

# DP2 2024

## Acme Software Factory

Repositorio: <https://github.com/DP2-2024-C1-029/Acme-Software-Factory.git>

### Miembro:

- Juan José Gómez Borrallo (juagombor@alum.us.es)

**Tutor:** José González Enríquez  
25/05/2024

GRUPO C1.029  
Versión 1.0

## Índice

|   |    |
|---|----|
| Historial de versiones.....             | 3  |
| Capítulo 1 – Pruebas funcionales.....   | 4  |
| Training Module .....                   | 4  |
| Training Session.....                   | 8  |
| Capítulo 2 – Pruebas de desempeño ..... | 14 |
| Bibliografía .....                      | 16 |

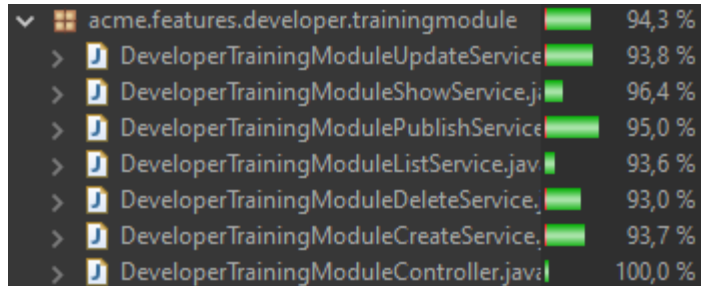
## Historial de versiones

| Fecha      | Versión | Descripción          | Entrega |
|------------|---------|----------------------|---------|
| 25/05/2024 | V1.0    | Inicio del documento | D04     |
|            |         |                      |         |

## Capítulo 1 – Pruebas funcionales

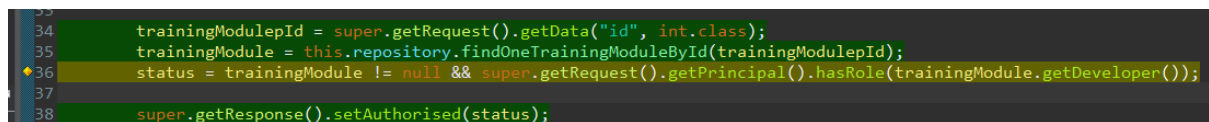
### Training Module

Tras ejecutar todos los test, se puede observar que para training module se cubre el 94.3%, valor que está por encima de la recomendación mínima del 90% que debería cubrir al menos todos los test.



|   |         |
|---|---------|
| acme.features.developer.trainingmodule  | 94,3 %  |
| DeveloperTrainingModuleUpdateService    | 93,8 %  |
| DeveloperTrainingModuleShowService.java | 96,4 %  |
| DeveloperTrainingModulePublishService   | 95,0 %  |
| DeveloperTrainingModuleListService.java | 93,6 %  |
| DeveloperTrainingModuleDeleteService    | 93,0 %  |
| DeveloperTrainingModuleCreateService    | 93,7 %  |
| DeveloperTrainingModuleController.java  | 100,0 % |

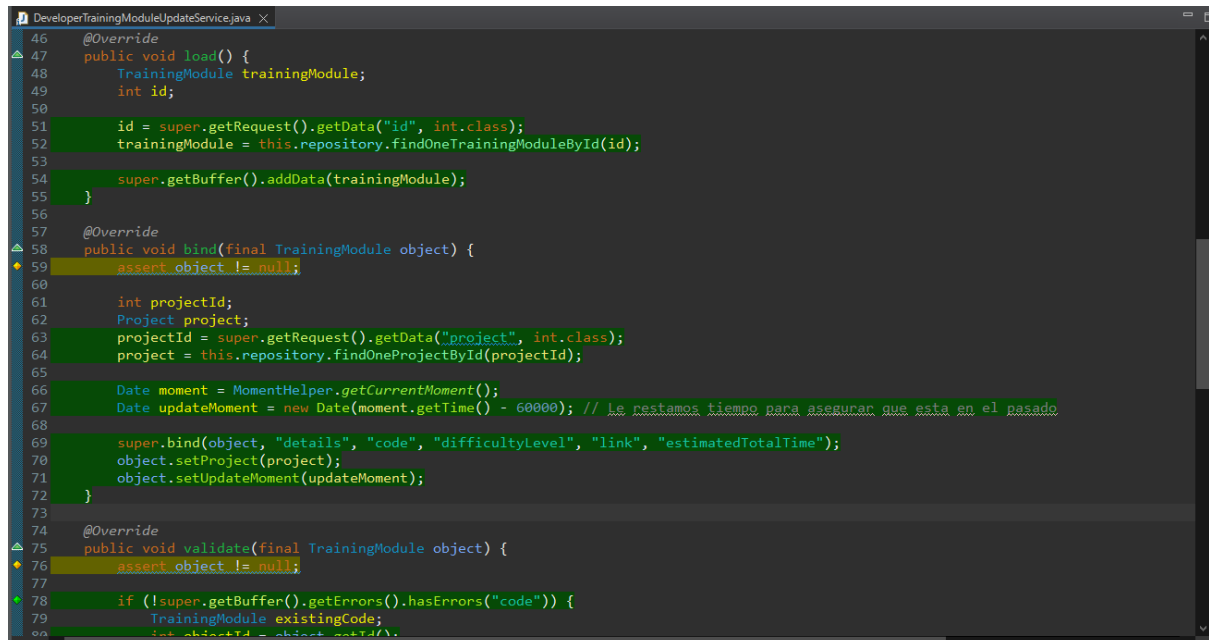
En primer lugar, para no repetirlo durante todo el documento, se va a comentar que las líneas que los “assert” siempre aparecen en amarillo, y que el status tampoco se puede poner en verde, porque hay un caso que nunca se puede probar.



```
34 trainingModuleId = super.getRequest().getData("id", int.class);
35 trainingModule = this.repository.findOneTrainingModuleById(trainingModuleId);
36 status = trainingModule != null && super.getRequest().getPrincipal().hasRole(trainingModule.getDeveloper());
37
38 super.getResponse().setAuthorised(status);
```

Voy a empezar hablando por el UpdateService.

Para el update service podemos observar que todo está en verde, menos alguna validación.



```
46 @Override
47 public void load() {
48     TrainingModule trainingModule;
49     int id;
50
51     id = super.getRequest().getData("id", int.class);
52     trainingModule = this.repository.findOneTrainingModuleById(id);
53
54     super.getBuffer().addData(trainingModule);
55 }
56
57 @Override
58 public void bind(final TrainingModule object) {
59     assert object != null;
60
61     int projectId;
62     Project project;
63     projectId = super.getRequest().getData("project", int.class);
64     project = this.repository.findOneProjectById(projectId);
65
66     Date moment = MomentHelper.getCurrentMoment();
67     Date updateMoment = new Date(moment.getTime() - 60000); // le restamos tiempo para asegurar que esta en el pasado
68
69     super.bind(object, "details", "code", "difficultyLevel", "link", "estimatedTotalTime");
70     object.setProject(project);
71     object.setUpdateMoment(updateMoment);
72 }
73
74 @Override
75 public void validate(final TrainingModule object) {
76     assert object != null;
77
78     if (!super.getBuffer().getErrors().hasErrors("code")) {
79         TrainingModule existingCode;
80         int objectId = object.getId();
```

Sin embargo, sí que tenemos en las validaciones algunas líneas amarillas porque hay pruebas que solo se pueden testear con postman.

```
97     if (!super.getBuffer().getErrors().hasErrors("updateMoment"))
98         super.state(object.getUpdateMoment().compareTo(object.getCreationMoment()) > 0, "updateMoment", "developer.trainingModule.form
```

Por ejemplo, aquí tenemos el “updateMoment” que siempre tiene que ser posterior al creationMoment. Sin embargo, tal y como he desarrollado el formulario, los campos aparecen bloqueado, y se le autoasigna los valores del sistema cuando hace una operación, por lo que manualmente no se pueden hacer pruebas. Sí hay que tener en cuenta esta validación por si se intenta hacer algún hack para meter algún dato no valido. El resto de validaciones que no necesitan postman si que están en verde, como por ejemplo el código duplicado.

```
78     if (!super.getBuffer().getErrors().hasErrors("code")) {
79         TrainingModule existingCode;
80         int objectId = object.getId();
81
82         existingCode = this.repository.findTrainingModuleByCode(object.getCode());
83
84         boolean isDuplicatedCode = existingCode != null && existingCode.getId() != objectId;
85         super.state(!isDuplicatedCode, "code", "developer.trainingModule.form.error.duplicated-code");
86     }
87 }
```

A continuación, vamos como el ShowService, el cual está todo en verde.

```
41     @Override
42     public void load() {
43         TrainingModule object;
44         int developerId;
45
46         developerId = super.getRequest().getData("id", int.class);
47         object = this.repository.findOneTrainingModuleById(developerId);
48
49         super.getBuffer().addData(object);
50     }
51
52     @Override
53     public void unbind(final TrainingModule object) {
54         assert object != null;
55
56         SelectChoices choices;
57         Dataset dataset;
58
59         Collection<Project> projects;
60         SelectChoices choicesProject;
61
62         choices = SelectChoices.from(DifficultyLevel.class, object.getDifficultyLevel());
63         projects = this.repository.findManyProjects();
64         choicesProject = SelectChoices.from(projects, "title", object.getProject());
65
66         dataset = super.unbind(object, "creationMoment", "details", "code", "updateMoment", "estimatedTotalTime", "link", "draftMode");
67         dataset.put("difficultyLevel", choices.getSelected().getKey());
68         dataset.put("difficultyLevels", choices);
69         dataset.put("project", choicesProject.getSelected().getKey());
70         dataset.put("projects", choicesProject);
71
72         super.getResponse().addData(dataset);
73     }
74 }
```

Continuamos con el publishService.

```
88     int masterId = super.getRequest().getData("id", int.class);
89     boolean someDraftTrainingSession = this.repository.findManyTrainingSessionsByTrainingModuleIdAndDraftMode(masterId).isEmpty();
90     super.state(someDraftTrainingSession, "", "developer.trainingModule.form.error.trainingSession-draft");
91
92     int id = object.getId();
93     boolean hasPublishedTrainingSession = !this.repository.findTrainingSessionsPublishedByTrainingModuleId(id).isEmpty();
94     super.state(hasPublishedTrainingSession, "", "developer.trainingModule.form.error.no-published-training-session");
95
96     if (!super.getBuffer().getErrors().hasErrors("project"))
97         super.state(!object.getProject().isDraftMode(), "project", "developer.trainingModule.form.error.drafted-project");
98
99     int trainingModuleId = super.getRequest().getData("id", int.class);
100     TrainingModule someTrainingModule = this.repository.findOneTrainingModuleById(trainingModuleId);
101     super.state(someTrainingModule != null, "", "developer.trainingModule.form.error.trainingModule-empty");
102 }
103
104 @Override
105 public void perform(final TrainingModule object) {
106     assert object != null;
107
108     object.setDraftMode(false);
109
110     this.repository.save(object);
111 }
112 }
```

```
114 public void unbind(final TrainingModule object) {
115     assert object != null;
116     SelectChoices choicesProject;
117     Collection<Project> projects;
118
119     SelectChoices choices = SelectChoices.from(DifficultyLevel.class, object.getDifficultyLevel());
120
121     projects = this.repository.findManyProjects();
122     choicesProject = SelectChoices.from(projects, "title", object.getProject());
123
124     Dataset dataset = super.unbind(object, "creationMoment", "details", "code", "difficultyLevel", "updateMoment", "link", "estimatedD
125
126     dataset.put("difficultyLevel", choices.getSelected().getKey());
127     dataset.put("difficultyLevels", choices);
128     dataset.put("project", choicesProject.getSelected().getKey());
129     dataset.put("projects", choicesProject);
130
131     super.getResponse().addData(dataset);
132 }
```

Como se puede comprobar todo en verde, exceptuando las validaciones que se realizan por si se hace postman, ya que por ejemplo en este caso, la entidad está configurada para que te muestre un desplegable con solo los proyectos con el draftmode en false, por lo que solo con postman se podría colocar una ID de un Project con el draftmode en true para que se cumpla la validación.

Seguimos con el ListService.

```
26 @Override
27 public void authorise() {
28     super.getResponse().setAuthorised(true);
29 }
30
31 @Override
32 public void load() {
33     Collection<TrainingModule> objects;
34     int trainingModuleId;
35
36     trainingModuleId = super.getRequest().getPrincipal().getActiveRoleId();
37     objects = this.repository.findManyTrainingModuleByDeveloperId(trainingModuleId);
38
39     super.getBuffer().addData(objects);
40 }
41
42 @Override
43 public void unbind(final TrainingModule object) {
44     assert object != null;
45
46     Dataset dataset;
47
48     dataset = super.unbind(object, "code", "details", "difficultyLevel");
49
50     if (object.isDraftMode()) {
51         Locale local = super.getRequest().getLocale();
52         dataset.put("draftMode", local.equals(Locale.ENGLISH) ? "Yes" : "Si");
53     } else
54         dataset.put("draftMode", "No");
55
56     super.getResponse().addData(dataset);
57 }
```

Está todo en verde, exceptuando la línea que cambia el texto “yes” por un “sí”. Esta línea no se ha podido comprobar en su totalidad con el modo recorder, porque para que funcione la aplicación se tiene que probar en inglés.

También tenemos el deleteService, que está todo en verde.

```
54 @Override
55 public void bind(final TrainingModule object) {
56     assert object != null;
57
58     int projectId;
59     Project project;
60     projectId = super.getRequest().getData("project", int.class);
61     project = this.repository.findOneProjectById(projectId);
62
63     super.bind(object, "creationMoment", "details", "code", "difficultyLevel", "updateMoment", "link", "estimatedTotalTime");
64     object.setProject(project);
65 }
66
67
68 @Override
69 public void validate(final TrainingModule object) {
70     assert object != null;
71
72     int masterId = object.getId();
73     // Obtener las sesiones de entrenamiento en modo borrador asociadas al módulo de entrenamiento
74     boolean hasDraftTrainingSessions = !this.repository.findManyTrainingSessionsByTrainingModuleIdAndDraftMode(masterId).isEmpty();
75
76     // Si no hay sesiones en modo borrador, es seguro eliminar el módulo de entrenamiento
77     super.state(!hasDraftTrainingSessions, "*", "developer.trainingModule.form.error.trainingSession-published");
78 }
79
80 @Override
81 public void unbind(final TrainingModule object) {
```

```
81 public void perform(final TrainingModule object) {
82     assert object != null;
83
84     Collection<TrainingSession> trainingSessions;
85
86     trainingSessions = this.repository.findManyTrainingSessionsByTrainingModuleId(object.getId());
87     this.repository.deleteAll(trainingSessions);
88     this.repository.delete(object);
89 }
90
91 @Override
92 public void unbind(final TrainingModule object) {
93     assert object != null;
94
95     SelectChoices choicesProject;
96     Collection<Project> projects;
97     SelectChoices choices;
98     Dataset dataset;
99
100     choices = SelectChoices.from(DifficultyLevel.class, object.getDifficultyLevel());
101     projects = this.repository.findManyProjects();
102     choicesProject = SelectChoices.from(projects, "title", object.getProject());
103
104     dataset = super.unbind(object, "creationMoment", "details", "code", "updateMoment", "link", "estimatedTotalTime", "draftMode");
105     dataset.put("difficultyLevel", choices.getSelected().getKey());
106     dataset.put("difficultyLevels", choices);
107     dataset.put("project", choicesProject.getSelected().getKey());
108     dataset.put("projects", choicesProject);
109
110     super.getResponse().addData(dataset);
111 }
```

Por último tenemos el createService

```
60 principal = super.getRequest().getPrincipal();
61 developer = this.repository.findOneDeveloperById(principal.getActiveRoleId());
62
63 projectId = super.getRequest().getData("project", int.class);
64 project = this.repository.findOneProjectById(projectId);
65
66 Date moment = MomentHelper.getCurrentMoment();
67 Date creationMoment = new Date(moment.getTime() - 600000); // Le restamos 9 min para asegurar que esta en el pasado
68
69 super.bind(object, "details", "code", "difficultyLevel", "link", "estimatedTotalTime");
70 object.setDeveloper(developer);
71 object.setProject(project);
72 object.setCreationMoment(creationMoment);
73 }
74
75 @Override
76 public void validate(final TrainingModule object) {
77     assert object != null;
78
79     if (!super.getBuffer().getErrors().hasErrors("code")) {
80         TrainingModule existingCode;
81
82         existingCode = this.repository.findTrainingModuleByCode(object.getCode());
83
84         super.state(existingCode == null, "code", "developer.trainingModule.form.error.duplicated-code");
85     }
86
87     if (!super.getBuffer().getErrors().hasErrors("project"))
88         super.state(!object.getProject().isDraftMode(), "project", "developer.trainingModule.form.error.drafted-project");
89
90     if (!super.getBuffer().getErrors().hasErrors("creationMoment")) {
91         Date creationMoment = object.getCreationMoment();
92         Calendar limitCalendar = Calendar.getInstance();
93         limitCalendar.set(1999, Calendar.DECEMBER, 31, 23, 59, 59);
```

```
108 public void unbind(final TrainingModule object) {
109     assert object != null;
110
111     SelectChoices choicesProject;
112     Collection<Project> projects;
113     SelectChoices choices;
114     Dataset dataset;
115
116     choices = SelectChoices.from(DifficultyLevel.class, object.getDifficultyLevel());
117     projects = this.repository.findManyProjects();
118     choicesProject = SelectChoices.from(projects, "title", object.getProject());
119
120     dataset = super.unbind(object, "creationMoment", "details", "code", "updateMoment", "link", "estimatedTotalTime", "draftMode");
121     dataset.put("difficultyLevel", choices.getSelected().getKey());
122     dataset.put("difficultyLevels", choices);
123     dataset.put("project", choicesProject.getSelected().getKey());
124     dataset.put("projects", choicesProject);
125
126     super.getResponse().addData(dataset);
127 }
```

Como podemos comprobar le ocurre lo mismo que al publish y al update, que hay métodos que solo se pueden validar si se hace postman por como está configurado el proyecto.

Como conclusión se puede sacar que todo el “Training Module” ha sido probado de manera muy exhaustiva, probando todas las validaciones posibles. Al final de este capítulo se muestra una imagen con la batería de datos que se ha usado para probar todo el proyecto. Se ha de comentar que no solo se ha hecho una petición, si no que todas las operaciones se han repetido muchas veces, en algunas ocasiones se ha probado entre 30-40 veces como pueden ser los casos del publish, update o créate.

### Training Session

En esta ocasión, también superamos el umbral del 90% sin embargo, se el porcentaje ha disminuido debido a que en el método delete no es posible acceder al unbind.

|  |         |
|--|---------|
| acme.features.developer.trainingsession    | 90,4 %  |
| > DeveloperTrainingSessionUpdateService    | 94,8 %  |
| > DeveloperTrainingSessionShowService.java | 96,4 %  |
| > DeveloperTrainingSessionPublishService   | 94,8 %  |
| > DeveloperTrainingSessionListService.java | 92,2 %  |
| > DeveloperTrainingSessionDeleteService.j  | 60,2 %  |
| > DeveloperTrainingSessionCreateService.j  | 94,3 %  |
| > DeveloperTrainingSessionController.java  | 100,0 % |

Como en el apartado anterior, vamos a comenzar con el updateService.



```

62
63     if (!super.getBuffer().getErrors().hasErrors("code")) {
64         TrainingSession existingCode;
65         int objectId = object.getId();
66
67         existingCode = this.repository.findTrainingSessionByCode(object.getCode());
68
69         boolean isDuplicatedCode = existingCode != null && existingCode.getId() != objectId;
70         super.state(!isDuplicatedCode, "code", "developer.trainingsession.form.error.duplicated");
71     }
72
73     if (!super.getBuffer().getErrors().hasErrors("startTime") && !super.getBuffer().getErrors().hasErrors("endTime")) {
74         Date startTime = MomentHelper.deltaFromMoment(object.getStartTime(), 1, ChronoUnit.WEEKS);
75
76         // Comprobamos que sea una semana
77         super.state(MomentHelper.isAfter(object.getEndTime(), startTime), "endTime", "developer.trainingsession.form.error.end-date-is");
78     }
79
80     if (!super.getBuffer().getErrors().hasErrors("startTime")) {
81         Date startTime = MomentHelper.deltaFromMoment(object.getTrainingModule().getCreationMoment(), 1, ChronoUnit.WEEKS);
82         super.state(MomentHelper.isAfter(object.getStartTime(), startTime), "startTime", "developer.trainingsession.form.error.date-before");
83     }
84
85
86     if (!super.getBuffer().getErrors().hasErrors("endTime")) {
87         boolean endBeforeCreation = MomentHelper.isAfter(object.getEndTime(), object.getTrainingModule().getCreationMoment());
88         super.state(endBeforeCreation, "endTime", "developer.trainingsession.form.error.end-before-creation");
89     }
90
91     int masterId = super.getRequest().getData("id", int.class);
92     TrainingSession trainingSession = this.repository.findOneTrainingSessionById(masterId);
93     boolean noDraftTrainingSession = trainingSession.isDraftMode();
94     super.state(noDraftTrainingSession, "*", "developer.trainingsession.form.error.training-module-no-draft");
95

```

```

97
98     @Override
99     public void perform(final TrainingSession object) {
100         assert object != null;
101         this.repository.save(object);
102     }
103
104     @Override
105     public void unbind(final TrainingSession object) {
106         assert object != null;
107
108         Dataset dataset;
109         dataset = super.unbind(object, "code", "startTime", "endTime", "location", "instructor", "contactEmail", "furtherInformationLink");
110
111         dataset.put("masterId", object.getTrainingModule().getId());
112         super.getResponse().addData(dataset);
113     }
114

```

Podemos observar que está todo en verde y comprobado perfectamente.

Pasamos con el showService que está todo en verde también.

```

40
41     @Override
42     public void load() {
43         TrainingSession object;
44         int developerId;
45
46         developerId = super.getRequest().getData("id", int.class);
47         object = this.repository.findOneTrainingSessionById(developerId);
48
49         super.getBuffer().addData(object);
50     }
51
52     @Override
53     public void unbind(final TrainingSession object) {
54         assert object != null;
55
56         int developerId;
57         Dataset dataset;
58
59         Collection<TrainingModule> trainingModules;
60         SelectChoices choicesTrainingModules;
61
62         developerId = super.getRequest().getPrincipal().getActiveRoleId();
63         trainingModules = this.repository.findManyTrainingModuleByDeveloperId(developerId);
64         choicesTrainingModules = SelectChoices.from(trainingModules, "code", object.getTrainingModule());
65
66         dataset = super.unbind(object, "code", "startTime", "endTime", "location", "instructor", "contactEmail", "furtherInformationLink");
67         dataset.put("trainingModule", choicesTrainingModules.getSelected().getKey());
68         dataset.put("trainingModules", choicesTrainingModules);
69         dataset.put("masterId", object.getTrainingModule().getId());
70
71         super.getResponse().addData(dataset);
72     }
73

```

Seguimos con el publishService, que es exactamente igual que el updateService y que también es igual que el createService, el cual tiene todo en verde..

```
62
63     if (!super.getBuffer().getErrors().hasErrors("code")) {
64         TrainingSession existingCode;
65         int objectId = object.getId();
66         existingCode = this.repository.findTrainingSessionByCode(object.getCode());
67
68         boolean isDuplicatedCode = existingCode != null && existingCode.getId() != objectId;
69         super.state(!isDuplicatedCode, "code", "developer.trainingsession.form.error.duplicated");
70     }
71
72     if (!super.getBuffer().getErrors().hasErrors("startTime") && !super.getBuffer().getErrors().hasErrors("endTime")) {
73         Date startTime = MomentHelper.deltaFromMoment(object.getStartTime(), 1, ChronoUnit.WEEKS);
74         // Comprobamos que sea una semana
75         super.state(MomentHelper.isAfter(object.getEndTime(), startTime), "endTime", "developer.trainingsession.form.error.end-date-ls");
76     }
77
78     if (!super.getBuffer().getErrors().hasErrors("startTime")) {
79         Date startTime = MomentHelper.deltaFromMoment(object.getTrainingModule().getCreationMoment(), 1, ChronoUnit.WEEKS);
80         super.state(MomentHelper.isAfter(object.getStartTime(), startTime), "startTime", "developer.trainingsession.form.error.date-be");
81     }
82
83     if (!super.getBuffer().getErrors().hasErrors("endTime")) {
84         boolean endBeforeCreation = MomentHelper.isAfter(object.getEndTime(), object.getTrainingModule().getCreationMoment());
85         super.state(endBeforeCreation, "endTime", "developer.trainingsession.form.error.end-before-creation");
86     }
87
88     int masterId = super.getRequest().getData("id", int.class);
89     TrainingSession trainingSession = this.repository.findOneTrainingSessionById(masterId);
90     boolean noDraftTrainingSession = trainingSession.isDraftMode();
91     super.state(noDraftTrainingSession, "id", "developer.trainingsession.form.error.training-module-no-draft");
92
93
94
95
```

```
97     @Override
98     public void perform(final TrainingSession object) {
99         assert object != null;
100         this.repository.save(object);
101     }
102
103     @Override
104     public void unbind(final TrainingSession object) {
105         assert object != null;
106         Dataset dataset;
107         dataset = super.unbind(object, "code", "startTime", "endTime", "location", "instructor", "contactEmail", "furtherInformationLi");
108         dataset.put("masterId", object.getTrainingModule().getId());
109         super.getResponse().addData(dataset);
110     }
111
112
113
```

Para el listService, al igual que en “TrainingModule” se puede observar que hay una línea que no se puede probar en su totalidad porque el testing se tiene que realizar en inglés.

```
45     masterId = super.getRequest().getData("masterId", int.class);
46     objects = this.repository.findManyTrainingSessionByTrainingModuleId(masterId);
47     super.getBuffer().addData(objects);
48 }
49
50
51     @Override
52     public void unbind(final TrainingSession object) {
53         assert object != null;
54         Dataset dataset;
55         dataset = super.unbind(object, "code", "location", "contactEmail");
56         if (object.isDraftMode()) {
57             Locale local = super.getRequest().getLocale();
58             dataset.put("draftMode", local.equals(Locale.ENGLISH) ? "Yes" : "Si");
59         } else {
60             dataset.put("draftMode", "No");
61         }
62         super.getResponse().addData(dataset);
63     }
64
65     @Override
66     public void unbind(final Collection<TrainingSession> object) {
67         assert object != null;
68         int masterId;
69         masterId = super.getRequest().getData("masterId", int.class);
70         super.getResponse().addGlobal("masterId", masterId);
71
72
73
74
75
76
```

Por último, tenemos el deleteService, que como se puede observar, nos baja el porcentaje porque nunca llega al unbind.

```
41     masterId = super.getRequest().getData("id", int.class);
42     object = this.repository.findOneTrainingSessionById(masterId);
43
44     super.getBuffer().addData(object);
45 }
46
47 @Override
48 public void bind(final TrainingSession object) {
49     assert object != null;
50
51     super.bind(object, "code", "startTime", "endTime", "location", "instructor", "contactEmail", "furtherInformationLink");
52 }
53
54 @Override
55 public void validate(final TrainingSession object) {
56     assert object != null;
57 }
58
59 @Override
60 public void perform(final TrainingSession object) {
61     assert object != null;
62
63     this.repository.delete(object);
64 }
65
66 @Override
67 public void unbind(final TrainingSession object) {
68     assert object != null;
69
70     Dataset dataset;
71
72     dataset = super.unbind(object, "code", "startTime", "endTime", "location", "instructor", "contactEmail", "furtherInformationLink");
73     dataset.put("masterId", object.getTrainingModule().getId());
74
75     super.getResponse().addData(dataset);
76 }
```

Como conclusión se puede sacar que todo el “Training Session” ha sido probado de manera muy exhaustiva, probando todas las validaciones posibles. Al final de este capítulo se muestra una imagen con la batería de datos que se ha usado para probar todo el proyecto. Se ha de comentar que no solo se ha hecho una petición, si no que todas las operaciones se han repetido muchas veces, en algunas ocasiones se ha probado entre 30-40 veces como pueden ser los casos del publish, update o create.

Para terminar este capítulo me gustaría mostrar todos los datos que se han usado para las pruebas. He usado el Excel que se ha añadido en la actualización del framework para tomar de ahí los datos, y lo he pegado en un bloc de notas para tenerlo a mano a la hora de realizar las pruebas.

[illegible]

```
61 -----
62 -----
63 Training Session: 2022/12/19 13:21 - 2022/12/30 13:21
64
65 vacío
66
67 CODE
68
69 Vacío
70 Espacio en blanco
71 Sin el formato
72 Caracteres: 국민경제의 발전
73 TS-GAS-253
74 TS-TTT-001
75 hacking: <h1>!</h1> // '' or 'A'='A
76
77 STARTTIME TM - 2021/12/12 13:20
78
79 vacío
80 espacio
81 Formato mal 11/11/2002 11:11
82 Pasado 2020/12/12 13:20
83 El mismo 2021/12/12 13:20
84 Un min despues 2021/12/12 13:21
85 Una semana despues 2021/12/19 13:20
86 Una semana y un min despues 2021/12/19 13:21
87
88 ENDTIME
89 vacío
90 espacio
91 Formato mal 11/11/2002 11:11
92 Pasado 2020/12/12 13:20
93 Un min despues 2021/12/12 13:21
94 Una semana y un min despues 2021/12/19 13:21
95 Dos semanas despues 2021/12/26 13:21
96 Dos semanas despues y un min 2021/12/26 13:22
97
98 LOCATION
99
00 Vacío
01 Espacio en blanco
02 1 caracter
03 75 Caracteres: Lorem ipsum dolor sit ametE consectetur adipiscing elite sed do eiusmod tem
04 74 caracteres: Lorem ipsum dolor sit ametE consectetur adipiscing elite sed do eiusmod te
05 76 Caracteres: Lorem ipsum dolor sit amete consectetur adipiscing elite sed do eiusmod temp
06 Caracteres raros: 국민경제의 발전을
07 hacking: <h1>!</h1>
08
09
10 INSTRUCTOR 2022/12/19 13:21 - 2022/12/30 13:21
11 Vacío
12 Espacio en blanco
13 1 caracter
14 75 Caracteres: Lorem ipsum dolor sit amete consectetur adipiscing elite sed do eiusmod tem
15 74 caracteres: Lorem ipsum dolor sit amete consectetur adipiscing elite sed do eiusmod te
16 76 Caracteres: Lorem ipsum dolor sit amete consectetur adipiscing elite sed do eiusmod temp
17 Caracteres raros: 국민경제의 발전을
18 hacking: <h1>!</h1>
19
```

```
EMAIL

nulo
espacio
1 caracter
esto es un email de pruebaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa@emaildeprueba.com
emaildeprueba@a.com

LINK

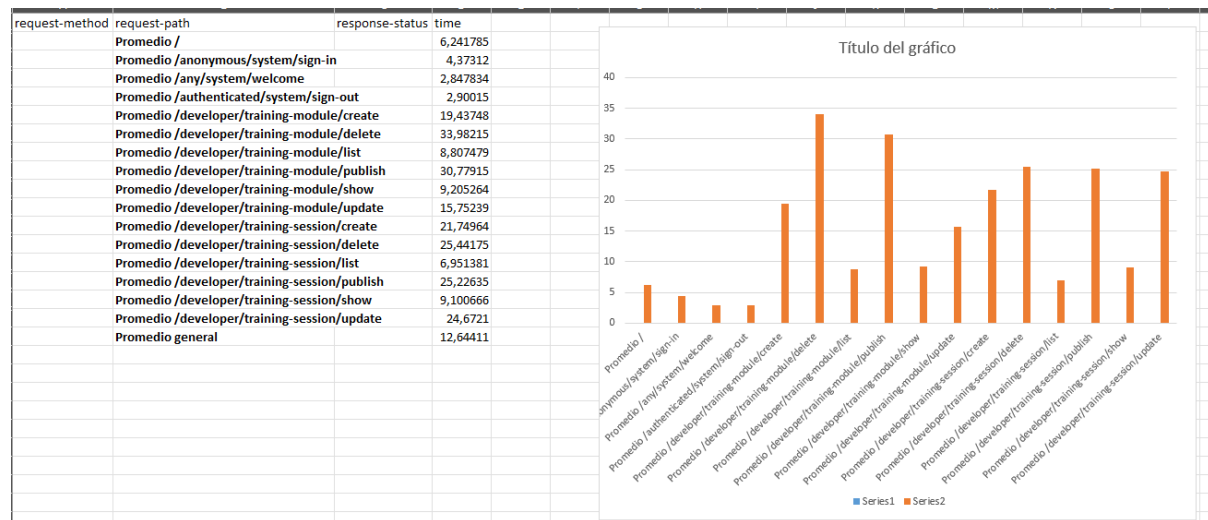
Nulo
Blanco
Pocos caracteres: ftp://
Uno por debajo: http://www.lorem-ipsum.org/dolor/sit/amete/consectetur/adipiscing/elite/sed/do/eiusmod/temp
Justo: http://www.lorem-ipsum.org/dolor/sit/amet,/consectetur/adipiscing/elit,/sed/do/eiusmod/tempor/incidi
Uno por encima: http://www.lorem-ipsum.org/dolor/sit/amet,/consectetur/adipiscing/elit,/sed/do/eiusmod/temp

Lorem ipsum dolor sit ametE consectetur - ipiscing,elite sed do eiusmod tem
```

## Capítulo 2 – Pruebas de desempeño

El desarrollo del software se ha ejecutado durante todo el cuatrimestre en el PC1. Obteniendo los resultados de ejecutar el replayer en eclipse, nos genera una batería de datos, los cuales, analizándolos mediante las técnicas enseñadas en clase, hemos podido obtener resultados claros.

Vamos a empezar por los promedios de los resultados de búsqueda.



Como se puede observar, ningún promedio supera los 50 ms, lo que es un resultado muy bueno, porque nos indica que las búsquedas se realizan de manera rápida. En este apartado tengo que comentar que la primera vez que realicé el test, observé que había un par de métodos que llegaban a los 50 ms, así que gracias a eso pude darme cuenta de que el código que había implementado no estaba refactorizado correctamente, porque me traía colecciones de datos enteras, y luego recorría todos esos datos para obtener los resultados, lo que en una base de datos mucho más grande provocaría mucha pérdida de tiempo, así que opté por solucionar el código y poner todo lo necesario directamente en las queries.

A continuación, vamos a observar el intervalo de confianza para **el PC 1** tanto antes de introducir índices, como después de meter índices.

| Before                    |            |            |  | After                     |            |            |
|---------------------------|------------|------------|--|---------------------------|------------|------------|
| Media                     | 12,6441063 |            |  | Media                     | 12,4331975 |            |
| Error típico              | 0,38597407 |            |  | Error típico              | 0,38456645 |            |
| Mediana                   | 8,2804     |            |  | Mediana                   | 8,0893     |            |
| Moda                      | 8,2325     |            |  | Moda                      | 7,7308     |            |
| Desviación estándar       | 12,0890627 |            |  | Desviación estándar       | 11,6961319 |            |
| Varianza de la muestra    | 146,145437 |            |  | Varianza de la muestra    | 136,799501 |            |
| Curtosis                  | 61,492175  |            |  | Curtosis                  | 34,7543068 |            |
| Coefficiente de asimetría | 5,41108282 |            |  | Coefficiente de asimetría | 4,00865021 |            |
| Rango                     | 185,0061   |            |  | Rango                     | 155,4288   |            |
| Mínimo                    | 1,5008     |            |  | Mínimo                    | 1,661      |            |
| Máximo                    | 186,5069   |            |  | Máximo                    | 157,0898   |            |
| Suma                      | 12103,8683 |            |  | Suma                      | 20957,7077 |            |
| Cuenta                    | 941        |            |  | Cuenta                    | 941        |            |
| Nivel de confianza(95,0%) | 0,75743073 |            |  | Nivel de confianza(95,0%) | 0,75472499 |            |
| Interval (ms)             | 11,8866756 | 13,401537  |  | Interval (ms)             | 11,6784725 | 13,1879225 |
| Interval (s)              | 0,01188668 | 0,01340154 |  | Interval (s)              | 0,01167847 | 0,01318792 |

Con estos datos se ha realizado un Z-Test, el cual se muestra a continuación.

| Prueba z para medias de dos muestras |               |              |
|--------------------------------------|---------------|--------------|
|                                      | <i>Before</i> | <i>after</i> |
| Media                                | 12,64410632   | 12,4331975   |
| Varianza (conocida)                  | 146,145437    | 136,799501   |
| Observaciones                        | 941           | 941          |
| Diferencia hipotética de las medias  | 0             |              |
| z                                    | 0,387091388   |              |
| P(Z<=z) una cola                     | 0,349344277   |              |
| Valor crítico de z (una cola)        | 1,644853627   |              |
| Valor crítico de z (dos colas)       | 0,698688553   |              |
| Valor crítico de z (dos colas)       | 1,959963985   |              |

Podemos observar que Alpha es 0.05, y que el p-value es 0.698... por lo que podemos decir que los cambios **no** dieron como resultado ninguna mejora significativa; los tiempos de muestreo son diferentes, pero son globalmente iguales.

**También se ha replicado estas pruebas en otro ordenador** (PC2 – características similares) y he obtenido los siguientes resultados:

| Before PC2                |            |            |  | After PC2                 |            |            |  |
|---------------------------|------------|------------|--|---------------------------|------------|------------|--|
| Media                     | 12,7755109 |            |  | Media                     | 12,5370975 |            |  |
| Error típico              | 0,39163019 |            |  | Error típico              | 0,38735775 |            |  |
| Mediana                   | 8,3936     |            |  | Mediana                   | 8,1438     |            |  |
| Moda                      | 9,6781     |            |  | Moda                      | 1,9404     |            |  |
| Desviación estándar       | 12,2662177 |            |  | Desviación estándar       | 11,7810262 |            |  |
| Varianza de la muestra    | 150,460096 |            |  | Varianza de la muestra    | 138,792577 |            |  |
| Curtosis                  | 63,5362241 |            |  | Curtosis                  | 34,237388  |            |  |
| Coefficiente de asimetría | 5,50562157 |            |  | Coefficiente de asimetría | 3,97976665 |            |  |
| Rango                     | 190,5563   |            |  | Rango                     | 155,4288   |            |  |
| Mínimo                    | 1,5458     |            |  | Mínimo                    | 1,661      |            |  |
| Máximo                    | 192,1021   |            |  | Máximo                    | 157,0898   |            |  |
| Suma                      | 12532,7762 |            |  | Suma                      | 12296,8152 |            |  |
| Cuenta                    | 935        |            |  | Cuenta                    | 935        |            |  |
| Nivel de confianza(95,0%) | 0,76853023 |            |  | Nivel de confianza(95,0%) | 0,76020303 |            |  |
| Interval (ms)             | 12,0069807 | 13,5440411 |  | Interval (ms)             | 11,7768945 | 13,2973005 |  |
| Interval (s)              | 0,01200698 | 0,01354404 |  | Interval (s)              | 0,01177689 | 0,0132973  |  |

Con estos resultados se ha hecho de nuevo un **Z-Test (PC 2)** para analizar correctamente los datos, y he obtenido:

| Prueba z para medias de dos muestras |            |            |
|--------------------------------------|------------|------------|
|                                      |            |            |
|                                      | 192,1021   | 122,8571   |
| Media                                | 12,5925246 | 12,4177036 |
| Varianza (conocida)                  | 150,460096 | 138,792577 |
| Observaciones                        | 930        | 930        |
| Diferencia hipotética de las medias  | 0          |            |
| z                                    | 0,31720738 |            |
| P(Z<=z) una cola                     | 0,37554313 |            |
| Valor crítico de z (una cola)        | 1,64485363 |            |
| Valor crítico de z (dos colas)       | 0,75108625 |            |
| Valor crítico de z (dos colas)       | 1,95996398 |            |

Podemos observar que Alpha es 0.05, y que el p-value es 0.751... por lo que podemos decir que los cambios no dieron como resultado ninguna mejora significativa; los tiempos de muestreo son diferentes, pero son globalmente iguales.

Como **conclusión**, ninguno de los PCs muestra una diferencia significativa en el rendimiento (antes y después) a un nivel de variación del 5%. Por lo tanto, las diferencias observadas en las medias no son estadísticamente significativas, lo que sugiere que ninguno de los PCs es concluyentemente más rápido o lento que el otro según los datos proporcionados.

## Bibliografía

Diapositivas de Diseño y Pruebas 2 – Universidad de Sevilla.