



Facultad de Ingeniería

Bernard Wand Polak

Obligatorio 1

Diseño de aplicaciones 2

Lucas Castro - N° 218709

Ricardo Poladura - N° 238052

Grupo: N5A

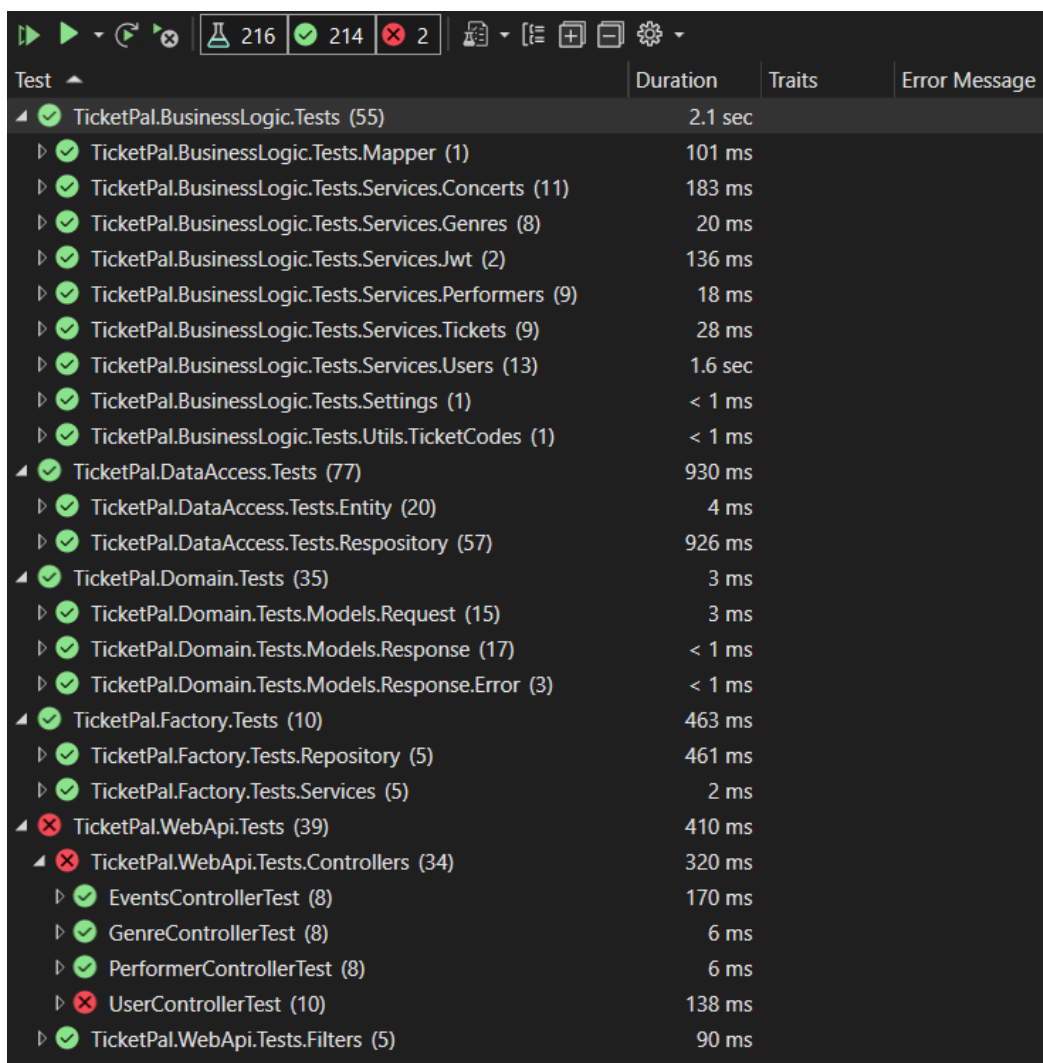
Docentes: Ignacio Valle – Nicolás Fierro – Nicolás Blanco

EVIDENCIA DE LA APLICACIÓN DE TDD Y CLEAN CODE

La estrategia utilizada para aplicar TDD en nuestro obligatorio fue la de outside-in, que consiste en escribir un test, con lo mínimo necesario, que falle, para luego escribir el código necesario para que el test pase, y luego refactorizar, volviendo a escribir otro test que falle, y así sucesivamente (“Red-Green-Refactor”). Y se van realizando las iteraciones necesarias de este bucle interno hasta conseguir pasar el test de aceptación.

Por regla general, casi toda la implementación del obligatorio se hizo aplicando TDD. Por un tema de tiempo, solamente la implementación de algunos Requests y Responses, se hizo sin aplicar TDD.

Como tuvimos algunos problemas con la implementación de algunos controladores, nos quedaron dos tests que no pasaron, de un total de 216 tests.



The screenshot shows a test runner interface with a toolbar at the top containing icons for running tests, a summary bar showing 216 passed, 214 successful, and 2 failed tests, and a table of test results. The table has columns for Test, Duration, Traits, and Error Message. The tests are organized into a tree structure, with some tests failing, indicated by a red 'X' icon.

Test	Duration	Traits	Error Message
TicketPal.BusinessLogic.Tests (55)	2.1 sec		
TicketPal.BusinessLogic.Tests.Mapper (1)	101 ms		
TicketPal.BusinessLogic.Tests.Services.Concerts (11)	183 ms		
TicketPal.BusinessLogic.Tests.Services.Genres (8)	20 ms		
TicketPal.BusinessLogic.Tests.Services.Jwt (2)	136 ms		
TicketPal.BusinessLogic.Tests.Services.Performers (9)	18 ms		
TicketPal.BusinessLogic.Tests.Services.Tickets (9)	28 ms		
TicketPal.BusinessLogic.Tests.Services.Users (13)	1.6 sec		
TicketPal.BusinessLogic.Tests.Settings (1)	< 1 ms		
TicketPal.BusinessLogic.Tests.Utils.TicketCodes (1)	< 1 ms		
TicketPal.DataAccess.Tests (77)	930 ms		
TicketPal.DataAccess.Tests.Entity (20)	4 ms		
TicketPal.DataAccess.Tests.Respository (57)	926 ms		
TicketPal.Domain.Tests (35)	3 ms		
TicketPal.Domain.Tests.Models.Request (15)	3 ms		
TicketPal.Domain.Tests.Models.Response (17)	< 1 ms		
TicketPal.Domain.Tests.Models.Response.Error (3)	< 1 ms		
TicketPal.Factory.Tests (10)	463 ms		
TicketPal.Factory.Tests.Repository (5)	461 ms		
TicketPal.Factory.Tests.Services (5)	2 ms		
TicketPal.WebApi.Tests (39)	410 ms		
TicketPal.WebApi.Tests.Controllers (34)	320 ms		
EventsControllerTest (8)	170 ms		
GenreControllerTest (8)	6 ms		
PerformerControllerTest (8)	6 ms		
UserControllerTest (10)	138 ms		
TicketPal.WebApi.Tests.Filters (5)	90 ms		

Con respecto a la cobertura de código, el resultado del reporte fue el siguiente:

Code Coverage Results				
lucas_DESKTOP-8Q3GD9N 2022-05-09 22_47				
Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
lucas_DESKTOP-8Q3GD9N 2022-...	1052	12.82%	7151	87.18%
ticketpal.businesslogic.dll	78	12.04%	570	87.96%
ticketpal.businesslogic.tests.dll	1	0.04%	2319	99.96%
ticketpal.dataaccess.dll	655	61.73%	406	38.27%
ticketpal.dataaccess.tests.dll	17	1.16%	1453	98.84%
ticketpal.domain.dll	39	14.89%	223	85.11%
ticketpal.domain.tests.dll	0	0.00%	154	100.00%
ticketpal.factory.dll	0	0.00%	53	100.00%
ticketpal.factory.tests.dll	16	11.27%	126	88.73%
ticketpal.webapi.dll	203	39.73%	308	60.27%
ticketpal.webapi.tests.dll	43	2.72%	1539	97.28%

El nivel de cobertura llegó a un total de 87,18%, debido a que por falta de tiempo, no hicimos tests de todas las clases de Request y Response, y porque quedaron algunas clases sueltas que no ameritaban tests, como por ejemplo la de los enumerados, que se contaron entre los resultados.

Pruebas de aplicación de TDD y Clean Code

Ejemplos de aplicación de TDD y Clean code en la implementación del obligatorio:

Commit 570b1ee (Primero se terminan de implementar los tests de UserServiceTests):

```

18     [TestClass]
19     public class UserServiceTest : BaseServiceTest
20     {
21         -         private string jwtTestSecret;
22         -         private string userPassword;
23         -
24         [TestMethod]
25         public void UserAuthenticateCorrectly()
26         {
27             -         int id = 1;
28             -         this.userPassword = "somePassword";
29             -         this.jwtTestSecret = "23jrb783v29fwfvfg2874gf286fce8";
30             -         var testAppsettings = Options.Create(new AppSettings { JwtSecret = jwtTestSecret });
31             -
32             +         int id = 1;
33             var authRequest = new AuthenticationRequest
34             {
35                 Email = "someone@example.com",
36             @@ -50,7 +43,7 @@ public void UserAuthenticateCorrectly()
37
38                 this.userService = new UserService(
39                     this.usersMock.Object,
40                     -         testAppsettings,
41                     +         this.testAppSettings,
42                     this.mapper
43                 );
44             User authenticatedUser = userService.Login(authRequest);

```

Commit e22c195 (Se implementa la clase UserService):

```
18 +     public class UserService : IUserService
19 +     {
20 +         private readonly IUserRepository repository;
21 +         private readonly IAppSettings appSettings;
22 +         private readonly IMapper mapper;
23 +         public UserService(
24 +             IUserRepository repository,
25 +             IOptions<IAppSettings> appSettings,
26 +             IMapper mapper
27 +         )
28 +         {
29 +             this.mapper = mapper;
30 +             this.repository = repository;
31 +             this.appSettings = appSettings.Value;
32 +         }
33 +         public OperationResult DeleteUser(int id)
34 +         {
35 +             try
36 +             {
37 +                 repository.Delete(id);
38 +                 return new OperationResult
39 +                 {
40 +                     ResultCode = ResultCode.SUCCESS,
41 +                     Message = "User removed successfully"
42 +                 };
43 +             }
44 +             catch (RepositoryException ex)
45 +             {
46 +                 return new OperationResult
47 +                 {
48 +                     ResultCode = ResultCode.FAIL,
49 +                     Message = ex.Message
```

```
49 +         Message = ex.Message
50 +     };
51 + }
52 + }
53 +
54 + public User GetUser(int id)
55 + {
56 +     return mapper.Map<User>(repository.Get(id));
57 + }
58 +
59 + public IEnumerable<User> GetUsers(UserRole role = UserRole.SPECTATOR)
60 + {
61 +     var users = repository.GetAll(u => u.Role.Equals(role.ToString()));
62 +     return mapper.Map<IEnumerable<UserEntity>, IEnumerable<User>>(users);
63 + }
64 +
65 + public User Login(AuthenticationRequest model)
66 + {
67 +     var found = repository.Get(u => u.Email.Equals(model.Email));
68 +     if (found == null || !BC.Verify(model.Password, found.Password))
69 +     {
70 +         return null;
71 +     }
72 +     var token = JwtUtils.GenerateJwtToken(appSettings.JwtSecret, "id", found.Id.ToString());
73 +     var user = mapper.Map<User>(found);
74 +
75 +     return user;
76 + }
```

Commit 0a22225 (Se crearon las pruebas de la clase TicketServiceTests):

Clase TicketServiceTests:

```
53 + [TestMethod]
54 + public void AddTicketSucesfullyTest()
55 + {
56 +     OperationResult result = ticketService.AddTicket(ticketRequest);
57 +
58 +     Assert.IsTrue(result.ResultCode == ResultCode.SUCCESS);
59 + }
60 +
61 + [TestMethod]
62 + public void AddTicketTwiceFailsTest()
63 + {
64 +     ticketService.AddTicket(ticketRequest);
65 +
66 +     this.ticketsMock.Setup(m => m.Exists(It.IsAny<int>())).Returns(true);
67 +     this.ticketsMock.Setup(m => m.Add(It.IsAny<TicketEntity>())).Throws(new RepositoryException());
68 +     ticketService = new TicketService(this.ticketsMock.Object, this.testAppSettings, this.mapper);
69 +
70 +     OperationResult result = ticketService.AddTicket(ticketRequest);
71 +
72 +     Assert.IsTrue(result.ResultCode == ResultCode.FAIL);
73 + }
```

```
107 + [TestMethod]
108 + public void UpdateTicketSucesfullyTest()
109 + {
110 +     var updateRequest = new UpdateTicketRequest
111 +     {
112 +         Code = ticket.Code,
113 +         Status = ticket.Status
114 +     };
115 +
116 +     this.ticketsMock.Setup(m => m.Update(It.IsAny<TicketEntity>())).Verifiable();
117 +     OperationResult expected = ticketService.UpdateTicket(updateRequest);
118 +
119 +     Assert.IsTrue(expected.ResultCode == ResultCode.SUCCESS);
120 + }
```

Class TicketService:

```
29 +         public OperationResult AddTicket(AddTicketRequest model)
30 +         {
31 +             throw new NotImplementedException();
32 +         }
33 +
34 +         public OperationResult DeleteTicket(int id)
35 +         {
36 +             throw new NotImplementedException();
37 +         }
38 +
39 +         public bool ExistsTicketByName(string name)
40 +         {
41 +             throw new NotImplementedException();
42 +         }
43 +
44 +         public Ticket GetTicket(int id)
45 +         {
46 +             throw new NotImplementedException();
47 +         }
48 +
49 +         public IEnumerable<Ticket> GetTickets()
50 +         {
51 +             throw new NotImplementedException();
52 +         }
53 +
54 +         public OperationResult UpdateTicket(UpdateTicketRequest model)
55 +         {
56 +             throw new NotImplementedException();
57 +         }
58 +     }
```


Commit 8fa6670 (Se implementa la clase TicketService):

```
19     public class TicketService : ITicketService
20     {
21         - private readonly ITicketRepository repository;
22         - private readonly IAppSettings appSettings;
23         + private readonly IServiceFactory serviceFactory;
24         private readonly IMapper mapper;
25         + public IGenericRepository<TicketEntity> ticketRepository;
26         + public IGenericRepository<ConcertEntity> concertRepository;
27
28         - public TicketService(ITicketRepository repository, IOptions<IAppSettings> appSettings, IMapper mapper)
29         + public TicketService(IServiceFactory factory, IMapper mapper)
30         {
31             this.mapper = mapper;
32             - this.repository = repository;
33             - this.appSettings = appSettings.Value;
34             + this.serviceFactory = factory;
35             + ticketRepository = serviceFactory.GetRepository<TicketEntity>() as IGenericRepository<TicketEntity>;
36             + concertRepository = serviceFactory.GetRepository<ConcertEntity>() as IGenericRepository<ConcertEntity>;
37         }
38
39         public OperationResult AddTicket(AddTicketRequest model)
40         {
41             - throw new NotImplementedException();
42             - }
43             + try
44             + {
45                 EventEntity newEvent = concertRepository.Get(model.Event);
46
47         public OperationResult DeleteTicket(int id)
48         {
49             - throw new NotImplementedException();
```

Commit 14563ee (Se crean tests para utilización de ServiceFactory):

```
8 + namespace TicketPal.Factory.Tests.Repository
9 + {
10 +     [TestClass]
11 +     public class RepositoryDependenciesTest :BaseTestFactoryConfig
12 +     {
13 +         private ServiceFactory factory;
14 +         private ServiceProvider serviceProvider;
15 +
16 +         [TestInitialize]
17 +         public void Init()
18 +         {
19 +             // Factory
20 +             this.factory = new ServiceFactory(this.services);
21 +
22 +             this.factory.AddDbContextService("SomeFakeConnectionString");
23 +             this.factory.RegisterRepositories();
24 +
25 +             this.serviceProvider = services.BuildServiceProvider();
26 +         }
27 +
28 +         [TestMethod]
29 +         public void UserRepositoryDependencyCheck()
30 +         {
31 +             var repository = serviceProvider.GetService<IGenericRepository<UserEntity>>();
32 +             Assert.IsNotNull(repository);
33 +         }
34 +     }
35 + } ⊖
```

Commit 129884f (Se crea clase ServiceFactory):

```
8 + namespace TicketPal.Factory
9 + {
10 +     public class ServiceFactory
11 +     {
12 +         private readonly IServiceCollection services;
13 +
14 +         public ServiceFactory(IServiceCollection services)
15 +         {
16 +             this.services = services;
17 +         }
18 +
19 +         public void AddDbContextService(string connectionString)
20 +         {
21 +             services.AddDbContext<DbContext, AppDbContext>
22 +                 (options => options.UseSqlServer(connectionString));
23 +         }
24 +
25 +         public void RegisterRepositories()
26 +         {
27 +             services.AddScoped(typeof(IGenericRepository<UserEntity>), typeof(UserRepository));
28 +         }
29 +     }
30 + } ⊖
```