Scenarios: Use cases

Voorbeeld format

De use cases zullen op twee manieren gerepresenteerd worden; elke use case zal in de vorm van een tabel opgebouwd worden:

Hier de titel	
Actor	Hier de uitvoerende actor
Precondities	Hier de precondities
Trigger	Hier de trigger
Postcondities	Hier de postcondities
Happy path	Hier de happy path
Alternatief	Hier een alternatief pad
Notities	Hier eventuelen notities

Ook zullen de use cases in de vorm van UML diagrammen opgebouwd worden:

```
@startuml usecase
left to right direction

rectangle "Update user data in database" {
    usecase interaction as "Update user data"
    usecase crud as "CRUD action" <<interface>>
    interaction <|.. crud : Extends
    usecase display as "Display data to user"
    display <|.. crud: Extends
}

actor User as "System user"

display -- User
User -- interaction
@enduml
```

Start tests

Starten van de tests	
Actor	Hier de uitvoerende actor
Precondities	De test case classes en de te testen code staan beschikbaar in de huidige directory

Starten van de tests	
Trigger	Het test commando wordt uitgevoerd
Postcondities	De tests zijn uitgevoerd en de commando geeft een display weer met testresultaten
Happy path	1. Test commando wordt aangeroepen 2. Framework doorzoekt files op zoek naar tests 3. Framework start database 4. Framework vult database met dummy data 5. Apps worden opgestart met behulp van docker image webserver 6. Tests worden ingeladen 7. Test wordt uitgevoerd 8. Commando laat testresultaat zien
Alternatief	1. Test commando wordt aangeroepen 2. Framework kan geen tests vinden 3. Commando laat error bericht zien
Notities	Voor stap 7. "Test wordt uitgevoerd" van happy path, zie Uitvoeren enkele test

```
@startuml usecase
skinparam linetype ortho

actor caller as "Caller"
note top of caller: Can be human or CircleCI

rectangle "Start tests" {
    usecase start as "Call framework"
    usecase search as "Search files for test classes
    --No test classes found:--
    Throw error"
    usecase db as "Start local test database"
    usecase init as "Fill database with dummy data"
    usecase servers as "Start server docker images"
    usecase load as "Load the test classes"
    usecase execute as "Execute test classes"
    usecase return as "Return test results"
    usecase error as "Display error "No testcases found""
}

caller- start
search --> error

start --> search
search --> db
db --> init
start --> servers
servers .|> load
init .|> load
```

```
load --> execute
execute --> return
return -> caller

@enduml
```

Start automatisch testen

Automatisch testen	
Actor	CircleCI software
Precondities	De nieuwe code is naar de github repository gepushed
Trigger	Er is een taak gestart om de functionele tests door CircleCI uit te laten voeren
Postcondities	CircleCI reageert op de testresultaten
Happy path	1. CircleCI krijgt signaal om te starten met testen 2. CircleCI container wordt opgebouwd 3. Cache van eerdere tests wordt opgehaald 4. Tests worden uitgevoerd 5. Files worden opgeschoont 6. CirlceCI gaat verder met de volgende task
Alternatief	1. CircleCI krijgt signaal om te starten met testen 2. CircleCI container wordt opgebouwd 3. Cache van eerdere tests is niet beschikbaar, deze zal niet gebruikt worden 4. Tests worden uitgevoerd 5. Files worden opgeschoont 6. Cache wordt opgeslagen 7. CirlceCI gaat verder met de volgende task
Notities	Voor stap 4. "Test worden uitgevoerd", zie Start tests

```
@startuml usecase
skinparam linetype ortho

actor CI as "CircleCI"

rectangle "Automated testing" {
    usecase start as "Bouw CircleCI container"
    usecase searchcache as "Check for existing cache
    --No cache available:--
        Skip to test execution,
        framework will build
        necessary files itself"
    usecase run as "Execute tests"
    usecase cleanup as "Clean up files"
    usecase addcache as "Create cache"
    usecase getcache as "Retrieve and use cache"
```

```
}

CI - start
start --> searchcache
searchcache - getcache
searchcache - addcache
searchcache --> run
run --> cleanup

@enduml
```

Uitvoeren enkele test

Uitvoeren enkele test	
Actor	Framework
Precondities	De omgeving is opgezet en de functionaliteit Starten van tests is doorlopen. Hierbij zijn de database en de backend web-api opgestelt en zijn alle huidige migraties op de database uitgevoerd. Ook is er een te testen test class bekend en meegegeven.
Trigger	Taak Starten van tests is afgerond en roept individuele tests aan.
Postcondities	De tests zullen uitgevoerd zijn en een resultaat terug hebben gegeven.
Happy path	1. Test class setup methode wordt uitgevoerd 2. Test class wordt doorzocht voor test functie annotaties 3. Tests worden uitgevoerd 4. Testresultaat wordt asynchroon aan de aanroeper doorgestuurd 5. Testclass teardown wordt uitgevoerd 6. Volgende test wordt gestart
Alternatief	1. Test class setup methode wordt uitgevoerd 2. Test class wordt doorzocht voor test functie annotaties 3. Tests worden uitgevoerd 4. Test loopt tegen een onverwachte error aan 5. Geef error door aan aanroeper 6. Sla deze test over, ga door naar volgende test
Notities	Deze use case gaat ervanuit dat use case Starten van tests al uitgevoerd is.

```
@startuml usecase
skinparam linetype ortho


actor caller as "Framework"

rectangle usecase as "Executing individual tests" {
    usecase setup as "Execute setup function"
    usecase search as "Search test class for
```

```
        @Test annotated functions
        --No test case found:--
        Throw error"
    usecase execute as "@Test functions are executed"
    usecase return as "Testresults are emitted concurrently
    to the framework"
    usecase teardown as "Teardown function is executed"
}

setup - caller
setup --> search
search --> execute
search -> caller
execute -|> return
return -> caller
execute --> teardown

@enduml
```

Process view: Activity diagrams

Executing tests

```
@startuml Execute functional tests activity diagram
start

:Scan for php files (recursive);

note right
    Directory to search is given
    as commandline argument to
    framework call.
    Argument 'n' can be given
    to indicate a preferered batch
    size. If n is not given, only
    one batch is spawned.
end note

if (Test found?) then (No: Throw error: "No files found")
    stop
else (Yes)
endif

if (n is defined in call) then (True)
    :Split files into batches of n size;
else (False)
endif

fork
    repeat
```

```
        if (File contains @Test annotations) then (False)
            end
            note left
                Skip any file that doesn't
                contain @Test annotated functions;
                continue to next file.
            end note
        else (True)
        endif
        :Execute the test's <i>Setup()</i> method;
        while (Have all tests been run?) is (False)
            :Execute the next <i>@test</i> annotated function;
        endwhile (True)
        :Execute the test's <i>Teardown()</i> method;
    repeat while (All files in batch tested?) is (False) not (True)
end fork

note right
    This is executed in parallel
    for all defined batches.
    If n == 1, execute once.
end note

stop
@enduml
```

CircleCI workflow

```
@startuml Functional testing workflow activity diagram
start
:Setup;
fork
    if (Cache contains Database?) then (True)
        :Restore cached database;
    else (False)
        :Build new database;
    endif
    fork
        :Start database;
    fork again
        :Pull bouw7/bouw7;
    end fork
    if (Database migration status) then (Not fully migrated)
        :Execute database migrations;
    else (Fully migrated)
    endif
    fork
        :Save database to cache;
        kill
    fork again
    end fork
```

```
    :Fill database with
    dummy data from artifact;
fork again
    if (Cache contains bouw7/web-api?) then (True)
        :Restore cached bouw7/web-api;
    else (False)
    endif
    :Git pull bouw7/web-api;
    :Composer install;
    fork
        :Save bouw7/web-api to cache;
        kill
    fork again
    end fork

end fork
:Start web-api docker container;
:Execute tests;
:Save test results in artifact;
fork
    :Terminate web-api docker;
fork again
    :Terminate database;
fork again
    :Clean up files;
end fork
stop
@enduml
```

Development view: Component diagram

```
@startuml Component diagram
skinparam Linetype ortho
skinparam componentStyle uml1
left to right direction

together {
    component "Web API" <<Code repository>> as web_api {
        component ":Test cases" <<PHP class>> as test_cases
        component ":Web server" <<Apache>> as api_server
        component ":Source code" as source
        component ":Database connection settings" as db_conn
    }
    interface "API endpoints" <<HTTPS>> as endpoints
    interface "Available tests" as tests
    interface "Environment variables" as env
}

component "Functional testing framework" <<Composer package>> as framework {
    component ":Test runner" <<PHP runtime>> as runner
```

```
    together {
        component ":Test batch manager" <<PHP runtime>> as manager
        component ":Test database" <<SQLite>> as test_db
        component ":Test class" <<PHP runtime>> as test
        interface "Database connection" as sqlite_conn
        interface "Spawn batch tester" as spawn
        interface "Run tests" as run
    }
}

together {
    component "bouw7" <<Code repository>> as bouw7 {
        component "Database migrations" as migrations
    }
    interface "Pull request" <<SSH>> as bouw7_pull
}

db_conn - env
runner ..( env

bouw7 - bouw7_pull
runner ..( bouw7_pull

endpoints - api_server
test ..( endpoints

tests -- test_cases
tests ). runner

api_server .> source

sqlite_conn - test_db
runner .( sqlite_conn

spawn - manager
runner .( spawn

test - run
run ).. manager
@enduml
```

Physical view: Deployment diagram

```
@startuml CircleCI Deployment Diagram
left to right direction
skinparam linetype ortho

cloud github as "Github" #line:LightGray;line.dashed {
    storage bouw7 as "Bouw7/bouw7 repository" {
        file frontend as "Frontend Vue.js code"
```

```
        file yii2 as "Yii2 code"
        file migrations as "Databasemigraties"
    }

    storage web_api as "Bouw7/web-api repository"{
        file api as "Web api code"
    }
}

github -[hidden] CircleCI_cloud

cloud CircleCI_cloud as "CircleCI" #line:LightGray;line.dashed {
    node CircleCI as "CircleCI host" {
        node main as "Main test container" << CircleCi container >>
#line:blue;line.dotted;text:blue {
            folder main_artifacts as "CircleCI Artifacts"{
                file func_coverage as "Code coverage output"
            }

            folder main_cache as "CircleCI Cache"{
                storage main_dependencies[
                    Opgeslagen dependencies
                    en packages
                ]
            }

            queue main_workflow [
                .circleci/functional_tests.yml
                ====
                Main testing workflow
            ]

            main_workflow = main_dependencies
            main_workflow = main_artifacts
            main_workflow <== api
        }

        node functional as "Functional test container" << CircleCi container >>
#line:blue;line.dotted;text:blue  {
            folder func_artifacts as "CircleCI Artifacts"{
                file func_dummy_data[
                    DummyData.sql
                    ----
                    Database
                    dummy data
                ]
                file func_results[
                    Testresultaten
                ]
            }

            folder func_cache as "CircleCI Cache"{
                storage func_dependencies[
                    Opgeslagen dependencies
```

```
                    en packages
                ]
                folder func_web_api_cache[
                    Cached web-api
                    code files
                ]
                database db_cache[
                    Cached database
                    files
                ]
            }

            folder func_ramdisk as "CircleCI Ramdisk"{
                node func_docker_web_api as "Web-api docker container"
            }

            database func_db as "Lokale database"

            queue func_workflow [
                .circleci/config.yml
                ====
                Functional testing workflow
            ]

            func_workflow == func_db
            func_workflow = func_artifacts
            func_workflow = func_ramdisk
            func_workflow = func_cache
            func_workflow <== api
            func_workflow <== migrations
        }

        folder artifacts as "CircleCI Artifacts"{

        }

        folder cache as "CircleCI Cache"{

        }

        folder ramdisk as "CircleCI Ramdisk"{

        }

        artifacts --[hidden] cache
        cache --[hidden] ramdisk
    }
}

main --[hidden] functional
@enduml
```

Logical view: Class diagram

```
@startuml Framework class diagram
skinparam Linetype ortho
together {
    interface TestCase {
        # Setup() : void
        # Teardown() : void
    }
    note left
        @Test annotations are used
        to define test functions
        within each TestCase
    end note

    interface Assert {
        {static} Assert(expected: mixed, actual: mixed): void
        {static} AssertQuery(expected: string, query: string): void
    }
    note left of Assert::AssertQuery
        AssertQuery Takes a value
        and a query and compares the
        return of the query to the value.
    end note
}

class TestCaller {
    + Run(argv: string[]) : int
    - SplitBatches(batchesAmount: int, tests: TestCase[], databasePath: string) :
    - ExecuteBatch() : Future
}

class TestBatchManager {
    - Batch: TestCase[]
    + __construct(Batch: TestCase[])
    - RunBatch(): void
}

TestBatchManager "1" - "*" TestCase: has >

TestCaller "1" -- "1..*" TestBatchManager: spawns >

TestCase "1" -- "1..*" Assert: uses >
@enduml
```