```java

package acme.features.client.contract;

import java.util.Collection;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import acme.client.data.datatypes.Money;
import acme.client.data.models.Dataset;
import acme.client.services.AbstractService;
import acme.client.views.SelectChoices;
import acme.entities.contract.Contract;
import acme.entities.project.Project;
import acme.features.client.progresslog.ClientProgressLogRepository;
import acme.roles.Client;

@Service
public class ClientContractPublishService extends AbstractService<Client, Contract> {

    @Autowired
    private ClientContractRepository    repository;
    @Autowired
    private ClientProgressLogRepository plRepo;


    @Override
    public void authorise() {
        boolean status;
        Contract contract;
        int id = super.getRequest().getData("id", int.class);
        contract = this.repository.findContractById(id);
        status = contract != null && contract.isDraft() && super.getRequest().getPrincipal().hasRole(Client.class) && contract.getClient().getId() == super.getRequest().getPrincipal().getActiveRoleId();
        super.getResponse().setAuthorised(status);
    }

    @Override
    public void load() {
        Contract contract;
        int id;

        id = super.getRequest().getData("id", int.class);
        contract = this.repository.findContractById(id);

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

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

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

        super.bind(contract, "code", "moment", "providerName", "customerName", "goals", "budget", "isDraft", "project", "isDraft");
        contract.setProject(project);
    }

    @Override
    public void validate(final Contract contract) {
        assert contract != null;
        Collection<Contract> contracts = null;
        if (!super.getBuffer().getErrors().hasErrors("code")) {
            Contract existing;

            existing = this.repository.findOneContractByCode(contract.getCode());

            super.state(existing == null || existing.equals(contract), "code", "client.contract.form.error.duplicated");
        }
        if (!super.getBuffer().getErrors().hasErrors("budget"))
            super.state(contract.getBudget().getAmount() >= 0, "budget", "client.contract.form.error.negative-budget");
        if (!super.getBuffer().getErrors().hasErrors("budget")) {
            contracts = this.repository.findAllContractsOfAProjectById(contract.getProject().getId());
            double totalBudget = contracts.stream().filter(p -> p.getId() != contract.getId()).mapToDouble(c -> this.eurConverter(c.getBudget())).sum();
            double projectCost = contract.getProject().getCost().getAmount();
            totalBudget += this.eurConverter(contract.getBudget());
            super.state(totalBudget <= projectCost, "budget", "client.contract.form.error.exceeds-project-cost");
        }

    }

    private double eurConverter(final Money money) {
        String currency = money.getCurrency();
        double amount = money.getAmount();

        if (currency.equals("EUR"))
```

```java
            amount = amount;
        else if (currency.equals("USD"))
            amount = amount * 0.90; // Tasa aproximada de conversión USD a EUR
        else if (currency.equals("GBP"))
            amount = amount * 1.17; // Tasa aproximada de conversión GBP a EUR
        else
            super.state(false, "budget", "client.contract.unsopportedCurrency");
        return amount;
    }

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

        contract.setDraft(false);
        this.repository.findAllProgressLogsByContractId(contract.getId()).stream().forEach(x -> {
            x.setDraft(false);
            this.plRepo.save(x);
        });
        this.repository.save(contract);

    }
    @Override
    public void unbind(final Contract contract) {
        assert contract != null;
        boolean isDraft;
        SelectChoices choices;

        choices = SelectChoices.from(this.repository.findAllPublishedProjects(), "title", contract.getProject());
        isDraft = contract.isDraft() == true;

        Dataset dataset;

        dataset = super.unbind(contract, "code", "moment", "providerName", "customerName", "goals", "budget", "isDraft", "project");
        dataset.put("contractId", contract.getId());
        dataset.put("isDraft", isDraft);
        dataset.put("projects", choices);
        super.getResponse().addData(dataset);
    }

}

```