

```
1
2 package acme.features.manager.project;
3
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.stereotype.Service;
6
7 import acme.client.data.models.Dataset;
8 import acme.client.services.AbstractService;
9 import acme.entities.project.Project;
10 import acme.roles.Manager;
11
12 @Service
13 public class ManagerProjectDeleteService extends AbstractService<Manager, Project> {
14
15     // Internal state -----
16
17     @Autowired
18     private ManagerProjectRepository repository;
19
20     // AbstractService interface -----
21
22     @Override
23     public void authorise() {
24         boolean status;
25         int masterId;
26         Project project;
27
28         masterId = super.getRequest().getData("id", int.class);
29         project = this.repository.findProjectById(masterId);
30         status = project != null && project.isDraft() && super.getRequest().getPrincipal
31             ().hasRole(Manager.class) && project.getManager().getId() == super.getRequest().getPrincipal
32             ().getActiveRoleId();
33         super.getResponse().setAuthorised(status);
34     }
35
36     @Override
37     public void load() {
38         Project project;
39         int id;
40
41         id = super.getRequest().getData("id", int.class);
42         project = this.repository.findProjectById(id);
43
44         super.getBuffer().addData(project);
45     }
46
47     @Override
48     public void bind(final Project project) {
49         assert project != null;
50
51         super.bind(project, "code", "title", "abstractText", "hasFatalErrors", "cost",
52             "link", "isDraft");
53     }
54
55     @Override
56     public void validate(final Project project) {
57         assert project != null;
58     }
59
60     @Override
```

```
60     public void perform(final Project project) {
61         assert project != null;
62
63         this.repository.deleteAll(this.repository.findAllAssignmentsOfAProjectById
64         (project.getId()));
65         this.repository.deleteAll(this.repository.findAllProgressLogsByProjectId
66         (project.getId()));
67         this.repository.deleteAll(this.repository.findAllContractOfAProjectById
68         (project.getId()));
69         this.repository.deleteAll(this.repository.findAllAuditRecordsOfAProjectById
70         (project.getId()));
71         this.repository.deleteAll(this.repository.findAllCodeAuditsOfAProjectById
72         (project.getId()));
73         this.repository.deleteAll(this.repository.findAllSponsorShipOfAProjectById
74         (project.getId()));
75         this.repository.deleteAll(this.repository.findAllTrainingSessionsOfAProjectById
76         (project.getId()));
77         this.repository.deleteAll(this.repository.findAllTrainingModuleOfAProjectById
78         (project.getId()));
79         this.repository.delete(project);
80     }
81
82     @Override
83     public void unbind(final Project project) {
84         assert project != null;
85         boolean userStoriesPublishables;
86         boolean isDraft;
87
88         userStoriesPublishables = this.repository.findAllUserStoriesOfAProjectById
89         (project.getId()).stream().allMatch(x -> x.isDraft() == false) &&
90         this.repository.findAllUserStoriesOfAProjectById(project.getId()).size() > 0
91         && project.isHasFatalErrors() == false;
92         isDraft = project.isDraft() == true;
93
94         Dataset dataset;
95
96         dataset = super.unbind(project, "code", "title", "abstractText", "hasFatalErrors",
97         "cost", "link", "isDraft");
98         dataset.put("masterId", project.getId());
99         dataset.put("publishable", userStoriesPublishables);
100        dataset.put("isDraft", isDraft);
101
102        super.getResponse().addData(dataset);
103    }
104 }
```