IntegrationTests.jl: a framework for the automatic generation of integration tests for Julia projects

Simeon Ehrig

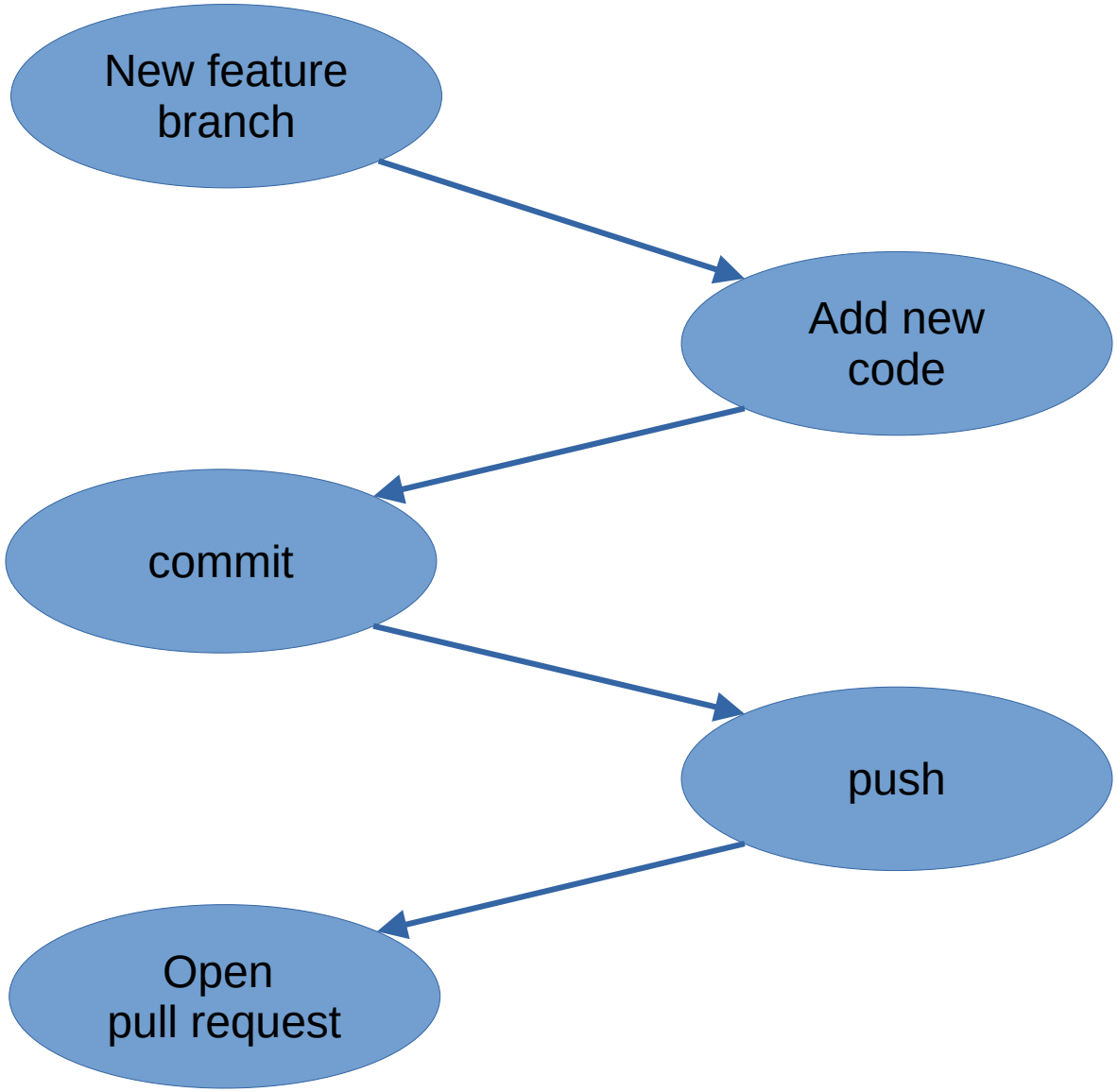
CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

www.casus.science

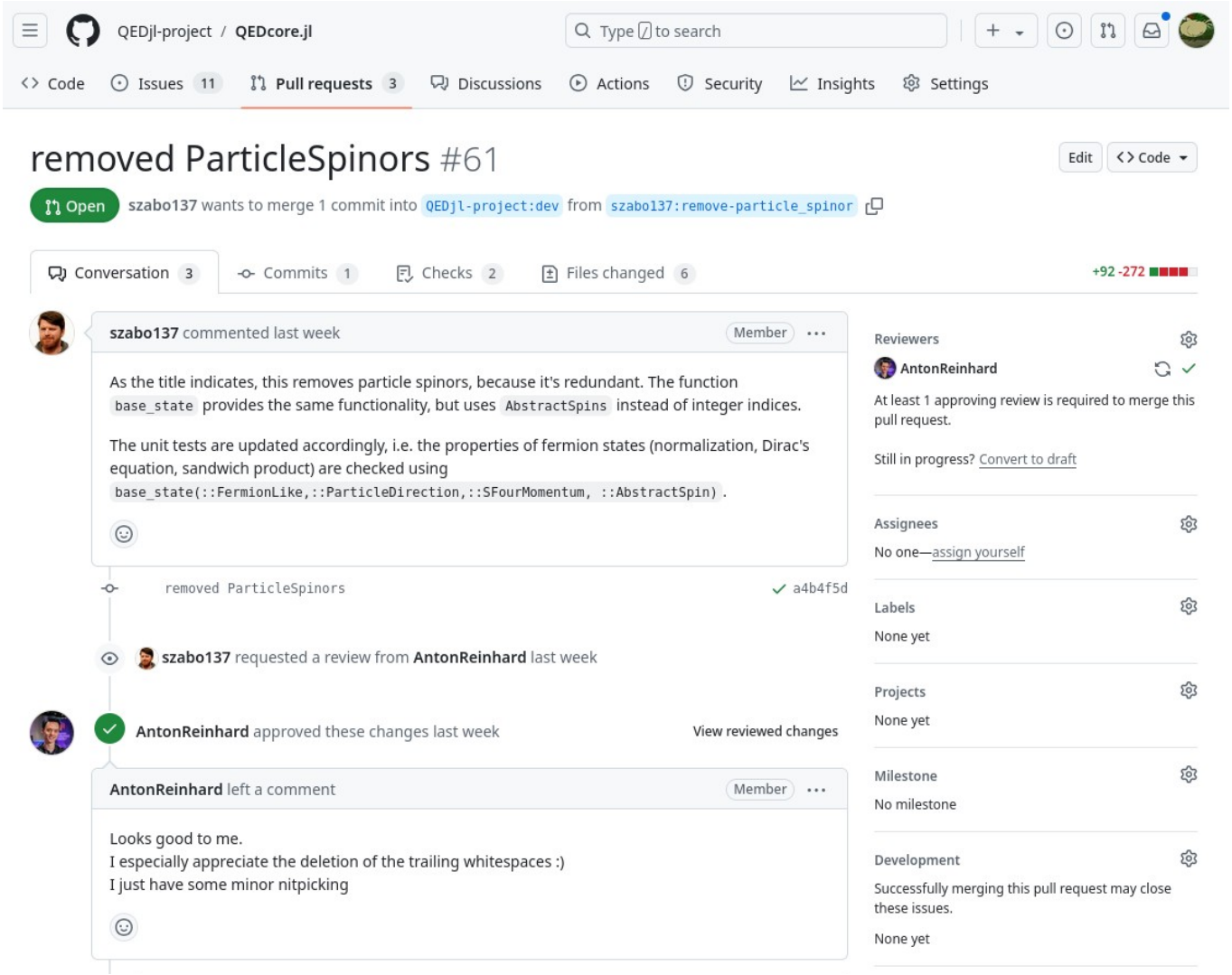
HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
UFZ HELMHOLTZ Centre for Environmental Research
Uniwersytet Wrocławski

SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS
Freistaat SACHSEN

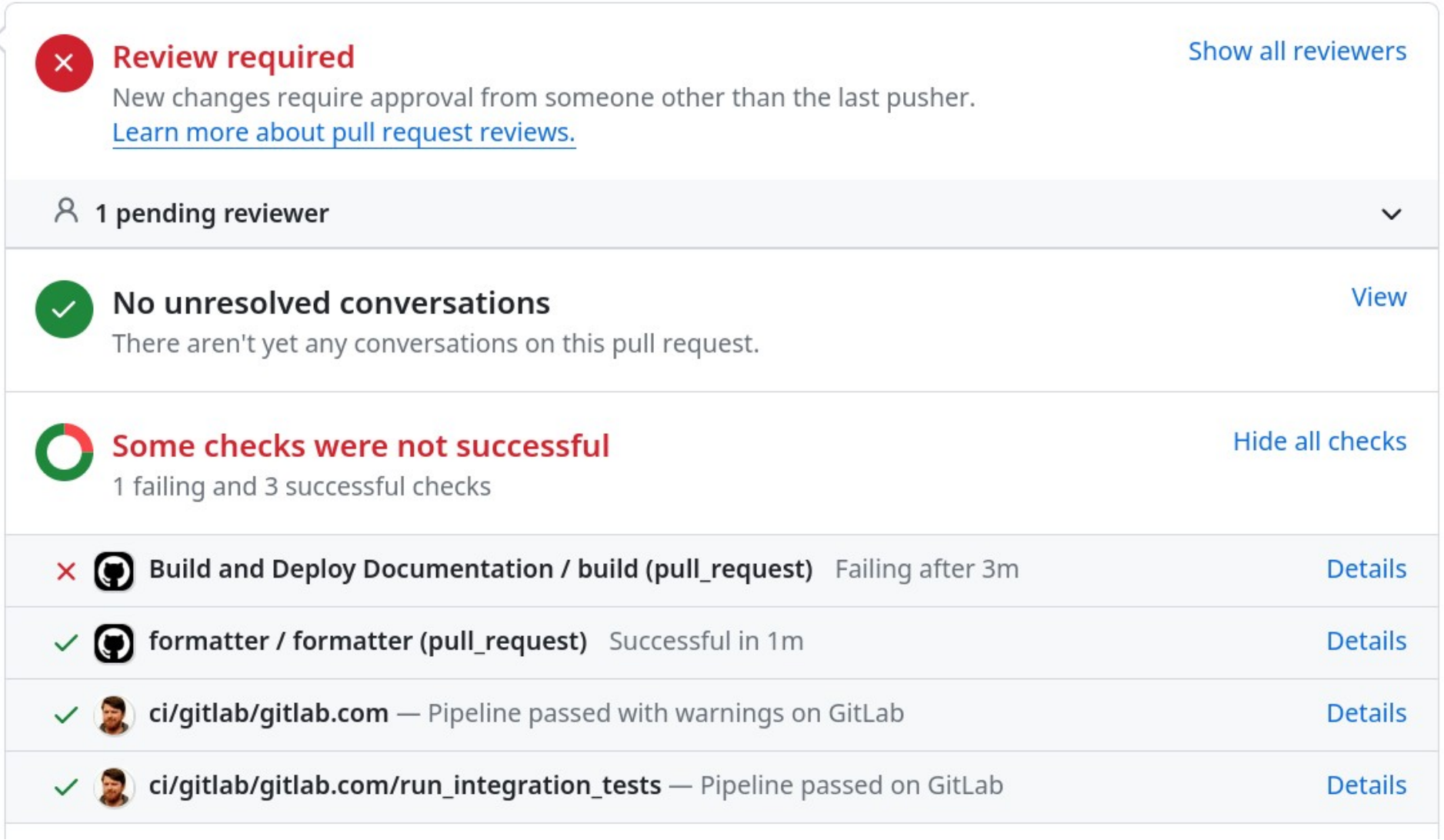
Git Workflow: new Code



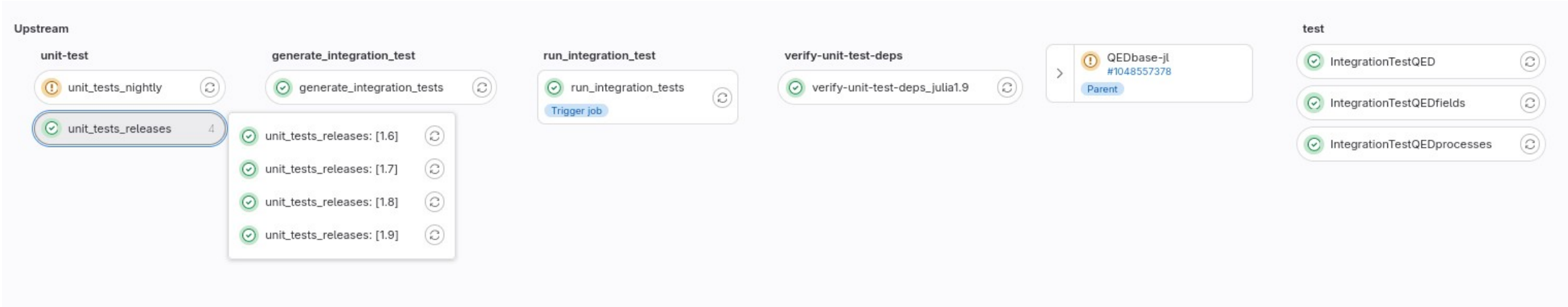
Open Pull Request



Pull Request triggers CI Tests



CI Pipeline


Upstream
unit-test
unit_tests_nightly
unit_tests_releases 4
generate_integration_test
generate_integration_tests
unit_tests_releases: [1.6]
unit_tests_releases: [1.7]
unit_tests_releases: [1.8]
unit_tests_releases: [1.9]
run_integration_test
run_integration_tests
Trigger job
verify-unit-test-deps
verify-unit-test-deps_julia1.9
QEDbase-jl
#1048557378
Parent
test
IntegrationTestQED
IntegrationTestQEDfields
IntegrationTestQEDprocesses

Defining Tests and execution

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Define the tests

```
using Test
using QEDcore

TESTPSDEF = TestImplementation.TestPhasespaceDef()

RNG = Random.MersenneTwister(727)
BUF = IOBuffer()

Anton Reinhard, 3 months ago | 1 author (Anton Reinhard)
@testset "broadcast" begin
    test_func(ps_def) = ps_def
    @test test_func.(TESTPSDEF) == TESTPSDEF
end
```

Defining Tests and execution

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Define the tests

```
using Test
using QEDcore

TESTPSDEF = TestImplementation.TestPhasespaceDef()

RNG = Random.MersenneTwister(727)
BUF = IOBuffer()

Anton Reinhard, 3 months ago | 1 author (Anton Reinhard)
@testset "broadcast" begin
    test_func(ps_def) = ps_def
    @test test_func.(TESTPSDEF) == TESTPSDEF
end
```

Define the execution of the tests

```
.untit_test_template:
  stage: unit-test
  script:
    - julia --project=. -e 'import Pkg; Pkg.instantiate()'
    - julia --project=. -e 'import Pkg; Pkg.test(; coverage = true)'
  interruptible: true
  tags:
    - cpuonly

unit_tests_releases:
  extends: .untit_test_template
  parallel:
    matrix:
      - JULIA_VERSION: ["1.6", "1.7", "1.8", "1.9", "1.10", "rc"]
  image: julia:$JULIA_VERSION
```

Unit and Integration Tests

Unit vs. Integration Test

Unit vs. Integration Test: explanation

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Unit Test

- Does the function work in the context of your own package?

Integration Test

- Does the function work when used in other projects or packages?

Example Unit Test PkgA

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

PkgA/src.jl

```
module PkgA

foo(i) = i + 3

end
```

PkgA/test/runtest.jl

```
using PkgA
using Test

@testset "PkgA.jl" begin
    @test PkgA.foo(4) == 7
end
```

Example Integration Test

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

PkgA/src.jl

```
module PkgA

foo(i) = i + 3

end
```

PkgB/src.jl

```
module PkgB
using PkgA

bar() = PkgA.foo(3)

end
```

Integration Test in PkgB

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

PkgB/src.jl

```
module PkgB
using PkgA

bar() = PkgA.foo(3)

end
```

PkgB/test/runtest.jl

```
using PkgB
using Test

@testset "PkgB.jl" begin
    @test PkgB.bar() == 6
end
```

Integration Test in PkgB

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

PkgB/src.jl

```
module PkgB
using PkgA

bar() = PkgA.foo(3)

end
```

PkgB/test/runtest.jl

```
using PkgB
using Test

@testset "PkgB.jl" begin
    @test PkgB.bar() == 6
end
```

- Integration test for PkgA
- Unit test for PkgB

PkgA Interface Change

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

PkgA/src.jl

```
module PkgA

foo(i, j) = i + j + 3

end
```

→ Change interface of PkgA

PkgA Interface Change

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

PkgA/src.jl

```
module PkgA

foo(i, j) = i + j + 3

end
```

→ Change interface of PkgA

```
module PkgB
using PkgA

bar() = PkgA.foo(3)

end
```

```
using PkgB
using Test

@testset "PkgB.jl" begin
    @test PkgB.bar() == 6
end
```

PkgA Interface Change

PkgA/src.jl

```
module PkgA

foo(i, j) = i + j + 3

end
```

→ Change interface of PkgA

```
module PkgB
using PkgA

b

end
```

ERROR: MethodError

```
using PkgB
using Test

@tes

end
```

Test failes

Integration Tests: Followings

Find incompatibilities soon as possible to avoid stacking problems

Find dependencies between packages automatically

Define strategies to solve incompatibilies

https://www.youtube.com/watch?v=D-IP55-RTdk

Integration Tests: Followings

Find incompatibilities soon as possible to avoid stacking problems

→ automatic tests in the CI in each Pull Request

Find dependencies between packages automatically

Define strategies to solve incompatibilies

https://www.youtube.com/watch?v=D-IP55-RTdk

Integration Tests: Followings

Find incompatibilities soon as possible to avoid stacking problems

→ automatic tests in the CI in each Pull Request

Find dependencies between packages automatically

→ IntegrationTests.jl

Define strategies to solve incompatibilies

https://www.youtube.com/watch?v=D-IP55-RTdk

Integration Tests: Followings

Find incompatibilities soon as possible to avoid stacking problems

→ automatic tests in the CI in each Pull Request

Find dependencies between packages automatically

→ IntegrationTests.jl

Define strategies to solve incompatibilies

→ JuliaHEP 2023 talk: Unit and Integration testing in modularized Julia package eco-systems

https://www.youtube.com/watch?v=D-IP55-RTdk

IntegrationTests.jl

General Steps

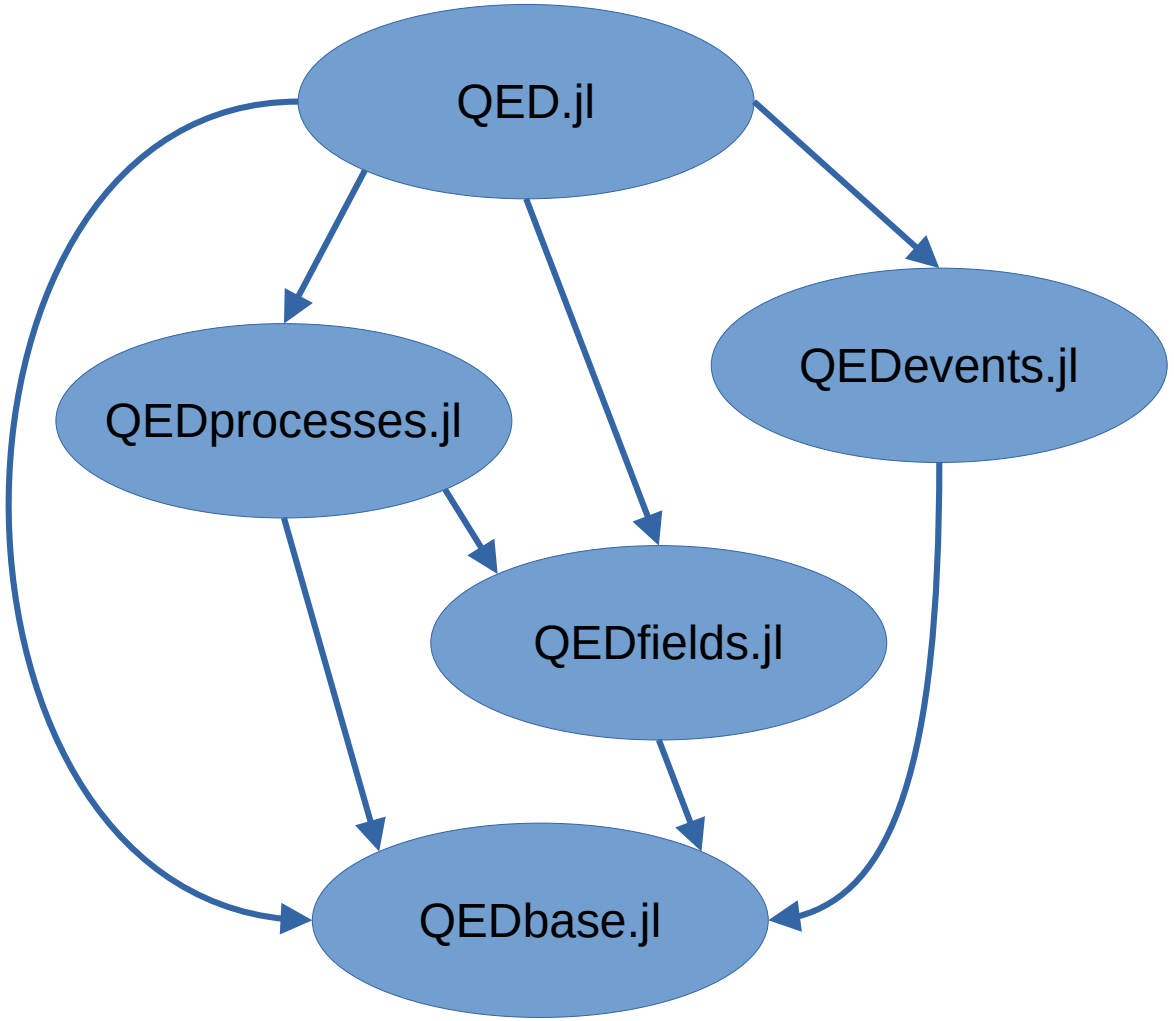
1) Generate project graph
2) Find packages who uses modified package
3) Generate CI jobs for integration tests (project depend)

Step 1: Construct Dependency Graph

Use the dependency graph of the Project.toml

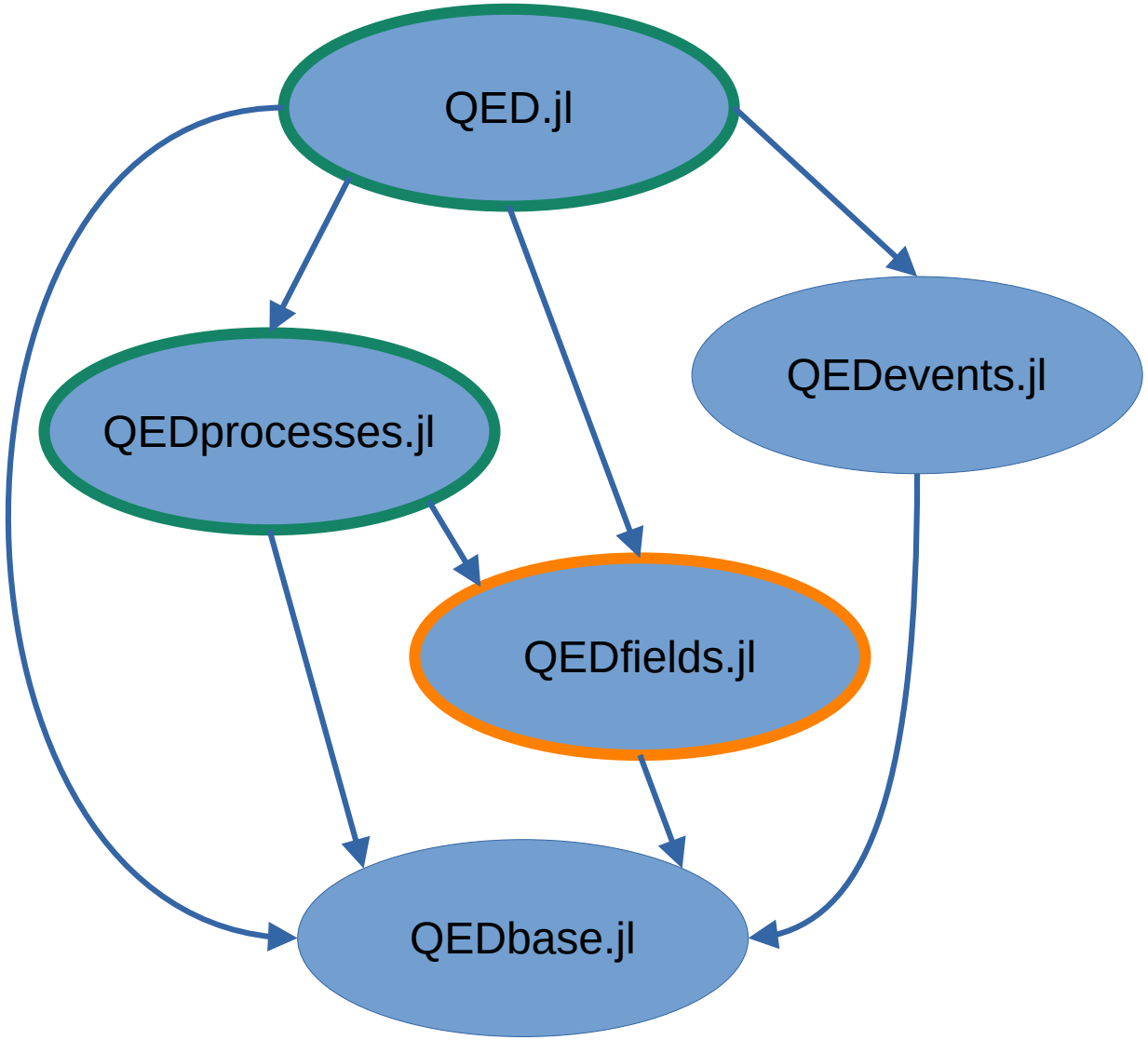
Step 1: Construct Dependency Graph

Use the dependency graph of the Project.toml



* graph is outdated

Step 2: Find using Packages



QEDfields.jl is modified

Dependent packages:

- QEDProcesses.jl
- QED.jl

Using IntegrationTests.jl

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

```
using IntegrationTests
using Pkg

if !isinteractive()
    # search packages, which are dependent on QEDfields.jl
    # ignore all packages which does not start with QED
    depending_packages = IntegrationTests.depending_projects(
                                "QEDfields.jl", r"^QED*")

    println(depending_packages)
end
```

Using IntegrationTests.jl

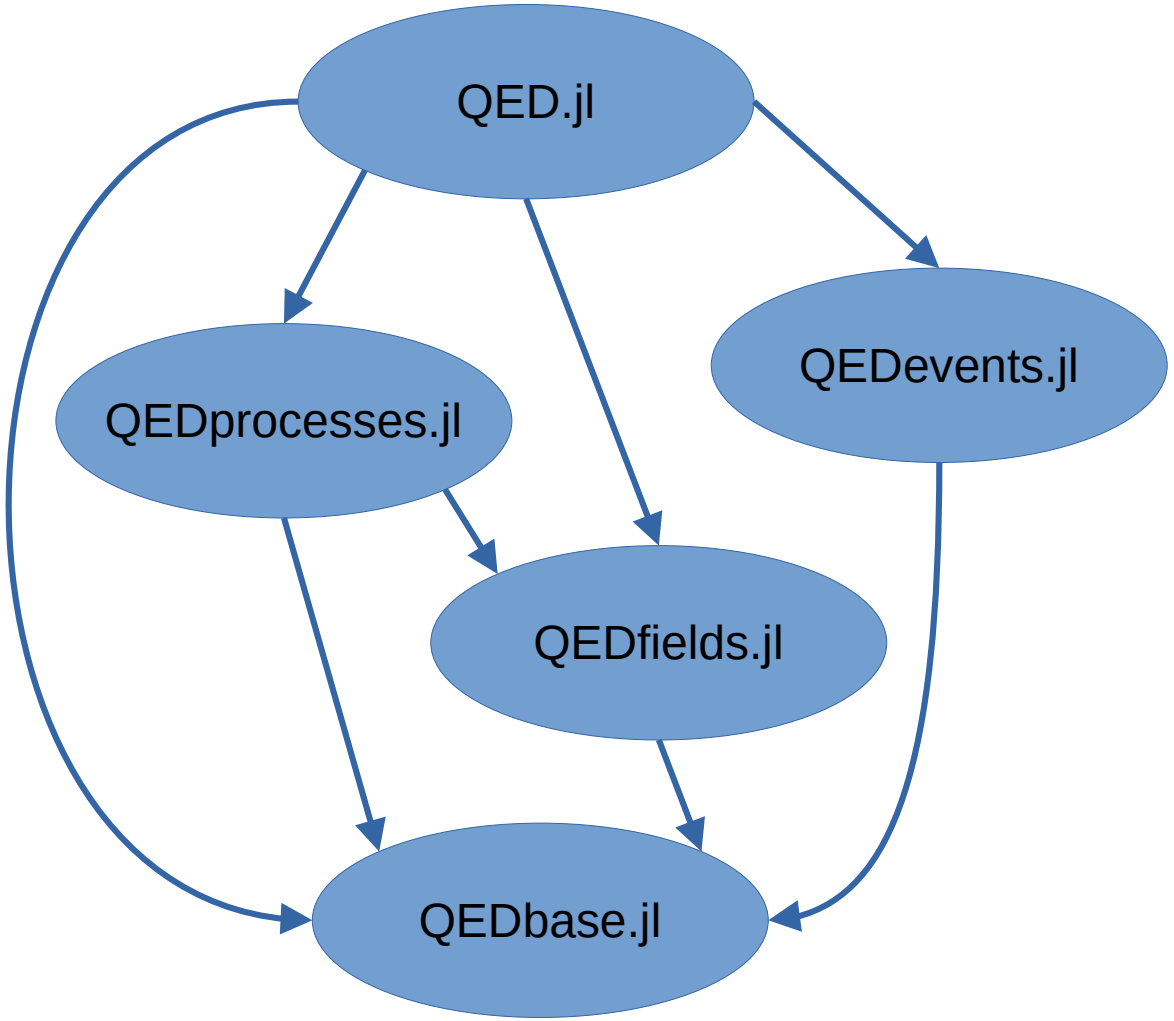
```julia
using IntegrationTests
using Pkg

if !isinteractive()
    # search packages, which are dependent on QEDfields.jl
    # ignore all packages which does not start with QED
    depending_packages = IntegrationTests.depending_projects(
                                "QEDfields.jl", r"^QED*")

    println(depending_packages)
end
```

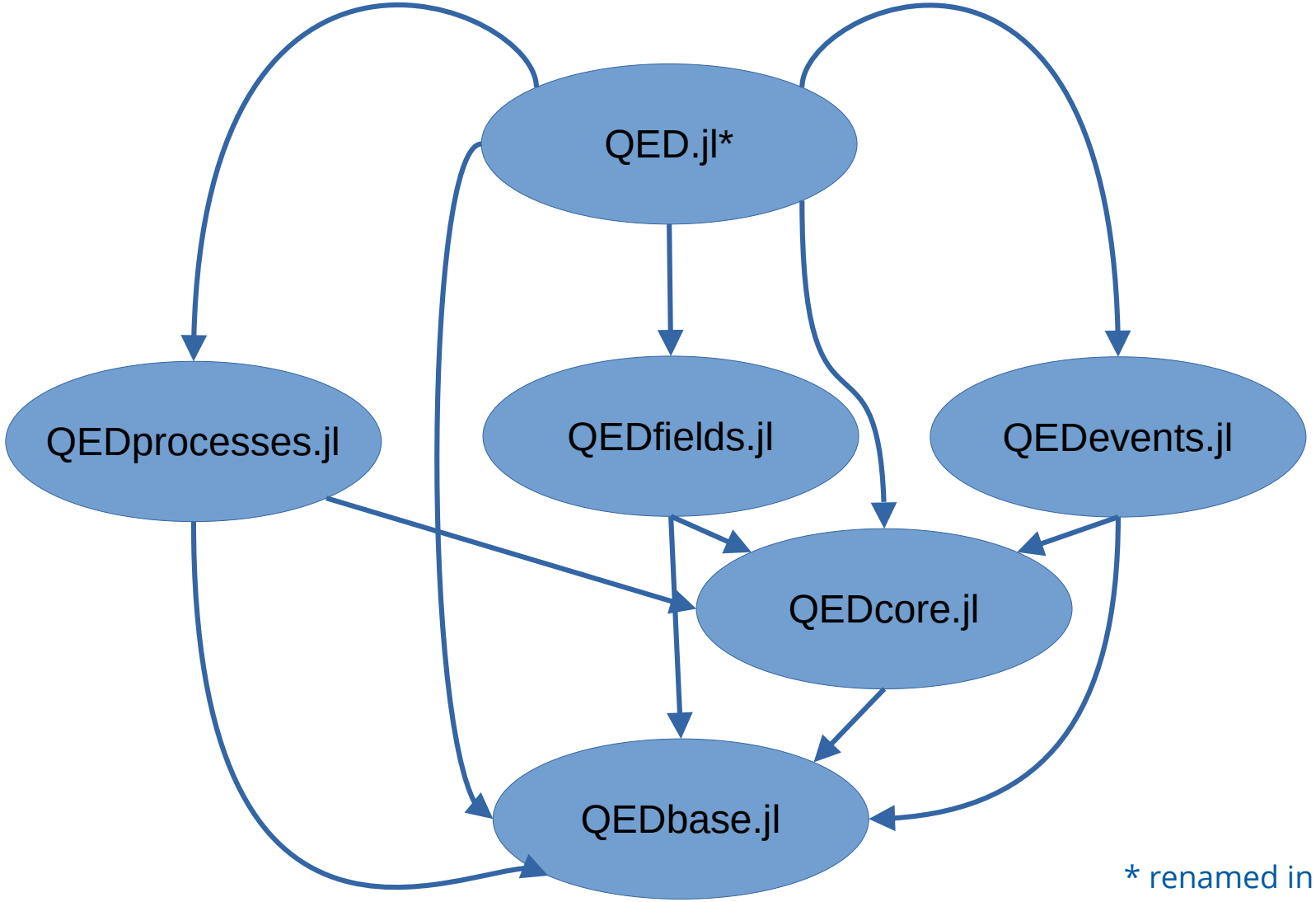
```
simeon@fwu066:~/projects/IntegrationTests.jl$ julia --project findIntegrationTests.jl
["QED.jl", "QEDprocess.jl"]
```

Old QED.jl Graph



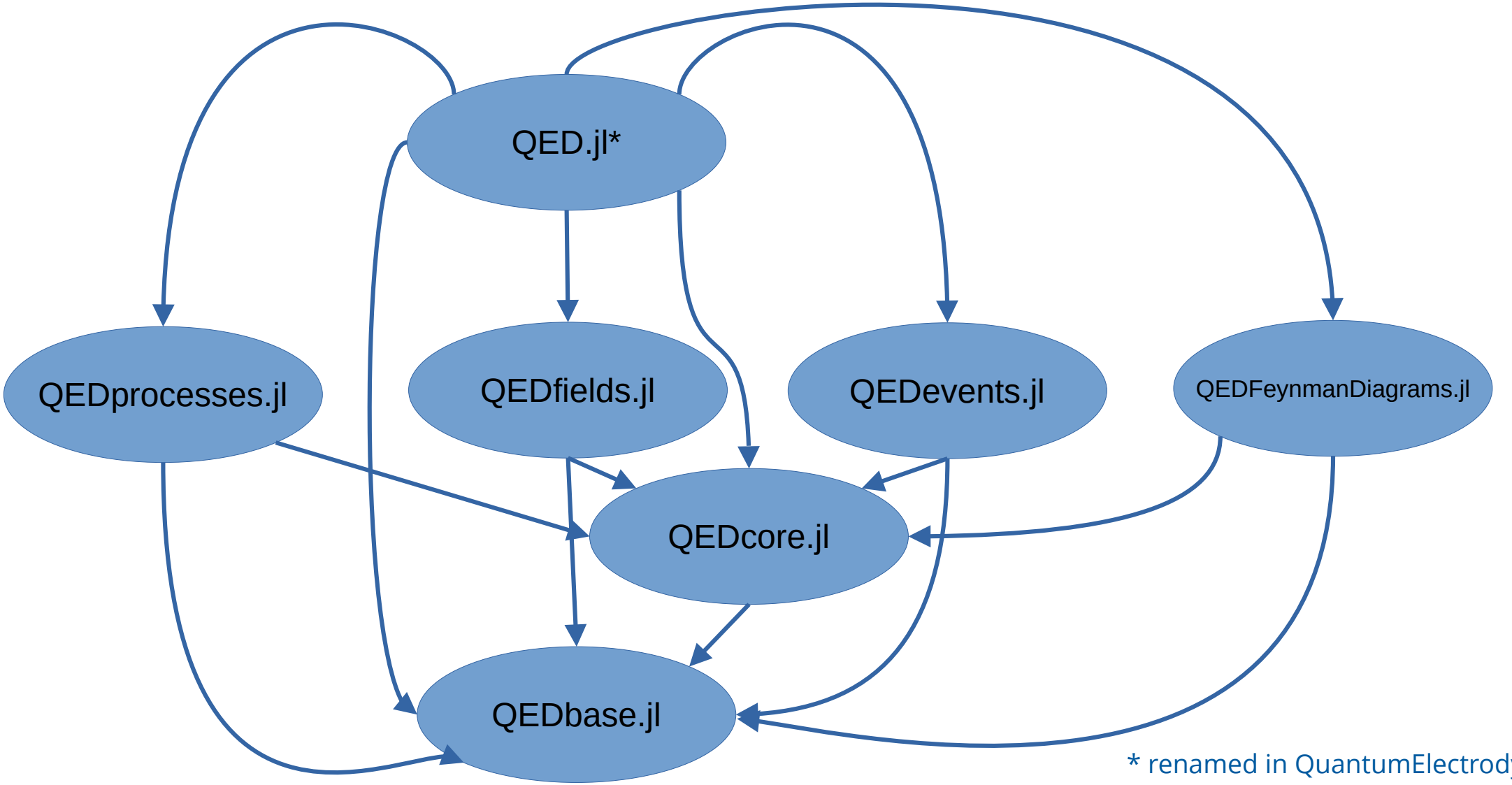
* graph is outdated

New QuantumElectrodynamics.jl Graph



* renamed in QuantumElectrodynamics.jl

Adding QEDFeynmanDiagrams.jl



* renamed in QuantumElectrodynamics.jl

IntegrationTests.jl with GitHub Actions

GitHub Actions

- GitHub Actions matrix mechanism
- Allows to create jobs with a given template and a range of parameters (e.g. Version numbers)
- Parameters can be hard coded or dynamically generated

https://qedjl-project.github.io/IntegrationTests.jl/stable/pipeline_tutorials/#GitHub-Actions

GitHub Actions: job yaml

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

```yaml
jobs:
    generate:
        name: Github Action - Generator Integration Jobs
        runs-on: ubuntu-latest
        steps:
        - uses: actions/checkout@v3
        - uses: julia-actions/setup-julia@v1
            with:
            version: "1.10"
            arch: x64
        - uses: julia-actions/cache@v1
        - uses: julia-actions/julia-buildpkg@v1
        - id: set-matrix
            run: echo "matrix=$(julia --project=${GITHUB_WORKSPACE} generateIntegrationTests.jl 2> /dev/null)" >> $GITHUB_OUTPUT
        outputs:
            matrix: ${{ steps.set-matrix.outputs.matrix }}

    run-matrix:
        needs: generate
        runs-on: ubuntu-latest
        strategy:
            matrix:
                package: ${{ fromJson(needs.generate.outputs.matrix) }}
        steps:
            # define the job body of your integration test here depending on the `matrix.package` parameter
            - run: echo "run Integration Test for package ${{ matrix.package }}"
```

https://qedjl-project.github.io/IntegrationTests.jl/stable/pipeline_tutorials/#GitHub-Actions

GitHub Actions: job yaml

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

```yaml
jobs:
  generate:
    name: Github Action - Generator Integration Jobs
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3
    - uses: julia-actions/setup-julia@v1
      with:
        version: "1.10"
        arch: x64
    - uses: julia-actions/cache@v1
    - uses: julia-actions/julia-buildpkg@v1
    - id: set-matrix
      run: echo "matrix=$(julia --project=${GITHUB_WORKSPACE} generateIntegrationTests.jl 2> /dev/null)" >> $GITHUB_OUTPUT
    outputs:
      matrix: ${{ steps.set-matrix.outputs.matrix }}

  run-matrix:
    needs: generate
    runs-on: ubuntu-latest
    strategy:
      matrix:
        package: ${{ fromJson(needs.generate.outputs.matrix) }}
    steps:
      # define the job body of your integration test here depending on the `matrix.package` parameter
      - run: echo "run Integration Test for package ${{ matrix.package }}"
```

Job parameter generator

https://qedjl-project.github.io/IntegrationTests.jl/stable/pipeline_tutorials/#GitHub-Actions

GitHub Actions: job yaml

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

```yaml
jobs:
  generate:
    name: Github Action - Generator Integration Jobs
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v3
    - uses: julia-actions/setup-julia@v1
      with:
        version: "1.10"
        arch: x64
    - uses: julia-actions/cache@v1
    - uses: julia-actions/julia-buildpkg@v1
    - id: set-matrix
      run: echo "matrix=$(julia --project=${GITHUB_WORKSPACE} generateIntegrationTests.jl 2> /dev/null)" >> $GITHUB_OUTPUT
    outputs:
      matrix: ${{ steps.set-matrix.outputs.matrix }}

  run-matrix:
    needs: generate
    runs-on: ubuntu-latest
    strategy:
      matrix:
        package: ${{ fromJson(needs.generate.outputs.matrix) }}
    steps:
      # define the job body of your integration test here depending on the `matrix.package` parameter
      - run: echo "run Integration Test for package ${{ matrix.package }}"
```

Job parameter generator

Job template

https://qedjl-project.github.io/IntegrationTests.jl/stable/pipeline_tutorials/#GitHub-Actions

GitHub Actions: job yaml

```yaml
    - id: set-matrix
      run: echo "matrix=$(julia --project=${GITHUB_WORKSPACE} generateIntegrationTests.jl 2> /dev/null)" >> $GITHUB_OUTPUT
  outputs:
    matrix: ${{ steps.set-matrix.outputs.matrix }}
```

GitHub Actions: job yaml

```julia
using IntegrationTests
using Pkg

if !isinteractive()
    # search packages, which are dependent on QEDfields.jl
    # ignore all packages which does not start with QED
    depending_packages = IntegrationTests.depending_projects(
                                "QEDfields.jl", r"^QED*")

    println(depending_packages)
end
```

```yaml
      - id: set-matrix
        run: echo "matrix=$(julia --project=${GITHUB_WORKSPACE} generateIntegrationTests.jl 2> /dev/null)" >> $GITHUB_OUTPUT
      outputs:
        matrix: ${{ steps.set-matrix.outputs.matrix }}
```

GitHub Actions: job yaml

```julia
using IntegrationTests
using Pkg

if !isinteractive()
    # search packages, which are dependent on QEDfields.jl
    # ignore all packages which does not start with QED
    depending_packages = IntegrationTests.depending_projects(
                                "QEDfields.jl", r"^QED*")

    println(depending_packages)
end
```

```yaml
      - id: set-matrix
        run: echo "matrix=$(julia --project=${GITHUB_WORKSPACE} generateIntegrationTests.jl 2> /dev/null)" >> $GITHUB_OUTPUT
      outputs:
        matrix: ${{ steps.set-matrix.outputs.matrix }}
```

```
simeon@fwu066:~/projects/IntegrationTests.jl$ julia --project findIntegrationTests.jl
["QED.jl", "QEDprocess.jl"]
```

GitHub Actions: job yaml

```julia
using IntegrationTests
using Pkg

if !isinteractive()
    # search packages, which are dependent on QEDfields.jl
    # ignore all packages which does not start with QED
    depending_packages = IntegrationTests.depending_projects(
                                "QEDfields.jl", r"^QED*")

    println(depending_packages)
end
```

```yaml
      - id: set-matrix
        run: echo "matrix=$(julia --project=${GITHUB_WORKSPACE} generateIntegrationTests.jl 2> /dev/null)" >> $GITHUB_OUTPUT
    outputs:
      matrix: ${{ steps.set-matrix.outputs.matrix }}
```

```
simeon@fwu066:~/projects/IntegrationTests.jl$ julia --project findIntegrationTests.jl
["QED.jl", "QEDprocess.jl"]
```

GitHub Actions: job yaml

```yaml
    - id: set-matrix
      run: echo "matrix=$(julia --project=${GITHUB_WORKSPACE} generateIntegrationTests.jl 2> /dev/null)" >> $GITHUB_OUTPUT
    outputs:
      matrix: ${{ steps.set-matrix.outputs.matrix }}
```



```yaml
  run-matrix:
    needs: generate
    runs-on: ubuntu-latest
    strategy:
      matrix:
        package: ${{ fromJson(needs.generate.outputs.matrix) }}
    steps:
      # define the job body of your integration test here depending on the `matrix.package` parameter
      - run: echo "run Integration Test for package ${{ matrix.package }}"
```

IntegrationTests.jl with GitLab CI

GitLab CI: parent-child pipeline

- GitLab CI parent-child pipelines
- Allows to run jobs from a (dynamically) generated job yaml
- Requires generator and trigger job

https://qedjl-project.github.io/IntegrationTests.jl/stable/pipeline_tutorials/#GitLab-CI

GitLab CI: job yaml

```yaml
stages:
  - generator
  - runTests

generateIntegrationTests:
  stage: generator
  image: "julia:1.10"
  script:
    - julia --project=. -e 'import Pkg; Pkg.instantiate()'
    - julia --project=. generateIntegrationTests.jl 2> /dev/null 1> jobs.yaml
    - cat jobs.yaml
  interruptible: true
  artifacts:
    paths:
      - jobs.yaml
    expire_in: 1 week

runIntegrationTests:
  stage: runTests
  trigger:
    include:
    - artifact: jobs.yaml
      job: generateIntegrationTests
    strategy: depend
```

https://qedjl-project.github.io/IntegrationTests.jl/stable/pipeline_tutorials/#GitLab-CI

GitLab CI: job yaml

```yaml
stages:
  - generator
  - runTests

generateIntegrationTests:
  stage: generator
  image: "julia:1.10"
  script:
    - julia --project=. -e 'import Pkg; Pkg.instantiate()'
    - julia --project=. generateIntegrationTests.jl 2> /dev/null 1> jobs.yaml
    - cat jobs.yaml
  interruptible: true
  artifacts:
    paths:
      - jobs.yaml
    expire_in: 1 week

runIntegrationTests:
  stage: runTests
  trigger:
    include:
    - artifact: jobs.yaml
      job: generateIntegrationTests
    strategy: depend
```

Job generator

Trigger Job

https://qedjl-project.github.io/IntegrationTests.jl/stable/pipeline_tutorials/#GitLab-CI

GitLab CI: job yaml

```
script:
  - julia --project=. -e 'import Pkg; Pkg.instantiate()'
  - julia --project=. generateIntegrationTests.jl 2> /dev/null 1> jobs.yaml
  - cat jobs.yaml
```

```
function print_job_yaml(package_name::AbstractString)
    job_yaml = """integrationTest$package_name:
          image: "alpine:latest"
          script:
              - echo "run Integration Test for package $package_name"

      """
    return print(job_yaml)
end

if !isinteractive()
    depending_packages = IntegrationTests.depending_projects(
                                "QEDfields.jl", r"^QED*")
    for dep in depending_packages
        print_job_yaml(dep)
    end
end
```

```
integrationTestQED.jl:
    image: "alpine:latest"
    script:
        - echo "run Integration Test for package QED.jl"

integrationTestQEDprocess.jl:
    image: "alpine:latest"
    script:
        - echo "run Integration Test for package QEDprocess.jl"
```

GitLab CI: job yaml

```yaml
stages:
  - generator
  - runTests

generateIntegrationTests:
  stage: generator
  image: "julia:1.10"
  script:
    - julia --project=. -e 'import Pkg; Pkg.instantiate()'
    - julia --project=. generateIntegrationTests.jl 2> /dev/null 1> jobs.yaml
    - cat jobs.yaml
  interruptible: true
  artifacts:
    paths:
      - jobs.yaml
    expire_in: 1 week

runIntegrationTests:
  stage: runTests
  trigger:
    include:
    - artifact: jobs.yaml
      job: generateIntegrationTests
    strategy: depend
```

Lessons Learned

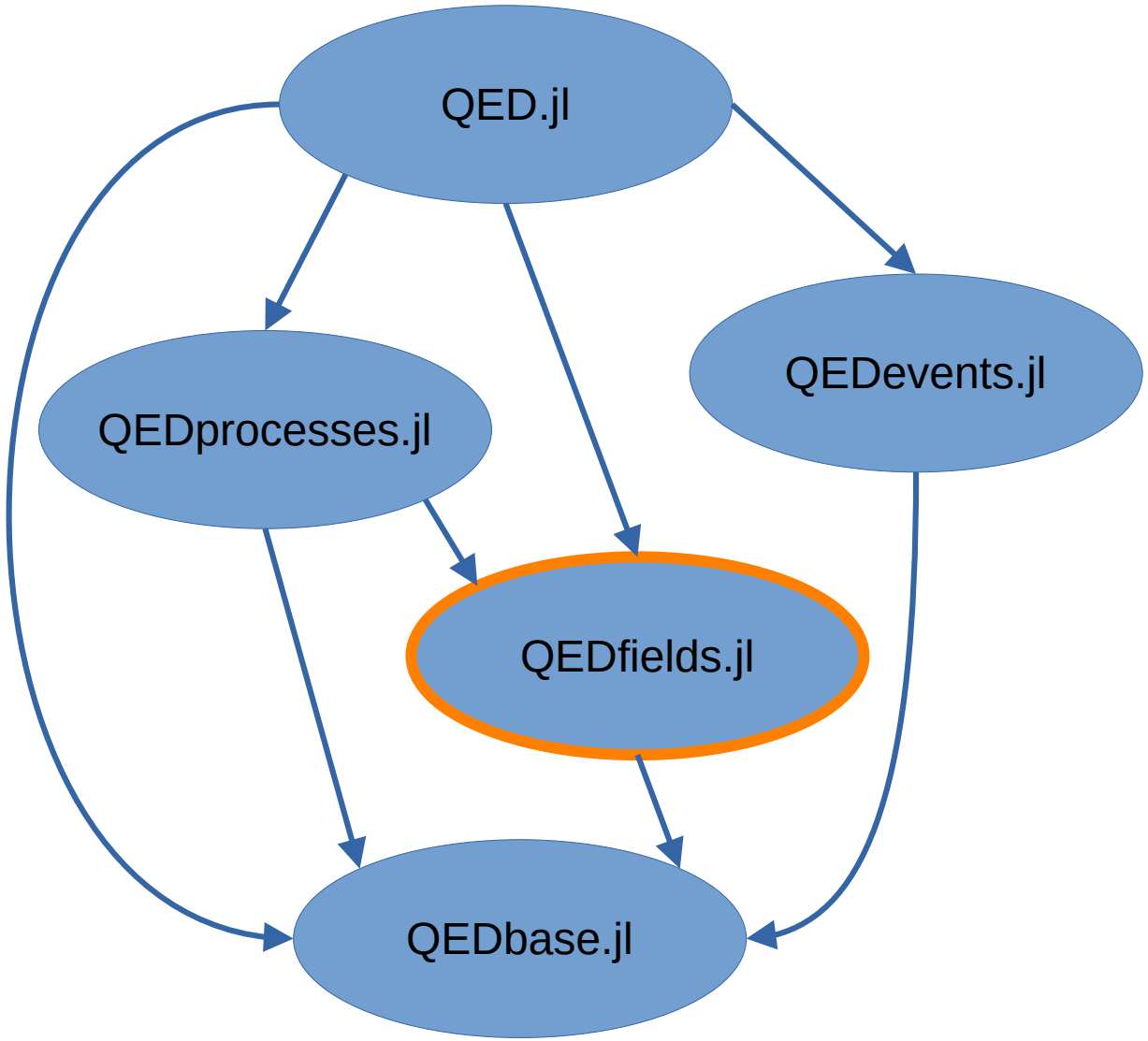
- Permanent integration testing avoids a lot of problems
- Automation is important for new developer

  → A new developer does not know that it's his code can break other codes
- Moving parts of helper code in extra packages improves quality and reliability

https://github.com/QEDjl-project/IntegrationTests.jl

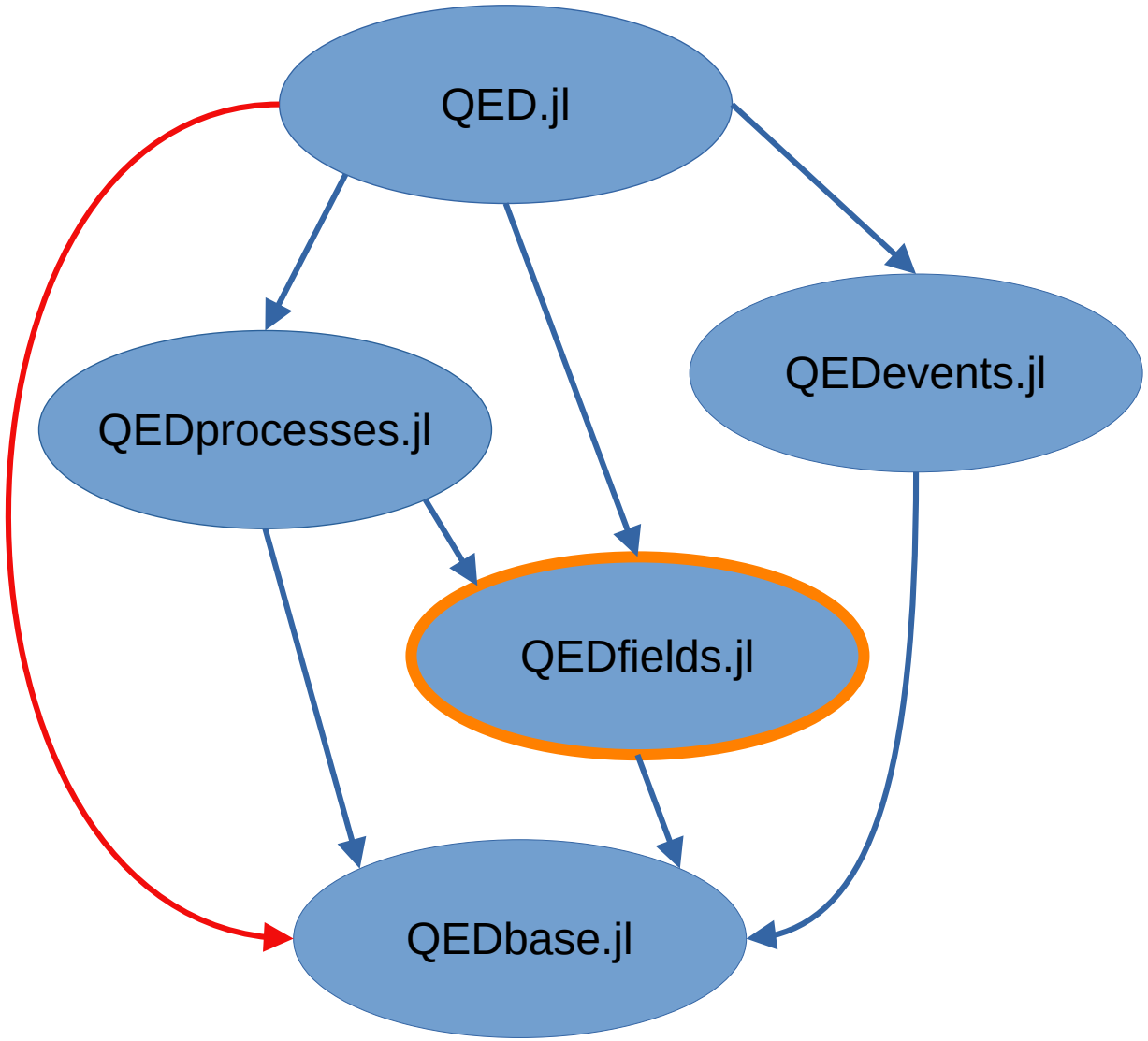
Backup

QED.jl Graph



QEDfields.jl modified

QED.jl Graph



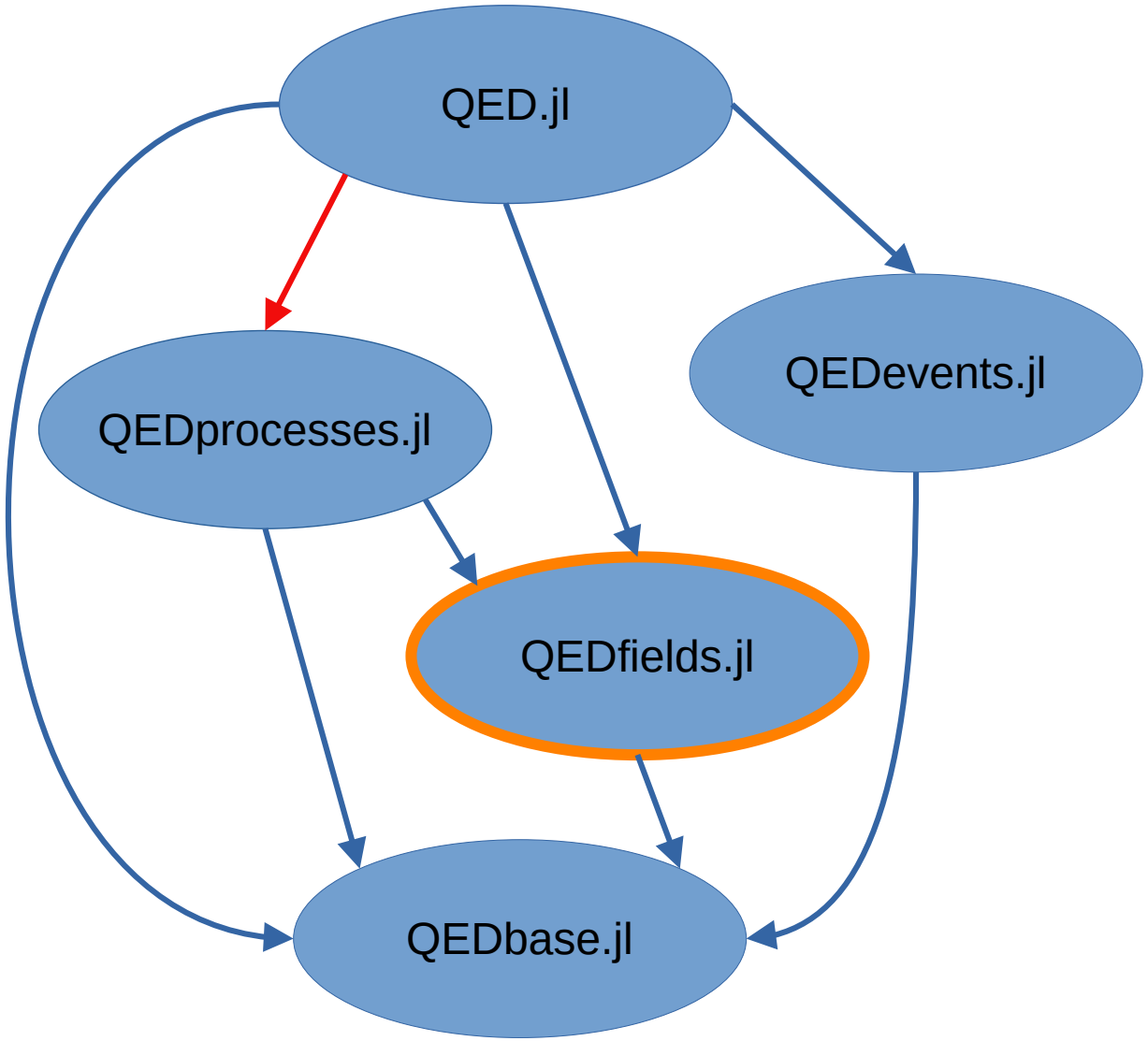
QEDfields.jl modified

Dependent packages:

Visited

- QEDbase.jl

QED.jl Graph



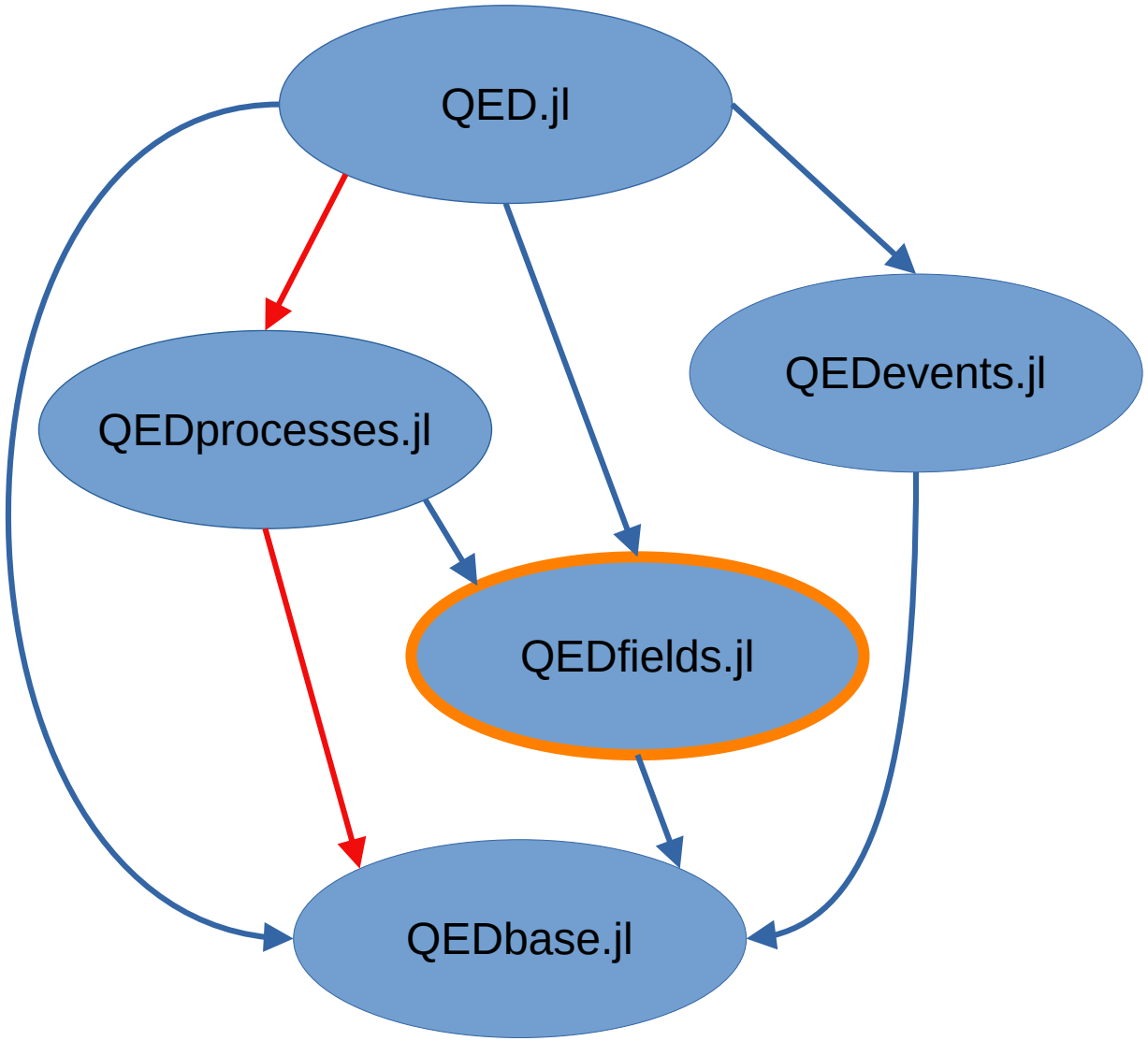
QEDfields.jl modified

Dependent packages:

Visited

- QEDbase.jl
- QEDprocesses.jl

QED.jl Graph



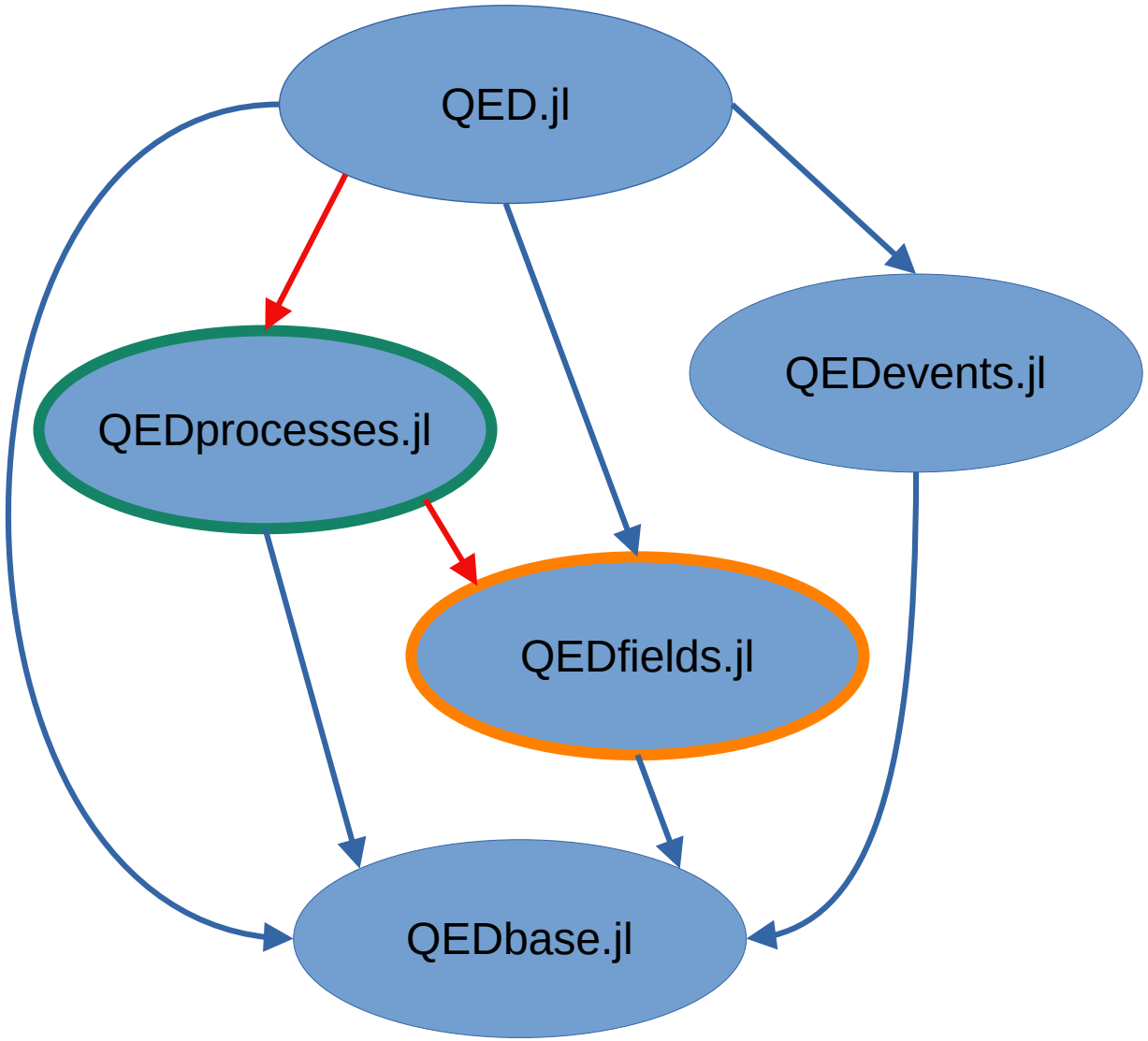
QEDfields.jl modified

Dependent packages:

Visited

- QEDbase.jl
- QEDprocesses.jl

QED.jl Graph



QEDfields.jl modified

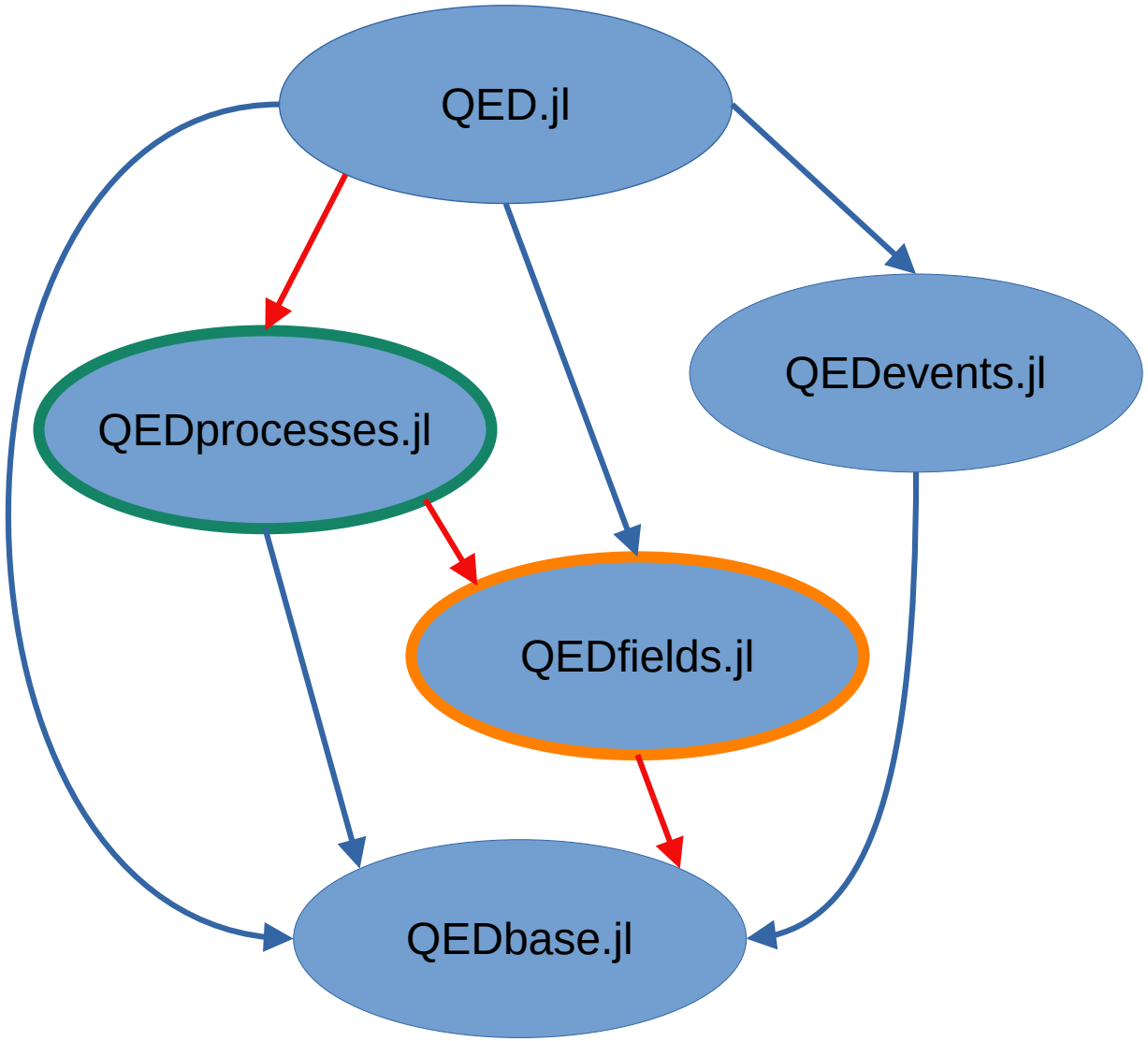
Dependent packages:

- QEDProcesses.jl

Visited

- QEDbase.jl
- QEDprocesses.jl
- QEDfields.jl

QED.jl Graph



QEDfields.jl modified

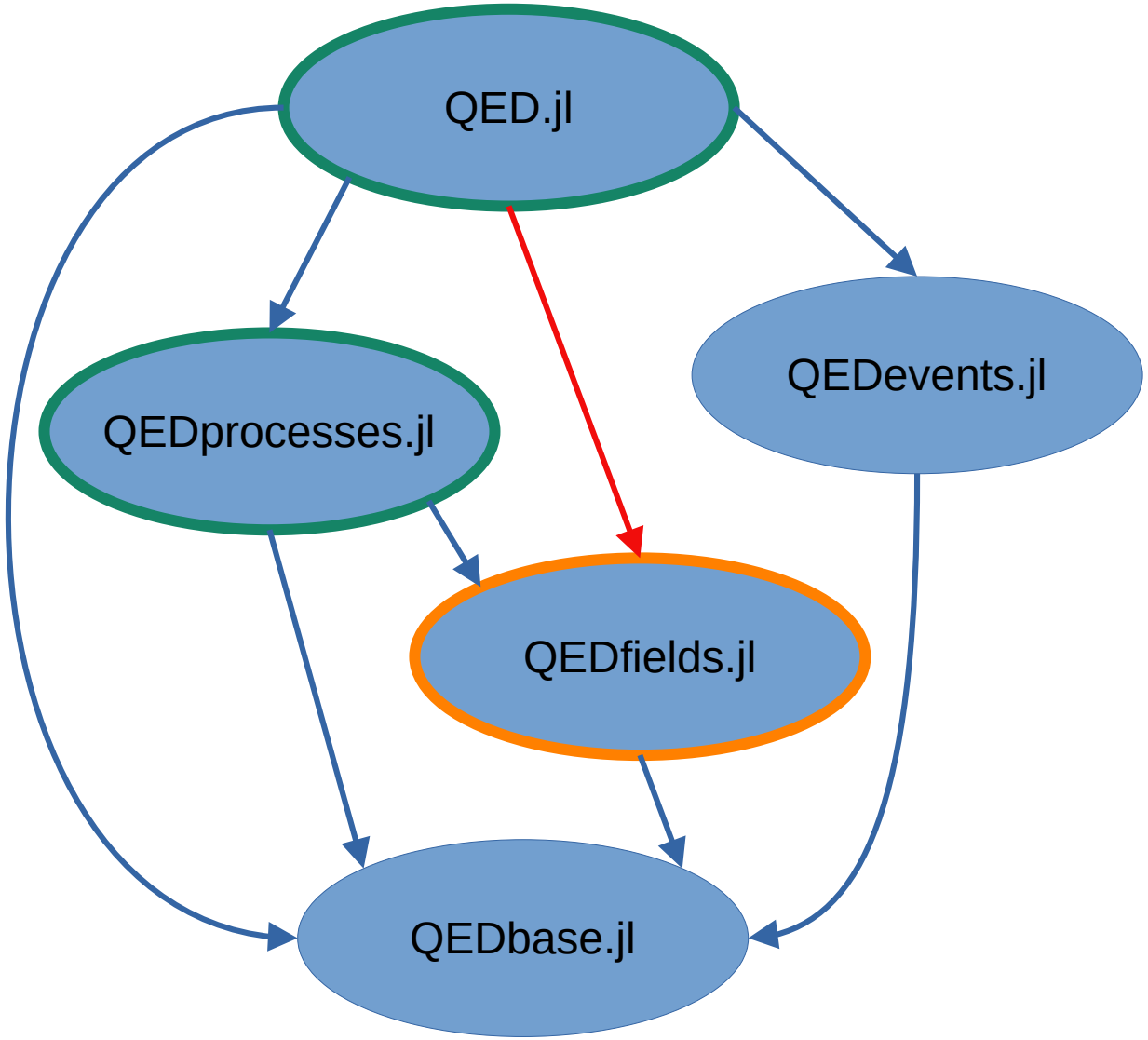
Dependent packages:

- QEDProcesses.jl

Visited

- QEDbase.jl
- QEDprocesses.jl
- QEDfields.jl

QED.jl Graph



QEDfields.jl modified

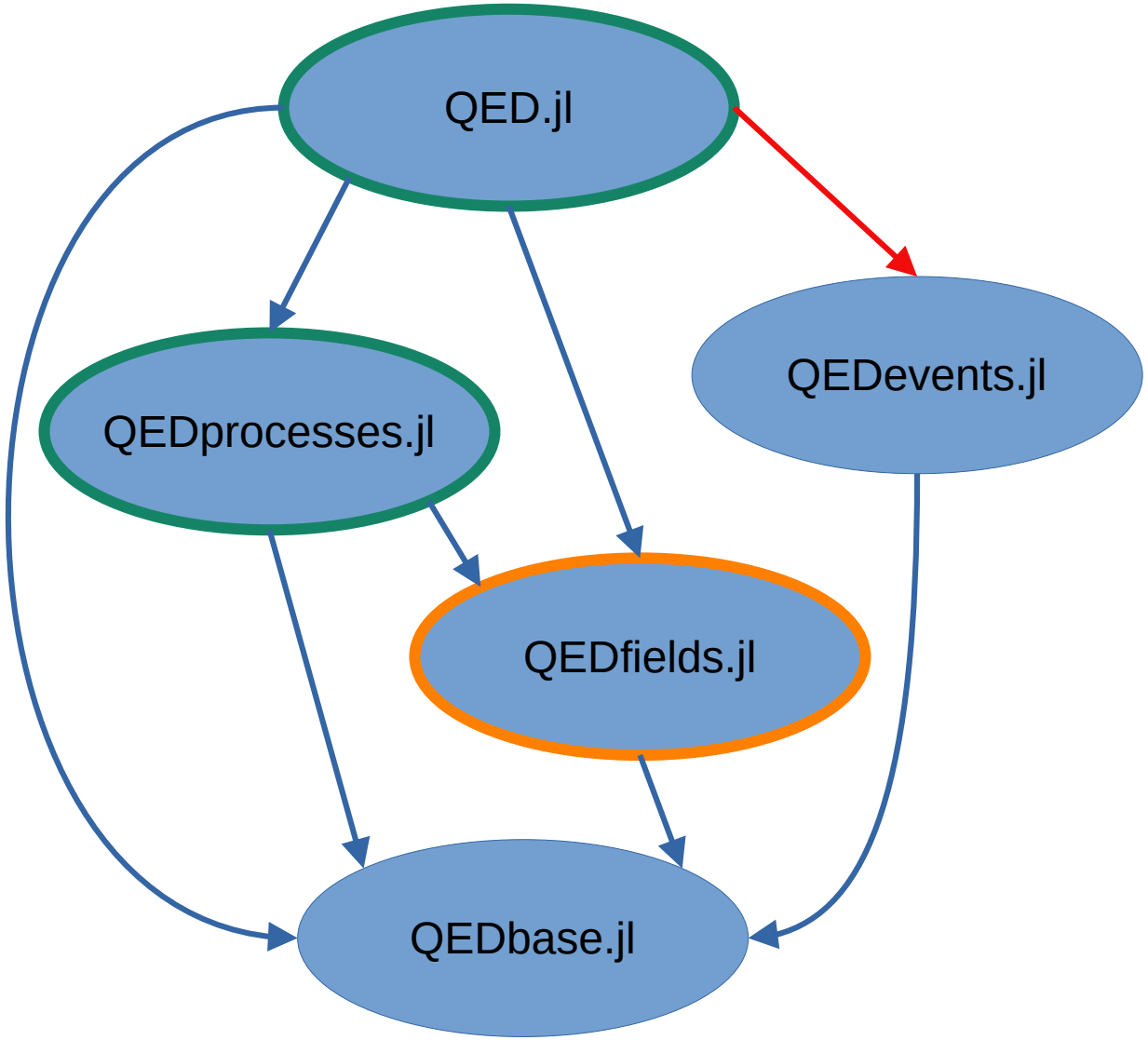
Dependent packages:

- QEDProcesses.jl
- QED.jl

Visited

- QEDbase.jl
- QEDprocesses.jl
- QEDfields.jl

QED.jl Graph



QEDfields.jl modified

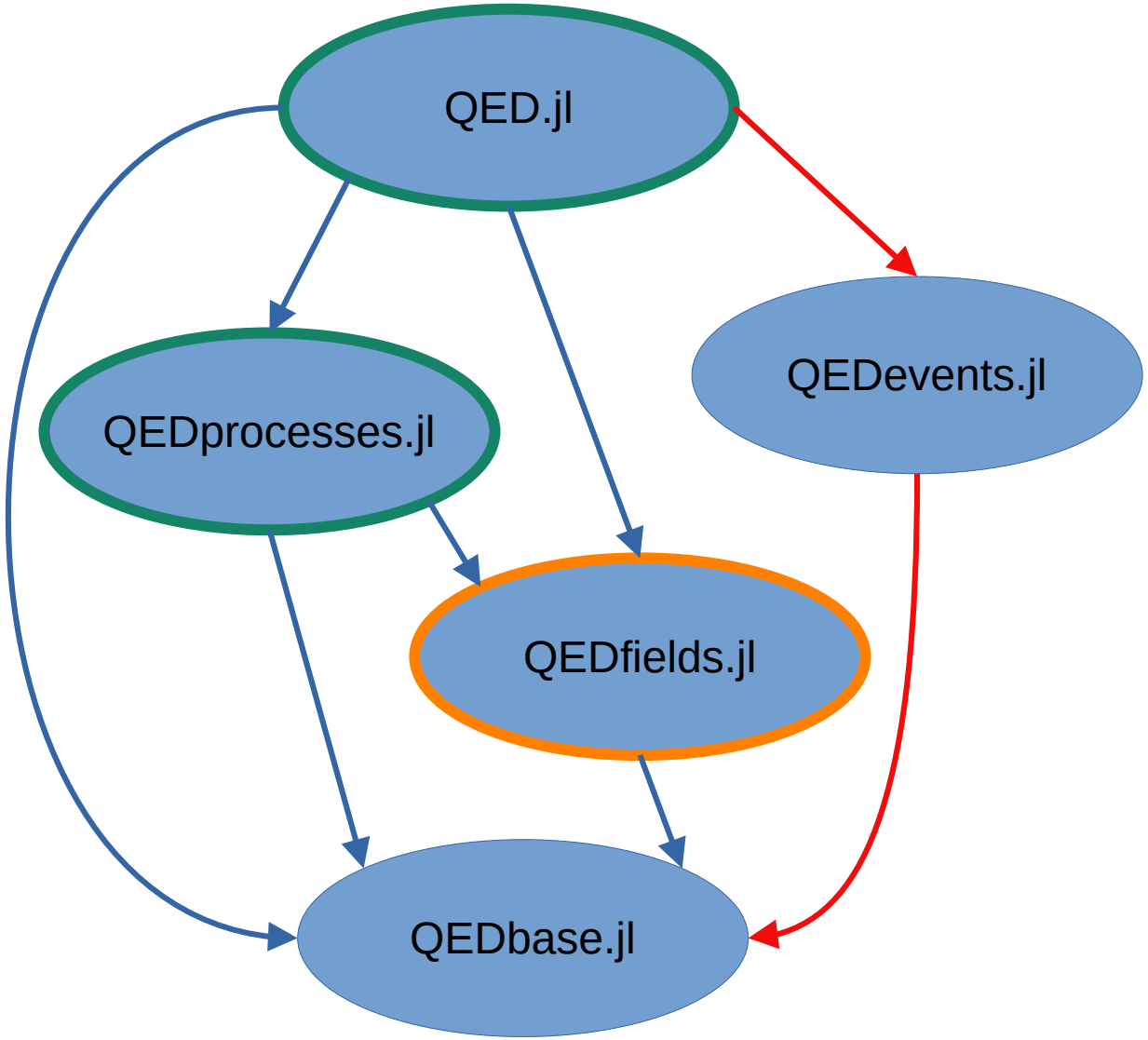
Dependent packages:

- QEDProcesses.jl
- QED.jl

Visited

- QEDbase.jl
- QEDprocesses.jl
- QEDfields.jl
- QEDevents.jl

QED.jl Graph



QEDfields.jl modified

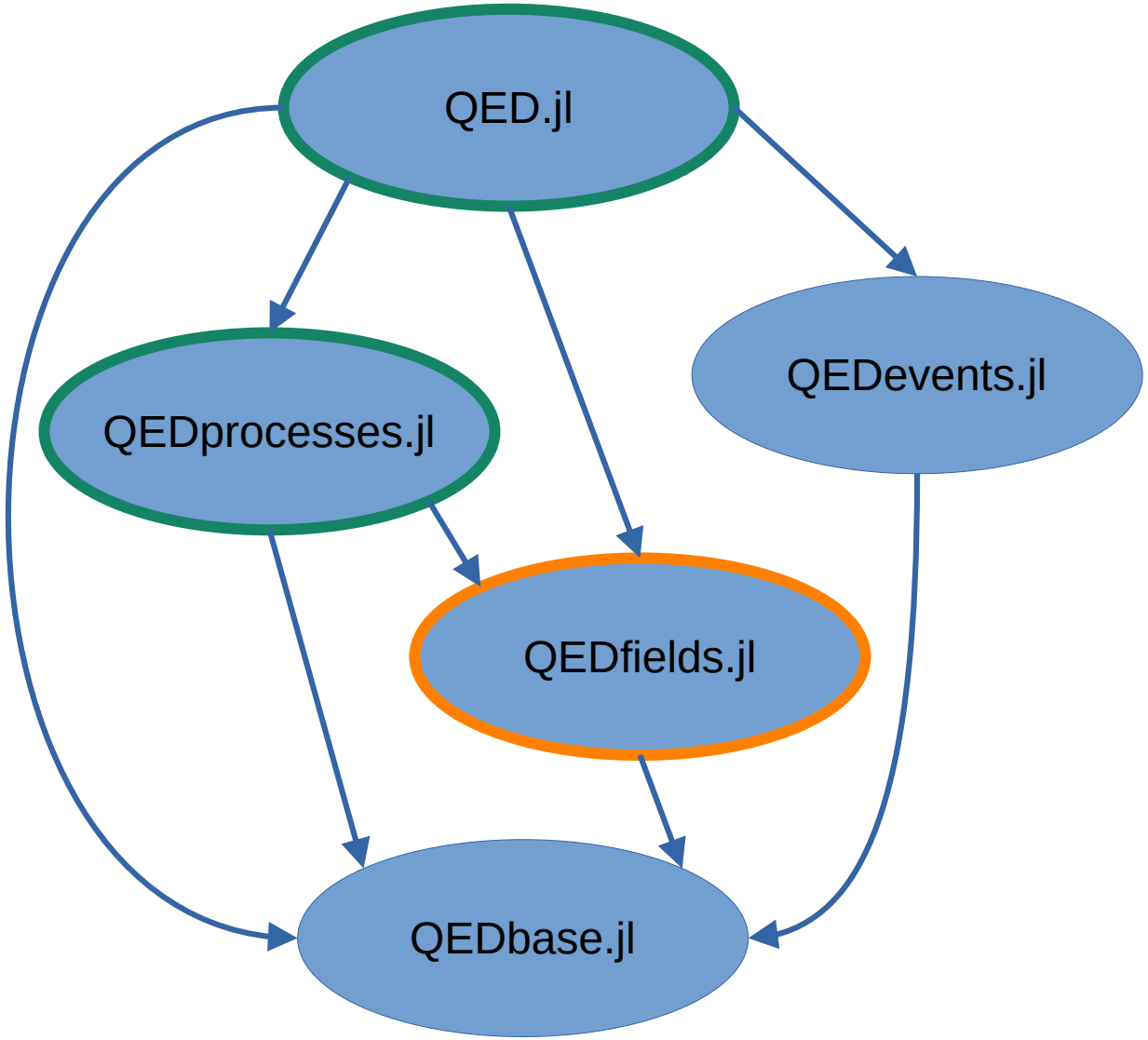
Dependent packages:

- QEDProcesses.jl
- QED.jl

Visited

- QEDbase.jl
- QEDprocesses.jl
- QEDfields.jl
- QEDevents.jl

QED.jl Graph



QEDfields.jl modified

Dependent packages:

- QEDProcesses.jl
- QED.jl

Visited

- QEDbase.jl
- QEDprocesses.jl
- QEDfields.jl
- QEDevents.jl