Tests d'intégration

Integration testing are usefull for validating user stories automatically. It is usefull for this test which user all the layers of the app including the database connection, to use a InMemory database.

Create Custom `WebApplicationFactory`

```
public class CustomWebApplicationFactory : WebApplicationFactory<Startup>
    {
        public ApplicationDbContext Context { get; private set; }

        protected override void ConfigureWebHost(IWebHostBuilder builder)
        {
            builder.ConfigureServices(services =>
            {
                // Remove or app real DbProvider from the DI container
                var descriptor = services.SingleOrDefault(
                    d => d.ServiceType ==
                         typeof(DbContextOptions<ApplicationDbContext>));
                services.Remove(descriptor);

                // Create a new service provider for the InMemory Database
                var serviceProvider = new ServiceCollection()
                    .AddEntityFrameworkInMemoryDatabase()
                    .BuildServiceProvider();

                // Add a database context (AppDbContext) using an in-memory database for
testing.
                services.AddDbContext<ApplicationDbContext>(options =>
                {
                    options.UseInMemoryDatabase("InMemoryAppDb");
                    options.UseInternalServiceProvider(serviceProvider);
```

```
                });

                // Build the service provider
                var sp = services.BuildServiceProvider();

                // Create a scope to obtain a reference to the database contexts (for
verification in testing)
                var scope = sp.CreateScope();
                var scopedServices = scope.ServiceProvider;
                var appDb = scopedServices.GetRequiredService<ApplicationDbContext>();

                var logger =
scopedServices.GetRequiredService<ILogger<CustomWebApplicationFactory>>();

                // Ensure the database is created (ensure migrations are executed)
                appDb.Database.EnsureCreated();
                Context = appDb;
            });
        }
    }
```

Use the new `WebApplicationFactory` in a test

```
 public partial class IntegrationTests : IClassFixture<CustomWebApplicationFactory>
    {
        private readonly HttpClient client;
        private readonly ApplicationDbContext db;

        public IntegrationTests(CustomWebApplicationFactory factory) // Constructor is
executed between each test method
        {
            this.client = factory.CreateClient();
            this.db = factory.Context;
            factory.Context.Database.EnsureDeleted(); // Ensure the database is emptied
between each test method
        }
```

```
        [Fact]
        public async void Register()
        {
        // Given
            var registerDto = new RegisterDTO()
            {
                Email = "john.shepard@n7.citadel",
                Username = "Shepard",
                Password = "NormandyTali<3"
            };

            // When
            var registerResponse = await client.PostAsJsonAsync("api/auth/register",
registerDto);

            // Then
            var result = await registerResponse.Content.ReadAsStringAsync();
            Assert.True(registerResponse.IsSuccessStatusCode);
            Assert.Equal(1,db.Set<User>().Count());
        }

    }
```

Created 2024-11-21 15:03:42 UTC by Nicolas
Updated 2024-11-21 15:04:11 UTC by Nicolas