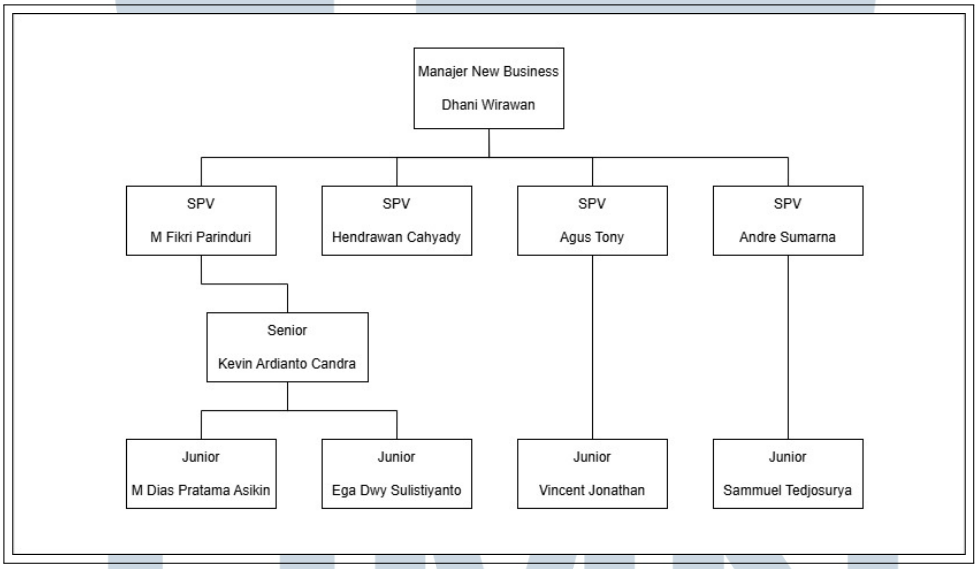
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Program magang dilaksanakan pada periode 1 Agustus 2025 hingga 25 November 2025 di PT Prima Solusi Computindo, dengan penempatan pada Tim *New Business*. Selama pelaksanaan magang, kegiatan berada di bawah bimbingan dan pengawasan langsung Andre Sumarna selaku *Supervisor*. Struktur organisasi Tim *New Business* ditampilkan pada Gambar 3.1.



Gambar 3.1. Struktur organisasi *Team New Business*

Dalam pelaksanaan program magang, penulis ditempatkan sebagai *Web Developer Intern* dengan fokus pada peran *Fullstack Developer*. Selama menjalankan tugas, penulis berkolaborasi secara aktif dengan *Supervisor* dan tim *Junior*, serta mengikuti prosedur kerja perusahaan yang memanfaatkan aplikasi Discord sebagai sarana komunikasi dan koordinasi.

3.2 Tugas yang Dilakukan

Selama menjalani program magang di PT Prima Solusi Computindo, tanggung jawab yang diberikan adalah pengembangan Aplikasi *Blackbird*. *Blackbird* adalah sebuah *platform* berbasis *web* yang digunakan untuk mendukung kegiatan investasi, serta mencatat transaksi investasi yang kemudian akan

menghasilkan sebuah jurnal. *Blackbird* digunakan oleh tim *back office* perusahaan dalam mendukung kegiatan investasi. Tugas yang dilakukan setiap minggu dapat dilihat pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Pengenalan *jobdesk*, sistem kerja, dan *project* yang akan dikerjakan.
2	Pemahaman tahap awal terhadap sistem *project* seperti bahasa yang digunakan, *flow web*, dan fungsi web.
3	Pembuatan design proposal menu CRUD
4	Pembuatan komponen-komponen data dan backend untuk menu CRUD
5	Pembuatan backend menu CRUD
6	Perapihan, pengecekan, dan perbaikan backend menu CRUD
7	Pembuatan UI menu CRUD dan testing menu *final*
8	Perancangan dan pembuatan design proposal menu *Fund Availability Projection*
9	Persiapan komponen-komponen data dan backend untuk menu *Fund Availability Projection*
10	Pembuatan backend menu *Fund Availability Projection*
11	Penyempurnaan backend menu *Fund Availability Projection*
12	Pengecekan, perapihan, dan perbaikan backend menu *Fund Availability Projection*
13	Pembuatan UI menu *Fund Availability Projection*
14	Pembuatan *logic frontend* menu *Fund Availability Projection*
15	Penyesuaian tambahan fitur menu *Fund Availability Projection*
16	Pengecekan dan perbaikan terakhir menu *Fund Availability Projection*

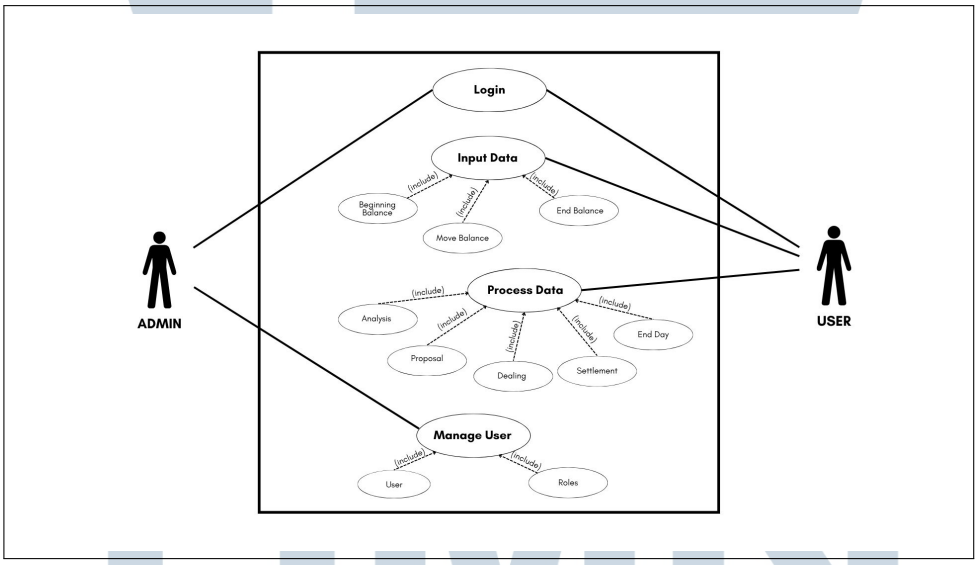
3.3 Uraian Pelaksanaan Magang

Selama kegiatan magang ini, tanggung jawab yang diberikan adalah mengembangkan Aplikasi yang bernama *Blackbird*. Aplikasi *Blackbird* dibuat menggunakan platform Dotnet. Untuk sisi frontend, aplikasi *Blackbird* menggunakan html dan javascript sebagai bahasa pemrograman, Bootstrap dan

Metronic sebagai framework, beserta kendo sebagai library js. Untuk sisi backend, menggunakan CSharp sebagai bahasa pemrograman dan untuk data sendiri menggunakan SSMS (SQL Server Management System)

A Usecase Diagram Blackbird

Use Case Diagram Blackbird yang ditampilkan pada Gambar 3.2 menggambarkan alur kerja sistem pada aplikasi web *Blackbird*. Pada sistem tersebut terdapat dua aktor utama, yaitu *Admin* dan *user*. Aktor *Admin* memiliki hak akses untuk melakukan *login* menggunakan akun administrator serta mengelola *Management User*. Melalui fitur *Management User*, *Admin* dapat melakukan pengaturan terhadap data *user* serta penetapan *Roles* yang sesuai.



Gambar 3.2. Usecase Diagram *Blackbird*

Sementara itu, aktor *user* memiliki akses untuk melakukan *input* data yang terbagi ke dalam tiga kategori, yaitu *beginning balance*, *move balance*, dan *end balance*. Setelah proses *input* data dilakukan, *user* dapat menjalankan proses pengolahan data untuk menghasilkan *output* berupa *Analysis*, *Proposal Dealing*, *Settlement*, dan *End day*.

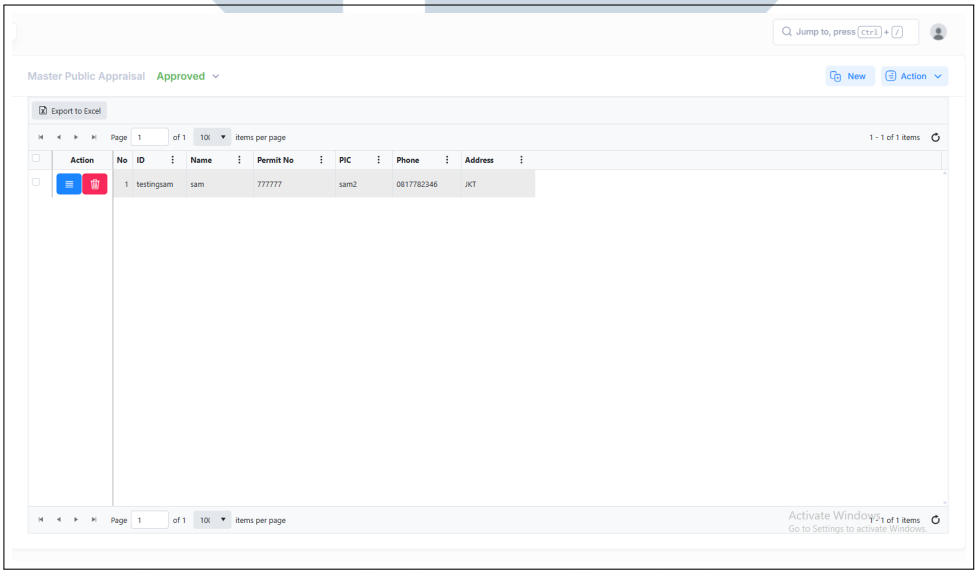
B Task yang dikerjakan

Terdapat dua menu yang dikembangkan pada Aplikasi *Blackbird*, yaitu Menu CRUD dan Menu *Fund Availability Projection*. Kedua menu ini memiliki perbedaan disisi *flow form* ketika ingin mengolah data.

B.1 Pembuatan Menu Crud

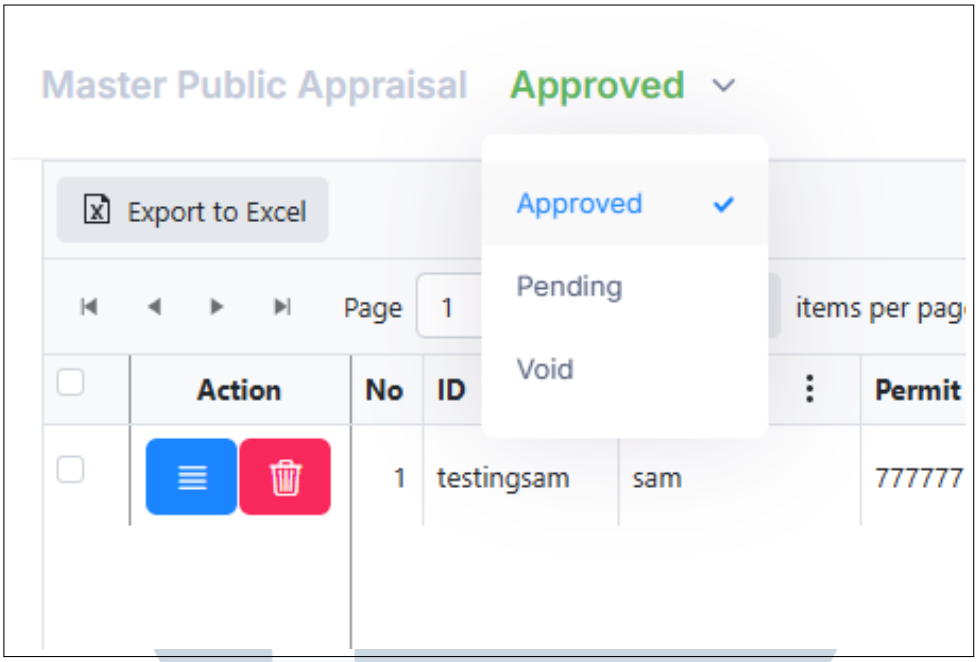
Menu CRUD terdiri dari dua halaman, halaman index yaitu halaman utama pada menu CRUD, dan halaman detail yaitu halaman yang menampilan data detail. Masing-masing halaman memiliki *behavior* dan kegunaannya masing-masing, berikut penjelasannya.

Gambar 3.3 memperlihatkan tampilan menu CRUD. Pada halaman Menu ini terdapat beberapa komponen seperti nama menu, status data, tombol *create new*, tombol *action*, dan *grid index*. Grid berisi data yang telah dibuat oleh pengguna berdasarkan status yang dimiliki oleh data tersebut. Kolom yang ditampilkan adalah. Lalu pada sisi kanan atas terdapat tombil *create new* yang ketika ditekan akan mengarahkan pengguna ke halaman detail. Disamping tombol create new juga terdapat tombol *action* yang berisi tombol *Approve* dan tombol reject. Disisi header grid juga terdapat fitur dari kendogrid yaitu *export to excel* yang digunakan untuk memindahkan data *copy* dari grid ke excel.



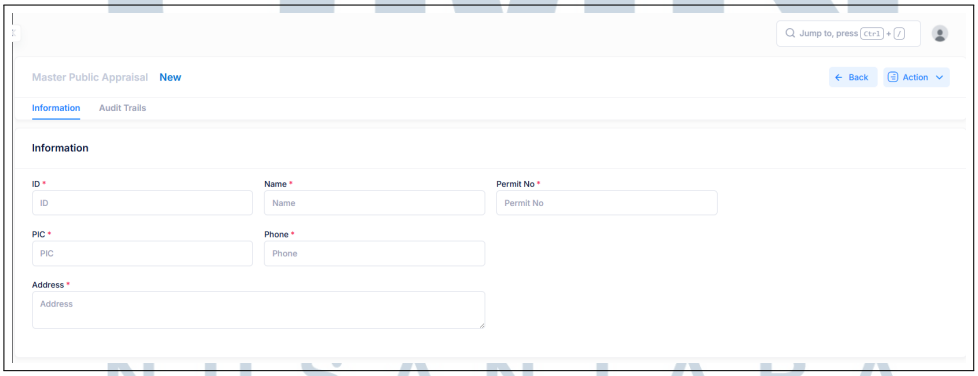
Gambar 3.3. Tampilan Menu Crud

Pada Gambar 3.4 menampilkan Komponen status data yang berfungsi untuk mengubah data yang ditampilkan berdasarkan status yang dimiliki pada data tersebut. Status data terdiri dari tiga status yaitu status *approved*, status *pending*, dan status *void*.



Gambar 3.4. Tampilan Status Data

Gambar 3.5 merupakan tampilan halaman ketika menekan tombol *new* pada halaman index. Pada halaman ini pengguna dapat mengisi tab information yang dimana terdapat beberapa field yang dapat diisi dengan *freetext*. Disisi kanan atas terdapat tombol *back* yang akan mengembalikan tampilan pengguna ke halaman index, dan tombol *action* yang berisi *option save and exit atau save and new*. Ketika tombol *save and exit*, data akan disimpan dan mengembalikan pengguna ke halaman index, namun apabila tombol *save and new*, data akan disimpan dan mengembalikan pengguna ke halaman *create new*.

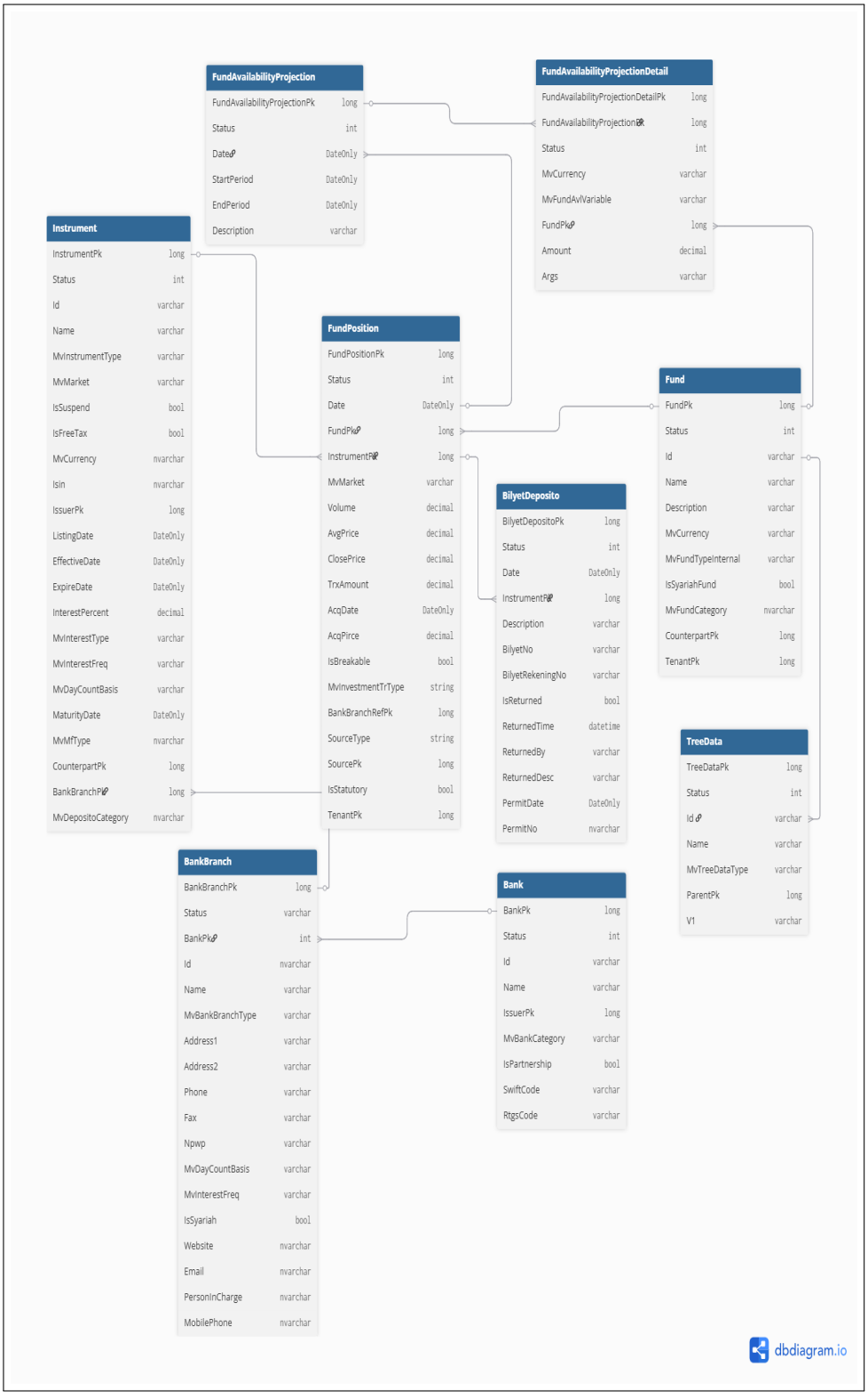


Gambar 3.5. Tampilan *Create New*

B.2 Pembuatan Menu Fund Availability Projection

Menu *Fund Availability Projection* merupakan menu yang berfungsi untuk menghitung proyeksi dana dari parameter yang diambil dan transaksi/dana yang dipilih. Menu ini dibuat dengan platform .NET beserta Framework/Library Jquery, Bootstrap, KendoGrid, Metronic, SSMS (*Sql Server Management System*) dan bahasa pemrograman HTML, Javascript, CSharp, OOP. Menu ini terdiri dari *page add*, *page index* dan *page detail* yang memiliki beberapa fitur seperti *filter index, create new header, create new detail, update data, approve data, void data, show header, show detail*, dan *draft*. Berikut struktur diagram database menu FundAvailabilityProjection.

Struktur basis data pada Gambar 3.6 untuk modul *Fund Availability Projection* dibangun menggunakan sejumlah tabel yang saling terhubung untuk memastikan konsistensi, keterlacakan, dan keakuratan data dalam proses perhitungan proyeksi dana. Tabel inti dari desain ini adalah FundAvailabilityProjection, yang menyimpan satu proyeksi untuk tanggal tertentu. Di dalamnya tercatat status data, tanggal proyeksi, periode perhitungan yang digunakan (*StartPeriod* dan *EndPeriod*), serta deskripsi yang memberi konteks terhadap proyeksi tersebut.



Gambar 3.6. Database Diagram Menu *Fund Availability Projection*

Rincian setiap proyeksi disimpan pada FundAvailabilityProjectionDetail, yang merujuk langsung ke entitas induknya melalui FundAvailabilityProjectionPk.

Tabel ini mencakup atribut seperti *MvCurrency*, *MvFundAvlVariable*, *Amount*, referensi ke *Fund* yang digunakan serta kolom Args atau *Arguments* yang berisi InstrumentPk atau barang yang akan diolah nantinya.

Data terkait *fund* yang menjadi acuan utama perhitungan berada dalam tabel *Fund*, yang menyimpan identitas lengkap setiap produk investasi seperti ID, nama, deskripsi, mata uang fund, kategori internal, informasi apakah fund tersebut merupakan Syariah *Fund*, hingga hubungan ke pihak lain seperti *Counterpart* dan *Tenant*. Relasi antara FundAvailabilityProjectionDetail dan *Fund* memastikan bahwa setiap komponen proyeksi merujuk pada *fund* yang terdaftar dan valid dalam sistem.

Untuk mendukung proses perhitungan berbasis data aktual, tabel FundPosition digunakan sebagai penyimpan seluruh posisi fund pada tanggal tertentu. Tabel ini berisi informasi seperti volume, harga rata-rata, harga penutupan, nilai transaksi, tanggal akuisisi, serta atribut tambahan seperti IsBreakable, MvInvestmentTrType, hingga sumber data (SourceType dan SourcePk). Relasi FundAvailabilityProjectionDate dan FundPositionDate memastikan bahwa proyeksi selalu dihitung berdasarkan posisi yang relevan pada tanggal proyeksi tersebut. Selain itu, FundPosition terhubung langsung ke Fund dan Instrument, sehingga setiap posisi dapat ditelusuri hingga ke instrumen spesifik yang membentuknya.

Karakteristik instrumen yang digunakan dalam posisi posisi fund dicatat dalam tabel Instrument. Di dalamnya terdapat atribut identitas instrumen, klasifikasi pasar, status suspend, mata uang, ISIN, tanggal-tanggal penting seperti ListingDate, EffectiveDate, dan MaturityDate, hingga parameter perhitungan bunga seperti InterestPercent, MvInterestType, dan MvDayCountBasis. Setiap instrumen dapat terhubung ke satu cabang bank melalui BankBranchPk, terutama untuk instrumen yang berupa produk perbankan seperti deposito.

Informasi lebih detail mengenai cabang bank tersimpan pada tabel BankBranch, yang menyimpan ID cabang, nama, alamat, nomor kontak, kategori cabang, serta data administratif lainnya seperti NPWP, frekuensi bunga, hingga PIC cabang. Cabang bank kemudian dirujuk ke entitas bank induknya melalui tabel Bank, yang menyimpan informasi lembaga keuangan seperti ID bank, nama, kategori bank, status kemitraan, serta kode transaksi seperti SWIFT dan RTGS. Rangkaian relasi Instrument ke BankBranch lalu ke Bank memungkinkan sistem mengetahui sumber atau penerbit instrumen secara lengkap.

Untuk instrumen deposito dalam bentuk fisik, tabel BilyetDeposito digunakan

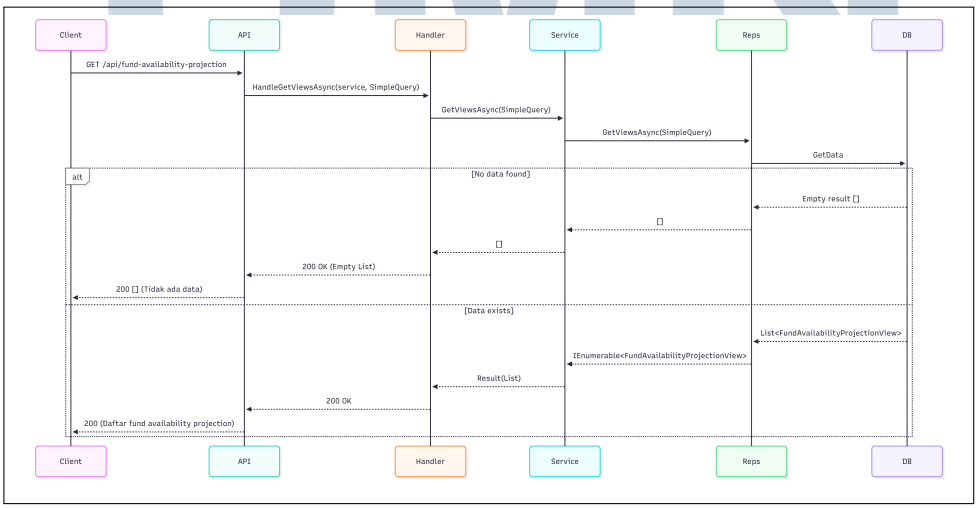
sebagai penyimpanan data bilyet, termasuk nomor bilyet, nomor rekening, status pengembalian, waktu pengembalian, dan data administratif seperti nomor izin (PermitNo) dan tanggal izin (PermitDate). Relasinya dengan FundPosition melalui InstrumentPk memastikan bahwa setiap bilyet selalu terhubung dengan instrumen deposito yang tercatat dalam posisi.

Terakhir, tabel TreeData berfungsi sebagai struktur referensi hierarkis yang dapat digunakan untuk mengelompokkan fund atau mengaitkan fund dengan kategori tertentu. Hubungannya dengan Fund melalui TreeDataId dan FundId memungkinkan pemetaan fund ke dalam struktur klasifikasi internal yang digunakan sistem.

Secara keseluruhan, seluruh relasi ini membentuk arsitektur data yang terintegrasi, mulai dari proyeksi, detail proyeksi, fund, posisi, instrumen, cabang bank, entitas bank, hingga referensi hierarkis. Dengan struktur seperti ini, proses perhitungan *Fund Availability Projection* dapat dilakukan secara akurat, valid, dan mudah ditelusuri sehingga mendukung keandalan sistem dalam pengelolaan data investasi. Berikut adalah rincian API menu FundAvailabilityProjection.

1. Alur dan Spesifikasi Api FundAvailabilityProjectionList

Sequence Diagram yang ditampilkan pada Gambar 3.7 menjelaskan alur proses pengambilan data *Fund Availability Projection* pada sistem yang berbasis API. Proses diawali ketika *pengguna* mengirimkan permintaan *HTTP GET* ke *endpoint* /api/fund-availability-projection. Permintaan tersebut diterima oleh API dan selanjutnya diteruskan ke *Handler* untuk diproses.



Gambar 3.7. Sequence Diagram *Grid Index Fund Availability Projection*

Pada tahap berikutnya, *Handler* mengeksekusi *method HandleGetViewsAsync(service, SimpleQuery)* yang berfungsi untuk menangani logika permintaan. Kemudian, *Handler* memanggil *method GetViewsAsync(SimpleQuery)* pada *Service* sebagai penghubung antara *Handler* dan *Reps*. Selanjutnya, *Service* melakukan komunikasi dengan *Repository* (*Reps*) guna mengambil data yang tersimpan pada *Database*.

Setelah data berhasil diperoleh dalam bentuk *List(FundAvailabilityProjectionView)*, hasil tersebut dikembalikan secara berurutan melalui *Service* dan *Handler* hingga kembali ke API. Terakhir, sistem mengirimkan respons *HTTP 200 (OK)* kepada pengguna yang berisi daftar data *Fund Availability Projection* sesuai dengan permintaan.

Tabel 3.2 menjelaskan API yang digunakan untuk menampilkan data *Grid Index*, yaitu *FundAvailabilityProjectionGetList*. API ini berfungsi untuk mengambil daftar data dari *database* dengan menggunakan *method GET* melalui *endpoint* /api/fund-availability-projection/. Parameter yang digunakan dalam pemanggilan API ini meliputi *Status*, *DateFrom*, dan *DateTo*.

Tabel 3.2. Spesifikasi Api *FundAvailabilityProjectionList*

Nama API	FundAvailabilityProjectionGetList
Method	GET
Endpoint Path	/api/fund-availability-projection/

Request Body/Params	`SimpleQuery : IdQuery`
	{
	`DateFrom : DateOnly`
	`DateTo: DateOnly`
	`Status: int`
	}

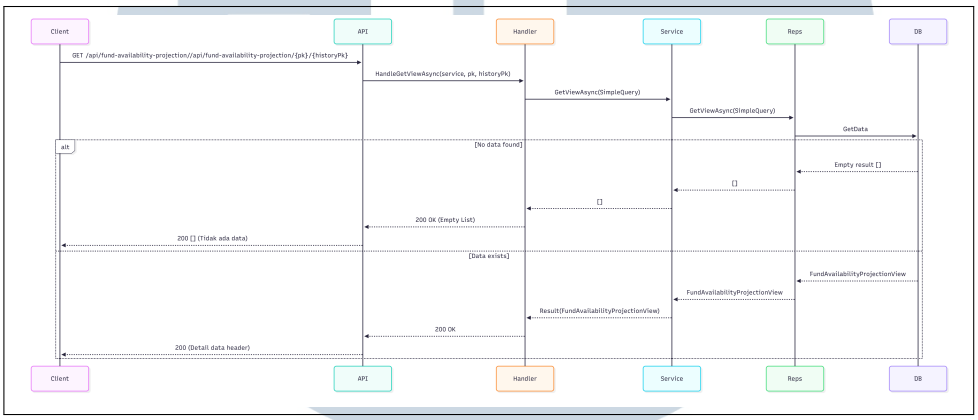
Response	*IEnumerable(FundAvailabilityProjectionView)* - (Daftar *fund availability projection*)
Permission Required	*fund-availability-projection:read*
Status Codes	200 OK

Response yang dihasilkan berupa *enumerable* dari objek *FundAvailabilityProjectionView*. Untuk dapat mengakses API tersebut, pengguna

harus memiliki *permission fund-availability-projection:read*. Adapun *status code* yang dapat dikembalikan oleh API ini adalah 200 OK.

2. Alur dan Spesifikasi Api FundAvailabilityProjectionGet

Sequence Diagram pada Gambar 3.8 menggambarkan alur sistem dalam menampilkan detail data header *Fund Availability Projection* berdasarkan parameter Pk dan HistoryPk. Proses dimulai ketika pengguna mengirimkan permintaan *GET* ke endpoint /api/fund-availability-projection/pk/historyPk.



Gambar 3.8. Sequence Diagram *Page Detail Header*

Permintaan tersebut diterima oleh lapisan API, yang kemudian meneruskannya ke Handler melalui metode HandleGetViewAsync dengan membawa parameter pk dan historyPk. Handler bertugas sebagai pengendali awal alur permintaan dan meneruskan permintaan tersebut ke lapisan Service dengan memanggil metode GetViewAsync menggunakan objek SimpleQuery.

Pada lapisan Service, sistem melanjutkan proses dengan memanggil Repository (Reps) untuk mengambil data detail header dari Database. Repository kemudian menjalankan proses pengambilan data melalui operasi GetData ke database.

Selanjutnya, terdapat dua kemungkinan kondisi yang digambarkan menggunakan blok alternative (alt) pada sequence diagram.

Pada kondisi pertama, yaitu ketika data tidak ditemukan, database mengembalikan hasil kosong (empty result). Hasil tersebut diteruskan secara berurutan dari repository ke service dan kemudian ke handler dalam bentuk koleksi kosong. Handler selanjutnya mengembalikan respons ke API dengan status 200 OK disertai data kosong. API kemudian meneruskan respons tersebut ke pengguna

dengan informasi bahwa data tidak tersedia, tanpa dianggap sebagai kesalahan sistem.

Pada kondisi kedua, yaitu ketika data ditemukan, database mengembalikan objek FundAvailabilityProjectionView ke repository. Data tersebut diteruskan ke service dan kemudian dikemas oleh handler sebagai hasil yang valid. API selanjutnya mengirimkan respons ke pengguna dengan status 200 OK beserta detail data header yang diminta.

Dengan alur ini, sistem secara eksplisit menangani kondisi data kosong dan data tersedia tanpa menggunakan status kesalahan seperti 404 Not Found. Pendekatan ini memastikan bahwa permintaan tetap dianggap valid selama parameter yang dikirim benar.

Tabel 3.3 menjelaskan API yang digunakan untuk mengambil data detail pada bagian header, yaitu *FundAvailabilityProjectionGet*. API ini berfungsi untuk memperoleh detail data header berdasarkan *primary key* (pk) dan *history key* (*historyPK*). Pemanggilan API dilakukan menggunakan *method GET* melalui *endpoint* /api/fund-availability-projection/pk/historyPK, dengan parameter yang diperlukan berupa pk dan *historyPK*.

Tabel 3.3. Spesifikasi Api *FundAvailabilityProjectionGet*

Nama API	FundAvailabilityProjectionGet
Method	GET
Endpoint Path	/api/fund-availability-projection/{pk}/{historyPk}

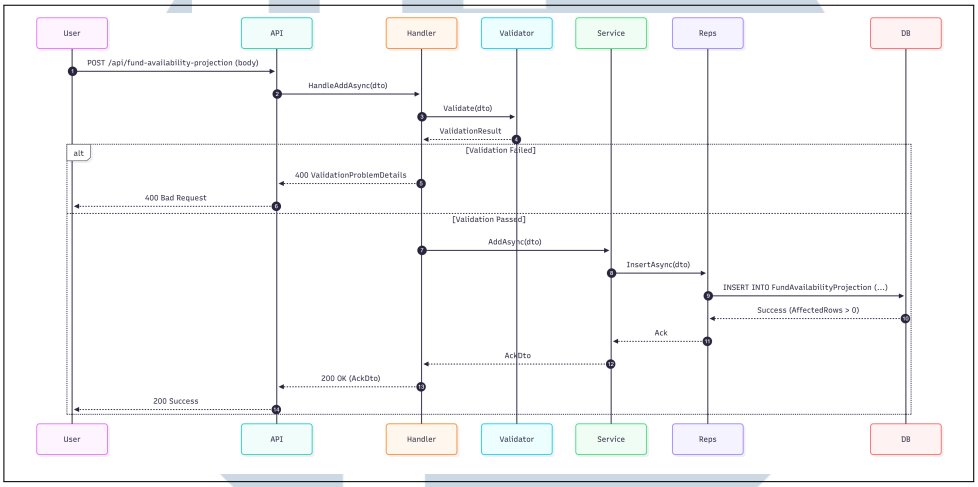
Request Body/Params	{
	`Pk: long`
	`HistoryPk: long`
	}

Response	FundAvailabilityProjectionView (Detail header fund availability projection)
Permission Required	fund-availability-projection:read
Status Codes	200 OK

Response yang dihasilkan berupa objek *FundAvailabilityProjectionView* yang memuat informasi detail data header. Untuk dapat mengakses API tersebut, pengguna harus memiliki *permission fund-availability-projection:read*. Adapun *status code* yang dikembalikan oleh API ini adalah 200 OK.

3. Alur dan Spesifikasi Api FundAvailabilityProjectionAdd

Sequence diagram pada Gambar 3.9 menggambarkan alur proses penambahan (add) data *Fund Availability Projection* mulai dari pengguna mengirimkan permintaan hingga data berhasil disimpan ke database, termasuk mekanisme validasi data.



Gambar 3.9. Sequence Diagram *Add Data Fund Availability Projection*

Proses diawali ketika pengguna mengirimkan permintaan POST ke endpoint /api/fund-availability-projection dengan membawa request body berupa DTO. Permintaan tersebut diterima oleh lapisan API, yang kemudian meneruskannya ke Handler melalui metode HandleAddAsync(dto).

Setelah menerima permintaan, Handler tidak langsung memproses penyimpanan data, melainkan terlebih dahulu memanggil Validator untuk melakukan proses Validate(dto). Validator melakukan pemeriksaan terhadap kelengkapan, format, dan aturan bisnis dasar dari data yang dikirimkan. Hasil dari proses ini dikembalikan ke handler dalam bentuk ValidationResult.

Pada tahap ini terdapat dua kemungkinan alur yang ditunjukkan dengan blok alternative (alt). Pada kondisi validasi gagal, handler langsung menghentikan proses dan mengembalikan respons berupa ValidationProblemDetails. API kemudian meneruskan respons tersebut ke pengguna dengan status 400 Bad Request, menandakan bahwa data yang dikirimkan tidak memenuhi aturan validasi.

Sebaliknya, pada kondisi validasi berhasil, handler melanjutkan proses dengan memanggil metode AddAsync(dto) pada lapisan Service. Service bertugas mengatur logika bisnis dan meneruskan proses penyimpanan ke Repository (Reps) melalui metode InsertAsync(dto).

Repository kemudian menjalankan perintah INSERT ke Database untuk menyimpan data *Fund Availability Projection*. Apabila proses penyimpanan berhasil, database mengembalikan status sukses yang ditandai dengan AffectedRows kurang dari 0. Informasi keberhasilan ini diteruskan kembali secara berurutan dari repository ke service dalam bentuk Ack, kemudian dari service ke handler sebagai AckDto.

Handler selanjutnya mengirimkan hasil tersebut ke API, yang kemudian mengembalikan respons akhir kepada pengguna dengan status 200 OK beserta AckDto sebagai tanda bahwa proses penambahan data berhasil dilakukan.

Dengan alur ini, sequence diagram menegaskan bahwa proses validasi dilakukan sebelum interaksi dengan database, sehingga sistem mampu menjaga konsistensi data dan mencegah penyimpanan data yang tidak valid. Selain itu, pemisahan peran antara API, handler, validator, service, dan repository membuat arsitektur sistem lebih terstruktur, mudah dipelihara, dan sesuai dengan prinsip separation of concerns.

Tabel 3.4 menjelaskan API yang digunakan untuk menambahkan data *Fund Availability Projection*, yaitu *FundAvailabilityProjectionAdd*. API ini berfungsi untuk menyimpan data baru dengan memanfaatkan *method POST* melalui *endpoint* /api/fund-availability-projection/. Data yang dikirimkan pada *request body* harus disusun dalam format *AddFundAvailabilityProjectionDto*.

Tabel 3.4. Spesifikasi Api *FundAvailabilityProjectionAdd*

Nama API	FundAvailabilityProjectionAdd
Method	POST
Endpoint Path	/api/fund-availability-projection/

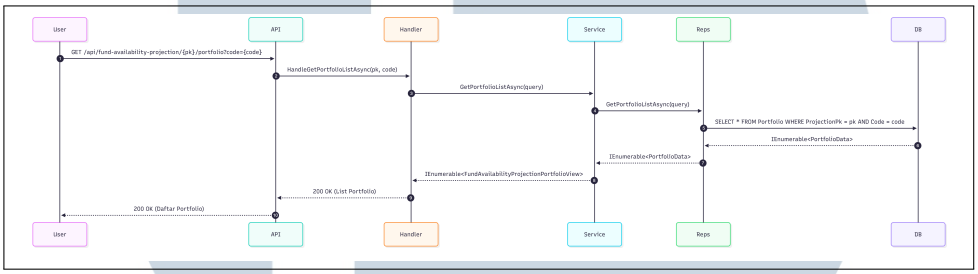
Request Body/Params	`AddFundAvailabilityProjectionDto`
	{
	`Date: DateOnly`
	`StartPeriod: DateOnly`
	`EndPeriod: DateOnly`
	`Description: string`
	}

Response	AckDto (Konfirmasi sukses) atau ValidationProblemDetails (Jika validasi gagal)
Permission Required	fund-availability-projection:add
Status Codes	200 OK, 400 Bad Request

Response yang dihasilkan dapat berupa *AckDto* sebagai tanda keberhasilan proses penambahan data, atau *ValidationProblemDetails* apabila terjadi kegagalan pada tahap validasi. Untuk mengakses API ini, pengguna diwajibkan memiliki *permission fund-availability-projection:add*. Adapun *status code* yang dikembalikan adalah 200 OK apabila proses berhasil, atau 400 Bad Request apabila terjadi *error*.

4. Alur dan Spesifikasi Api FundAvailabilityProjectionGetPortfolioList

Sequence Diagram pada Gambar 3.10 menggambarkan alur proses ketika sistem menampilkan daftar data *portfolio* berdasarkan *Primary Key* (Pk) dan kode kategori (*Code*) dari suatu *Fund Availability Projection*. Alur dimulai ketika Pengguna mengirimkan permintaan *GET* ke endpoint `/api/fund-availability-projection/pk/portfolio` dengan menyertakan kedua parameter tersebut.



Gambar 3.10. Sequence Diagram *Grid Portfolio*

Setelah menerima permintaan, *API layer* akan meneruskan data permintaan ke *Handler*. Pada tahap ini, *Handler* membentuk objek permintaan (*query*) yang kemudian dikirimkan ke *Service*. *Service* bertindak sebagai penghubung proses bisnis dan meminta *Repository* untuk mengambil data dari *database* berdasarkan Pk dan kode yang diberikan.

Repository mengeksekusi perintah *SELECT* ke *database*. Hasil dari *database* kemudian dikembalikan ke *Repository*, yang selanjutnya diteruskan kembali ke *Service*. Pada proses ini, tidak terdapat skenario kesalahan seperti *404 Not Found*, sebab permintaan ini bersifat pengambilan daftar (*list*). Apabila tidak ada data yang ditemukan, sistem tetap memberikan *respons 200 OK* dengan daftar kosong.

Pada langkah akhir, *Service* mengembalikan data ke *Handler*, kemudian *Handler* meneruskannya ke *API layer* untuk dikirimkan kembali kepada Pengguna sebagai *respons 200 OK*. Dengan demikian, Pengguna memperoleh daftar *portfolio* sesuai filter Pk dan *code* yang diberikan.

Pada Tabel 3.5 menunjukkan API yang digunakan untuk mengambil data portofolio. API ini berfungsi untuk mendapatkan data portfolio berdasarkan *primary key* (pk) dan *code* yang diisi dengan 'DEPO-MATURED'. Menggunakan *method GET* dengan *endpoint* /api/fund-availability-projection/pk/portfolio. *Response* berupa *FundAvailabilityProjectionPortfolioView* yang berisi data

portfolio. *Permission* yang diperlukan adalah *fund-availability-projection:read*, dan *status code* yang dikembalikan adalah 200 OK.

Tabel 3.5. Spesifikasi Api *FundAvailabilityProjectionGetPortfolioList*

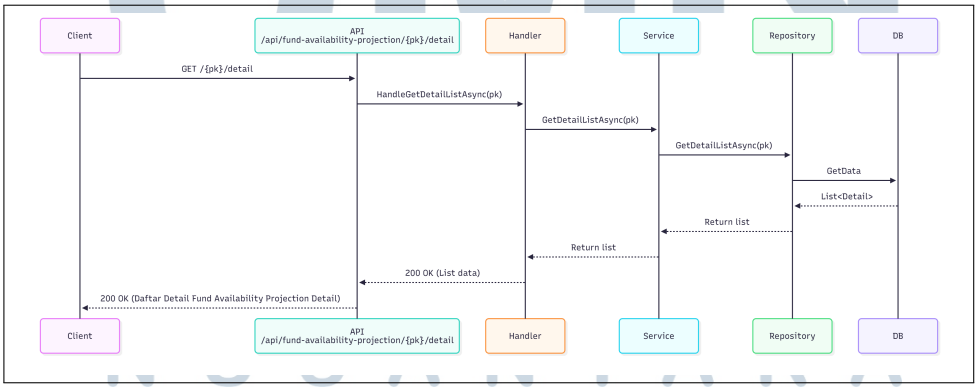
Nama API	FundAvailabilityProjectionGetPortfolioList
Method	GET
Endpoint Path	/api/fund-availability-projection/{pk}/portfolio

Request Body/Params	{
	Pk: long
	Code: string
	}

Response	IEnumerable(FundAvailabilityProjectionPortfolioView) (Daftar Portfolio)
Permission Required	fund-availability-projection:read
Status Codes	200 OK

5. Alur dan Spesifikasi Api FundAvailabilityProjectionGetDetailList

Sequence Diagram pada Gambar 3.11 menggambarkan alur proses ketika pengguna meminta daftar data detail *Fund Availability Projection* Detail berdasarkan parameter *primary key header*. Proses dimulai ketika pengguna mengirim permintaan *GET* ke endpoint /api/fund-availability-projection/pk/detail. Permintaan ini kemudian diteruskan oleh lapisan API ke *handler* yang bertanggung jawab memproses permintaan tersebut.



Gambar 3.11. Sequence Diagram *Grid Detail Fund Availability Projection Detail*

Selanjutnya, handler memanggil fungsi pada lapisan *service* yang kemudian meneruskan permintaan pengambilan data ke lapisan *repository*. Repository

melakukan eksekusi query ke *database* untuk mengambil seluruh data detail yang terkait dengan *primary key* yang diberikan.

Hasil pengambilan data dapat berupa daftar berisi satu atau beberapa data, atau bahkan kosong. Pada konteks ini, daftar kosong tidak dianggap sebagai kesalahan, melainkan kondisi valid karena API bertipe *list endpoint*. Oleh karena itu, berapa pun jumlah datanya, database tetap mengembalikan hasil ke repository, kemudian diteruskan kembali secara berjenjang ke service, handler, dan API.

Akhirnya, API memberikan respons *200 OK* kepada pengguna beserta daftar data detail yang berhasil diperoleh. Dengan demikian, diagram ini menunjukkan bahwa proses pengambilan daftar detail berjalan linear tanpa kondisi error khusus.

Pada Tabel 3.6 menunjukkan API yang digunakan untuk mengambil data *Fund Availability Projection* Detail. API ini berfungsi untuk mendapatkan data *Fund Availability Projection* Detail berdasarkan *primary key* (pk). Menggunakan *method GET* dengan *endpoint* /api/fund-availability-projection/pk/detail. *Response* berupa *FundAvailabilityProjectionPortfolioView* yang berisi data *Fund Availability Projection* Detail. *Permission* yang diperlukan adalah *fund-availability-projection:read*, dan *status code* yang dikembalikan adalah 200 OK.

Tabel 3.6. Spesifikasi Api *FundAvailabilityProjectionGetDetailList*

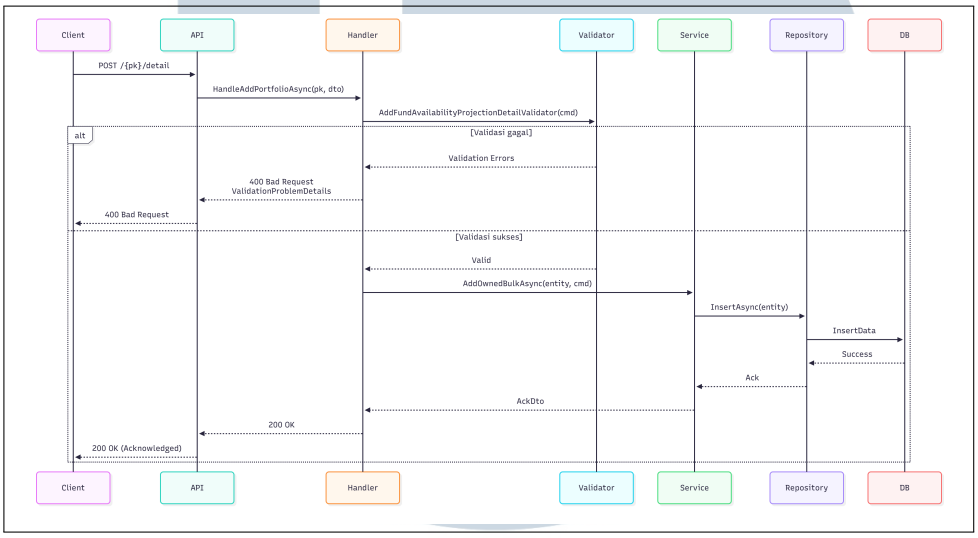
Nama API	FundAvailabilityProjectionGetDetailList
Method	GET
Endpoint Path	/api/fund-availability-projection/{pk}/detail

Request Body/Params	{
	`Pk: long`
	}

Response	IEnumerable(FundAvailabilityProjectionPortfolioView) (Daftar Detail *Fund Availability Projection*)
Permission Required	fund-availability-projection:read
Status Codes	200 OK

6. Alur dan Spesifikasi Api FundAvailabilityProjectionAddDetail

Sequence Diagram pada Gambar 3.12 menggambarkan alur proses ketika sistem menerima permintaan untuk menambahkan data detail *Fund Availability Projection*. Proses diawali oleh pengguna yang mengirimkan permintaan *POST* ke endpoint /api/fund-availability-projection/pk/detail. Permintaan ini membawa data detail dalam bentuk objek *AddFundAvailabilityProjectionDetailBulk*.



Gambar 3.12. Sequence Diagram *Add Data Fund Availability Projection Detail*

Setelah permintaan diterima oleh API, eksekusi diteruskan ke *handler* melalui metode HandleAddPortfolioAsync. Handler bertugas mengoordinasikan seluruh proses penambahan data, dan mempersiapkan data yang akan diteruskan ke service dalam bentuk cmd.

Apabila validasi berhasil, validator mengembalikan status valid kepada handler. Handler kemudian melanjutkan proses dengan memanggil service melalui metode AddOwnedBulkAsync. Lapisan service bertanggung jawab menyiapkan entitas yang akan disimpan, termasuk memvalidasi struktur dan isi data yang dikirim. Untuk itu, handler memanggil AddFundAvailabilityProjectionDetailValidator yang melakukan serangkaian pengecekan terhadap nilai-nilai pada *request*.

Jika proses validasi mendeteksi kesalahan—misalnya data tidak lengkap, tipe data tidak sesuai, atau aturan bisnis tidak terpenuhi—validator mengembalikan daftar error kepada handler. Handler kemudian membalas permintaan dengan *status 400 Bad Request* disertai *ValidationProblemDetails*. Pada kondisi ini, proses

dihentikan dan tidak ada data yang disimpan. Setelah proses ini selesai, service meneruskan entitas tersebut ke repository untuk diproses ke dalam database.

Pada tahap berikutnya, repository menjalankan operasi InsertAsync untuk menambahkan seluruh data detail dan portfolio ke dalam tabel terkait di database. Jika penyimpanan berhasil, database mengembalikan respons sukses kepada repository, yang kemudian diteruskan lagi ke service dan handler dalam bentuk AckDto sebagai tanda bahwa proses telah selesai tanpa kendala.

Terakhir, API mengirimkan respons *200 OK* kepada pengguna sebagai tanda bahwa data detail berhasil ditambahkan dan seluruh proses telah berjalan sesuai alur yang diharapkan.

Tabel 3.7 menjelaskan API yang digunakan untuk menambahkan data detail *Fund Availability Projection*, yaitu *FundAvailabilityProjectionAdd*. API ini berfungsi untuk menambahkan data detail baru dengan menggunakan *method POST* melalui *endpoint* /api/fund-availability-projection/. Data yang dikirimkan pada *request body* harus disusun dalam format *AddFundAvailabilityProjectionDto*.

Tabel 3.7. Spesifikasi Api *FundAvailabilityProjectionAddDetail*

Nama API	FundAvailabilityProjectionAddDetail
Method	POST
Endpoint Path	/api/fund-availability-projection/{pk}/detail

Request Body/Params	AddFundAvailabilityProjectionDetailBulk
	{
	Items: IEnumerable(
	AddFundAvailabilityProjectionDetailDto
	)
	}
	AddFundAvailabilityProjectionDetailDto
	{
	FundAvailabilityProjectionPk: long
	MvCurrency: string
	MvFundAvlVariable: string
	FundPk: long
	Amount: decimal
	Args: FundAvailabilityProjectionDetailArgsDto
	}
	FundAvailabilityProjectionDetailArgsDto
	{
	Portfolio: IEnumerable(PortfolioDto)
	}
	PortfolioDto
	{
	InstrumentPk: long
	}

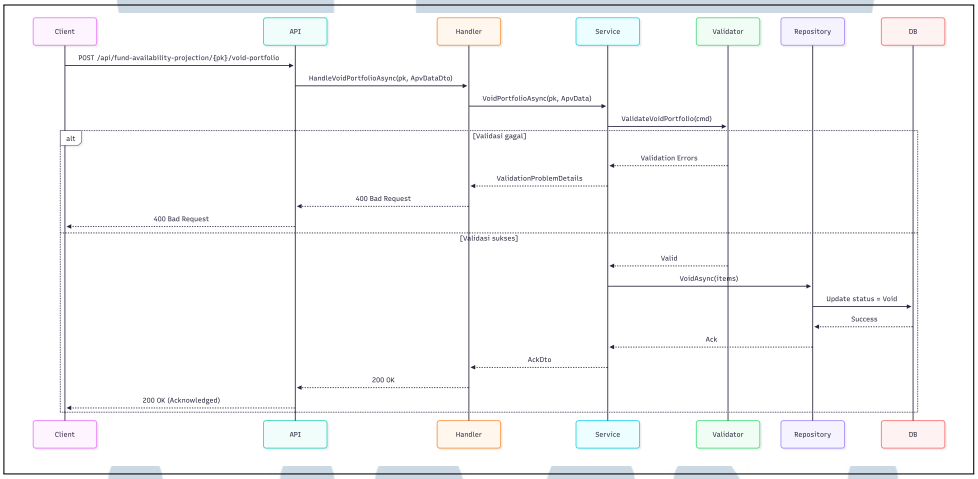
Response	AckDto (Konfirmasi sukses) atau ValidationProblemDetails (Jika validasi gagal)
Permission Required	fund-availability-projection:add
Status Codes	200 OK, 400 Bad Request

Response yang dihasilkan dapat berupa *AckDto* sebagai indikasi keberhasilan proses penambahan data, atau *ValidationProblemDetails* apabila terjadi kegagalan

pada tahap validasi. Untuk mengakses API tersebut, pengguna diwajibkan memiliki *permission fund-availability-projection:add*. Adapun *status code* yang dapat dikembalikan adalah 200 OK jika proses berhasil atau 400 Bad Request apabila terjadi *error*.

7. Alur dan Spesifikasi Api FundAvailabilityProjectionVoidPortfolio

Sequence Diagram pada Gambar 3.13 menggambarkan proses pembatalan (void) data detail *Fund Availability Projection*. Proses diawali ketika pengguna mengirimkan permintaan *POST* ke endpoint /api/fund-availability-projection/pk/void-portfolio dengan membawa data ApvDataDto yang berisi daftar data yang akan dibatalkan.



Gambar 3.13. Sequence Diagram *Add Data Fund Availability Projection Detail*

Permintaan tersebut diterima oleh API dan diteruskan ke handler melalui metode HandleVoidPortfolioAsync. Handler tidak melakukan proses bisnis secara langsung, melainkan meneruskan permintaan ke lapisan service melalui metode VoidPortfolioAsync.

Di dalam service, sistem terlebih dahulu melakukan proses validasi terhadap data permintaan. Validasi ini memastikan bahwa setiap item memiliki informasi yang lengkap, seperti Pk, HistoryPk, alasan pembatalan (Notes), serta jenis data (DataKind), dan memastikan bahwa data tersebut masih dapat dilakukan proses void.

Apabila validasi gagal, service akan mengembalikan informasi kesalahan ke handler dalam bentuk ValidationProblemDetails. Handler kemudian mengirimkan respons *400 Bad Request* kepada pengguna melalui API, dan proses dihentikan tanpa ada perubahan pada database.

Jika validasi berhasil, service melanjutkan proses dengan memanggil repository untuk melakukan pembatalan data. Repository akan menjalankan operasi ke database untuk memperbarui status data menjadi VOID serta mencatat riwayat perubahan ke dalam tabel histori.

Setelah proses penyimpanan berhasil, database mengembalikan status sukses ke repository, yang kemudian diteruskan ke service dalam bentuk AckDto. Service mengembalikan respons keberhasilan tersebut ke handler, dan handler mengirimkan respons akhir *200 OK* kepada pengguna melalui API.

Dengan alur ini, proses pembatalan data dilakukan secara terkontrol, memastikan validasi bisnis terpenuhi sebelum data diubah, serta menjaga konsistensi data melalui pencatatan histori.

Tabel 3.8 menjelaskan API yang digunakan untuk melakukan penambahan data detail *Fund Availability Projection*, yaitu *FundAvailabilityProjectionAdd*. API ini berfungsi untuk menambahkan data baru dengan menggunakan *method POST* melalui *endpoint* /api/fund-availability-projection/. Data yang dikirimkan pada *request body* harus disertakan dalam format *AddFundAvailabilityProjectionDto*.

Tabel 3.8. Spesifikasi Api *FundAvailabilityProjectionVoidPortfolio*

Nama API	FundAvailabilityProjectionVoidPortfolio
Method	POST
Endpoint Path	/api/fund-availability-projection/{pk}/void-portfolio

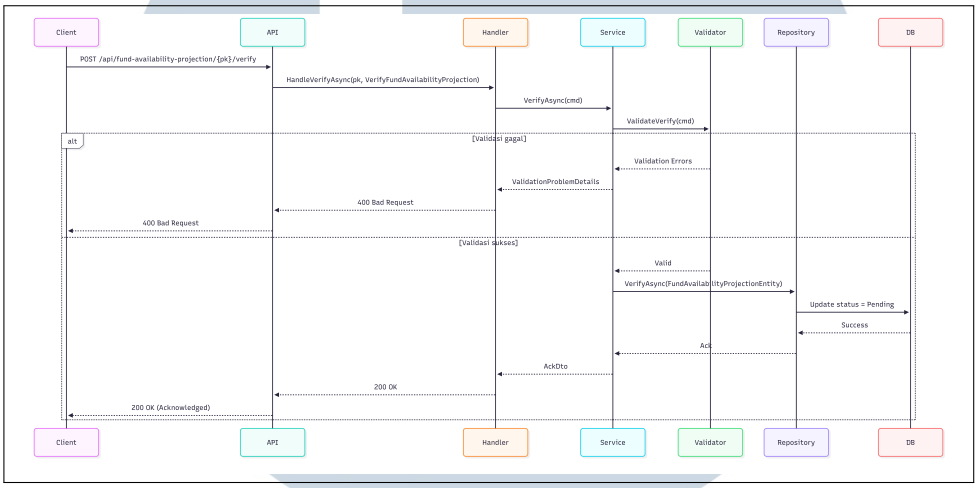
Request Body/Params	`ApvDataDto`
	{
	`Items: IEnumerable(ApvDataItemDto)`
	}
	`ApvDataItemDto`
	{
	`Pk: long`
	`HistoryPk: long`
	`Notes: string`
	}

Response	AckDto (Konfirmasi sukses) atau ValidationProblemDetails (Jika validasi gagal)
Permission Required	fund-availability-projection:void
Status Codes	200 OK, 400 Bad Request

Response yang dikembalikan dapat berupa *AckDto* sebagai tanda keberhasilan proses, atau *ValidationProblemDetails* apabila terjadi kegagalan pada proses validasi. Untuk dapat mengakses API ini, pengguna harus memiliki *permission fund-availability-projection:add*. Adapun *status code* yang dihasilkan adalah 200 OK apabila proses berhasil, atau 400 Bad Request jika terjadi *error*.

8. Alur dan Spesifikasi Api FundAvailabilityProjectionVerify

Sequence Diagram pada Gambar 3.14 menggambarkan alur proses verifikasi data *Fund Availability Projection*. Proses dimulai ketika pengguna mengirimkan permintaan *POST* ke endpoint /api/fund-availability-projection/pk/verify dengan membawa data VerifyFundAvailabilityProjection yang berisi identitas data yang akan diverifikasi.



Gambar 3.14. Sequence Diagram *Verify Data Fund Availability Projection*

Permintaan tersebut diterima oleh API dan diteruskan ke handler melalui metode HandleVerifyAsync. Handler berfungsi sebagai penghubung awal dan meneruskan proses bisnis ke lapisan service melalui metode VerifyAsync.

Pada lapisan service, sistem terlebih dahulu menjalankan proses validasi terhadap permintaan verifikasi. Validasi ini bertujuan untuk memastikan bahwa data dengan FundAvailabilityProjectionPk yang dikirimkan benar-benar ada, masih berada pada status yang dapat diverifikasi, serta memenuhi aturan bisnis yang berlaku.

Apabila validasi gagal, service akan mengembalikan informasi kesalahan dalam bentuk ValidationProblemDetails ke handler. Handler kemudian meneruskan respons tersebut ke API, dan pengguna akan menerima respons *400 Bad Request*.

Jika validasi berhasil, service melanjutkan proses dengan memanggil repository untuk memperbarui status data menjadi VERIFIED. Repository akan melakukan operasi pembaruan status ke dalam database, dan apabila proses tersebut berhasil, database akan mengembalikan respons sukses.

Respons keberhasilan tersebut diteruskan kembali dari repository ke service dalam bentuk Ack, kemudian ke handler sebagai AckDto. Handler selanjutnya

mengirimkan respons *200 OK* ke API, dan API mengembalikan konfirmasi keberhasilan kepada pengguna.

Dengan alur ini, proses verifikasi data dilakukan secara terstruktur melalui arsitektur berlapis, memastikan bahwa hanya data yang valid dan memenuhi aturan bisnis yang dapat berubah status menjadi verified.

Tabel 3.9 menjelaskan API yang digunakan untuk melakukan proses verifikasi data detail *Fund Availability Projection*, yaitu *FundAvailabilityProjectionAdd*. API ini dimanfaatkan untuk menambahkan data baru dengan menggunakan *method POST* melalui *endpoint* /api/fund-availability-projection/. Data yang dikirimkan pada *request body* wajib disusun dalam format *AddFundAvailabilityProjectionDto*.

Tabel 3.9. Spesifikasi Api *FundAvailabilityProjectionVerify*

Nama API	FundAvailabilityProjectionVerify
Method	POST
Endpoint Path	/api/fund-availability-projection/{pk}/verify

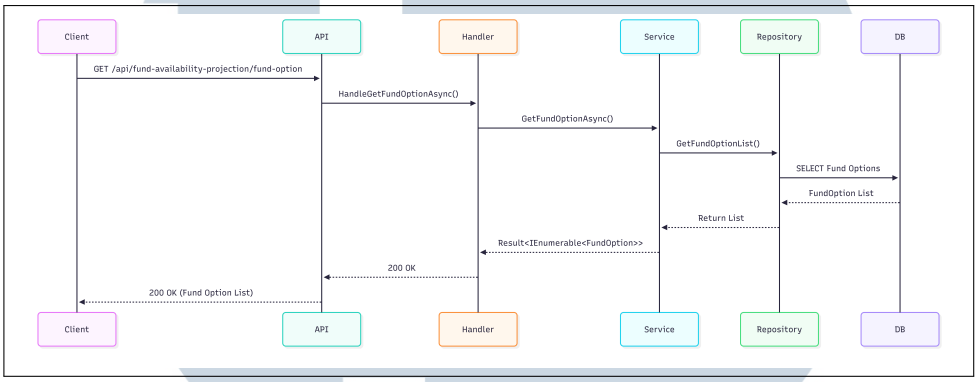
Request Body/Params	`VerifyFundAvailabilityProjection`
	{
	`FundAvailabilityProjectionPk: long`
	}

Response	AckDto (Konfirmasi sukses) atau ValidationProblemDetails (Jika validasi gagal)
Permission Required	fund-availability-projection:add
Status Codes	200 OK, 400 Bad Request

Response yang dihasilkan dapat berupa *AckDto* sebagai konfirmasi keberhasilan proses, atau *ValidationProblemDetails* apabila terjadi kegagalan pada tahap validasi. Untuk dapat mengakses API tersebut, pengguna harus memiliki *permission fund-availability-projection:add*. Adapun *status code* yang dikembalikan adalah 200 OK jika proses berjalan dengan baik, atau 400 Bad Request apabila terjadi *error*.

9. Alur dan Spesifikasi Api FundAvailabilityProjectionGetFundOption

Sequence Diagram pada Gambar 3.15 menggambarkan alur proses pengambilan data pilihan fund yang digunakan pada fitur *Fund Availability Projection*. Proses diawali ketika pengguna mengirimkan permintaan *GET* ke endpoint /api/fund-availability-projection/fund-option tanpa membawa parameter tambahan.



Gambar 3.15. Sequence Diagram *Fund Option*

Permintaan tersebut diterima oleh API dan diteruskan ke handler melalui metode HandleGetFundOptionAsync. Handler berperan sebagai penghubung awal yang meneruskan permintaan ke lapisan service untuk diproses lebih lanjut.

Pada lapisan service, sistem menjalankan logika bisnis untuk mengambil daftar fund yang tersedia dengan memanggil repository melalui metode GetFundOptionList. Repository kemudian berinteraksi langsung dengan database untuk mengambil data fund yang diperlukan.

Setelah database mengembalikan hasil berupa daftar fund option, data tersebut diteruskan kembali secara berurutan dari repository ke service, kemudian ke handler. Handler selanjutnya mengemas data hasil pengambilan tersebut ke dalam respons API dengan status *200 OK*.

Akhirnya, API mengirimkan respons kepada pengguna berupa daftar fund option yang dapat digunakan sebagai pilihan pada proses pengisian atau penyaringan data *Fund Availability Projection*. Apabila data yang tersedia kosong, sistem tetap mengembalikan respons *200 OK* dengan isi data berupa daftar kosong, menandakan bahwa permintaan berhasil diproses tanpa kesalahan.

Tabel 3.10 menjelaskan API yang digunakan untuk melakukan penambahan data detail *Fund Availability Projection*, yaitu *FundAvailabilityProjectionAdd*. API

ini berfungsi untuk menyimpan data baru dengan menggunakan *method POST* melalui *endpoint* /api/fund-availability-projection/. Data yang dikirimkan pada *request body* harus disusun dalam format *AddFundAvailabilityProjectionDto*.

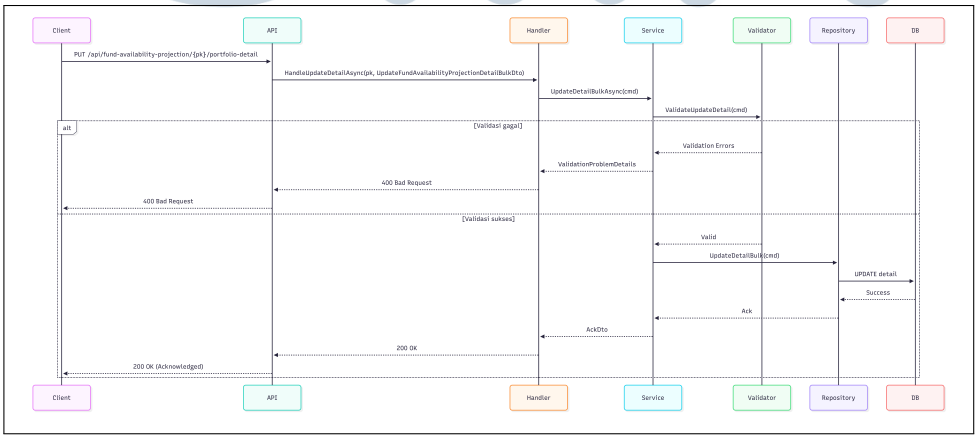
Tabel 3.10. Spesifikasi Api *FundAvailabilityProjectionGetFundOption*

Nama API	FundAvailabilityProjectionGetFundOption
Method	GET
Endpoint Path	/api/fund-availability-projection/fund-option
Request Body/Params	-
Response	-
Permission Required	fund-availability-projection:read
Status Codes	-

Response yang dihasilkan dapat berupa *AckDto* sebagai indikasi keberhasilan proses, atau *ValidationProblemDetails* apabila terjadi kegagalan pada tahap validasi. Untuk mengakses API tersebut, pengguna diwajibkan memiliki *permission fund-availability-projection:add*. Adapun *status code* yang dapat dikembalikan adalah 200 OK jika proses berhasil atau 400 Bad Request apabila terjadi *error*.

10. Alur dan Spesifikasi Api FundAvailabilityProjectionUpdateDetail

Sequence Diagram pada Gambar 3.16 menggambarkan alur proses pembaruan data detail *Fund Availability Projection*. Proses dimulai ketika pengguna mengirimkan permintaan *PUT* ke endpoint /api/fund-availability-projection/pk/portfolio-detail dengan membawa data UpdateFundAvailabilityProjectionDetailBulkDto yang berisi daftar detail yang akan diperbarui.



Gambar 3.16. Sequence Diagram *Update Data Detail*

Permintaan tersebut diterima oleh API dan diteruskan ke handler melalui metode HandleUpdateDetailAsync. Handler berperan sebagai penghubung awal dan meneruskan proses ke lapisan service untuk menjalankan logika bisnis pembaruan data.

Pada lapisan service, sistem terlebih dahulu melakukan validasi terhadap data yang dikirimkan. Proses validasi ini bertujuan untuk memastikan bahwa data detail yang akan diperbarui valid, memiliki relasi yang sesuai dengan FundAvailabilityProjectionPk, serta memenuhi aturan bisnis yang berlaku. Apabila validasi gagal, service akan mengembalikan ValidationProblemDetails ke handler, yang kemudian diteruskan ke API sebagai respons *400 Bad Request*.

Jika proses validasi berhasil, service melanjutkan proses pembaruan data dengan memanggil repository untuk melakukan pembaruan data secara massal. Repository kemudian mengeksekusi perintah pembaruan data detail ke dalam database sesuai dengan informasi yang dikirimkan. Setelah proses pembaruan di database berhasil, respons keberhasilan dikembalikan dari database ke repository, kemudian ke service dalam bentuk Ack.

Service selanjutnya mengembalikan hasil tersebut ke handler dalam bentuk AckDto, yang kemudian diteruskan oleh handler ke API. API akhirnya mengirimkan respons *200 OK* kepada pengguna sebagai konfirmasi bahwa proses pembaruan data detail *Fund Availability Projection* telah berhasil dilakukan.

Dengan alur ini, pembaruan data detail dilakukan secara terstruktur dan aman melalui arsitektur berlapis, sekaligus memastikan konsistensi data dan validitas perubahan yang dilakukan.

Tabel 3.11 menjelaskan API yang digunakan untuk melakukan proses *update* data *Fund Availability Projection*. API ini berfungsi untuk memperbarui data yang telah tersimpan sebelumnya dengan menggunakan *method PUT* melalui *endpoint* /api/fund-availability-projection/pk/portfolio-detail. Data yang dikirimkan pada *request body* harus disusun dalam format *UpdateFundAvailabilityProjectionDetailBulkDto*.

Tabel 3.11. Spesifikasi Api *FundAvailabilityProjectionUpdateDetail*

Nama API	FundAvailabilityProjectionUpdateDetail
Method	PUT
Endpoint Path	/api/fund-availability-projection/{pk}/portfolio-detail

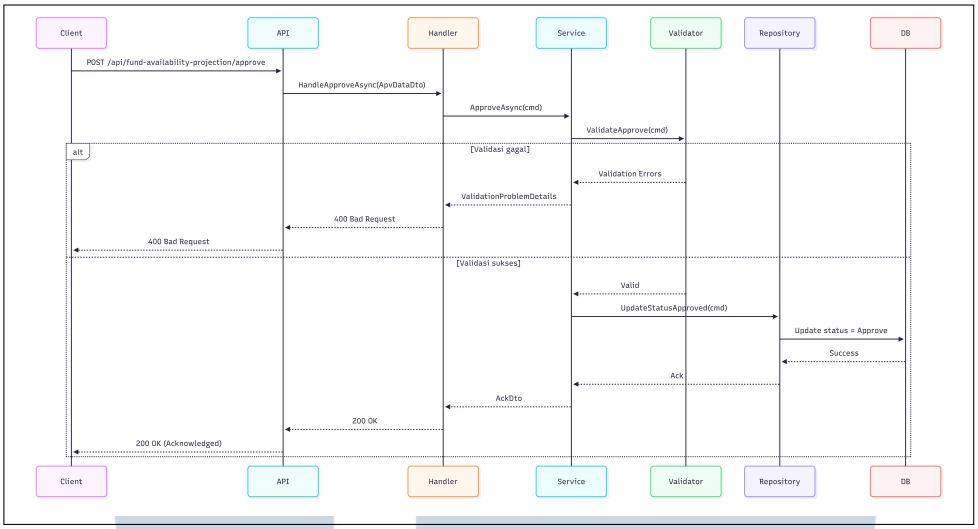
Request Body/Params	`UpdateFundAvailabilityProjectionDetailBulkDto`
	{
	`FundAvailabilityProjectionPk: long`
	`Items: IEnumerable(`
	`UpdateFundAvailabilityProjectionDetailDto`
	)
	}
	`UpdateFundAvailabilityProjectionDetailDto`
	{
	`HistoryPk: long`
	`FundAvailabilityProjectionDetailPk: long`
	`Amount: decimal`
	}

Response	AckDto (Konfirmasi sukses) atau ValidationProblemDetails (Jika validasi gagal)
Permission Required	fund-availability-projection:update
Status Codes	200 OK, 400 Bad Request

Response yang dihasilkan dapat berupa *AckDto* sebagai konfirmasi keberhasilan proses pembaruan data, atau *ValidationProblemDetails* apabila terjadi kegagalan pada tahap validasi. Untuk dapat mengakses API ini, pengguna diwajibkan memiliki *permission fund-availability-projection:update*. Adapun *status code* yang dikembalikan adalah 200 OK apabila proses berjalan dengan baik, atau 400 Bad Request jika terjadi kesalahan.

11. Alur dan Spesifikasi Api FundAvailabilityProjectionApprove

Sequence Diagram pada Gambar 3.17 menggambarkan alur proses persetujuan (approve) data *Fund Availability Projection*. Proses diawali ketika pengguna mengirimkan permintaan *POST* ke endpoint /api/fund-availability-projection/approve dengan membawa data ApvDataDto yang berisi informasi identitas data, riwayat perubahan, serta catatan persetujuan.



Gambar 3.17. Sequence Diagram *Approve Data*

Permintaan tersebut diterima oleh API dan diteruskan ke handler melalui metode HandleApproveAsync. Handler kemudian meneruskan proses ke lapisan service untuk menjalankan logika bisnis persetujuan data.

Pada lapisan service, sistem terlebih dahulu melakukan proses validasi terhadap data yang dikirimkan. Validasi ini bertujuan untuk memastikan bahwa data yang akan disetujui berada pada status yang tepat, memiliki relasi data yang valid, serta memenuhi aturan bisnis yang berlaku. Apabila validasi gagal, service akan mengembalikan informasi kesalahan dalam bentuk ValidationProblemDetails ke handler, yang kemudian diteruskan ke API sebagai respons *400 Bad Request*.

Jika validasi berhasil, service melanjutkan proses persetujuan dengan memanggil repository untuk memperbarui status data menjadi APPROVED. Repository akan melakukan pembaruan status tersebut ke dalam database. Setelah proses berhasil, database mengembalikan respons sukses yang diteruskan kembali melalui repository dan service dalam bentuk Ack.

Service kemudian mengembalikan AckDto ke handler, dan handler meneruskannya ke API sebagai respons *200 OK*. API selanjutnya mengirimkan konfirmasi keberhasilan kepada pengguna bahwa proses persetujuan data *Fund Availability Projection* telah selesai.

Dengan alur ini, proses persetujuan data dilakukan secara terkontrol melalui arsitektur berlapis, sehingga memastikan bahwa hanya data yang valid dan memenuhi ketentuan bisnis yang dapat berubah ke status approved.

Tabel 3.12 menjelaskan API yang digunakan untuk melakukan proses *approve*

data *Fund Availability Projection*. API ini berfungsi untuk memberikan persetujuan terhadap data dengan menggunakan *method POST* melalui *endpoint* /api/fund-availability-projection/approve. Data yang dikirimkan pada *request body* harus memuat informasi yang akan disetujui.

Tabel 3.12. Spesifikasi Api *FundAvailabilityProjectionApprove*

Nama API	FundAvailabilityProjectionApprove
Method	POST
Endpoint Path	/api/fund-availability-projection/approve

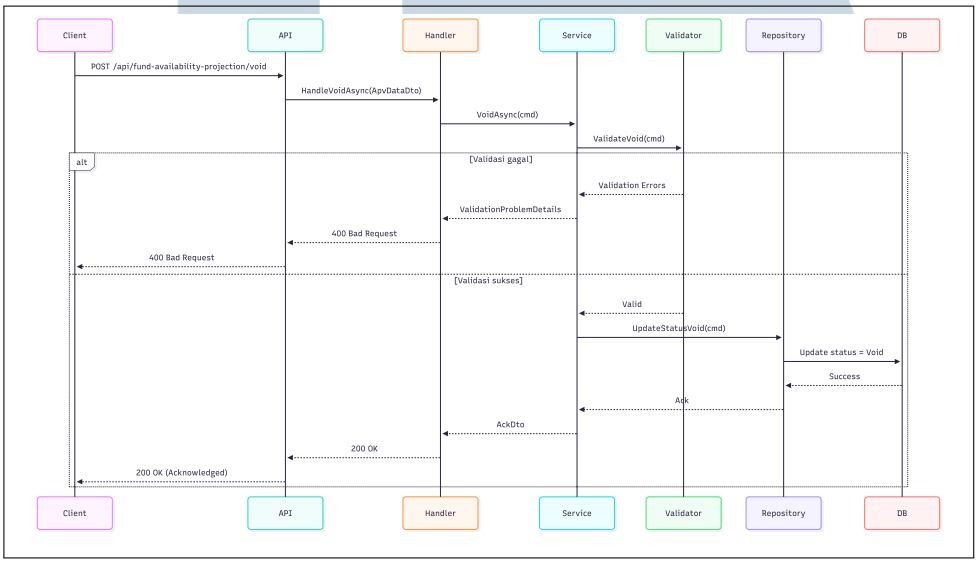
Request Body/Params	`ApvDataDto`
	{
	`Pk: long`
	`HistoryPk: long`
	`Notes: string`
	}

Response	AckDto (Konfirmasi sukses) atau ValidationProblemDetails (Jika validasi gagal)
Permission Required	fund-availability-projection:approve
Status Codes	200 OK, 400 Bad Request

Response yang dihasilkan dapat berupa *AckDto* sebagai tanda keberhasilan proses persetujuan, atau *ValidationProblemDetails* apabila terjadi kegagalan pada tahap validasi. Untuk dapat mengakses API ini, pengguna diwajibkan memiliki *permission fund-availability-projection:approve*. Adapun *status code* yang dikembalikan adalah 200 OK jika proses berhasil, atau 400 Bad Request apabila terjadi kesalahan.

12. Alur dan Spesifikasi Api FundAvailabilityProjectionVoid

Sequence Diagram pada Gambar 3.18 menggambarkan alur proses pembatalan (void) data *Fund Availability Projection*. Proses diawali ketika pengguna mengirimkan permintaan *POST* ke endpoint /api/fund-availability-projection/void dengan membawa data ApvDataDto yang berisi identitas data, riwayat perubahan, serta catatan pembatalan.



Gambar 3.18. Sequence Diagram *Void Data*

Permintaan tersebut diterima oleh API dan diteruskan ke handler melalui metode HandleVoidAsync. Handler berperan sebagai penghubung awal dan meneruskan permintaan ke lapisan service untuk menjalankan proses bisnis pembatalan data.

Pada lapisan service, sistem terlebih dahulu melakukan proses validasi terhadap data yang dikirimkan. Validasi ini bertujuan untuk memastikan bahwa data yang akan dibatalkan masih berada pada status yang diperbolehkan untuk di-void, memiliki identitas data yang valid, serta memenuhi aturan bisnis yang berlaku. Apabila validasi gagal, service akan mengembalikan ValidationProblemDetails ke handler, yang kemudian diteruskan ke API sebagai respons *400 Bad Request*.

Jika validasi berhasil, service melanjutkan proses pembatalan dengan memanggil repository untuk memperbarui status data menjadi VOID. Repository kemudian melakukan pembaruan status tersebut ke dalam database. Setelah proses pembaruan berhasil, database mengembalikan respons sukses yang diteruskan kembali ke service melalui repository dalam bentuk Ack.

Service kemudian mengembalikan AckDto ke handler, yang selanjutnya meneruskannya ke API sebagai respons *200 OK*. API akhirnya mengirimkan konfirmasi kepada pengguna bahwa proses pembatalan data *Fund Availability Projection* telah berhasil dilakukan.

Dengan alur ini, proses pembatalan data dapat dilakukan secara terkontrol dan terdokumentasi dengan baik melalui arsitektur berlapis, sehingga menjaga konsistensi dan integritas data dalam sistem.

Pada Tabel 3.13 dijelaskan API yang digunakan untuk melakukan proses *void* pada data *Fund Availability Projection*. API ini berfungsi untuk membatalkan atau menghapus data dengan menggunakan *method POST* melalui *endpoint* /api/fund-availability-projection/void. Data yang dikirimkan pada *request body* harus memuat informasi data yang akan dihapus.

Tabel 3.13. Spesifikasi Api *FundAvailabilityProjectionVoid*

Nama API	FundAvailabilityProjectionVoid
Method	POST
Endpoint Path	/api/fund-availability-projection/void

Request Body/Params	`ApvDataDto`
	{
	`Pk: long`
	`HistoryPk: long`
	`Notes: string`
	}

Response	AckDto (Konfirmasi sukses) atau ValidationProblemDetails (Jika validasi gagal)
Permission Required	fund-availability-projection:void
Status Codes	200 OK, 400 Bad Request

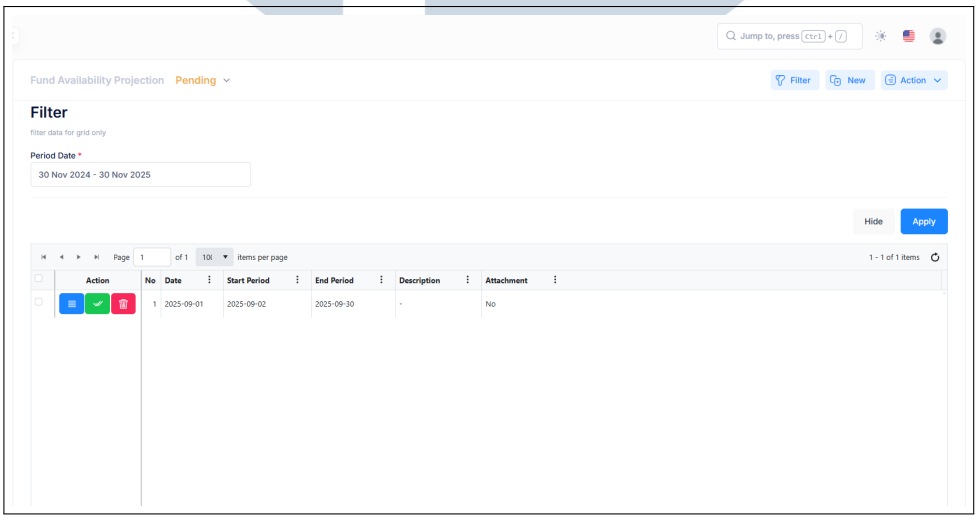
Response yang dikembalikan dapat berupa *AckDto* sebagai indikasi bahwa proses berhasil dijalankan, atau *ValidationProblemDetails* apabila terjadi kegagalan pada proses validasi. Untuk mengakses API ini, pengguna diwajibkan memiliki *permission fund-availability-projection:void*. Adapun *status code* yang dihasilkan adalah 200 OK jika proses berhasil, atau 400 Bad Request apabila terjadi kesalahan.

Tampilan Menu *Fund Availability Projection* dibuat dengan design yang sederhana namun fungsional dengan tujuan agar pengguna mudah untuk

mengoperasikannya. Pada sisi *frontend* menu ini menggunakan html dan javascript sebagai bahasa pemrograman, Bootstrap dan Metronic sebagai framework, beserta kendo sebagai library js. Berikut adalah rincian Tampilan menu FundAvailabilityProjection.

1. Tampilan Page Index

Gambar 3.19 merupakan tampilan *grid* pada *index* menu *Fund Availability Projection*. *Grid* dibuat dengan tujuan agar *pengguna* dapat melihat data hasil *inputan* sesuai dengan status masing-masing data tersebut, serta dapat melakukan beberapa aksi untuk mengelola data tersebut. Pada sisi tengah terdapat grid yang berisikan data index *Fund Availability Projection*, disisi kanan atas terdapat *action button* yang terdiri dari *Approve* dan *Void* yang digunakan untuk mengelola data yang dipilih pada grid, *filter button* untuk menampilan filter yang terdiri dari *date range* beserta *button hide* dan *button apply*, serta *button new* untuk membuat data proyeksi baru.

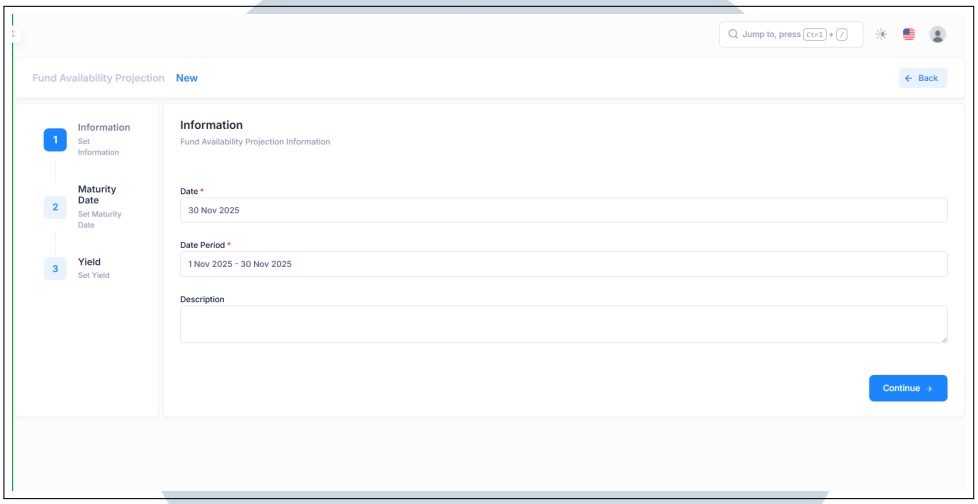


Gambar 3.19. Tampilan *Page Index* Menu *Fund Availability Projection*

2. Tampilan Page Create New Data Header

Gambar 3.20 merupakan tampilan fitur *create new data header Fund Availability Projection*. *Create new data header* dibuat dengan tujuan agar *pengguna* dapat menambahkan data berdasarkan *field* yang telah disediakan seperti *Date, Date Period* dan juga *Description* sekaligus menjadi data dari *field* tersebut digunakan sebagai parameter untuk langkah berikutnya. Setelah mengisi *field* yang disediakan, *pengguna* dapat melakukan *Save* pada tombol *continue* dibawah kanan

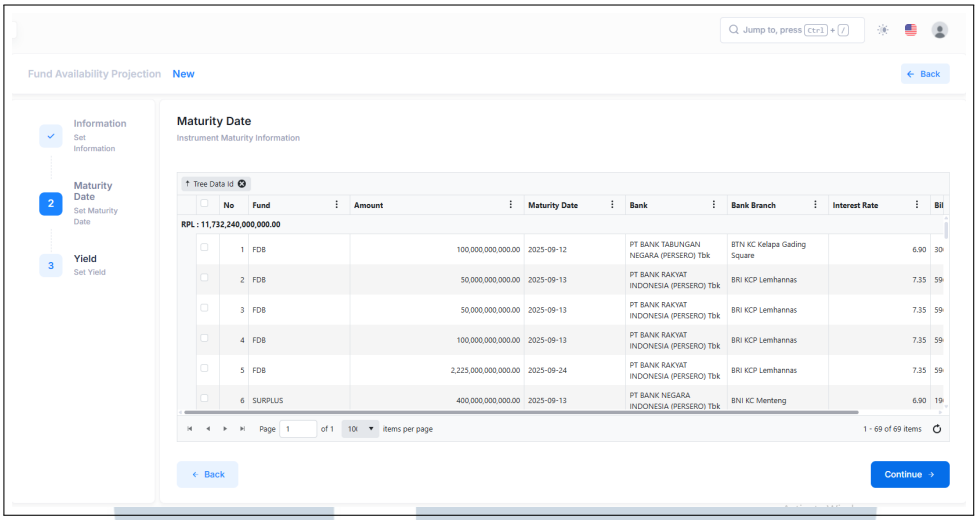
atau kembali ke halaman utama dengan menekan tombol *Back* pada sisi kanan atas. Pada sisi kiri terdapat *section-section* yang harus dilalui sebelum akhirnya seluruh data selesai di input.



Gambar 3.20. Tampilan *Create New Data Header Fund Availability Projection*

3. Tampilan Page Create New Data Portfolio Mature

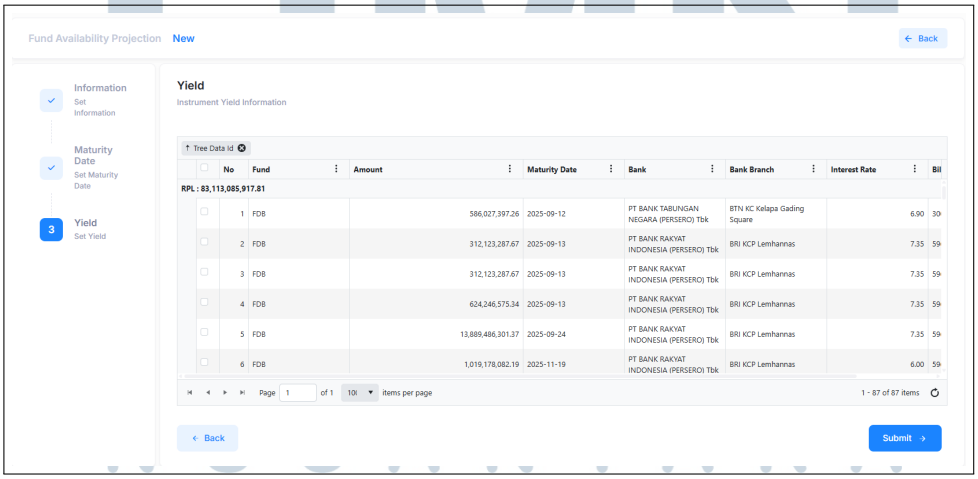
Gambar 3.21 merupakan tampilan *page create new data portfolio mature*. Page ini berada pada *section* kedua setelah melewati *section* pertama. Data pada page ini *dependency* dengan page sebelumnya pada *section create new data header* dengan parameter yang dibawa. Pada page ini pengguna dapat melihat data dari portofolio dengan tanggal yang akan jatuh tempo sesuai dengan parameter di page sebelumnya, yang nantinya akan dipilih pengguna untuk dimasukkan ke database. Pada sisi kanan bawah terdapat tombol *Continue* untuk melanjutkan ke *section* terakhir, dan tombol *Back* dikiri bawah untuk kembali ke *section* sebelumnya.



Gambar 3.21. Tampilan *Page Create New Data Portfolio Mature*

4. Tampilan Page Create New Data Portfolio Yield

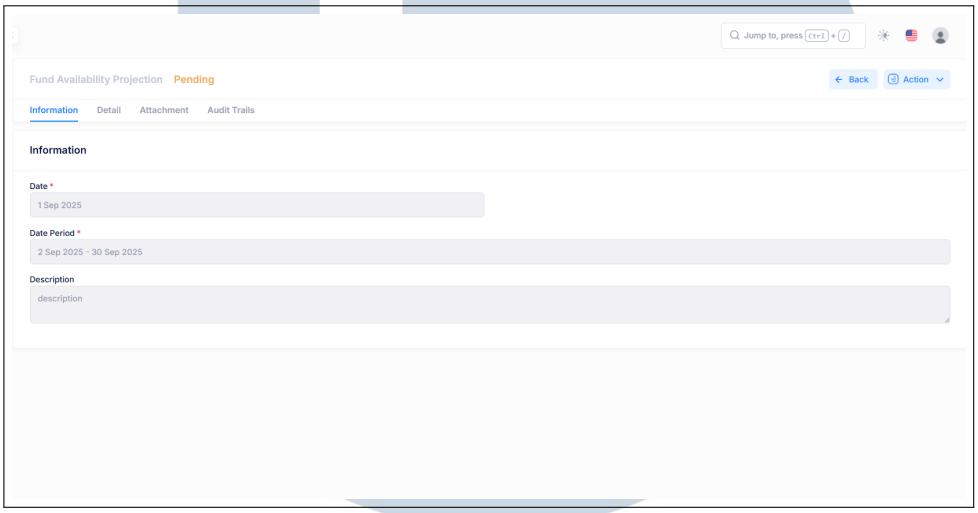
Gambar 3.22 merupakan tampilan *page create new data portfolio yield*. Page ini berada pada *section* ketiga atau *section* terakhir setelah melewati *section* kedua. Secara keseluruhan *behavior* pada page ini mirip dengan page pada *section* kedua. Namun data pada page ini diambil dengan logika yang berbeda yang nantinya akan berisikan data yang akan diproyeksi cair, yang nantinya juga akan dipilih pengguna untuk dimasukkan ke database. Pada sisi kanan bawah terdapat tombol *Submit* untuk menyelesaikan *seluruh section*, lalu membawa data ke *status Pending*, serta tombol *Back* dikiri bawah untuk kembali ke *section* sebelumnya.



Gambar 3.22. Tampilan *Page Create New Data Portfolio Yield*

5. Tampilan Page Detail Header

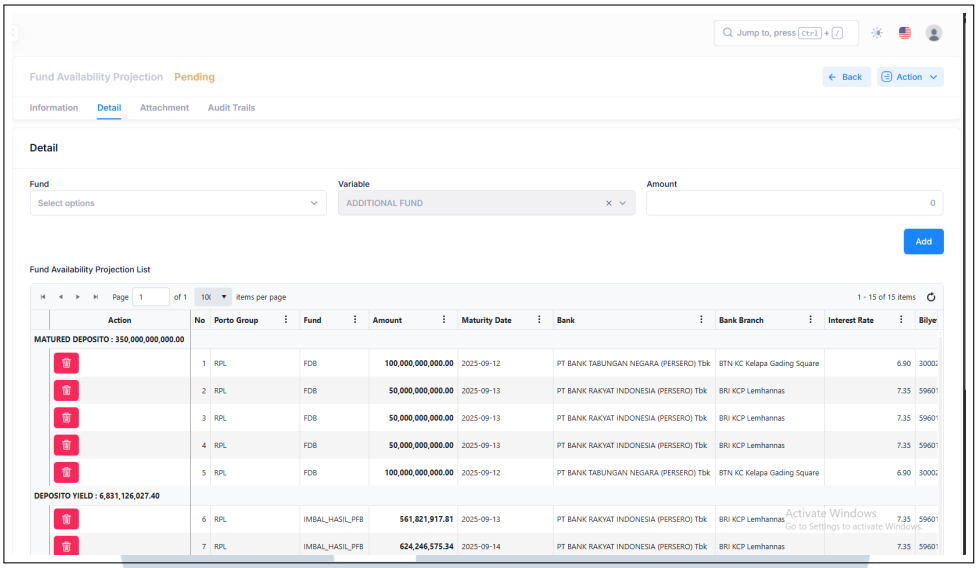
Gambar 3.23 merupakan tampilan *page detail header* pada menu *Fund Availability Projection*. *Page detail header* ditempatkan pada tab *Information* dengan tujuan agar pengguna melihat detail data header yang terdiri dari *Date, Date Period* dan *Description*. Data pada page ini tidak dapat dirubah atau bersifat *readable only*.



Gambar 3.23. Tampilan *Page Detail Header Menu Fund Availability Projection*

6. Tampilan Page Detail Portfolio

Gambar 3.24 merupakan tampilan *page detail portfolio* yang ditempatkan pada *tab detail*. Page ini menampilkan data-data barang yang telah dipilih oleh pengguna oleh grid. Pada grid detail data di *grouping* berdasarkan *tree fund* dan juga berdasarkan *variable fund*, lalu amount dijumlahkan berdasarkan *grouping* tersebut yang terdapat pada bagian *footer grid*. Data pada page ini dapat di *Approve, Void* dan *Update*. Tombol *update* berguna untuk merubah *value* dari *amount* itu sendiri. Lalu terdapat juga *field* tambahan yang diisi seperti *Fund, Amount* dan juga *Variable Fund* yang sudah terisi dan tidak dapat dirubah. Lalu setelah seluruh *field* diisi, pengguna akan menekan tombol update untuk menambahkan data portfolio pada *grid portfolio*.



Gambar 3.24. Tampilan *Page Detail Portfolio*

3.4 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan kegiatan magang, kendala utama yang dihadapi adalah kesulitan dalam memahami struktur kode pada aplikasi *Blackbird*. Aplikasi ini dikembangkan dengan standar kode dan tingkat kedisiplinan yang tinggi, serta memiliki jumlah fungsi yang banyak dengan tingkat kompleksitas yang cukup tinggi. Kondisi tersebut menuntut pemahaman yang mendalam terhadap alur program dan keterkaitan antar modul. Selain itu, dalam proses magang juga terdapat tuntutan untuk mengerjakan beberapa *project* secara berurutan dengan fokus penuh, karena setiap *project* tidak boleh mengganggu atau berdampak pada *project* lain. Hal ini disebabkan aplikasi *Blackbird* digunakan secara aktif oleh banyak pengguna dari berbagai klien, sehingga stabilitas dan konsistensi sistem harus selalu terjaga.

Sebagai solusi atas kendala tersebut, peran *supervisor* sangat membantu dalam proses adaptasi dan pembelajaran. *supervisor* secara aktif memberikan bimbingan, penjelasan, serta arahan terkait struktur kode, standar pengembangan, dan logika bisnis yang diterapkan dalam aplikasi. Dengan adanya pendampingan yang berkelanjutan, proses pemahaman terhadap sistem menjadi lebih terarah dan efisien. Selain itu, bimbingan tersebut membantu mempercepat proses adaptasi terhadap lingkungan kerja dan kompleksitas aplikasi, sehingga kendala yang dihadapi dapat diatasi secara bertahap dan mendukung kelancaran pelaksanaan kegiatan magang.