

CI/CD Setup

Um unsere Unittests zu verbessern haben wir nun 3 Pipelines.

Die Tests sind unterteilt in:

Android.yml

Datei konfiguriert eine GitHub Actions CI/CD-Pipeline, die bei Pushes und Pull Requests auf den master-Branch eine Android-App mit JDK 17 auf Ubuntu baut und testet

```
Qiriba Update android.yml  e445b96 - last month  History
Code Blame 26 lines (21 loc) · 476 Bytes  Code 55% faster with GitHub Copilot  Raw  Copy Download Edit View
1 name: Android CI
2
3 on:
4   push:
5     branches: [ "master" ]
6   pull_request:
7     branches: [ "master" ]
8
9 jobs:
10  build:
11
12    runs-on: ubuntu-latest
13
14    steps:
15      - uses: actions/checkout@v4
16      - name: set up JDK 17
17        uses: actions/setup-java@v3
18        with:
19          java-version: '17'
20          distribution: 'temurin'
21          cache: gradle
22
23      - name: Grant execute permission for gradlew
24        run: chmod +x gradlew
25      - name: Build with Gradle
26        run: ./gradlew build
```

Build.yml

Datei konfiguriert eine GitHub Actions CI/CD-Pipeline, die manuell ausgelöst wird, um eine Android-App mit JDK 17 zu bauen, SonarQube-Analysen durchzuführen und Pakete zu cachen.

```
Code Blame 35 lines (31 loc) · 999 Bytes  Code 55% faster with GitHub Copilot  Raw  Copy Download Edit View
1 name: Build
2
3 on: workflow_dispatch
4
5
6 jobs:
7  build:
8    name: Build and analyze
9    runs-on: ubuntu-latest
10
11    steps:
12      - uses: actions/checkout@v4
13        with:
14          fetch-depth: 0 # shallow clones should be disabled for a better relevancy of analysis
15      - name: Set up JDK 17
16        uses: actions/setup-java@v1
17        with:
18          java-version: 17
19      - name: Cache SonarQube packages
20        uses: actions/cache@v1
21        with:
22          path: ~/.sonar/cache
23          key: ${runner.os}-sonar
24          restore-keys: |
25            ${runner.os}-sonar
26      - name: Cache gradle packages
27        uses: actions/cache@v1
28        with:
29          path: ~/.gradle/caches
30          key: ${runner.os}-gradle-${hashFiles('**/*.gradle')}
31          restore-keys: |
32            ${runner.os}-gradle
33      - name: Build and analyze
34        env:
35          SONAR_TOKEN: ${secrets.SONAR_TOKEN}
36          SONAR_HOST_URL: ${secrets.SONAR_HOST_URL}
37        run: ./gradlew build sonar --info
```

build	
succeeded 3 weeks ago in 2m 31s	
> Set up job	1s
> Run actions/checkout@v4	1s
> set up JDK 17	8s
> Grant execute permission for gradlew	8s
> Run unit tests	1m 41s
> Post set up JDK 17	47s
> Post Run actions/checkout@v4	8s
> Complete job	8s

Unittests.yml

Datei konfiguriert eine GitHub Actions CI/CD-Pipeline, die bei Pushes und Pull Requests auf den master-Branch Unit-Tests für eine Android-App mit JDK 17 auf Ubuntu ausführt.

```

1 name: unit tests
2
3 on:
4   push:
5     branches: [ "master" ]
6   pull_request:
7     branches: [ "master" ]
8
9 jobs:
10  build:
11
12    runs-on: ubuntu-latest
13
14    steps:
15      - uses: actions/checkout@v4
16      - name: set up JDK 17
17        uses: actions/setup-java@v3
18        with:
19          java-version: '17'
20          distribution: 'temurin'
21          cache: gradle
22
23      - name: Grant execute permission for gradlew
24        run: chmod +x gradlew
25
26      - name: Run unit tests
27        run: ./gradlew test

```

Wir haben uns außerdem dafür entschieden, weil CI/CD die Zusammenarbeit innerhalb des Entwicklungsteams fördert. Dadurch wird die Kommunikation verbessert und das Risiko von Konflikten minimiert.

README

Only for Android App

Contains all code :)

Android CI passing

Unit tests passing

Insgesamt bieten CI/CD Pipelines eine effiziente und zuverlässige Möglichkeit, Software zu entwickeln, zu testen und bereitzustellen, was zu einer verbesserten Produktqualität und einer erhöhten Entwicklerproduktivität führt.