16/04/2024

REPORTE LINT

Grupo: C1.39

Repositorio: https://github.com/pabalcber/C1.039-Acme-SF

Integrantes:

Nombre	Apellidos	Correo Corporativo
Pablo	Alcántara Bernal	pabalcber@alum.us.es
María del Mar	Ávila Maqueda	maravimaq@alum.us.es
María	Barrancos Márquez	marbarmar16@alum.us.es
Sheng	Chen	sheche1@alum.us.es
Jun	Yao	junyao@alum.us.es

Tabla de versiones:

Fecha	Versión	Descripción de los cambios	Sprint
16/04/2024	1.0	Creación del documento y redacción sus campos	3
25/04/2024	1.1	Revision final	3

1.Índice

2. Resumen Ejecutivo

Este documento detalla los errores detectados por el análisis llevado a cabo por SonarLint, sus localizaciones y cómo se pueden corregir.

3.Tabla de revisiones

Número de revisión	Fecha	Descripción
1	25/04/2024	Revisión final antes de la entrega

4. Introducción

En este documento se mostrarán los errores agrupados por impacto detectados por SonarLint, en qué consisten y cuál es la forma en la que se han solucionado del código realizado por el estudiante 2.

5.Impacto Alto

Resource	Date	Description
		High (100 of 2093 items)
AuthenticatedClientCreateService.java	12 minutes ago	Define a constant instead of duplicating this literal "identification" 4 times. [+4 locations]
ClientContractCreateService.java	10 minutes ago	Define a constant instead of duplicating this literal "budget" 7 times. [+7 locations]
ClientContractCreateService.java	10 minutes ago	Refactor this method to reduce its Cognitive Complexity from 16 to the 15 allowed. [+7 locations]
ClientContractPublishService.java	10 minutes ago	Define a constant instead of duplicating this literal "budget" 6 times. [+6 locations]
ClientContractUpdateService.java	10 minutes ago	Define a constant instead of duplicating this literal "budget" 6 times. [+6 locations]
ClientContractUpdateService.java	10 minutes ago	Define a constant instead of duplicating this literal "customerName" 3 times. [+3 locations]
ClientProgressLogCreateService.java	10 minutes ago	Define a constant instead of duplicating this literal "masterId" 6 times. [+6 locations]
ClientProgressLogCreateService.java	10 minutes ago	Define a constant instead of duplicating this literal "recordId" 4 times. [+4 locations]
ClientProgressLogCreateService.java	10 minutes ago	Define a constant instead of duplicating this literal "responsiblePerson" 3 times. [+3 locations]
ClientProgressLogListService.java	10 minutes ago	Define a constant instead of duplicating this literal "masterId" 4 times. [+4 locations]
ClientProgressLogUpdateService.java	10 minutes ago	Define a constant instead of duplicating this literal "responsiblePerson" 3 times. [+3 locations]

En la imagen anterior se muestran los errores de impacto alto detectados por SonarLint.

5.1. AuthenticatedClientCreateService Class

En la clase . AuthenticatedClientCreateService nos encontramos con el siguiente error:

```
@Override
public void bind(final Client object) {
    assert object != null;

    Duplication
    super.bind(object, 1 "identification", "companyName", "email", "furtherInformation", "type");
}

@Override
public void validate(final Client object) {
    assert object != null;
    Duplication
    if (!super.getBuffer().getErrors().hasErrors( 2 "identification")) {
        Client existing;

        existing = this.repository.findClientByIdentification(object.getIdentification());
        Duplication
        super.state(existing == null, 3 "identification", "authenticated.client.form.error.duplicated");
    }
}

@Override
public void perform(final Client object) {
    assert object != null;

    this.repository.save(object);
}

@Override
public void unbind(final Client object) {
    assert object != null;

    SelectChoices choices;
    Dataset dataset;

    choices = SelectChoices.from(ClientType.class, object.getType());

    Duplication
    dataset = super.unbind(object, 4 "identification", "companyName", "email", "furtherInformation", "type");
    dataset.put("types", choices);

    super.getResponse().addData(dataset);
}
```

- Descripción:

String literals should not be duplicated

Adaptability | Not distinct Maintainability

Why is this an issue? How can I fix it?

Duplicated string literals make the process of refactoring complex and error-prone, as any change would need to be propagated on all occurrences.

Exceptions

String literals should not be duplicated

Adaptability | Not distinct Maintainability

Why is this an issue? How can I fix it?

Use constants to replace the duplicated string literals. Constants can be referenced from many places, but only need to be updated in a single place.

Noncompliant code example

```
public void run() {
  prepare("action1");                 // Noncompliant - "action1" is duplicated 3 times
  execute("action1");
  release("action1");
}

@SuppressWarning("all")               // Compliant - annotations are excluded
private void method1() { /* ... */ }
@SuppressWarning("all")
private void method2() { /* ... */ }

public String printInQuotes(String a, String b) {
  return "'" + a + "' " + b + "'";    // Compliant - literal "'" has less than 5 characters and is excluded
}
```

Compliant solution

```
private static final String ACTION_1 = "action1"; // Compliant

public void run() {
  prepare(ACTION_1);                  // Compliant
  execute(ACTION_1);
  release(ACTION_1);
}
```

- Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
private static String identification = "identification";
```

```
public void bind(final Client object) {
    assert object != null;

    Duplication
    super.bind(object, AuthenticatedClientCreateService.identification, "companyName", "email", "furtherInformation", "type");
}

@Override
public void validate(final Client object) {
    assert object != null;
    Duplication
    if (!super.getBuffer().getErrors().hasErrors(AuthenticatedClientCreateService.identification)) {
        Client existing;

        existing = this.repository.findClientByIdentification(object.getIdentification());
        Duplication
        super.state(existing == null, AuthenticatedClientCreateService.identification, "authenticated.client.form.error.duplicated");
    }
}

@Override
public void perform(final Client object) {
    assert object != null;

    this.repository.save(object);
}

@Override
public void unbind(final Client object) {
    assert object != null;

    SelectChoices choices;
    Dataset dataset;

    choices = SelectChoices.from(ClientType.class, object.getType());

    Duplication
    dataset = super.unbind(object, AuthenticatedClientCreateService.identification, "companyName", "email", "furtherInformation", "type");
    dataset.put("types", choices);

    super.getResponse().addData(dataset);
}
```

5.2. ClientContractCreateService Class

En la clase ClientContractCreateService nos encontramos con los siguientes errores:

```
    Duplication
    super.bind(object, "code", "instantiationMoment", "providerName", "customerName", "goals", 1 "budget");
    object.setProject(project);
}

@Override
public void validate(final Contract object) {
    assert object != null;

    if (!super.getBuffer().getErrors().hasErrors("code")) {
        Contract existing;

        existing = this.repository.findOneContractByCode(object.getCode());
        super.state(existing == null, "code", "client.contract.form.error.duplicated");
    }

    Duplication
    if (!super.getBuffer().getErrors().hasErrors( 2 "budget"))
        if (object.getBudget() != null) {
            Money budget = object.getBudget();
            Project project = object.getProject();

            Duplication
            super.state(budget.getAmount() >= 0, 3 "budget", "client.contract.form.error.negative-budget");

            if (project != null) {
                Money projectCost = project.getCost();

                if (!budget.getCurrency().equals(projectCost.getCurrency()))
                    Duplication
                    super.state(false, 4 "budget", "client.contract.form.error.different-currency");

                if (budget.getAmount() > projectCost.getAmount())
                    Duplication
                    super.state(false, 5 "budget", "client.contract.form.error.budget-exceeds-project-cost");
            }
        } else
            Duplication
            super.state(false, 6 "budget", "client.contract.form.error.budget-cannot-be-null");
}
```

```
   +1
1  if (!super.getBuffer().getErrors().hasErrors("code")) {
    Contract existing;

    existing = this.repository.findOneContractByCode(object.getCode());
    super.state(existing == null, "code", "client.contract.form.error.duplicated");
}

   +1
2  if (!super.getBuffer().getErrors().hasErrors("budget"))
       +2 (incl 1 for nesting)
3      if (object.getBudget() != null) {
        Money budget = object.getBudget();
        Project project = object.getProject();


        super.state(budget.getAmount() >= 0, "budget", "client.contract.form.error.negative-budget");

           +3 (incl 2 for nesting)
4       if (project != null) {
            Money projectCost = project.getCost();

               +4 (incl 3 for nesting)
5           if (!budget.getCurrency().equals(projectCost.getCurrency()))

                super.state(false, "budget", "client.contract.form.error.different-currency");

               +4 (incl 3 for nesting)
6           if (budget.getAmount() > projectCost.getAmount())

                super.state(false, "budget", "client.contract.form.error.budget-exceeds-project-cost");
        }
    +1
    }  7 else

        super.state(false, "budget", "client.contract.form.error.budget-cannot-be-null");
```

- Descripción:

Problema 1: Igual que el de la clase AuthenticatedClientCreateService.

Problema 2:

Cognitive Complexity of methods should not be too high

Adaptability | Not focused Maintainability

This rule raises an issue when the code cognitive complexity of a function is above a certain threshold.

Why is this an issue? How can I fix it? More Info

Cognitive Complexity is a measure of how hard it is to understand the control flow of a unit of code. Code with high cognitive complexity is hard to read, understand, test, and modify.

As a rule of thumb, high cognitive complexity is a sign that the code should be refactored into smaller, easier-to-manage pieces.

Which syntax in code does impact cognitive complexity score?

Here are the core concepts:

- **Cognitive complexity is incremented each time the code breaks the normal linear reading flow.**
  This concerns, for example, loop structures, conditionals, catches, switches, jumps to labels, and conditions mixing multiple operators.
- **Each nesting level increases complexity.**
  During code reading, the deeper you go through nested layers, the harder it becomes to keep the context in mind.
- **Method calls are free**
  A well-picked method name is a summary of multiple lines of code. A reader can first explore a high-level view of what the code is performing then go deeper and deeper by looking at called functions content.
  Note: This does not apply to recursive calls, those will increment cognitive score.

Reducing cognitive complexity can be challenging.
Here are a few suggestions:

- **Extract complex conditions in a new function.**
  Mixed operators in condition will increase complexity. Extracting the condition in a new function with an appropriate name will reduce cognitive load.
- **Break down large functions.**
  Large functions can be hard to understand and maintain. If a function is doing too many things, consider breaking it down into smaller, more manageable functions. Each function should have a single responsibility.
- **Avoid deep nesting by returning early.**
  To avoid the nesting of conditions, process exceptional cases first and return early.

Extraction of a complex condition in a new function.

```
double calculateFinalPrice(User user, Cart cart) {
  double total = calculateTotal(cart);
  if (user.hasMembership()                    // +1 (if)
    && user.ordersCount() > 10                // +1 (more than one condition)
    && user.isAccountActive()
    && !user.hasDiscount()
    || user.ordersCount() == 1) {             // +1 (change of operator in condition)
    total = applyDiscount(user, total);
  }
  return total;
}
```

Compliant solution

Even if the cognitive complexity of the whole program did not change, it is easier for a reader to understand the code of the `calculateFinalPrice` function, which now only has a cognitive cost of 1

```
double calculateFinalPrice(User user, Cart cart) {
  double total = calculateTotal(cart);
  if (isEligibleForDiscount(user)) {          // +1 (if)
    total = applyDiscount(user, total);
  }
  return total;
}

boolean isEligibleForDiscount(User user) {
  return user.hasMembership()
    && user.ordersCount() > 10                // +1 (more than one condition)
    && user.isAccountActive()
    && !user.hasDiscount()
    || user.ordersCount() == 1;               // +1 (change of operator in condition)
}
```

Break down large functions.

Noncompliant code example

For example, consider a function that calculates the total price of a shopping cart, including sales tax and shipping.

```
double calculateTotal(Cart cart) {
  double total = 0;
  for (Item item : cart.items()) {      // +1 (for)
    total += item.price;
  }

  // calculateSalesTax
  for (Item item : cart.items()) {      // +1 (for)
    total += 0.2 * item.price;
  }

  //calculateShipping
  total += 5 * cart.items().size();

  return total;
}
```

This function could be refactored into smaller functions: The complexity is spread over multiple functions and the complex `calculateTotal` has now a complexity score of zero.

Compliant solution

```
double calculateTotal(Cart cart) {
  double total = 0;
  total = calculateSubtotal(cart, total);
  total += calculateSalesTax(cart, total);
  total += calculateShipping(cart, total);

  return total;
}

double calculateShipping(Cart cart, double total) {
```

```
double calculateShipping(Cart cart, double total) {
  total += 5 * cart.items().size();
  return total;
}

double calculateSalesTax(Cart cart, double total) {
  for (Item item : cart.items()) {      // +1 (for)
    total += 0.2 * item.price;
  }
  return total;
}

double calculateSubtotal(Cart cart, double total) {
  for (Item item : cart.items()) {      // +1 (for)
    total += item.price;
  }
  return total;
}
```

Avoid deep nesting by returning early.

Noncompliant code example

```
double calculateDiscount(double price, User user) {
  if (isEligibleForDiscount(user)) {     // +1 ( if )
    if (user.hasMembership()) {          // +2 ( nested if )
      return price * 0.9;
    } else if (user.ordersCount() == 1) { // +1 ( else )
      return price * 0.95;
    } else {                             // +1 ( else )
      return price;
    }
  } else {                               // +1 ( else )
    return price;
  }
}
```

Compliant solution

Avoid deep nesting by returning early.

Noncompliant code example

```
double calculateDiscount(double price, User user) {
  if (isEligibleForDiscount(user)) {     // +1 ( if )
    if (user.hasMembership()) {          // +2 ( nested if )
      return price * 0.9;
    } else if (user.ordersCount() == 1) { // +1 ( else )
      return price * 0.95;
    } else {                             // +1 ( else )
      return price;
    }
  } else {                               // +1 ( else )
    return price;
  }
}
```

Compliant solution

Checking for the edge case first flattens the `if` statements and reduces the cognitive complexity to 2

```
double calculateDiscount(double price, User user) {
  if (!isEligibleForDiscount(user)) {    // +1 ( if )
    return price;
  }
  if (user.hasMembership()) {            // +1
    return price * 0.9;
  }
  if (user.ordersCount() == 1) {         // +1 ( if )
    return price * 0.95;
  }
  return price;
}
```

- ## Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
private static String budget = "budget";
```

```
    super.bind(object, "code", "instantiationMoment", "providerName", "customerName", "goals", ClientContractCreateService.budget);
    object.setProject(project);
}
    clientId = super.getRequest().getPrincipal().getActiveRoleId();
    projects = this.repository.findManyProjectsByClientId(clientId);
    choices = SelectChoices.from(projects, "code", object.getProject());

    Duplication
    dataset = super.unbind(object, "code", "instantiationMoment", "providerName", "customerName", "goals", ClientContractCreateService.budget, "draftMode");
    dataset.put("project", choices.getSelected().getKey());
    dataset.put("projects", choices);

    super.getResponse().addData(dataset);
}

// Ancillary methods ------------------------------------------------------

private void validateUniqueCode(final Contract object) {
    if (!super.getBuffer().getErrors().hasErrors("code")) {
        Contract existing = this.repository.findOneContractByCode(object.getCode());
        super.state(existing == null, "code", "client.contract.form.error.duplicated");
    }
}

private void validateBudget(final Contract object) {
    if (!super.getBuffer().getErrors().hasErrors(ClientContractCreateService.budget)) {
        Money b = object.getBudget();
        Project project = object.getProject();

        if (b == null) {
            super.state(false, ClientContractCreateService.budget, "client.contract.form.error.budget-cannot-be-null");
            return;
        }

        super.state(b.getAmount() >= 0, ClientContractCreateService.budget, "client.contract.form.error.negative-budget");

        if (project != null) {
            Money projectCost = project.getCost();

            if (!b.getCurrency().equals(projectCost.getCurrency()))
                super.state(false, ClientContractCreateService.budget, "client.contract.form.error.different-currency");

            if (b.getAmount() > projectCost.getAmount())
                super.state(false, ClientContractCreateService.budget, "client.contract.form.error.budget-exceeds-project-cost");
        }
    `
```

5.3. ClientContractPublishService Class

En la clase . ClientContractPublishService

Class nos encontramos con el siguiente error:

```
        Duplication
        super.bind(object, "code", "instantiationMoment", "providerName", "customerName", "goals", 1 "budget");
        object.setProject(project);
    }

    @Override
    public void validate(final Contract object) {
        assert object != null;

        if (!super.getBuffer().getErrors().hasErrors("code")) {
            Contract existing;

            existing = this.repository.findOneContractByCode(object.getCode());
            super.state(existing == null || existing.equals(object), "code", "client.contract.form.error.duplicated");
        }

        Duplication
        if (!super.getBuffer().getErrors().hasErrors( 2 "budget")) {
            Money budget = object.getBudget();
            Project project = object.getProject();

            Duplication
            super.state(budget.getAmount() >= 0, 3 "budget", "client.contract.form.error.negative-budget");

            if (project != null) {
                Money projectCost = project.getCost();

                if (!budget.getCurrency().equals(projectCost.getCurrency()))
                    Duplication
                    super.state(false, 4 "budget", "client.contract.form.error.different-currency");

                if (budget.getAmount() > projectCost.getAmount())
                    Duplication
                    super.state(false, 5 "budget", "client.contract.form.error.budget-exceeds-project-cost");

                Double existingCombinedBudget = this.repository.combinedBudgetByContract(project.getId());
                double totalCombinedBudget = (existingCombinedBudget != null ? existingCombinedBudget : 0.0) + budget.getAmount();
                double projectTotalCost = projectCost.getAmount();
```

- ## Descripción:

Igual que el de la clase AuthenticatedClientCreateService.

- ## Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
    private static String               budget  = "budget";

    Duplication
    super.bind(object, "code", "instantiationMoment", "providerName", "customerName", "goals", ClientContractPublishService.budget);
    object.setProject(project);
}

@Override
public void validate(final Contract object) {
    assert object != null;

    if (!super.getBuffer().getErrors().hasErrors("code")) {
        Contract existing;

        existing = this.repository.findOneContractByCode(object.getCode());
        super.state(existing == null || existing.equals(object), "code", "client.contract.form.error.duplicated");
    }

    Duplication
    if (!super.getBuffer().getErrors().hasErrors(ClientContractPublishService.budget)) {
        Money budgt = object.getBudget();
        Project project = object.getProject();

        Duplication
        super.state(budgt.getAmount() >= 0, ClientContractPublishService.budget, "client.contract.form.error.negative-budget");

        if (project != null) {
            Money projectCost = project.getCost();

            if (!budgt.getCurrency().equals(projectCost.getCurrency()))
                Duplication
                super.state(false, ClientContractPublishService.budget, "client.contract.form.error.different-currency");

            if (budgt.getAmount() > projectCost.getAmount())
                Duplication
                super.state(false, ClientContractPublishService.budget, "client.contract.form.error.budget-exceeds-project-cost");
```

5.4. ClientContractUpdateService Class

En la clase ClientContractUpdateService nos encontramos con los siguientes errores:

```
        projectId = super.getRequest().getData("project", int.class);
        project = this.repository.findOneProjectById(projectId);

        Duplication
        super.bind(object, "code", "instantiationMoment", "providerName", "customerName", "goals", 1 "budget");
        object.setProject(project);
    }

    @Override
    public void validate(final Contract object) {
        assert object != null;

        if (!super.getBuffer().getErrors().hasErrors("code")) {
            Contract existing;

            existing = this.repository.findOneContractByCode(object.getCode());
            super.state(existing == null || existing.equals(object), "code", "client.contract.form.error.duplicated"
        }

        Duplication
        if (!super.getBuffer().getErrors().hasErrors( 2 "budget")) {
            Money budget = object.getBudget();
            Project project = object.getProject();

            Duplication
            super.state(budget.getAmount() >= 0, 3 "budget", "client.contract.form.error.negative-budget");

            if (project != null) {
                Money projectCost = project.getCost();

                if (!budget.getCurrency().equals(projectCost.getCurrency()))
                    Duplication
                    super.state(false, 4 "budget", "client.contract.form.error.different-currency");

                if (budget.getAmount() > projectCost.getAmount())
                    Duplication
                    super.state(false, 5 "budget", "client.contract.form.error.budget-exceeds-project-cost");
            }
```

```
    Duplication
    super.bind(object, "code", "instantiationMoment", "providerName", 1 "customerName", "goals", "budget");
    object.setProject(project);
}
```

```
    Duplication
    dataset = super.unbind(object, "code", "instantiationMoment", "providerName", 2 "customerName", "goals", "budget", "draftMode");
    dataset.put("project", choices.getSelected().getKey());
    dataset.put("projects", choices);
    Duplication
    dataset.put( 3 "customerName", client.getIdentification());
```

- Descripción:

Problema 1: Igual que el de la clase AuthenticatedClientCreateService.

Problema 2: Igual que el de la clase AuthenticatedClientCreateService.

- **Solución:**

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
    super.bind(object, "code", "instantiationMoment", "providerName", ClientContractUpdateService.customerName, "goals", ClientContractUpdateService.budget);
    object.setProject(project);
}

@Override
public void validate(final Contract object) {
    assert object != null;

    if (!super.getBuffer().getErrors().hasErrors("code")) {
        Contract existing;

        existing = this.repository.findOneContractByCode(object.getCode());
        super.state(existing == null || existing.equals(object), "code", "client.contract.form.error.duplicated");
    }

    Duplication
    if (!super.getBuffer().getErrors().hasErrors(ClientContractUpdateService.budget)) {
        Money budgt = object.getBudget();
        Project project = object.getProject();

        Duplication
        super.state(budgt.getAmount() >= 0, ClientContractUpdateService.budget, "client.contract.form.error.negative-budget");

        if (project != null) {
            Money projectCost = project.getCost();

            if (!budgt.getCurrency().equals(projectCost.getCurrency()))
                Duplication
                super.state(false, ClientContractUpdateService.budget, "client.contract.form.error.different-currency");

            if (budgt.getAmount() > projectCost.getAmount())
                Duplication
                super.state(false, ClientContractUpdateService.budget, "client.contract.form.error.budget-exceeds-project-cost");
        }

Duplication
dataset = super.unbind(object, "code", "instantiationMoment", "providerName", ClientContractUpdateService.customerName, "goals", ClientContractUpdateService.budget, "draftMode"
dataset.put("project", choices.getSelected().getKey());
dataset.put("projects", choices);
dataset.put(ClientContractUpdateService.customerName, client.getIdentification());

super.getResponse().addData(dataset);
```

5.5 ClientProgressLogCreateService Class

En la clase ClientProgressLogCreateService nos encontramos con los siguientes errores:

```
Duplication
masterId = super.getRequest().getData( 1 "masterId", int.class);
contract = this.repository.findOneContractById(masterId);
status = contract != null && (!contract.isDraftMode() || super.getRequest(

super.getResponse().setAuthorised(status);
```

```
masterId = super.getRequest().getData( 2 "masterId", int.class);
contract = this.repository.findOneContractById(masterId);
client = contract.getClient().getIdentification();
```

```
Duplication
masterId = super.getRequest().getData( 3 "masterId", int.class);
moment = MomentHelper.getCurrentMoment();
contract = this.repository.findOneContractById(masterId);

masterId = super.getRequest().getData( 4 "masterId", int.class);
contract = this.repository.findOneContractById(masterId);
client = contract.getClient().getIdentification();

Duplication | Duplication
dataset.put( 5 "masterId", super.getRequest().getData( 6 "masterId", int.class));
dataset.put("draftMode", object.getContract().isDraftMode());
dataset.put("responsiblePerson", client);

Duplication
super.bind(object, 1 "recordId", "completeness", "comment", '
```

```
public void validate(final ProgressLog object) {
    assert object != null;
    Duplication
    if (!super.getBuffer().getErrors().hasErrors( 2 "recordId")) {
        ProgressLog existing;

        existing = this.repository.findOneProgressLogByRecordId(object.getRecordId());
        Duplication
        super.state(existing == null, 3 "recordId", "client.progressLog.form.error.duplicated");
    }
}

Duplication
dataset = super.unbind(object, 4 "recordId", "completeness", "cc
```

```
Duplication
super.bind(object,    "recordId", "completeness", "comment", "registrationMoment", 1 "responsiblePerson", "contract");

verride

Duplication
dataset = super.unbind(object, "recordId", "completeness", "comment", "registrationMoment", 2 "responsiblePerson",

dataset.put("masterId", super.getRequest().getData("masterId", int.class));
dataset.put("draftMode", object.getContract().isDraftMode());
Duplication
dataset.put( 3 "responsiblePerson", client);
```

- ## Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
private static String responsiblePerson = "responsiblePerson";
private static String recordId          = "recordId";
private static String id                = "masterId";
```

```
masterId = super.getRequest().getData(ClientProgressLogCreateService.id, int.class);
contract = this.repository.findOneContractById(masterId);
client = contract.getClient().getIdentification();
```

```
public void bind(final ProgressLog object) {
    assert object != null;

    super.bind(object, ClientProgressLogCreateService.recordId, "completeness", "comment", "registrationMoment", ClientProgressLogCreateService.responsiblePerson,
}

@Override
public void validate(final ProgressLog object) {
    assert object != null;
    if (!super.getBuffer().getErrors().hasErrors(ClientProgressLogCreateService.recordId)) {
        ProgressLog existing;

        existing = this.repository.findOneProgressLogByRecordId(object.getRecordId());
        super.state(existing == null, ClientProgressLogCreateService.recordId, "client.progressLog.form.error.duplicated");
    }
```

```
masterId = super.getRequest().getData(ClientProgressLogCreateService.id, int.class);
moment = MomentHelper.getCurrentMoment();
contract = this.repository.findOneContractById(masterId);
client = contract.getClient().getIdentification();
```

```
masterId = super.getRequest().getData(ClientProgressLogCreateService.id, int.class);
contract = this.repository.findOneContractById(masterId);
client = contract.getClient().getIdentification();

Dataset dataset;

dataset = super.unbind(object, ClientProgressLogCreateService.recordId, "completeness", "comment", "registrationMoment", ClientProgressLogCreateService.responsiblePerson,
Duplication | Duplication
dataset.put(ClientProgressLogCreateService.id, super.getRequest().getData(ClientProgressLogCreateService.id, int.class));
dataset.put("draftMode", object.getContract().isDraftMode());
dataset.put(ClientProgressLogCreateService.responsiblePerson, client);

super.getResponse().addData(dataset);
```

5.6 ClientProgressLogListService Class

En la clase . ClientProgressLogListService nos encontramos con el siguiente error:

```
public void authorise() {
    boolean status;
    int masterId;
    Contract contract;

    Duplication
    masterId = super.getRequest().getData( 1 "masterId", int.class);
    contract = this.repository.findOneContractById(masterId);
    status = contract != null && (!contract.isDraftMode() || super.getRequest().getPrincipal().hasRole(contract.getClient()))

    super.getResponse().setAuthorised(status);
}
```

```
)Override
ublic void load() {
    Collection<ProgressLog> objects;
    int masterId;

    Duplication
    masterId = super.getRequest().getData( 2 "masterId", int.class);
    objects = this.repository.findManyProgressLogsByMasterId(masterId);

    super.getBuffer().addData(objects);
```

```
Duplication
masterId = super.getRequest().getData( 3 "masterId", int.class);
contract = this.repository.findOneContractById(masterId);
showCreate = contract.isDraftMode() && super.getRequest().getPrincipal().hasRole(contract.getClient());

Duplication
super.getResponse().addGlobal( 4 "masterId", masterId);
super.getResponse().addGlobal("showCreate", showCreate);
```

- ## Descripción:

Igual que el de la clase AuthenticatedClientCreateService.

- ## Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
private static String              id  = "masterId";


masterId = super.getRequest().getData(ClientProgressLogListService.id, int.class);
contract = this.repository.findOneContractById(masterId);
status = contract != null && (!contract.isDraftMode() || super.getRequest().getPrincipal().hasRole(contract.getClient()));

masterId = super.getRequest().getData(ClientProgressLogListService.id, int.class);
objects = this.repository.findManyProgressLogsByMasterId(masterId);

super.getBuffer().addData(objects);

 Duplication
masterId = super.getRequest().getData(ClientProgressLogListService.id, int.class);
contract = this.repository.findOneContractById(masterId);
showCreate = contract.isDraftMode() && super.getRequest().getPrincipal().hasRole(contract.getClient());

 Duplication
super.getResponse().addGlobal(ClientProgressLogListService.id, masterId);
super.getResponse().addGlobal("showCreate", showCreate);
```

5.7 ClientProgressLogUpdateService Class

En la clase . ClientProgressLogUpdateService nos encontramos con el siguiente error:

```
@Override
public void bind(final ProgressLog object) {
    assert object != null;

    Duplication
    super.bind(object, "recordId", "completeness", "comment", "registrationMoment", 1 "responsiblePerson");
}

@Override
public void validate(final ProgressLog object) {
    assert object != null;
}

@Override
public void perform(final ProgressLog object) {
    assert object != null;

    Client client = object.getContract().getClient();

    object.setResponsiblePerson(client.getIdentification());
    this.repository.save(object);
}

@Override
public void unbind(final ProgressLog object) {
    assert object != null;

    Dataset dataset;

    Duplication
    dataset = super.unbind(object, "recordId", "completeness", "comment", "registrationMoment", 2 "responsiblePerson");
    dataset.put("masterId", object.getContract().getId());
    dataset.put("draftMode", object.getContract().isDraftMode());
    dataset.put("contract", object.getContract().getCode());
    Duplication
    dataset.put( 3 "responsiblePerson", object.getContract().getClient().getIdentification());
}
```

- ## Descripción:

Igual que el de la clase AuthenticatedClientCreateService.

- ## Solución:

Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
private static String               responsiblePerson    = "responsiblePerson";

@Override
public void bind(final ProgressLog object) {
    assert object != null;

    Duplication
    super.bind(object, "recordId", "completeness", "comment", "registrationMoment", ClientProgressLogUpdateService.responsiblePerson);
}

@Override
public void validate(final ProgressLog object) {
    assert object != null;
}

@Override
public void perform(final ProgressLog object) {
    assert object != null;

    Client client = object.getContract().getClient();

    object.setResponsiblePerson(client.getIdentification());
    this.repository.save(object);
}

@Override
public void unbind(final ProgressLog object) {
    assert object != null;

    Dataset dataset;

    Duplication
    dataset = super.unbind(object, "recordId", "completeness", "comment", "registrationMoment", ClientProgressLogUpdateService.responsiblePerson);
    dataset.put("masterId", object.getContract().getId());
    dataset.put("draftMode", object.getContract().isDraftMode());
    dataset.put("contract", object.getContract().getCode());
    Duplication
    dataset.put(ClientProgressLogUpdateService.responsiblePerson, object.getContract().getClient().getIdentification());
}
```

6.Impacto Medio

En las siguientes imágenes se muestran los errores de impacto medio detectados por SonarLint.

AuthenticatedClientController.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedClientController.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedClientCreateService.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedClientCreateService.java	4 hours ago	Replace this assert with a proper check.
AuthenticatedClientCreateService.java	4 hours ago	Replace this assert with a proper check.
AuthenticatedClientCreateService.java	4 hours ago	Replace this assert with a proper check.
AuthenticatedClientCreateService.java	4 hours ago	Replace this assert with a proper check.
AuthenticatedClientUpdateService.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedClientUpdateService.java	4 hours ago	Replace this assert with a proper check.
AuthenticatedClientUpdateService.java	4 hours ago	Replace this assert with a proper check.
AuthenticatedClientUpdateService.java	4 hours ago	Replace this assert with a proper check.
AuthenticatedClientUpdateService.java	4 hours ago	Replace this assert with a proper check.

AuthenticatedContractController.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedContractController.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedContractListAllService.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedContractListAllService.java	4 hours ago	Replace this assert with a proper check.
AuthenticatedContractShowService.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedContractShowService.java	4 hours ago	Remove this useless assignment to local variable "dataset".
AuthenticatedContractShowService.java	4 hours ago	Replace this assert with a proper check.
AuthenticatedProgressLogController.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedProgressLogController.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedProgressLogListService.java	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedProgressLogListService.java	4 hours ago	Replace this assert with a proper check.
AuthenticatedProgressLogShowService.ja	4 hours ago	Remove this field injection and use constructor injection instead.
AuthenticatedProgressLogShowService.ja	4 hours ago	Replace this assert with a proper check.

ClientClientDashboardController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientClientDashboardShowService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractCreateService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractCreateService.java	4 hours ago	Replace this assert with a proper check.
ClientContractCreateService.java	4 hours ago	Replace this assert with a proper check.
ClientContractCreateService.java	4 hours ago	Replace this assert with a proper check.
ClientContractCreateService.java	4 hours ago	Replace this assert with a proper check.
ClientContractDeleteService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractDeleteService.java	4 hours ago	Replace this assert with a proper check.
ClientContractDeleteService.java	4 hours ago	Replace this assert with a proper check.
ClientContractDeleteService.java	4 hours ago	Replace this assert with a proper check.
ClientContractDeleteService.java	4 hours ago	Replace this assert with a proper check.
ClientContractListAllService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractListAllService.java	4 hours ago	Replace this assert with a proper check.
ClientContractListService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractListService.java	4 hours ago	Replace this assert with a proper check.
ClientContractPublishService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractPublishService.java	4 hours ago	Replace this assert with a proper check.
ClientContractPublishService.java	4 hours ago	Replace this assert with a proper check.
ClientContractPublishService.java	4 hours ago	Replace this assert with a proper check.
ClientContractPublishService.java	4 hours ago	Replace this assert with a proper check.
ClientContractShowService.java	4 hours ago	Remove this expression which always evaluates to "true" [+2 locations]

ClientContractShowService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractShowService.java	4 hours ago	Replace this assert with a proper check.
ClientContractUpdateService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientContractUpdateService.java	4 hours ago	Replace this assert with a proper check.
ClientContractUpdateService.java	4 hours ago	Replace this assert with a proper check.
ClientContractUpdateService.java	4 hours ago	Replace this assert with a proper check.
ClientContractUpdateService.java	4 hours ago	Replace this assert with a proper check.
ClientProgressLogController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientProgressLogController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientProgressLogController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientProgressLogController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientProgressLogController.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientProgressLogCreateService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientProgressLogCreateService.java	4 hours ago	Replace this assert with a proper check.
ClientProgressLogCreateService.java	4 hours ago	Replace this assert with a proper check.
ClientProgressLogCreateService.java	4 hours ago	Replace this assert with a proper check.
ClientProgressLogCreateService.java	4 hours ago	Replace this assert with a proper check.
ClientProgressLogDeleteService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientProgressLogDeleteService.java	4 hours ago	Replace this assert with a proper check.
ClientProgressLogDeleteService.java	4 hours ago	Replace this assert with a proper check.
ClientProgressLogDeleteService.java	4 hours ago	Replace this assert with a proper check.
ClientProgressLogDeleteService.java	4 hours ago	Replace this assert with a proper check.
ClientProgressLogListService.java	4 hours ago	Remove this field injection and use constructor injection instead.
ClientProgressLogListService.java	4 hours ago	Replace this assert with a proper check.

6.1. ClientProgressLogShowService, ClientProgressLogListService, ClientProgressLogDeleteService, ClientProgressLogCreateService, ClientProgressLogUpdateService, ClientContractUpdateService, ClientContractShowService, ClientContractPublishService, ClientContractListService, ClientContractListAllService, ClientContractDeleteService, ClientContractCreateService, AuthenticatedClientCreateService, AuthenticatedClientUpdateService, AuthenticatedContractListAllService, AuthenticatedContractShowService, AuthenticatedProgressLogListService y AuthenticatedProgressLogShowService

En estas clases aparece este mismo error repetido varias veces:

```
assert object != null;
```

- Descripción:

Asserts should not be used to check the parameters of a public method

Intentionality | Not logical Maintainability

Why is this an issue? More Info

An `assert` is inappropriate for parameter validation because assertions can be disabled at runtime in the JVM, meaning that a bad operational setting would completely eliminate the intended checks. Further, `assert`s that fail throw `AssertionError`s, rather than throwing some type of `Exception`. Throwing `Error`s is completely outside of the normal realm of expected `catch`/`throw` behavior in normal programs.

This rule raises an issue when a `public` method uses one or more of its parameters with `assert`s.

```
public void setPrice(int price) {
  assert price >= 0 && price <= MAX_PRICE;
  // Set the price
}
```

Compliant solution

```
public void setPrice(int price) {
  if (price < 0 || price > MAX_PRICE) {
    throw new IllegalArgumentException("Invalid price: " + price);
  }
  // Set the price
}
```

- Solución:

Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
private static String invalidObject = "Invalid object: ";

if (object == null)
    Duplication
    throw new IllegalArgumentException(ClientProgressLogUpdateService.invalidObject + object);
```

6.2. ClientContractShowService

En esta clase encontramos el siguiente error:

```
implies 'contract' is not null.' | Expression is always true.
status = super.getRequest().getPrincipal().hasRole( 1 contract.getClient()) || 2 contract != null && !contract.isDraftMode();
super.getResponse().setAuthorised(status);
```

- Descripción:

Boolean expressions should not be gratuitous

Intentionality | Not logical Maintainability java:S2589

Why is this an issue? More Info

If a boolean expression doesn't change the evaluation of the condition, then it is entirely unnecessary, and can be removed. If it is gratuitous because it does not match the programmer's intent, then it's a bug and the expression should be fixed.

```
a = true;
if (a) { // Noncompliant
  doSomething();
}

if (b && a) { // Noncompliant; "a" is always "true"
  doSomething();
}

if (c || !a) { // Noncompliant; "!a" is always "false"
  doSomething();
}

if (c || (!c && b)) { // Noncompliant; c || (!c && b) is equal to c || b
  doSomething();
}
```

Compliant solution

```
a = true;
if (foo(a)) {
  doSomething();
}

if (b) {
  doSomething();
}

if (c) {
  doSomething();
}
```

- Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
contractId = super.getRequest().getData( id , int.class);
contract = this.repository.findContractById(contractId);
status = contract != null ? super.getRequest().getPrincipal().hasRole(contract.getClient()) || !contract.isDraftMode() : false;
super.getResponse().setAuthorised(status);
```

6.3 AuthenticatedContractShowService

En esta clase encontramos este error:

```
dataset = super.unbind(object, "code", "providerName", "customerName", "budget", "project");
```

- Descripción:

Unused assignments should be removed

Intentionality | Not logical Maintainability java:S1854

Why is this an issue? More Info

A dead store happens when a local variable is assigned a value that is not read by any subsequent instruction. Calculating or retrieving a value only to then overwrite it or throw it away, could indicate a serious error in the code. Even if it's not an error, it is at best a waste of resources. Therefore all calculated values should be used.

```
i = a + b; // Noncompliant; calculation result not used before value is overwritten
i = compute();
```

Compliant solution

```
i = a + b;
i += compute();
```

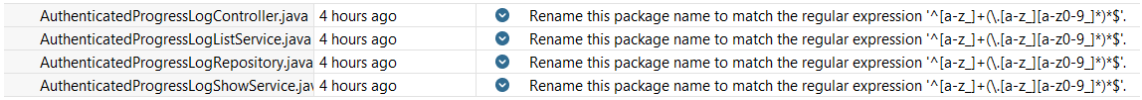
- Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

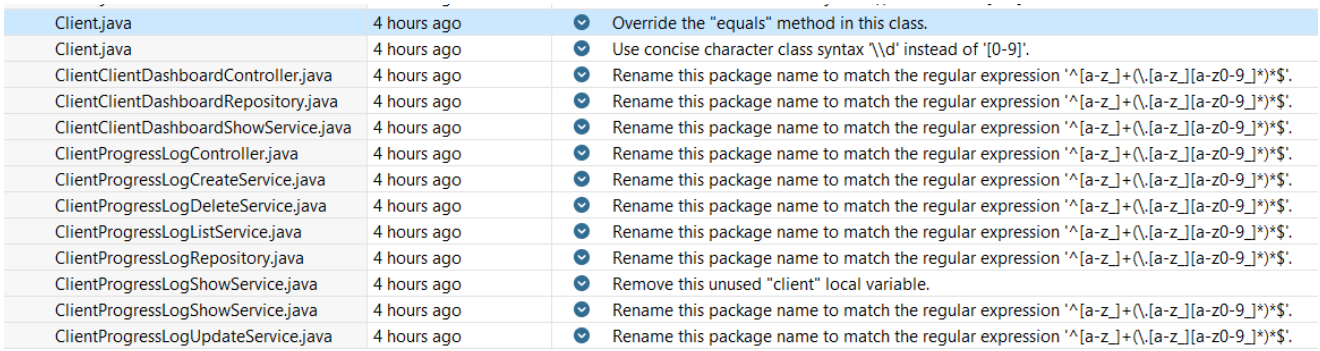
```
Dataset dataset;
```

7.Impacto Bajo

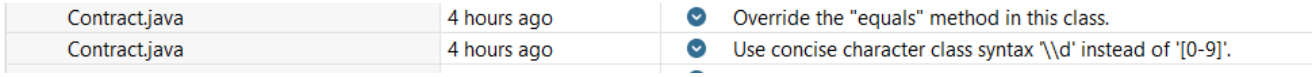
En las siguientes imágenes se muestran los errores de impacto medio detectados por SonarLint.



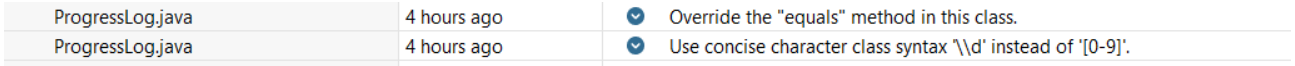
AuthenticatedProgressLogController.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
AuthenticatedProgressLogListService.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
AuthenticatedProgressLogRepository.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
AuthenticatedProgressLogShowService.ja	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.



Client.java	4 hours ago	Override the "equals" method in this class.
Client.java	4 hours ago	Use concise character class syntax '\\d' instead of '[0-9]'.
ClientClientDashboardController.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
ClientClientDashboardRepository.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
ClientClientDashboardShowService.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
ClientProgressLogController.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
ClientProgressLogCreateService.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
ClientProgressLogDeleteService.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
ClientProgressLogListService.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
ClientProgressLogRepository.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
ClientProgressLogShowService.java	4 hours ago	Remove this unused "client" local variable.
ClientProgressLogShowService.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.
ClientProgressLogUpdateService.java	4 hours ago	Rename this package name to match the regular expression '^[a-z_]+(\.[a-z_][a-z0-9_]*)*$'.



Contract.java	4 hours ago	Override the "equals" method in this class.
Contract.java	4 hours ago	Use concise character class syntax '\\d' instead of '[0-9]'.



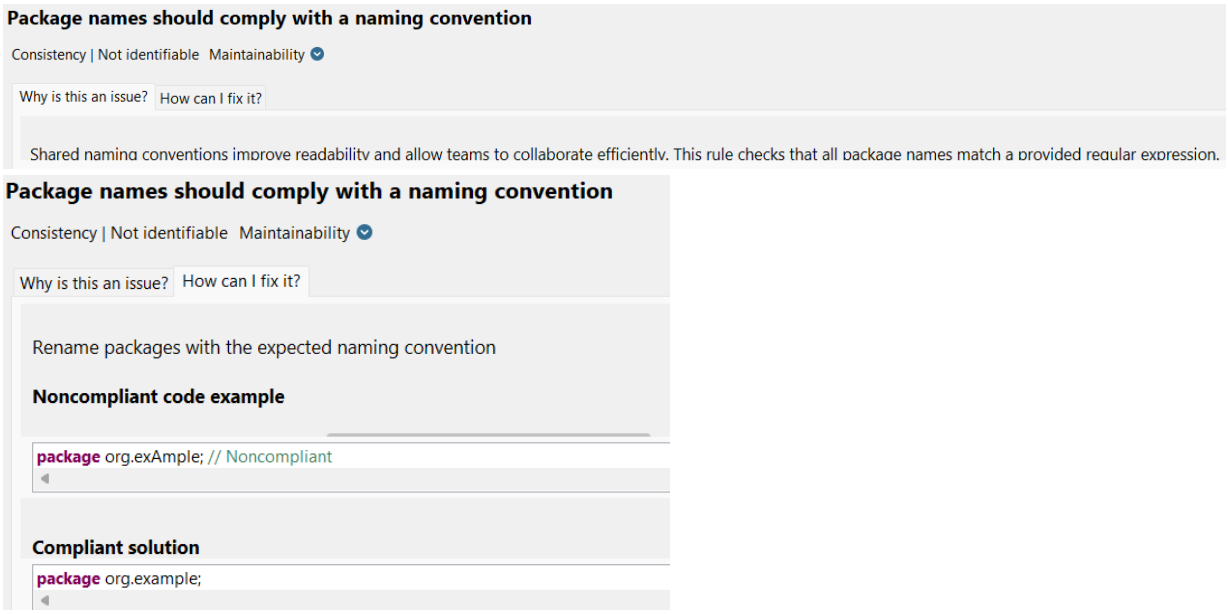
ProgressLog.java	4 hours ago	Override the "equals" method in this class.
ProgressLog.java	4 hours ago	Use concise character class syntax '\\d' instead of '[0-9]'.

7.1. AuthenticatedProgressLogShowService, AuthenticatedProgressLogRepository, AuthenticatedProgressLogListService y AuthenticatedProgressLogController

En estas clases nos encontramos con el mismo error:

```
package acme.features.authenticated.progressLog;
```

- Descripción:



- Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
package acme.features.authenticated.progresslog;
```

7.2. ClientProgressLogDeleteService, ClientProgressLogListService, ClientProgressLogRepository, ClientProgressLogShowService, ClientProgressLogUpdateService, ClientProgressLogCreateService y ClientProgressLogController

En estas clases nos encontramos con el mismo error:

```
2 package acme.features.client.progressLog;
```

- Descripción:

  Igual que la de AuthenticatedProgressLogShowService, AuthenticatedProgressLogRepository, AuthenticatedProgressLogListService y AuthenticatedProgressLogController.

- Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
package acme.features.client.progresslog;
```

7.3. ClientClientDashboardShowService, ClientClientDashboardRepository y ClientClientDashboardController

En estas clases nos encontramos con el mismo error:

```
package acme.features.client.clientDashboard;
```

- Descripción:

  Igual que la de AuthenticatedProgressLogShowService, AuthenticatedProgressLogRepository, AuthenticatedProgressLogListService y AuthenticatedProgressLogController.

- Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
package acme.features.client.clientdashboard;
```

7.4. Client

En client nos encontramos los siguientes errores:

```
  @Pattern(regexp = "CLI-[0-9]{4}", message = "CLI-[0-9]{4}")
  private String                    identification;

public class Client extends AbstractRole {
```

- Descripción:

Problema 1:

Regular expression quantifiers and character classes should be used concisely

Intentionality | Not clear Maintainability java:S63

Why is this an issue?

With regular expressions syntax, it's possible to express the same thing in many ways. For example, to match a two-digit number, one could write `[0-9]{2,2}` or `\d{2}`. Latter is not only shorter in terms of expression length, but also easier to read and thus to maintain. This rule recommends to replace some bulky quantifiers and character classes with more concise equivalents:

- `\d` for `[0-9]` and `\D` for `[^0-9]`
- `\w` for `[A-Za-z0-9_]` and `\W` for `[^A-Za-z0-9_]`
- `.` for character classes matching everything (e.g. `[\w\W]`, `[\d\D]`, or `[\s\S]` with `s` flag)
- `x?` for `x{0,1}`, `x*` for `x{0,}`, `x+` for `x{1,}`, `x{N}` for `x{N,N}`

```
"[0-9]" // Noncompliant - same as "\\d"
"[^0-9]" // Noncompliant - same as "\\D"
"[A-Za-z0-9_]" // Noncompliant - same as "\\w"
"[\\w\\W]" // Noncompliant - same as "."
"a{0,}" // Noncompliant - same as "a*"
```

Compliant solution

```
"\\d"
"\\D"
"\\w"
"."
"a*"
```

Problema 2:

Subclasses that add fields to classes that override "equals" should also override "equals"

Intentionality | Not complete Maintainability java:S2160

This rule raises an issue when a subclass of a class that overrides `Object.equals` introduces new fields but does not also override the `Object.equals` method.

Why is this an issue? How can I fix it? More Info

When a class overrides `Object.equals`, this indicates that the class not just considers object identity as equal (the default implementation of `Object.equals`) but implements another logic for what is considered equal in the context of this class. Usually (but not necessarily), the semantics of `equals` in this case is that two objects are equal when their state is equal field by field.

Subclasses that add fields to classes that override "equals" should also override "equals"

Intentionality | Not complete Maintainability java:S2160

This rule raises an issue when a subclass of a class that overrides `Object.equals` introduces new fields but does not also override the `Object.equals` method.

Why is this an issue? How can I fix it? More Info

Consider the following example:

```
class Foo {

  final int a;

  @Override
  public boolean equals(Object other) {
    if (other == null) return false;
    if (getClass() != other.getClass()) return false;
    return a == ((Foo) other).a;
  }
}
```

```
class Bar extends Foo { // Noncompliant, `equals` ignores the value of `b`
  final int b;
}
```

Override the `equals` method in the subclass to incorporate the new fields into the comparison:

```
class Bar extends Foo { // Compliant, `equals` now also considers `b`

  final int b;

  @Override
  public boolean equals(Object other) {
    if (!super.equals(other)) return false;
    return b == ((Bar) other).b;
  }
}
```

In case the new fields should not be part of the comparison because they are, for example, auxiliary variables not contributing to the object value (), still override the method to make the point clear that this was not just forgotten:

```
class Bar extends Foo { // Compliant, we do explicitly not want to take `b` into account

  final int b;

  @Override
  public boolean equals(Object other) {
    return super.equals(other);
  }
}
```

- ## Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

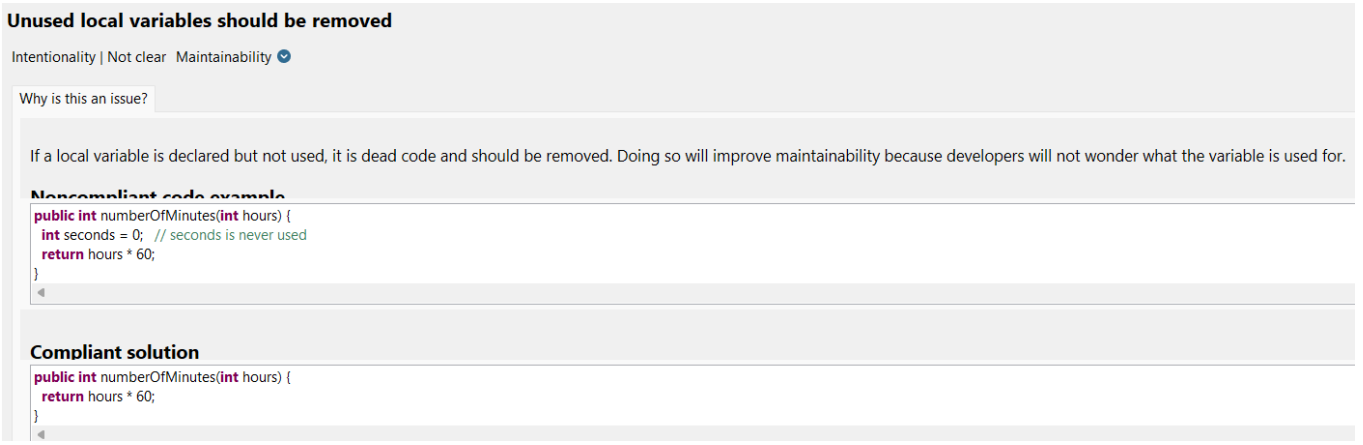
```
@NotBlank
@Pattern(regexp = "CLI-\\d{4}", message = "CLI-[0-9]{4}")
private String              identification;

@Override
public int hashCode() {
    final int prime = 31;
    int result = super.hashCode();
    result = prime * result + Objects.hash(this.companyName, this.email, this.furtherInformation, this.identification, this.type);
    return result;
}

@Override
public boolean equals(final Object obj) {
    if (this == obj)
        return true;
    if (!super.equals(obj))
        return false;
    if (this.getClass() != obj.getClass())
        return false;
    Client other = (Client) obj;
    return Objects.equals(this.companyName, other.companyName) && Objects.equals(this.email, other.email) && Objects.equals(this.furtherInformation, other.furtherInformation)
        && this.type == other.type;
}
```

7.5. ClientProgressLogShowService

En ClientProgressLogShowService nos encontramos el siguiente error:

```
public void unbind(final ProgressLog object) {
    assert object != null;

    String client;
    Dataset dataset;

    client = object.getContract().getClient().getIdentification();

    dataset = super.unbind(object, "recordId", "completeness", "comment", "regist
    dataset.put("masterId", object.getContract().getId());
    dataset.put("draftMode", object.getContract().isDraftMode());
    dataset.put("contract", object.getContract().getCode());

    super.getResponse().addData(dataset);
}
```

- Descripción:



- Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
@Override
public void unbind(final ProgressLog object) {
    assert object != null;

    Dataset dataset;

    dataset = super.unbind(object, "recordId", "completeness", "comment", "registrationMoment", "re
    dataset.put("masterId", object.getContract().getId());
    dataset.put("draftMode", object.getContract().isDraftMode());
    dataset.put("contract", object.getContract().getCode());

    super.getResponse().addData(dataset);
}
```

7.6. Contract

En contract nos encontramos los siguientes errores:

```
@Pattern(regexp = "^[A-Z]{1,3}-[0-9]{3}$", message = "[A-Z]{1,3}-[0-9]{3}")
private String              code;

public class Contract extends AbstractEntity {
    // Serialisation identifier --------------------
```

- Descripción:

  Problema 1: Igual que el problema 1 del 7.4

  Problema 2: Igual que el problema 2 del 7.4

- Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```
@Pattern(regexp = "^[A-Z]{1,3}-\\d{3}$", message = "[A-Z]{1,3}-[0-9]{3}")
private String              code;


@Override
public int hashCode() {
    final int prime = 31;
    int result = super.hashCode();
    result = prime * result + Objects.hash(this.budget, this.client, this.code, this.customerName, this.draftMode, this.goals, this.instantiationMoment, this.project, this.
    return result;
}

@Override
public boolean equals(final Object obj) {
    if (this == obj)
        return true;
    if (!super.equals(obj))
        return false;
    if (this.getClass() != obj.getClass())
        return false;
    Contract other = (Contract) obj;
    return Objects.equals(this.budget, other.budget) && Objects.equals(this.client, other.client) && Objects.equals(this.code, other.code) && Objects.equals(this.customerNa
        && Objects.equals(this.goals, other.goals) && Objects.equals(this.instantiationMoment, other.instantiationMoment) && Objects.equals(this.project, other.project) &&
}
```

7.7. ProgressLog

En progressLog nos encontramos los siguientes errores:

```
@Pattern(regexp = "^PG-[A-Z]{1,2}-[0-9]{4}$", message = "PG-[A-Z]{1,2}-[0-9]{4}")
private String              recordId;

public class ProgressLog extends AbstractEntity {
```

- Descripción:

  Problema 1: Igual que el problema 1 del 7.4

  Problema 2: Igual que el problema 2 del 7.4

- Solución:

  Una vez llevadas a cabo las correcciones sugeridas por SonarLint el código queda así:

```java
@NotBlank
@Pattern(regexp = "^PG-[A-Z]{1,2}-//d{4}$", message = "PG-[A-Z]{1,2}-[0-9]{4}")
private String              recordId;


@Override
public int hashCode() {
    final int prime = 31;
    int result = super.hashCode();
    result = prime * result + Objects.hash(this.comment, this.completeness, this.contract, this.recordId, this.registrationMomer
    return result;
}

@Override
public boolean equals(final Object obj) {
    if (this == obj)
        return true;
    if (!super.equals(obj))
        return false;
    if (this.getClass() != obj.getClass())
        return false;
    ProgressLog other = (ProgressLog) obj;
    return Objects.equals(this.comment, other.comment) && Double.doubleToLongBits(this.completeness) == Double.doubleToLongBits(
        && Objects.equals(this.registrationMoment, other.registrationMoment) && Objects.equals(this.responsiblePerson, other.res
}
```

8.Conclusión

Pese a la cantidad considerable de errores que han aparecido en el análisis de SonarLint, muchos de ellos se repinten en varias ocasiones y todos son fáciles de arreglar. Por lo que se considera que el código era adecuado en un principio, aunque ahora es mucho más correcto.

9.Bibliografía

En blanco intencionalmente.