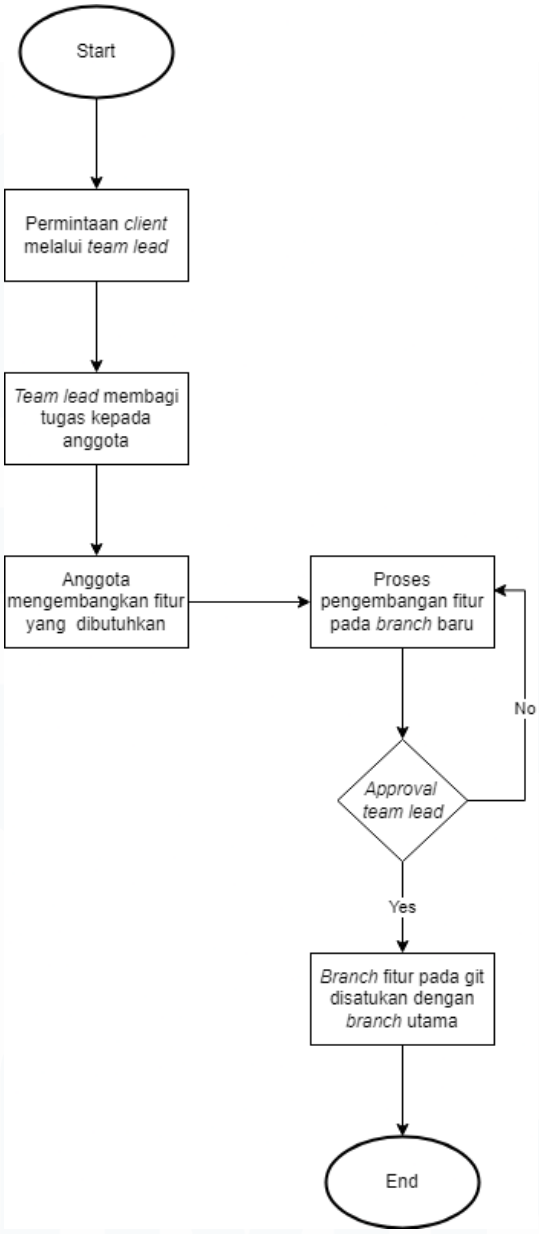
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Kegiatan magang di PT Adicipta Inovasi Teknologi sebagai *Web Developer Intern* berada dalam divisi *Implementation*, tepatnya berada di *Business Unit* 1 di PT Adicipta Inovasi Teknologi. *Business Unit* 1 atau yang biasa disebut CONFINS adalah salah satu bisnis utama dan produk dari PT Adicipta Inovasi Teknologi yang merupakan sistem berbasis web yang dibuat khusus untuk perusahaan atau klien dalam ranah *multifinance*. Divisi *implementation* dalam melakukan komunikasi antar anggota tim, secara umum menggunakan platform *Microsoft Teams* untuk berkomunikasi melalui *chat*, *call*, *meet*, dan penggunaan lainnya. Pembagian jadwal kerja di divisi *implementation* menetapkan tiga hari bekerja secara langsung di kantor (WFO) yaitu di hari Senin, Selasa, dan Rabu, dan dua hari bekerja secara daring dari rumah (WFH) yaitu di hari Kamis dan Jumat.

Alur kerja pada divisi *Implementation* diawali dari produk mentah yang sudah ada dari PT Adicipta Inovasi Teknologi ditawarkan kepada calon *client*. Ketika *client* sudah memutuskan untuk menandatangani kontrak, maka dari pihak PT Adicipta Inovasi Teknologi akan melakukan tahapan *user requirements* untuk mencatat fitur-fitur apa saja yang perlu dikembangkan atau dihilangkan sesuai dari permintaan *client*. Pengembangan fitur ini merupakan pekerjaan yang perlu dilakukan oleh *developer* di divisi *Implementation.* Gambaran alur kerja seperti pada Gambar 3. 1 berikut.



Gambar 3. 1 Alur Kerja Magang

Pada Gambar 3. 1 menggambarkan permintaan *client* berupa pengembangan fitur disampaikan kepada *team leader* atau *project manager*, selanjutnya beberapa permintaan tersebut akan dibagikan *team leader* kepada anggota timnya. Setelah mendapat tugas masing-masing, setiap anggota akan melakukan pengembangan fitur-fitur yang diminta oleh *client* yang disampaikan oleh *team leader*. Selanjutnya pada proses pengembangan fitur, dilakukan di *branch* masing-masing anggota sehingga tidak bertumpuk satu sama lain. Ketika hasil perubahan tersebut sudah

dilakukan testing dan di *approve* oleh *team leader*, maka hasil pengembangan fitur di *branch* pribadi akan di *update* ke utama, dan jika belum mendapat *approval* dari *team leader* atau belum sesuai dengan ketentuan yang ada, maka kembali lagi ke tahap proses pengembangan fitur di branch masing-masing.

3.2 Tugas dan Uraian Kerja Magang

Staf *developer* magang di divisi *Implementation* pada PT Adicipta Inovasi Teknologi secara umum bertugas sebagai penghubung antar *client* dengan perusahaan PT Adicipta Inovasi Teknologi. Awalnya, produk mentah atau awal (*core*) dari PT Adicipta Inovasi Teknologi ditawarkan kepada *client multifinance*. Setiap *client multifinance* memiliki cara kerjanya tersendiri, dengan fitur dan modul yang berbeda, sehingga perlu dilakukan *adjustment* sesuai dengan permintaan *client*. Divisi *implementation* bertugas untuk menerapkan (*implementing*) dan mengembangkan permintaan *client* sesuai dari hasil *user requirement*, sehingga divisinya disebut dengan *implementation*.

Selama kegiatan magang berlangsung, para staf magang di PT Adicipta Inovasi Teknologi dibagi ke dalam tiga bagian, yaitu masa pelatihan intensif, masa pelatihan praktek kerja, dan praktek kerja. Pelaksanaan pelatihan praktek kerja dan praktek kerja diterapkan dalam menangani 1 proyek dengan 1 *client*. Selama pengerjaan proyek dalam masa praktek kerja, para peserta magang mendapatkan pembelajaran *hard skills* dan *soft skils* dari praktek kerja dengan terjun langsung ke proyek. Tabel 3. 1 menjelaskan perincian tugas dan uraian pelaksanaan kerja magang di PT Adicipta Inovasi Teknologi.

Tabel 3. 1 Perincian Tugas Kerja Magang

No	Pekerjaan	Ming-gu	Tanggal Mulai	Tanggal Selesai
1	*On Boarding* dan pelatihan intensif	1-5	20 Februari 2024	20 Maret 2024

No	Pekerjaan	Ming-gu	Tanggal Mulai	Tanggal Selesai
	1.1 Pelatihan *database* SQL	1	21 Februari 2024	21 Februari 2024
	1.2 Pelatihan Basic LINQ dan framework ASP.NET	1	22 Februari 2024	22 Februari 2024
	1.3 Pelatihan migrasi database dan pembuata data model	1	23 Februari 2024	23 Februari 2024
	1.4 Pelatihan membuat api *create, read, update, delete*, dan *get,* dan *error message*	2	26 Februari 2024	26 Februari 2024
	1.5 Pelatihan *code guideline*	2	27 Februari 2024	27 Februari 2024
	1.6 Pelatihan *engine camunda*	2	28 Februari 2024	28 Februari 2024
	1.7 Pelatihan *framework* Angular	2-3	29 Februari 2024	08 Maret 2024
	1.8 Ujian hasil pelatihan	4-5	12 Maret 2024	20 Maret 2024
2	Pengenalan ke tim dan pengenalan proyek	5-6	21 Maret 2024	25 Maret 2024
	2.1 *Mapping framework*	5	21 Maret 2024	21 Maret 2024

No	Pekerjaan	Ming-gu	Tanggal Mulai	Tanggal Selesai
	2.2 Memasukkan data aplikasi	6	22 Maret 2024	25 Maret 2024
3	Pembuatan *report* dan *document printing*	6-7	26 Maret 2024	5 April 2024
	3.1 Membuat *report*	6	26 Maret 2024	28 Maret 2024
	3.2 Membuat *document printing*	7	1 April 2024	5 April 2024
4	Pembuatan mirroring	8-9	8 April 2024	16 April 2024
	4.1 Melakukan *mirroring* tabel	8	8 April 2024	15 April 2024
	4.2 *Testing hit* API di *consumer*	9	15 April 2024	16 April 2024
5	Penyelesaian *minor bug*	9-10	18 April 2024	22 April 2024
6	*Testing New Application*	10-11	23 April 2024	6 Mei 2024
7	*System Integration Testing*	12	7 Mei 2024	16 Mei 2024

3.2.1 *On Boarding* dan Pelatihan Intensif

Peserta magang PT Adicipta Inovasi Teknologi yang telah menyelesaikan proses seleksi, dan menandatangani surat kontrak penerimaan kerja, telah resmi bekerja di PT Adicipta Inovasi Teknologi pada tanggal 20 Februari 2024. Pada hari pertama bekerja, para peserta magang diarahkan untuk ke ruangan *training* guna diberi pembekalan mengenai profil perusahaan, visi-misi perusahaan, struktur

organisasi perusahaan, kebijakan dan aturan kerja yang berlaku di perusahaan untuk memastikan para peserta magang menaati kebijakan yang ada, pengenalan anggota dan pimpinan departemen, produk dan proyek hasil pengerjaan perusahaan. Perwakilan pimpinan per departemen yang ada di PT Adicipta Inovasi juga hadir untuk memberi sambutan kepada para peserta magang yang baru bergabung di perusahaan.

Pada tahapan *on boarding* perusahaan juga memberikan *office tour* untuk memperkenalkan tata letak yang ada pada perusahaan seperti ruang *meeting*, *cafeteria*, *pantry*, toilet, beberapa ruang kerja, parkiran motor dan mobil, musala. Setelah *office tour* dilakukan, para peserta magang juga diberikan peralatan dan perlengkapan untuk menunjang kebutuhan kegiatan magang selama periode satu tahun kedepan. Beberapa peralatan dan perlengkapan yang diberikan oleh *IT Support* PT Adicipta Inovasi Teknologi berupa akun *Microsoft Teams*, laptop, pengisi daya laptop, instalasi VPN (*Virtual Private Network*) kantor, dan *ID Card* yang berfungsi sebagai kartu pengenal dan kartu akses masuk kantor atau ruangan. Instalasi *tools* dan aplikasi yang akan digunakan selama masa kerja magang juga dilakukan setelah pembagian laptop. Beberapa *tools* dan aplikasi tersebut seperti *Visual Studio Code, Microsoft Visual Studio, SQL Server Management Studio, pgAdmin, Postman*, dan lainnya. Apabila semua perlengkapan dan peralatan berfungsi dengan baik, maka para peserta magang diizinkan pulang untuk mempersiapkan *training* keesokan harinya.

3.2.1.1 Pelatihan Database SQL

Pada hari pertama pelatihan, materi yang dibawakan adalah mengenai database SQL. *Software* yang digunakan untuk pelatihan SQL ini adalah *SQL Server Management Studio* (SSMS). Pelatihan dimulai pada pukul 08.30 yang dimulai dari pengenalan terhadap lingkungan SSMS dan penjelasan tentang konsep-konsep fundamental database seperti tabel, kolom, baris, dan tipe data. Setelah sesi pengenalan, instruktur mengenalkan peserta magang ke materi yang lebih dalam dengan menjelaskan operasi-

operasi dasar SQL seperti membuat *database* dan tabel seperti pada Gambar 3. 2.

```
CREATE DATABASE RENTAL

USE RENTAL

CREATE TABLE MSvideo (
    KdVideo char(5) not null PRIMARY KEY,
    NmVideo varchar(35) not null,
    Produksi varchar(15) not null,
    JenisVideo int not null,
    Stok int not null,
    Bahasa  varchar(10) not null
    );
```

Gambar 3. 2 *Query* Membuat *Database* dan Tabel

Serta operasi dasar SQL seperti *SELECT*, *INSERT*, *UPDATE*, dan *DELETE* seperti pada Gambar 3. 3Gambar 3. 3

```
INSERT INTO MSvideo (KdVideo, NmVideo, Produksi, JenisVideo, Stok, Bahasa)
VALUES ('V001', 'Pisces', 'Star', 1, 10, 'Indonesia'),
       ('V002', 'Capricorn', 'Fish', 2, 5, 'English'),
       ('V003', 'Libra', 'Robot', 3, 15, 'Korea'),
       ('V004', 'Scorpio', 'Lele', 4, 20, 'Jepang'),
       ('V005', 'Gemini', 'Soap', 1, 10, 'Malaysia');
```

Gambar 3. 3 *Query Insert* data ke Tabel dalam *Database*

Selain itu, peserta diajarkan bagaimana melakukan *query* sederhana untuk mengambil data dari database yang telah ada serta cara memanipulasi dan mengupdate data tersebut sesuai kebutuhan. Setelah pelatihan diberikan, instruktur memberikan soal latihan yang berkaitan dengan materi SQL yang baru dijelaskan. Soal terdiri dari sepuluh soal dengan menggunakan dua *database* yang berbeda. Soal yang diberikan bertujuan untuk menguji kemampuan para mahasiswa magang terkait pemahaman SQL dalam penerapan nyata studi kasus. Soal dan penyelesaian seperti Gambar 3. 4.

```sql
-- No 9 Buat SP untuk menampilkan data customer (db : northwind) yang tinggal di negara tertentu(pakai parameter).
GO
USE NORTHWND
GO
IF OBJECT_ID ('GetCustomersByCountry', 'P') IS NOT NULL
    DROP PROCEDURE GetCustomersByCountry;
GO
CREATE PROCEDURE GetCustomersByCountry
    @Country varchar(20)
AS
    SELECT *
    FROM NORTHWND.dbo.Customers
    WHERE Country = @Country
GO
--Execute Procedure--
    EXECUTE GetCustomersByCountry 'Spain'
    EXECUTE GetCustomersByCountry 'UK'
    EXECUTE GetCustomersByCountry 'Argentina'
```

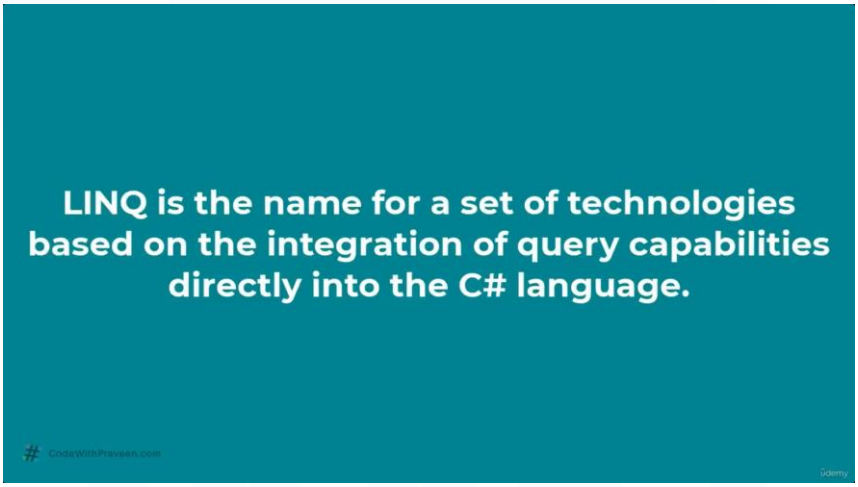
Gambar 3. 4 Soal *Stored Procedure*

Soal tersebut adalah mengenai *Stored Procedure*, yaitu kumpulan *statement* yang dapat dipanggil dari *query* lain atau *stored procedure* lain [6]. Sebuah prosedur dapat menerima argumen input dan mengembalikan nilai sebagai output. Prosedur ini diberi nama dan disimpan dalam sebuah dataset SQL. *Stored procedure* dapat mengakses atau memodifikasi data di beberapa dataset oleh beberapa pengguna. Dalam contoh soal di atas, para mahasiswa diminta untuk membuat SP yang dapat menampilkan data pelanggan dari *database* 'Northwind' dengan menggunakan parameter *country*.

3.2.1.2 Pelatihan Basic LINQ dan *framework* ASP.NET

Pada hari pelatihan kedua, materi yang diajarkan kepada peserta magang yaitu mengenai LINQ (*Language Integrated Query*) dan *framework* ASP.NET. Pelatihan dimulai pada pukul 08.30 dan diawali dengan melakukan *mapping framework* dari repositori perusahaan ke penyimpanan lokal. Perusahaan menggunakan Microsoft TFS (*Team Foundation Server*) yang merupakan sebuah platform yang dikembangkan oleh Microsoft untuk mengelola *lifecycle software development*. TFS menyediakan *tools* manajemen untuk *software development* dan memungkinkan tim untuk bekerja kolaboratif melalui *tools* dan prosesnya. TFS menawarkan beberapa fitur seperti kontrol versi, manajemen proyek, manajemen *build*, pengujian, dan lainnya [7].

LINQ adalah sebuah fitur yang diperkenalkan dalam .NET Framework versi 3.5 yang memungkinkan para *developer* untuk menuliskan *query* yang jelas dan mempermudah dalam bahasa pemrograman seperti C#. LINQ memungkinkan dan mempermudah untuk mengakses data dari berbagai sumber seperti contohnya *database*[8]. Penggunaan *query* di LINQ mirip dengan sintaks pada SQL seperti *query* '*from'*, '*where'*, dan '*select*'. Pada pelatihan LINQ, mahasiswa magang diberi pembekalan awal berupa beberapa video *course* dari *platform* Udemy yang berisikan penjelasan mengenai LINQ dimulai dari pengenalan LINQ, tipe data LINQ, object LINQ, operasi proyeksi, operasi *join*, operasi elemen, dan entitas LINQ seperti Gambar 3. 5.



Gambar 3. 5 Tampilan *Course* LINQ di Udemy

Pelaithan juga mencakup pelatihan dasar mengenai *framework* .NET, yaitu platform sumber terbuka untuk membangun aplikasi desktop, web, dan seluler yang dapat berjalan secara native pada sistem operasi apa pun. .NET terdiri dari *tools*, *library*, dan bahasa pemrograman yang mendukung pengembangan perangkat lunak modern, dan beperforma tinggi, .NET memungkinkan *developer* untuk menerjemahkan kode bahasa pemrograman .NET ke dalam instruksi yang dapat diproses oleh perangkat komputasi, menyediakan utilitas efisien untuk pengembangan perangkat lunak, dan menentukan satu set tipe data untuk menyimpan informasi [9].

Pelatihan LINQ di *platform* .NET mencakup mempelajari penggunaan perintah dasar .NET seperti '*dotnet new console*' untuk membuat proyek konsol baru, '*dotnet add reference*' untuk menambahkan referensi ke proyek lain, '*dotnet new classlib*' untuk membuat class library baru. Serta '*dotnet run*' untuk menjalankan aplikasi yang telah dibuat.

Pelatihan di hari kedua juga diakhiri dengan pemberian tugas pribadi untuk menguji kemampuan para mahasiswa magang akan materi yang baru dipelajarinya. Soal terdiri dari lima soal wajib dan satu bonus soal. Soal dikerjakan dengan membuat *class* terlebih dahulu seperti Gambar 3. 6.

```
public class Customer
{

    public Customer(int CustId, string CustName, string PhoneNumber, int CustJobId, int Rating, int CustAddressId)
        {
            this.CustId = CustId;
            this.CustName = CustName;
            this.PhoneNumber = PhoneNumber;
            this.CustJobId = CustJobId;
            this.Rating = Rating;
            this.CustAddressId = CustAddressId;

        }

        public int CustId { get; set; }

        public string CustName { get; set; }

        public string PhoneNumber { get; set; }

        public int CustJobId { get; set; }

        public int Rating { get; set; }

        public int CustAddressId { get; set; }


}
```

Gambar 3. 6 *Class Customer* pada .NET

Selanjutnya memasukkan data customer pada *class Program* dengan *function initialize.*

```
static void Initialize()
{
    customers.Add(new Customer(1, "Hadi Wijaya", "082381234723", 2, 75, 1));
    customers.Add(new Customer(2, "Suted budiman", "082176423922", 3, 66, 2));
    customers.Add(new Customer(3, "Nikolaj parkov", "082133498878", 2, 34, 2));
    customers.Add(new Customer(4, "Alena Valery", "081098487637", 1, 97, 4));
    customers.Add(new Customer(5, "Nausha Zasha", "082174837772", 7, 54, 3));
    customers.Add(new Customer(6, "Nika Olien", "081482742231", 4, 100, 2));
    customers.Add(new Customer(7, "Alexei Tolya", "082148882777", 5, 80, 3));
    customers.Add(new Customer(8, "Pavlo Valery", "084231472423", 6, 76, 1));
    customers.Add(new Customer(9, "Rostislav Wijaya", "082177263513", 6, 64, 1));
```

Gambar 3. 7 *Add* Data *Customer*

Berikut contoh soal dan jawaban pada tugas yang diberikan, di mana pada soal diminta untuk membuat query yang menampilkan *customer* dengan *rating* lebih dari 50 dan memiliki pekerjaan Petani.

```
static void SoalSatu()
{
    Console.WriteLine("Customer dengan Rating > 50 dan Pekerjaan = petani: ");
    var query = from c in customers
                join j in jobs on c.CustJobId equals j.CustJobId
                where j.JobName.Equals("Petani")
                where c.Rating > 50
                select new {c.CustId, c.CustName, c.PhoneNumber, c.CustJobId, j.JobName, c.Rating, c.CustAddressId};
    foreach (var item in query)
    {
        Console.WriteLine($"Cust ID = {item.CustId}\n" +
                          $" Nama = {item.CustName}\n " +
                          $"Phone = {item.PhoneNumber}\n " +
                          $"JobID = {item.CustJobId} || " +
                          $"Pekerjaan = {item.JobName}\n " +
                          $"Rating = {item.Rating}\n " +
                          $"AddressID = {item.CustAddressId}");
    }
}
```

Gambar 3. 8 Soal LINQ

3.2.1.3 Pelatihan Migrasi Database dan Pembuatan Data Model

Pada hari pelatihan ketiga, materi yang diajarkan yaitu cara melakuan migrasi *database* dan membuat data model pada *framework* perusahaan yang berbasis ASP.NET. Tahap pertama sebelum melakukan migrasi *database* adalah membuat data model berupa entitas yang merepresentasikan sebuah tabel dalam *database* seperti Gambar 3. 9.

```
[Table("REF_OFFICE")]

25 references
public partial class RefOffice : BaseEntity
{
    [Key]
    [Column("REF_OFFICE_ID")]
    [Required]
    7 references
    public long RefOfficeId { get; set; }

    [Column("OFFICE_CODE")]
    [Required]
    [StringLength(50)]
    11 references
    public string OfficeCode { get; set; }

    [Column("OFFICE_NAME")]
    [Required]
    [StringLength(100)]
    3 references
    public string OfficeName { get; set; }
```

Gambar 3. 9 Entitas dalam *Framework*

Pada Gambar 3.9 terdapat atribut [Table] yang menentukan nama tabel di *database*, selanjutnya terdapat atribut [Key] yang menandakan bahwa kolom tersebut merupakan *primary key* pada tabel, atribut [Column] digunakan untuk memetakan nama kolom ke dalam kolom di tabel *database*, atribut [Required] untuk menentukan secara spesifik bahwa properti kolom tersebut bersifat wajib atau tidak boleh *null*/kosong, atribut [StringLength(x)] digunakan untuk menentukan jumlah karakter maksimal pada string kolom sebesar x. Pada Gambar 3.9 juga terdapat properti 'public *data_type column_name* {get; set;} yang menggambarkan kolom pada tabel di *database*. Selain dari beberapa atribut tersebut, ada juga atribut [InverseProperty] yang digunakan untuk menentukan *inverse* navigasi antara dua entitas seperti menghubungkan dua entitas dan menandakan property mana yang berkaitan satu sama lain, lalu ada atribut [ForeignKey] yang menandakan kolom mana yang merupakan *foreign key* dalam suatu tabel seperti Gambar 3. 10.

```
[ForeignKey(nameof(ParentId))]
[InverseProperty("RefOffices")]
2 references
public virtual RefOffice RefOfficeParents { get; set; }
```

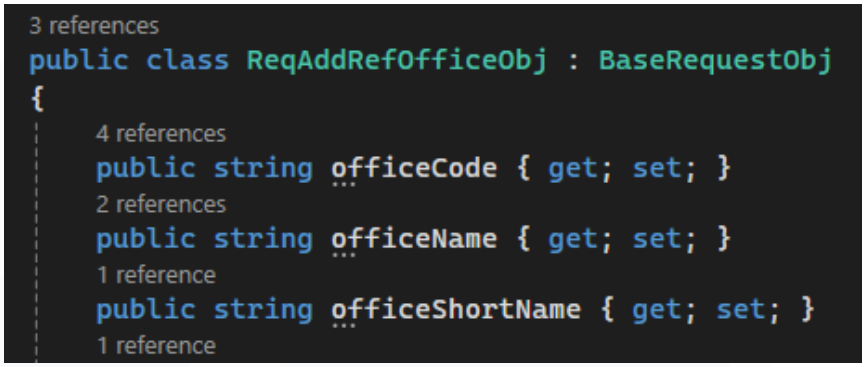
Gambar 3. 10 Atribut *ForeignKey* dan *InverseProperty*

3.2.1.4 Pelatihan Membuat API CRUD, *Get* dan *Error Message*

Pada hari pelatihan keempat, materi yang diajarkan masih melanjutkan menggunakan *framework* ASP.NET pada Microsoft Visual Studio. Setelah mahasiswa memahami cara membuat *database*, tabel, dan kolom, dan memahami cara melakukan migrasi dari *framework* .NET ke *database* PostgreSQL, selanjutnya dipelajari cara membuat API (*Application Programming Interface*) yaitu seperangkat aturan dan mekanisme yang memungkinkan berbagai *software* untuk berkomunikasi satu sama lain. API memungkinkan aplikasi untuk mengakses data atau layanan dari aplikasi lain secara terstruktur dan aman melalui internet [10].

API memungkinkan *developer* untuk mengintegrasikan fungsionalitas aplikasi mereka tanpa harus memahami detail implementasi aplikasi lain yang mereka akses. Selama masa pelatihan, API yang dibuat mencakup API *create, read, update, delete*. API *create* atau yang biasa disebut dengan API *add* yang dibuat dengan tujuan untuk menambahkan entitas baru ke dalam sistem dengan menggunakan metode HTTP POST, lalu ada api *read* atau yang biasa disebut dengan API *get* yang dibuat dengan tujuan untuk membaca atau mengambil data atau entitas dari sistem dengan menggunakan metode HTTP GET, selanjutnya ada api *update* atau yang biasa disebut dengan API *edit* yang dibuat dengan tujuan untuk memperbarui atau mengedit entitas yang sudah ada di dalam sistem dengan menggunakan metode HTTP PUT untuk mengganti keseluruhan entitas atau metode HTTP PATCH untuk mengganti sebagian entitas, ada juga api *delete* yang dibuat dengan tujuan untuk menghapus entitas dari sistem dengan menggunakan metode HTTP POST.

Contoh penggunaan keempat API tersebut dalam pelatihan adalah dalam pembuatan tabel RefOffice. Langkah pertama dalam membuat API di *framework* .NET dengan menggunakan C# adalah membuat DTO (*Data Transfer Object*) atau objek yang digunakan untuk mentransfer data antar komponen dalam sebuah aplikasi. DTO biasanya digunakan untuk mengurangi jumlah panggilan jaringan dengan menggabungkan beberapa data ke dalam satu objek yang kemudian dikirimkan sekaligus. DTO biasanya bersifat sederhana dan tidak mengandung logika bisnis. Contoh DTO yang dibuat untuk tabel RefOffice seperti pada Gambar 3.11.



Gambar 3. 11 *Code* DTO *Request* RefOffice

Gambar 3.11 merupakan DTO *request* untuk API *add* tabel *RefOffice* dengan nama *class* ReqAddRefOfficeObj yang mewariskan *class* BaseRequestObj yang artinya *class* ReqAddRefOfficeObj mengambil properti dan metode dari *class* BaseRequestObj. Lalu *class* ReqAddRefOfficeObj memiliki properti officeCode, officeName, dan officeShortName yang diperlukan untuk menambahkan *add* kantor baru ke dalam tabel di *database*. Setelah membuat DTO, Langkah selanjutnya adalah membuat *business service*, yaitu komponen yang menjalankan logika bisnis aplikasi, *service* biasanya digunakan untuk memisahkan logika bisnis dari lapisan presentasi (controller) dan data, sehingga membuat kode lebih terstruktur dan mudah untuk diuji. Contoh *business service* pada API *add* RefOffice seperti Gambar 3.12.

```
//Add RefOffice
[Decorate(typeof(TransactionHandler))]
2 references
public async Task<ResponseSuccessObj> AddRefOffice(ReqAddRefOfficeObj request)
{
    //Check only 1 office code di RefOffice
    RefOffice refOffice = GetRefOfficeByCode(request.officeCode).Result;
    if (refOffice != null)
    {
        string[] err = { request.officeCode };
        throw new AdInsCustomException(ExceptionConstant.DUPL_CODE, err);
    }

    if (request.mrOfficeTypeCode == "HO")
    { RefOffice type = GetRefOfficeByType().Result;
        if (type != null)
        {
            string[] err = {};
            throw new AdInsCustomException(ExceptionConstant.HO_ALRDY_EXIST, err);
        }
    }

    long? parentId;
    parentId = (from x in context.RefOffice
                    where x.OfficeCode == request.parentCode
                    select x.RefOfficeId).FirstOrDefault();

    refOffice = new RefOffice()
    {
        //entity = request.dtoreq
        OfficeCode = request.officeCode,
        OfficeName = request.officeName,
        OfficeShortName = request.officeShortName
    };
    await repository.AddAsync(refOffice);
    await repository.SaveChangesAsync();
    return new ResponseSuccessObj();
}
```

Gambar 3. 12 Kode *Service Add* RefOffice

Kode di Gambar 3.12 merupakan service dari AddRefOffice, di mana awalnya dilakukan validasi terlebih dahulu, apakah sudah ada kantor dengan kode yang sama pada tabel di *database*, jika ada, maka akan menampikan *error message* yang sudah dikustomisasi dari template perusahaan dengan menggunakan ExceptionConstant, sehingga akan menampilkan pesan *error* '*Duplicate Code*'. Dilakukan juga pengecekan apabila tipe kantor nya adalah HO (*Head Office*), maka akan menampilkan *error message* dengan pesan 'HO *Already Exists*' karena dalam satu perusahaan, hanya memiliki satu *head office*, sehingga tidak boleh ada duplikat. Kode selanjutnya, menuliskan entitas RefOffice baru dibuat dari data yang disediakan dalam objek *request* dari *class* ReqAddRefOfficeObj. Entitas tersebut kemudian ditambahkan ke repository dan perubahan

disimpan ke tabel RefOffice di *database*. Setelah *business service* telah dibuat, langkah selanjutnya adalah membuat *Interface*, yaitu kontrak dalam pemrograman yang mendefinisikan kumpulan metode tanpa memberikan implementasinya. Contoh *Interface* seperti pada Gambar 3.13

```
public interface IRefOfficeService
{
    Task<ResponseSuccessObj> AddRefOffice(ReqAddRefOfficeObj request);
```

Gambar 3. 13 Kode *Interface* AddRefOffice

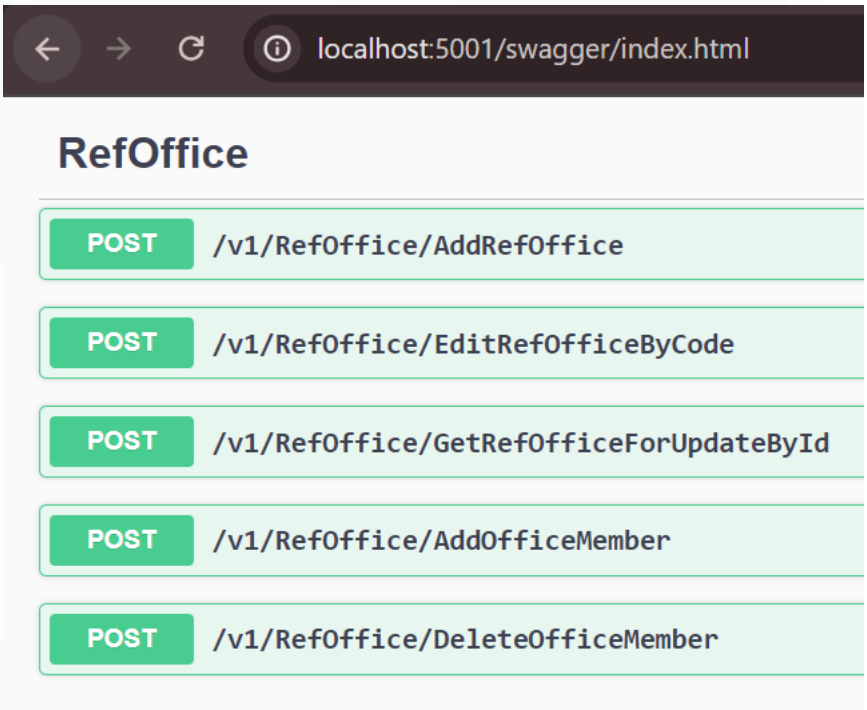
Interface pada Gambar 3.13 bernama IRefOfficeService. Salah satu metode yang dapat dipanggil adalah AddRefOffice yang sudah dibuat di *Business Service* sebelumnya. Langkah terakhir dalam membuat API di *framework* .NET adalah membuat *Controller*, yaitu komponen dalam arsitektur *Model-View-Controller* (MVC) yang bertugas untuk menangani permintaan pengguna, memproses *input*, dan mengembalikan hasil ke pengguna. Dalam aplikasi berbasis *web*, *controller* menerima HTTP *request*, memprosesnya dengan memanggil layanan atau komponen lain, dan kemudian menentukan HTTP *response* yang tepat. Contoh *Controller* seperti pada Gambar 3.14.

```
//Untuk Add Office
[Route("[action]")]
[HttpPost]
[ValidateDTO]
[MapToApiVersion("1")]
[AllowAnonymous]
0 references
public async Task<JsonResult> AddRefOffice(ReqAddRefOfficeObj request)
{
    ResponseSuccessObj response = iOfficeService.AddRefOffice(request).Result;
    return new JsonResult(response);
}
```

Gambar 3. 14 Kode *Controller* AddRefOffice

Pada *Controller* di Gambar 3.14 memiliki beberapa atribut seperti '*Route*' untuk mengatur rute berdasarkan nama aksi/metode, ada '*HttpPost'* untuk menyatakan bahwa metode ini merespons permintaan HTTP POST, selanjutnya ada '*ValidateDTO'* untuk memvalidasi DTO sebelum masuk ke metode di mana *custom attribute* ini harus diimplementasikan sesuai

kebutuhan, '*MapToApiVersion*(1)' yang mengatur versi API, serta '*AllowAnonymous'* yang mengizinkan akses tanpa autentikasi. Metode AddRefOffice menggunakan *async*/*await* untuk operasi asinkron dan memanggil metode *service* dengan menggunakan *interface* yang sudah dibuat sebelumnya. Setelah API berhasil dibuat, maka Microsoft Visual Studio dapat di *run*, dan membuka kumpulan API di *local* dengan 'Swagger' seperti Gambar 3. 15.



Gambar 3. 15 Tampilan Swagger

3.2.1.5 Pelatihan *Code Guideline*

Pada hari kelatihan kelima, materi yang diajarkan adalah tata cara penulisan *code*. Pelatihan *code guideline* ini diwajibkan untuk memastikan semua *development* software di perusahaan memahami dan mengikuti standar penulisan kode yang telah ditetapkan. Pelatihan ini bertujuan untuk meningkatkan kualitas dan konsistensi kode yang dikembangkan, serta memfasilitasi proses review kode dan kolaborasi antar tim. Selama pelatihan, peserta magang diajarkan berbagai aspek penting dari penulisan kode yang baik, termasuk konvensi penamaan yang sesuai dengan

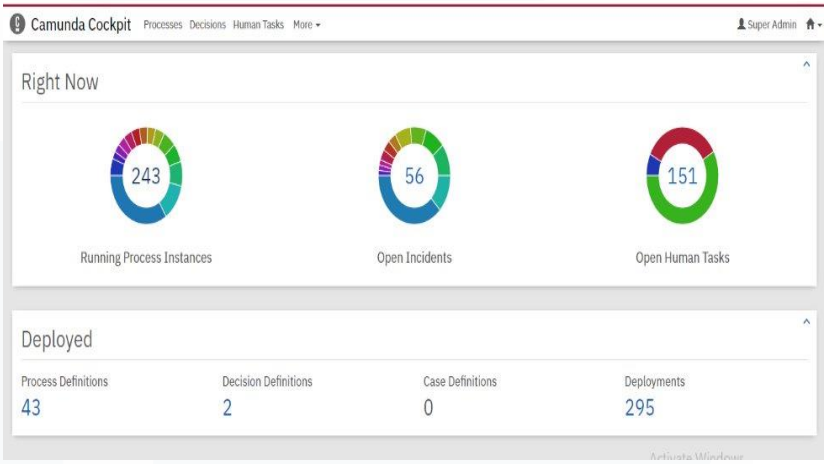
standarisasi perusahaan, struktur kode, penggunaan komentar yang efektif, dan cara menulis bahasa pemrograman yang sesuai dengan standar perusahaan. Pelatihan ini tidak hanya meningkatkan kemampuan teknis, tetapi juga melatih *skill* peserta magang dalam pengembangan perangkat lunak aplikasi web yang berkontribusi pada pemeliharaan dan skalabilitas sistem.

Selain itu, Pelatihan Code Guideline juga menyediakan sesi interaktif di mana peserta dapat menerapkan prinsip-prinsip yang telah mereka pelajari dalam latihan praktis. Sesi ini memberikan kesempatan bagi pengembang untuk mendapatkan umpan balik langsung dari instruktur, yang membantu dalam mengidentifikasi dan memperbaiki kekurangan dalam teknik pemrograman mereka. Peserta pelatihan akan diberikan contoh kode nyata dan diminta untuk mengoptimalkannya sesuai dengan pedoman yang disetujui, sehingga memperkuat pemahaman mereka tentang praktik terbaik dan meningkatkan keterampilan pemecahan masalah mereka secara keseluruhan. Ini adalah langkah penting untuk memastikan bahwa semua kode yang dikembangkan memenuhi standar tinggi yang diharapkan dalam lingkungan profesional modern.

3.2.1.6 Pelatihan *Engine Camunda*

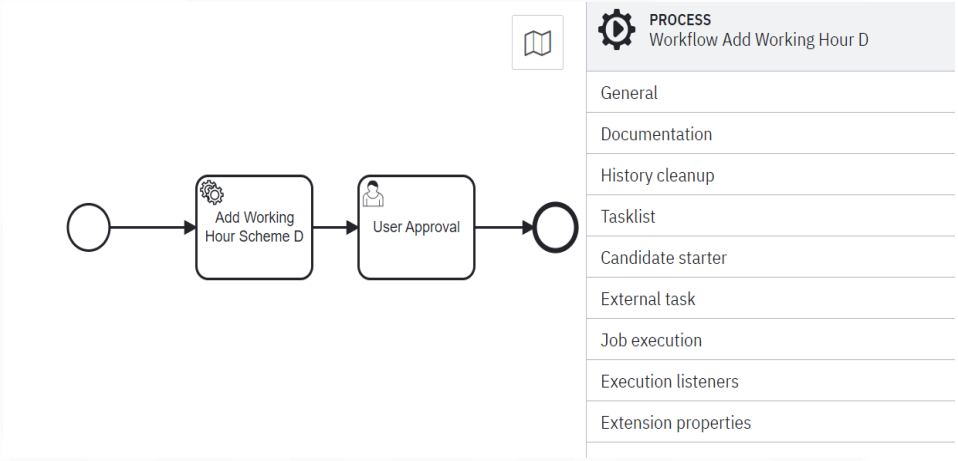
Pada hari pelatihan keenam, materi yang dipelajari adalah *engine* Camunda Modeler. Camunda Modeler merupakan aplikasi perangkat lunak yang dirancang khusus untuk memfasilitasi pemodelan proses bisnis menggunakan notasi seperti *Business Process Model and Notation* (BPMN), *Decision Model and Notation* (DMN), dan *Case Management Model and Notation* (CMMN)[11]. *Tool* ini memainkan peran penting dalam memungkinkan organisasi untuk memvisualisasikan, merancang, dan mengimplementasikan proses bisnis mereka dengan lebih efisien dan efektif. Sebagai alat pemodelan, Camunda Modeler tidak hanya mendukung pembuatan diagram yang representatif, tetapi juga menawarkan fungsionalitas untuk mengembangkan model yang dapat dieksekusi, yang

mendukung otomatisasi dan pengoptimalan proses. Berikut tampilan *dashboard* Camunda pada Gambar 3.16



Gambar 3. 16 Tampilan *Dashboard* Camunda

Berikut contoh tampilan *workflow* pada salah satu proses di Camunda Modeler seperti pada Gambar 3.17



Gambar 3. 17 Tampilan Proses Camunda

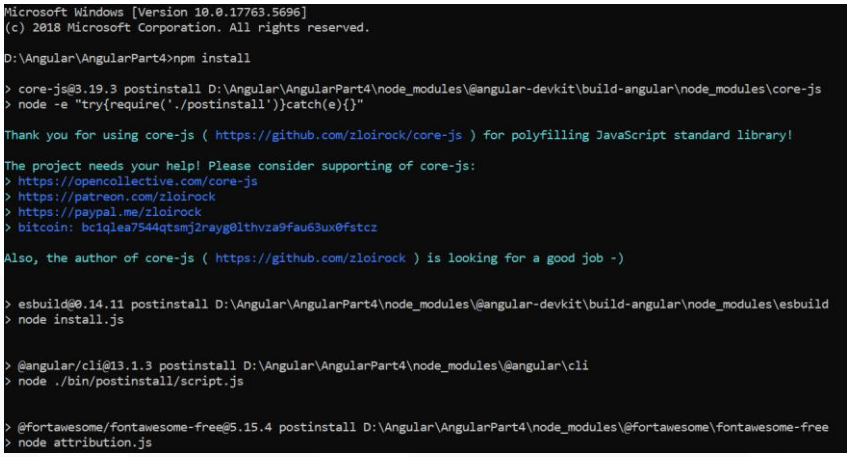
3.2.1.7 Pelatihan *Framework* Angular

Pada hari pelatihan ketujuh, mulai dipelajari pembuatan aplikasi web dari sisi *frontend* menggunakan *framework* Angular. Angular merupakan kerangka kerja berbasis CLI (*Command Line Interface*) yang dirancang untuk membangun aplikasi web yang efisien [12].

Gambar 3. 18 Logo Angular [12]

CLI ini memudahkan *developer* dengan menyediakan alat untuk menginisiasi, mengembangkan, dan mengelola kode aplikasi langsung dari terminal yang terdapat pada Visual Studio Code. Fitur utama dari Angular, yaitu *component-based*, *two-way data binding*, dan *dependency injection*, membantu dalam pembuatan aplikasi yang responsif. Angular menggunakan Bahasa TypeScript sebagai Bahasa utamanya sehingga memiliki keunggulan dalam pengelolaan kode yang besar dan kompleks. Sesi ini dilaksanakan selama tujuh hari kerja yang dibagi menjadi lima hari pelatihan Angular dan dua hari dilakukan *review* atau mengulas Kembali pembelajaran. Banyaknya komponen dan *template* yang harus dipelajari dalam pembelajaran *frontend* dengan Angular ini membuat training dilaksanakan dengan cukup lama dan intensif.

Pada hari pelatihan Angular hari pertama, instruktur menugaskan untuk melakukan *mapping framework* Angular dengan menggunakan *template* perusahaan. *Template* perusahaan dikloning dari TFS (*Team Foundation Server*) ke *platform* Visual Studio Code dan dilakukan instalasi *node modules* di terminal dengan *command* "*npm install*". Berikut contoh penerapan *install node modules* di *terminal*.



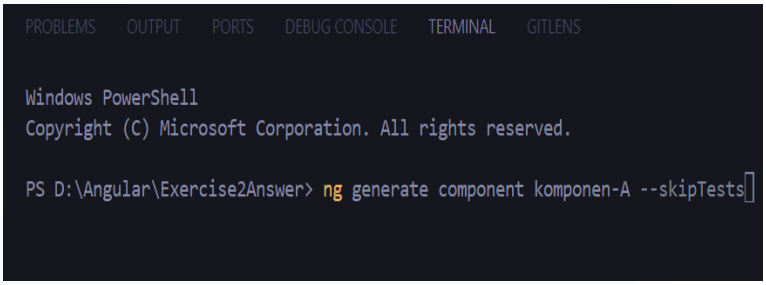
Gambar 3. 19 *Install Node Modules*

Pelatihan dimulai dari dasar-dasar pengantar Angular yang mencakup pemahaman dasar komponen dan bahasa aplikasi web Angular seperti HTML, CSS, Bootstrap, komponen dan modul Angular, *data binding*, *directives*, *custom properties* dan *event* di Angular. Setelah pemahaman dasar telah diberikan, selanjutnya pelatihan fokus pada pengembangan Angular pada tingkat yang lebih kompleks, pembahasan dilakukan lebih mendalam terhadap konsep-konsep kompleks seperti *routing*, *form-builder*, dan integrasi antara Angular dengan sisi *back-end* perusahaan yaitu .NET. Penggunaan Git juga diajarkan untuk mengelola kode proyek dengan efisien, hal ini penting dalam lingkungan *software development* modern di mana kolaborasi tim dan manajemen kode menjadi kunci keberhasilan proyek. Pelatihan ini memberikan pemahaman yang lebih dalam tentang cara membangun aplikasi *front-end* yang kompleks dan terintegrasi dengan *back-end* yang sudah ada.

Pelatihan selanjutnya yang masih mengenai Angular, diperkenalkan dengan penggunaan *library front-end* perusahaan, yang mencakup berbagai fitur seperti *UCPaging, UCView, UCAddtoTemp, UCDropDownList, dan UCAddress*. Pelatihan ini memberikan pemahaman mendalam tentang cara menggunakan fitur-fitur ini dalam proyek Angular yang sedang dikembangkan. Penggunaan *library* ini merupakan salah satu teknik standarisasi untuk mempercepat dan menyamakan kode untuk satu fitur

yang sama dalam berbagai halaman di aplikasi web. Angular merupakan kerangka kerja yang berbasis CLI, sehingga untuk menginisialisasi sesuatu, dapat memanfaatkan terminal. Berikut contoh penggunaan terminal di Angular:

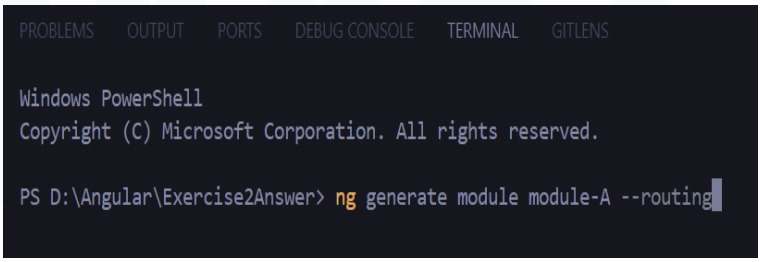
1. Membuat *Component* Baru



Gambar 3. 20 Kode Membuat *Component*

Gambar 3.20 merupakan perintah untuk membuat komponen baru dalam folder kerangka kerja Angular dengan nama komponen-A. Beberapa file yang akan dihasilkan dari perintah ini yaitu file TypeScript, HTML, CSS, dan SpecTS. Perintah '*skipTests'* berfungsi untuk tidak menghasilkan file SpecTS yang jarang digunakan sehingga lebih baik dihilangkan.

2. Membuat *Module* Baru

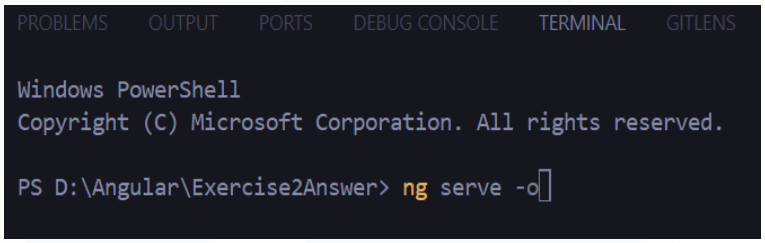


Gambar 3. 21 Kode Membuat *Module*

Gambar 3.21 merupakan perintah untuk membuat modul baru dalam folder kerangka kerja Angular dengan nama module-A. File yang akan dihasilkan dari perintah ini yaitu file TypeScript. Selain itu, file *routing* baru bernama

‘module-A-routing.module.ts’ juga akan dibuat di dalam direktori "module-A". Routing module ini akan berisi definisi rute-rute yang terkait dengan modul "module-A".

3. Menjalankan Aplikasi



Gambar 3. 22 Kode Menjalankan Aplikasi

Gambar 3.22 merupakan perintah untuk menjalankan aplikasi Angular secara local. Perintah -o merupakan akronim dari *open* yang berfungsi untuk secara otomatis membuka aplikasi tersebut di *browser default*, sehingga tidak perlu secara *manual* membuka *browser* dan mengetik alamat local.

3.2.1.8 Ujian hasil pelatihan

Setelah tahap pelatihan dan *review* hasil pelatihan dilewati, dilaksanakan ujian untuk mengetahui tingkat kemampuan para peserta magang *pasca* menjalani pelatihan intensif mengenai web aplikasi dari sisi *frontend* dan *backend*. Ujian dilaksanakan dalam jangka waktu tujuh hari kerja dengan sistem soal *take home*. Soal diberikan dalam bentuk rincian dan tampilan untuk membuat halaman aplikasi web dengan menggunakan *framework* ASP.NET dan bahasa pemrograman C# dari sisi *backend*, lalu menggunakan *framework* Angular dengan penggunaan bahasa pemrograman TypeScript dan HTML dari sisi *frontend*, serta database pgAdmin dengan menggunakan PostgreSQL. Pada ujian pelatihan, soal yang diberikan sebagai berikut:

1. Membuat halaman *Paging Office*

Beberapa ketentuan untuk membuat halaman pencarian data kantor dengan berbagai kriteria, seperti OfficeCode, ParentId, OfficeName, OfficeAddress, OfficeType, dan ActiveStatus adalah sebagai berikut: terdapat tombol *reset* untuk mengosongkan semua input pencarian dan mengatur ulang *dropdown* ke 'ALL', *button search* untuk menampilkan data kantor yang sesuai dengan kriteria pencarian, *button add* untuk navigasi ke halaman penambahan data kantor, tampilan grid yang berisi data kantor yang sesuai dengan kriteria pencarian, dengan opsi *action* untuk mengedit atau menghapus data tersebut.

2. Membuat API *Add Edit Office*

   Beberapa ketentuan untuk membuat API *add edit office* adalah sebagai berikut: dapat menampilkan formulir untuk menambah atau mengedit data kantor, terdapat beberapa kolom yang wajib diisi, seperti OfficeCode, OfficeName, OfficeType, dan lainnya, jika kosong maka diberi validasi, membuat pengaturan spesifik tergantung tipe kantor yang berupa *Head Office, Branch, Center Group* yang mempengaruhi beberapa inputan yang diperlukan, membuat ketentuan mode *edit* di mana data yang sudah ada ditampilkan dan beberapa kolom tidak dapat diubah.

3. Membuat halaman *Paging Office Member*

   Beberapa ketentuan untuk membuat API halaman pencarian *office member* adalah sebagai berikut: dapat menampilkan data *office member* untuk kantor pusat tertentu, membuat *subsection* yang berisi informasi tentang kantor pusat yang dipilih serta halaman pencarian *office member*, membuat *button add* untuk menambahkan *office member* baru, membuat tampilan grid yang menunjukkan anggota kantor yang sudah ada dengan opsi untuk menghapus.

4. Membuat API *Add Edit Office Member*

   Beberapa ketentuan untuk membuat API *add edit office* adalah sebagai berikut: dapat menampilkan formulir untuk menambah atau mengedit *office member*, membuat *subsection* yang berisi informasi tentang kantor pusat yang dipilih dan halaman pencarian *office member*, membuat tampilan *grid* untuk menampilkan office member yang belum pernah menjadi *office member* kantor pusat tersebut, membuat *grid sementara* untuk menambahkan anggota kantor baru sebelum disimpan dengan menggunakan *library addtotemp*.

3.2.2 Pengenalan Tim dan Pengenalan Proyek

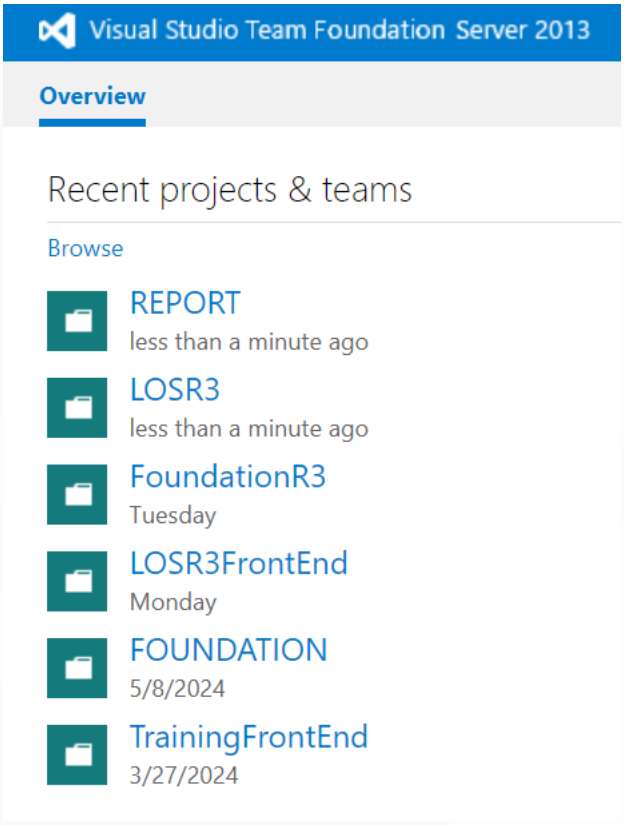
Peserta magang yang telah melewati tahap pelatihan intensif dan melaksanakan ujian, dianggap telah menyelesaikan tahap pelatihan intensif yang diberikan perusahaan, sehingga untuk pelatihan selanjutnya, dilakukan dalam proyek langsung dan masuk ke dalam tim. Pada hari pertama memasuki tim *project delivery*, mahasiswa magang diperkenalkan kepada seluruh anggota tim. Pemanggilan nama tim di PT Adicipta Inovasi Teknologi disesuaikan dengan nama perusahaan klien. Memasuki tim *project delivery*, nama perusahaan klien adalah PT XYZ, sehingga nama tim disebut dengan tim XYZ. Pada saat peserta magang memasuki tim XYZ, tim XYZ sedang berada pada fase *development*, sehingga merupakan fase yang tepat untuk mempelajari hal-hal terkait *develop* fitur untuk klien PT XYZ.

Pada fase development ini, mahasiswa magang diberikan kesempatan untuk berkontribusi langsung dalam pengembangan fitur-fitur baru untuk tim XYZ. Setiap mahasiswa magang diberi tugas spesifik yang disesuaikan dengan kemampuan dan minat mereka, serta didampingi oleh *supervisor* dari tim XYZ untuk memastikan mereka mendapatkan bimbingan yang tepat. Melalui proses ini, mahasiswa magang tidak hanya belajar mengenai teknis pengembangan perangkat lunak, tetapi juga memahami cara kerja tim dalam proyek nyata, termasuk

manajemen waktu, kolaborasi antar anggota tim, dan komunikasi dengan klien. Mahasiswa magang diharapkan mampu menunjukkan inisiatif dan kreativitas dalam menyelesaikan tugas yang diberikan, serta mengembangkan kemampuan *problem-solving* dengan mengerjakan tugas-tugas yang diberikan.

3.2.2.1 *Mapping Framework*

Pada hari pertama memasuki tim XYZ, mahasiswa diberi tugas pertama untuk melakukan *mapping framework*, yang merupakan langkah penting dalam memahami arsitektur dan alur kerja yang ada di tim XYZ. Tugas mapping framework ini melibatkan pemetaan seluruh modul yang digunakan oleh tim XYZ dalam melakukan pengembangan aplikasi web, dalam kasus ini adalah modul FOU dan LOS. *Tools* yang digunakan dalam proyek, termasuk bahasa pemrograman, *framework*, dan layanan pihak ketiga telah dipelajari pada saat pelatihan. Mahasiswa magang harus mengidentifikasi bagaimana setiap komponen berinteraksi satu sama lain, serta peran masing-masing dalam keseluruhan sistem. *Mapping framework* yang dilakukan adalah modul FOU dari sisi *front-end* dan *back-end*¸ modul LOS dari sisi *front-end* dan *back-end*, serta *back-end framework Report* menggunakan Microsoft Team Foundation Server (TFS) seperti pada Gambar 3. 23.



Gambar 3. 23 List *Framework* yang di *Mapping*

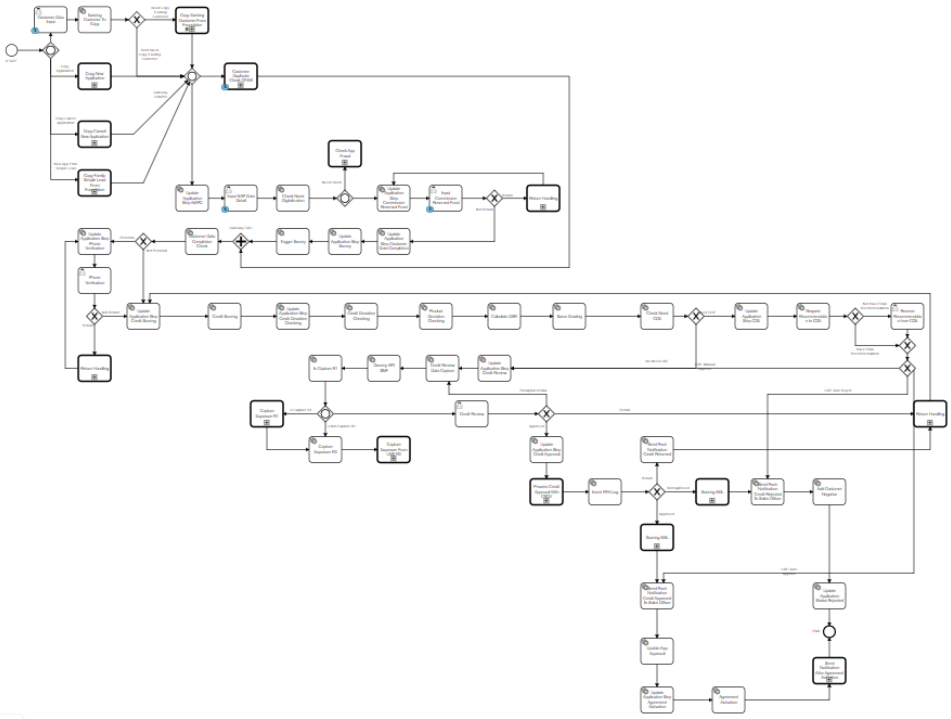
Semua *framework* akan digunakan pada saat pemberian tugas oleh *team leader*, saat mengembangkan fitur baru, saat men-debug *error*, atau saat melakukan *testing* aplikasi.

3.2.2.2 Memasukkan Data Aplikasi

Hari kedua memasuki tim XYZ, mahasiswa diberi tugas untuk memasukkan data aplikasi pada aplikasi web CONFINS. Data aplikasi yang dimaksud adalah membuat satu nomor aplikasi di modul LOS di mulai dari *log in* untuk memilih kantor saat ini, memilih *Business Line*, memasukkan dan melengkapi data *customer* baru, memasukkan dan melengkapi data anggota keluarga dari *customer* tersebut, serta memasukkan dan melengkapi data penjamin dari *customer* tersebut. Setelah data *customer* lengkap, tahapan selanjutnya adalah memasukkan data aplikasi. Tahapan pertama dalam data aplikasi adalah, *tab referantor* yang diisi hanya jika terdapat *referantor*, lalu tahapan selanjutnya memasukkan detail aplikasi seperti

sumber aplikasi, tipe bunga, skema angsuran, metode pembayaran, frekuensi pembayaran, tenor, dan lainnya, tahapan setelah detail aplikasi adalah memasukkan dan melengkapi data aset serta ada atau tidaknya aksesoris pada asset tersebut, selanjutnya memasukkan data asuransi terkait asset tersebut jika ada, lalu berikutnya memasukkan data finansial, dan mengisi subsidi jika ada subsidi yang diberikan kepada nomor aplikasi tersebut, pada halaman ini juga dilakukan perhitungan untuk rencana angsuran, tahapan berikutnya mengenai *Terms and Condition* di mana ada beberapa dokumen wajib yang harus di cek apakah *customer* sudah melengkapi, tahapan berikutnya adalah unggah dokumen.

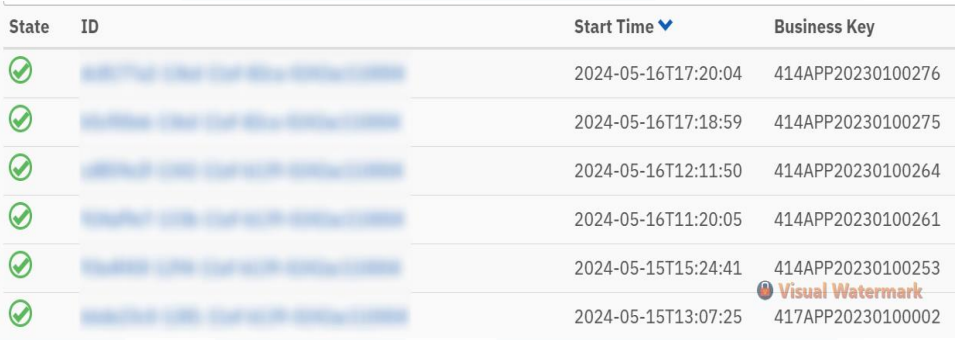
Membuat nomor aplikasi baru ini dilakukan sembari memantau *flow* di Camunda seperti di Gambar 3.24.



Gambar 3. 24 *Flow* Camunda

Tujuan dilakukan adalah agar mahasiswa magang memahami *flow* pembuatan aplikasi, misalnya Ketika baru dilakukan *input* data *customer*, maka akan terlihat lingkaran biru di bagian *Customer Data Input*. Proses

pada Camunda dapat dilihat dari *Process Instances* seperti pada Gambar 3.25.



State	ID	Start Time	Business Key
✓	[redacted]	2024-05-16T17:20:04	414APP20230100276
✓	[redacted]	2024-05-16T17:18:59	414APP20230100275
✓	[redacted]	2024-05-16T12:11:50	414APP20230100264
✓	[redacted]	2024-05-16T11:20:05	414APP20230100261
✓	[redacted]	2024-05-15T15:24:41	414APP20230100253
✓	[redacted]	2024-05-15T13:07:25	417APP20230100002

Gambar 3. 25 *Tab Process Instances* di Camunda

Pada Gambar 3.25, ID menunjukkan id *flow* dari aplikasi yang dibuat, *Start Time* menunjukkan waktu aplikasi memulai proses, *Business Key* menunjukkan nomor aplikasi yang telah dibuat.

Pembuatan aplikasi ini dilakukan selama dua hari kerja dengan mencoba berbagai menu dan *business line* yang berbeda-beda. Tujuan pembuatan aplikasi ini adalah agar mahasiswa magang memahami dan lebih mengenal letak menu dan *flow* dalam pembuatan nomor aplikasi. Sehingga untuk kedepannya, jika ada tugas untuk melakukan sesuatu, dapat lebih cepat mengerti karena sudah memiliki sedikit gambaran mengenai *flow* pengisian datanya.

3.2.3 Pembuatan *Report* dan *Document Printing*

Salah satu fitur yang diminta oleh *client* XYZ adalah menu fitur *report* dan *document printing*, di mana menu ini ada pada modul LOS. Konsep dasar mambuat *report* dan *document printing* sama, hanya ada perbedaan pada SP (*Stored Procedure*). Pembuatan *report* dan *document printing* menggunakan *framework* yang sudah dilakukan *mapping* sebelumnya yaitu *framework report*.

3.2.3.1 Membuat *Report*

Menu *Report* adalah menu dalam aplikasi yang memungkinkan user untuk melakukan *print report* yang berfungsi untuk pelaporan, hasil

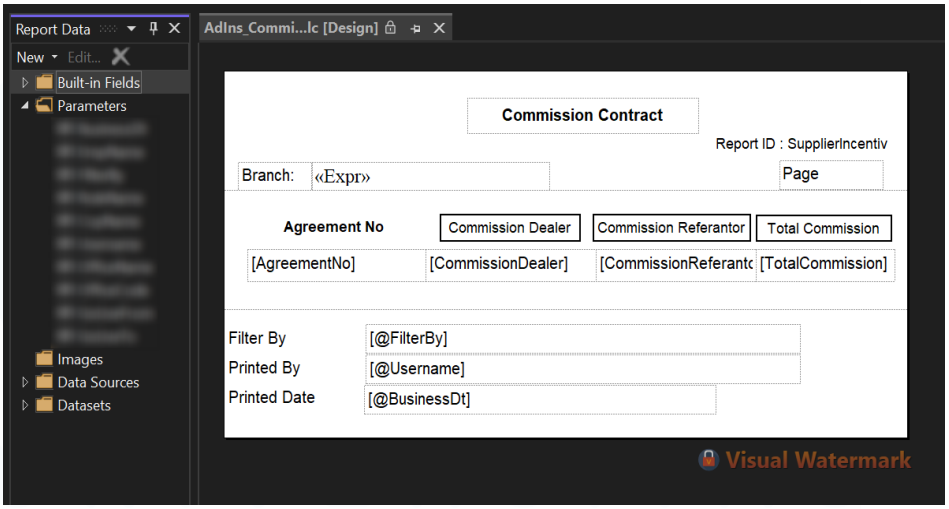
summary, dan lainnya yang biasa dalam format .pdf atau .xls. Langkah pertama dalam membuat report adalah membuat SP (*Stored Procedure*) seperti pada Gambar 3.26.

```
RAISE NOTICE '%',v_whereCond;
query_text := 'SELECT
 COALESCE(                                    AS "OfficeName",
 COALESCE(                                    S "AgreementNo",
 SUM(COALE                                    SION_AMT, 0)) AS "CommissionDealer",
 SUM(COALE                                    SION_AMT, 0)) AS "CommissionReferantor",
 SUM(COALE                                    SION_AMT, 0) +                                  , 0)) AS "TotalCommissic
 FROM AGRMNT AGR LEFT JOIN
 AGRMNT_COMMISSION_H A                                                                        T_TYPE_COD
 LEFT JOIN AGRMNT_COMM                                                                        RECIPIENT
 '|| coalesce(v_whereCond,'') ||'
 GROUP BY "OfficeName", "AgreementNo"'
;
RETURN QUERY EXECUTE query_text;
```

Visual Watermark

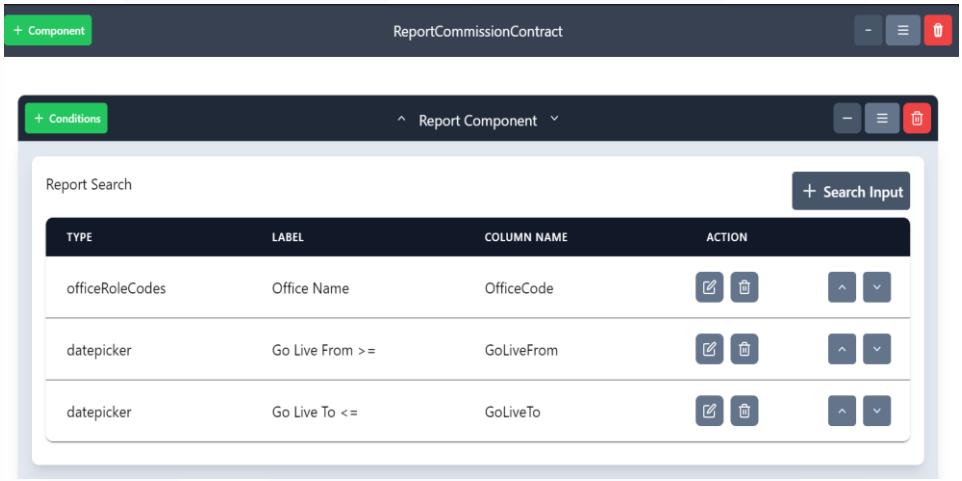
Gambar 3. 26 Kode SP Membuat *Report*

Setelah membuat SP dengan parameter-parameter yang digunakan untuk filter hasil *print* dari *report* tersebut. Langkah selanjutnya adalah membuat file .rdlc yang merupakan *extension* dalam Microsoft Visual Studio. Pembuatan file .rdlc disertai dengan pembuatan parameter wajib seperti FilterBy, OfficeName, dan lainnya agar *report* dapat di *print*. Contoh file .rdlc seperti pada Gambar 3.27.



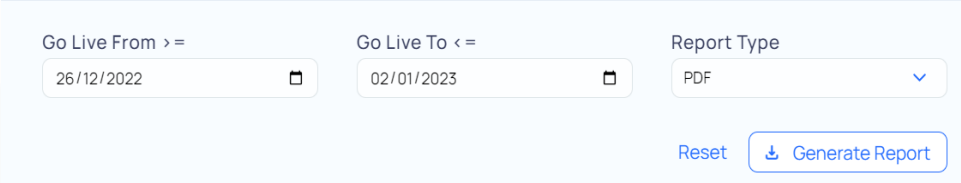
Gambar 3. 27 Tampilan File RDLC

Langkah membuat report setelah SP dan RDLC selesai adalah membuat menu untuk melakukan *print report* nya. Pembuatan menu menggunakan *framework* Angular dengan *Template Editor* (TE) seperti Gambar 3.28.



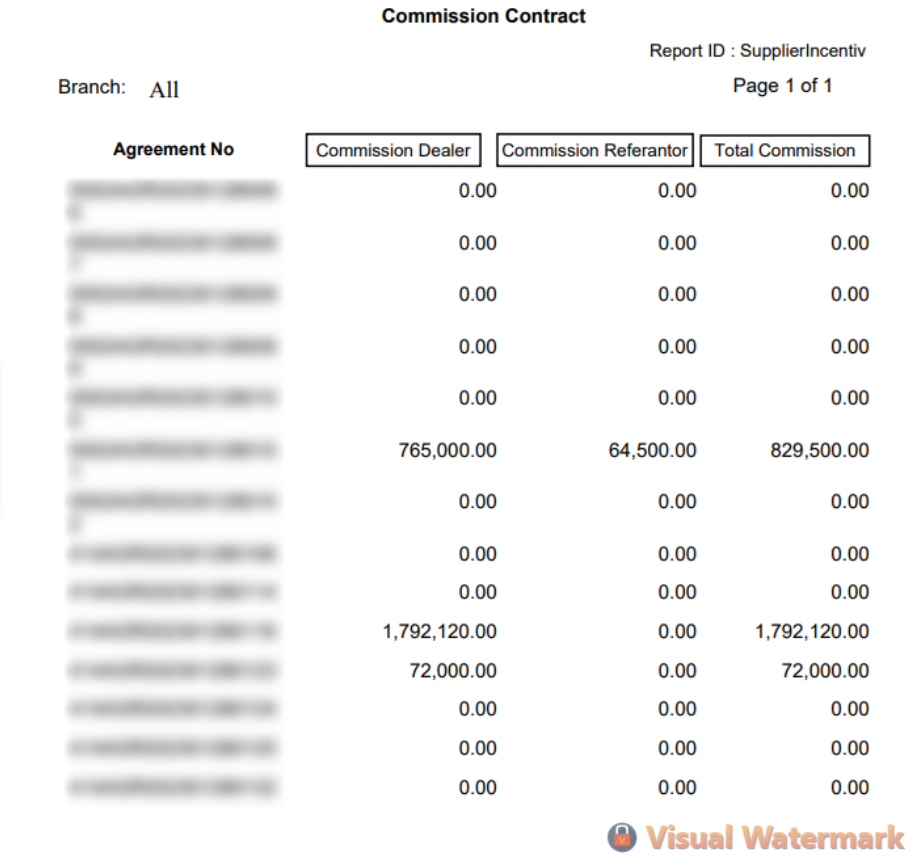
Gambar 3. 28 Tampilan *Template Editor*

Setelah TE dibuat, maka TE akan *generate* file JSON yang akan diletakkan dalam *framework* Angular dan tampilan menu akan secara otomatis seperti pada Gambar 3.29.



Gambar 3. 29 Tampilan Menu di Web

Ketika hit API *Generate Report* yang ada pada menu sesuai dengan parameter '*Go Live From*', '*Go Live To*', dan '*Report Type*' yang sudah dibuat di SP, maka tampilan *report* akan ter-*generate* dengan tampilan seperti Gambar 3.30.



Gambar 3. 30 Tampilan *Report*

3.2.3.2 Membuat *Document Printing*

Konsep pembuatan *document printing* secara garis besar sama dengan pembuatan *report*. Perbedaannya terletak di sisi *front-end* di mana pada *document printing* tidak perlu dibuatkan tampilan dengan menggunakan TE, dan hanya membuat SP dan RDLC.

3.2.4 Pembuatan *Mirorring*

Tugas selanjutnya setelah pembuatan report dan document printing adalah mirroring. Konsep Mirrorring adalah pada *environment* terdapat dua modul, yaitu FOU dan LOS dengan dua *database* berbeda, pada tabel database LOS, terdapat tabel dari *database* FOU, untuk setiap perubahan entitas atau data di tabel FOU, data dari tabel LOS tidak berubah, untuk itu dibuat *Mirroring* dari setiap tabel di

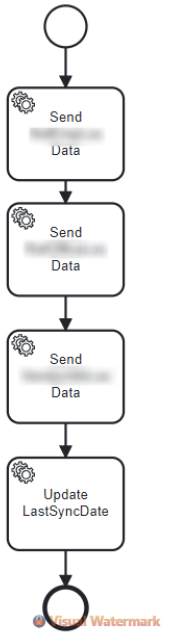
FOU agar dapat diakses di LOS. *Mirroring* dilakukan langsung di *branch Core* dengan membuat API per tabel.

3.2.4.1 Melakukan *Mirroring* Tabel

Tahapan pertama dalam membuat mirroring adalah membuat API untuk memeriksa apakah ada perubahan data di tabel FOU dengan ketentuan dalam baris tersebut dengan parameter *datetime update* lebih dari *last sync date* yang menandakan adanya perubahan. API tersebut bernama *startmirrormaster* seperti pada Gambar 3.31.

Gambar 3. 31 Kode *Start Mirror Master*

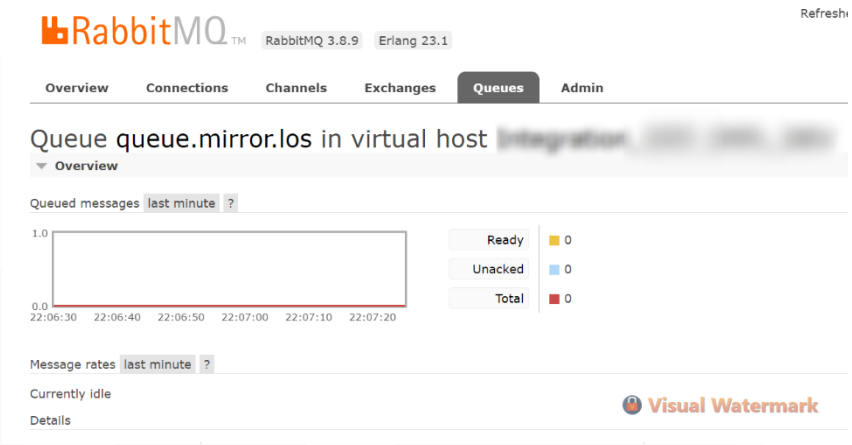
Pada Gambar 3.31, API *startmirrormaster* akan melakukan validasi, apakah terdapat data baru atau tidak, dan jika terdapat data baru atau perubahan data, maka API akan memanggil *workflow* Camunda yang akan mengirimkan data dari setiap tabel dan memperbarui *lastsyncdate* nya seperti *flow* pada Gambar 3.32



Gambar 3. 32 *Flow* Mengirimkan Data

3.2.4.2 *Testing Hit* API di *Consumer*

Setelah semua API berhasil dibuat, Langkah berikutnya adalah *testing hit api* di *Consumer* dengan menggunakan Postman. *Testing* ini bertujuan untuk memastikan bahwa API yang dibuat dapat berfungsi dengan baik dan memberikan respons yang diharapkan sesuai dengan spesifikasi yang telah ditetapkan sebelumnya. *Consumer* menggunakan RabbitMQ dengan *queue* yang akan masuk ke *queued messages*, yang akan di *consume* dan menjalankan API *receivemirrordata* seperti pada Gambar 3.33



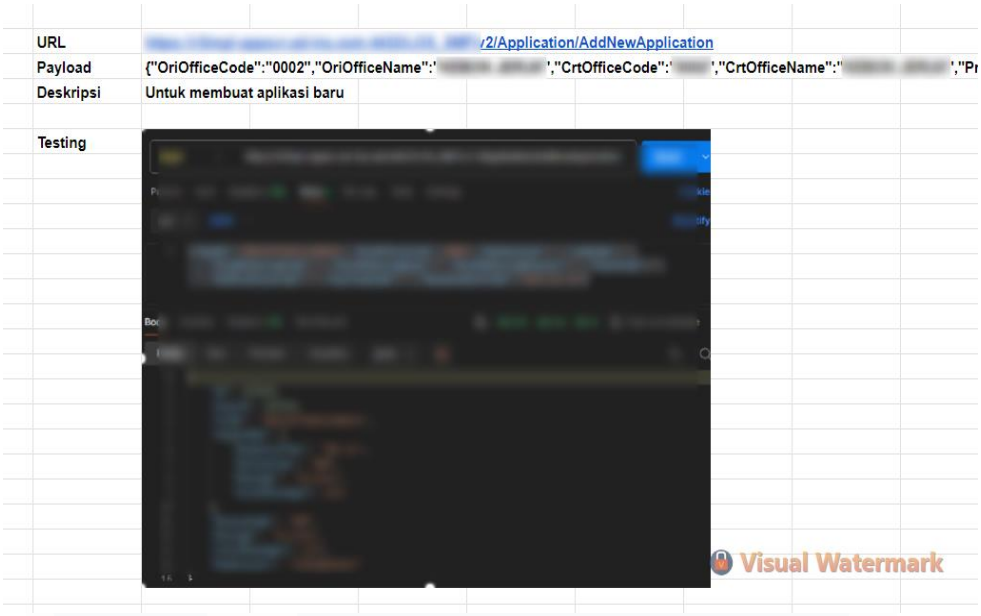
Gambar 3. 33 Tampilan RabbitMQ

3.2.5 Penyelesaian *Minor Bug*

Setelah beberapa tugas telah diselesaikan, *team leader* memberikan tugas agar mahasiswa magang mencoba untuk menyelesaikan beberapa *bug* yang terdapat di web aplikasi. Beberapa di antaranya seperti menghilangkan suatu kolom di halaman tertentu dengan ketentuan apabila tipe jenis *customer* pada halaman tersebut adalah *Referantor*. Lalu ada permasalahan muncul *error* '*remote module not available'*, dan setelah dilakukan pengecekan, isu tersebut merupakan isu dari tim *core* sehingga tidak bisa langsung diperbaiki, karena tim XYZ tidak diperbolehkan untuk mengubah kode dari *core* sehingga permasalahan diatasi dengan menyampaikan isu ke tim *core* agar dapat diatasi oleh tim *core*.

3.2.6 *Testing New Application*

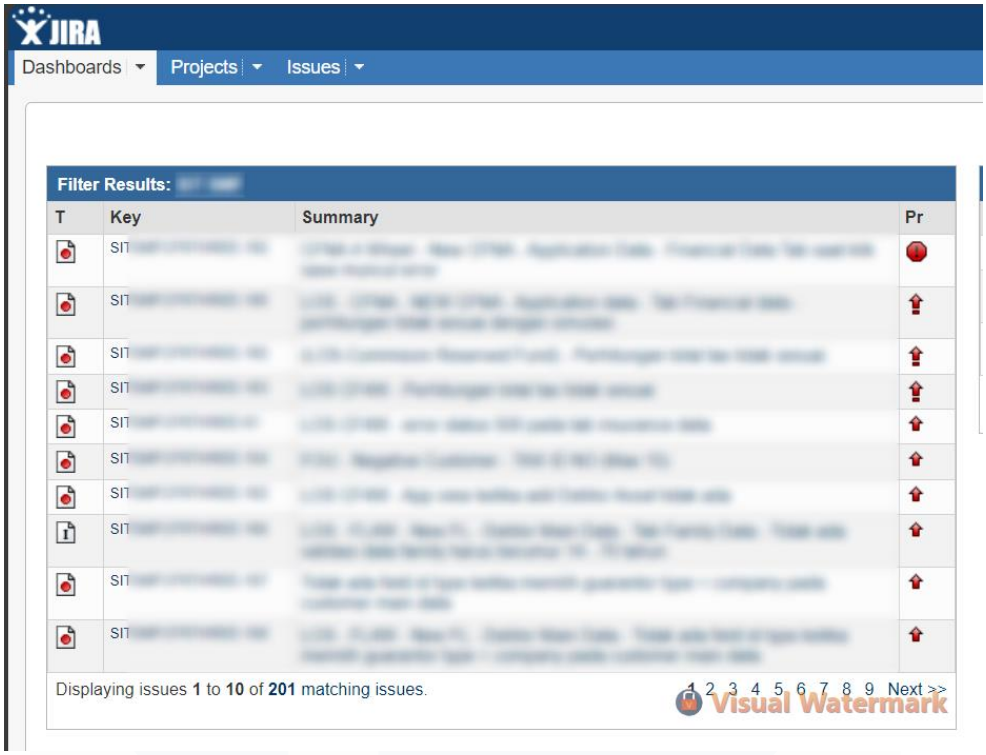
Berbagai macam tugas yang sudah diberikan *team leader* telah berhasil diselesaikan, tugas selanjutnya adalah melakukan *testing* dengan membuat nomor aplikasi baru di berbagai menu dan memasukkanya ke dalam UMT (*User Manual Testing*) sebagai dokumentasi keberhasilan hit API agar dapat diperiksa oleh tim QC dan *team leader.*



Gambar 3. 34 User Manual Testing

3.2.5 SIT (*System Integration Testing*)

System Integration Testing (SIT) adalah tahap dalam proses pengujian perangkat lunak di mana berbagai komponen atau subsistem dari suatu sistem diuji bersama untuk memastikan bahwa mereka berinteraksi dan berfungsi dengan baik satu sama lain. Tujuan utamanya adalah untuk memastikan bahwa semua bagian dari sistem tersebut berintegrasi dengan benar dan dapat berkomunikasi dengan efektif. SIT pada PT Adicipta Inovasi Teknologi dilakukan oleh tim QC (*Quality Control*). SIT dilakukan menggunakan bantuan platform JIRA, yaitu sebuah platform perangkat lunak yang umum digunakan untuk manajemen proyek, pelacakan tugas, pengelolaan siklus hidup *development* perangkat lunak, dan kolaborasi tim. Berikut tampilan JIRA dalam fase SIT tim XYZ.



Gambar 3. 35 Tampilan JIRA SIT

Alur pengajuan isu SIT oleh tim QC yaitu pertama, tim QC akan menyampaikan isu nya melalui JIRA kepada *team leader*, kemudian team leader akan mengecek isunya dan memutuskan apakah isu tersebut valid, jika valid maka *team leader* akan membagikan isu kepada anggota tim developer untuk mengatasi isu tersebut masing-masing, dan apabila isu tersebut tidak valid, maka isu akan di *reject* dan dikembalikan ke tim QC.

3.3 Kendala yang Ditemukan

Selama proses kerja magang pada PT Adicipta Inovasi Teknologi dengan peran sebagai *Web Developer*, terdapat beberapa kendala yang dialami dan menghambat kelancaran proses kerja magang. Rincian dari kendala yang ada sebagai berikut:

1. Kesulitan dalam penggunaan tools atau aplikasi yang tidak familiar
   Memasuki lingkungan kerja baru, dan memasuki tim yang mengerjakan proyek secara langsung, banyak menggunakan *tools* yang kurang familiar bahkan dalam pelatihan intensif sekalipun, sehingga terkadang menagalami kesulitan dalam penggunaanya.

2. Hambatan dalam penemuan kendala dan error dari tim *core*

   Tim XYZ yang bertugas untuk menambah fitur berdasarkan permintaan dari *client* menggunakna produk *core* yang sudah ada, sehingga seringkali menemukan *error* atau *bug* dari tim *core*, hanya saja tim XYZ tidak boleh mengubah kode dari *core* sehingga pekerjaan terhambat.
3. Hambatan dalam penemuan isu saat SIT oleh tim QC

   Tim QC yang bertugas untuk mengecek apakah produk yang dihasilkan sudah sesuai standar terkadang mengalami miskomunikasi dengan tim XYZ, sehingga hal yang seharusnya bukan isu, dijadikan isu.

3.4 Solusi atas Kendala yang Ditemukan

Selama menjalani magang di PT Adicipta Inovasi Teknologi sebagai Web Developer, beberapa kendala menghambat kelancaran proses kerja. Hal tersebut dapat diatasi dengan solusi berupa:

1. Membaca dokumentasi dan bertanya

   Beberapa penggunaan *tools* dan aplikasi lainnya yang kurang familiar atau tidak pernah digunakan, dapat dipelajari cara menggunakan atau penerapannya dengan memanfaatkan berbagai *resource* di internet mulai dari dokumentasi, komunitas *developer*, dan banyak lainnya. Peserta magang juga bisa bertanya langsung ke atasan masing-masing yang sudah familiar dengan penggunaan *tools* tersebut.
2. Menyampaikan dan memperjelas kendala secara detail

   Salah satu kendala utama adalah hambatan dalam penemuan kendala dan *error* dari tim *core*. Untuk mengatasi hal ini, perlu memastikan bahwa semua error dan kendala yang ditemukan didokumentasikan dengan jelas dan rinci sehingga memudahkan tim *core* untuk melakukan pelacakan dan penyelesaian masalah.
3. Memperjelas komunikasi dengan tim QC

   Kendala komunikasi dengan tim QC dapat diatasi dengan menetapkan kriteria pengujian yang standar dan disetujui oleh kedua tim, sehingga dapat mengurangi miskomunikasi dan memperjelas standar yang

diharapkan. Selain itu dapat dibuatkan *group chat* untuk saling bertanya dalam penyampaian isu sehingga tidak ada miskomunikasi dalam pengerjaannya.