Pengembangan Unit Testing dan Integration Testing REST API Pengelola Data Bootcamp PT Mitra Integrasi Informatika

Rikky Setiawan[#1], Risal[*2]

[#]*Program Studi Sistem Informasi,Universitas Kristen Maranatha*
Jalan Surya Sumantri no 65, Bandung, Indonesia
[1]2073019@maranatha.ac.id
[2]laurentius.risal@it.maranatha.edu

Abstract — **Mitra Integrasi Informatika LLC is a company focused on Information, Communication, and Technology business. One of its main business groups is Metrodata Academy which focuses primarily on training and certification; bootcamp being one of the programs. In order to help manage the data related to boot camp programs, a web app was made with React (front-end) and ASP.NET Core (back-end). The app is good enough to support the business process, but it has some a problem related to its development process, where there is a lack of testing in its development. A solution was implemented to solve this problem which is the addition of unit testing and integration testing.**

Keywords— **API, ASP.NET Core, C#, integration test, .NET, unit test.**

I. PENDAHULUAN

PT Mitra Integrasi Informatika adalah perusahaan yang bergerak di bidang ICT (*Information and Communication Technology*). Perusahaan ini memiliki berbagai bidang bisnis, antara lain seperti *cloud services*, *hybrid IT infrastructure*, *managed services*, dan lainnya. Selain pengembangan aplikasi, PT. Mitra Integrasi Informatika juga menyediakan layanan pelatihan dan sertifikasi profesional yang bernaung dengan nama Metrodata Academy.

Operasional dari Metrodata Academy sendiri didukung oleh aplikasi-aplikasi internal yang dapat memudahkan proses bisnisnya, salah satunya adalah aplikasi untuk melakukan manajemen terhadap data peserta program *bootcamp* yang disediakan oleh perusahaan tersebut. Aplikasi ini sudah sebagian besar berjalan dan bisa digunakan dalam proses bisnis sehari-hari, namun masih terdapat kebutuhan yang dapat dipenuhi dari segi pengembangan aplikasi.

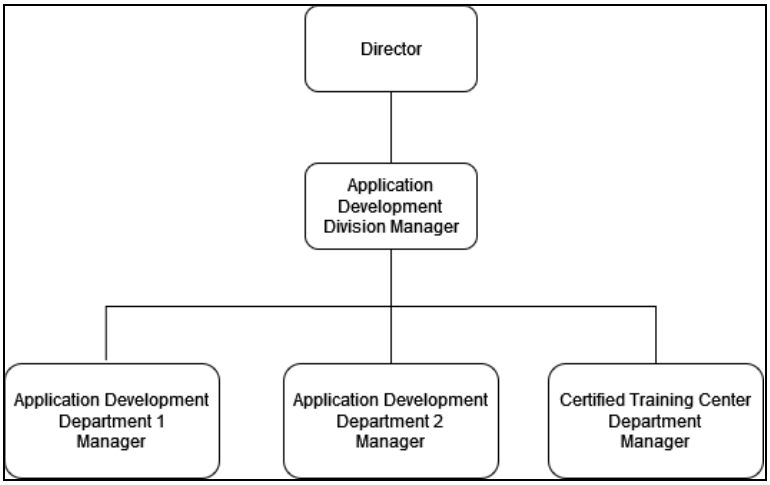
Masalah yang dihadapi dalam bidang pengembangan aplikasi ini adalah tidak adanya mekanisme pengujian yang dapat memverifikasi kebenaran alur program atau bagaimana cara data diproses, sehingga dapat memperbesar kemungkinan terjadinya kesalahan ketika ada usaha pengembangan di masa depan. Sesuai dengan kebutuhan yang ada, maka akan dikembangan beberapa solusi untuk menyelesaikan permasalahan-permasalahan tersebut.

Berdasarkan penjelasan latar belakang tersebut, penelitian ini bertujuan untuk mengimplementasikan teknik pengujian perangkat lunak berupa *unit test* dan *integration test* pada aplikasi internal yang digunakan oleh PT Mitra Integrasi Informatika. Penelitian ini diharapkan agar dapat memberikan kontribusi terhadap proses pengembangan aplikasi yang dilakukan oleh PT Mitra Integrasi Informatika.

II. INSTANSI DAN DESKRIPSI PEKERJAAN

A. Profil Perusahaan

PT Mitra Integrasi Informatika (MII) merupakan anak perusahaan dari PT Metrodata Electronics, Tbk yang didirikan pada tanggal 1 Maret 1996 [1]. PT MII sendiri bergerak di bidang *ICT* (*Information, Communication, and Technology*) *solutions*. Bisnis utamanya meliputi delapan bagian, yaitu *cloud services*, *business application*, *digital platform business*, *Big Data & analytics*, *security*, *hybrid IT infrastructure*, *consulting & advisory*, dan *managed services*. Bidang *managed services* pada PT. MII terbagi menjadi dua jenis, yaitu keamanan dan infrastruktur, serta pengembangan aplikasi. Fokus pada kegiatan magang sendiri terdapat pada bagian pengembangan aplikasi.



Gambar 1. Struktur Organisasi

Gambar 1 menunjukkan struktur organisasi dari PT. Mitra Integrasi Informatika yang berhubungan dengan kegiatan magang yang dilakukan. Dari direktur, terdapat *Application Development Division Manager* yang memiliki tiga bawahan, yaitu *Application Development Department 1* (ADD 1) *Manager*, *Application Development Department 2* (ADD 2) *Manager*, dan *Certified Training Center* (CTC) *Department Manager*.

ADD 1 merupakan departemen yang menangani hal-hal yang berhubungan dengan akademik dan pengembangan talenta, termasuk pelaksanaan program magang. ADD 2 merupakan departemen yang berhubungan dengan manajemen proyek dengan konsumen, sedangkan CTC merupakan departemen yang menangani program training dan sertifikasi profesional, termasuk sebagai salah satu pelaksana program Studi Independen Kampus Merdeka.

Saat menjalani program magang, posisi yang ditempati adalah Application Developer yang termasuk dalam department ADD 1. Posisi ini memiliki tugas utama untuk ikut terlibat dalam pengembangan proyek-proyek tertentu perusahaan, baik internal maupun eksternal, sesuai dengan kompetensi yang ada.

B. Deskripsi Pekerjaan dan Tanggung Jawab

Selama kegiatan magang berlangsung, posisi yang diterima adalah sebagai Application Developer dengan tugas untuk mengembangkan aplikasi internal perusahaan. Tugas spesifik yang diberikan adalah melanjutkan pengembangan REST API (*Application Programming Interface*) pengelola data bootcamp PT Mitra Integrasi Informatika, serta menguji hasil pekerjaan dan memastikan alurnya berjalan seperti yang diinginkan. Sedangkan tanggung jawab yang diberikan antara lain seperti menjalin kerjasama dan komunikasi yang baik dengan mentor dan anggota tim, menerima saran dan arahan dari mentor dan anggota tim, hadir dalam pertemuan-pertemuan yang telah ditentukan, dan mengkonsultasikan hasil pekerjaan dan masalah yang dihadapi dengan mentor.

C. Timeline Proyek

TABEL I
TIMELINE PROYEK

	Februari 2024	Maret 2024	April 2024	Mei 2024	Juni 2024
Onboarding					
Pelatihan					
Pengembangan					
Penilaian Pertengahan Program					
Penilaian Akhir					

Tabel 1 menunjukkan timeline proyek yang dikerjakan. Pada bulan Februari 2024, akan dilaksanakan program onboarding atau pengenalan perusahaan terhadap pemagang. Setelahnya, dilaksanakan program pelatihan sesuai dengan

kompetensi dan posisi pemagang. Kegiatan pengembangan aplikasi akan dilakukan mulai awal Maret 2024 sampai Juni 2024. Pada bulan April 2024, akan dilaksanakan penilaian pertengahan program, dan pada Juni 2024, juga akan dilaksanakan penilaian akhir program.

III. LANDASAN TEORI

A. REST API

Application Programming Interface (API) adalah perantara antara satu perangkat lunak dengan perangkat lunak lainnya. API biasanya akan mendefinisikan bagaimana tiap perangkat lunak tersebut dapat berkomunikasi satu sama lain menggunakan pola *request* dan *response*. Ada berbagai jenis API yang umum digunakan, antara lain seperti SOAP (*Simple Object Access Protocol*), RPC (*Remote Procedure Call*), dan REST (*Representational State Transfer*). Dalam proyek aplikasi manajemen bootcamp ini, REST API digunakan sebagai perantara antara aplikasi web dengan data yang ada.

Representational State Transfer atau REST menjamin adanya interoperabilitas antara *client* dan *server*, dimana dua entitas akan berkomunikasi melalui *request* HTTP dan server akan merespon dalam bentuk JSON (JavaScript Object Notation) [2]. Dalam penerapannya, REST memiliki beberapa prinsip [3], yaitu:

1. *Interface* yang seragam, dimana tiap permintaan atau request terhadap sebuah sumber daya (data) seharusnya memiliki bentuk yang sama dimanapun *request* tersebut dibuat.
2. *Client* dan *server* tidak saling bergantung satu sama lain. *Client* disini diartikan sebagai aplikasi yang membutuhkan data dari *server*, umumnya seperti sebuah aplikasi web, aplikasi *mobile*, dan lainnya, sedangkan *server* dapat diartikan sebagai aplikasi yang memiliki dan mengolah data. Kedua aplikasi tersebut seharusnya dapat berjalan secara independen dan informasi penting yang dibutuhkan oleh *client* adalah lokasi dimana sumber daya yang diinginkan tersebut berada, biasanya direpresentasikan dalam bentuk sebuah URL (Uniform Resource Locator).
3. Bersifat *stateless*, dimana sebuah *request* terhadap suatu sumber daya seharusnya sudah mengandung tiap informasi yang diperlukan oleh *server* untuk dapat memproses permintaan tersebut.
4. Datanya yang dapat di-*cache*. Saat memungkinkan, suatu sumber daya atau data yang ada sebaiknya dapat di-*cache*, baik dari sisi *client* maupun *server*. *Cache* dapat diartikan sebagai sebuah komponen yang bertugas untuk menyimpan sementara data yang sering diakses untuk mengurangi waktu yang dibutuhkan untuk mengambil data tersebut ketika permintaan lain datang.
5. Arsitektur sistem yang ber-*layer*. Dalam sebuah REST API, *request* dan *response* dapat diproses oleh berbagai *layer*. *Client* dan *server* mungkin tidak terhubung secara langsung dan mungkin terdapat beberapa perantara di antara komunikasi dua entitas tersebut.

B. .NET

.NET (dotnet) merupakan sebuah platform yang dapat digunakan untuk membuat aplikasi *desktop*, *mobile*, dan *web* [4]. Platform ini memiliki beberapa komponen di dalamnya, yaitu *runtime, libraries, compiler, SDK,* dan *app stacks*. *Runtime* bertugas untuk mengeksekusi kode aplikasi, *libraries* berguna untuk menambahkan fungsionalitas baru pada kode program, contohnya seperti kemampuan untuk memproses dokumen JSON. *Compiler* bertugas untuk mengkompilasi kode program menjadi kode yang dapat dieksekusi oleh *runtime*. SDK berguna untuk membuat dan memonitor aplikasi, sedangkan *app stacks* berguna untuk mempermudah pengguna untuk membuat aplikasi menggunakan platform .NET, contohnya dengan menggunakan *framework* ASP.NET untuk membuat aplikasi web [5].

Terdapat beberapa implementasi dari platform .NET sendiri, yaitu .NET Framework, Mono, dan .NET Core [5]. .NET Framework adalah implementasi awal dari platform ini, dimana penggunaannya spesifik ke sistem operasi Windows dan Windows Server. Mono adalah implementasi .NET pertama yang diawali dari komunitas dan bersifat *open-source*, dimana setiap orang memiliki kesempatan untuk turut berkontribusi pada pengembangannya. Pada tahun 2016, Microsoft mengumumkan .NET Core yang memungkinkan adanya sistem *cross-platform*, dimana pengembangan maupun penggunaan aplikasi yang menggunakan platform .NET tidak lagi terbatas hanya pada Windows, namun juga dapat dilakukan di sistem operasi MacOS dan juga Linux [6].

C. Unit Test

Unit testing adalah teknik yang menguji bentuk atau bagian terkecil dari sebuah susunan kode atau sebuah modul berfungsi sebagaimana mestinya [7]. Umumnya dalam proses pengembangan suatu perangkat lunak, *unit test* akan dijalankan secara otomatis tiap kali terjadi perubahan dalam kode sehingga jika terjadi kesalahan dalam perilaku program, dapat dideteksi sedini mungkin. Cara sebuah *unit test* untuk dapat berinteraksi dengan sebuah blok kode adalah melalui input yang diberikan pada kode tersebut lalu melakukan validasi terhadap hasilnya. Sebuah blok kode atau sebuah fungsi dapat memiliki lebih dari satu *test case*.

Terdapat beberapa strategi yang dapat digunakan dalam pembuatan *unit test*, diantara lain seperti *logic check*, *boundary check*, dan *error handling*. *Logic check* berhubungan dengan memastikan apakah kode telah menjalankan alur yang tepat sesuai dengan input yang diberikan dan apakah percabangan yang ada telah masuk ke dalam cakupan pengujian. *Boundary check* berhubungan dengan menguji bagaimana suatu kode berinteraksi dengan input yang beragam, terutama input yang tidak sesuai dengan spesifikasi atau kebutuhan kode tersebut. Sedangkan *error handling* berhubungan dengan menguji bagaimana suatu kode berperilaku jika terjadi *error* akibat input yang salah.

Cakupan sebuah *unit test* adalah satu bagian tertentu saja tiap tes-nya. Oleh karena itu, jika suatu unit memiliki ketergantungan (*dependency)* dengan unit lain, maka akan dilakukan *mocking*. *Mocking* adalah sebuah metode untuk mengganti suatu objek *dependency* dengan objek palsu. Objek palsu tersebut didesain untuk bertindak seperti objek aslinya, namun tidak menjalankan fungsi-fungsi yang ada di objek asli tersebut. Data yang keluar ketika suatu fungsi dipanggil di objek palsu tersebut adalah data palsu yang sudah dikonfigurasi sesuai dengan kebutuhan tes tersebut. Hal ini dilakukan agar jangkauan *unit test* tetap pada unit yang diuji dan tidak melebar ke komponen-komponen lain.

D. Integration Test

Integration testing secara umumnya dibuat setelah *unit test*, dimana tujuan utamanya adalah untuk memastikan setiap komponen dapat bekerja sama untuk memenuhi suatu kebutuhan [8]. Perbedaan antara *unit test* dan *integration test* terletak pada subjek yang diuji, dimana *unit test* biasanya berfokus pada satu bagian tertentu di tiap tes-nya, sedangkan *integration test* biasanya menguji beberapa komponen sekaligus dan memastikan tiap komponen tersebut berfungsi dengan baik.

Keuntungan dari penerapan *integration testing* adalah memberikan kepercayaan diri lebih bahwa tiap komponen yang ada dapat bekerja sama dengan baik dan mampu memberikan fungsionalitas yang diinginkan [9]. Hal ini juga berlaku untuk komponen atau jasa eksternal yang digunakan oleh produk tersebut, sehingga jika terjadi kesalahan yang berhubungan dengan penggunaan jasa eksternal tersebut, dapat segera dideteksi dan diperbaiki. *Integration testing* juga berguna untuk melakukan validasi terhadap bagaimana data diproses di tiap-tiap bagiannya.

Perbedaan lainnya antara *integration testing* dan *unit testing* adalah bagaimana pengujian ini berhadapan dengan objek yang bergantung pada objek lain (*dependency*). Dalam *unit test*, tiap *dependency* akan diganti dengan objek palsu agar cakupannya tetap kecil. Sedangkan dalam *integration test*, tiap *dependency* tetap berhubungan dengan objek aslinya sehingga fungsionalitasnya tetap sama dan komponen-komponen tersebut dapat diuji sebagai satu kesatuan.

IV. HASIL PEKERJAAN

E. Deskripsi Proyek

Proyek yang akan dikerjakan merupakan sebuah aplikasi web pengelola data bootcamp PT Mitra Integrasi Informatika. Tujuan utama aplikasi ini adalah untuk membantu proses bisnis yang berhubungan dengan program bootcamp, baik dari segi perencanaan dan manajemen kelas, pendataan peserta, penilaian dan evaluasi performa peserta, dan lainnya. Target pengguna dari aplikasi ini sendiri merupakan karyawan dari PT Mitra Integrasi Informatika yang memiliki posisi sebagai trainer, manajer, serta admin.

Fitur utama dalam aplikasi ini dapat dibagi menjadi beberapa modul, yaitu:

1. Modul pegawai.
   Para peserta program bootcamp akan diklasifikasikan sebagai salah satu pegawai dari PT Mitra Integrasi Informatika. Data diri dari peserta bootcamp akan dikelola di modul ini. Fitur utamanya adalah menambahkan pegawai baru, mengubah data pegawai, menghapus data, serta mengunduh data pegawai.
2. Modul kelas.
   Modul ini terdiri dari beberapa bagian pendukung, yaitu bagian kategori kelas, batch, segment, topik pembelajaran, dan trainer. Tiap modul pendukung tersebut memiliki fitur untuk menambahkan, mengubah, dan menghapus data. Data yang dibuat dari modul pendukung tersebut akan digunakan untuk membuat data kelas. Sebuah kelas bootcamp akan berisi data mengenai batch peserta, topik utama pembelajarannya, segment pembelajaran dari kelas tersebut, peserta yang terdaftar di kelas tersebut, nilai akhir peserta, dan lainnya. Data kelas juga dapat diunduh dalam bentuk *spreadsheet*.
3. Modul aset.
   Modul ini berisi dengan informasi yang berhubungan dengan aset perusahaan yang berkaitan dengan berjalannya program bootcamp. Contoh asetnya antara lain seperti *access card*, USB *wireless*, dan *locker*. Modul ini berisi fitur untuk menambahkan, mengubah, dan menghapus aset, serta mencatat pegawai yang meminjam suatu aset, tanggal diterimanya aset tersebut, serta tanggal pengembaliannya.
4. Modul interview.
   Peserta bootcamp yang telah menyelesaikan programnya akan menjadi pegawai *idle*. Jika ada perusahaan yang tertarik untuk merekrut pegawai tersebut, maka PT MII akan menjadi perantara dan mendaftarkan wawancara antara

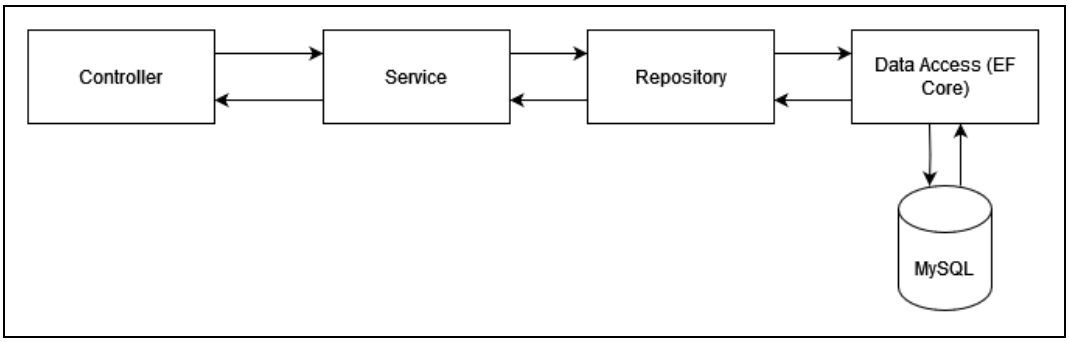
perusahaan dan pegawai yang bersangkutan. Modul ini memiliki fitur untuk mengelola data wawancara tersebut serta pengguna juga dapat mengunduh data interview dalam bentuk *spreadsheet*.

5. Modul placement.
   Saat pegawai *idle* tersebut berhasil lolos tahapan wawancara, maka datanya akan didaftarkan di modul ini. Informasi di dalamnya antara lain seperti pegawai yang bersangkutan, *client* atau perusahaan yang merekrut, serta tanggal mulai dan tanggal berakhirnya kontrak pegawai tersebut. Modul ini juga memiliki fitur untuk mengunduh data *placement* dalam bentuk *spreadsheet*.

Secara teknis, aplikasi web ini terbagi menjadi dua bagian, yaitu *front-end*, dan *back-end*. Bagian *front-end* bertugas untuk mengembangkan bagian *user interface* dari aplikasi web ini menggunakan React, sedangkan bagian *back-end* bertugas untuk mengembangkan REST API yang nantinya akan digunakan oleh bagian *front-end* untuk mendapatkan dan memproses data yang relevan. Fokus pada penelitian ini adalah mengembangkan proyek testing untuk REST API pengelola data bootcamp tersebut menggunakan framework xUnit.

F. Struktur Proyek

Proyek REST API ini dibuat menggunakan framework ASP.NET Core Web Api dan platform .NET Core 6. Proyek ini menggunakan konsep *controller* dan beberapa layer abstraksi.



Gambar 2. Struktur Proyek

Gambar 2 menunjukkan struktur proyek REST API tersebut, dimana tiap *endpoint* atau URL yang menuju ke sumber daya yang ada akan didefinisikan di bagian controller. Saat *request* datang, maka controller akan memanggil service yang bersangkutan untuk meminta data yang relevan. Service layer bertugas untuk mengurus semua logika bisnis yang ada. Saat service layer membutuhkan data dari database, maka service akan memanggil repository untuk mengambil data yang relevan. Repository layer akan memanfaatkan EntityFramework Core sebagai perantara antara database dan repository. Dalam kasus ini, database yang digunakan adalah MySQL.

G. Pengembangan Fitur Pengujian Perangkat Lunak

Pada proses pengembangan sebelumnya, tiap fitur yang diinginkan akan dikerjakan terlebih dahulu dan kemudian akan dilaksanakan proses testing di akhir. Tahapan pengembangan tersebut akan berpotensi menimbulkan masalah dan pekerjaan yang lebih banyak jika terdeteksi *bug* atau kesalahan pada kode pada akhir pengembangan, sehingga para pengembang harus kembali lagi ke fitur yang telah dibuat untuk memperbaiki *bug* yang ada. Oleh karena itu, dibutuhkan implementasi teknik pengujian berupa *unit testing* dan *integration testing* yang dapat memberikan validasi kepada pengembang bahwa kode yang ada telah berjalan sesuai keinginan.

Unit testing memiliki cakupan yang lebih kecil dan berfokus pada satu “unit” saja, sedangkan integration testing memiliki cakupan yang lebih luas dan menguji bagaimana tiap komponen saling berintegrasi satu sama lain. Proyek yang dikerjakan berbentuk sebuah API yang menerapkan pola desain abstraksi berupa *service layer* dan *repository layer* sebagai perantara antara *controller* dan database. Unit testing akan dilakukan dalam cakupan *service layer* yang memuat sebagian besar logika bisnis yang ada, sedangkan integration testing akan diterapkan di bagian *controller*, dimana tiap endpoint-nya didefinisikan.

Mengingat proyek ini telah dikembangkan secara cukup signifikan dan fitur-fitur baru berupa peningkatan dari pengembangan sebelumnya, maka *unit testing* dan *integration testing* akan diterapkan terhadap kode yang berhubungan dengan pengembangan fitur baru saja. Hal ini dilakukan untuk menghindari cakupan yang terlalu besar.

Dua proyek *testing* akan dibuat di dalam satu *solution* yang sama dengan proyek API tersebut. *Solution* adalah sebuah kontainer yang digunakan untuk mengorganisir satu proyek atau lebih [10]. Proyek testing pertama akan menampung segala

hal yang berhubungan dengan unit testing, sedangkan proyek testing kedua akan berisi setiap hal yang berhubungan dengan integration testing. Dalam operasinya, kedua proyek testing tersebut akan bergantung pada proyek API.

H. Pengembangan Proyek Unit Test

Proyek unit testing menggunakan template xUnit Test Project dan xUnit sebagai framework testing-nya. Digunakan juga beberapa library untuk membantu proses pengujian, antara lain seperti NSubstitute, AutoFixture, dan FluentAssertions. Library dapat diartikan sebagai suatu koleksi berisi kode-kode yang dapat digunakan untuk mengoptimalisasi suatu tugas [11]. NSubstitute adalah library yang memberikan kemampuan untuk mengganti suatu objek dengan objek palsu. AutoFixture berfungsi untuk membuat sebuah objek konkrit dari sebuah model secara otomatis tanpa harus menspesifikasikan propertinya satu persatu. Sedangkan FluentAssertions berfungsi untuk melakukan validasi terhadap output menggunakan sintaks yang mirip seperti sedang membuat kalimat dalam bahasa Inggris.

Semua logika bisnis yang ada di proyek API ini terdapat pada *layer service*, oleh karena itu, *unit test* akan dibuat untuk service untuk tiap modul utamanya. Contohnya seperti modul Asset yang memiliki *service* berupa *AssetService*, modul pegawai yang memiliki service berupa *EmployeeService*, dan lainnya.

```
public class AssetServicesTests
{
    // Dependencies
    private IAssetRepository _assetRepository = Substitute.For<IAssetRepository>();
    private IConfiguration? _configuration = Substitute.For<IConfiguration>();
    private IEmployeeAssetRepository _employeeAssetRepository = Substitute.For<IEmployeeAssetRepository>();
    private IEmployeeRepository _employeeRepository = Substitute.For<IEmployeeRepository>();
    private IWebHostEnvironment _environment = Substitute.For<IWebHostEnvironment>();
    private IParameterRepository _parameterRepository = Substitute.For<IParameterRepository>();
    private ITypeRepository _typeRepository = Substitute.For<ITypeRepository>();
    private TriggerFlowHandler _triggerFlowHandler;

    private AssetServices _assetService;

    private Fixture _fixture;

    public AssetServicesTests()
    {
        // Fixture
        _fixture = new Fixture();
        _fixture.Behaviors.Remove(item: new ThrowingRecursionBehavior());
        _fixture.Behaviors.Add(item: new OmitOnRecursionBehavior());

        // System Under Test
        _assetService = new AssetServices(
            _assetRepository,
            _employeeRepository,
            _typeRepository,
            _employeeAssetRepository,
            _parameterRepository,
            _triggerFlowHandler,
            _environment,
            _configuration
            );
```

Gambar 3. Konfigurasi Unit Test untuk AssetService

Gambar 3 menunjukkan *class* AssetServiceTests yang akan menjadi pusat dari tiap unit test yang berhubungan dengan AssetService. Unit test ini akan menggunakan objek asli dari AssetService, namun tidak dengan *dependency*-nya, oleh karena itu, semua objek yang dibutuhkan oleh AssetService akan diubah terlebih dahulu dengan objek palsu (*mock*) sebelum diberikan sebagai salah satu parameter di *constructor* AssetService. Proses *mocking* tersebut dibantu dengan library *NSubstitute*.

```
[Fact]
public void GetCurrentAsset_OrderByAccessType_ReturnOrderedListAscending()
{
    // Arrange
    string[] accessTypes = { "USB Wireless", "Locker", "Access Card" };
    var assets = _fixture.Build<Asset>().CreateMany(count: 3).ToList();

    for (int i = 0; i < assets.Count; i++)
    {
        assets[i].TypeNavigation!.Name = accessTypes[i];
    }

    // Method mocks.
    _assetRepository.Get(
        Arg.Any<Expression<Func<Asset, bool>>>(),
        Arg.Any<Func<IQueryable<Asset>, IOrderedQueryable<Asset>>>(),
        Arg.Any<Expression<Func<Asset, object>>[]>()
        ).Returns(assets);

    // Act
    var result = _assetService.GetCurrentAssets(page: 1, limit: 10, search: "", AlphabeticalOrderType.ASCENDING, orderByColumn: "access_type");
    var resultList = result.Data.ToList();

    // Assert
    result.Data.Should().NotBeNullOrEmpty();
    resultList[index: 0].AccessType.Should().Contain(expected: "Access Card");
    resultList[index: 1].AccessType.Should().Contain(expected: "Locker");
    resultList[index: 2].AccessType.Should().Contain(expected: "USB Wireless");
}
```

Gambar 4. Contoh test case untuk AssetService

Gambar 4 menunjukkan beberapa test case untuk AssetService dan method GetCurrentAsset. Sebuah test case biasanya akan dibagi ke dalam tiga tahapan utama, yaitu Arrange, Act, dan Assert. Tahap Arrange adalah dimana data-data yang dibutuhkan untuk melaksanakan test akan dipersiapkan. Hal ini juga meliputi fungsi dari objek palsu yang akan dipanggil oleh method GetCurrentAsset dari AssetService.

Setelah tahap Arrange selesai, maka akan dilanjutkan dengan tahap Act. Pada tahap ini, fungsi yang ingin diuji akan dipanggil dan hasilnya akan dicatat. Dalam kasus ini, sebuah variabel result akan menampung data yang dihasilkan dari fungsi GetCurrentAsset tersebut. Setelahnya, akan dilakukan verifikasi terhadap hasil fungsi tersebut pada tahapan Assert. Pada test case yang ditunjukkan pada Gambar 4, hasil yang diharapkan adalah data yang diurutkan berdasarkan huruf terkecil sampai terbesar dan atribut yang dijadikan acuan adalah AccessType. Oleh karena itu, hasil yang dipastikan adalah bahwa data pertama dari hasil GetCurrentAsset adalah aset dengan AccessType berupa “Access Card”, dilanjutkan dengan “Locker”, dan diakhiri dengan “USB Wireless”.

Penamaan test case pun mengikuti format tertentu, dimana penamaan sebuah test case akan dibagi ke dalam tiga bagian. Bagian pertama adalah method atau fungsi yang akan diuji, bagian kedua berupa kondisi pengujian, dan bagian ketiga adalah hasil yang diharapkan. Tiap bagian ini akan dipisahkan dengan simbol “_”.

I. Pengembangan Proyek Integration Test

Pendekatan dalam pembuatan proyek integration testing ini berbeda dengan unit testing, karena tujuan utamanya adalah menguji bagaimana tiap komponen yang berhubungan saling berinteraksi atau berintegrasi satu sama lain. Oleh karena itu, tidak diperlukan adanya *mocking* atau substitusi *dependency* dengan objek palsu, karena akan digunakan objek asli dalam proses pengujian, termasuk akses ke database-nya. Pengujian akan dilakukan dari sisi *controller*, dimana tiap *endpoint* atau URL yang menuju sumber daya tertentu didefinisikan.

Proyek integration testing juga menggunakan framework xUnit dan template xUnit Test Project. Library-library yang digunakan untuk proyek integration testing ini antara lain adalah “Microsoft.AspNetCore.Mvc.Testing” dan “FluentAssertions”. “FluentAssertions” digunakan untuk membantu proses verifikasi hasil dari objek yang sedang diuji, dalam hal ini berupa *endpoint* dari API aplikasi ini, sama seperti yang dilakukan pada proyek unit testing. Sedangkan library “Microsoft.AspNetCore.Mvc.Testing” akan berguna untuk menjalankan aplikasi API tersebut dalam konteks testing sehingga tiap endpoint-nya dapat diuji dalam kondisi yang mirip ketika sebuah *request* asli diterima.

Sebelum membuat test case, diperlukan konfigurasi terlebih dahulu terkait akses databasenya karena integration testing akan menggunakan koneksi database asli. Konfigurasi akan dilakukan terkait substitusi DbContext (objek dari EntityFramework Core yang merepresentasikan entitas dan perantara untuk akses data terhadap database), serta konfigurasi untuk menyiapkan database testing ketika test dijalankan dan pengembalian data ke kondisi awal setelah semua tes dijalankan.

```
protected override void ConfigureWebHost(IWebHostBuilder builder)
{
    builder.ConfigureTestServices(services =>
    {
        // Remove the default db context.
        services.RemoveAll(typeof(DbContextOptions<BootcampDbContext>));

        // And then add the db context that targets the Test DB.
        services.AddDbContext<BootcampDbContext>(options =>
        {
            var connectionString = $"server={HOST};user id={USERNAME};password=;port={PORT};persistsecurityinfo=True;database={DB_NAME};allo
            options.UseMySql(connectionString, ServerVersion.AutoDetect(connectionString));
        });
    });

    // Use Development as the environment when running the app.
    builder.UseEnvironment(environment: "Development");
}
```

Gambar 5. Konfigurasi terhadap substitusi DbContext

Gambar 5 menunjukkan potongan kode terhadap proses substitusi terhadap DbContext awal. Tahapan yang dilakukan adalah dengan menghapus DbContext standar (BootcampDbContext), lalu mendaftarkan DbContext baru dengan nama yang sama, namun dengan koneksi database yang berbeda, yaitu database testing. Setelahnya, dispesifikasikan juga bahwa *environment* yang digunakan untuk menjalankan proyek testing ini adalah *environment* "Development".

Setelahnya, akan dibuat sebuah *class* pembantu yang akan digunakan untuk mengelola kondisi database pengujian. *Class* ini akan mengimplementasi *interface* yang disediakan oleh framework xUnit, yaitu *IasyncLifetime*. *Interface* tersebut akan memberikan akses ke dua fungsi, yaitu *InitializeAsync* dan *DisposeAsync*. Fungsi *InitializeAsync* akan dijalankan sebelum semua *test case* dijalankan, sedangkan fungsi *DisposeAsync* akan dijalankan setelah semua *test* case dijalankan.

```
public async Task InitializeAsync()
{
    // This method will be executed before any tests in the collection.
    // It'll preapare the test database if user haven't created it.
    var dbConnection = new MySqlConnection(connectionString);
    await dbConnection.OpenAsync();

    // This script will check the existence of the test database.
    var commandText = $"SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA WHERE SCHEMA_NAME = '{DB_NAME}';";
    var dbCommand = new MySqlCommand(commandText, dbConnection);
    var result = await dbCommand.ExecuteScalarAsync();
    await dbConnection.CloseAsync();

    // If the database is not found, then run the DB creation script.
    if (result is null)
    {
        string filePath = Directory.GetCurrentDirectory() + "/Config/db_brm_test_creation.sql";

        if (!File.Exists(filePath))
        {
            throw new Exception();
        }

        string script = File.ReadAllText(filePath);

        var mysqlConnection = new MySqlConnection(connectionString);

        await mysqlConnection.OpenAsync();

        MySqlCommand command = new(script, mysqlConnection);

        try
        {
            await command.ExecuteNonQueryAsync();
        }
        catch (Exception ex)
        {
            throw new Exception(ex.Message);
        }
        finally
        {
            await mysqlConnection.CloseAsync();
        }
    }
}
```

Gambar 6. Kode pada fungsi InitializeAsync

Gambar 6 menunjukkan kode pada fungsi *InitializeAsync*. Fungsi tersebut akan memeriksa keberadaan database pengujian. Jika tidak ditemukan, makan akan dieksekusi sebuah *script* SQL untuk membuat database pengujian tersebut beserta mempopulasikan database-nya dengan data-data pengujian.

```
public async Task DisposeAsync()
{
    // This method will be executed once all tests contained within DatabaseCleanupCollection have completed.

    string filePath = Directory.GetCurrentDirectory() + "/Config/db_brm_test_init.sql";

    if (!File.Exists(filePath))
    {
        throw new Exception();
    }

    string connectionString = $"server={HOST},{PORT};user
id={USERNAME};password=;database={DB_NAME};persistsecurityinfo=True;allowuservariables=True;ConvertZeroDatetime='True';
DefaultCommandTimeout=0";
    string script = File.ReadAllText(filePath);

    var mysqlConnection = new MySqlConnection(connectionString);

    await mysqlConnection.OpenAsync();

    MySqlCommand command = new(script, mysqlConnection);

    try
    {
        await command.ExecuteNonQueryAsync();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message);
    }

}
```

Gambar 7. Kode pada fungsi DisposeAsync

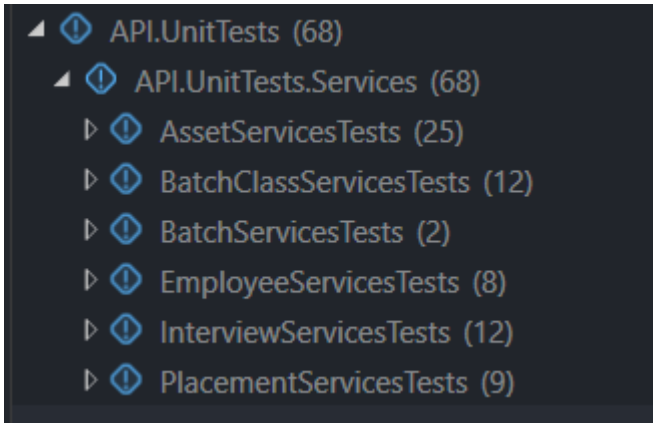
Gambar 7 menunjukkan kode yang terdapat pada fungsi *DisposeAsync*. Fungsi ini mengeksekusi sebuah script SQL untuk mengembalikan setiap data yang ada di database pengujian kembali ke kondisi awal. Setelah proses konfigurasi tersebut selesai, maka akan dilanjutkan dengan proses pembuatan test case. Mirip seperti pengembangan unit testing sebelumnya, pengembangan proyek integration testing ini juga berfokus pada modul-modul utamanya beserta controller yang berhubungan dengan modul tersebut, contohnya seperti modul aset yang memiliki *controller* berupa *AssetController*.

```
[Fact]
public async Task ListAsset_OrderByAccessType_ReturnOrderedList()
{
    // Arrange
    _client = _client.DefaultRequestHeaders.Authorization?.Scheme is null ? await LoginAsync(_client) : _client;

    // Act
    var body = await _client.GetAsync(requestUri: "/assets/current?page=1&limit=10&orderAlphabetically=1&orderByColumn=access_type");
    var content = await body.Content.ReadAsStringAsync();
    var result = JsonConvert.DeserializeObject<ResponsePaginationHandler<GetAssetCurrentDto>>(content);

    // Assert
    result.Should().NotBeNull();
    result.Code.Should().Be(expected: 200);
    result.Data.Should().NotBeNullOrEmpty();
    result.Data.First().AccessType.Should().Be(expected: "Access Card");
}
```

Gambar 8. Contoh salah satu test case untuk AssetController

Gambar 8 menunjukkan salah satu contoh test case yang dibuat untuk AssetController dimana fungsi ListAsset akan diuji dengan kondisi bahwa pengguna menginginkan adanya sorting data berdasarkan AccessType dan hasil yang diharapkan adalah data yang diurutkan berdasarkan alfabetnya, dengan atribut Access Type sebagai acuan. Pada bagian Arrange, tahapan yang dilakukan adalah mengatur nilai dari header "Authorization" dari HttpClient yang akan digunakan untuk mengirimkan sebuah GET request kepada endpoint aset yang didefinisikan oleh fungsi ListAsset. Setelahnya, akan dipanggil sebuah method GetAsync dengan parameter berupa endpoint yang ingin diuji dan query-nya. Hasil dari pemanggilan fungsi tersebut kemudian akan diubah dari JSON menjadi objek yang kompatibel. Sedangkan pada tahapan Assert, hasil tersebut akan diverifikasi berdasarkan hasil yang diharapkan.

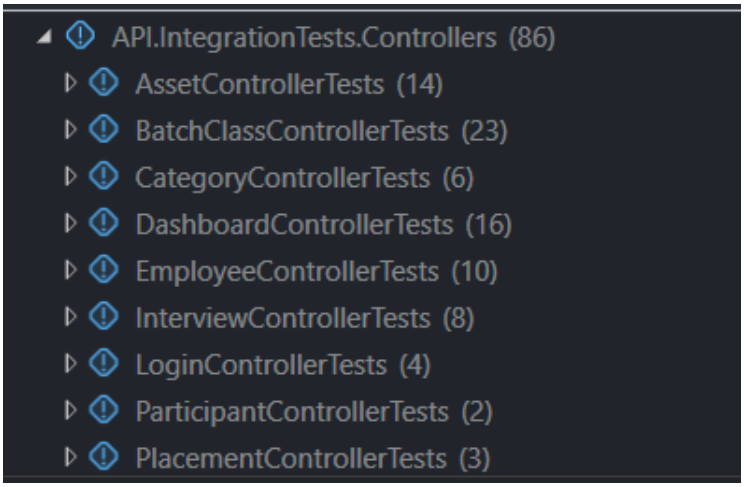
J. Hasil Pengembangan Unit Test



Gambar 9. Hasil pengembangan proyek unit test

Gambar 9 menunjukkan hasil dari pengembangan proyek unit test. Total terdapat enam puluh delapan *test case* yang telah dibuat untuk proyek unit test tersebut. Dua puluh lima diantaranya terdapat pada unit test untuk *class AssetService*, dua belas untuk *class BatchClassService*, dua untuk *class BatchService*, delapan untuk *class EmployeeService*, dua belas untuk *class InterviewService*, dan sembilan untuk *class PlacementService*.

K. Hasil Pengembangan Integration Test

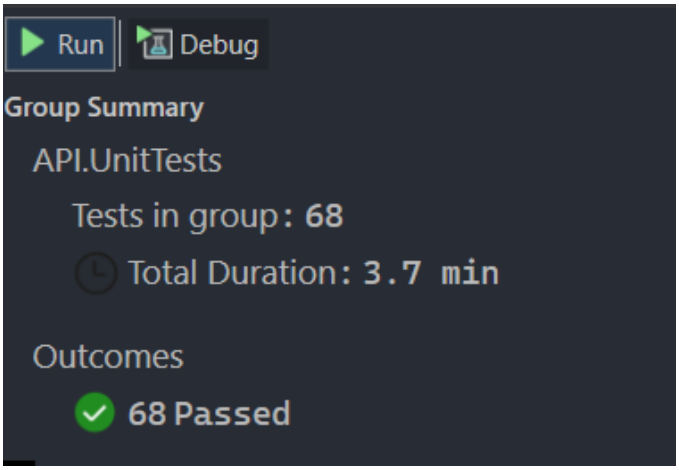


Gambar 10. Hasil pengembangan proyek integration test

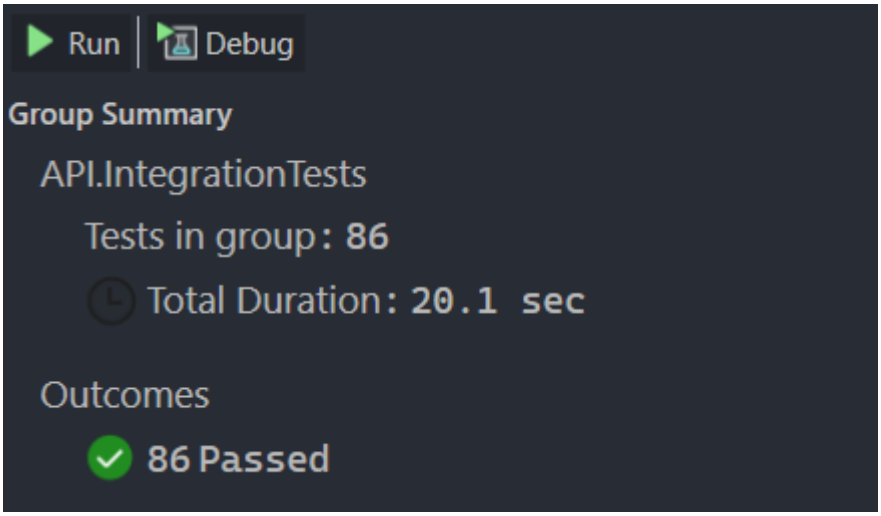
Gambar 10 menunjukkan hasil dari integration test yang telah dikembangkan untuk berbagai controller. Terdapat total delapan puluh enam *test case* dari proyek tersebut, dimana empat belas diantaranya terdapat pada *AssetController*, dua puluh tiga pada *BatchClassController*, enam pada *CategoryController*, enam belas pada *DashboardController*, sepuluh pada *EmployeeController*, delapan pada *InterviewController*, empat pada *LoginController*, dua pada *ParticipantController*, dan tiga pada *PlacementController*.

L. Evaluasi Hasil Kerja

Berdasarkan hasil pekerjaan yang telah dihasilkan dan kebutuhan dari PT MII, maka dapat dikatakan bahwa pekerjaan yang dilakukan telah berhasil memenuhi kebutuhan dan menyelesaikan masalah yang dihadapi. Dalam pengerjaannya, tidak terlalu banyak masalah teknis yang ditemukan, namun diperlukan beberapa konsultasi dengan tim agar implementasi solusi dapat sesuai dengan kebutuhan yang ada.



Gambar 11. Hasil unit test



Gambar 12. Hasil integration test

Gambar 11 menunjukkan hasil dari eksekusi proyek unit test. Dari total enam puluh delapan *test case* yang ada, setiap *test case* telah berhasil lolos tahap pengujian. Begitupun dengan delapan puluh enam *test case* yang terdapat pada proyek *integration test*, seperti yang ditunjukkan pada Gambar 12. Oleh karena itu, dapat disimpulkan bahwa fitur yang telah dikembangkan untuk aplikasi tersebut telah berjalan sebagaimana mestinya.

V. SIMPULAN DAN SARAN

A. *Simpulan*

Berdasarkan masalah yang telah dirumuskan, maka dapat disimpulkan bahwa permasalahan yang ada telah berhasil diatasi dengan solusi yang diterapkan dimana kebutuhan akan pengujian diatasi dengan penerapan teknik pengujian perangkat lunak berupa *unit testing* dan *integration testing*.

B. *Saran*

Saran yang dapat diberikan untuk pengembangan penelitian ini adalah penerapan sistem *Continuous Integration* (CI). *Continuous Integration* adalah sebuah praktek dimana perubahan yang terjadi dalam suatu kode akan diintegrasikan secara otomatis. Dalam proses ini, saat seorang pengembang mengajukan perubahan kode, perubahan tersebut akan diuji dulu ketepatannya sebelum digabungkan ke kode yang sudah ada. Hal tersebut dilakukan agar perubahan baru tidak menyebabkan permasalahan baru. Dengan menerapkan CI menggunakan bantuan jasa seperti GitHub Actions, maka proyek pengujian seperti *unit testing* dan *integration testing* dapat dijalankan secara otomatis sehingga proses integrasi kode menjadi lebih mudah.

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada PT Mitra Integrasi Informatika yang telah memberikan kesempatan untuk melaksanakan kegiatan magang. Penulis juga ingin mengucapkan kepada semua pihak yang telah memberikan bantuan dan bimbingan sehingga kegiatan magang dapat berjalan dengan lancar.

DAFTAR PUSTAKA

[1] PT Mitra Integrasi Informatika website. [Online]. Tersedia: https://www.mii.co.id/id/about-us
[2] A. Tarkowska, D. Carvalho-Silva, C. E. Cook., Edd T., R. D. Finn, A. D. Yates, "Eleven quick tips to build a usable REST API for life sciences," PLoS Comput Biol, vol. 14 (12), 2018.
[3] C. Rodriguez, M. Baez, F. Daniel, F. Casati, J. C. Trabucco, L. Canali dan G. Percannella, "REST APIs: A Large-Scale Analysis of Compliance with Principles and Best Practices," dalam *International Conference on Web Engineering*, 2016.
[4] Amazon Web Services, "What Is .NET?," Amazon Web Services, [Online]. Available: https://aws.amazon.com/what-is/net/. [Diakses 31 Maret 2024].
[5] Microsoft, "Introduction to .NET," Microsoft, 10 Januari 2024. [Online]. Tersedia: https://learn.microsoft.com/en-us/dotnet/core/introduction?WT.mc_id=dotnet-35129-website.
[6] P. Japikse dan A. Troelsen, Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming 11th ed. Edition, Apress, 2022.
[7] Z. Yuan, M. Liu, S. Ding, K. Wang, Y. Chen, X. Peng dan Y. Lou, "No More Manual Tests? Evaluating and Improving ChatGPT," *arXiv preprint arXiv:2305.04207*, vol. 1, 2024.
[8] R. Mohanan, "What Is Integration Testing? Types, Tools, and Best Practices," Spiceworks, 21 November 2022. [Online]. Available: https://www.spiceworks.com/tech/devops/articles/what-is-integration-testing/
[9] Katalon, "What is Integration Testing? Definition, How-to, Examples," Katalon, [Online]. Available: https://katalon.com/resources-center/blog/integration-testing.
[10] Microsoft, "Introduction to projects and solutions," Microsoft, 5 Desember 2023. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/get-started/tutorial-projects-solutions?view=vs-2022.
[11] R. Barons, "What are libraries in programming?," iDTech, 11 September 2020. [Online]. Available: https://www.idtech.com/blog/what-are-libraries-in-coding.