Test Report- D04

INTEGRANTES DEL GRUPO C1.026:

- Ignacio Blanquero Blanco (ignblabla@alum.us.es)
- Adrián Cabello Martín (adrcabmar@alum.us.es)
- María de la Salud Carrera Talaverón (marcartal1@alum.us.es)
- Joaquín González Ganfornina (joagongan@alum.us.es)
- Natalia Olmo Villegas (natolmvil@alum.us.es)

FECHA: Sevilla, 27 de mayo 2024

Tabla de contenido

RESUMEN DEL INFORME

Este informe se divide en dos secciones principales: testing funcional, donde se proporciona un listado detallado de los casos de prueba implementados, organizados por características del sistema. Cada caso de prueba incluye una descripción concisa y una evaluación de su efectividad en la detección de errores; y análisis del rendimiento, donde se presentan gráficos que ilustran el tiempo de respuesta (wall time) del sistema al ejecutar los tests funcionales en dos computadoras diferentes, así como los resultados obtenidos con índices y sin índices. Se calcula un intervalo de confianza del 95% para estos tiempos y se determinará cuál de las dos computadoras es más poderosa en términos de rendimiento al servir las solicitudes del sistema.

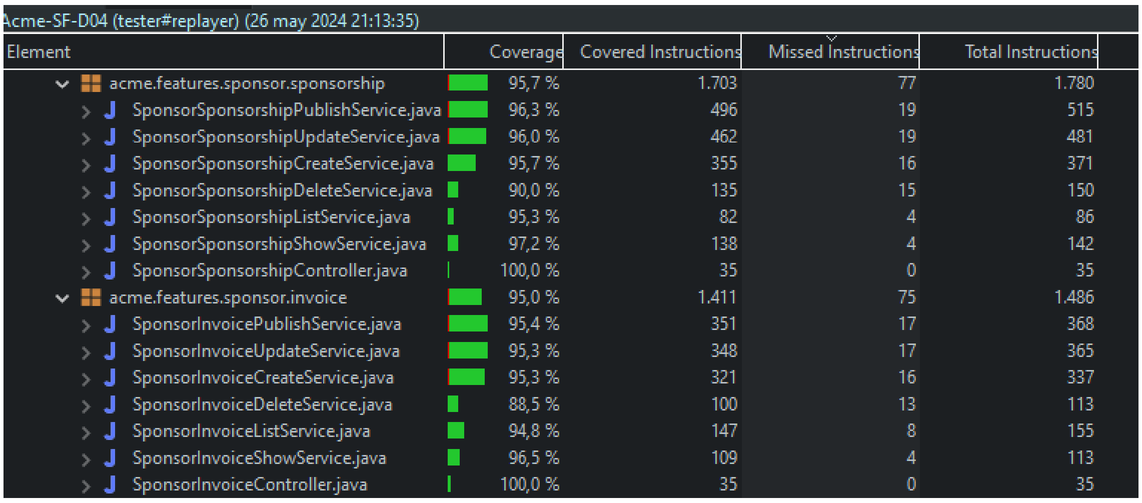
HISTORIAL DE VERSIONES

Versión	**Contenidos**	**Fecha**	**Contribuyente**
V.1.0	Documento al completo	27/05/2024	Adrian Cabello Martin

INTRODUCCION

El presente informe detalla el desarrollo y resultados de un conjunto de pruebas de software (test suite) aplicadas al proyecto desarrollado por este grupo Acme-SF. El objetivo principal es evaluar tanto la funcionalidad como el rendimiento del sistema bajo prueba. Para lograr esto, se han implementado diversas pruebas funcionales agrupadas por características, cada una diseñada para detectar errores y asegurar que las funcionalidades del sistema operen conforme a lo esperado. Además, se realiza un análisis del rendimiento, comparando el tiempo de respuesta del sistema en dos computadoras diferentes, con el fin de determinar cuál de ellas ofrece un mejor desempeño

COBERTURA

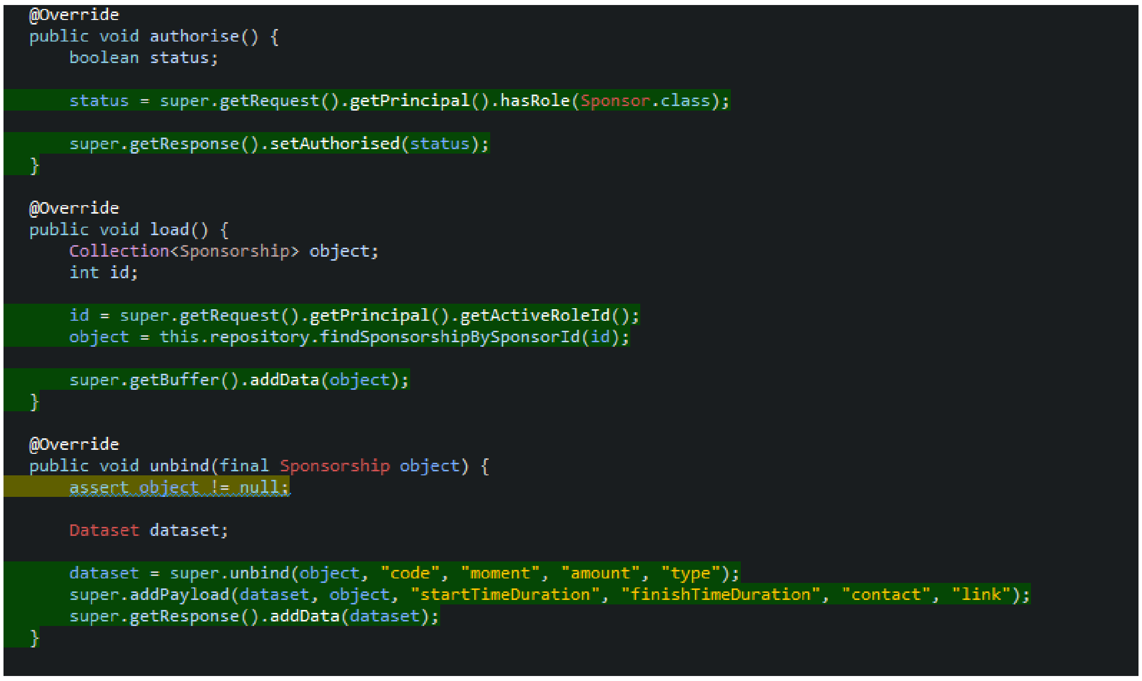


Acme-SF-D04 (tester#replayer) (26 may 2024 21:13:35)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
acme.features.sponsor.sponsorship	95,7 %	1.703	77	1.780
SponsorSponsorshipPublishService.java	96,3 %	496	19	515
SponsorSponsorshipUpdateService.java	96,0 %	462	19	481
SponsorSponsorshipCreateService.java	95,7 %	355	16	371
SponsorSponsorshipDeleteService.java	90,0 %	135	15	150
SponsorSponsorshipListService.java	95,3 %	82	4	86
SponsorSponsorshipShowService.java	97,2 %	138	4	142
SponsorSponsorshipController.java	100,0 %	35	0	35
acme.features.sponsor.invoice	95,0 %	1.411	75	1.486
SponsorInvoicePublishService.java	95,4 %	351	17	368
SponsorInvoiceUpdateService.java	95,3 %	348	17	365
SponsorInvoiceCreateService.java	95,3 %	321	16	337
SponsorInvoiceDeleteService.java	88,5 %	100	13	113
SponsorInvoiceListService.java	94,8 %	147	8	155
SponsorInvoiceShowService.java	96,5 %	109	4	113
SponsorInvoiceController.java	100,0 %	35	0	35

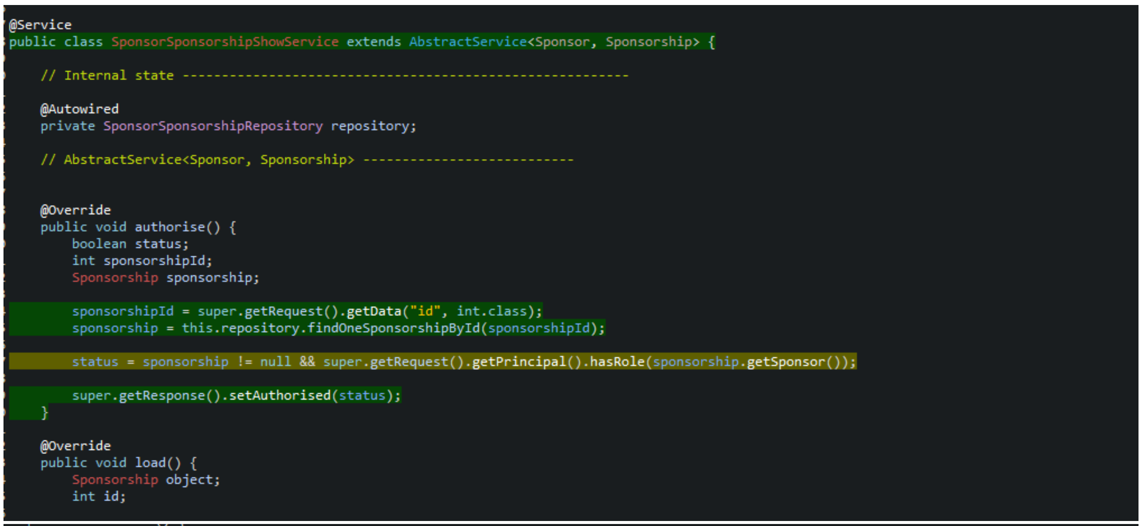
Se ha logrado alcanzar un gran porcentaje de cobertura como es un 95%, donde las líneas amarillas de código son de los assert object y de los status del authorise ya que se puede ver que estaban planteadas para un requisito opcional como es que se pudieran ver los publicados siendo de otro rol, pero este requisito al final no se desarrolló. Aun así, se ha conseguido un gran porcentaje de cobertura.

SponsorSponsorshipListService.java

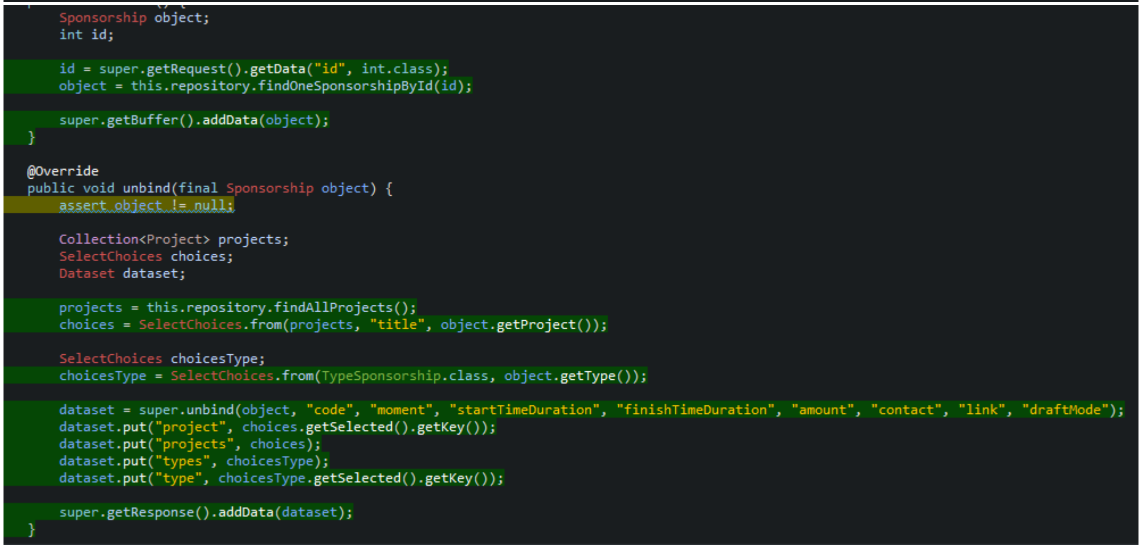


```
@Override
public void authorise() {
    boolean status;

    status = super.getRequest().getPrincipal().hasRole(Sponsor.class);

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

@Override
public void load() {
    Collection<Sponsorship> object;
    int id;

    id = super.getRequest().getPrincipal().getActiveRoleId();
    object = this.repository.findSponsorshipBySponsorId(id);

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

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

    Dataset dataset;

    dataset = super.unbind(object, "code", "moment", "amount", "type");
    super.addPayload(dataset, object, "startTimeDuration", "finishTimeDuration", "contact", "link");
    super.getResponse().addData(dataset);
}
```

SponsorSponsorshipShowService.java

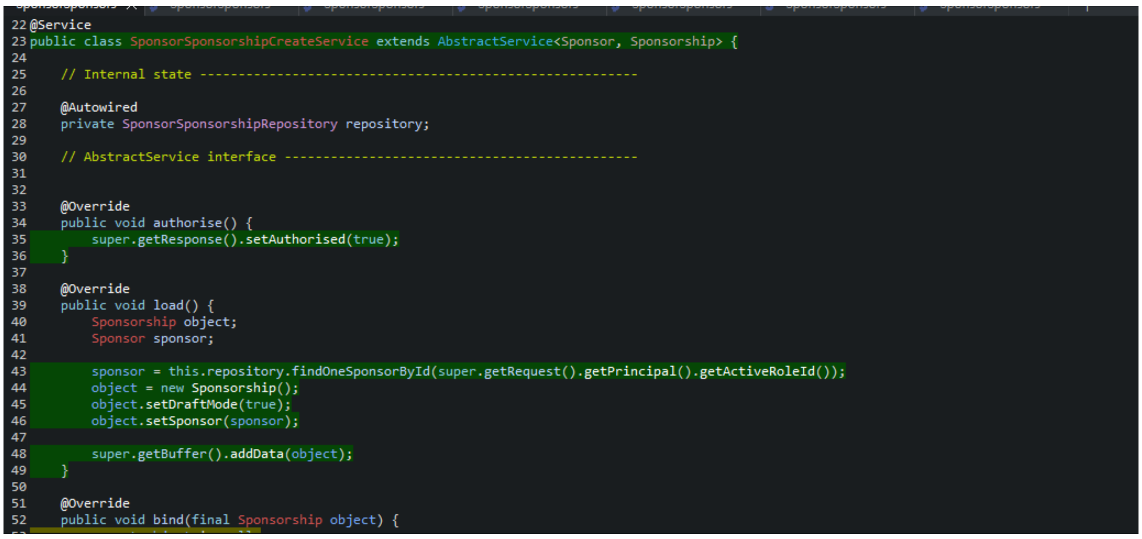


```java
@Service
public class SponsorSponsorshipShowService extends AbstractService<Sponsor, Sponsorship> {

    // Internal state ---------------------------------------------------------

    @Autowired
    private SponsorSponsorshipRepository repository;

    // AbstractService<Sponsor, Sponsorship> --------------------------


    @Override
    public void authorise() {
        boolean status;
        int sponsorshipId;
        Sponsorship sponsorship;

        sponsorshipId = super.getRequest().getData("id", int.class);
        sponsorship = this.repository.findOneSponsorshipById(sponsorshipId);

        status = sponsorship != null && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());

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

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

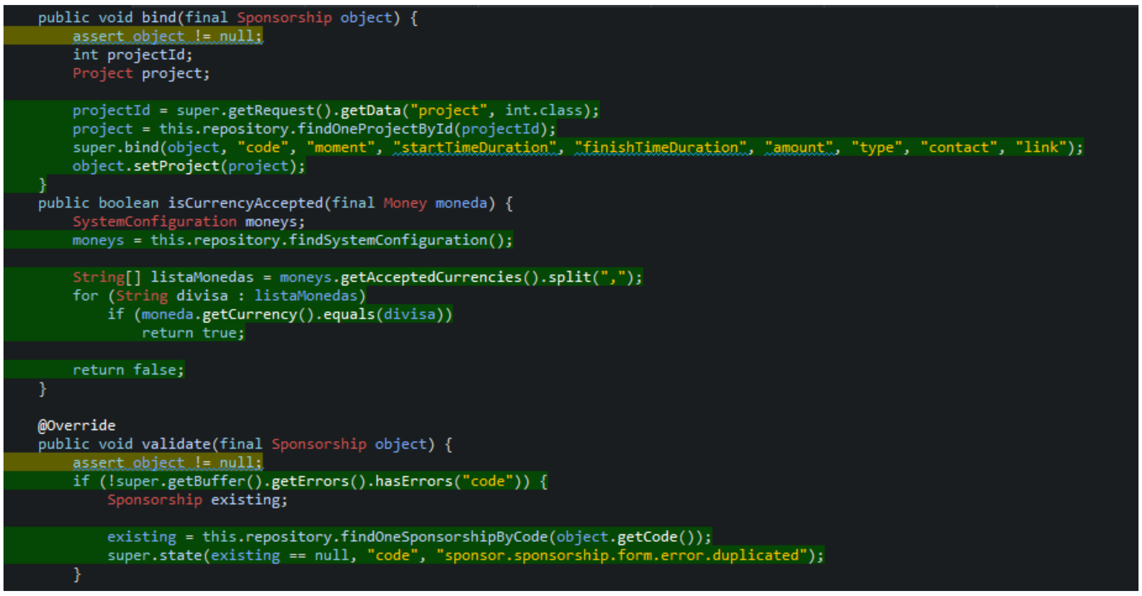


```java
        Sponsorship object;
        int id;

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

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

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

        Collection<Project> projects;
        SelectChoices choices;
        Dataset dataset;

        projects = this.repository.findAllProjects();
        choices = SelectChoices.from(projects, "title", object.getProject());

        SelectChoices choicesType;
        choicesType = SelectChoices.from(TypeSponsorship.class, object.getType());

        dataset = super.unbind(object, "code", "moment", "startTimeDuration", "finishTimeDuration", "amount", "contact", "link", "draftMode");
        dataset.put("project", choices.getSelected().getKey());
        dataset.put("projects", choices);
        dataset.put("types", choicesType);
        dataset.put("type", choicesType.getSelected().getKey());

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

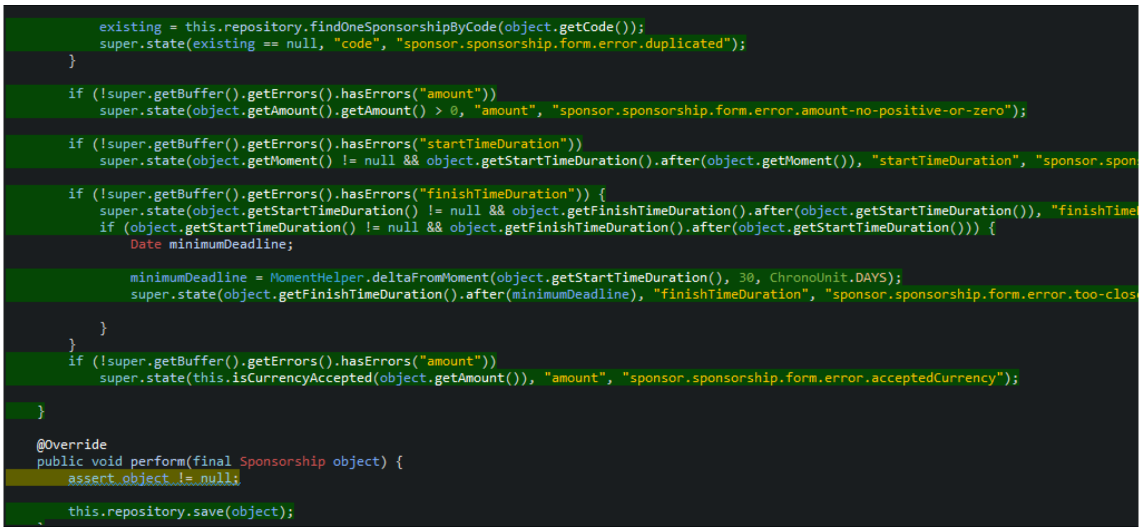
SponsorSponsorshipCreateService.java



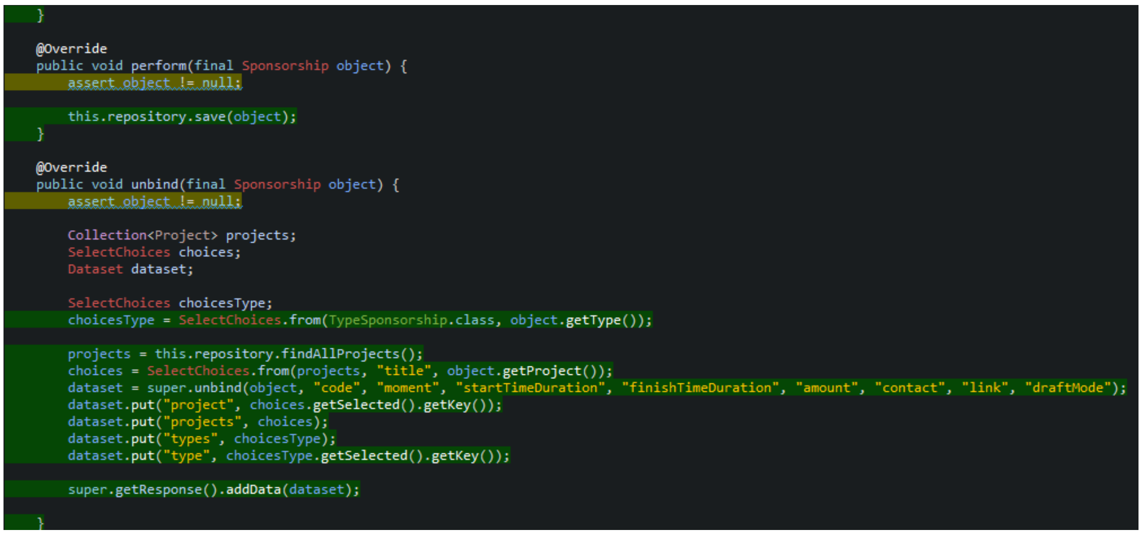
```java
@Service
public class SponsorSponsorshipCreateService extends AbstractService<Sponsor, Sponsorship> {

    // Internal state ---------------------------------------------------------

    @Autowired
    private SponsorSponsorshipRepository repository;

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


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

    @Override
    public void load() {
        Sponsorship object;
        Sponsor sponsor;

        sponsor = this.repository.findOneSponsorById(super.getRequest().getPrincipal().getActiveRoleId());
        object = new Sponsorship();
        object.setDraftMode(true);
        object.setSponsor(sponsor);

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

    @Override
    public void bind(final Sponsorship object) {
```



```java
public void bind(final Sponsorship object) {
    assert object != null;
    int projectId;
    Project project;

    projectId = super.getRequest().getData("project", int.class);
    project = this.repository.findOneProjectById(projectId);
    super.bind(object, "code", "moment", "startTimeDuration", "finishTimeDuration", "amount", "type", "contact", "link");
    object.setProject(project);
}
public boolean isCurrencyAccepted(final Money moneda) {
    SystemConfiguration moneys;
    moneys = this.repository.findSystemConfiguration();

    String[] listaMonedas = moneys.getAcceptedCurrencies().split(",");
    for (String divisa : listaMonedas)
        if (moneda.getCurrency().equals(divisa))
            return true;

    return false;
}

@Override
public void validate(final Sponsorship object) {
    assert object != null;
    if (!super.getBuffer().getErrors().hasErrors("code")) {
        Sponsorship existing;

        existing = this.repository.findOneSponsorshipByCode(object.getCode());
        super.state(existing == null, "code", "sponsor.sponsorship.form.error.duplicated");
    }
```

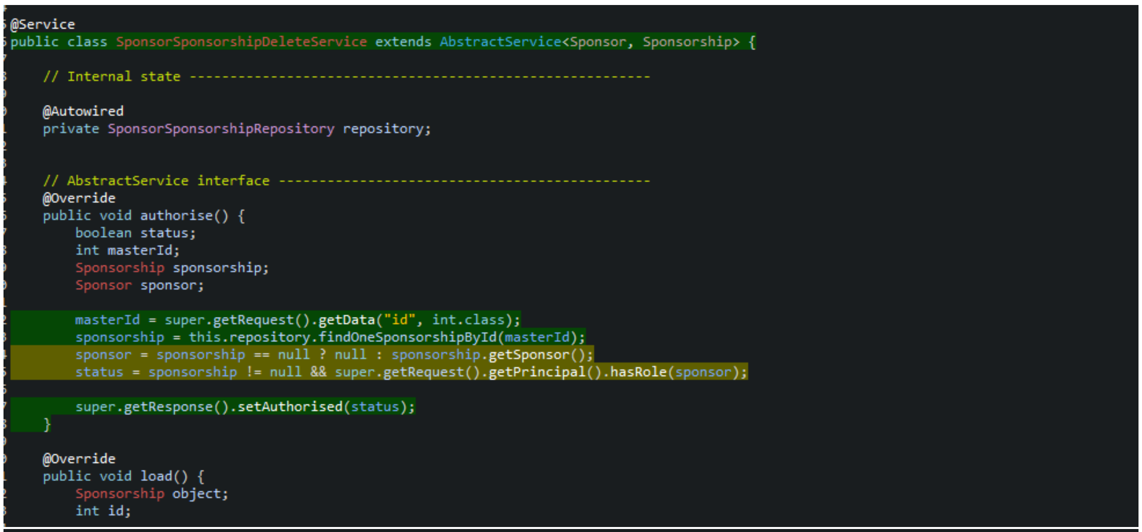


```java
            existing = this.repository.findOneSponsorshipByCode(object.getCode());
            super.state(existing == null, "code", "sponsor.sponsorship.form.error.duplicated");
        }

        if (!super.getBuffer().getErrors().hasErrors("amount"))
            super.state(object.getAmount().getAmount() > 0, "amount", "sponsor.sponsorship.form.error.amount-no-positive-or-zero");

        if (!super.getBuffer().getErrors().hasErrors("startTimeDuration"))
            super.state(object.getMoment() != null && object.getStartTimeDuration().after(object.getMoment()), "startTimeDuration", "sponsor.spon

        if (!super.getBuffer().getErrors().hasErrors("finishTimeDuration")) {
            super.state(object.getStartTimeDuration() != null && object.getFinishTimeDuration().after(object.getStartTimeDuration()), "finishTime
            if (object.getStartTimeDuration() != null && object.getFinishTimeDuration().after(object.getStartTimeDuration())) {
                Date minimumDeadline;

                minimumDeadline = MomentHelper.deltaFromMoment(object.getStartTimeDuration(), 30, ChronoUnit.DAYS);
                super.state(object.getFinishTimeDuration().after(minimumDeadline), "finishTimeDuration", "sponsor.sponsorship.form.error.too-clos

            }
        }
        if (!super.getBuffer().getErrors().hasErrors("amount"))
            super.state(this.isCurrencyAccepted(object.getAmount()), "amount", "sponsor.sponsorship.form.error.acceptedCurrency");

    }

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

        this.repository.save(object);
```

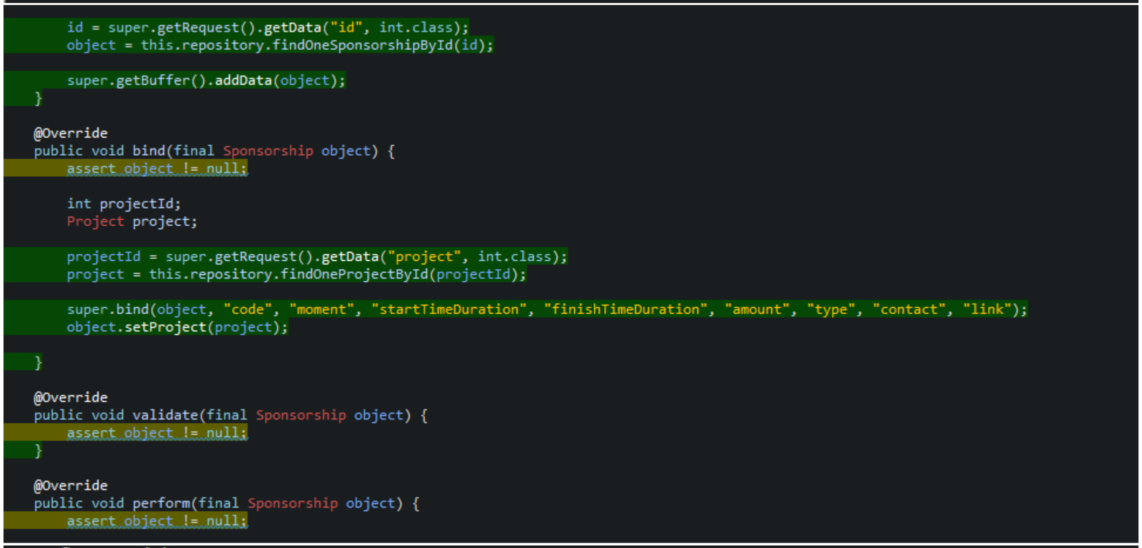


```java
    }

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

        this.repository.save(object);
    }

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

        Collection<Project> projects;
        SelectChoices choices;
        Dataset dataset;

        SelectChoices choicesType;
        choicesType = SelectChoices.from(TypeSponsorship.class, object.getType());

        projects = this.repository.findAllProjects();
        choices = SelectChoices.from(projects, "title", object.getProject());
        dataset = super.unbind(object, "code", "moment", "startTimeDuration", "finishTimeDuration", "amount", "contact", "link", "draftMode");
        dataset.put("project", choices.getSelected().getKey());
        dataset.put("projects", choices);
        dataset.put("types", choicesType);
        dataset.put("type", choicesType.getSelected().getKey());

        super.getResponse().addData(dataset);

    }
```

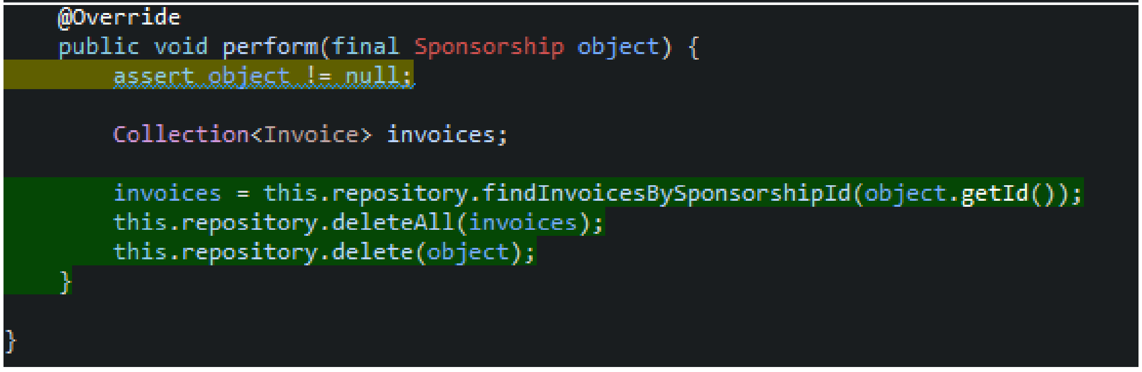
SponsorSponsorshipDeleteService.java



```java
@Service
public class SponsorSponsorshipDeleteService extends AbstractService<Sponsor, Sponsorship> {

    // Internal state ---------------------------------------------------------

    @Autowired
    private SponsorSponsorshipRepository repository;


    // AbstractService interface ----------------------------------------------
    @Override
    public void authorise() {
        boolean status;
        int masterId;
        Sponsorship sponsorship;
        Sponsor sponsor;

        masterId = super.getRequest().getData("id", int.class);
        sponsorship = this.repository.findOneSponsorshipById(masterId);
        sponsor = sponsorship == null ? null : sponsorship.getSponsor();
        status = sponsorship != null && super.getRequest().getPrincipal().hasRole(sponsor);

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

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

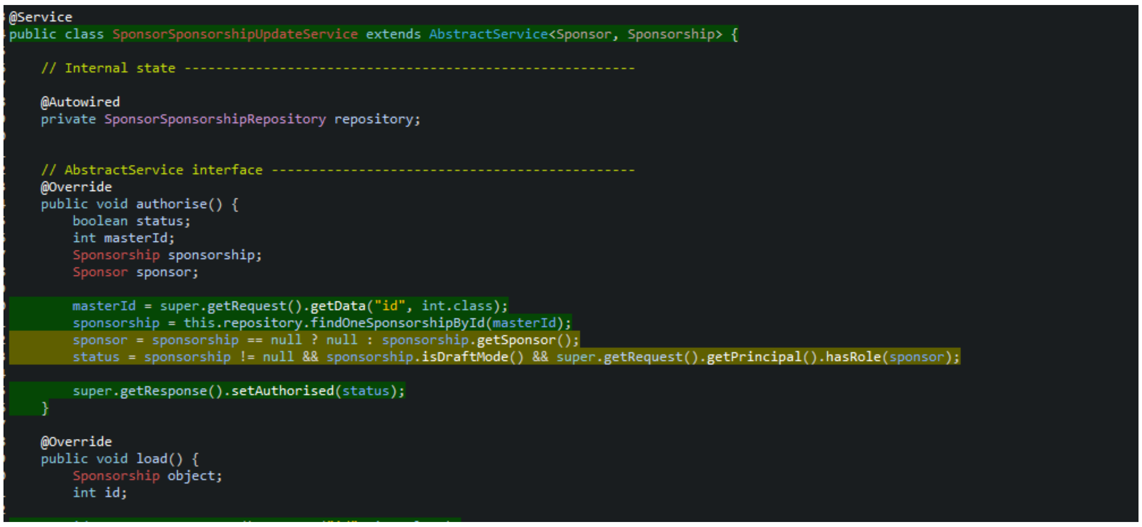


```java
        id = super.getRequest().getData("id", int.class);
        object = this.repository.findOneSponsorshipById(id);

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

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

        int projectId;
        Project project;

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

        super.bind(object, "code", "moment", "startTimeDuration", "finishTimeDuration", "amount", "type", "contact", "link");
        object.setProject(project);

    }

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

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

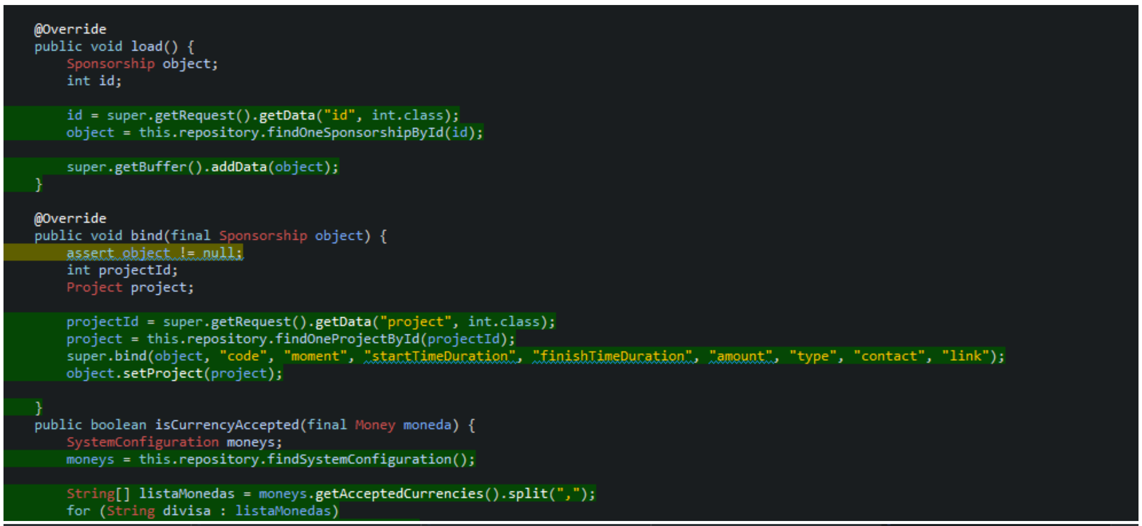


```java
    @Override
    public void perform(final Sponsorship object) {
        assert object != null;

        Collection<Invoice> invoices;

        invoices = this.repository.findInvoicesBySponsorshipId(object.getId());
        this.repository.deleteAll(invoices);
        this.repository.delete(object);
    }

}
```

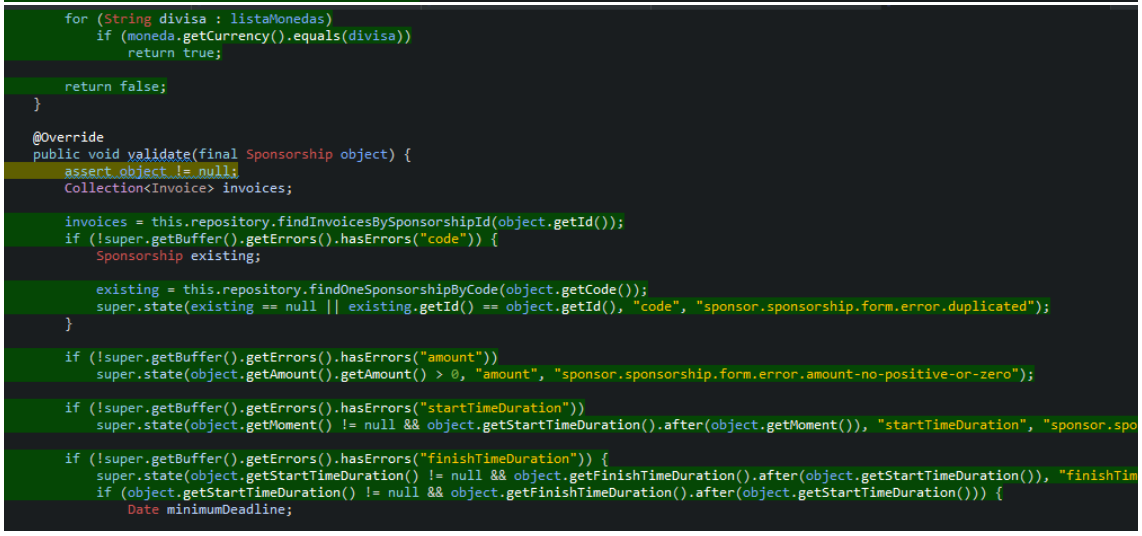
SponsorSponsorshipUpdateService.java



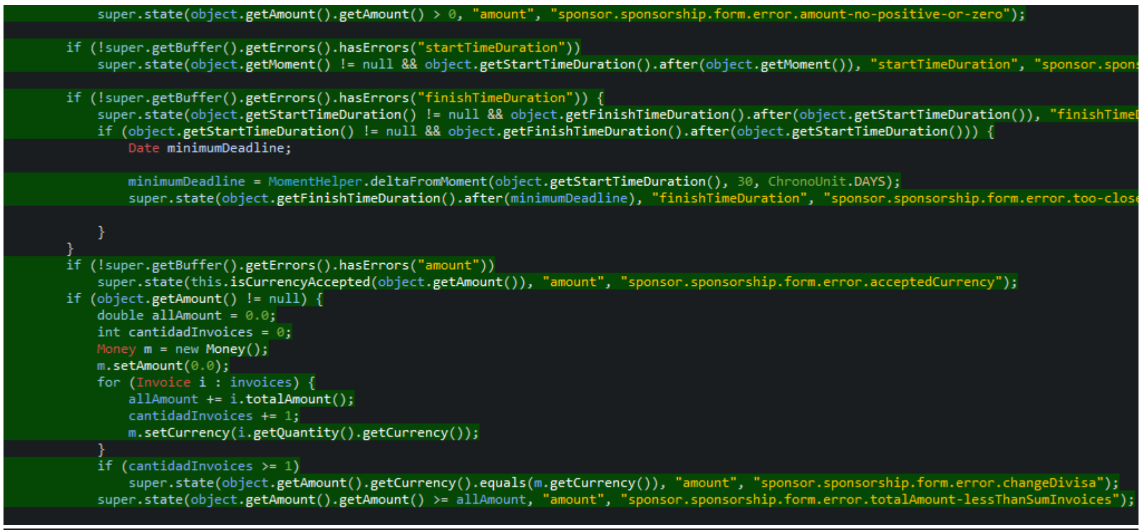
```java
@Service
public class SponsorSponsorshipUpdateService extends AbstractService<Sponsor, Sponsorship> {

    // Internal state ---------------------------------------------------------

    @Autowired
    private SponsorSponsorshipRepository repository;


    // AbstractService interface ----------------------------------------------
    @Override
    public void authorise() {
        boolean status;
        int masterId;
        Sponsorship sponsorship;
        Sponsor sponsor;

        masterId = super.getRequest().getData("id", int.class);
        sponsorship = this.repository.findOneSponsorshipById(masterId);
        sponsor = sponsorship == null ? null : sponsorship.getSponsor();
        status = sponsorship != null && sponsorship.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsor);

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

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



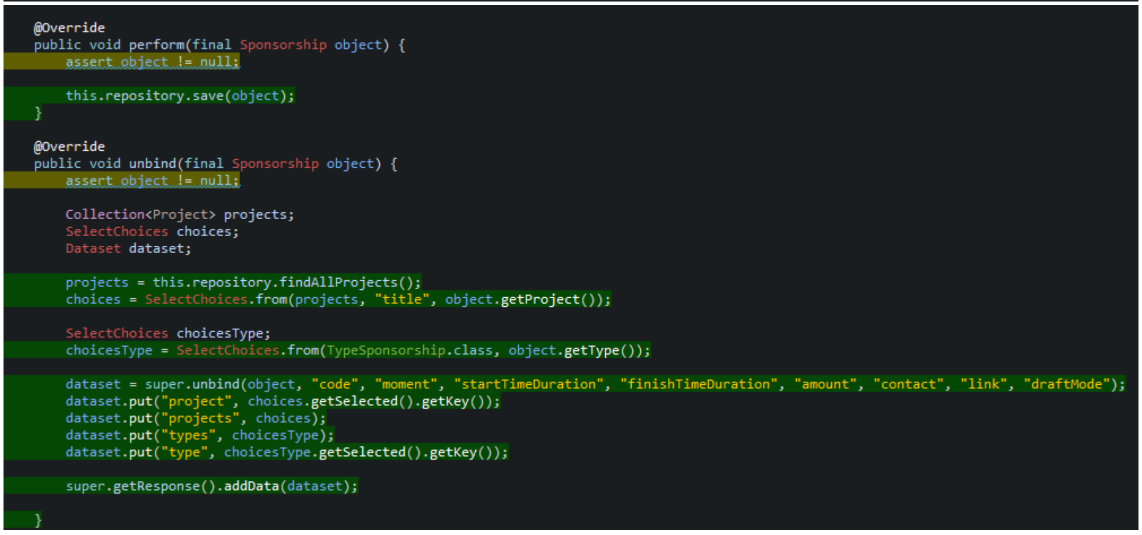
```java
    @Override
    public void load() {
        Sponsorship object;
        int id;

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

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

    @Override
    public void bind(final Sponsorship object) {
        assert object != null;
        int projectId;
        Project project;

        projectId = super.getRequest().getData("project", int.class);
        project = this.repository.findOneProjectById(projectId);
        super.bind(object, "code", "moment", "startTimeDuration", "finishTimeDuration", "amount", "type", "contact", "link");
        object.setProject(project);

    }
    public boolean isCurrencyAccepted(final Money moneda) {
        SystemConfiguration moneys;
        moneys = this.repository.findSystemConfiguration();

        String[] listaMonedas = moneys.getAcceptedCurrencies().split(",");
        for (String divisa : listaMonedas)
```



```java
        for (String divisa : listaMonedas)
            if (moneda.getCurrency().equals(divisa))
                return true;

        return false;
    }

    @Override
    public void validate(final Sponsorship object) {
        assert object != null;
        Collection<Invoice> invoices;

        invoices = this.repository.findInvoicesBySponsorshipId(object.getId());
        if (!super.getBuffer().getErrors().hasErrors("code")) {
            Sponsorship existing;

            existing = this.repository.findOneSponsorshipByCode(object.getCode());
            super.state(existing == null || existing.getId() == object.getId(), "code", "sponsor.sponsorship.form.error.duplicated");
        }

        if (!super.getBuffer().getErrors().hasErrors("amount"))
            super.state(object.getAmount().getAmount() > 0, "amount", "sponsor.sponsorship.form.error.amount-no-positive-or-zero");

        if (!super.getBuffer().getErrors().hasErrors("startTimeDuration"))
            super.state(object.getMoment() != null && object.getStartTimeDuration().after(object.getMoment()), "startTimeDuration", "sponsor.spo

        if (!super.getBuffer().getErrors().hasErrors("finishTimeDuration")) {
            super.state(object.getStartTimeDuration() != null && object.getFinishTimeDuration().after(object.getStartTimeDuration()), "finishTim
            if (object.getStartTimeDuration() != null && object.getFinishTimeDuration().after(object.getStartTimeDuration())) {
                Date minimumDeadline;
```

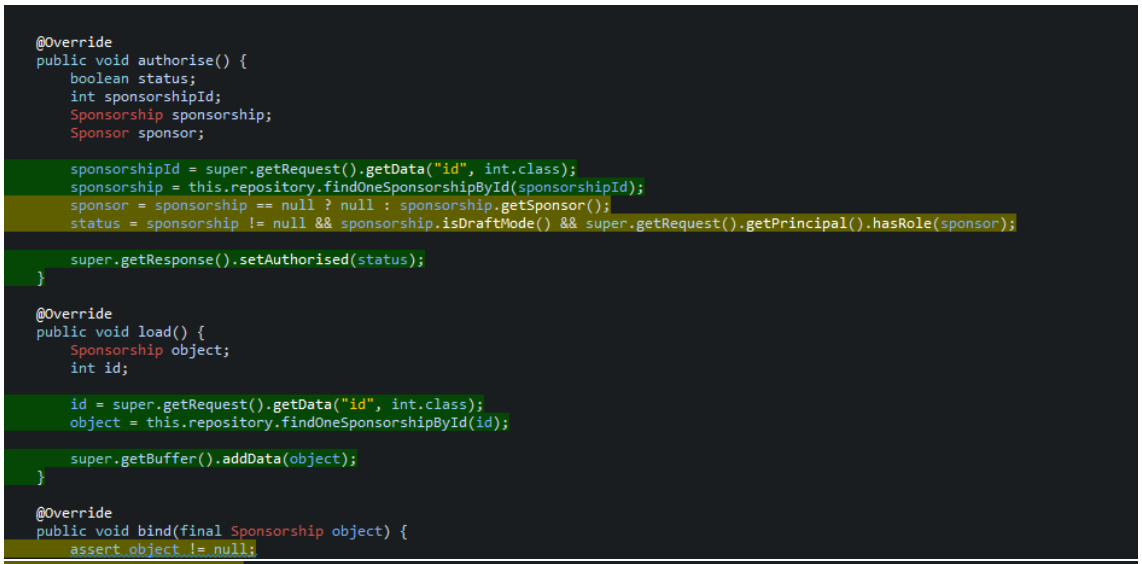


```java
            super.state(object.getAmount().getAmount() > 0, "amount", "sponsor.sponsorship.form.error.amount-no-positive-or-zero");

        if (!super.getBuffer().getErrors().hasErrors("startTimeDuration"))
            super.state(object.getMoment() != null && object.getStartTimeDuration().after(object.getMoment()), "startTimeDuration", "sponsor.spons

        if (!super.getBuffer().getErrors().hasErrors("finishTimeDuration")) {
            super.state(object.getStartTimeDuration() != null && object.getFinishTimeDuration().after(object.getStartTimeDuration()), "finishTimeD
            if (object.getStartTimeDuration() != null && object.getFinishTimeDuration().after(object.getStartTimeDuration())) {
                Date minimumDeadline;

                minimumDeadline = MomentHelper.deltaFromMoment(object.getStartTimeDuration(), 30, ChronoUnit.DAYS);
                super.state(object.getFinishTimeDuration().after(minimumDeadline), "finishTimeDuration", "sponsor.sponsorship.form.error.too-close

            }
        }
        if (!super.getBuffer().getErrors().hasErrors("amount"))
            super.state(this.isCurrencyAccepted(object.getAmount()), "amount", "sponsor.sponsorship.form.error.acceptedCurrency");
        if (object.getAmount() != null) {
            double allAmount = 0.0;
            int cantidadInvoices = 0;
            Money m = new Money();
            m.setAmount(0.0);
            for (Invoice i : invoices) {
                allAmount += i.totalAmount();
                cantidadInvoices += 1;
                m.setCurrency(i.getQuantity().getCurrency());
            }
            if (cantidadInvoices >= 1)
                super.state(object.getAmount().getCurrency().equals(m.getCurrency()), "amount", "sponsor.sponsorship.form.error.changeDivisa");
            super.state(object.getAmount().getAmount() >= allAmount, "amount", "sponsor.sponsorship.form.error.totalAmount-lessThanSumInvoices");
```

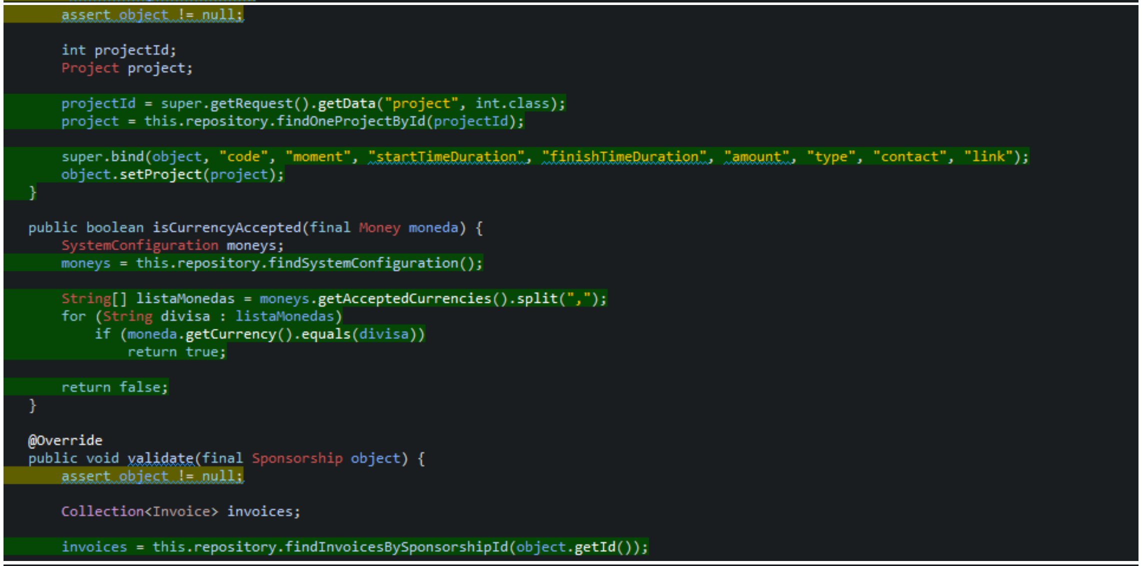


```java
    @Override
    public void perform(final Sponsorship object) {
        assert object != null;

        this.repository.save(object);
    }

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

        Collection<Project> projects;
        SelectChoices choices;
        Dataset dataset;

        projects = this.repository.findAllProjects();
        choices = SelectChoices.from(projects, "title", object.getProject());

        SelectChoices choicesType;
        choicesType = SelectChoices.from(TypeSponsorship.class, object.getType());

        dataset = super.unbind(object, "code", "moment", "startTimeDuration", "finishTimeDuration", "amount", "contact", "link", "draftMode");
        dataset.put("project", choices.getSelected().getKey());
        dataset.put("projects", choices);
        dataset.put("types", choicesType);
        dataset.put("type", choicesType.getSelected().getKey());

        super.getResponse().addData(dataset);

    }
```

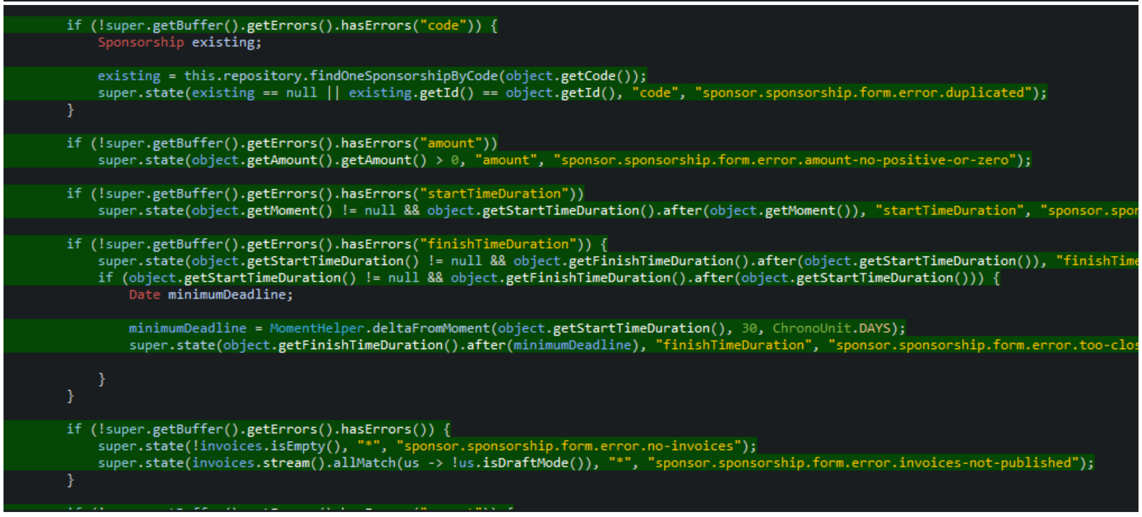
SponsorSponsorshipPublishService.java



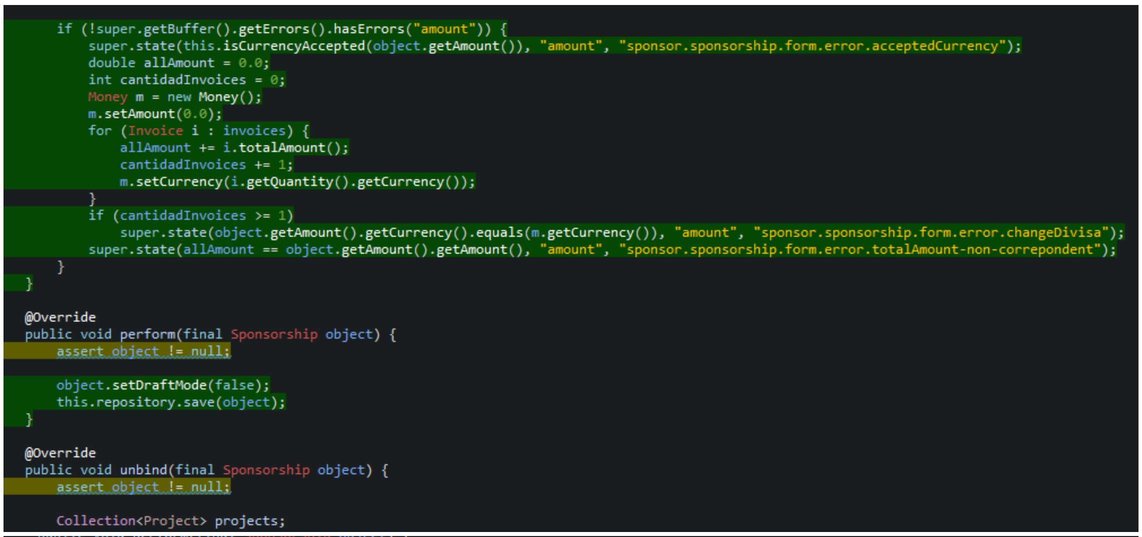
```java
    @Override
    public void authorise() {
        boolean status;
        int sponsorshipId;
        Sponsorship sponsorship;
        Sponsor sponsor;

        sponsorshipId = super.getRequest().getData("id", int.class);
        sponsorship = this.repository.findOneSponsorshipById(sponsorshipId);
        sponsor = sponsorship == null ? null : sponsorship.getSponsor();
        status = sponsorship != null && sponsorship.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsor);

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

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

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

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

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



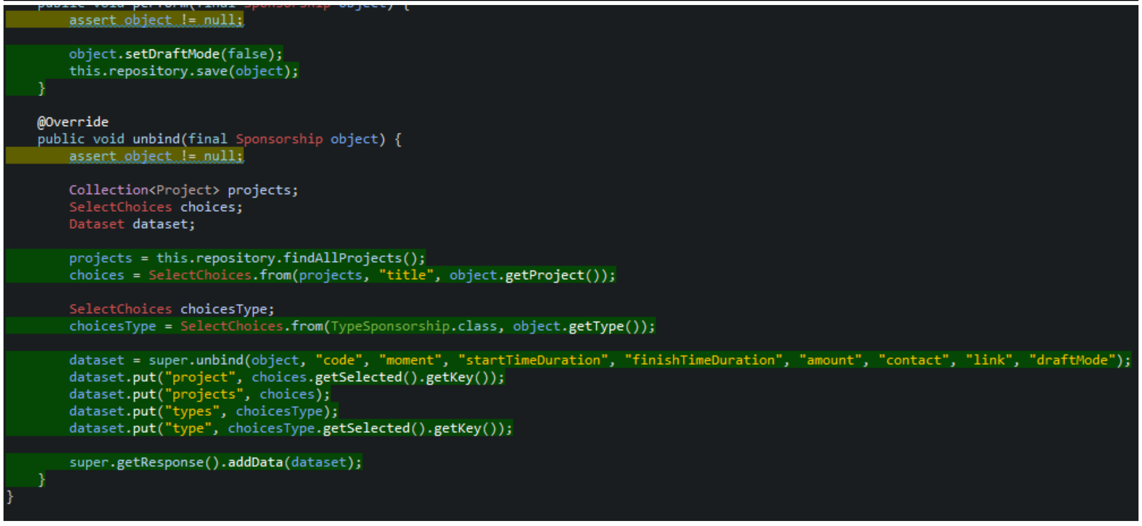
```java
        assert object != null;

        int projectId;
        Project project;

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

        super.bind(object, "code", "moment", "startTimeDuration", "finishTimeDuration", "amount", "type", "contact", "link");
        object.setProject(project);
    }

    public boolean isCurrencyAccepted(final Money moneda) {
        SystemConfiguration moneys;
        moneys = this.repository.findSystemConfiguration();

        String[] listaMonedas = moneys.getAcceptedCurrencies().split(",");
        for (String divisa : listaMonedas)
            if (moneda.getCurrency().equals(divisa))
                return true;

        return false;
    }

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

        Collection<Invoice> invoices;

        invoices = this.repository.findInvoicesBySponsorshipId(object.getId());
```



```java
        if (!super.getBuffer().getErrors().hasErrors("code")) {
            Sponsorship existing;

            existing = this.repository.findOneSponsorshipByCode(object.getCode());
            super.state(existing == null || existing.getId() == object.getId(), "code", "sponsor.sponsorship.form.error.duplicated");
        }

        if (!super.getBuffer().getErrors().hasErrors("amount"))
            super.state(object.getAmount().getAmount() > 0, "amount", "sponsor.sponsorship.form.error.amount-no-positive-or-zero");

        if (!super.getBuffer().getErrors().hasErrors("startTimeDuration"))
            super.state(object.getMoment() != null && object.getStartTimeDuration().after(object.getMoment()), "startTimeDuration", "sponsor.spor

        if (!super.getBuffer().getErrors().hasErrors("finishTimeDuration")) {
            super.state(object.getStartTimeDuration() != null && object.getFinishTimeDuration().after(object.getStartTimeDuration()), "finishTime
            if (object.getStartTimeDuration() != null && object.getFinishTimeDuration().after(object.getStartTimeDuration())) {
                Date minimumDeadline;

                minimumDeadline = MomentHelper.deltaFromMoment(object.getStartTimeDuration(), 30, ChronoUnit.DAYS);
                super.state(object.getFinishTimeDuration().after(minimumDeadline), "finishTimeDuration", "sponsor.sponsorship.form.error.too-clos

            }
        }

        if (!super.getBuffer().getErrors().hasErrors()) {
            super.state(!invoices.isEmpty(), "*", "sponsor.sponsorship.form.error.no-invoices");
            super.state(invoices.stream().allMatch(us -> !us.isDraftMode()), "*", "sponsor.sponsorship.form.error.invoices-not-published");
        }

        if (!super.getBuffer().getErrors().hasErrors("[illegible]")) {
```

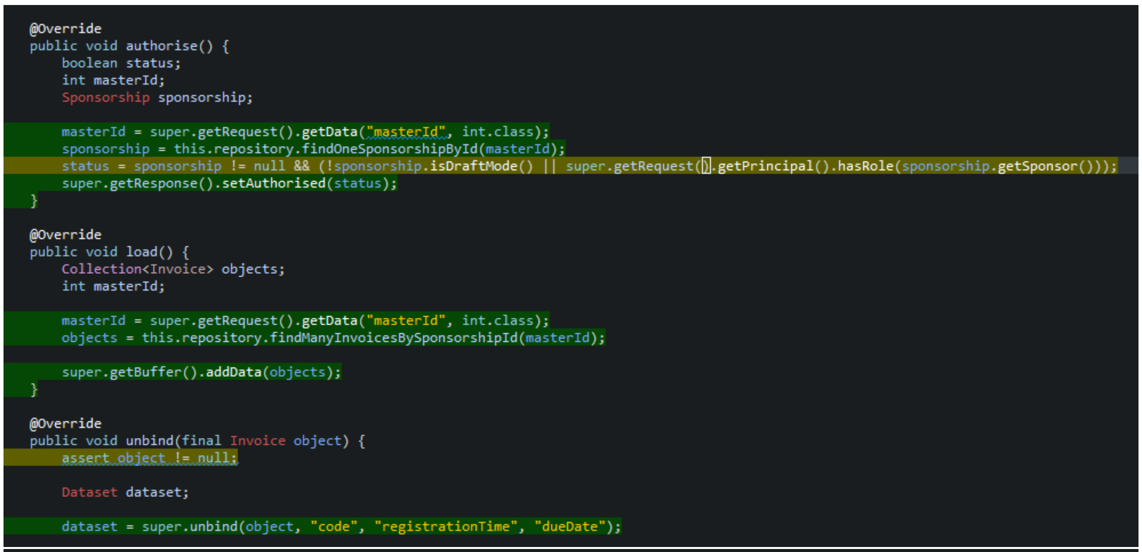


```java
        if (!super.getBuffer().getErrors().hasErrors("amount")) {
            super.state(this.isCurrencyAccepted(object.getAmount()), "amount", "sponsor.sponsorship.form.error.acceptedCurrency");
            double allAmount = 0.0;
            int cantidadInvoices = 0;
            Money m = new Money();
            m.setAmount(0.0);
            for (Invoice i : invoices) {
                allAmount += i.totalAmount();
                cantidadInvoices += 1;
                m.setCurrency(i.getQuantity().getCurrency());
            }
            if (cantidadInvoices >= 1)
                super.state(object.getAmount().getCurrency().equals(m.getCurrency()), "amount", "sponsor.sponsorship.form.error.changeDivisa");
            super.state(allAmount == object.getAmount().getAmount(), "amount", "sponsor.sponsorship.form.error.totalAmount-non-correpondent");
        }
    }

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

        object.setDraftMode(false);
        this.repository.save(object);
    }

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

        Collection<Project> projects;
```

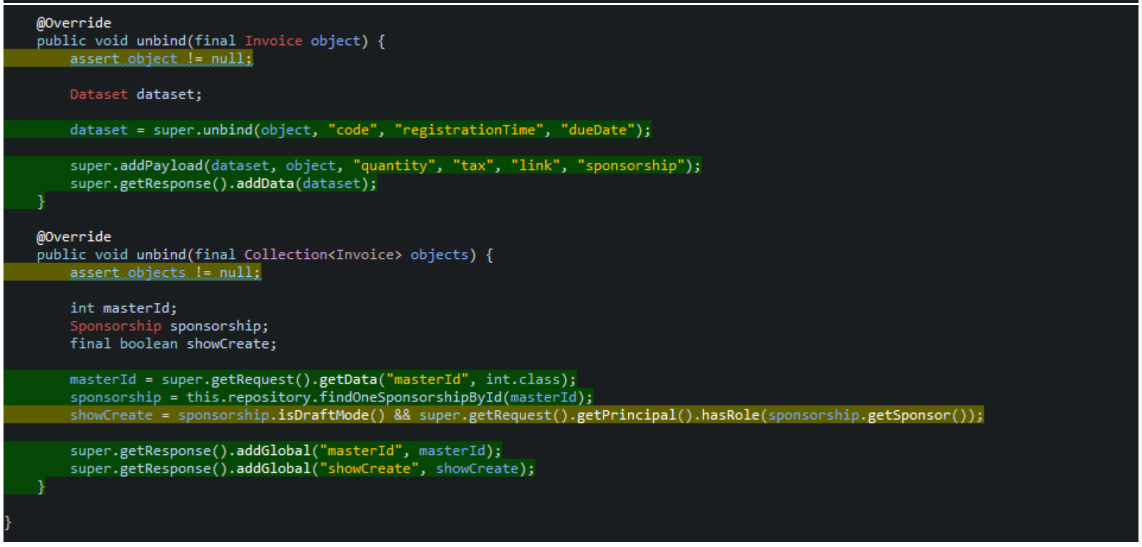


```java
    public void perform(final Sponsorship object) {
        assert object != null;

        object.setDraftMode(false);
        this.repository.save(object);
    }

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

        Collection<Project> projects;
        SelectChoices choices;
        Dataset dataset;

        projects = this.repository.findAllProjects();
        choices = SelectChoices.from(projects, "title", object.getProject());

        SelectChoices choicesType;
        choicesType = SelectChoices.from(TypeSponsorship.class, object.getType());

        dataset = super.unbind(object, "code", "moment", "startTimeDuration", "finishTimeDuration", "amount", "contact", "link", "draftMode");
        dataset.put("project", choices.getSelected().getKey());
        dataset.put("projects", choices);
        dataset.put("types", choicesType);
        dataset.put("type", choicesType.getSelected().getKey());

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

SponsorInvoiceListService.java

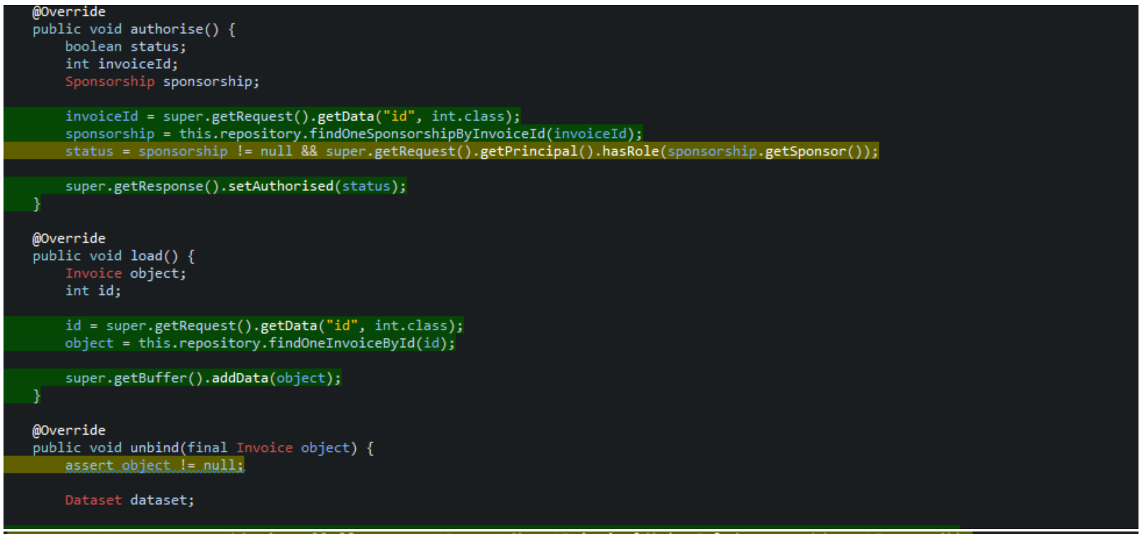


```java
    @Override
    public void authorise() {
        boolean status;
        int masterId;
        Sponsorship sponsorship;

        masterId = super.getRequest().getData("masterId", int.class);
        sponsorship = this.repository.findOneSponsorshipById(masterId);
        status = sponsorship != null && (!sponsorship.isDraftMode() || super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor()));
        super.getResponse().setAuthorised(status);
    }

    @Override
    public void load() {
        Collection<Invoice> objects;
        int masterId;

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

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

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

        Dataset dataset;

        dataset = super.unbind(object, "code", "registrationTime", "dueDate");
```

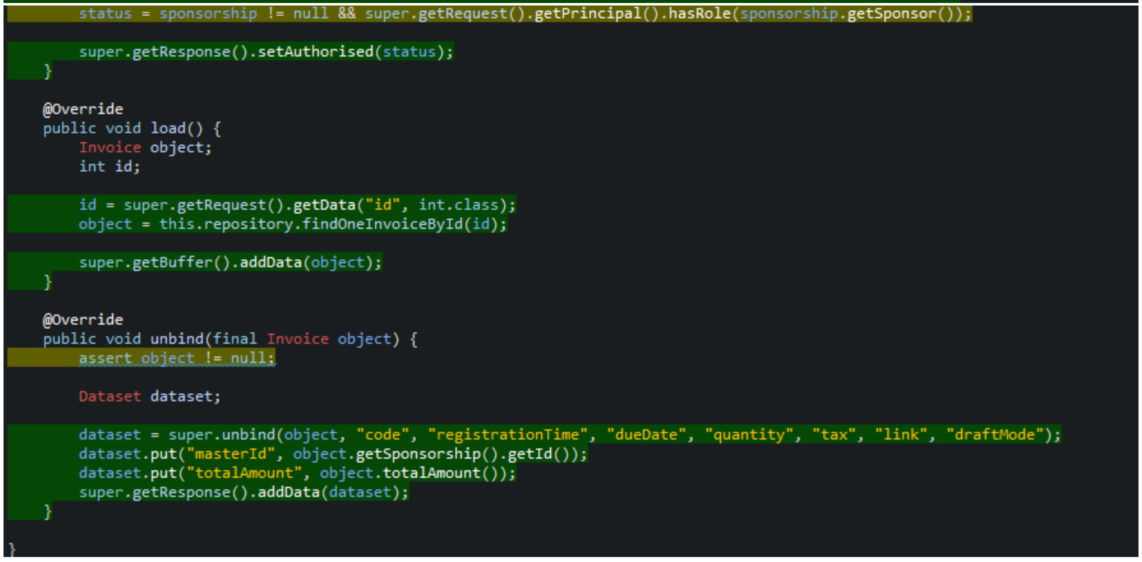


```java
    @Override
    public void unbind(final Invoice object) {
        assert object != null;

        Dataset dataset;

        dataset = super.unbind(object, "code", "registrationTime", "dueDate");

        super.addPayload(dataset, object, "quantity", "tax", "link", "sponsorship");
        super.getResponse().addData(dataset);
    }

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

        int masterId;
        Sponsorship sponsorship;
        final boolean showCreate;

        masterId = super.getRequest().getData("masterId", int.class);
        sponsorship = this.repository.findOneSponsorshipById(masterId);
        showCreate = sponsorship.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());

        super.getResponse().addGlobal("masterId", masterId);
        super.getResponse().addGlobal("showCreate", showCreate);
    }

}
```

SponsorInvoiceShowService.java



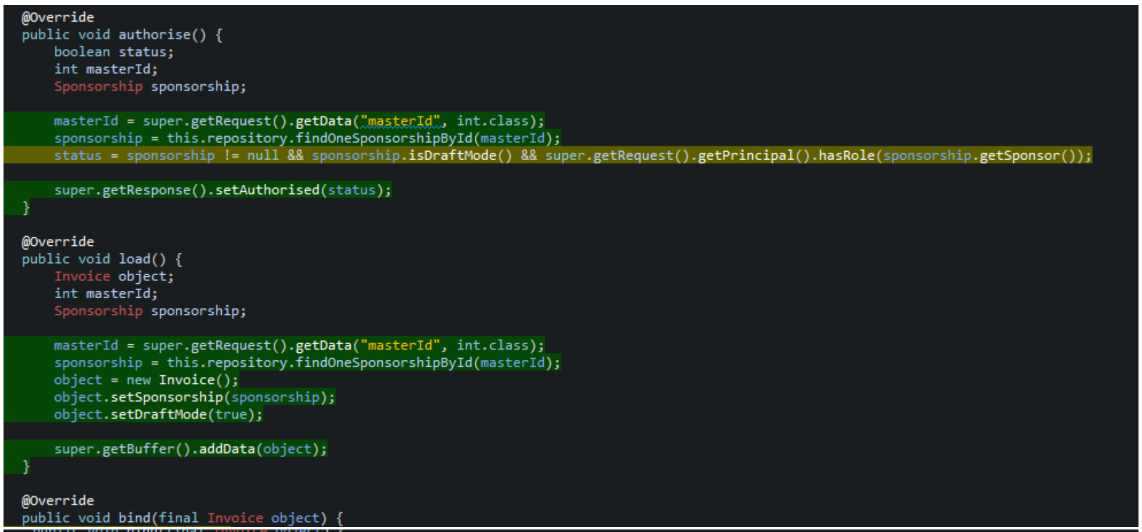
```
    @Override
    public void authorise() {
        boolean status;
        int invoiceId;
        Sponsorship sponsorship;

        invoiceId = super.getRequest().getData("id", int.class);
        sponsorship = this.repository.findOneSponsorshipByInvoiceId(invoiceId);
        status = sponsorship != null && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());

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

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

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

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

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

        Dataset dataset;
```



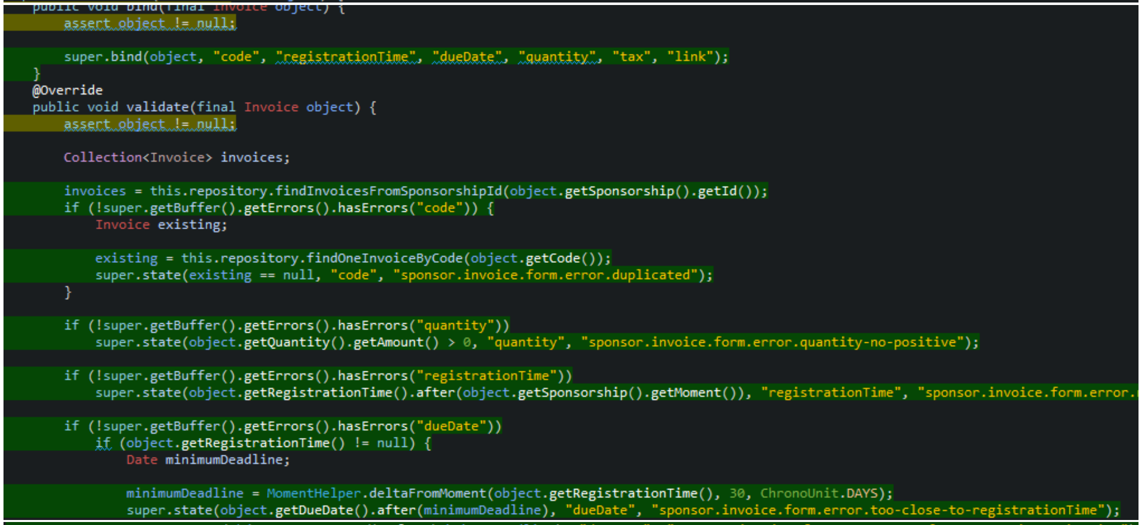
```
        status = sponsorship != null && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());

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

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

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

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

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

        Dataset dataset;
```

```
        dataset = super.unbind(object, "code", "registrationTime", "dueDate", "quantity", "tax", "link", "draftMode");
        dataset.put("masterId", object.getSponsorship().getId());
        dataset.put("totalAmount", object.totalAmount());
        super.getResponse().addData(dataset);
    }

}
```

SponsorInvoiceCreateService.java

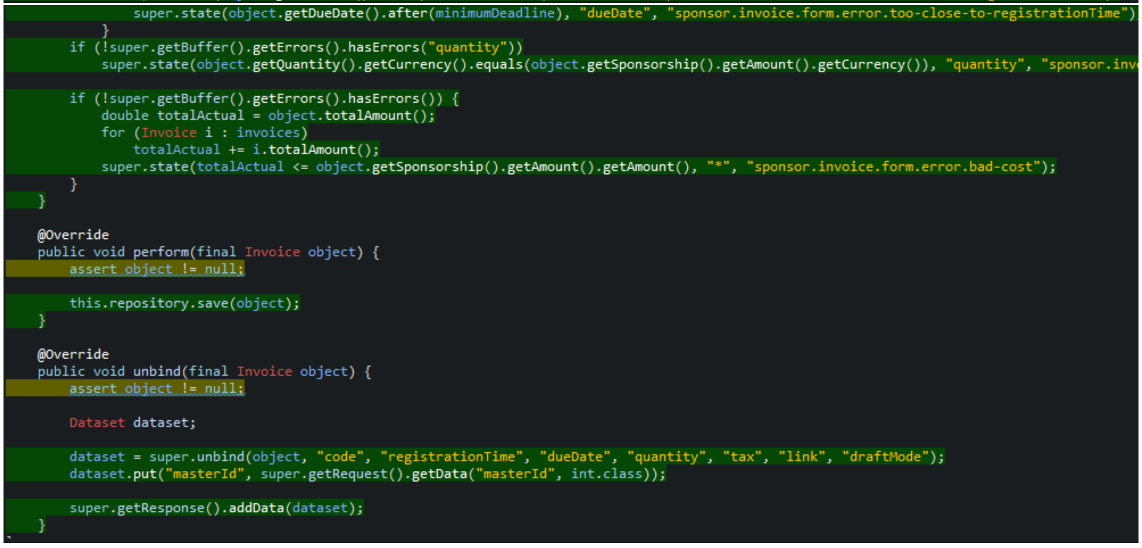


```java
@Override
public void authorise() {
    boolean status;
    int masterId;
    Sponsorship sponsorship;

    masterId = super.getRequest().getData("masterId", int.class);
    sponsorship = this.repository.findOneSponsorshipById(masterId);
    status = sponsorship != null && sponsorship.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());

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

@Override
public void load() {
    Invoice object;
    int masterId;
    Sponsorship sponsorship;

    masterId = super.getRequest().getData("masterId", int.class);
    sponsorship = this.repository.findOneSponsorshipById(masterId);
    object = new Invoice();
    object.setSponsorship(sponsorship);
    object.setDraftMode(true);

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

@Override
public void bind(final Invoice object) {
```



```java
public void bind(final Invoice object) {
```

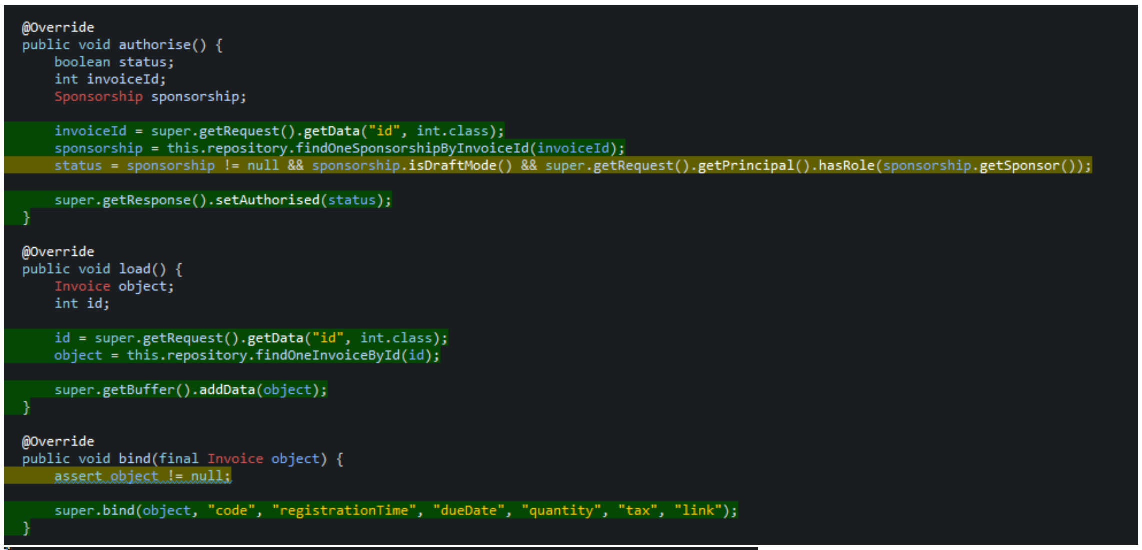
```java
    assert object != null;

    super.bind(object, "code", "registrationTime", "dueDate", "quantity", "tax", "link");
}
@Override
public void validate(final Invoice object) {
    assert object != null;

    Collection<Invoice> invoices;

    invoices = this.repository.findInvoicesFromSponsorshipId(object.getSponsorship().getId());
    if (!super.getBuffer().getErrors().hasErrors("code")) {
        Invoice existing;

        existing = this.repository.findOneInvoiceByCode(object.getCode());
        super.state(existing == null, "code", "sponsor.invoice.form.error.duplicated");
    }

    if (!super.getBuffer().getErrors().hasErrors("quantity"))
        super.state(object.getQuantity().getAmount() > 0, "quantity", "sponsor.invoice.form.error.quantity-no-positive");

    if (!super.getBuffer().getErrors().hasErrors("registrationTime"))
        super.state(object.getRegistrationTime().after(object.getSponsorship().getMoment()), "registrationTime", "sponsor.invoice.form.error.

    if (!super.getBuffer().getErrors().hasErrors("dueDate"))
        if (object.getRegistrationTime() != null) {
            Date minimumDeadline;

            minimumDeadline = MomentHelper.deltaFromMoment(object.getRegistrationTime(), 30, ChronoUnit.DAYS);
            super.state(object.getDueDate().after(minimumDeadline), "dueDate", "sponsor.invoice.form.error.too-close-to-registrationTime");
```



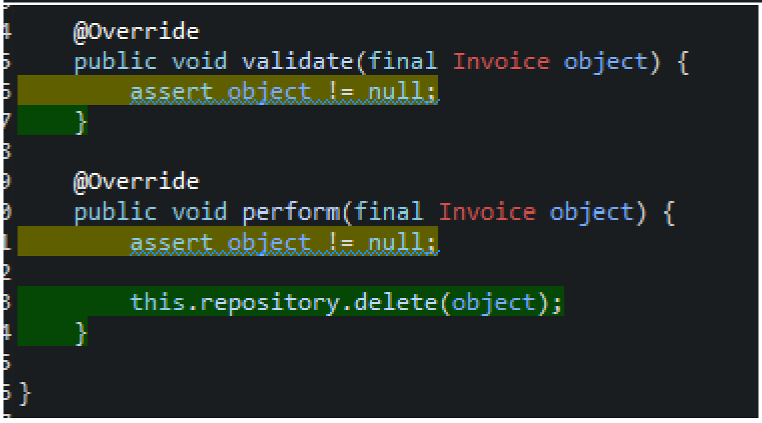
```java
            super.state(object.getDueDate().after(minimumDeadline), "dueDate", "sponsor.invoice.form.error.too-close-to-registrationTime")
```

```java
        }
    if (!super.getBuffer().getErrors().hasErrors("quantity"))
        super.state(object.getQuantity().getCurrency().equals(object.getSponsorship().getAmount().getCurrency()), "quantity", "sponsor.inv

    if (!super.getBuffer().getErrors().hasErrors()) {
        double totalActual = object.totalAmount();
        for (Invoice i : invoices)
            totalActual += i.totalAmount();
        super.state(totalActual <= object.getSponsorship().getAmount().getAmount(), "*", "sponsor.invoice.form.error.bad-cost");
    }
}

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

    this.repository.save(object);
}

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

    Dataset dataset;

    dataset = super.unbind(object, "code", "registrationTime", "dueDate", "quantity", "tax", "link", "draftMode");
    dataset.put("masterId", super.getRequest().getData("masterId", int.class));

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

SponsorInvoiceDeleteService.java

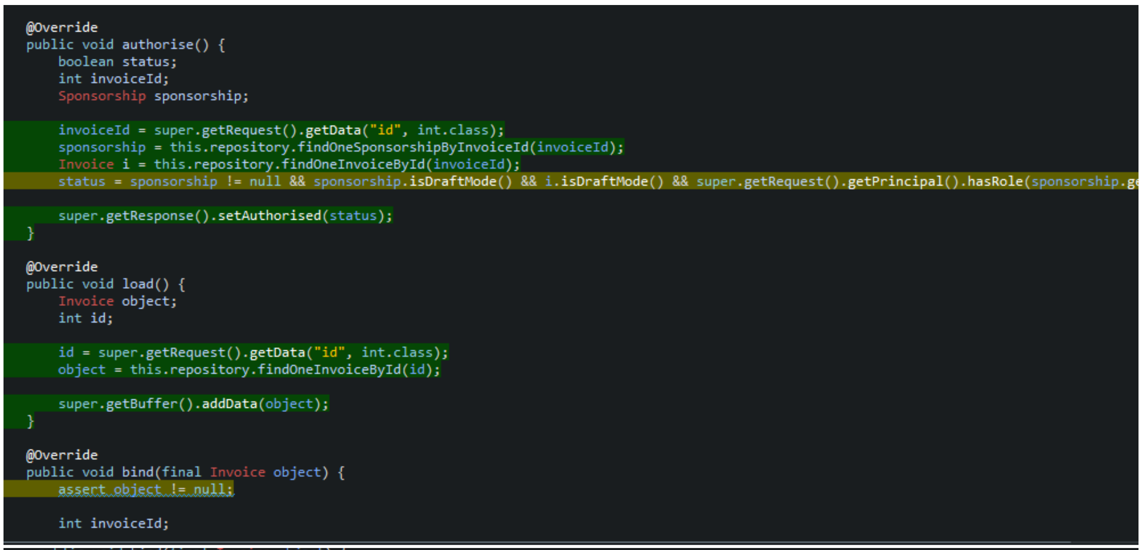


```java
@Override
public void authorise() {
    boolean status;
    int invoiceId;
    Sponsorship sponsorship;

    invoiceId = super.getRequest().getData("id", int.class);
    sponsorship = this.repository.findOneSponsorshipByInvoiceId(invoiceId);
    status = sponsorship != null && sponsorship.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsorship.getSponsor());

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

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

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

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

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

    super.bind(object, "code", "registrationTime", "dueDate", "quantity", "tax", "link");
}
```

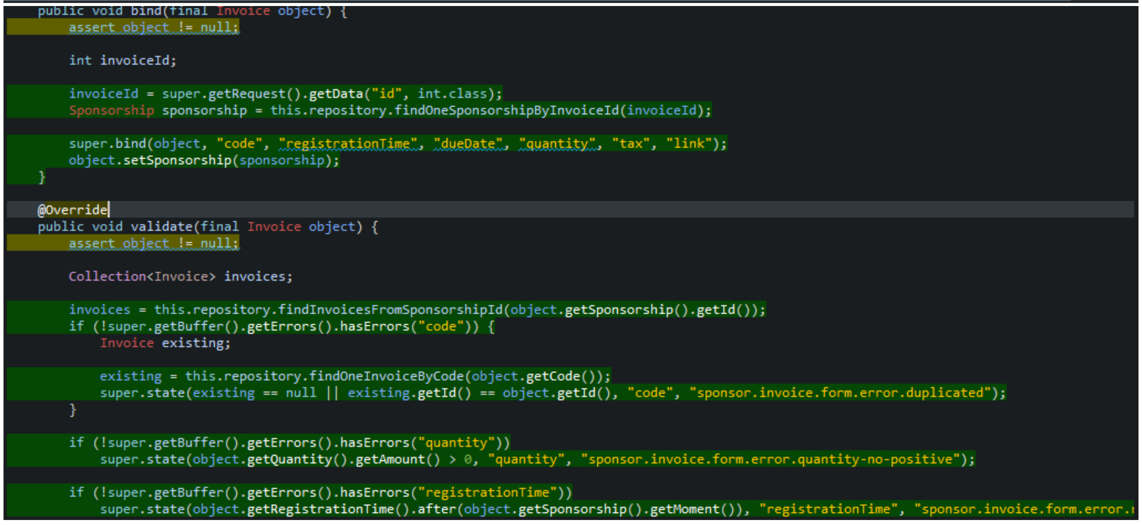


```java
    @Override
    public void validate(final Invoice object) {
        assert object != null;
    }

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

        this.repository.delete(object);
    }

}
```

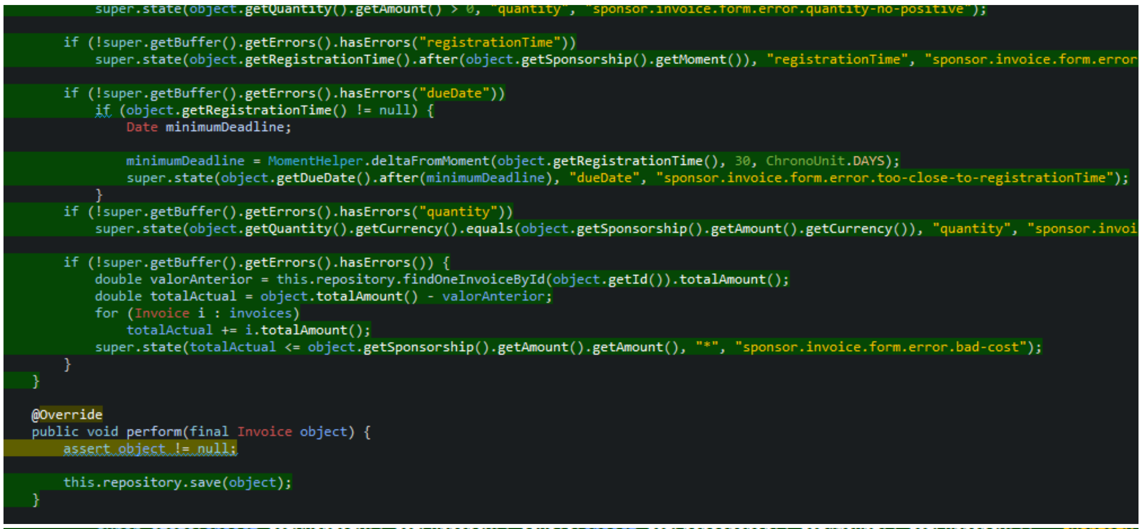
SponsorInvoiceUpdateService.java



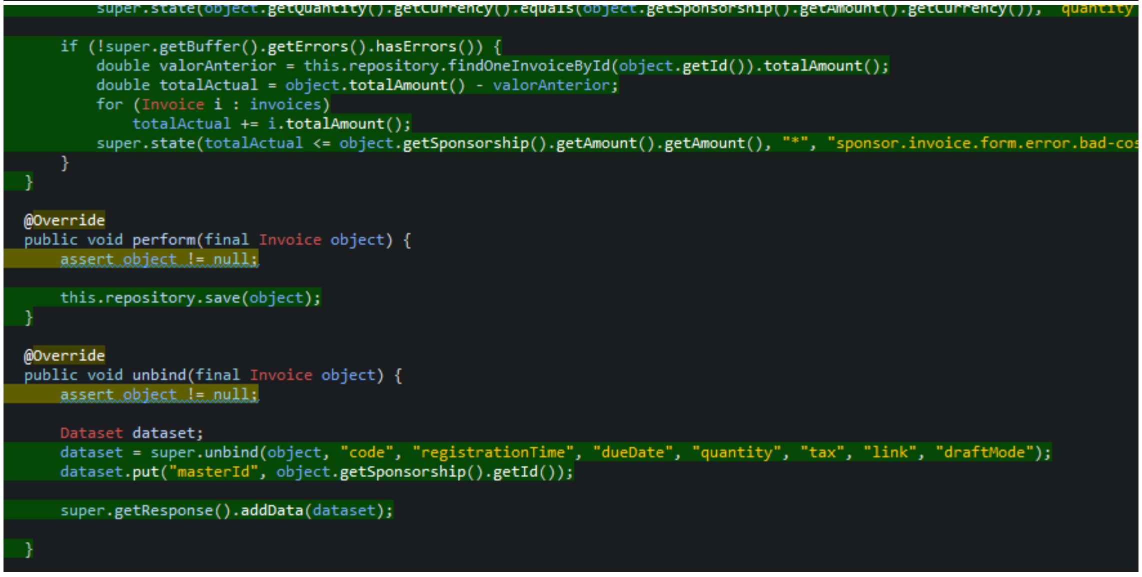
```java
@Override
public void authorise() {
    boolean status;
    int invoiceId;
    Sponsorship sponsorship;

    invoiceId = super.getRequest().getData("id", int.class);
    sponsorship = this.repository.findOneSponsorshipByInvoiceId(invoiceId);
    Invoice i = this.repository.findOneInvoiceById(invoiceId);
    status = sponsorship != null && sponsorship.isDraftMode() && i.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsorship.ge

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

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

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

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

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

    int invoiceId;
```



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

    int invoiceId;

    invoiceId = super.getRequest().getData("id", int.class);
    Sponsorship sponsorship = this.repository.findOneSponsorshipByInvoiceId(invoiceId);

    super.bind(object, "code", "registrationTime", "dueDate", "quantity", "tax", "link");
    object.setSponsorship(sponsorship);
}

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

    Collection<Invoice> invoices;

    invoices = this.repository.findInvoicesFromSponsorshipId(object.getSponsorship().getId());
    if (!super.getBuffer().getErrors().hasErrors("code")) {
        Invoice existing;

        existing = this.repository.findOneInvoiceByCode(object.getCode());
        super.state(existing == null || existing.getId() == object.getId(), "code", "sponsor.invoice.form.error.duplicated");
    }

    if (!super.getBuffer().getErrors().hasErrors("quantity"))
        super.state(object.getQuantity().getAmount() > 0, "quantity", "sponsor.invoice.form.error.quantity-no-positive");

    if (!super.getBuffer().getErrors().hasErrors("registrationTime"))
        super.state(object.getRegistrationTime().after(object.getSponsorship().getMoment()), "registrationTime", "sponsor.invoice.form.error.
```

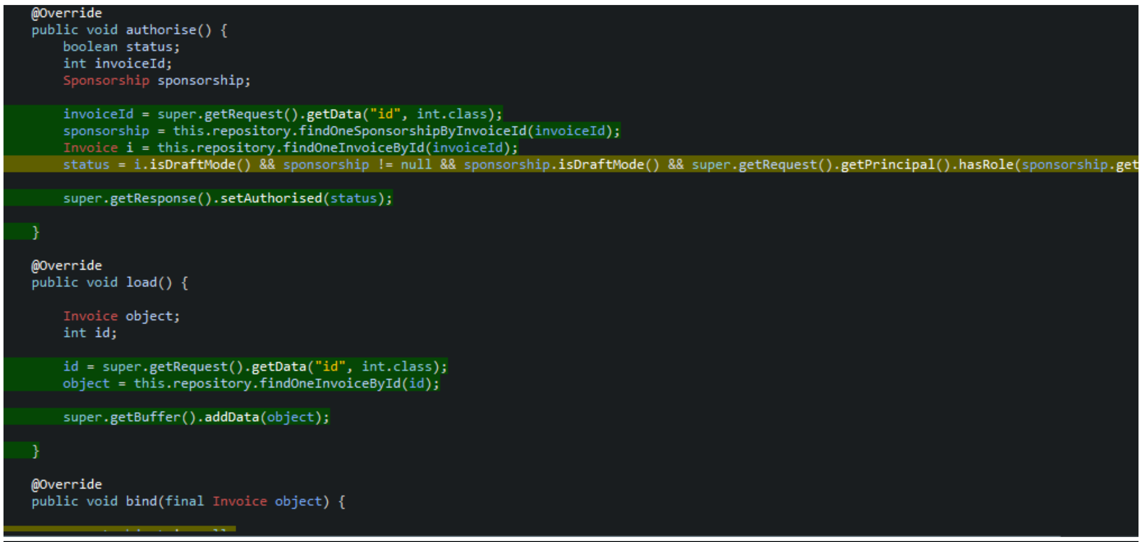


```
            super.state(object.getQuantity().getAmount() > 0, "quantity", "sponsor.invoice.form.error.quantity-no-positive");

        if (!super.getBuffer().getErrors().hasErrors("registrationTime"))
            super.state(object.getRegistrationTime().after(object.getSponsorship().getMoment()), "registrationTime", "sponsor.invoice.form.error

        if (!super.getBuffer().getErrors().hasErrors("dueDate"))
            if (object.getRegistrationTime() != null) {
                Date minimumDeadline;

                minimumDeadline = MomentHelper.deltaFromMoment(object.getRegistrationTime(), 30, ChronoUnit.DAYS);
                super.state(object.getDueDate().after(minimumDeadline), "dueDate", "sponsor.invoice.form.error.too-close-to-registrationTime");
            }
        if (!super.getBuffer().getErrors().hasErrors("quantity"))
            super.state(object.getQuantity().getCurrency().equals(object.getSponsorship().getAmount().getCurrency()), "quantity", "sponsor.invoi

        if (!super.getBuffer().getErrors().hasErrors()) {
            double valorAnterior = this.repository.findOneInvoiceById(object.getId()).totalAmount();
            double totalActual = object.totalAmount() - valorAnterior;
            for (Invoice i : invoices)
                totalActual += i.totalAmount();
            super.state(totalActual <= object.getSponsorship().getAmount().getAmount(), "*", "sponsor.invoice.form.error.bad-cost");
        }
    }

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

        this.repository.save(object);
    }
```

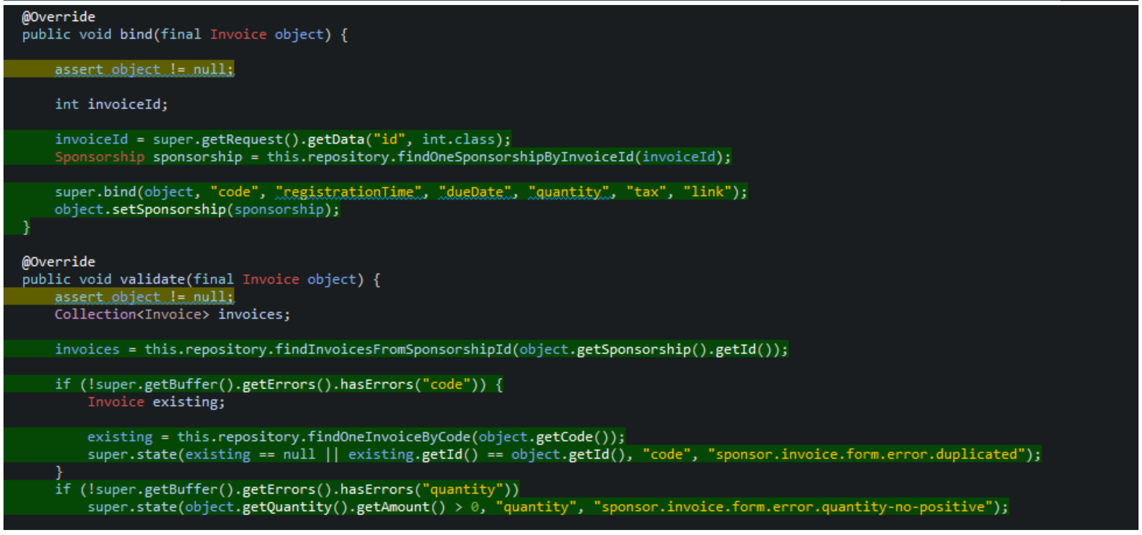


```
            super.state(object.getQuantity().getCurrency().equals(object.getSponsorship().getAmount().getCurrency()), "quantity"

        if (!super.getBuffer().getErrors().hasErrors()) {
            double valorAnterior = this.repository.findOneInvoiceById(object.getId()).totalAmount();
            double totalActual = object.totalAmount() - valorAnterior;
            for (Invoice i : invoices)
                totalActual += i.totalAmount();
            super.state(totalActual <= object.getSponsorship().getAmount().getAmount(), "*", "sponsor.invoice.form.error.bad-cos
        }
    }

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

        this.repository.save(object);
    }

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

        Dataset dataset;
        dataset = super.unbind(object, "code", "registrationTime", "dueDate", "quantity", "tax", "link", "draftMode");
        dataset.put("masterId", object.getSponsorship().getId());

        super.getResponse().addData(dataset);

    }
```

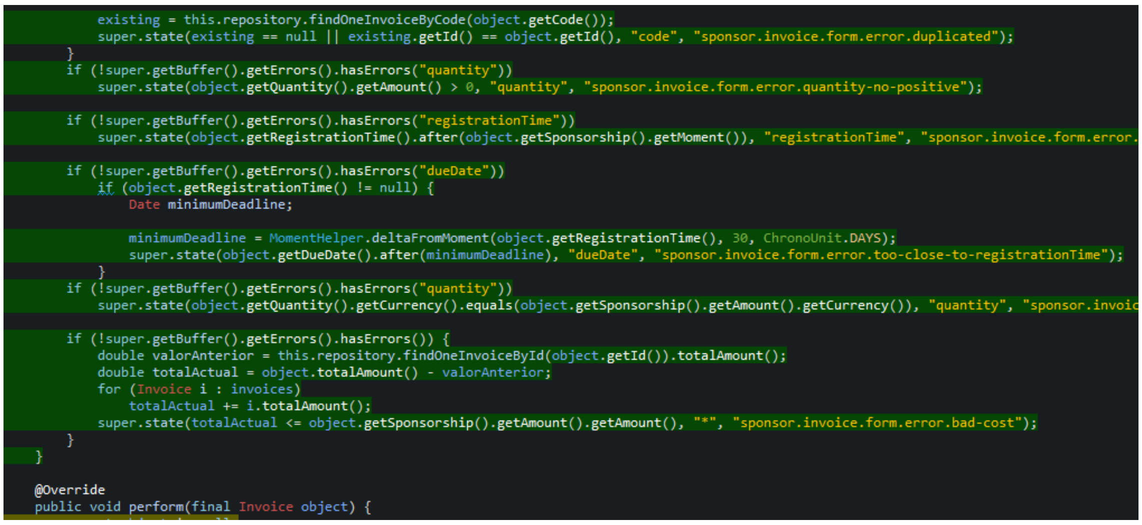
SponsorInvoicePublishService.java



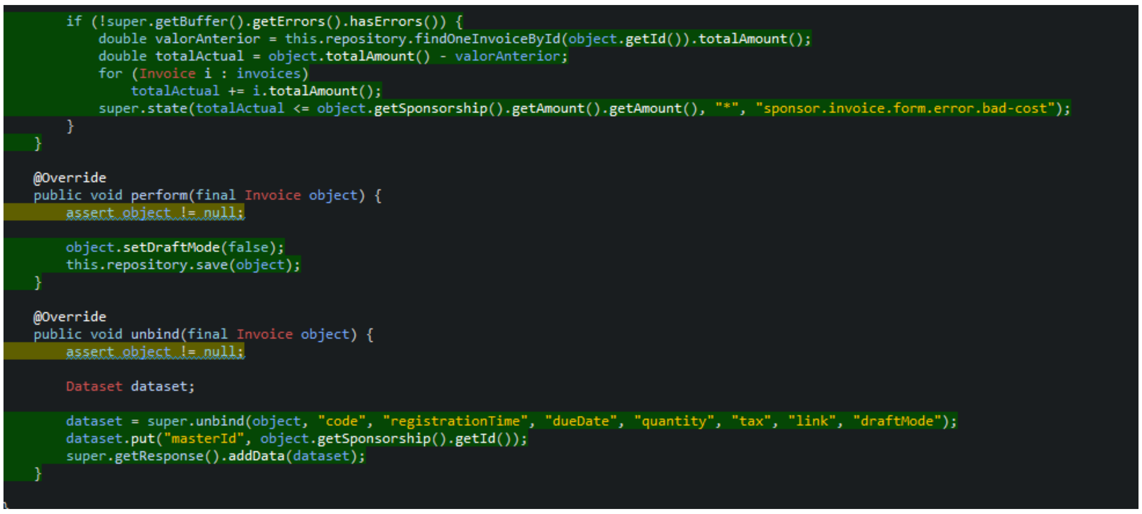
```
    @Override
    public void authorise() {
        boolean status;
        int invoiceId;
        Sponsorship sponsorship;

        invoiceId = super.getRequest().getData("id", int.class);
        sponsorship = this.repository.findOneSponsorshipByInvoiceId(invoiceId);
        Invoice i = this.repository.findOneInvoiceById(invoiceId);
        status = i.isDraftMode() && sponsorship != null && sponsorship.isDraftMode() && super.getRequest().getPrincipal().hasRole(sponsorship.get

        super.getResponse().setAuthorised(status);

    }

    @Override
    public void load() {

        Invoice object;
        int id;

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

        super.getBuffer().addData(object);

    }

    @Override
    public void bind(final Invoice object) {
```



```
  @Override
  public void bind(final Invoice object) {

      assert object != null;

      int invoiceId;

      invoiceId = super.getRequest().getData("id", int.class);
      Sponsorship sponsorship = this.repository.findOneSponsorshipByInvoiceId(invoiceId);

      super.bind(object, "code", "registrationTime", "dueDate", "quantity", "tax", "link");
      object.setSponsorship(sponsorship);
  }

  @Override
  public void validate(final Invoice object) {
      assert object != null;
      Collection<Invoice> invoices;

      invoices = this.repository.findInvoicesFromSponsorshipId(object.getSponsorship().getId());

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

          existing = this.repository.findOneInvoiceByCode(object.getCode());
          super.state(existing == null || existing.getId() == object.getId(), "code", "sponsor.invoice.form.error.duplicated");
      }
      if (!super.getBuffer().getErrors().hasErrors("quantity"))
          super.state(object.getQuantity().getAmount() > 0, "quantity", "sponsor.invoice.form.error.quantity-no-positive");
```



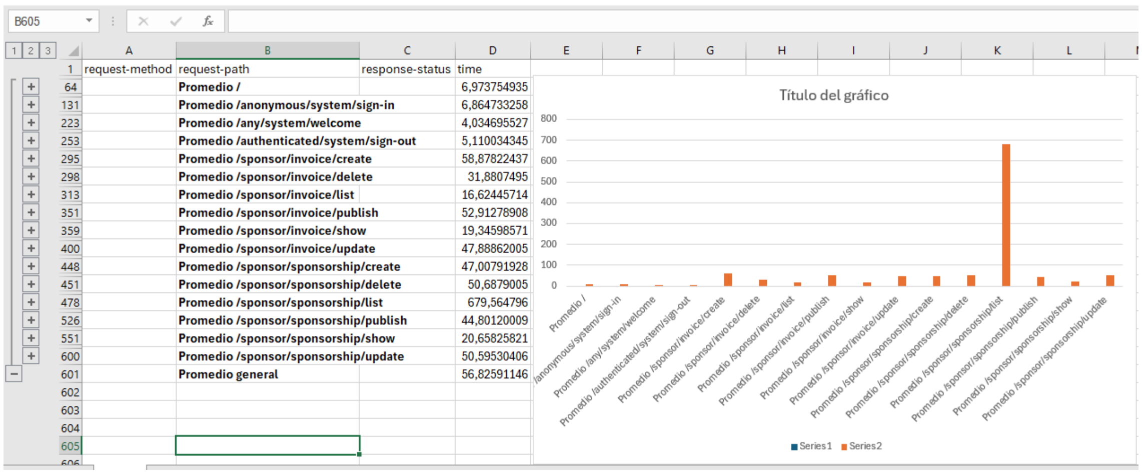
```java
            existing = this.repository.findOneInvoiceByCode(object.getCode());
            super.state(existing == null || existing.getId() == object.getId(), "code", "sponsor.invoice.form.error.duplicated");
        }
        if (!super.getBuffer().getErrors().hasErrors("quantity"))
            super.state(object.getQuantity().getAmount() > 0, "quantity", "sponsor.invoice.form.error.quantity-no-positive");

        if (!super.getBuffer().getErrors().hasErrors("registrationTime"))
            super.state(object.getRegistrationTime().after(object.getSponsorship().getMoment()), "registrationTime", "sponsor.invoice.form.error.

        if (!super.getBuffer().getErrors().hasErrors("dueDate"))
            if (object.getRegistrationTime() != null) {
                Date minimumDeadline;

                minimumDeadline = MomentHelper.deltaFromMoment(object.getRegistrationTime(), 30, ChronoUnit.DAYS);
                super.state(object.getDueDate().after(minimumDeadline), "dueDate", "sponsor.invoice.form.error.too-close-to-registrationTime");
            }
        if (!super.getBuffer().getErrors().hasErrors("quantity"))
            super.state(object.getQuantity().getCurrency().equals(object.getSponsorship().getAmount().getCurrency()), "quantity", "sponsor.invoic

        if (!super.getBuffer().getErrors().hasErrors()) {
            double valorAnterior = this.repository.findOneInvoiceById(object.getId()).totalAmount();
            double totalActual = object.totalAmount() - valorAnterior;
            for (Invoice i : invoices)
                totalActual += i.totalAmount();
            super.state(totalActual <= object.getSponsorship().getAmount().getAmount(), "*", "sponsor.invoice.form.error.bad-cost");
        }
    }

    @Override
    public void perform(final Invoice object) {
```



```java
        if (!super.getBuffer().getErrors().hasErrors()) {
            double valorAnterior = this.repository.findOneInvoiceById(object.getId()).totalAmount();
            double totalActual = object.totalAmount() - valorAnterior;
            for (Invoice i : invoices)
                totalActual += i.totalAmount();
            super.state(totalActual <= object.getSponsorship().getAmount().getAmount(), "*", "sponsor.invoice.form.error.bad-cost");
        }
    }

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

        object.setDraftMode(false);
        this.repository.save(object);
    }

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

        Dataset dataset;

        dataset = super.unbind(object, "code", "registrationTime", "dueDate", "quantity", "tax", "link", "draftMode");
        dataset.put("masterId", object.getSponsorship().getId());
        super.getResponse().addData(dataset);
    }
```

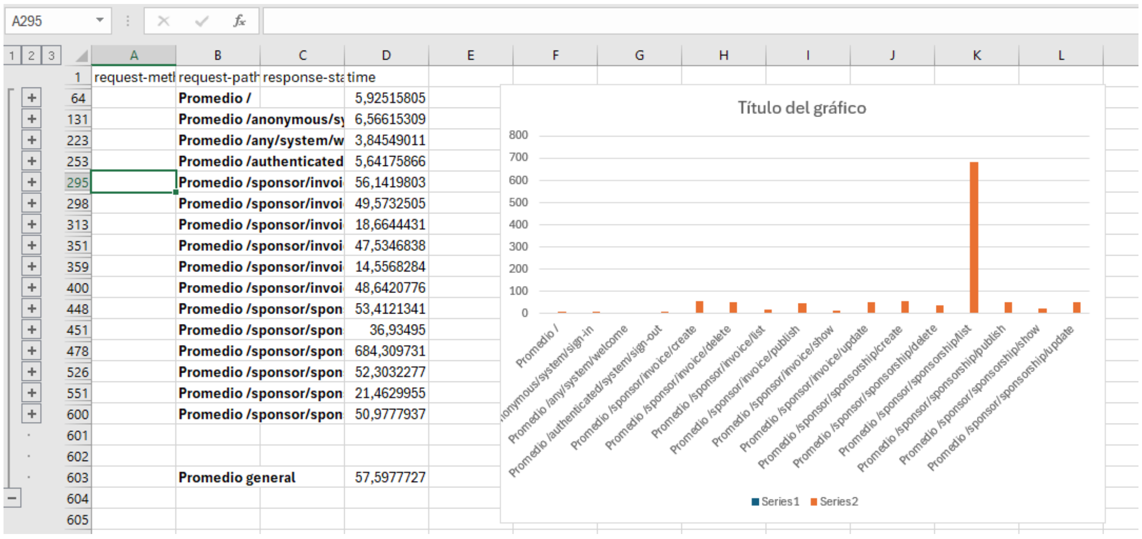
RENDIMIENTO

Sin los índices:


B605
Título del gráfico
800
700
600
500
400
300
200
100
0
Promedio /
Promedio /anonymous/system/sign-in
Promedio /any/system/welcome
Promedio /authenticated/system/sign-out
Promedio /sponsor/invoice/create
Promedio /sponsor/invoice/delete
Promedio /sponsor/invoice/list
Promedio /sponsor/invoice/publish
Promedio /sponsor/invoice/show
Promedio /sponsor/invoice/update
Promedio /sponsor/sponsorship/create
Promedio /sponsor/sponsorship/delete
Promedio /sponsor/sponsorship/list
Promedio /sponsor/sponsorship/publish
Promedio /sponsor/sponsorship/show
Promedio /sponsor/sponsorship/update
Series1
Series2

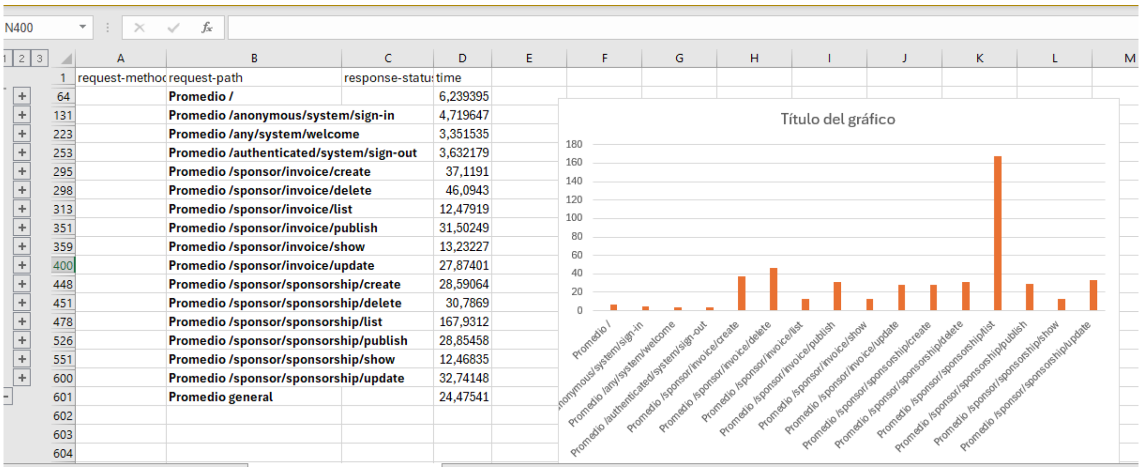
	A	B	C	D
1	request-method	request-path	response-status	time
64		**Promedio /**		6,973754935
131		**Promedio /anonymous/system/sign-in**		6,864733258
223		**Promedio /any/system/welcome**		4,034695527
253		**Promedio /authenticated/system/sign-out**		5,110034345
295		**Promedio /sponsor/invoice/create**		58,87822437
298		**Promedio /sponsor/invoice/delete**		31,8807495
313		**Promedio /sponsor/invoice/list**		16,62445714
351		**Promedio /sponsor/invoice/publish**		52,91278908
359		**Promedio /sponsor/invoice/show**		19,34598571
400		**Promedio /sponsor/invoice/update**		47,88862005
448		**Promedio /sponsor/sponsorship/create**		47,00791928
451		**Promedio /sponsor/sponsorship/delete**		50,6879005
478		**Promedio /sponsor/sponsorship/list**		679,564796
526		**Promedio /sponsor/sponsorship/publish**		44,80120009
551		**Promedio /sponsor/sponsorship/show**		20,65825821
600		**Promedio /sponsor/sponsorship/update**		50,59530406
601		**Promedio general**		56,82591146
602				
603				
604				
605				

Con los índices:


A295
Título del gráfico
800
700
600
500
400
300
200
100
0
Promedio /
Promedio /anonymous/system/sign-in
Promedio /any/system/welcome
Promedio /authenticated/system/sign-out
Promedio /sponsor/invoice/create
Promedio /sponsor/invoice/delete
Promedio /sponsor/invoice/list
Promedio /sponsor/invoice/publish
Promedio /sponsor/invoice/show
Promedio /sponsor/invoice/update
Promedio /sponsor/sponsorship/create
Promedio /sponsor/sponsorship/delete
Promedio /sponsor/sponsorship/list
Promedio /sponsor/sponsorship/publish
Promedio /sponsor/sponsorship/show
Promedio /sponsor/sponsorship/update
Series1
Series2

	A	B	C	D
1	request-metl	request-path	response-sta	time
64		**Promedio /**		5,92515805
131		**Promedio /anonymous/sy**		6,56615309
223		**Promedio /any/system/w**		3,84549011
253		**Promedio /authenticated**		5,64175866
295		**Promedio /sponsor/invoi**		56,1419803
298		**Promedio /sponsor/invoi**		49,5732505
313		**Promedio /sponsor/invoi**		18,6644431
351		**Promedio /sponsor/invoi**		47,5346838
359		**Promedio /sponsor/invoi**		14,5568284
400		**Promedio /sponsor/invoi**		48,6420776
448		**Promedio /sponsor/spon**		53,4121341
451		**Promedio /sponsor/spon**		36,93495
478		**Promedio /sponsor/spon**		684,309731
526		**Promedio /sponsor/spon**		52,3032277
551		**Promedio /sponsor/spon**		21,4629955
600		**Promedio /sponsor/spon**		50,9777937
601				
602				
603		**Promedio general**		57,5977727
604				
605				

Desde el ordenador de un compañero:


N400
Título del gráfico
180
160
140
120
100
80
60
40
20
0
Promedio /
Promedio /anonymous/system/sign-in
Promedio /any/system/welcome
Promedio /authenticated/system/sign-out
Promedio /sponsor/invoice/create
Promedio /sponsor/invoice/delete
Promedio /sponsor/invoice/list
Promedio /sponsor/invoice/publish
Promedio /sponsor/invoice/show
Promedio /sponsor/invoice/update
Promedio /sponsor/sponsorship/create
Promedio /sponsor/sponsorship/delete
Promedio /sponsor/sponsorship/list
Promedio /sponsor/sponsorship/publish
Promedio /sponsor/sponsorship/show
Promedio /sponsor/sponsorship/update

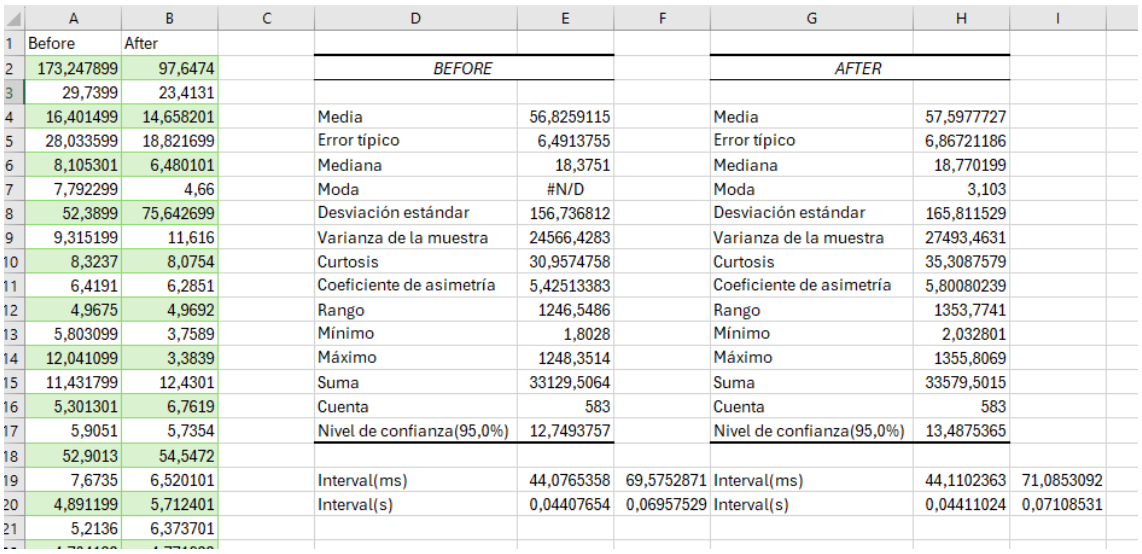
	A	B	C	D
1	request-methoc	request-path	response-statu:	time
64		**Promedio /**		6,239395
131		**Promedio /anonymous/system/sign-in**		4,719647
223		**Promedio /any/system/welcome**		3,351535
253		**Promedio /authenticated/system/sign-out**		3,632179
295		**Promedio /sponsor/invoice/create**		37,1191
298		**Promedio /sponsor/invoice/delete**		46,0943
313		**Promedio /sponsor/invoice/list**		12,47919
351		**Promedio /sponsor/invoice/publish**		31,50249
359		**Promedio /sponsor/invoice/show**		13,23227
400		**Promedio /sponsor/invoice/update**		27,87401
448		**Promedio /sponsor/sponsorship/create**		28,59064
451		**Promedio /sponsor/sponsorship/delete**		30,7869
478		**Promedio /sponsor/sponsorship/list**		167,9312
526		**Promedio /sponsor/sponsorship/publish**		28,85458
551		**Promedio /sponsor/sponsorship/show**		12,46835
600		**Promedio /sponsor/sponsorship/update**		32,74148
601		**Promedio general**		24,47541
602				
603				
604				

Podemos observar que la petición más es la de listar patrocinios, esto puede ser consecuencia a la gran cantidad de patrocinios que lista y a que es la petición más solicitada del sistema.

El uso de índices no ha hecho un gran cambio en sistema, aunque ha agilizado varias peticiones, pero ralentizado otras, en este caso los índices no son de gran utilidad.

Por último, respecto al resultado en otro ordenador, este ha sido exitoso mejorando incluso las prestaciones, esto se debe a que el nuevo ordenador utilizado es más potente.

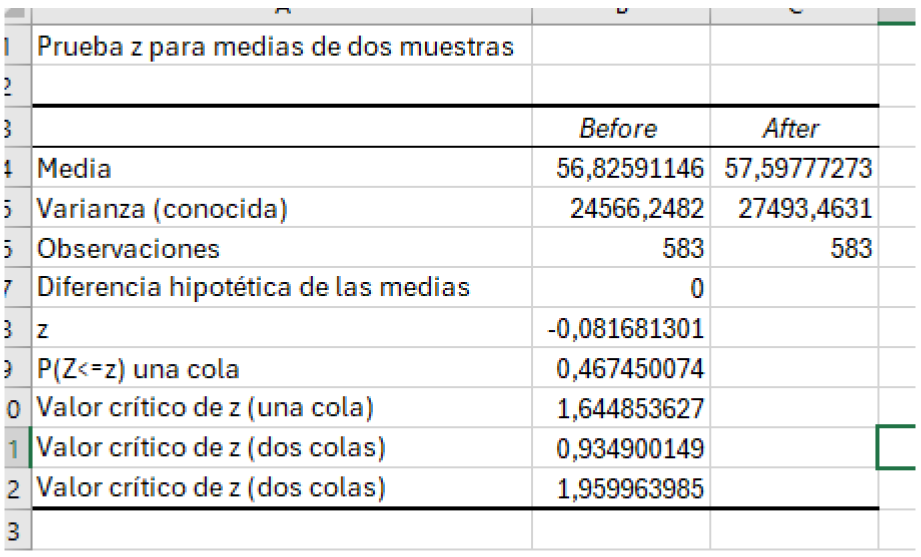
INTERVALOS DE CONFIANZA



	A	B	C	D	E	F	G	H	I
1	Before	After							
2	173,247899	97,6474		*BEFORE*			*AFTER*		
3	29,7399	23,4131							
4	16,401499	14,658201		Media	56,8259115		Media	57,5977727	
5	28,033599	18,821699		Error típico	6,4913755		Error típico	6,86721186	
6	8,105301	6,480101		Mediana	18,3751		Mediana	18,770199	
7	7,792299	4,66		Moda	#N/D		Moda	3,103	
8	52,3899	75,642699		Desviación estándar	156,736812		Desviación estándar	165,811529	
9	9,315199	11,616		Varianza de la muestra	24566,4283		Varianza de la muestra	27493,4631	
10	8,3237	8,0754		Curtosis	30,9574758		Curtosis	35,3087579	
11	6,4191	6,2851		Coeficiente de asimetría	5,42513383		Coeficiente de asimetría	5,80080239	
12	4,9675	4,9692		Rango	1246,5486		Rango	1353,7741	
13	5,803099	3,7589		Mínimo	1,8028		Mínimo	2,032801	
14	12,041099	3,3839		Máximo	1248,3514		Máximo	1355,8069	
15	11,431799	12,4301		Suma	33129,5064		Suma	33579,5015	
16	5,301301	6,7619		Cuenta	583		Cuenta	583	
17	5,9051	5,7354		Nivel de confianza(95,0%)	12,7493757		Nivel de confianza(95,0%)	13,4875365	
18	52,9013	54,5472							
19	7,6735	6,520101		Interval(ms)	44,0765358	69,5752871	Interval(ms)	44,1102363	71,0853092
20	4,891199	5,712401		Interval(s)	0,04407654	0,06957529	Interval(s)	0,04411024	0,07108531
21	5,2136	6,373701							

Analizando los intervalos, determinamos que el intervalo de confianza del 95% sin índices sería [44.07, 69.57] milisegundos, mientras que con índices será [44.11, 71,08] milisegundos. Al comprobar la correspondencia de milisegundos a segundos, se asegura que es un intervalo comprendido en menos de un segundo, lo cual era requerido para esta asignatura.

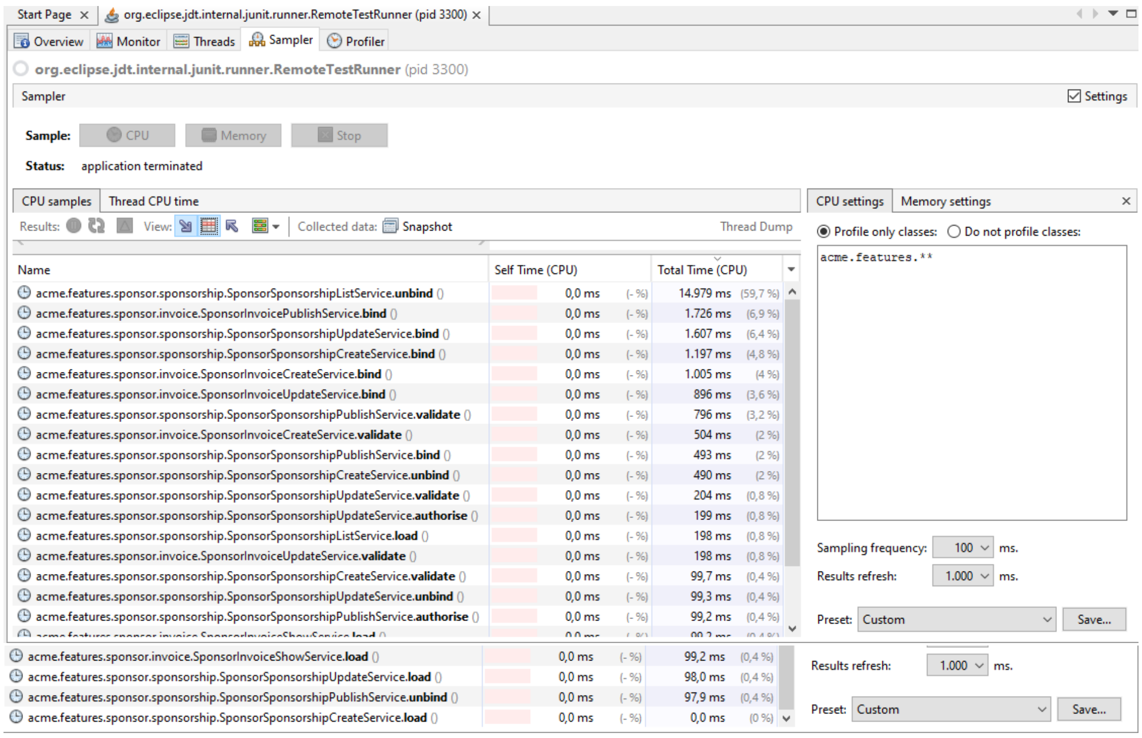
HIPOTESIS DE CONTRASTE



Prueba z para medias de dos muestras		
	Before	*After*
Media	56,82591146	57,59777273
Varianza (conocida)	24566,2482	27493,4631
Observaciones	583	583
Diferencia hipotética de las medias	0	
z	-0,081681301	
P(Z<=z) una cola	0,467450074	
Valor crítico de z (una cola)	1,644853627	
Valor crítico de z (dos colas)	0,934900149	
Valor crítico de z (dos colas)	1,959963985	

Como se puede observar en la celda “Valor crítico de z(dos colas)” está contenido entre 0 y 0,95 y que los cambios entre sin índices y con índices son mínimos

PERFILANDO SOFTWARE



org.eclipse.jdt.internal.junit.runner.RemoteTestRunner (pid 3300)

Sampler

Sample:

Status: application terminated

Name	Self Time (CPU)		Total Time (CPU)	
acme.features.sponsor.sponsorship.SponsorSponsorshipListService.**unbind** ()	0,0 ms	(- %)	14.979 ms	(59,7 %)
acme.features.sponsor.invoice.SponsorInvoicePublishService.**bind** ()	0,0 ms	(- %)	1.726 ms	(6,9 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipUpdateService.**bind** ()	0,0 ms	(- %)	1.607 ms	(6,4 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipCreateService.**bind** ()	0,0 ms	(- %)	1.197 ms	(4,8 %)
acme.features.sponsor.invoice.SponsorInvoiceCreateService.**bind** ()	0,0 ms	(- %)	1.005 ms	(4 %)
acme.features.sponsor.invoice.SponsorInvoiceUpdateService.**bind** ()	0,0 ms	(- %)	896 ms	(3,6 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipPublishService.**validate** ()	0,0 ms	(- %)	796 ms	(3,2 %)
acme.features.sponsor.invoice.SponsorInvoiceCreateService.**validate** ()	0,0 ms	(- %)	504 ms	(2 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipPublishService.**bind** ()	0,0 ms	(- %)	493 ms	(2 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipCreateService.**unbind** ()	0,0 ms	(- %)	490 ms	(2 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipUpdateService.**validate** ()	0,0 ms	(- %)	204 ms	(0,8 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipUpdateService.**authorise** ()	0,0 ms	(- %)	199 ms	(0,8 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipListService.**load** ()	0,0 ms	(- %)	198 ms	(0,8 %)
acme.features.sponsor.invoice.SponsorInvoiceUpdateService.**validate** ()	0,0 ms	(- %)	198 ms	(0,8 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipCreateService.**validate** ()	0,0 ms	(- %)	99,7 ms	(0,4 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipUpdateService.**unbind** ()	0,0 ms	(- %)	99,3 ms	(0,4 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipPublishService.**authorise** ()	0,0 ms	(- %)	99,2 ms	(0,4 %)
acme.features.sponsor.invoice.SponsorInvoiceShowService.**load** ()	0,0 ms	(- %)	99,2 ms	(0,4 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipUpdateService.**load** ()	0,0 ms	(- %)	98,0 ms	(0,4 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipPublishService.**unbind** ()	0,0 ms	(- %)	97,9 ms	(0,4 %)
acme.features.sponsor.sponsorship.SponsorSponsorshipCreateService.**load** ()	0,0 ms	(- %)	0,0 ms	(0 %)

(●) Profile only classes: ( ) Do not profile classes:

```
acme.features.**
```

Sampling frequency: 100 ms.

Results refresh: 1.000 ms.

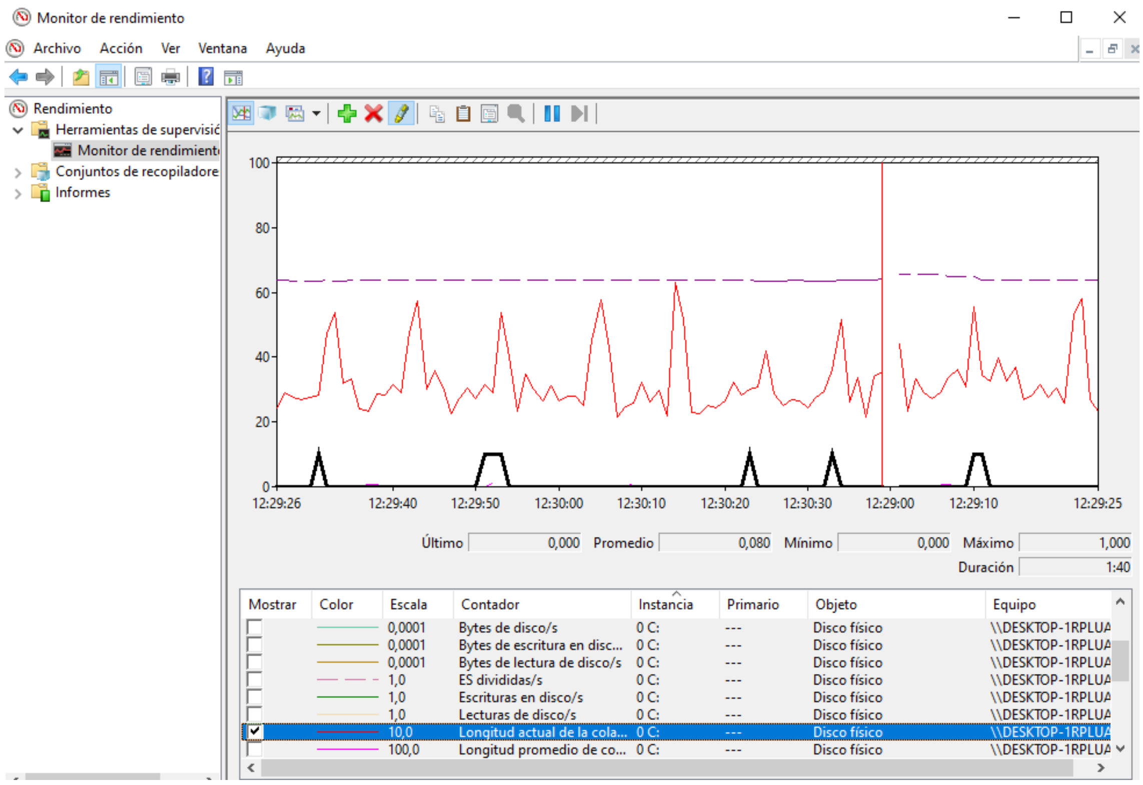
Preset: Custom

Results refresh: 1.000 ms.

Preset: Custom

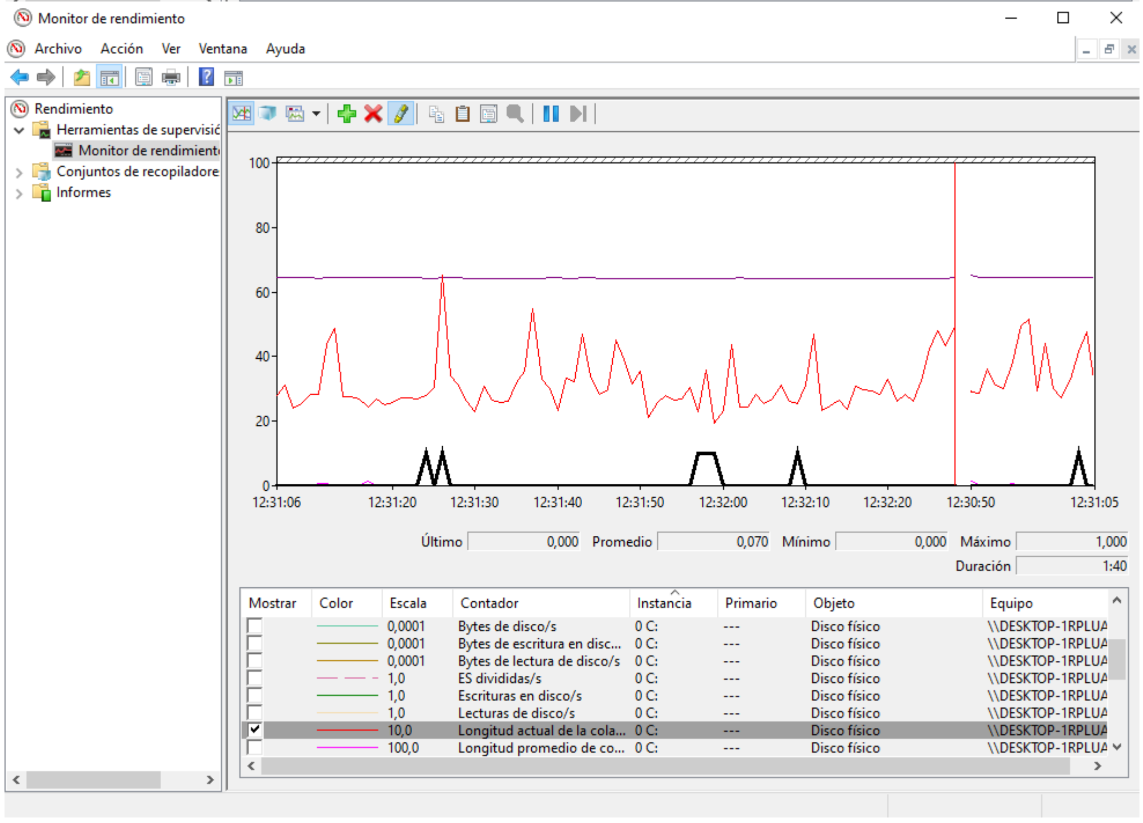
Podemos apreciar que el unbind del sponsorshipList es el que más tiempo consume y esto se correlaciona con los gráficos previamente vistos donde el List del sponsorship era el que más tiempo consumía.

PERFILANDO HARDWARE



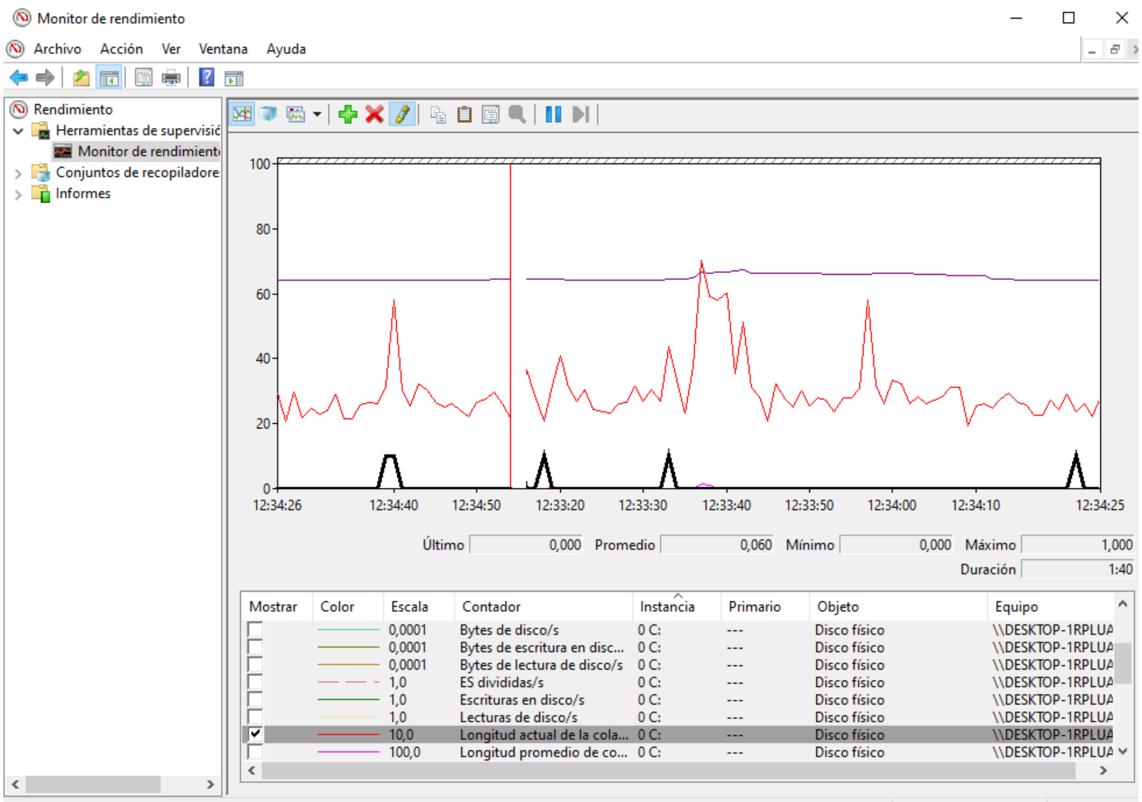
Último	Promedio	Mínimo	Máximo	Duración
0,000	0,080	0,000	1,000	1:40

Mostrar	Color	Escala	Contador	Instancia	Primario	Objeto	Equipo
☐		0,0001	Bytes de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		0,0001	Bytes de escritura en disc...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		0,0001	Bytes de lectura de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		1,0	ES divididas/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		1,0	Escrituras en disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		1,0	Lecturas de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
✓		10,0	Longitud actual de la cola...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		100,0	Longitud promedio de co...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA



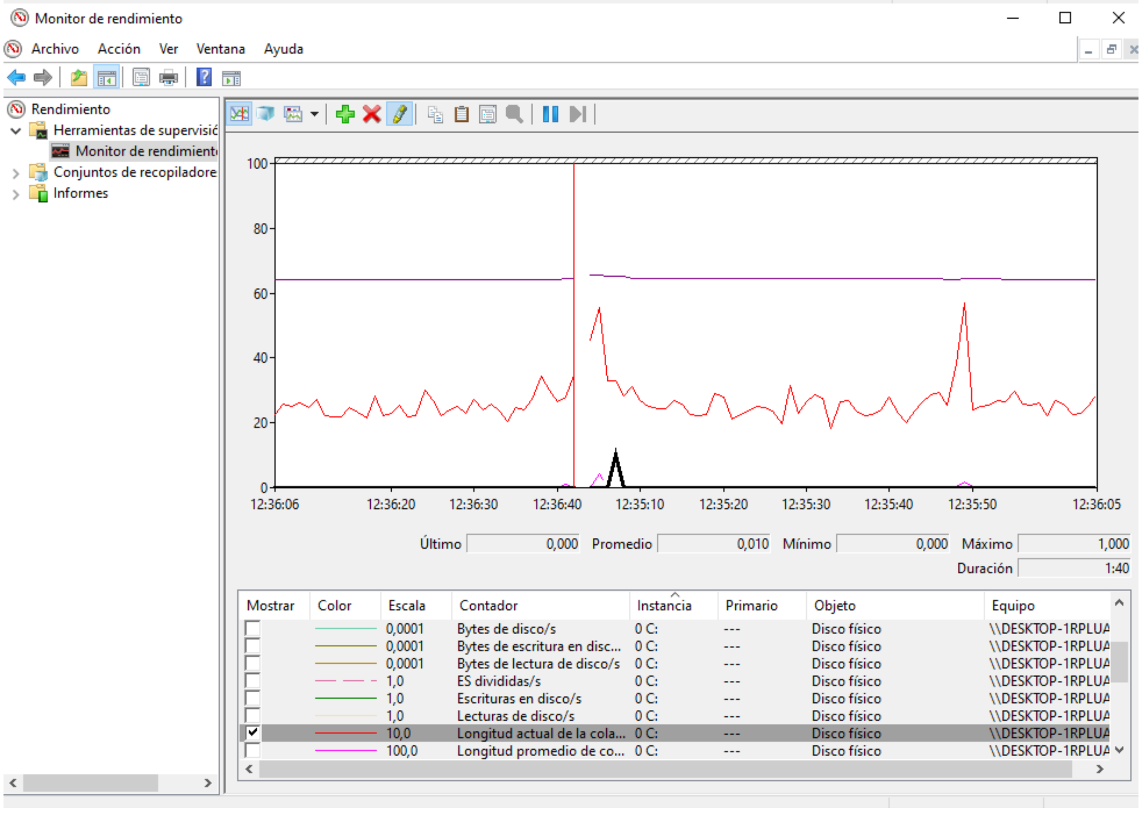
Último	Promedio	Mínimo	Máximo	Duración
0,000	0,070	0,000	1,000	1:40

Mostrar	Color	Escala	Contador	Instancia	Primario	Objeto	Equipo
☐		0,0001	Bytes de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		0,0001	Bytes de escritura en disc...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		0,0001	Bytes de lectura de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		1,0	ES divididas/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		1,0	Escrituras en disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		1,0	Lecturas de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
✓		10,0	Longitud actual de la cola...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐		100,0	Longitud promedio de co...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA



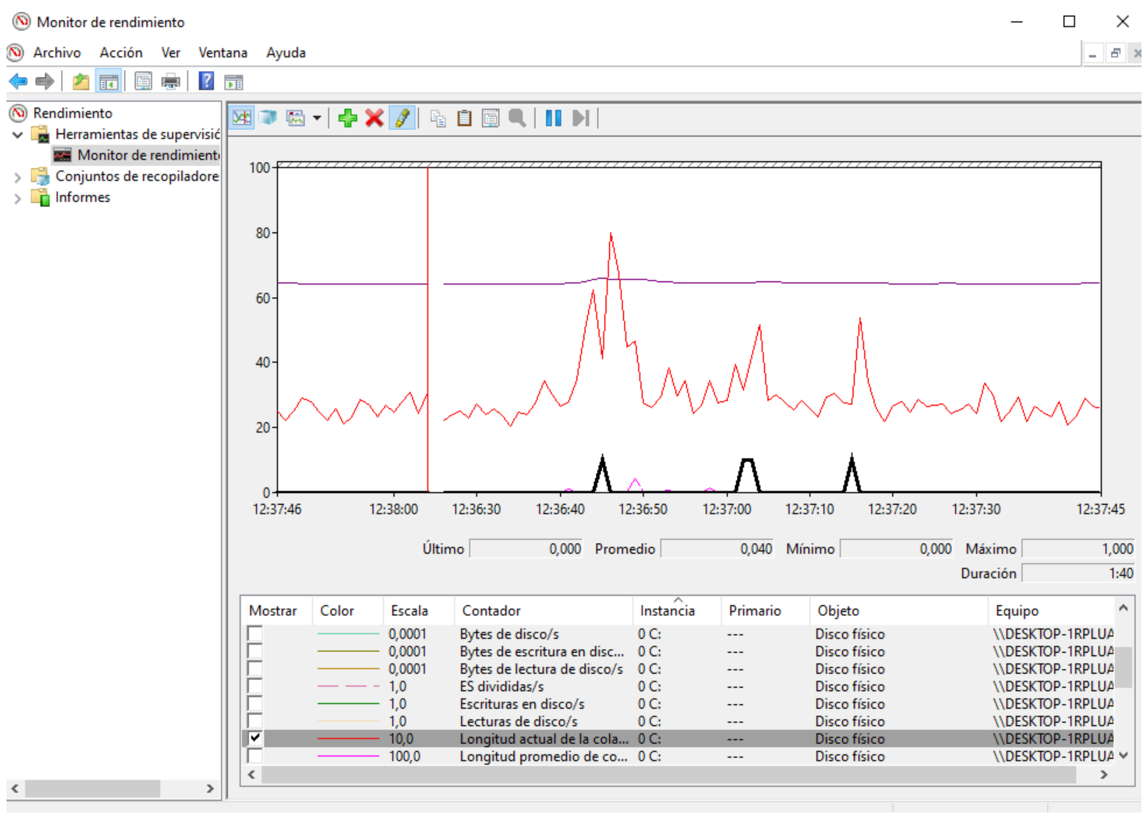
Último 0,000 Promedio 0,060 Mínimo 0,000 Máximo 1,000
Duración 1:40

Mostrar	Color	Escala	Contador	Instancia	Primario	Objeto	Equipo
	🟦	0,0001	Bytes de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟫	0,0001	Bytes de escritura en disc...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟫	0,0001	Bytes de lectura de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟥	1,0	ES divididas/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟩	1,0	Escrituras en disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟨	1,0	Lecturas de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
✓	🟥	10,0	Longitud actual de la cola...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟪	100,0	Longitud promedio de co...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA



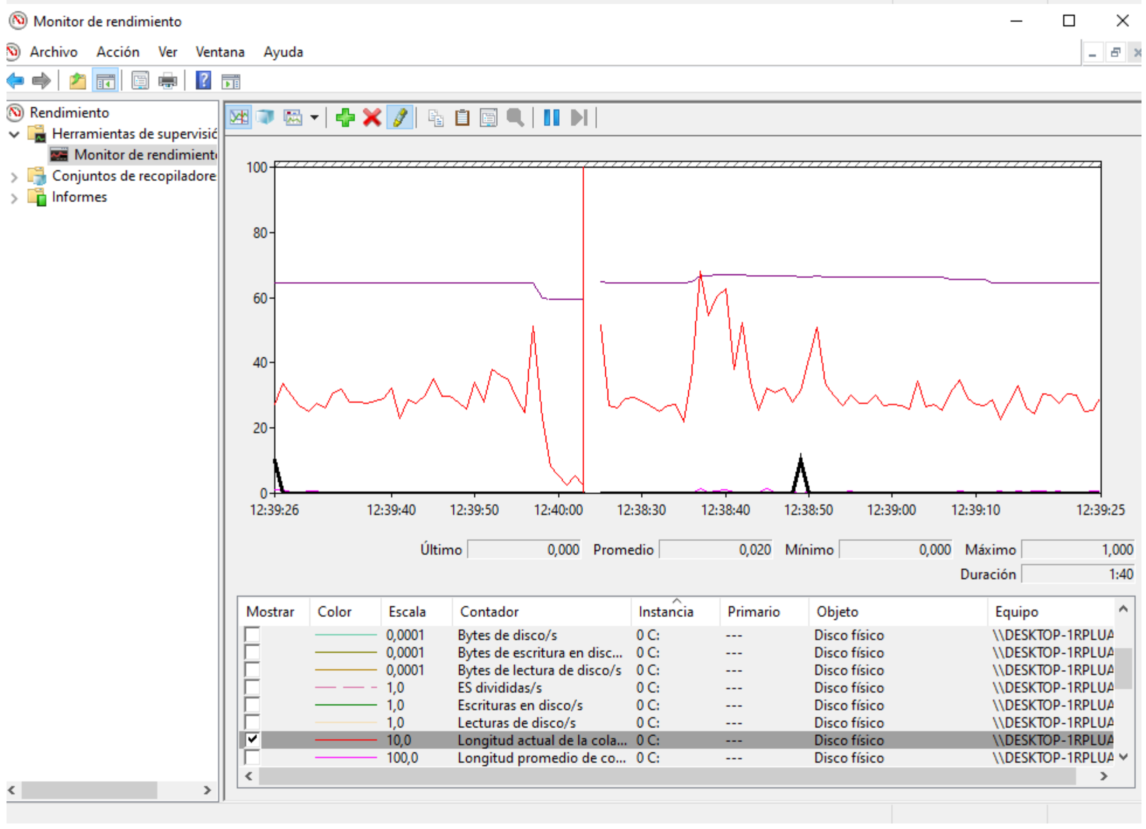
Último 0,000 Promedio 0,010 Mínimo 0,000 Máximo 1,000
Duración 1:40

Mostrar	Color	Escala	Contador	Instancia	Primario	Objeto	Equipo
	🟦	0,0001	Bytes de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟫	0,0001	Bytes de escritura en disc...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟫	0,0001	Bytes de lectura de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟥	1,0	ES divididas/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟩	1,0	Escrituras en disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟨	1,0	Lecturas de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
✓	🟥	10,0	Longitud actual de la cola...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
	🟪	100,0	Longitud promedio de co...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA



Último 0,000 Promedio 0,040 Mínimo 0,000 Máximo 1,000
Duración 1:40

Mostrar	Color	Escala	Contador	Instancia	Primario	Objeto	Equipo
☐	—	0,0001	Bytes de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	—	0,0001	Bytes de escritura en disc...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	—	0,0001	Bytes de lectura de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	– –	1,0	ES divididas/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	—	1,0	Escrituras en disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	—	1,0	Lecturas de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
✓	—	10,0	Longitud actual de la cola...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	—	100,0	Longitud promedio de co...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA



Último 0,000 Promedio 0,020 Mínimo 0,000 Máximo 1,000
Duración 1:40

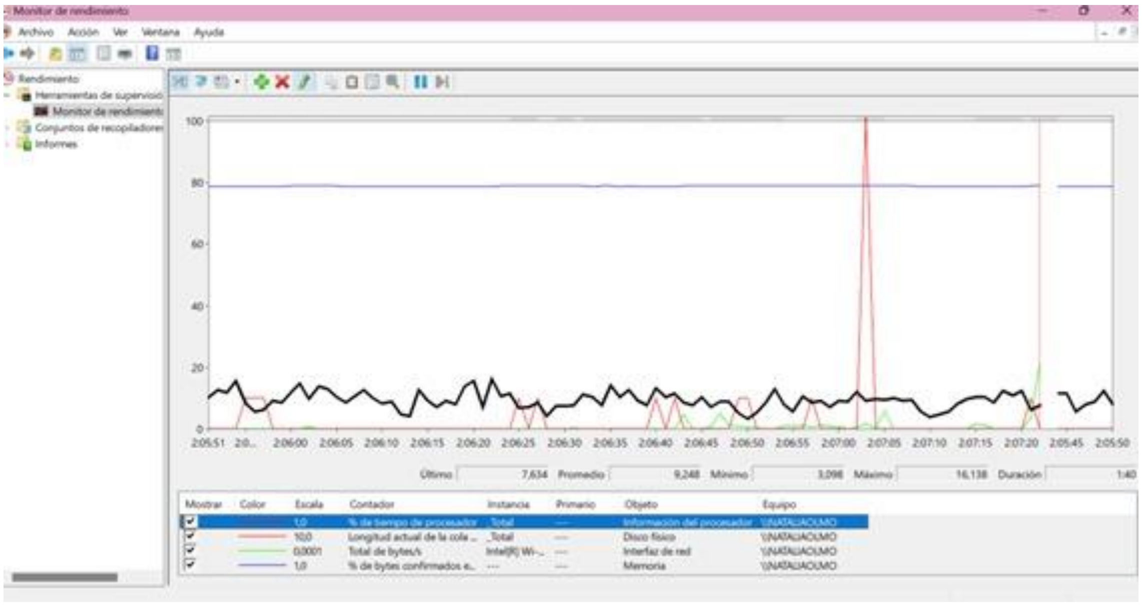
Mostrar	Color	Escala	Contador	Instancia	Primario	Objeto	Equipo
☐	—	0,0001	Bytes de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	—	0,0001	Bytes de escritura en disc...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	—	0,0001	Bytes de lectura de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	– –	1,0	ES divididas/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	—	1,0	Escrituras en disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	—	1,0	Lecturas de disco/s	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
✓	—	10,0	Longitud actual de la cola...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA
☐	—	100,0	Longitud promedio de co...	0 C:	---	Disco físico	\\DESKTOP-1RPLUA

Según la leyenda de colores, podemos ver que se ha representado de la siguiente forma:

- rojo: información del procesador
- negro (pone rojo 10 pero eso es negro): longitud actual de cola del disco físico
- rosa: total de bytes de la interfaz de red
- morado: %de bytes confirmados en memoria

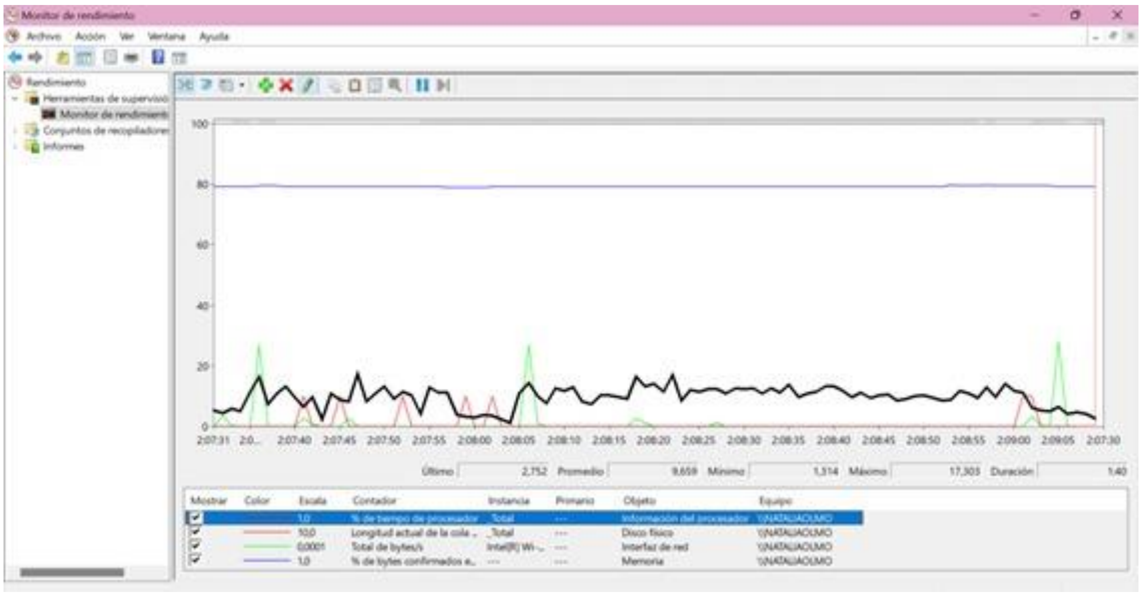
Con respecto al total de bytes transferidos a través de la interfaz de red, tenemos valores que se mantienen constantes, oscilando entre el 40% y el 60% y con un pico del 80%, y no se usan el 100% del tiempo. Esto refleja que en este entorno la red no es intensamente utilizada o el ancho de banda disponible es suficiente para manejar la carga de trabajo sin problemas.

A continuación, los gráficos del ordenador del compañero que también ejecuto las pruebas:



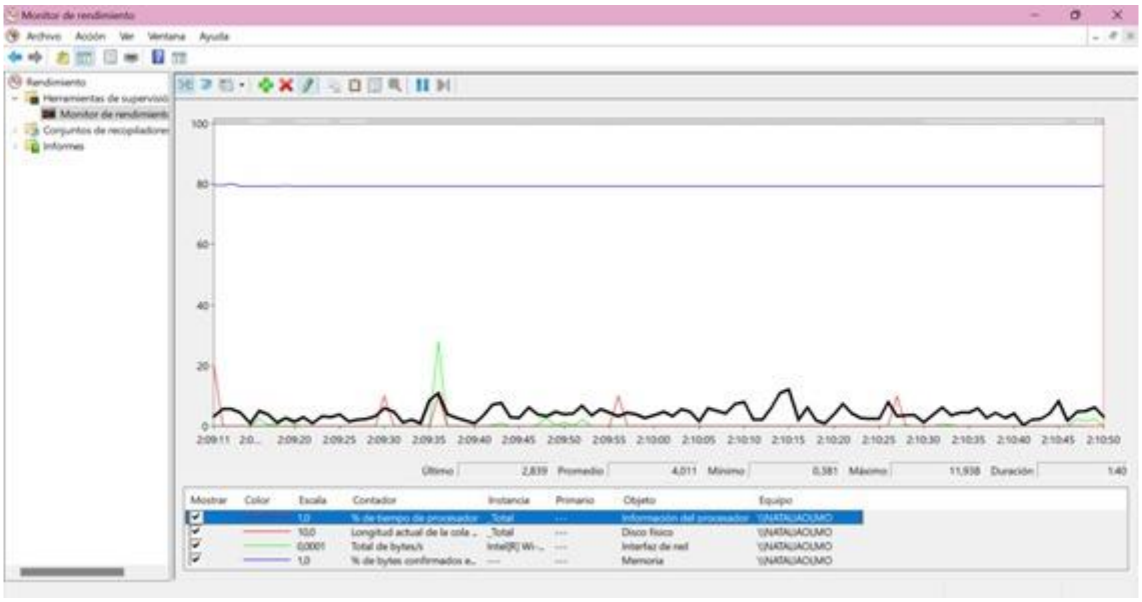
Último 7,634 Promedio 9,248 Mínimo 3,098 Máximo 16,138 Duración 1:40

Mostrar	Color	Escala	Contador	Instancia	Primario	Objeto	Equipo
✓		1,0	% de tiempo de procesador	_Total	---	Información del procesador	\\NATALIAOLMO
✓	🟥	10,0	Longitud actual de la cola ...	_Total	---	Disco físico	\\NATALIAOLMO
✓	🟩	0,0001	Total de bytes/s	Intel(R) Wi-...	---	Interfaz de red	\\NATALIAOLMO
✓	🟦	1,0	% de bytes confirmados e...	---	---	Memoria	\\NATALIAOLMO



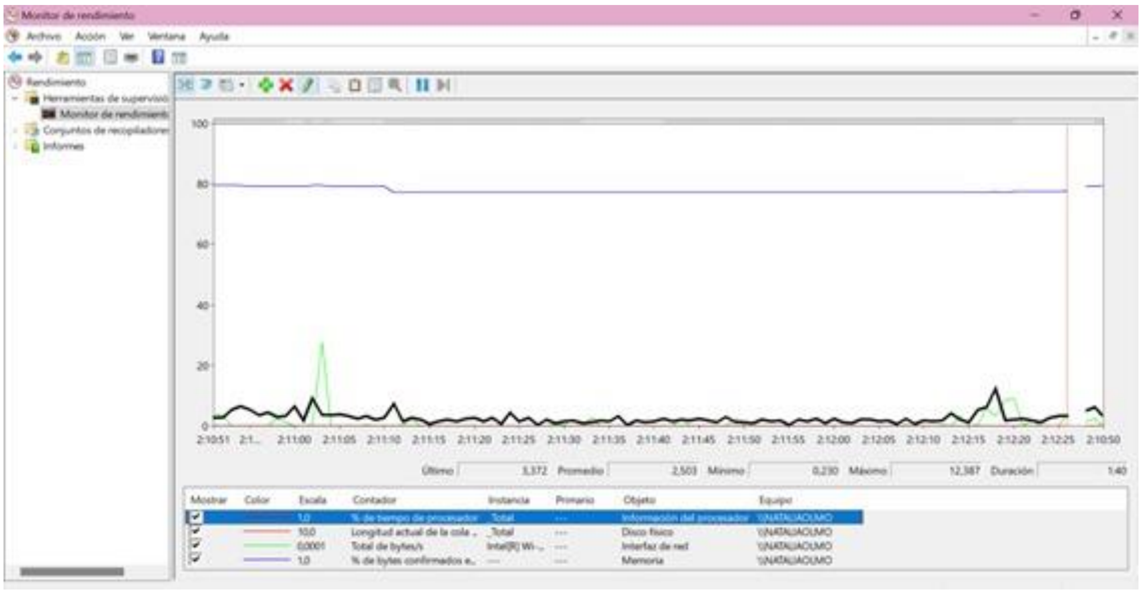
Último 2,752 Promedio 9,659 Mínimo 1,314 Máximo 17,303 Duración 1:40

Mostrar	Color	Escala	Contador	Instancia	Primario	Objeto	Equipo
✓		1,0	% de tiempo de procesador	_Total	---	Información del procesador	\\NATALIAOLMO
✓	🟥	10,0	Longitud actual de la cola ...	_Total	---	Disco físico	\\NATALIAOLMO
✓	🟩	0,0001	Total de bytes/s	Intel(R) Wi-...	---	Interfaz de red	\\NATALIAOLMO
✓	🟦	1,0	% de bytes confirmados e...	---	---	Memoria	\\NATALIAOLMO



Último 2,839 Promedio 4,011 Mínimo 0,381 Máximo 11,938 Duración 1:40

Mostrar	Color	Escala	Contador	Instancia	Primario	Objeto	Equipo
✓		1,0	% de tiempo de procesador	_Total	---	Información del procesador	\\NATALIAOLMO
✓	🟥	10,0	Longitud actual de la cola ...	_Total	---	Disco físico	\\NATALIAOLMO
✓	🟩	0,0001	Total de bytes/s	Intel(R) Wi-...	---	Interfaz de red	\\NATALIAOLMO
✓	🟦	1,0	% de bytes confirmados e...	---	---	Memoria	\\NATALIAOLMO



Último 3,372 Promedio 2,503 Mínimo 0,230 Máximo 12,387 Duración 1:40

Mostrar	Color	Escala	Contador	Instancia	Primario	Objeto	Equipo
✓		1,0	% de tiempo de procesador	_Total	---	Información del procesador	\\NATALIAOLMO
✓	🟥	10,0	Longitud actual de la cola ...	_Total	---	Disco físico	\\NATALIAOLMO
✓	🟩	0,0001	Total de bytes/s	Intel(R) Wi-...	---	Interfaz de red	\\NATALIAOLMO
✓	🟦	1,0	% de bytes confirmados e...	---	---	Memoria	\\NATALIAOLMO