Подключение и настройка

Root-плагин

В корневом `build.gradle`

```
impactAnalysis {
    enable = true
    diffFile = "scripts/testing/diff"
    configFile = "scripts/testing/impact_analysis_config.yaml"
}
```

Подключение и настройка

Root-плагин

enable

Включение/отключение плагина

diffFile

Список измененных файлов

configFile

acceptedPackages

Фильтр внешних зависимостей

exclusions

Фильтр файлов, не влияющих на тесты

Подключение и настройка

App-плагин

Для модулей из дерева зависимостей соответствующего Application-а

Можно подключать Impact Analysis не ко всем Application-ам в проекте

Отдельная настройка разных Application-ов

Общие модули

- переиспользование тасок при совпадении BuildVariant-ов
- при несовпадении BuildVariant-ов тесты запускаются отдельно на каждом из них

Независимые модули

- Тесты не запускаются

Подключение и настройка

App-плагин

build.gradle в Application-модуле

```
impactAnalysis {
    fullBuildVariantName = "fullDebug"
}
```

Подключение и настройка

App-плагин

fullBuildVariantName = «fullDebug»

Java Module

- test

Android Module

- testDebugUnitTest
- testReleaseUnitTest

Android Module

- testFullDebugUnitTest
- testDemoDebugUnitTest
- testFullReleaseUnitTest
- testDemoReleaseUnitTest

Разделяем BuildVariant на составные части

Проверяем каждую часть на принадлежность *fullBuildVariantName*

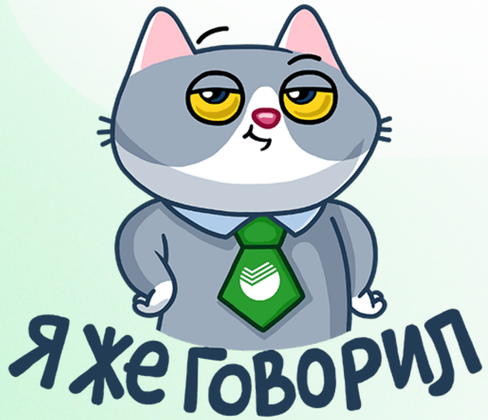
Выбираем тот BuildVariant, в котором все части содержатся в *fullBuildVariantName*

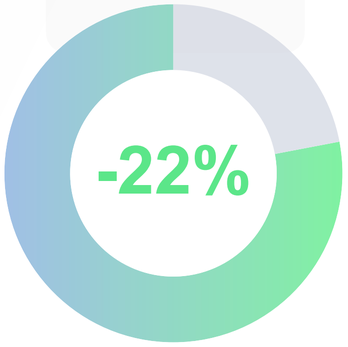
Результаты

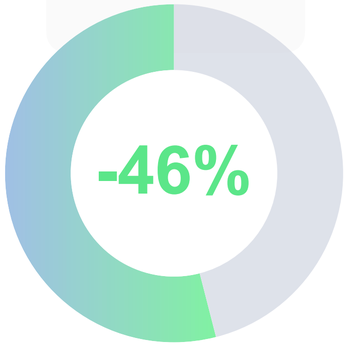
Запуск тестов в Сбербанк Онлайн (Android)

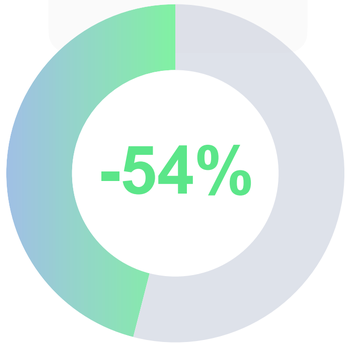
Время

	50%%	75%%	90%%	95%%
Анализ зависимостей между **модулями**	**10.33** мин	**16.33** мин	**47.25** мин	**90.18** мин
Анализ зависимостей между **классами**	**8.76** мин	**12.63** мин	**25.24** мин	**41.05** мин









Полезные ссылки

Полезные ссылки

- Про Test Impact Amalysis от Paul Hammant
- ClassGraph для анализа зависимостей


СПАСИБО

Вопросы