Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática

ANALYSIS REPORT INDIVIDUAL STUDENT 4

Grado en Ingeniería Informática – Ingeniería del Software

Diseño y Pruebas II

Curso 2022 – 2023

Fecha	Versión
<26/05/2023>	v1r1

Grupo de Prácticas: C1.01.01	
Repositorio: https://github.com/sebasruii/Acme-L3-D04	
Autores por orden alfabético	**Correo**
Pérez Romero, Lucía	lucperrom1@alum.us.es

Índice de contenido

1. Resumen Ejecutivo

En este documento se presenta el resultado del proceso de testing de los requisitos 14 y 15.

2. Tabla de revisiones

Fecha	Revisión	Descripción
<13/05/2023	v1r1	Versión final

3. Introducción

Este documento tiene como objetivo proporcionar los resultados clave tras el proceso de testing de las clases pertenecientes a las carpetas feature.company.practicum y feature.company.practicumSession en ese orden.

Primero, por cada apartado se mostrarán capturas en las que se visualice la cobertura por cada clase de manera detallada, así como un breve texto explicativo al final del mismo indicando lo que se están viendo en las mismas.

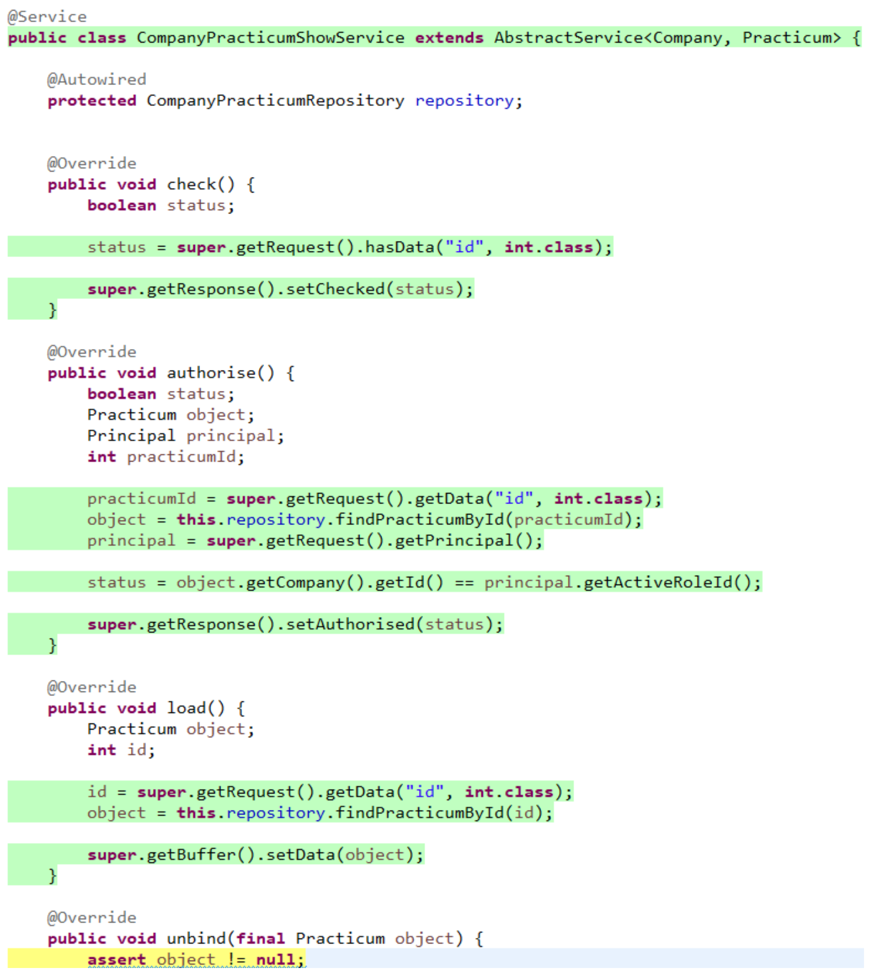
Finalmente, se proporcionará una conclusión basada en lo expuesto anteriormente.

4. Contenido

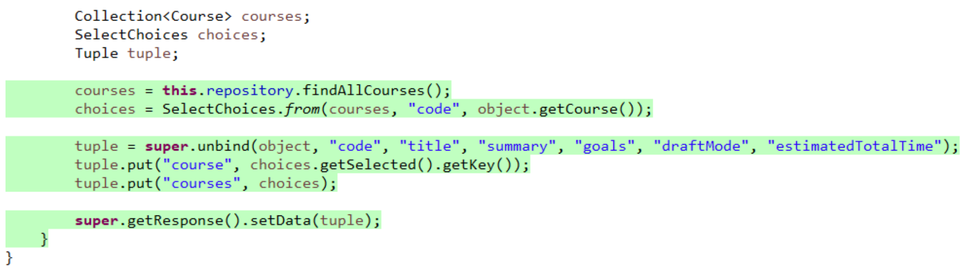
4.1. CompanyPracticumTest

```java
@Service
public class CompanyPracticumListService extends AbstractService<Company, Practicum> {

    @Autowired
    protected CompanyPracticumRepository repository;


    @Override
    public void check() {
        super.getResponse().setChecked(true);
    }

    @Override
    public void authorise() {
        super.getResponse().setAuthorised(true);
    }

    @Override
    public void load() {
        Collection<Practicum> objects;
        Principal principal;

        principal = super.getRequest().getPrincipal();
        objects = this.repository.findPracticaByCompanyId(principal.getActiveRoleId());

        super.getBuffer().setData(objects);
    }

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

        Tuple tuple;

        tuple = super.unbind(object, "code", "title");
        tuple.put("courseCode", object.getCourse().getCode());

        super.getResponse().setData(tuple);
    }

}
```

CompanyPracticumListService.java

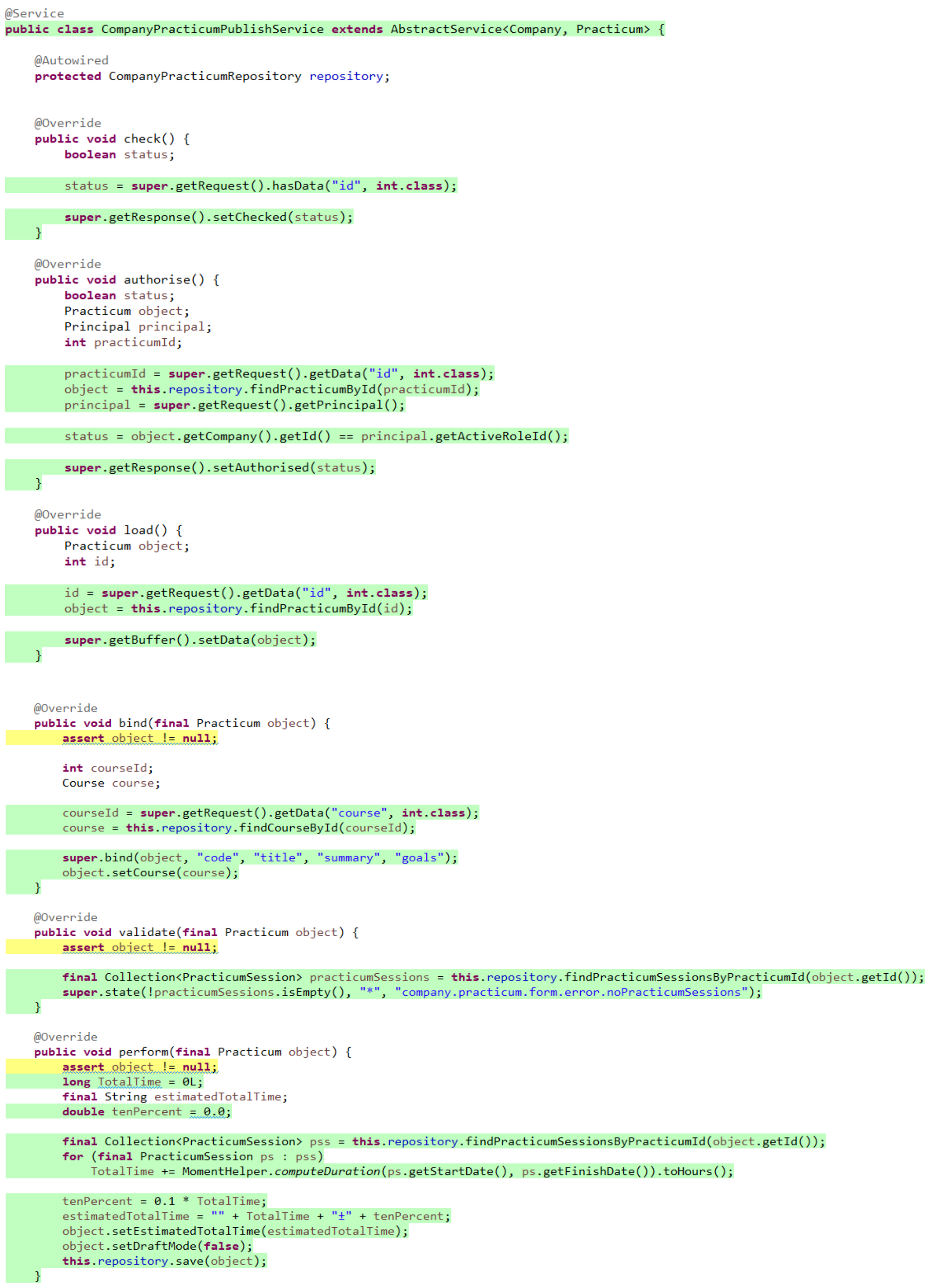


```
@Service
public class CompanyPracticumShowService extends AbstractService<Company, Practicum> {

    @Autowired
    protected CompanyPracticumRepository repository;


    @Override
    public void check() {
        boolean status;

        status = super.getRequest().hasData("id", int.class);

        super.getResponse().setChecked(status);
    }

    @Override
    public void authorise() {
        boolean status;
        Practicum object;
        Principal principal;
        int practicumId;

        practicumId = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumById(practicumId);
        principal = super.getRequest().getPrincipal();

        status = object.getCompany().getId() == principal.getActiveRoleId();

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

    @Override
    public void load() {
        Practicum object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumById(id);

        super.getBuffer().setData(object);
    }

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

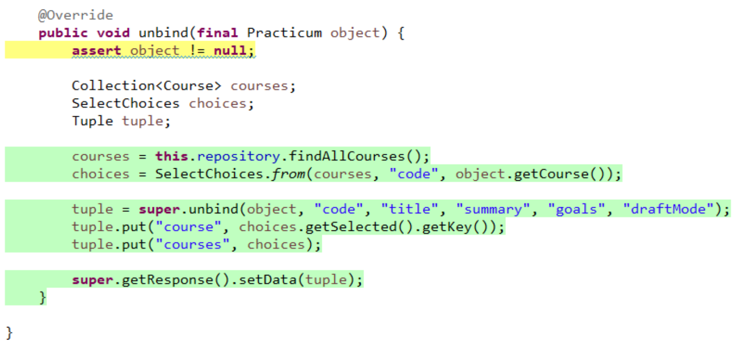


```
        Collection<Course> courses;
        SelectChoices choices;
        Tuple tuple;

        courses = this.repository.findAllCourses();
        choices = SelectChoices.from(courses, "code", object.getCourse());

        tuple = super.unbind(object, "code", "title", "summary", "goals", "draftMode", "estimatedTotalTime");
        tuple.put("course", choices.getSelected().getKey());
        tuple.put("courses", choices);

        super.getResponse().setData(tuple);
    }
}
```

CompanyPracticumShowService.java

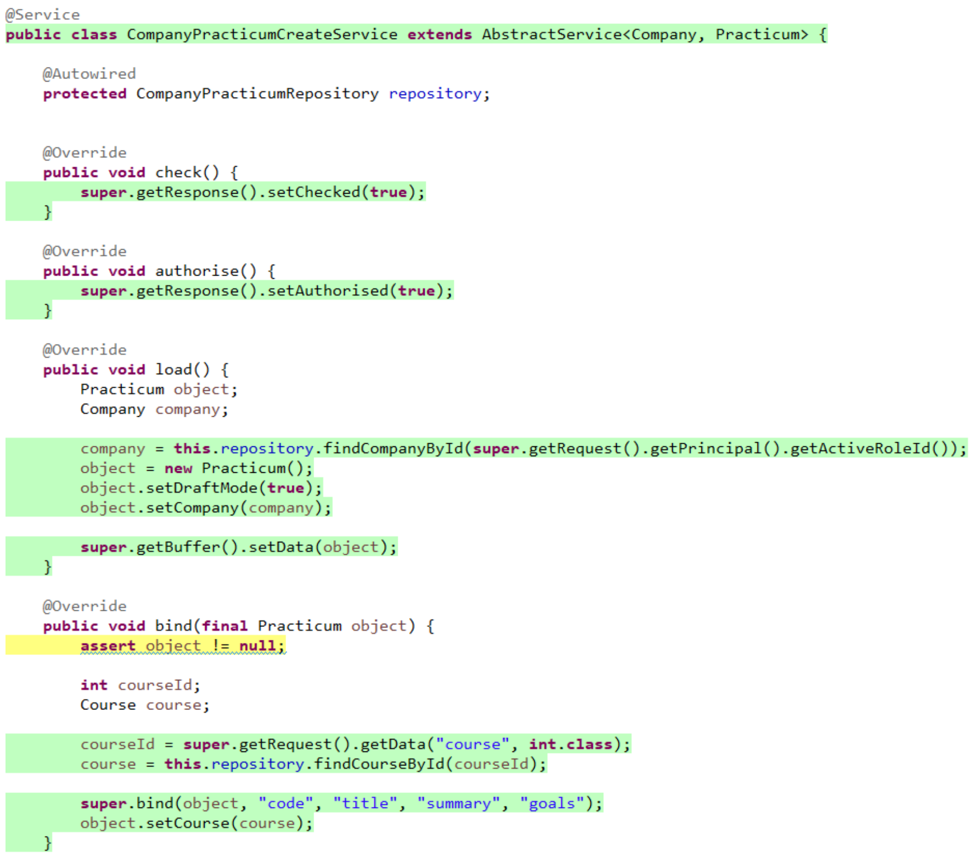


```
@Service
public class CompanyPracticumPublishService extends AbstractService<Company, Practicum> {

    @Autowired
    protected CompanyPracticumRepository repository;


    @Override
    public void check() {
        boolean status;

        status = super.getRequest().hasData("id", int.class);

        super.getResponse().setChecked(status);
    }

    @Override
    public void authorise() {
        boolean status;
        Practicum object;
        Principal principal;
        int practicumId;

        practicumId = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumById(practicumId);
        principal = super.getRequest().getPrincipal();

        status = object.getCompany().getId() == principal.getActiveRoleId();

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

    @Override
    public void load() {
        Practicum object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumById(id);

        super.getBuffer().setData(object);
    }


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

        int courseId;
        Course course;

        courseId = super.getRequest().getData("course", int.class);
        course = this.repository.findCourseById(courseId);

        super.bind(object, "code", "title", "summary", "goals");
        object.setCourse(course);
    }

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

        final Collection<PracticumSession> practicumSessions = this.repository.findPracticumSessionsByPracticumId(object.getId());
        super.state(!practicumSessions.isEmpty(), "*", "company.practicum.form.error.noPracticumSessions");
    }

    @Override
    public void perform(final Practicum object) {
        assert object != null;
        long TotalTime = 0L;
        final String estimatedTotalTime;
        double tenPercent = 0.0;

        final Collection<PracticumSession> pss = this.repository.findPracticumSessionsByPracticumId(object.getId());
        for (final PracticumSession ps : pss)
            TotalTime += MomentHelper.computeDuration(ps.getStartDate(), ps.getFinishDate()).toHours();

        tenPercent = 0.1 * TotalTime;
        estimatedTotalTime = "" + TotalTime + "±" + tenPercent;
        object.setEstimatedTotalTime(estimatedTotalTime);
        object.setDraftMode(false);
        this.repository.save(object);
    }
```

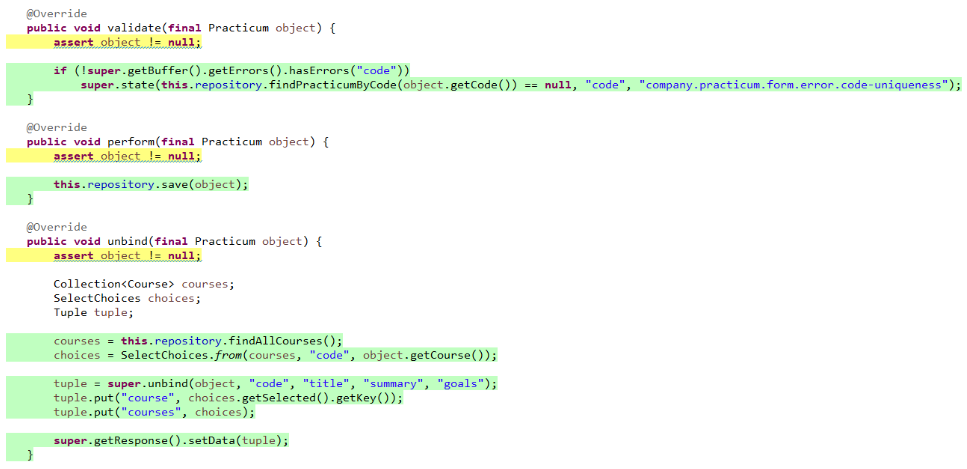


```
    @Override
    public void unbind(final Practicum object) {
        assert object != null;

        Collection<Course> courses;
        SelectChoices choices;
        Tuple tuple;

        courses = this.repository.findAllCourses();
        choices = SelectChoices.from(courses, "code", object.getCourse());

        tuple = super.unbind(object, "code", "title", "summary", "goals", "draftMode");
        tuple.put("course", choices.getSelected().getKey());
        tuple.put("courses", choices);

        super.getResponse().setData(tuple);
    }

}
```

CompanyPracticumPublishService.java

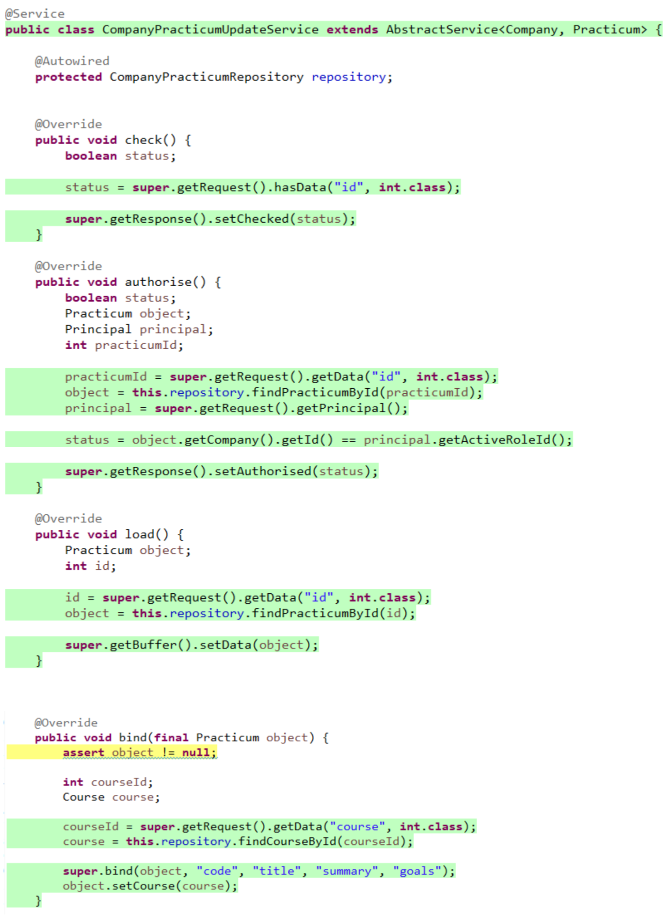


```
@Service
public class CompanyPracticumCreateService extends AbstractService<Company, Practicum> {

    @Autowired
    protected CompanyPracticumRepository repository;


    @Override
    public void check() {
        super.getResponse().setChecked(true);
    }

    @Override
    public void authorise() {
        super.getResponse().setAuthorised(true);
    }

    @Override
    public void load() {
        Practicum object;
        Company company;

        company = this.repository.findCompanyById(super.getRequest().getPrincipal().getActiveRoleId());
        object = new Practicum();
        object.setDraftMode(true);
        object.setCompany(company);

        super.getBuffer().setData(object);
    }

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

        int courseId;
        Course course;

        courseId = super.getRequest().getData("course", int.class);
        course = this.repository.findCourseById(courseId);

        super.bind(object, "code", "title", "summary", "goals");
        object.setCourse(course);
    }
```

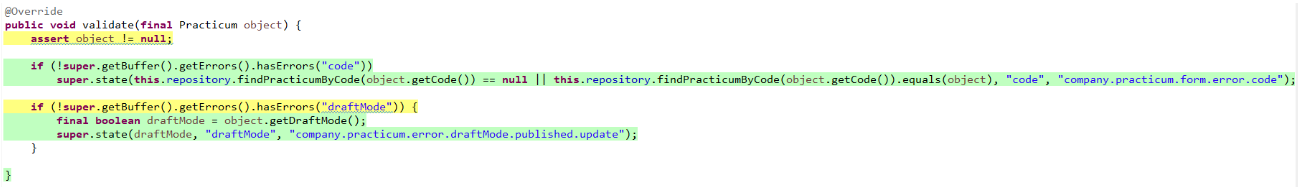


```java
    @Override
    public void validate(final Practicum object) {
        assert object != null;

        if (!super.getBuffer().getErrors().hasErrors("code"))
            super.state(this.repository.findPracticumByCode(object.getCode()) == null, "code", "company.practicum.form.error.code-uniqueness");
    }

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

        this.repository.save(object);
    }

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

        Collection<Course> courses;
        SelectChoices choices;
        Tuple tuple;

        courses = this.repository.findAllCourses();
        choices = SelectChoices.from(courses, "code", object.getCourse());

        tuple = super.unbind(object, "code", "title", "summary", "goals");
        tuple.put("course", choices.getSelected().getKey());
        tuple.put("courses", choices);

        super.getResponse().setData(tuple);
    }
```

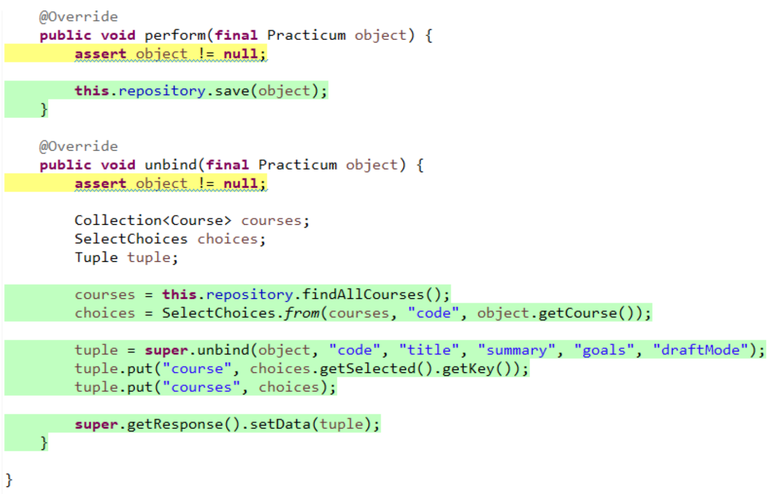
CompanyPracticumCreateService.java



```java
@Service
public class CompanyPracticumUpdateService extends AbstractService<Company, Practicum> {

    @Autowired
    protected CompanyPracticumRepository repository;


    @Override
    public void check() {
        boolean status;

        status = super.getRequest().hasData("id", int.class);

        super.getResponse().setChecked(status);
    }

    @Override
    public void authorise() {
        boolean status;
        Practicum object;
        Principal principal;
        int practicumId;

        practicumId = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumById(practicumId);
        principal = super.getRequest().getPrincipal();

        status = object.getCompany().getId() == principal.getActiveRoleId();

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

    @Override
    public void load() {
        Practicum object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumById(id);

        super.getBuffer().setData(object);
    }


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

        int courseId;
        Course course;

        courseId = super.getRequest().getData("course", int.class);
        course = this.repository.findCourseById(courseId);

        super.bind(object, "code", "title", "summary", "goals");
        object.setCourse(course);
    }
```

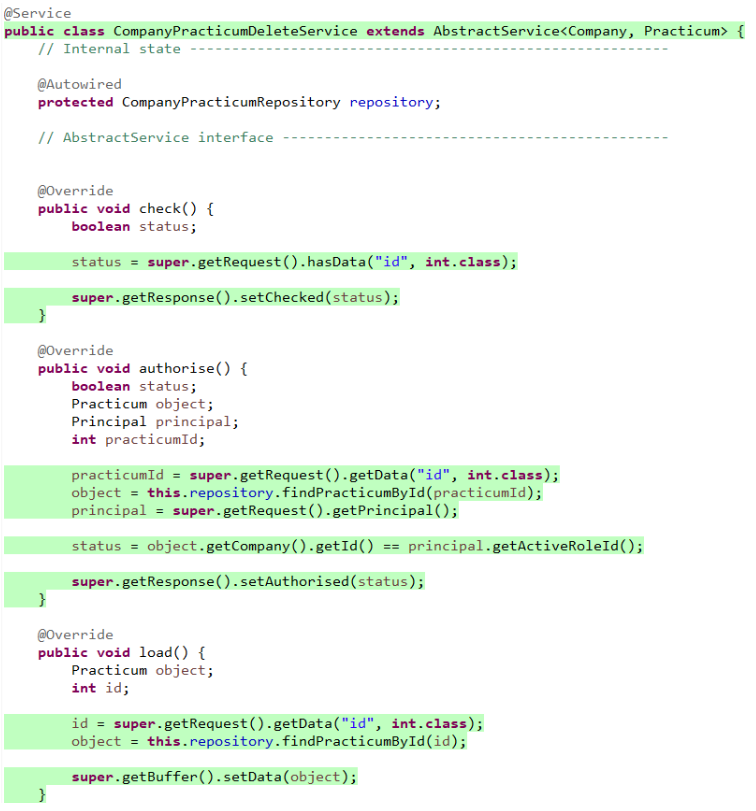


```java
@Override
public void validate(final Practicum object) {
    assert object != null;

    if (!super.getBuffer().getErrors().hasErrors("code"))
        super.state(this.repository.findPracticumByCode(object.getCode()) == null || this.repository.findPracticumByCode(object.getCode()).equals(object), "code", "company.practicum.form.error.code");

    if (!super.getBuffer().getErrors().hasErrors("draftMode")) {
        final boolean draftMode = object.getDraftMode();
        super.state(draftMode, "draftMode", "company.practicum.error.draftMode.published.update");
    }

}
```

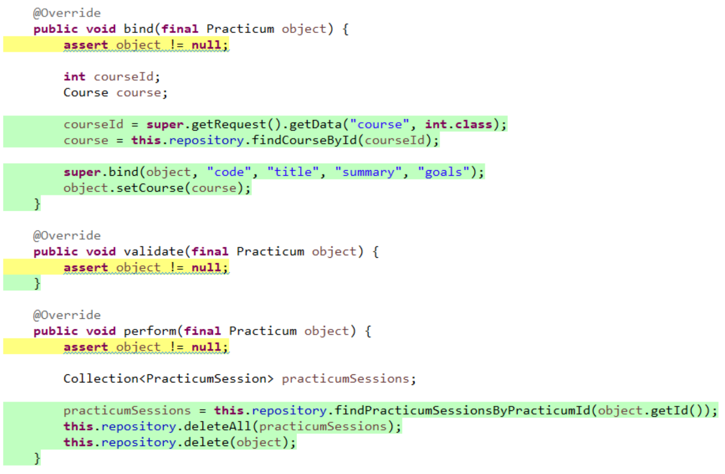


```java
    @Override
    public void perform(final Practicum object) {
        assert object != null;

        this.repository.save(object);
    }

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

        Collection<Course> courses;
        SelectChoices choices;
        Tuple tuple;

        courses = this.repository.findAllCourses();
        choices = SelectChoices.from(courses, "code", object.getCourse());

        tuple = super.unbind(object, "code", "title", "summary", "goals", "draftMode");
        tuple.put("course", choices.getSelected().getKey());
        tuple.put("courses", choices);

        super.getResponse().setData(tuple);
    }

}
```

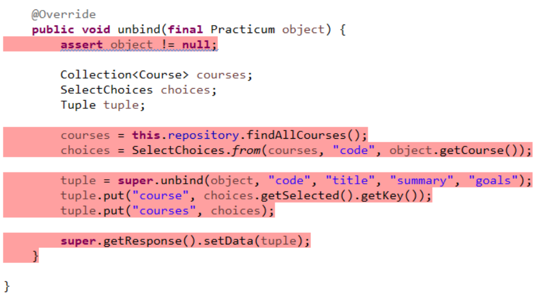
CompanyPracticumUpdateService.java



```java
@Service
public class CompanyPracticumDeleteService extends AbstractService<Company, Practicum> {
    // Internal state ---------------------------------------------------------

    @Autowired
    protected CompanyPracticumRepository repository;

    // AbstractService interface ---------------------------------------------


    @Override
    public void check() {
        boolean status;

        status = super.getRequest().hasData("id", int.class);

        super.getResponse().setChecked(status);
    }

    @Override
    public void authorise() {
        boolean status;
        Practicum object;
        Principal principal;
        int practicumId;

        practicumId = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumById(practicumId);
        principal = super.getRequest().getPrincipal();

        status = object.getCompany().getId() == principal.getActiveRoleId();

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

    @Override
    public void load() {
        Practicum object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumById(id);

        super.getBuffer().setData(object);
    }
```



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

		int courseId;
		Course course;

		courseId = super.getRequest().getData("course", int.class);
		course = this.repository.findCourseById(courseId);

		super.bind(object, "code", "title", "summary", "goals");
		object.setCourse(course);
	}

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

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

		Collection<PracticumSession> practicumSessions;

		practicumSessions = this.repository.findPracticumSessionsByPracticumId(object.getId());
		this.repository.deleteAll(practicumSessions);
		this.repository.delete(object);
	}
```



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

		Collection<Course> courses;
		SelectChoices choices;
		Tuple tuple;

		courses = this.repository.findAllCourses();
		choices = SelectChoices.from(courses, "code", object.getCourse());

		tuple = super.unbind(object, "code", "title", "summary", "goals");
		tuple.put("course", choices.getSelected().getKey());
		tuple.put("courses", choices);

		super.getResponse().setData(tuple);
	}

}
```

CompanyPracticumDeleteService.java

- **¿Qué estamos viendo?**

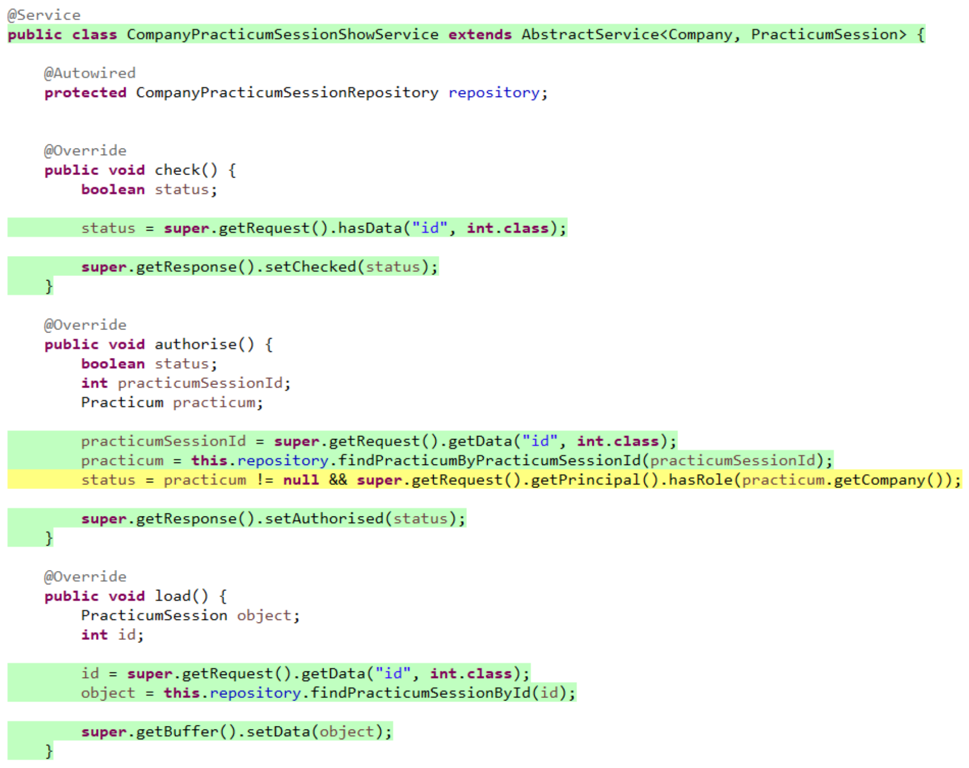
Color amarillo: entra solo en una de las dos condiciones. De este color tenemos:

- "*assert object!=null*" que se aprecia en todas las clases y que resulta imposible de testear porque el framework no devuelve un objeto nulo
- *"If (!super.getBuffer().getErrors().hasErrors("draftmode"))"* que se localiza en CompanyPracticumUpdateService.java

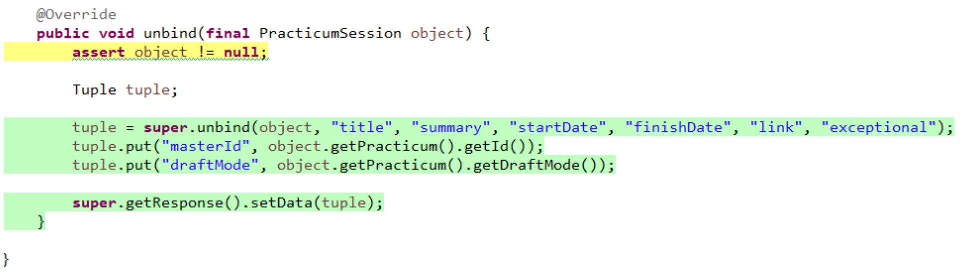
Color rojo: los tests no está testeando esas líneas. De este color tenemos:

- Método unbind del CompanyPracticumDeleteService.java. Este es imposible de testear ya que no hay caso negativo de eliminar una práctica

4.2. CompanyPracticumSessionTest

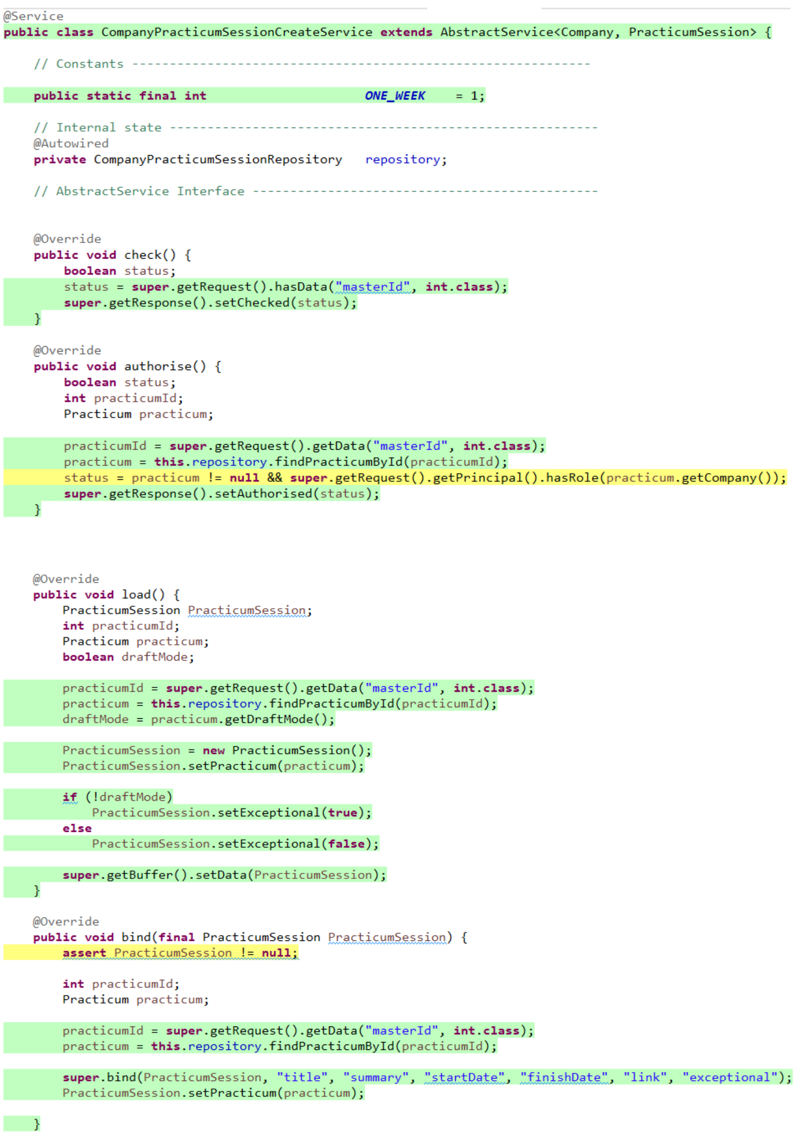


```
@Service
public class CompanyPracticumSessionShowService extends AbstractService<Company, PracticumSession> {

    @Autowired
    protected CompanyPracticumSessionRepository repository;


    @Override
    public void check() {
        boolean status;

        status = super.getRequest().hasData("id", int.class);

        super.getResponse().setChecked(status);
    }

    @Override
    public void authorise() {
        boolean status;
        int practicumSessionId;
        Practicum practicum;

        practicumSessionId = super.getRequest().getData("id", int.class);
        practicum = this.repository.findPracticumByPracticumSessionId(practicumSessionId);
        status = practicum != null && super.getRequest().getPrincipal().hasRole(practicum.getCompany());

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

    @Override
    public void load() {
        PracticumSession object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumSessionById(id);

        super.getBuffer().setData(object);
    }
```

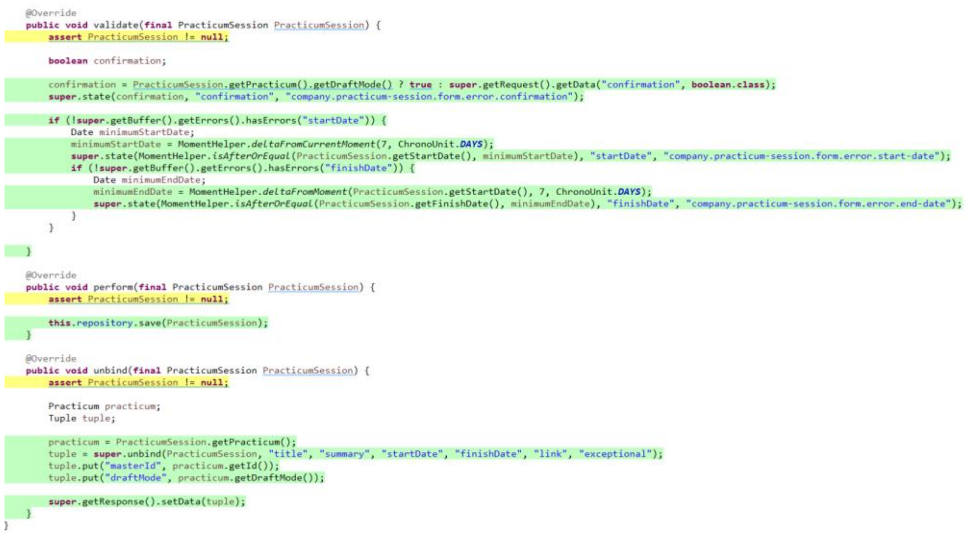


```
    @Override
    public void unbind(final PracticumSession object) {
        assert object != null;

        Tuple tuple;

        tuple = super.unbind(object, "title", "summary", "startDate", "finishDate", "link", "exceptional");
        tuple.put("masterId", object.getPracticum().getId());
        tuple.put("draftMode", object.getPracticum().getDraftMode());

        super.getResponse().setData(tuple);
    }

}
```

CompanyPracticumSessionShowService.java

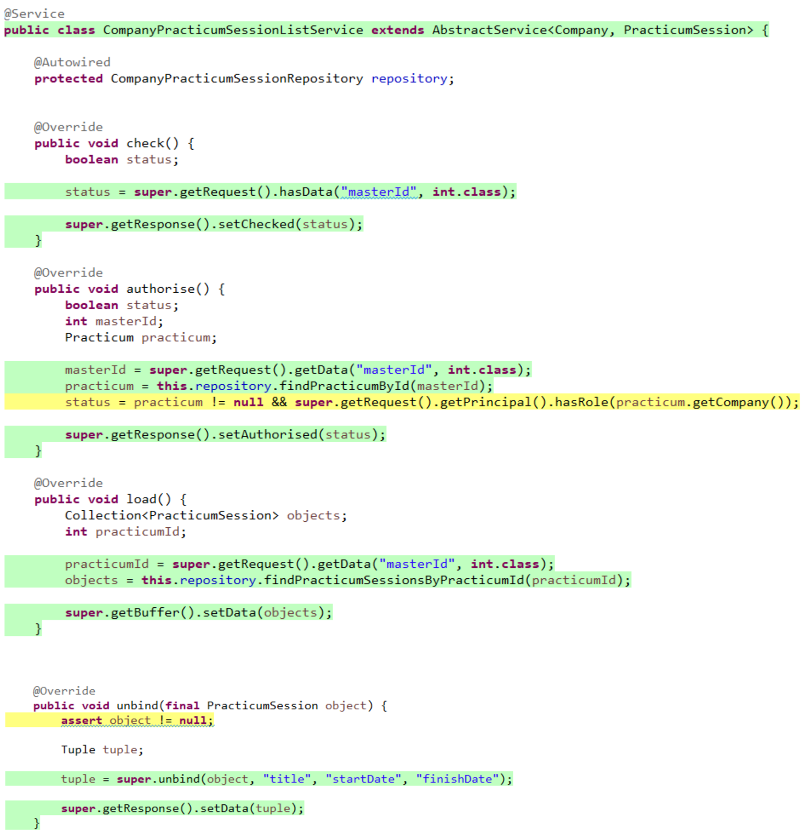


```
@Service
public class CompanyPracticumSessionCreateService extends AbstractService<Company, PracticumSession> {

    // Constants -------------------------------------------------------------

    public static final int                         ONE_WEEK    = 1;

    // Internal state ---------------------------------------------------------
    @Autowired
    private CompanyPracticumSessionRepository    repository;

    // AbstractService Interface ----------------------------------------------


    @Override
    public void check() {
        boolean status;
        status = super.getRequest().hasData("masterId", int.class);
        super.getResponse().setChecked(status);
    }

    @Override
    public void authorise() {
        boolean status;
        int practicumId;
        Practicum practicum;

        practicumId = super.getRequest().getData("masterId", int.class);
        practicum = this.repository.findPracticumById(practicumId);
        status = practicum != null && super.getRequest().getPrincipal().hasRole(practicum.getCompany());
        super.getResponse().setAuthorised(status);
    }


    @Override
    public void load() {
        PracticumSession PracticumSession;
        int practicumId;
        Practicum practicum;
        boolean draftMode;

        practicumId = super.getRequest().getData("masterId", int.class);
        practicum = this.repository.findPracticumById(practicumId);
        draftMode = practicum.getDraftMode();

        PracticumSession = new PracticumSession();
        PracticumSession.setPracticum(practicum);

        if (!draftMode)
            PracticumSession.setExceptional(true);
        else
            PracticumSession.setExceptional(false);

        super.getBuffer().setData(PracticumSession);
    }

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

        int practicumId;
        Practicum practicum;

        practicumId = super.getRequest().getData("masterId", int.class);
        practicum = this.repository.findPracticumById(practicumId);

        super.bind(PracticumSession, "title", "summary", "startDate", "finishDate", "link", "exceptional");
        PracticumSession.setPracticum(practicum);

    }
```

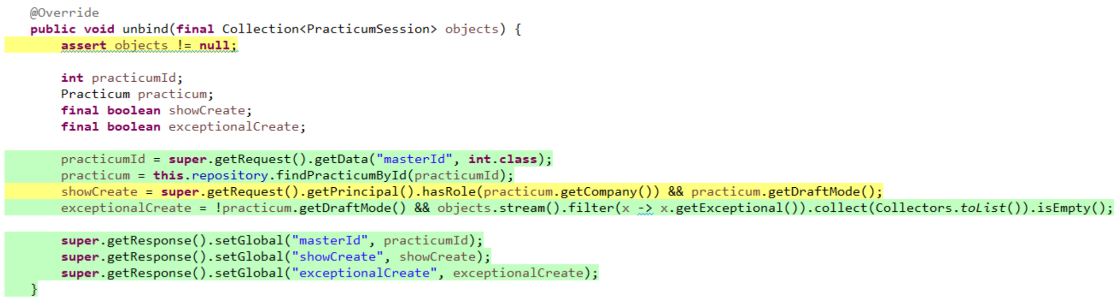


```
    @Override
    public void validate(final PracticumSession PracticumSession) {
        assert PracticumSession != null;

        boolean confirmation;

        confirmation = PracticumSession.getPracticum().getDraftMode() ? true : super.getRequest().getData("confirmation", boolean.class);
        super.state(confirmation, "confirmation", "company.practicum-session.form.error.confirmation");

        if (!super.getBuffer().getErrors().hasErrors("startDate")) {
            Date minimumStartDate;
            minimumStartDate = MomentHelper.deltaFromCurrentMoment(7, ChronoUnit.DAYS);
            super.state(MomentHelper.isAfterOrEqual(PracticumSession.getStartDate(), minimumStartDate), "startDate", "company.practicum-session.form.error.start-date");
            if (!super.getBuffer().getErrors().hasErrors("finishDate")) {
                Date minimumEndDate;
                minimumEndDate = MomentHelper.deltaFromMoment(PracticumSession.getStartDate(), 7, ChronoUnit.DAYS);
                super.state(MomentHelper.isAfterOrEqual(PracticumSession.getFinishDate(), minimumEndDate), "finishDate", "company.practicum-session.form.error.end-date");
            }
        }

    }

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

        this.repository.save(PracticumSession);
    }

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

        Practicum practicum;
        Tuple tuple;

        practicum = PracticumSession.getPracticum();
        tuple = super.unbind(PracticumSession, "title", "summary", "startDate", "finishDate", "link", "exceptional");
        tuple.put("masterId", practicum.getId());
        tuple.put("draftMode", practicum.getDraftMode());

        super.getResponse().setData(tuple);
    }
}
```

CompanyPracticumSessionCreateService.java

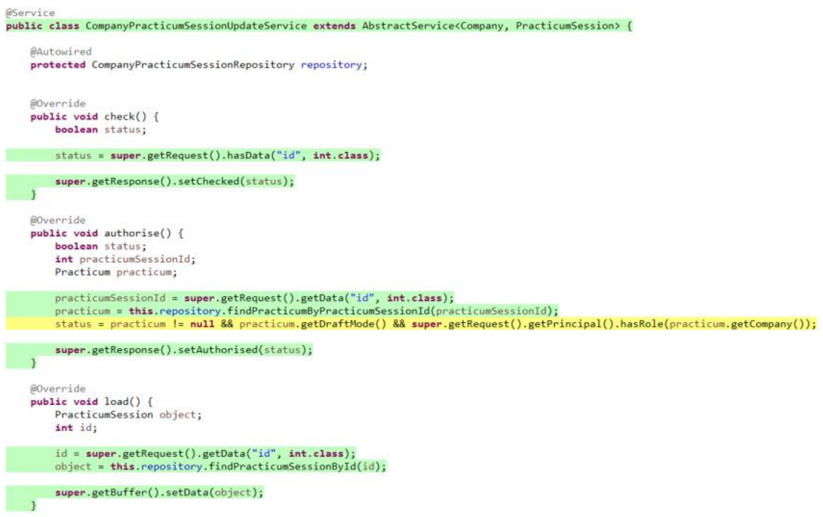


```
@Service
public class CompanyPracticumSessionListService extends AbstractService<Company, PracticumSession> {

    @Autowired
    protected CompanyPracticumSessionRepository repository;


    @Override
    public void check() {
        boolean status;

        status = super.getRequest().hasData("masterId", int.class);

        super.getResponse().setChecked(status);
    }

    @Override
    public void authorise() {
        boolean status;
        int masterId;
        Practicum practicum;

        masterId = super.getRequest().getData("masterId", int.class);
        practicum = this.repository.findPracticumById(masterId);
        status = practicum != null && super.getRequest().getPrincipal().hasRole(practicum.getCompany());

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

    @Override
    public void load() {
        Collection<PracticumSession> objects;
        int practicumId;

        practicumId = super.getRequest().getData("masterId", int.class);
        objects = this.repository.findPracticumSessionsByPracticumId(practicumId);

        super.getBuffer().setData(objects);
    }


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

        Tuple tuple;

        tuple = super.unbind(object, "title", "startDate", "finishDate");

        super.getResponse().setData(tuple);
    }
```

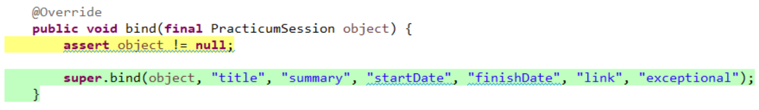


```
    @Override
    public void unbind(final Collection<PracticumSession> objects) {
        assert objects != null;

        int practicumId;
        Practicum practicum;
        final boolean showCreate;
        final boolean exceptionalCreate;

        practicumId = super.getRequest().getData("masterId", int.class);
        practicum = this.repository.findPracticumById(practicumId);
        showCreate = super.getRequest().getPrincipal().hasRole(practicum.getCompany()) && practicum.getDraftMode();
        exceptionalCreate = !practicum.getDraftMode() && objects.stream().filter(x -> x.getExceptional()).collect(Collectors.toList()).isEmpty();

        super.getResponse().setGlobal("masterId", practicumId);
        super.getResponse().setGlobal("showCreate", showCreate);
        super.getResponse().setGlobal("exceptionalCreate", exceptionalCreate);
    }
```

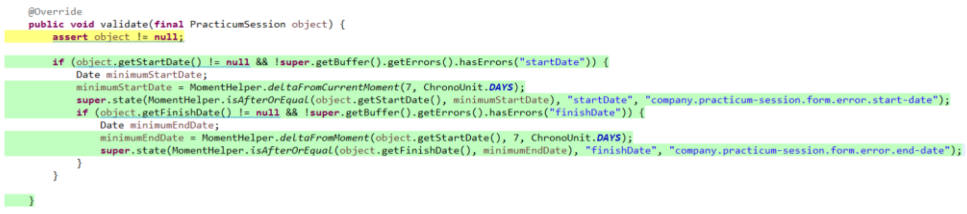
CompanyPracticumSessionListService.java



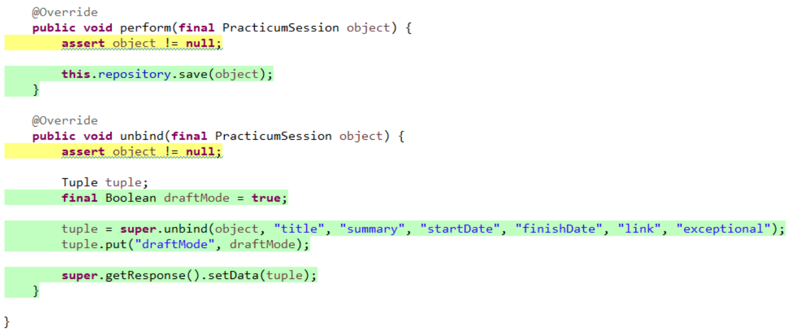
```
@Service
public class CompanyPracticumSessionUpdateService extends AbstractService<Company, PracticumSession> {

    @Autowired
    protected CompanyPracticumSessionRepository repository;


    @Override
    public void check() {
        boolean status;

        status = super.getRequest().hasData("id", int.class);

        super.getResponse().setChecked(status);
    }

    @Override
    public void authorise() {
        boolean status;
        int practicumSessionId;
        Practicum practicum;

        practicumSessionId = super.getRequest().getData("id", int.class);
        practicum = this.repository.findPracticumByPracticumSessionId(practicumSessionId);
        status = practicum != null && practicum.getDraftMode() && super.getRequest().getPrincipal().hasRole(practicum.getCompany());

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

    @Override
    public void load() {
        PracticumSession object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumSessionById(id);

        super.getBuffer().setData(object);
    }
```



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

        super.bind(object, "title", "summary", "startDate", "finishDate", "link", "exceptional");
    }
```

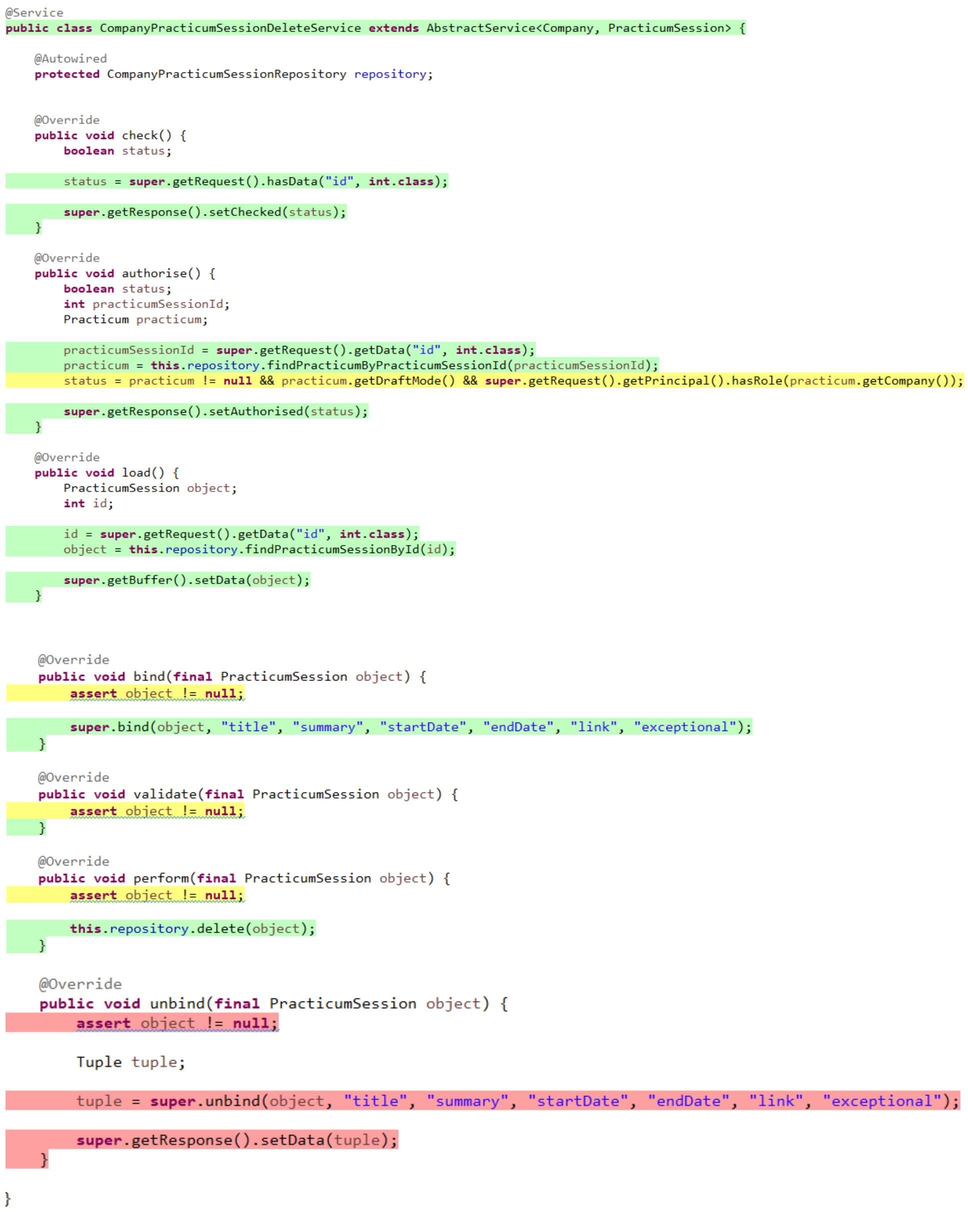


```
    @Override
    public void validate(final PracticumSession object) {
        assert object != null;

        if (object.getStartDate() != null && !super.getBuffer().getErrors().hasErrors("startDate")) {
            Date minimumStartDate;
            minimumStartDate = MomentHelper.deltaFromCurrentMoment(7, ChronoUnit.DAYS);
            super.state(MomentHelper.isAfterOrEqual(object.getStartDate(), minimumStartDate), "startDate", "company.practicum-session.form.error.start-date");
            if (object.getFinishDate() != null && !super.getBuffer().getErrors().hasErrors("finishDate")) {
                Date minimumEndDate;
                minimumEndDate = MomentHelper.deltaFromMoment(object.getStartDate(), 7, ChronoUnit.DAYS);
                super.state(MomentHelper.isAfterOrEqual(object.getFinishDate(), minimumEndDate), "finishDate", "company.practicum-session.form.error.end-date");
            }
        }

    }
```



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

        this.repository.save(object);
    }

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

        Tuple tuple;
        final Boolean draftMode = true;

        tuple = super.unbind(object, "title", "summary", "startDate", "finishDate", "link", "exceptional");
        tuple.put("draftMode", draftMode);

        super.getResponse().setData(tuple);
    }

}
```

CompanyPracticumSessionUpdate.java



```
@Service
public class CompanyPracticumSessionDeleteService extends AbstractService<Company, PracticumSession> {

    @Autowired
    protected CompanyPracticumSessionRepository repository;


    @Override
    public void check() {
        boolean status;

        status = super.getRequest().hasData("id", int.class);

        super.getResponse().setChecked(status);
    }

    @Override
    public void authorise() {
        boolean status;
        int practicumSessionId;
        Practicum practicum;

        practicumSessionId = super.getRequest().getData("id", int.class);
        practicum = this.repository.findPracticumByPracticumSessionId(practicumSessionId);
        status = practicum != null && practicum.getDraftMode() && super.getRequest().getPrincipal().hasRole(practicum.getCompany());

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

    @Override
    public void load() {
        PracticumSession object;
        int id;

        id = super.getRequest().getData("id", int.class);
        object = this.repository.findPracticumSessionById(id);

        super.getBuffer().setData(object);
    }


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

        super.bind(object, "title", "summary", "startDate", "endDate", "link", "exceptional");
    }

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

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

        this.repository.delete(object);
    }

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

        Tuple tuple;

        tuple = super.unbind(object, "title", "summary", "startDate", "endDate", "link", "exceptional");

        super.getResponse().setData(tuple);
    }

}
```

CompanyPracticumSessionDelete.java

- **¿Qué estamos viendo?**

Color amarillo: entra solo en una de las dos condiciones. De este color tenemos:

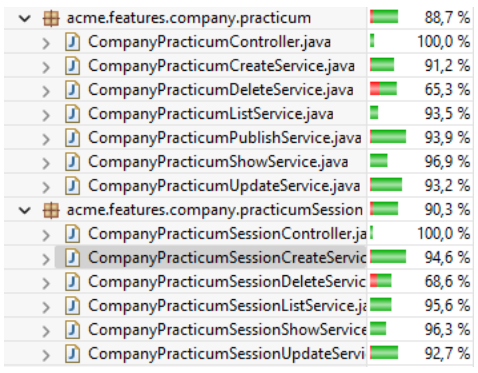
- "*assert object!=null" o "assert PracticumSession!=null"* que se aprecia en todas las clases y que resulta imposible de testear porque el framework no devuelve un objeto nulo.
- *"status = practicum != null && super.getRequest().getPrincipal().hasRole(Practicum.getCompany())"* que se encuentra en todas las clases y que resulta imposible de testear más.
- La iniciación de "showCreate" que se localiza en CompanyPracticumSessionListService.java

Color rojo: los tests no está testeando esas líneas. De este color tenemos:

- Método unbind del CompanyPracticumSessionDeleteService.java. Estos no se pueden testear ya que no existen casos negativos de eliminar una sesión de prácticas

5. Conclusiones

En resumen, considero que los tests tanto de CompanyPracticum como de CompanyPracticumSession tienen una alta cobertura y consiguen abarcar casi la totalidad del código exigido.



acme.features.company.practicum	88,7 %
CompanyPracticumController.java	100,0 %
CompanyPracticumCreateService.java	91,2 %
CompanyPracticumDeleteService.java	65,3 %
CompanyPracticumListService.java	93,5 %
CompanyPracticumPublishService.java	93,9 %
CompanyPracticumShowService.java	96,9 %
CompanyPracticumUpdateService.java	93,2 %
acme.features.company.practicumSession	90,3 %
CompanyPracticumSessionController.ja	100,0 %
CompanyPracticumSessionCreateServic	94,6 %
CompanyPracticumSessionDeleteServic	68,6 %
CompanyPracticumSessionListService.j	95,6 %
CompanyPracticumSessionShowService	96,3 %
CompanyPracticumSessionUpdateServi	92,7 %

6. Bibliografía

intencionalmente en blanco.