

G2-8 Design & Testing II

Documentación refactor realizadas:

Refactor PacienteController.

Remove Dispensables:

En el controllador de Pacientes podemos encontrar varios dispensables. Para refactorizar PacienteController y que la deuda técnica sea reducida debemos de eliminar este código que no aporta nada a nuestro proyecto.

- Dead Code:

Eliminación de código comentado:



- Imports dispensables:

```
import org.hibernate.TypeMismatchException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.samples.petclinic.model.Cita;
import org.springframework.samples.petclinic.model.Paciente;
import org.springframework.samples.petclinic.service.AuthoritiesService;
import org.springframework.samples.petclinic.service.CitaService;
import org.springframework.samples.petclinic.service.HistoriaClinicaService;
import org.springframework.samples.petclinic.service.MedicoService;
import org.springframework.samples.petclinic.service.PacienteService;
import org.springframework.samples.petclinic.service.UserService;
import org.springframework.samples.petclinic.util.DniValidator;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
```

- Declaración de servicio el cual no es usado:

`private final HistoriaClinicaService historiaClinicaService;`

Remove this unused "historiaClinicaService" private field. Why is this an issue? 2 months ago ▾ L45 🔗

🔗 Code Smell 🚨 Major 🔵 Open Not assigned 5min effort Comment 🗑️ unused

- Eliminar variables y declaraciones no utilizadas:

`String view = "/pacientes";`

Remove this useless assignment to local variable "view". Why is this an issue? 2 months ago ▾ L151 🔗

🔗 Code Smell 🚨 Major 🔵 Open Not assigned 15min effort Comment 🗑️ cert_cwe, unused

`Authentication authentication = SecurityContextHolder.getContext().getAuthentication();`

Remove this useless assignment to local variable "authentication". Why is this an issue? 2 months ago ▾ L153 🔗

🔗 Code Smell 🚨 Major 🔵 Open Not assigned 15min effort Comment 🗑️ cert_cwe, unused

Remove this unused "authentication" local variable. Why is this an issue? 2 months ago ▾ L153 🔗

🔗 Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort Comment 🗑️ unused

- Eliminar comprobaciones de boolean:

`if (paciente.getApellidos() == "") {`

Use the "equals" method if value comparison was intended. Why is this an issue? 2 months ago ▾ L110 🔗

🐛 Bug 🚨 Major 🔵 Open Not assigned 5min effort Comment 🗑️ cert_cwe

`} else if (mismoMedico == false) {`

Remove the literal "false" boolean value. Why is this an issue? 22 days ago ▾ L177 🔗

🔗 Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort Comment 🗑️ clumsy

```

boolean pacienteValid = noTieneContacto == false && dniOk == true;

```

Remove the literal "true" boolean value. Why is this an issue? 23 days ago ▾ L214 [🔗](#)
 ⚙️ Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort [Comment](#) [clumsy](#)

Remove the literal "false" boolean value. Why is this an issue? 23 days ago ▾ L214 [🔗](#)
 ⚙️ Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort [Comment](#) [clumsy](#)

```

boolean telefonoOk = ((paciente.getN_telefono() == null) ? true

```

Remove the literal "true" boolean value. Why is this an issue? 23 days ago ▾ L215 [🔗](#)
 ⚙️ Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort [Comment](#) [clumsy](#)

```

    : (paciente.getN_telefono().toString().length() == 9));

model.addAttribute("isNewPaciente", false);

if (result.hasErrors() || !pacienteValid || !telefonoOk) {
    model.addAttribute("medicoList", this.medicoService.getMedicos());
    if (noTieneContacto) {
        result.rejectValue("domicilio", "error.formaContacto", "No tiene forma de contacto.");
    }
    if (dniOk == false) {

```

Remove the literal "false" boolean value. Why is this an issue? 23 days ago ▾ L225 [🔗](#)
 ⚙️ Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort [Comment](#) [clumsy](#)

```

        result.rejectValue("DNI", "error.DNI", "DNI invalido.");
    }
    if (telefonoOk == false) {

```

Remove the literal "false" boolean value. Why is this an issue? 23 days ago ▾ L228 [🔗](#)
 ⚙️ Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort [Comment](#) [clumsy](#)

```

boolean pacienteValid = noTieneContacto == false && dniOk == true;

```

Remove the literal "false" boolean value. Why is this an issue? 23 days ago ▾ L256 [🔗](#)
 ⚙️ Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort [Comment](#) [clumsy](#)

Remove the literal "true" boolean value. Why is this an issue? 23 days ago ▾ L256 [🔗](#)
 ⚙️ Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort [Comment](#) [clumsy](#)

```

boolean telefonoOk = ((paciente.getN_telefono() == null) ? true

```

Remove the literal "true" boolean value. Why is this an issue? 22 days ago ▾ L257 [🔗](#)
 ⚙️ Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort [Comment](#) [clumsy](#)

```

    if (dniOk == false) {

```

Remove the literal "false" boolean value. Why is this an issue? 23 days ago ▾ L268 [🔗](#)
 ⚙️ Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort [Comment](#) [clumsy](#)

```

        result.rejectValue("DNI", "error.DNI", "DNI invalido.");
    }
    if (telefonoOk == false) {

```

Remove the literal "false" boolean value. Why is this an issue? 23 days ago ▾ L271 [🔗](#)
 ⚙️ Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort [Comment](#) [clumsy](#)

- Código duplicado

A continuación, se muestra el código duplicado el cual será declarado en variables.

```
@GetMapping(value = "/pacientes/find")
public String initFindForm(final Map<String, Object> model) {
    model.put("paciente", new Paciente());
}
```

Define a constant instead of duplicating this literal "paciente" 3 times. Why is this an issue? 2 months ago ▾ L98 3 🔗
⚙️ Code Smell 🚨 Critical 🔵 Open Not assigned 8min effort Comment 🛠️ design

```
return "redirect:/pacientes/" + paciente.getId();
```

Define a constant instead of duplicating this literal "redirect:/pacientes/" 7 times. Why is this an issue? 9 days ago ▾ L121 7 🔗
⚙️ Code Smell 🚨 Critical 🔵 Open Not assigned 16min effort Comment 🛠️ design

```
model.addAttribute("isNewPaciente", false);
```

Define a constant instead of duplicating this literal "isNewPaciente" 4 times. Why is this an issue? 21 days ago ▾ L199 4 🔗
⚙️ Code Smell 🚨 Critical 🔵 Open Not assigned 10min effort Comment 🛠️ design

```
model.addAttribute("medicoList", this.medicoService.getMedicos());
```

Define a constant instead of duplicating this literal "medicoList" 4 times. Why is this an issue? 23 days ago ▾ L198 4 🔗
⚙️ Code Smell 🚨 Critical 🔵 Open Not assigned 10min effort Comment 🛠️ design

Esta son las variables asignadas que arreglan estos malos olores:

```
private static final String VIEWS_PACIENTE_CREATE_OR_UPDATE_FORM = "pacientes/createOrUpdatePacientesForm";
private static final String REDIRECT_PACIENTES = "redirect:/pacientes/";
private static final String PACIENTE_STRING = "paciente";
private static final String MEDICO_LIST_URL = "medicoList";
private static final String IS_NEW_PACIENTE_LABEL = "isNewPaciente";
```

Bloaters:

Long Method:

En método borrarPaciente, como podemos ver en la imagen inferior, es bastante largo.

```

@RequestMapping(value = "/pacientes/{pacienteId}/delete")
public String borrarPaciente(@PathVariable("pacienteId") final int pacienteId, final ModelMap modelMap) {
    String view = "/pacientes";

    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    Optional<Paciente> paciente = this.pacienteService.findPacienteById(pacienteId);

    if (paciente.isPresent()) {
        Collection<Cita> citas = this.citaService.findAllByPaciente(paciente.get());

        boolean puedeBorrarse = citas.isEmpty();
        if (!citas.isEmpty()) {
            LocalDate ultimaCita = citas.stream().map(Cita::getFecha).max(LocalDate::compareTo).get();
            LocalDate hoy = LocalDate.now();
            puedeBorrarse = hoy.compareTo(ultimaCita) >= 6
                || hoy.compareTo(ultimaCita) == 5 && hoy.getDayOfYear() > ultimaCita.getDayOfYear();
        }

        boolean mismoMedico = paciente.get().getMedico().equals(this.userService.getCurrentMedico());
        boolean medicoEnabled = this.medicoService.getMedicoById(this.userService.getCurrentMedico().getId())
            .getUser().isEnabled();

        puedeBorrarse = puedeBorrarse && mismoMedico && medicoEnabled;

        if (puedeBorrarse) {
            this.pacienteService.deletePacienteByMedico(pacienteId, this.userService.getCurrentMedico().getId());
            modelMap.addAttribute("message", "Paciente borrado exitosamente");
            view = "redirect:/pacientes";
        } else if (mismoMedico == false) {
            modelMap.addAttribute("message", "No tiene acceso para borrar a este paciente");
            view = "redirect:/pacientes/" + pacienteId;
        } else {
            modelMap.addAttribute("message", "Paciente no puede borrarse");
            view = "redirect:/pacientes/" + pacienteId;
        }
    } else {
        modelMap.addAttribute("message", "Paciente no encontrado");
        view = "accessNotAuthorized";
    }
    return view;
}

```

Para refactorizar este método hemos dividido las comprobaciones en varios métodos que también podrán ser reutilizados por el resto de métodos. A continuación, se muestra como quedaría el método borrarPaciente.

```

@GetMapping(value = "/pacientes/{pacienteId}/delete")
public String borrarPaciente(@PathVariable("pacienteId") final int pacienteId, final ModelMap modelMap) {
    String view;
    Optional<Paciente> optionalPaciente = this.pacienteService.findPacienteById(pacienteId);
    String labelMessage = "message";

    if (optionalPaciente.isPresent()) {
        Paciente paciente = optionalPaciente.get();
        boolean puedeBorrarse = checkDeleteforCitas(paciente) && sameMedico(paciente)
            && paciente.getMedico().getUser().isEnabled();

        if (puedeBorrarse) {
            this.pacienteService.deletePacienteByMedico(pacienteId, this.userService.getCurrentMedico().getId());
            modelMap.addAttribute(labelMessage, "Paciente borrado exitosamente");
            view = "redirect:/pacientes";
        } else if (!sameMedico(paciente)) {
            modelMap.addAttribute(labelMessage, "No tiene acceso para borrar a este paciente");
            view = REDIRECT_PACIENTES + pacienteId;
        } else {
            modelMap.addAttribute(labelMessage, "Paciente no puede borrarse");
            view = REDIRECT_PACIENTES + pacienteId;
        }
    } else {
        modelMap.addAttribute(labelMessage, "Paciente no encontrado");
        view = "accessNotAuthorized";
    }
    return view;
}

```

Estos son los métodos auxiliares creados que podrán ser reutilizados por otros métodos del controlador.

```

public Paciente checkPacienteIsPresent(int pacienteId) {
    return this.pacienteService.findPacienteById(pacienteId).orElse(null);
}

public boolean sameMedico(Paciente paciente) {
    return paciente.getMedico().equals(this.userService.getCurrentMedico());
}

public boolean checkDeleteforCitas(Paciente paciente) {
    Collection<Cita> citas = this.citaService.findAllByPaciente(paciente);
    if (!citas.isEmpty()) {
        LocalDate ultimaCita = citas.stream().filter(c -> c.getFecha() != null).map(Cita::getFecha)
            .max(LocalDate::compareTo).orElse(null);
        if (ultimaCita != null) {
            LocalDate hoy = LocalDate.now();

            int compareDate = hoy.compareTo(ultimaCita);

            return compareDate >= 6 || compareDate >= 5 && hoy.getDayOfYear() > ultimaCita.getDayOfYear();
        } else {
            return true;
        }
    } else {
        return true;
    }
}

public boolean hasContact(Paciente paciente) {
    return paciente.getN_telefono() == null && paciente.getDomicilio().isEmpty() && paciente.getEmail().isEmpty();
}

public boolean telefonoOk(Paciente paciente) {
    if ((paciente.getN_telefono() == null)) {
        return true;
    } else {
        return paciente.getN_telefono().toString().length() == 9;
    }
}

```

Arreglo de bugs:

Sonar nos señala ciertos bugs, controlados por otras clases. A continuación, se muestran los bugs y como han sido fixeados.

El siguiente error se nos muestra que no comprobamos como se nos devuelve un Optional<Paciente>.



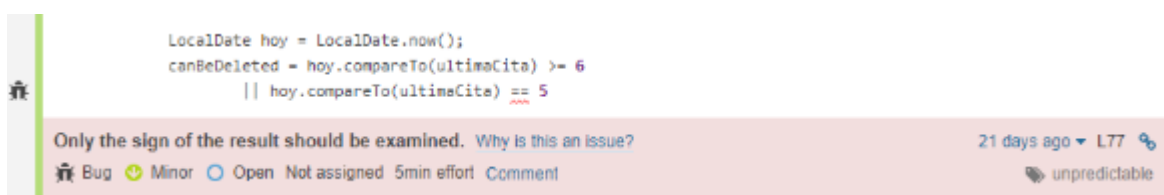
Para arreglarlo, hemos creado un método auxiliar que nos comprueba esto y nos devuelve el Paciente.

```

public Paciente checkPacienteIsPresent(int pacienteId) {
    return this.pacienteService.findPacienteById(pacienteId).orElse(null);
}

```

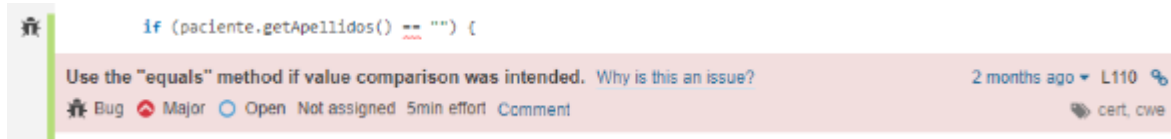
Se nos muestra el siguiente error, el cual comprueba que la última cita del Paciente sea mayor a 5 años desde el día actual.



Este error es fácilmente fixado cambiando la igualdad por un comparador de mayor o igual.

```
return compareDate >= 6 || compareDate >= 5 && hoy.getDayOfYear() > ultimaCita.getDayOfYear();
```

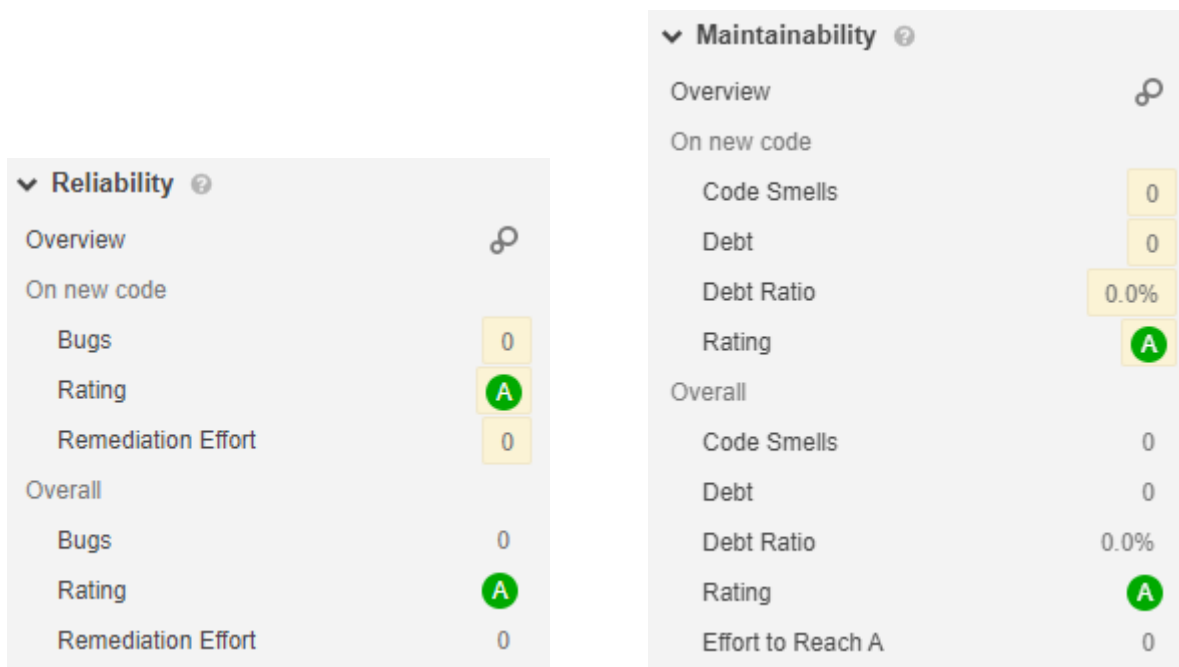
El siguiente equals es mostrado como error.



Para arreglarlo cambiamos el comparador de igualdad por un equals.

```
if (paciente.getApellidos().equals("")) {
```

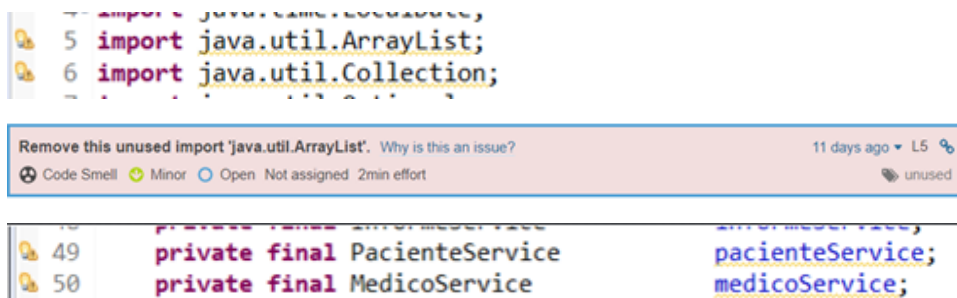
Una vez hemos refactorizado la clase PacienteController, podemos observar como la deuda técnica y los bugs se reducen al 0%.



RefactorInformeController:

Disposables:

Eliminación de Imports Innecesarios.



Remove this unused "pacienteService" private field. Why is this an issue? 2 months ago ▾ L49 🔗

🔗 Code Smell 🚫 Major 🔵 Open Not assigned 5min effort 🗑️ unused

Eliminación de código comentado:

```
69 // @InitBinder
70 // public void setAllowedFields(final WebDataBinder dataBinder) {
71 // dataBinder.setDisallowedFields("id");
72 // }
73
```

This block of commented-out lines of code should be removed. Why is this an issue? last month ▾ L70 🔗

🔗 Code Smell 🚫 Major 🔵 Open Not assigned 5min effort 🗑️ unused

Eliminación de Excepciones Innecesarias:

```
84 @GetMapping(path = "informes/delete/{informeId}")
85 public String borrarInforme(@PathVariable("informeId") final int informeId, final ModelMap modelMap) throws DataAccessException {
86     Optional<Informe> informe = this.informeService.findInformeById(informeId);
87
88     Medico medicoactual = this.userService.getCurrentMedico();
89     Informe informeaux = informe.get();
90     Medico medicocorrecto = informeaux.getCita().getPaciente().getMedico();
91
```

Remove the declaration of thrown exception 'org.springframework.dao.DataAccessException' which is a runtime exception. Why is this an issue? 9 days ago ▾ L85 🔗

🔗 Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort 🗑️ clumsy, error-handling, redundant, unu...

```
@GetMapping(value = "/informes/{informeId}/addtohistoriaclinica")
public String addHistoriaClinicaToInforme(@PathVariable("informeId") final int informeId) throws DataAccessException, IllegalArgumentException {
```

Remove the declaration of thrown exception 'org.springframework.dao.DataAccessException' which is a runtime exception. Why is this an issue? 9 days ago ▾ L181 🔗

🔗 Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort 🗑️ clumsy, error-handling, redundant, unu...

```
@GetMapping(value = "/informes/{informeId}/deletefromhistoriaclinica")
public String deleteFromHistoriaClinica(@PathVariable("informeId") final int informeId) throws DataAccessException, IllegalArgumentException {
```

Remove the declaration of thrown exception 'java.lang.IllegalArgumentException', as it cannot be thrown from method's body. Why is this an issue? 9 days ago ▾ L198 🔗

🔗 Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort 🗑️ clumsy, error-handling, redundant, unu...

Remove the declaration of thrown exception 'org.springframework.dao.DataAccessException' which is a runtime exception. Why is this an issue? 9 days ago ▾ L198 🔗

🔗 Code Smell 🟡 Minor 🔵 Open Not assigned 5min effort 🗑️ clumsy, error-handling, redundant, unu...

Código Duplicado:

```
ModelAndView error = new ModelAndView("accessNotAuthorized");
```

Use already-defined constant 'VIEWS_ACCESS_NOT_AUTHORIZED' instead of duplicating its value here. Why is this an issue? 9 days ago ▾ L116 🔗

🔗 Code Smell 🚫 Critical 🔵 Open Not assigned 4min effort 🗑️ design

Immediately return this expression instead of assigning it to the temporary variable "error". Why is this an issue? 9 days ago ▾ L116 🔗

🔗 Code Smell 🟡 Minor 🔵 Open Not assigned 2min effort 🗑️ clumsy

```
Informe informe = new Informe();
informe.setCita(cita);
model.put("informe", informe);
```

Define a constant instead of duplicating this literal "informe" 4 times. Why is this an issue? last month ▾ L156 🔗

🔗 Code Smell 🚫 Critical 🔵 Open Not assigned 10min effort 🗑️ design

Estas serían las variables estáticas tras la refactorización de estos disponibles:


```

private static final String VIEWS_INFORME_CREATE_OR_UPDATE_FORM = "informes/createOrUpdateInformeForm";
private static final String VIEWS_ACCESS_NOT_AUTHORIZED = "accessNotAuthorized";
private static final String REDIRECTION_TO_CITA = "redirect:/citas/";
private static final String INFORME_MODELO = "informe";
private final InformeService informeService;
private final UserService userService;
private final CitaService citaService;
private final HistoriaClinicaService historiaClinicaService;
private final TratamientoService tratamientoService;

```

Eliminar comprobaciones de Boolean:

`if (!medicoactual.equals(medicocorrecto) && noasociado) {`

Use the primitive boolean expression here. Why is this an issue? 9 days ago ▾ L115

Code Smell Minor Open Not assigned 5min effort

```

if (!medicoactual.equals(medicocorrecto)) {
    return InformeController.VIEWS_ACCESS_NOT_AUTHORIZED;
} else {
    if (cita.getFecha().equals(today) && !hasCitaInforme) {

```

Use the primitive boolean expression here. Why is this an issue? 24 days ago ▾ L153

Code Smell Minor Open Not assigned 5min effort

```

        return InformeController.VIEWS_INFORME_CREATE_OR_UPDATE_FORM;
    } else if (hasCitaInforme) {

```

Use the primitive boolean expression here. Why is this an issue? 10 days ago ▾ L158

Code Smell Minor Open Not assigned 5min effort

Para arreglar este tipo de errores, se ha realizado dos cosas:

En caso de tener que reutilizar la condición en otro método, crear un método auxiliar que lo haga.

```

public boolean hasCitaInforme(Cita cita){
    return this.informeService.citaHasInforme(cita);
}

```

En caso de ser una condición usada una sola vez, se ha añadido en el propio if:

```

if (!sameMedico(informe) && informe.getHistoriaClinica() == null) {
    return new ModelAndView(VIEWS_ACCESS_NOT_AUTHORIZED);
}

```

Bloaters:

Método Largo:

Métodos como ShowInforme eran muy largos antes de realizar una refactorización:

```

@GetMapping(value = "/informes/{informeId}")
public ModelAndView showInforme(@PathVariable("informeId") final int informeId) {

    ModelAndView mav = new ModelAndView("informes/informeDetails");
    Informe informe = this.informeService.findInformeById(informeId).get();
    Medico medicoactual = this.userService.getCurrentMedico();
    Medico medicocorrecto = informe.getCita().getPaciente().getMedico();
    Boolean noasociado = informe.getHistoriaClinica() == null;

    if (!medicoactual.equals(medicocorrecto) && noasociado) {
        ModelAndView error = new ModelAndView("accessNotAuthorized");
        return error;
    } else {

        mav.addObject(informe);

        Cita cita = this.citaService.findCitaById(informe.getCita().getId()).get();
        Paciente paciente = cita.getPaciente();
        mav.addObject(paciente);

        Boolean canBeDeleted = informe.getCita().getFecha().plusDays(1).equals(LocalDate.now().plusDays(1)) && informe.getHistoriaClinica() != null;
        mav.getModel().put("cannotbedeleted", !canBeDeleted);

        Collection<Tratamiento> tratamientos = this.tratamientoService.findTratamientosByInforme(informe);
        mav.getModel().put("editTratamientoOk", cita.getFecha().equals(LocalDate.now()));
        mav.getModel().put("tratamientos", tratamientos);

        Boolean canBeEdited = informe.getCita().getFecha().equals(LocalDate.now());
        mav.getModel().put("canbeedited", canBeEdited);

        return mav;
    }
}

```

Tras realizar diversas refactorizaciones, incluyendo una paginación para los tratamientos a raíz del tercer Profiling Realizado, el método dado queda así:

```

@GetMapping(value = "/informes/{informeId}")
public ModelAndView showInforme(@PathVariable("informeId") final int informeId,
@PageableDefault(value = 5, page= 0) Pageable pageable){

    ModelAndView mav = new ModelAndView("informes/informeDetails");
    Informe informe = checkInformeIsPresent(informeId);

    if (!sameMedico(informe) && informe.getHistoriaClinica() == null) {
        return new ModelAndView(VIEWS_ACCESS_NOT_AUTHORIZED);
    } else {

        mav.addObject(informe);

        Cita cita = findCita(informe.getCita().getId());
        Paciente paciente = cita.getPaciente();
        mav.addObject(paciente);

        Boolean canBeDeleted = informe.getCita().getFecha().plusDays(1).equals(LocalDate.now().plusDays(1)) && informe.getHistoriaClinica() != null;
        mav.getModel().put("cannotbedeleted", !canBeDeleted);

        if(informe.getTratamientos() != null){
            Page<Tratamiento> tratamientos = this.tratamientoService.findTratamientosByInforme(informeId, pageable);

            mav.getModel().put("editTratamientoOk", cita.getFecha().equals(LocalDate.now()));
            mav.getModel().put("tratamientos", tratamientos.getContent());
            mav.getModel().put("tratapages", tratamientos.getTotalPages()-1);
        }

        Boolean canBeEdited = informe.getCita().getFecha().equals(LocalDate.now());
        mav.getModel().put("canbeedited", canBeEdited);

        return mav;
    }
}

```

Como podemos observar se han eliminado muchas declaraciones innecesarias y varias de las condiciones han sido movidas hacia métodos auxiliares. Cabe destacar también la inclusión de la paginación para los tratamientos tal. La refactorización realizada para la paginación está completamente desarrollada en el documento de “Profiling”.

Los métodos auxiliares que han descargado este bloater son los siguientes:

```

public boolean sameMedico(Informe informe){
    return informe.getCita().getPaciente().getMedico().equals(this.userService.getCurrentMedico());
}

```

```
public Informe checkInformeIsPresent(int informeId){
    return this.informeService.findInformeById(informeId).orElse(null);
}
```

```
@ModelAttribute("cita")
public Cita findCita(@PathVariable("citaId") final int citaId) {
    return this.citaService.findCitaById(citaId).orElse(null);
}
```

Constructor largo:

```
57# @Autowired
58 public InformeController(final PacienteService pacienteService, final MedicoService medicoService, final UserService
59     final HistoriaClinicaService historiaClinicaService, final InformeService informeService, final TratamientoServ
60     this.informeService = informeService;
61     this.pacienteService = pacienteService;
62     this.medicoService = medicoService;
63     this.userService = userService;
64     this.citaService = citaService;
65     this.historiaClinicaService = historiaClinicaService;
66     this.tratamientoService = tratamientoService;
67 }
```

Constructor has 8 parameters, which is greater than 7 authorized. Why is this an issue?

9 days ago ▾ L58 🔗

🔗 Code Smell 🚫 Major 🔗 Open Not assigned 20min effort

🧠 brain-overload

Dado que el constructor estaba recibiendo demasiados parámetros (Y algunos de ellos completamente innecesarios, por lo que también hablamos de un Disposable), se han eliminado aquellos que no eran usados:

```
@Autowired
public InformeController(final UserService userService, final AuthoritiesService authoritiesService,
    final CitaService citaService, final HistoriaClinicaService historiaClinicaService, final InformeService informeService,
    final TratamientoService tratamientoService) {
    this.informeService = informeService;
    this.userService = userService;
    this.citaService = citaService;
    this.historiaClinicaService = historiaClinicaService;
    this.tratamientoService = tratamientoService;
}
```

Arreglo de Bugs:

Dado el uso del Tipo Optional, hemos optado por aplicar la misma solución que en Paciente Controller y comprobar si el objeto está presente antes de asignarlo en una variable:

```
public Informe checkInformeIsPresent(int informeId){
    return this.informeService.findInformeById(informeId).orElse(null);
}
```

Conclusiones de la refactorización de Informe Controller:

Aunque se han refactorizado la clase Informe Controller completa, sólo figuran aquí los ejemplos a destacar.

Tras los cambios realizados está es la Fiabilidad y Mantenibilidad de la clase Refactorizada:

Reliability	
Bugs	0
Reliability Rating	A
Reliability Remediation Effort	0
Maintainability	
Code Smells	0
Effort to Reach Maintainability Rating A	0
Maintainability Rating	A
Technical Debt	0
Technical Debt Ratio	0.0%

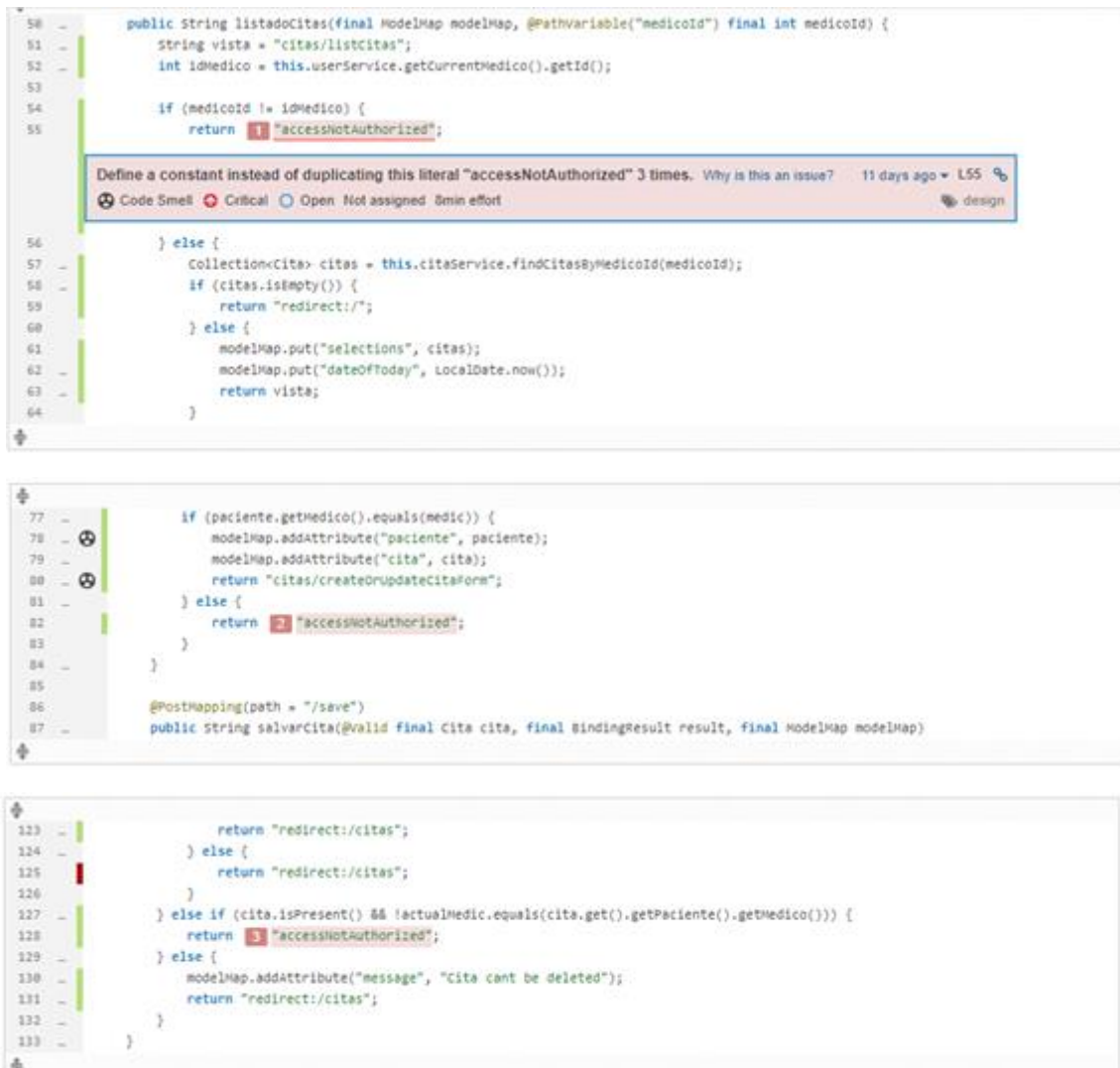
Quedando una deuda técnica de 0.

Refactor CitaController

Una vez ejecutado SonarCloud observamos que teníamos 17 malos olores de nivel crítico, así que decidimos centrarnos en su refactorización correspondiente. De este modo, el análisis llevado a cabo por SonarCloud nos mostraba que la clase CitaController poseía casi un tercio de estos malos olores críticos tal como podemos ver en la siguiente imagen, por lo que decidimos solucionarlos.

src/...samples/petclinic/web/CitaController.java	
Define a constant instead of duplicating this literal "accessNotAuthorized" 3 times. Why is this an issue?	11 days ago • L55
Code Smell Critical Open Not assigned 8min effort	design
Define a constant instead of duplicating this literal "paciente" 4 times. Why is this an issue?	11 days ago • L78
Code Smell Critical Open Not assigned 10min effort	design
Define a constant instead of duplicating this literal "citas/createOrUpdateCitaForm" 4 times. Why is this an issue?	11 days ago • L80
Code Smell Critical Open Not assigned 10min effort	design
Define a constant instead of duplicating this literal "message" 4 times. Why is this an issue?	24 days ago • L97
Code Smell Critical Open Not assigned 10min effort	design
Define a constant instead of duplicating this literal "fecha" 3 times. Why is this an issue?	10 days ago • L98
Code Smell Critical Open Not assigned 8min effort	design
Define a constant instead of duplicating this literal "redirect:/citas" 4 times. Why is this an issue?	4 days ago • L108
Code Smell Critical Open Not assigned 10min effort	design

Como podemos observar, estos malos olores críticos son en su totalidad de literales duplicados que se invocan en múltiples ocasiones, por lo que resulta una buena práctica convertirlos en constantes individuales que se invoquen cuando sea necesario. Se trata de un mal olor de tipo "Código duplicado". Así pues, observemos un ejemplo otorgado por SonarCloud donde se aprecian estos literales duplicados:



En estas imágenes, SonarCloud nos hace observar la múltiple repetición de un literal que podría ser declarado como constante. Este tipo de mal olor crítico se repite hasta un total de seis veces en esta clase, por lo que en pos de refactorizar nuestro código, procedemos a solucionar estos seis malos olores declarando los parámetros indicados como constantes. De este modo, una vez declaradas las constantes que producen estos malos olores, el código añadido es el siguiente:

```
public class CitaController {  
  
    private static final String accessNotAuthorized = "accessNotAuthorized";  
    private static final String pacienteModelName = "paciente";  
    private static final String VIEW_CREATE_OR_UPDATE_FORM = "citas/createOrUpdateCitaForm";  
    private static final String message = "message";  
    private static final String fecha = "fecha";  
    private static final String VIEW_CITAS = "redirect:/citas";  
  
}
```

Con este fragmento añadido y la correspondiente sustitución de los métodos que usan estos parámetros, los malos olores críticos de esta clase deberían desaparecer. Para comprobarlo, volvemos a ejecutar SonarCloud y el resultado es el siguiente:

Type
CODE SMELL
Clear

Vulnerability
27

Code Smell
8

Security Hotspot
0

Ctrl + click to add to selection

Severity
CRITICAL
Clear

Blocker
4
Minor
151

Critical
8
Info
185

Major
77

Ctrl + click to add to selection

Resolution

Status

Security Category

Creation Date

Language

Rule

Tag

Directory

File

Assignee

```
src/.../getClinic/configuration/SecurityConfiguration.java
```

Define a constant instead of duplicating this literal "admin" 7 times. Why is this an issue?
2 months ago • L42 • T -
Code Smell Critical Open Not assigned 10min effort
design

```
src/.../samples/getClinic/service/PacienteService.java
```

Refactor this method to reduce its Cognitive Complexity from 21 to the 15 allowed. Why is this an issue?
yesterday • L110 • T -
Code Smell Critical Open Not assigned 11min effort
brain-overload

```
src/.../samples/getClinic/service/TratamientoService.java
```

Define a constant instead of duplicating this literal "El tratamiento no ha podido guardarse," 5 times. Why is this an issue?
7 days ago • L50 • T -
Code Smell Critical Open Not assigned 5min effort
design

```
src/.../samples/getClinic/web/HistoriaClinicaController.java
```

Define a constant instead of duplicating this literal "historiaclinica" 5 times. Why is this an issue?
last month • L60 • T -
Code Smell Critical Open Not assigned 12min effort
design

Define a constant instead of duplicating this literal "pacientes" 4 times. Why is this an issue?
last month • L77 • T -
Code Smell Critical Open Not assigned 5min effort
design

```
src/.../samples/getClinic/web/TratamientoController.java
```

Define a constant instead of duplicating this literal "informe" 4 times. Why is this an issue?
7 days ago • L73 • T -
Code Smell Critical Open Not assigned 10min effort
design

Define a constant instead of duplicating this literal "tratamiento" 4 times. Why is this an issue?
7 days ago • L74 • T -
Code Smell Critical Open Not assigned 10min effort
design

```
src/.../samples/getClinic/web/UserController.java
```

Define a constant instead of duplicating this literal "message" 4 times. Why is this an issue?
2 months ago • L92 • T -
Code Smell Critical Open Not assigned 10min effort
design

8 of 8 shown

Efectivamente, observamos que los malos olores críticos de CitaController han desaparecido después de la refactorización del código, con lo que podemos concluir que la refactorización ha sido realizada de forma efectiva.

Tras realizar este refactor, los datos proporcionados por SonarCloud sobre fiabilidad y mantenibilidad son los siguientes:

Reliability

Bugs

Reliability Rating

Reliability Remediation Effort

1

C

10min

Maintainability

Code Smells

Effort to Reach Maintainability Rating A

Maintainability Rating

Technical Debt

Technical Debt Ratio

4

0

A

10min

0.2%

Conclusiones finales:

Acorde a la información proporcionada por SonarCloud, se resolvieron un total de 160 code smells y bugs:

Unresolved	465	False Positive	0
Fixed	160	Removed	0
Won't Fix	0		

Se puede concluir que las refactorizaciones detalladas en este documento han sido efectivos quedando finalmente los siguientes datos de mantenibilidad:

Overall	
Code Smells	408
Debt	2d 4h
Debt Ratio	1.2%
Rating	A
Effort to Reach A	0