```java

package acme.features.manager.assignment;

import java.util.Collection;

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.client.views.SelectChoices;
import acme.entities.project.Assignment;
import acme.entities.project.UserStory;
import acme.features.manager.project.ManagerProjectRepository;
import acme.roles.Manager;

@Service
public class ManagerAssignmentDeleteService extends AbstractService<Manager, Assignment> {

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

    @Autowired
    private ManagerProjectRepository repository;

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


    @Override
    public void authorise() {
        boolean status;
        int assignmentId = super.getRequest().getData("id", int.class);
        Manager manager = this.repository.findManagerProjectByAssignmentId(assignmentId);
        status = super.getRequest().getPrincipal().getActiveRoleId() == manager.getId() && super.getRequest().getPrincipal().hasRole(Manager.class);

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

    @Override
    public void load() {
        Assignment assignment;
        int id;

        id = super.getRequest().getData("id", int.class);
        assignment = this.repository.findAssignmentById(id);

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

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

        super.bind(assignment, "project.title", "userStory");
    }

    @Override
    public void validate(final Assignment assignment) {
        boolean updateable = this.repository.findProjectOfAnAssignmentByAssignmentId(assignment.getId()).isDraft();

        if (!super.getBuffer().getErrors().hasErrors("project"))
```

```java
            super.state(updateable, "*", "manager.project.form.updateable");

    }

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

        this.repository.delete(assignment);
    }

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

        Dataset dataset;

        int id = super.getRequest().getPrincipal().getActiveRoleId();
        SelectChoices userStoriesChoices;
        Collection<UserStory> userStories = this.repository.findAllUserStoriesOfAManagerById(id);

        userStoriesChoices = SelectChoices.from(userStories, "title", assigment.getUserStory());

        dataset = super.unbind(assigment, "project.title", "userStory");

        dataset.put("masterId", assigment.getProject().getId());
        dataset.put("userStories", userStoriesChoices);

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

}
```