A talk by Claudia Roșu

claudia.rosu@mozaicworks.com
@claudia_rosu

Agenda

- Feature delivery
- Tests in the delivery flow
- Power of tests in practice
- What's next
- Core ideas

About me

- Software crafter
- Experience with Groovy, Grails, Spock, Java
- Active in communities


1. Feature delivery flow

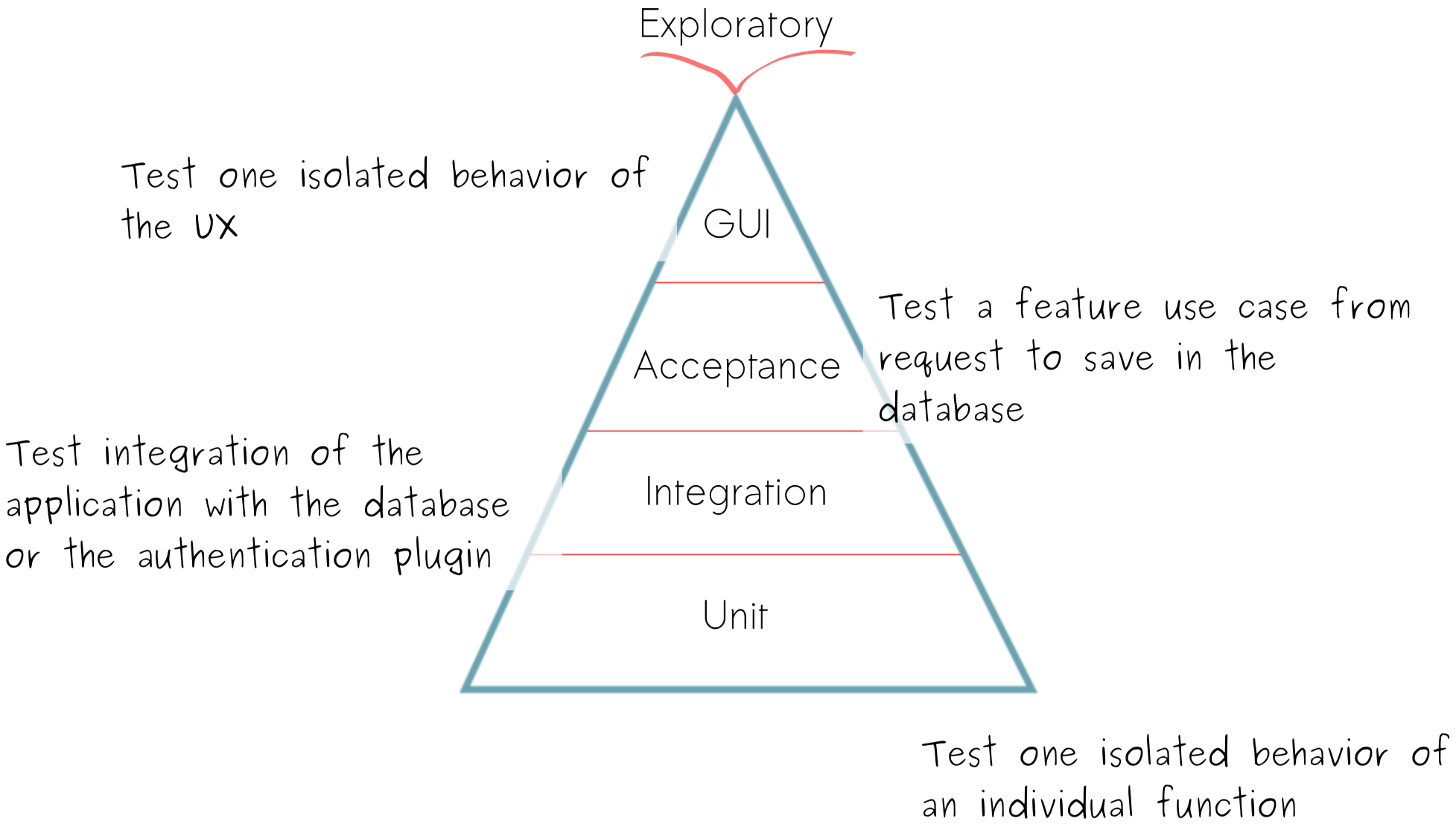
Background

- Innovative eHealth application for a general practitioner doctors association
- Client is not a product owner, nor a business analyst
- Development life cycle evolved over time

Development life cycle

2. Tests in the delivery flow

Testing strategy



Implementing the strategy

Building tests with Grails, Groovy and JUnit

Using tests for learning Grails framework faster

Using tests for preventing regression bugs

Groovy and Grails

Groovy is a powerful, optionally typed and dynamic language for the Java platform. - http://www.groovy-lang.org/

Grails is a powerful web framework, for the Java platform aimed at multiplying developers' productivity thanks to a Convention-over-Configuration - https://grails.org/

Time passes and

We know Grails&Groovy now

We want to get away without regression bugs

Package of tests to maintain

Future tests to write

Reduce tests number

Maximize the work not done

Invest in Software Design

Enjoying writing tests

Spock is a testing and specification framework for Java and Groovy applications. What makes it stand out from the crowd is its beautiful and highly expressive specification language.

http://spockframework.github.io/

Spock

```
class StringToNumberConverterSpec extends Specification {

    void "1 converts to 1"() {
        expect:
        1 == StringToNumberConverter.convertToDouble("1")
    }

    void "0.5 converts to 0.5"() {
        expect:
        0.5 == StringToNumberConverter.convertToDouble("0.5")
    }
```

```
void "item is parsed correctly"() {
    given:
    def item = [ParameterName: "hémoglobine", ValueChar: "13,6", Unit: "g/100ml", Range: "12-16g/100ml",
                LRange: "12", URange: "16"]

    when:
    def parsedItem = parser.parse(item)

    then:
    item.ParameterName == parsedItem.analysisName
    item.ValueChar == parsedItem.analysisValue
}
```

Where we are now

Building tests with Grails, Groovy and Spock

Using tests for analysis

Using tests for software design

Using tests for checking definition of done

3. The power of tests in practice

Search patients

3.1 Analysis

Initial UI

PATIENTS

Sélection par MT

NOM	PRÉNOM	NE(É)	ADRESSE		TEL

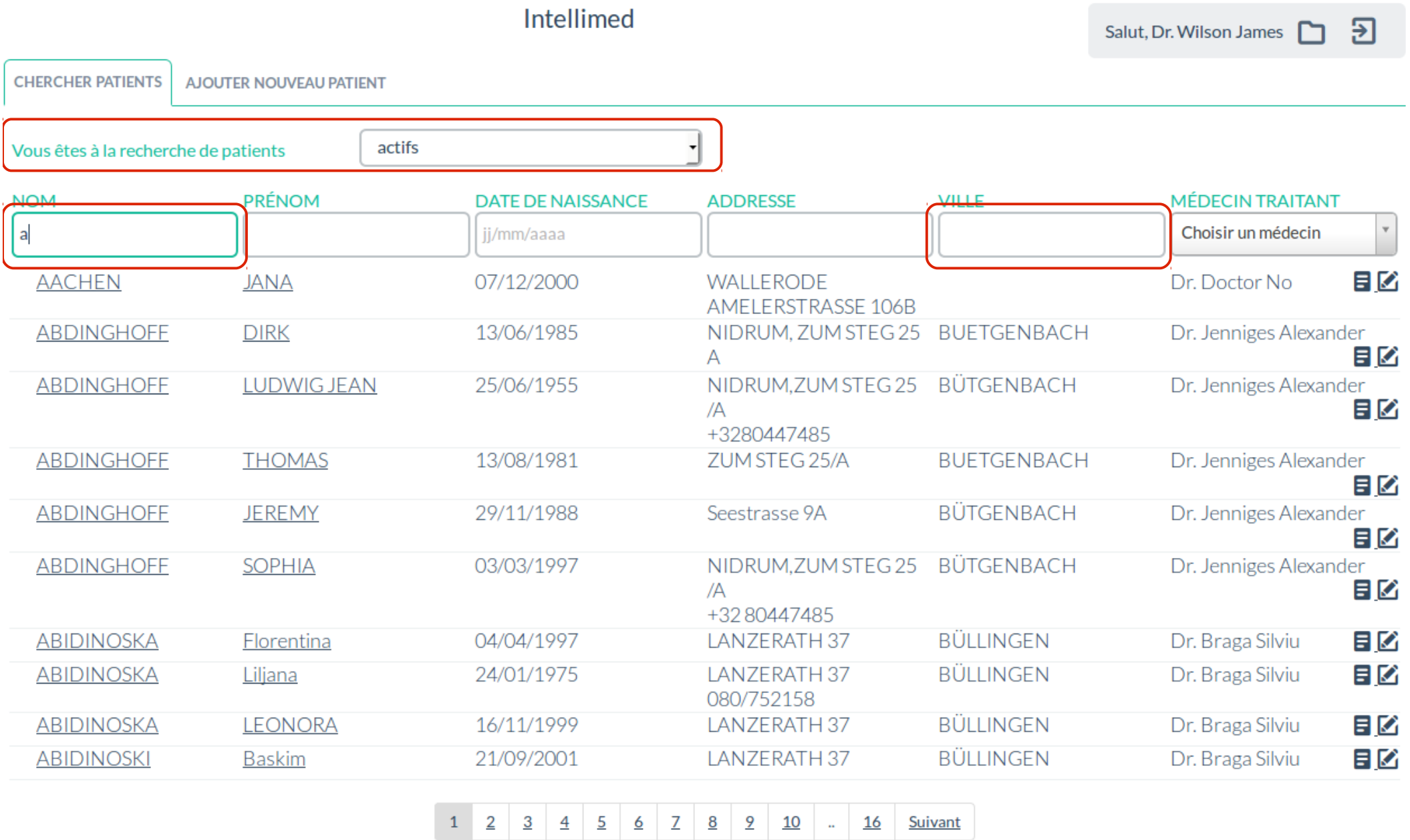
Front-end unit tests

```
describe("select patient is initialized when ", function(){
    it("searching patient by name or address is possible by just typing the search terms", function(){
        spyOn(searchAsYouTypeInput, "activate")

        selectPatient.init()

        expect(searchAsYouTypeInput.activate).
            toHaveBeenCalledWith(selectPatientUIElements.searchPatientByNameBirthDateAndAddress,
            selectPatientUIElements.searchTerm, searchPatientForSelectUrl)
    })

    it("searching the patient by birth date is possible on change date submit", function(){
        spyOn(submitOnChangeControl, "activate")

        selectPatient.init()

        expect(submitOnChangeControl.activate).
            toHaveBeenCalledWith(selectPatientUIElements.searchPatientByNameBirthDateAndAddress,
            selectPatientUIElements.birthDateSearchTerm, searchPatientForSelectUrl)
    })

    it("searching the patient by doctor is possible on change doctor submit", function(){
        spyOn(submitOnChangeControl, "activate")

        selectPatient.init()

        expect(submitOnChangeControl.activate).
            toHaveBeenCalledWith(selectPatientUIElements.searchPatientByNameBirthDateAndAddress,
            selectPatientUIElements.doctorSelect, searchPatientForSelectUrl)
    })
```

Acceptance tests

```
void "doctor sees matching patients by partial last name among all active and internal patients created inside his association"() {
    given:
    def lastNameSearchTerm = "d"

    when:
    doctorEntersTextInPatientLastNameSearchInput(lastNameSearchTerm)

    then:
    assertAllPatientsHaveMatchingLastName()
    assertAllPatientsHaveBeenCreatedInsideCurrentAssociation()
    assertAllPatientsAreActiveAndInternal()
}

void "doctor sees matching patients by partial first name among all active and internal patients created inside his association"() {
    given:
    def firstNameSearchTerm = "j"

    when:
    doctorEntersTextInPatientFirstNameSearchInput(firstNameSearchTerm)

    then:
    assertAllPatientsHaveMatchingFirstName()
    assertAllPatientsHaveBeenCreatedInsideCurrentAssociation()
    assertAllPatientsAreActiveAndInternal()
}

void "doctor sees matching patients by birthDate among all active and internal patients created inside his association"() {
    given:
    def doctorMatchingPatient = firstPatientFromDoctorPatientsList()
    def birthDate = doctorMatchingPatient.birthDate

    when:
    doctorSelectsBirthDateForSearchPatient(birthDate)

    then:
    assertAllPatientsHaveMatchingBirthDate(birthDate)
    assertAllPatientsHaveBeenCreatedInsideCurrentAssociation()
    assertAllPatientsAreActiveAndInternal()
}
```

Final UI


Intellimed
Salut, Dr. Wilson James
CHERCHER PATIENTS
AJOUTER NOUVEAU PATIENT
Vous êtes à la recherche de patients
actifs
NOM
PRÉNOM
DATE DE NAISSANCE
ADDRESSE
VILLE
MÉDECIN TRAITANT
a
jj/mm/aaaa
Choisir un médecin
AACHEN JANA 07/12/2000 WALLERODE AMELERSTRASSE 106B Dr. Doctor No
ABDINGHOFF DIRK 13/06/1985 NIDRUM, ZUM STEG 25 A BUETGENBACH Dr. Jenniges Alexander
ABDINGHOFF LUDWIG JEAN 25/06/1955 NIDRUM,ZUM STEG 25 /A +3280447485 BÜTGENBACH Dr. Jenniges Alexander
ABDINGHOFF THOMAS 13/08/1981 ZUM STEG 25/A BUETGENBACH Dr. Jenniges Alexander
ABDINGHOFF JEREMY 29/11/1988 Seestrasse 9A BÜTGENBACH Dr. Jenniges Alexander
ABDINGHOFF SOPHIA 03/03/1997 NIDRUM,ZUM STEG 25 /A +32 80447485 BÜTGENBACH Dr. Jenniges Alexander
ABIDINOSKA Florentina 04/04/1997 LANZERATH 37 BÜLLINGEN Dr. Braga Silviu
ABIDINOSKA Liljana 24/01/1975 LANZERATH 37 080/752158 BÜLLINGEN Dr. Braga Silviu
ABIDINOSKA LEONORA 16/11/1999 LANZERATH 37 BÜLLINGEN Dr. Braga Silviu
ABIDINOSKI Baskim 21/09/2001 LANZERATH 37 BÜLLINGEN Dr. Braga Silviu
1 2 3 4 5 6 7 8 9 10 .. 16 Suivant


3.2 Software Design


3.2 Software Design

Front-end unit tests

```
describe("select patient is initialized when ", function(){
    it("searching patient by name or address is possible by just typing the search terms", function(){
        spyOn(searchAsYouTypeInput, "activate")

        selectPatient.init()

        expect(searchAsYouTypeInput.activate).
            toHaveBeenCalledWith(selectPatientUIElements.searchPatientByNameBirthDateAndAddress,
            selectPatientUIElements.searchTerm, searchPatientForSelectUrl)
    })

    it("searching the patient by birth date is possible on change date submit", function(){
        spyOn(submitOnChangeControl, "activate")

        selectPatient.init()

        expect(submitOnChangeControl.activate).
            toHaveBeenCalledWith(selectPatientUIElements.searchPatientByNameBirthDateAndAddress,
            selectPatientUIElements.birthDateSearchTerm, searchPatientForSelectUrl)
    })

    it("searching the patient by doctor is possible on change doctor submit", function(){
        spyOn(submitOnChangeControl, "activate")

        selectPatient.init()

        expect(submitOnChangeControl.activate).
            toHaveBeenCalledWith(selectPatientUIElements.searchPatientByNameBirthDateAndAddress,
            selectPatientUIElements.doctorSelect, searchPatientForSelectUrl)
    })
```

Controller Unit tests

```groovy
void "test listMatchingPatientsPage calls searchPatients service with correct search params and pageParams when patients"() {
    given: "params for searching patients received"
    def expectedSearchParams = searchPatientParamsReceived()
    def expectedPageParams = [max: 2, offset: 0]
    controller.paramsConverter.toPageParams(_) >> expectedPageParams

    when: "user searches among all active and internal patients"
    controller.listMatchingPatientsPage()

    then: "searchPatients services is called with received search params for active and internal patients"
    1 * controller.searchPatientService.searchPatients(_) >>  { args ->
        assert expectedSearchParams == args[0]
        assert expectedPageParams == args[1]
        return []
    }

    when: "user searches among all archived patients"
    receiveSearchPatientTypeRequestParam(PatientType.archived, expectedSearchParams)
    controller.listMatchingPatientsPage()

    then: "searchPatients services is called with received search params for archived patients"
    1 * controller.searchPatientService.searchPatients(_) >>  { args ->
        assert expectedSearchParams == args[0]
        assert expectedPageParams == args[1]
        return []
    }

    when: "user searches among all deceased patients"
    receiveSearchPatientTypeRequestParam(PatientType.deceased, expectedSearchParams)
    controller.listMatchingPatientsPage()

    then: "searchPatients services is called with received search params for deceased patients"
    1 * controller.searchPatientService.searchPatients(_) >>  { args ->
        assert expectedSearchParams == args[0]
        assert expectedPageParams == args[1]
        return []
    }
```

Back-end Unit tests

```
void "test searchPatients searches patients by received search params"() {
    given:
    def searchParams = searchPatientsParams()
    def currentAssociation = stubGetCurrentAssociation()
    def patientQueryBuilderInstance = stubPatientQueryBuilder()
    def expectedPatientQuery = Mock(DetachedCriteria)
    def pageParams = pageParams()

    when:
    service.searchPatients(searchParams, pageParams)

    then:
    1 * patientQueryBuilderInstance.addFilterByActivityStatus(searchParams.activityStatus) >> patientQueryBuilderInstance
    1 * patientQueryBuilderInstance.addFilterByRegisteredToGPStatus(searchParams.registeredToGPStatus) >> patientQueryBuilderInstance
    1 * patientQueryBuilderInstance.addFilterByPartialLastName(searchParams.lastName) >> patientQueryBuilderInstance
    1 * patientQueryBuilderInstance.addFilterByPartialFirstName(searchParams.firstName) >> patientQueryBuilderInstance
    1 * patientQueryBuilderInstance.addFilterByBirthDate(searchParams.birthDate) >> patientQueryBuilderInstance
    1 * patientQueryBuilderInstance.addFilterByPartialStreetAddress(searchParams.streetAddress) >> patientQueryBuilderInstance
    1 * patientQueryBuilderInstance.addFilterByPartialCityAddress(searchParams.cityAddress) >> patientQueryBuilderInstance
    1 * patientQueryBuilderInstance.addFilterByCreatedInAssociation(currentAssociation) >> patientQueryBuilderInstance
    1 * patientQueryBuilderInstance.addFilterByPersonalGP(searchParams.doctor) >> patientQueryBuilderInstance
    1 * patientQueryBuilderInstance.build() >> expectedPatientQuery
    1 * expectedPatientQuery.list(_ ) >> {args ->
        assert pageParams.max == args[0].max
        assert pageParams.offset == args[0].offset
        assert "lastName" == args[0].sort
    }
}
```

3.3 Checking definition of done

Running all the tests

Grails test-app unit: → running all unit tests

Grails test-app integration: → running all integration tests

Grails test-app acceptance: → running all acceptance tests

Karma start → running all jasmine unit tests

Running all the tests

DoctorSearchesPatientBeforeOpeningHisFileSpec

Executed 12 tests without a single error or failure!

✔ **doctor sees matching patients by partial last name among all active and internal patients created inside his association**

Executed in 3.615 seconds.

System output

```
    - Doctor 'gaaron' logged in
    - Doctor is on home page
    - Doctor enters 'd' as search term for patient last name
    => Doctor sees patient 'John Doe, M, 09/07/1977' which has lastName starting with 'd'
    => Doctor sees patient 'JohnFromDrGary Doe, M, 09/07/1977' which has lastName starting with
'd'
    => Doctor sees patient 'John DoeNoDoctor, M, 09/07/1978' which has lastName starting with 'd'
    => Doctor sees patient 'John Doe, M, 09/07/1977' which has been created for his association
'EifelArzt'
    => Doctor sees patient 'JohnFromDrGary Doe, M, 09/07/1977' which has been created for his
association 'EifelArzt'
    => Doctor sees patient 'John DoeNoDoctor, M, 09/07/1978' which has been created for his
association 'EifelArzt'
    => Doctor sees only internal and active patients
(Doctor 'gaaron' logged out)
```

✔ **doctor sees matching patients by partial first name among all active and internal patients created inside his association**

Executed in 0.197 seconds.

System output

```
    - Doctor 'gaaron' logged in
    - Doctor is on home page
    - Doctor enters 'j' as search term for patient first name
    => Doctor sees patient 'John Doe, M, 09/07/1977' which has firstName starting with 'j'
    => Doctor sees patient 'JohnFromDrGary Doe, M, 09/07/1977' which has firstName starting with
'j'
    => Doctor sees patient 'John DoeNoDoctor, M, 09/07/1978' which has firstName starting with
'j'
    => Doctor sees patient 'John Doe, M, 09/07/1977' which has been created for his association
'EifelArzt'
    => Doctor sees patient 'JohnFromDrGary Doe, M, 09/07/1977' which has been created for his
association 'EifelArzt'
    => Doctor sees patient 'John DoeNoDoctor, M, 09/07/1978' which has been created for his
association 'EifelArzt'
    => Doctor sees only internal and active patients
(Doctor 'gaaron' logged out)
```


3.4 Demo

Acceptance tests report

DoctorSearchesPatientBeforeOpeningHisFileSpec

Executed 12 tests without a single error or failure!

✔ doctor sees matching patients by partial last name among all active and internal patients created inside his association

Executed in 3.615 seconds.

System output

```
    - Doctor 'gaaron' logged in
    - Doctor is on home page
    - Doctor enters 'd' as search term for patient last name
    => Doctor sees patient 'John Doe, M, 09/07/1977' which has lastName starting with 'd'
    => Doctor sees patient 'JohnFromDrGary Doe, M, 09/07/1977' which has lastName starting with
'd'
    => Doctor sees patient 'John DoeNoDoctor, M, 09/07/1978' which has lastName starting with 'd'
    => Doctor sees patient 'John Doe, M, 09/07/1977' which has been created for his association
'EifelArzt'
    => Doctor sees patient 'JohnFromDrGary Doe, M, 09/07/1977' which has been created for his
association 'EifelArzt'
    => Doctor sees patient 'John DoeNoDoctor, M, 09/07/1978' which has been created for his
association 'EifelArzt'
    => Doctor sees only internal and active patients
(Doctor 'gaaron' logged out)
```

✔ doctor sees matching patients by partial first name among all active and internal patients created inside his association

Executed in 0.197 seconds.

System output

```
    - Doctor 'gaaron' logged in
    - Doctor is on home page
    - Doctor enters 'j' as search term for patient first name
    => Doctor sees patient 'John Doe, M, 09/07/1977' which has firstName starting with 'j'
    => Doctor sees patient 'JohnFromDrGary Doe, M, 09/07/1977' which has firstName starting with
'j'
    => Doctor sees patient 'John DoeNoDoctor, M, 09/07/1978' which has firstName starting with
'j'
    => Doctor sees patient 'John Doe, M, 09/07/1977' which has been created for his association
'EifelArzt'
    => Doctor sees patient 'JohnFromDrGary Doe, M, 09/07/1977' which has been created for his
association 'EifelArzt'
    => Doctor sees patient 'John DoeNoDoctor, M, 09/07/1978' which has been created for his
association 'EifelArzt'
    => Doctor sees only internal and active patients
(Doctor 'gaaron' logged out)
```

And some manual tests

Results

- Happy customer
- Improved collaboration
- Maximize the work not done
- Faster development life cycle
- Happy me

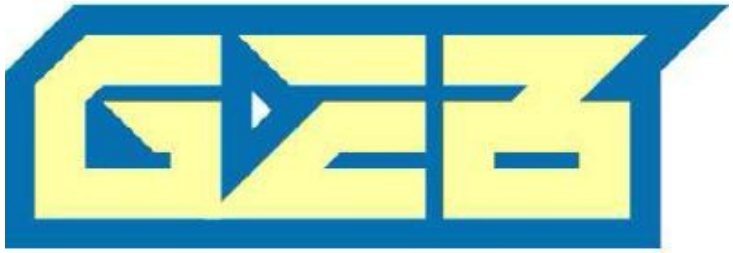

5. Next

Functional tests for acceptance

Functional testing answers questions like:

"can the user do this"

"does this particular feature work"



GEB = Browser automation tool.

WebDriver + jQuery + Page Object + Groovy

=

Easy to write & read functional tests

Functional test example

```
def setup() {
    given:"doctor is on home page after login"
    to LoginPage
    username = correctUsername
    password = correctPassword

    when: signin.click()

    then: at HomePage
}

def "doctor searches patient by first letter of last name"() {
    when:"doctor enters 'd' as first letter of the last name"
    searchPatientByNameBirthDateAndAddress.find("input", name: "lastName") << 'd'
    println("Search patients started at ${DateUtils.currentTime()}")

    then:"all patients with last name starting with 'd' letter are displayed"
    patientRow.size() > 0
    println("Search patients ended at ${DateUtils.currentTime()}")
}

def "doctor searches patient by first letter of first name"() {
    when:"doctor enters 'd' as first letter of the first name"
    searchPatientByNameBirthDateAndAddress.find("input", name: "firstName") << 'd'
    println("Search patients started at ${DateUtils.currentTime()}")

    then:"all patients with last name starting with 'd' letter are displayed"
    patientRow.size() > 0
    println("Search patients ended at ${DateUtils.currentTime()}")
}
```


5. Core ideas

4 Core Ideas

1. Best prevention of undesired side effects
2. Best analysis tool I have ever used
3. Best and fastest feedback I have received
4. Best software design tool

Dev

=

+

Test

Your Questions?

Mozaic Works

Think. Design. Work smart.

Claudia.rosu@mozaicworks.com
@claudia_rosu

Resources

http://www.groovy-lang.org/

https://grails.org/

http://spockframework.github.io/spock/docs/1.0/index.html

http://www.gebish.org/

http://mozaicworks.com/category/blog/testing/