```java

package acme.features.manager.project;

import java.util.stream.Stream;

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

import acme.client.data.models.Dataset;
import acme.client.services.AbstractService;
import acme.entities.project.Project;
import acme.entities.project.acceptedCurrency;
import acme.roles.Manager;

@Service
public class ManagerProjectPublishService extends AbstractService<Manager, Project> {

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

    @Autowired
    private ManagerProjectRepository repository;

    // AbstractService<Manager, Project> --------------------------------------


    @Override
    public void authorise() {
        boolean status;
        int masterId;
        Project project;

        masterId = super.getRequest().getData("id", int.class);
        project = this.repository.findProjectById(masterId);
        boolean publishable;
        publishable = this.repository.findAllUserStoriesOfAProjectById(project.getId()).stream().allMatch(x -> x.isDraft() == false) && this.repository.findAllUserStoriesOfAProjectById(project.getId()).size() > 0 && project.isHasFatalErrors() == false;
        status = project != null && publishable && project.isDraft() && super.getRequest().getPrincipal().hasRole(Manager.class);

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

    @Override
    public void load() {
        Project project;
        int id;

        id = super.getRequest().getData("id", int.class);
        project = this.repository.findProjectById(id);

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

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

        super.bind(project, "code", "title", "abstractText", "hasFatalErrors", "cost", "link", "isDraft");
    }
```

```
58
59      @Override
60      public void validate(final Project project) {
61          boolean condition = this.repository.findAllUserStoriesOfAProjectById(project.getId
    ()).stream().allMatch(x -> x.isDraft() == false) &&
    this.repository.findAllUserStoriesOfAProjectById(project.getId()).size() > 0
62              && project.isHasFatalErrors() == false;
63          super.state(condition, "*", "manager.project.form.error.publishable");
64
65          if (!super.getBuffer().getErrors().hasErrors("hasFatalErrors"))
66              super.state(!project.isHasFatalErrors(), "hasFatalErrors",
    "manager.project.form.error.fatal-errors");
67
68          if (!super.getBuffer().getErrors().hasErrors("cost")) {
69              boolean isCurrencyAccepted = Stream.of(acceptedCurrency.values()).map(x ->
    x.toString().toLowerCase().trim()).anyMatch(currency -> currency.equals(project.getCost
    ().getCurrency().toLowerCase().trim()));
70
71              super.state(isCurrencyAccepted, "cost",
    "manager.project.form.error.incorrectConcurrency");
72              super.state(project.getCost().getAmount() >= 0, "cost",
    "manager.project.form.error.negative-cost");
73          }
74
75          if (!super.getBuffer().getErrors().hasErrors("code")) {
76              Project existing;
77
78              existing = this.repository.findOneProjectByCode(project.getCode());
79
80              super.state(existing == null || existing.equals(project), "code",
    "manager.project.form.error.duplicated");
81          }
82
83      }
84
85      @Override
86      public void perform(final Project project) {
87          assert project != null;
88
89          project.setDraft(false);
90          this.repository.save(project);
91      }
92
93      @Override
94      public void unbind(final Project project) {
95          assert project != null;
96          boolean userStoriesPublishables;
97          boolean isDraft;
98
99          userStoriesPublishables = this.repository.findAllUserStoriesOfAProjectById
    (project.getId()).stream().allMatch(x -> x.isDraft() == false) &&
    this.repository.findAllUserStoriesOfAProjectById(project.getId()).size() > 0
100             && project.isHasFatalErrors() == false;
101         isDraft = project.isDraft() == true;
102
103         Dataset dataset;
104
105         dataset = super.unbind(project, "code", "title", "abstractText", "hasFatalErrors",
    "cost", "link", "isDraft");
106         dataset.put("projectId", project.getId());
107         dataset.put("publishable", userStoriesPublishables);
108         dataset.put("isDraft", isDraft);
```

```
109
110         super.getResponse().addData(dataset);
111     }
112
113 }
114
```