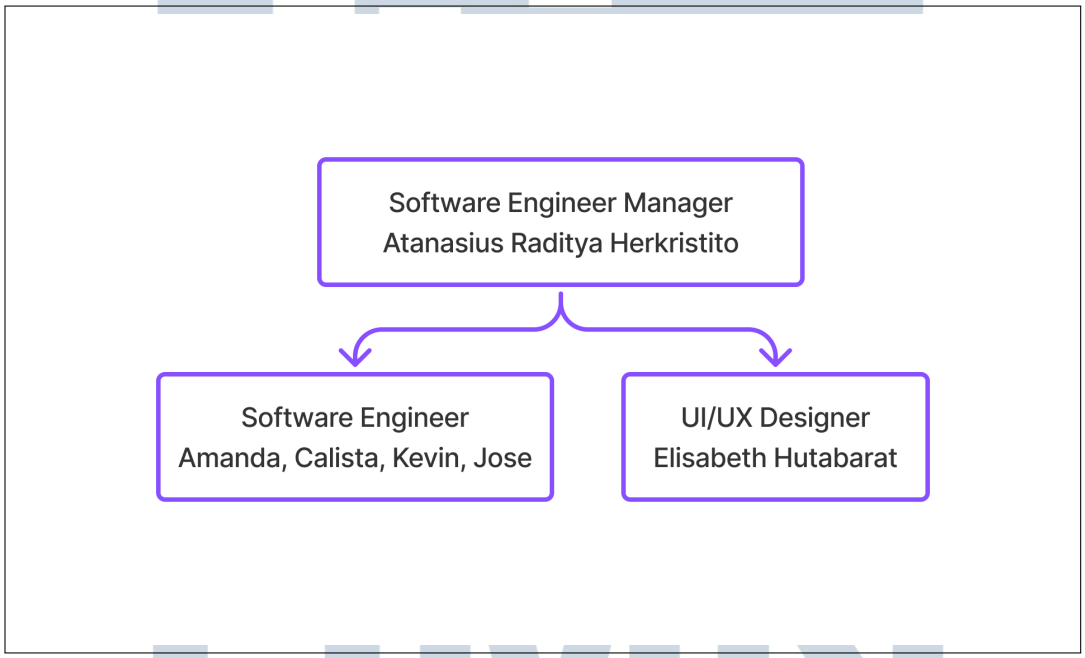
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Dalam pelaksanaan magang di PT Visi Karya Nusantara, posisi yang ditempatkan adalah sebagai *Software Engineer*. Berikut merupakan struktur organisasi untuk pengembangan *Human Resource Information System* (HRIS) pada gambar 3.1.



Gambar 3.1. Kedudukan dan Organisasi

Dalam pengembangan HRIS terdapat lima anggota tim yang terbagi menjadi empat *software engineer* dan satu *UI/UX Designer*. Selama pengembangan HRIS, praktik kerja dan koordinasi dipimpin oleh Atanasius Raditya Herkistito sebagai *Software Engineer Manager* bertugas untuk mengatur, merencanakan keberlangsungan proyek dari awal hingga akhir [7]. Untuk memantau hasil kinerja dilakukan *weekly meeting* agar tugas yang dilakukan masih sesuai dengan rencana yang sudah ditetapkan.

Selama proses pengerjaan, adapun *tools* yang digunakan dalam pengembangan HRIS seperti Apidog yang digunakan untuk membantu percobaan *REST API* yang telah dikembangkan, DBeaver yang digunakan untuk melihat

data-data yang tersimpan ke dalam *database* secara visual, serta Visual Studio Code yang dipilih sebagai Lingkungan Pengembangan Terintegrasi (IDE) untuk meningkatkan efisiensi penulisan kode.

3.2 Tugas yang Dilakukan

Adapun tugas-tugas yang dilakukan selama periode magang di antaranya membangun dan merancang pembuatan fitur API *contract* dan melakukan beberapa perubahan fitur yang sudah ada untuk dikaitkan ke fitur *contract* seperti, *attendance date, payslip date, daily attendance*, dan juga pengimplementasian *socket*. Dengan menggunakan *Express.js* untuk pembuatan *Application Programming Interface* (API), lalu menguji API yang telah dirancang.

3.3 Uraian Pelaksanaan Magang

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

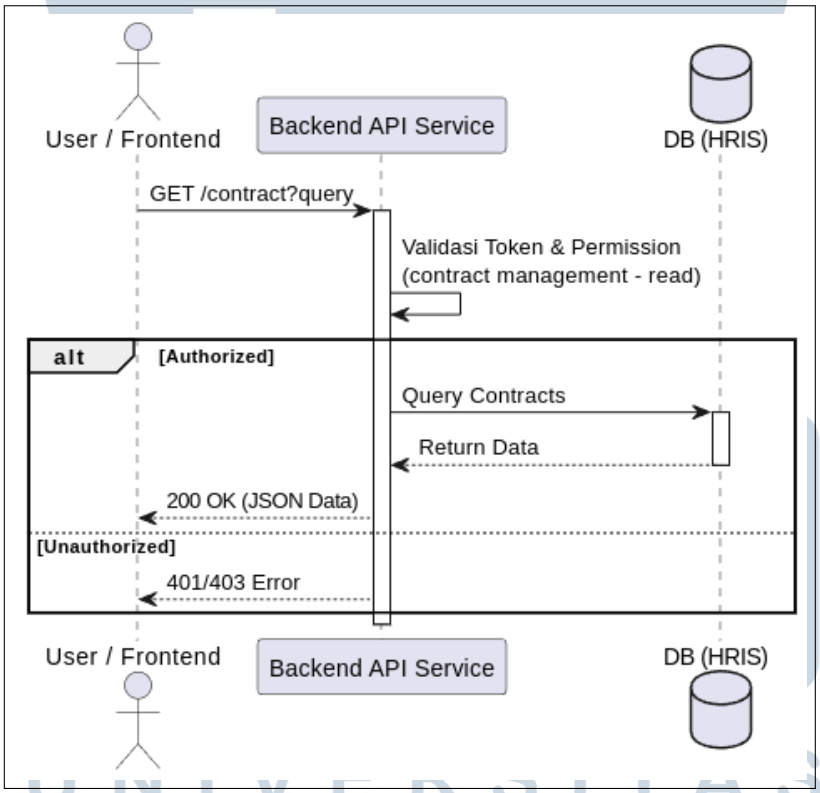
Minggu Ke -	Pekerjaan yang dilakukan
1	*Onboarding* dan developing API contract
2	Developing API contract dan refactor reimbursement
3	Refactor absensi API
4	Melanjutkan refactor absensi API, refactor upload file, dan cronjob untuk absensi
5	Melanjutkan refactor absensi API dan cronjob untuk contract
6	Melanjutkan refactor absensi API dan membuat api untuk payslip date
7	Melanjutkan payslip date, menambahkan zone waktu pada beberapa fitur, dan menambahkan sistem file upload pada leave permit
8	Memperbaiki bug pada payslip date dan pada beberapa fitur
9	Menambahkan zona waktu pada cron job
10	Menambahkan edge cases pada fitur absensi dan payslip date, serta refactor user API. dan memperbaiki permission
11	Mengubah nama table dan struktur kodenya dan integrasi socket pada beberapa fitur

3.4 Perancangan dan Pengembangan Sistem

Perancangan dan pengembangan sistem HRIS di PT Visi Karya Nusantara akan menjelaskan perancangan sistem untuk fitur-fitur utama yang dikembangkan selama magang, yaitu sistem *Contract*, sistem *Payslip date*, sistem *Attendance date*, pengembangan sistem *Daily Attendance*, dan mengimplementasikan Socket.io pada backend.

3.4.1 Perancangan Arsitektur Sistem

A Sequence Diagram Endpoint List Contract



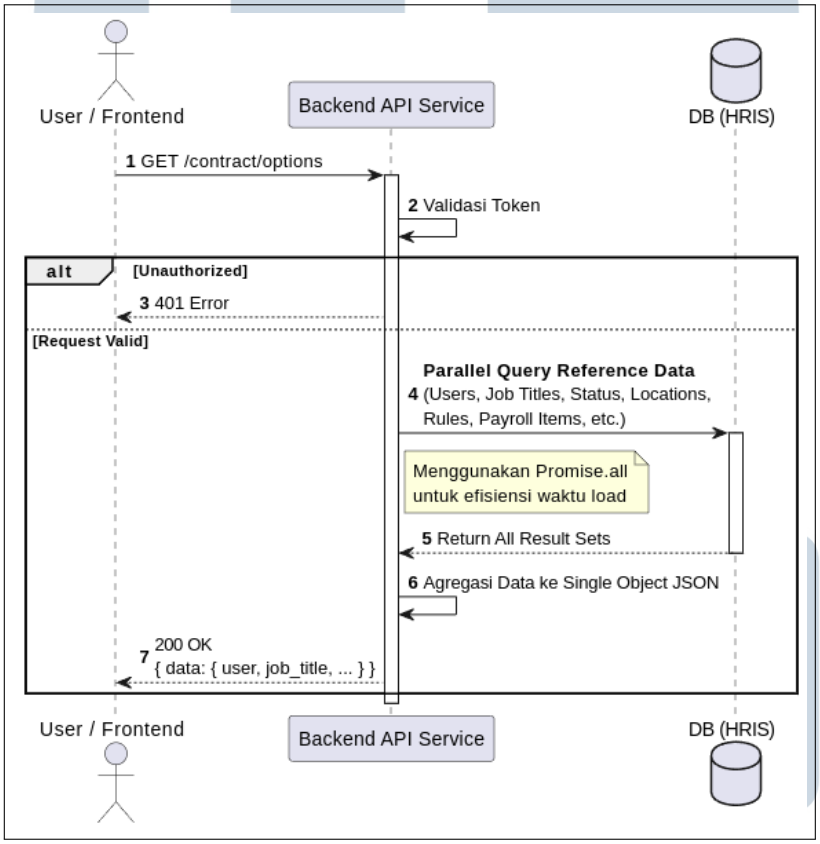
Gambar 3.2. List Contract

Pada gambar 3.2, merupakan alur arsitektur dari endpoint `GET /contract/`, dimana alur proses dimulai ketika *frontend* mengirimkan *request* HTTP `GET /contract/` beserta parameternya. Sebelum memproses data, pada *backend* melakukan validasi terlebih dahulu dan juga memvalidasi *permission* untuk melihat data kontrak ini. Jika, validasi gagal, sistem akan langsung mengembalikan respon

error dengn kode 401 ataupun 403, tanpa melanjutkan proses pengambilan data lagi.

Namun jika lolos validasi, *backend* akan menjalankan logika bisnis dengan paramter yang diberikan dari *frontend*, dimana akan mengambil data-data kontrak yang ada dari *database* di mana akan diberikan dalam bentuk json data dengan status kode 200.

B Sequence Diagram Endpoint GET Asset Contract



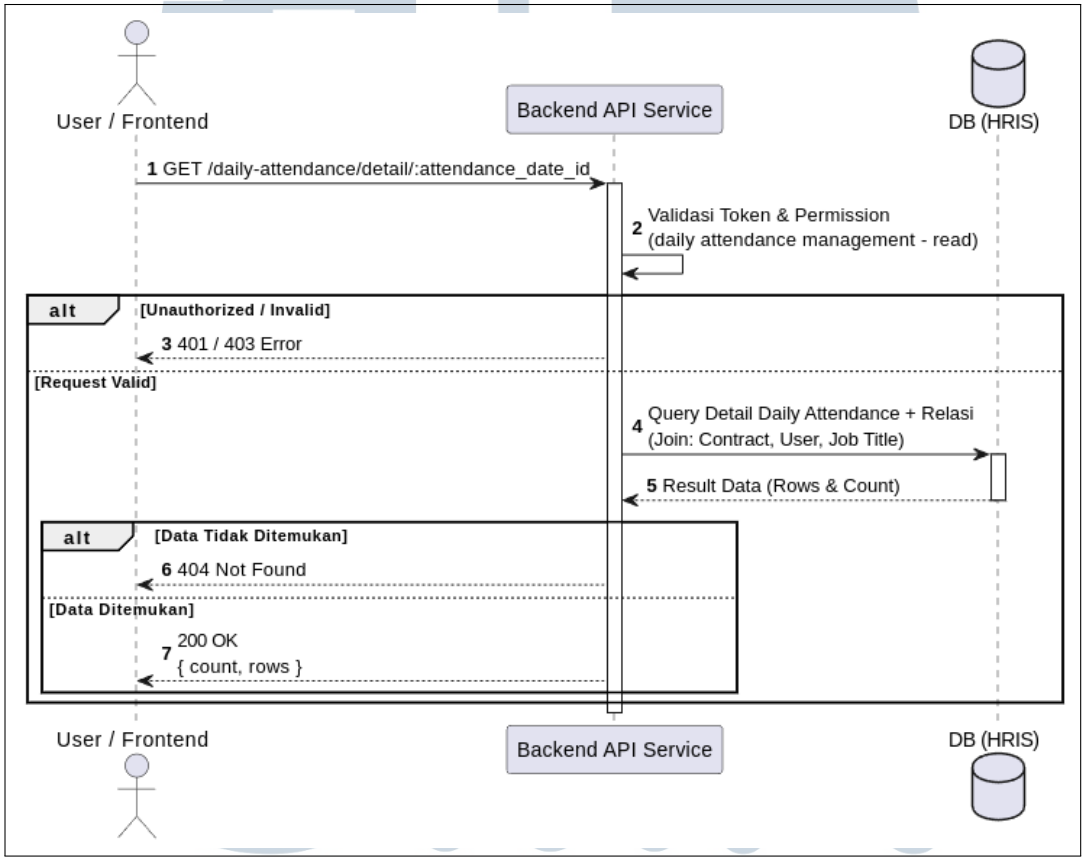
Gambar 3.3. Asset Contract

Pada gambar 3.3, merupakan proses dari endpoint `/contract/options` dengan metode GET. Dimana ketika *frontend* menembak *endpoint* tersebut. Pada sisi *backend* akan dilakukan validasi autentikasi. Jika autentikasi itu tidak valid maka akan langsung diberikan status kode 401 dengan pesan errornya.

Namun jika berhasil, sistem *backend* akan melakukan queri data-data referensi sebelum pembuatan kontrak, di mana queri-queri ini dilakukan secara paralel menggunakan `promise.all` untuk efisiensi waktu. Di mana, tabel-tabel

yang di referensikan di antara lain, *users, job title, location, payroll item*. Setelah semua data-data berhasil di dapatkan, dilakukan agregasi data, di mana agregasi data ini agar data-data yang telah diambil melalui database dapat di ringkas dalam satu single object json yang bertujuan untuk mempermudah *frontend* dalam melakukan integrasi data.

C Sequence Diagram Endpoint Detail Attendance Date

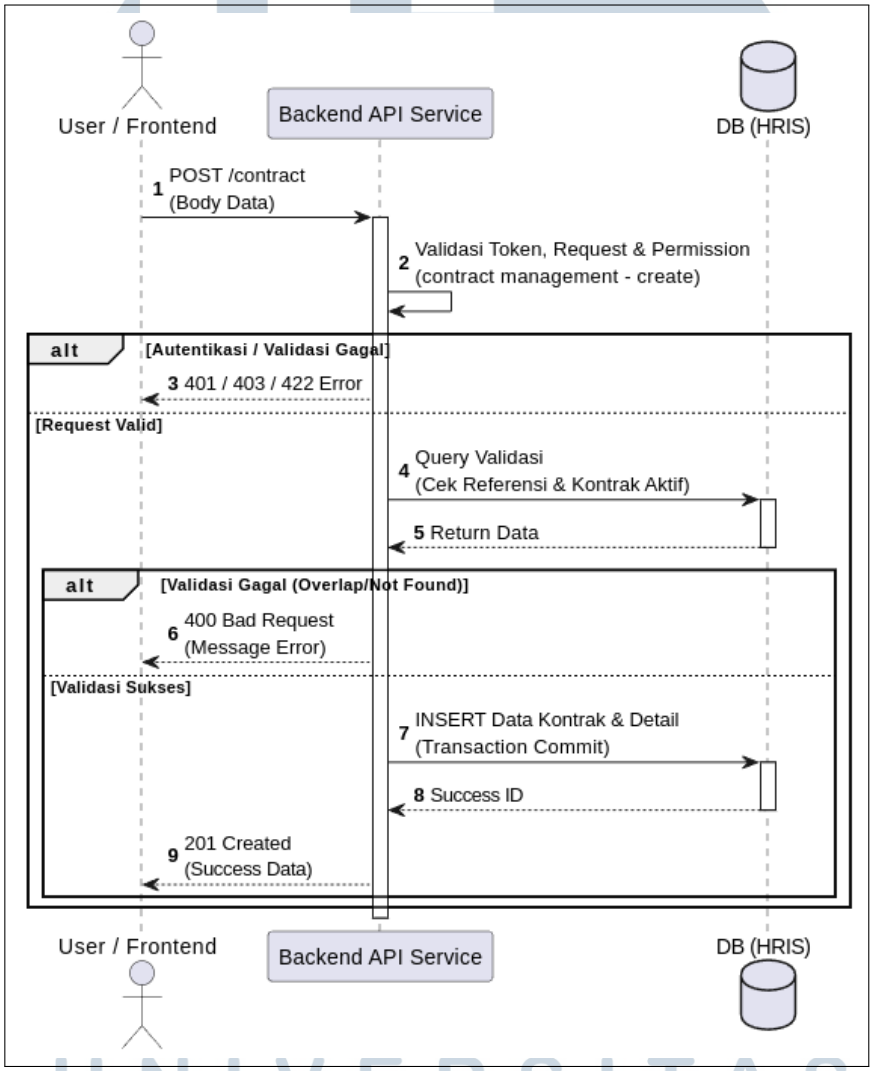


Gambar 3.4. Detail Attendance Date

Pada gambar 3.4, merupakan proses arsitektur dari *endpoint* `daily-attendance/detail/:id`. Di mana, proses ini dilakukan ketika *frontend* melakukan *request* terhadap *endpoint* tersebut dan *backend* akan melakukan validasi autentikasi dan *permission* untuk memastikan bahwa dari sisi *frontend* atau pengguna memiliki *permission* yang bersangkutan. Jika berhasil sistem akan melakulakan queri data ke database pada tabel *daily attendance* dengan relasi-relasinya menggunakan `JOIN` ke tabel-tabel *contract, user,* dan *job title*. Data yang

dikembalikan akan berbentuk *count* dan *rows*. Setelah itu, data dikirimkan lagi ke *frontend* dalam bentuk json dengan kode 200.

D Sequence Diagram Endpoint Create Contract

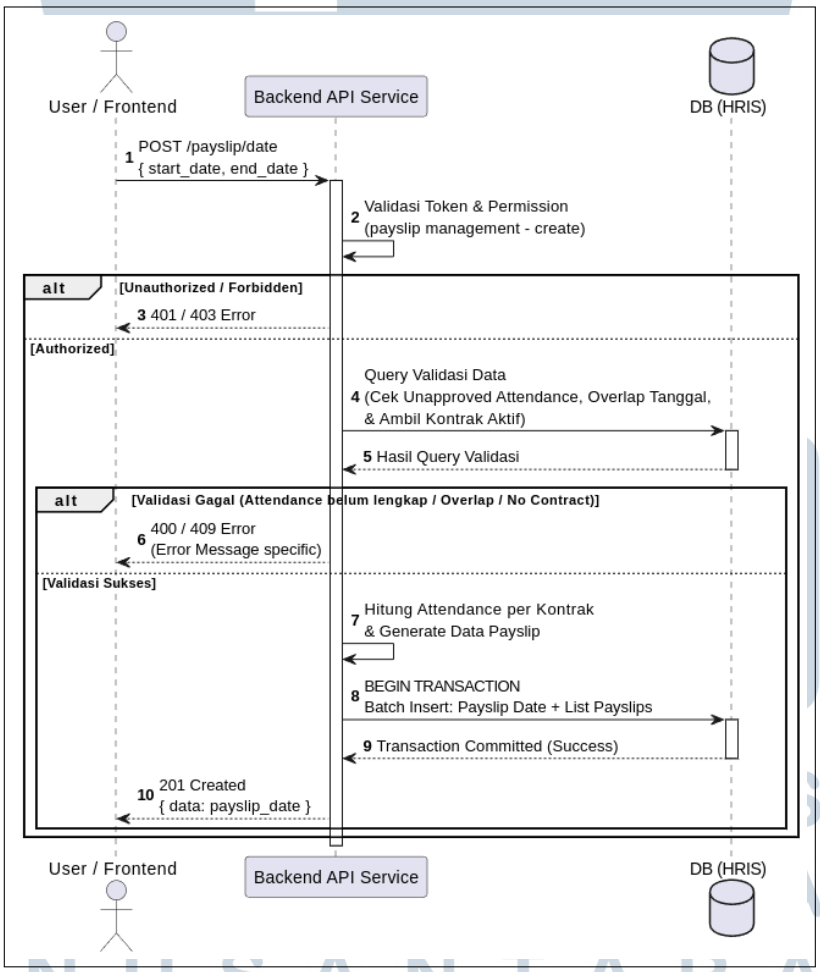


Gambar 3.5. Create Contract

Pada gambar 3.5, merupakan arsitektur dari pembuatan kontrak terhadap *endpoint* `/contract` dengan metode POST. *Frontend* akan mengirimkan data-data yang akan di submit, lalu *backend* akan melakukan validasi autentikasi, request, dan *permission* sebelum melakukan proses data. Dimana jika salah satu mengalami gagal maka akan dikirimkan kode status 401, 403, maupun 422 dengan pesan-pesan errornya. Namun jika berhasil data-data yang diberikan oleh *frontend* akan

dilakukan validasi lagi terhadap tabel-tabel referensi yang ada, dan dilakukan pengecekan kontrak untuk kontrak yang dibuat. Jika validasi maupun kontrak yang dibuat mengalami kegagalan maka akan dikirimkan pesan error dengan kode 400. Namun jika semua data berhasil lolos dari validasi, semua data akan dimasukkan kedalam database menggunakan *transaction*, *transaction* digunakan untuk mengantisipasi terjadinya kegagalan ketika menginput data di level database [8]. Jika semua berhasil *backend* akan mengirimkan kode 201 dengan pesan berhasil ke *frontend*.

E Sequence Diagram Endpoint Create Payslip Date

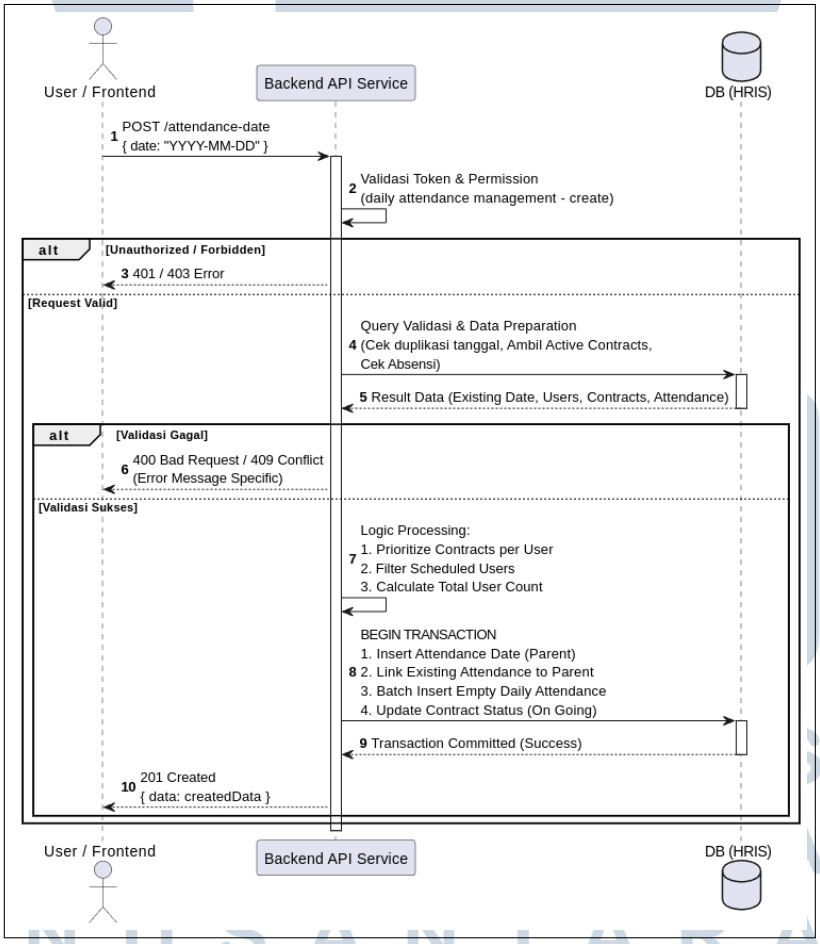


Gambar 3.6. Create Payslip Date

Pada gambar 3.6, merupakan proses dari pembuatan payslip date, dimana *frontend* akan mengirimkan dua *request* yaitu `start_date` dan `end_date`, lalu

dilakukan lah validasi autentikasi serta *permission*. Jika berhasil maka *backend* akan melakukan pengecekan terhadap rentang tanggal yang telah diberikan oleh *frontend* untuk mengecek *attendance*, tanggal yang *overlap*, dan mengecek kontrak yang aktif. Jika salah satu validasi gagal akan mengembalikan kode 400 ataupun 409 dengan pesan error masing-masing, jika berhasil sistem akan menghitung *attendance* per kontrak yaung aktif dan melakukan *generate* data untuk setiap payslip yang dibuat. Lalu data-data tersebut dimasukkan ke database menggunakan *transaction*, kedua table yang berbeda yaitu `payslip_date` dan `payslip`. Setelah itu, respon akan dikembalikan ke *frontend* dengan kode 201 berupa datanya.

F Sequence Diagram Endpoint Create Attendance Date



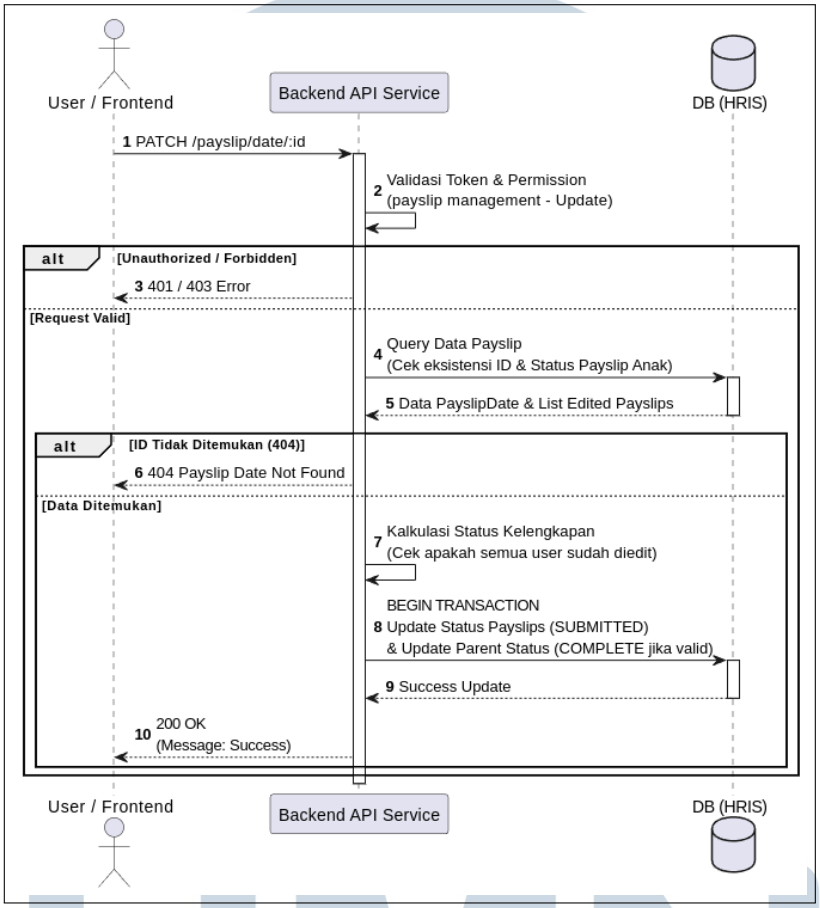
Gambar 3.7. Create Attendance Date

Pada gambar 3.7, merupakan arsitektur dari pembuatan *attendance date* di mana *frontend* mengirim *request* `date` dalam format `YYYY-MM-DD`, Lalu *backend*

akan memvalidasi autentikasi dan *permisson* terlebih dahulu sebelum melakukan proses data. Jika terjadi error akan dikembalikan kode 401 atau 403 dengan pesan errornya, Namun jika validasi berhasil maka akan dilakukan pengecekan lagi terhadap tanggal yang telah dibuat, juga akan dilakukan pengecekan kontrak yang aktif, serta data absensi pada tanggal tersebut ke database. Jika berhasil, sistem akan melakukan prioritas kontrak per masing-masing pengguna untuk mengatasi jika kontrak ada yang lebih dari satu disetiap pengguna. Dan akan mengambil jadwal dari kontrak pengguna tersebut untuk hari atau tanggal yang dikirimkan oleh *frontend.* Lalu, akan dilakukan perhitungan total pengguna yang akan dibuat. Untuk memastikan jumlah yang dibuat dengan data yang terbuat sama.

Setelah itu, dilakukan *insert* data ke database ke dalam tabel `daily_attendance` dan `attendance_date`, jika kontrak yang pengguna itu masi baru akan dilakukan juga pembaruan status terhadap kontrak tersebut menjadi *on going*. Setelah semua berhasil, status kode berserta pesan maupun data dikirimkan balik ke *frontend*.

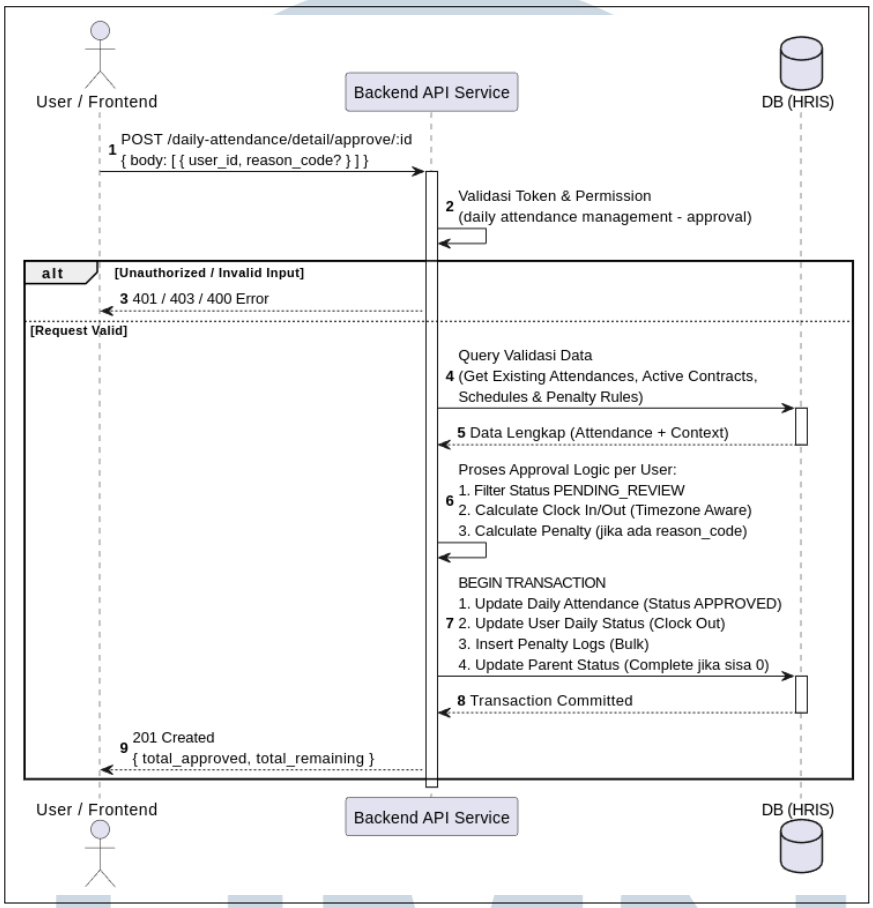
G Sequence Diagram Endpoint Update Payslip Date



Gambar 3.8. Update Payslip Date

Pada gambar 3.8, merupakan arsitektur proses dari *approval payslip date* menggunakan metode PATCH. Pada arsitektur ini *frontend* hanya perlu menembak *endpoint* ini tanpa harus mengirim *request* karna semuanya dilakukan dari sisi *backend*. Dimana jika sudah berhasil melewati validasi token dan *permission*, akan dilakukan queri data *payslip* dengan status dari *payslip* tersebut berdasarkan *payslip date* nya, dimana jika masing-masing *payslip* telah diedit dari sisi *frontend* maka *payslip* akan memiliki nilai *true* terhadap atribut `edited`. Ketika, nilai *true* ini telah mencakup ke semua *payslip* maka status dari *payslip* tersebut akan diperbarui menjadi *submitted*, di mana jika jumlah payslip yang telah diedit dengan jumlah pengguna yang dibuat sama maka status dari *payslip date* ini akan berubah menjadi *complete* menandakan bahwa *payslip* untuk bulan tersebut sudah selesai dan semua payslip dikirimkan ke masing-masing pengguna.

H Sequence Diagram Endpoint Approval Attendance Date



Gambar 3.9. Approve Attendance Date

Pada gambar 3.9, merupakan proses dari *approval* absensi harian, dimana menggunakan metode POST dan *frontend* akan mengirimkan id dari *user-user* dan jika ada *reason* maka *frontend* harus memasukkan kode dari masing-masing *reason* tersebut kedalam *request*. *reason-reason* ini berkaitan dengan keterlambatan dari setiap pengguna. Dimana akan dilakukan queri validasi data terhadap jam pengguna yang masuk berdasarkan jadwal dari kontrak yang aktif dari setiap pengguna dengan penalti-penalti dari kontrak tersebut. *Approval* hanya dapat dilakukan untuk absensi-absensi yang telah berstatus *pending* dan juga dilakukan normalisasi waktu berdasarkan lokasi kontrak yang dibuat dan juga penalti yang berasal dari *reason code*. Setelah berhasil akan dilakukan menginputan ke database menggunakan *transaction* dengan mengupdate status dari setiap absensi ke status *approved* dan juga memasukkan penalti dari setiap pengguna jika melakukan keterlambatan, jika

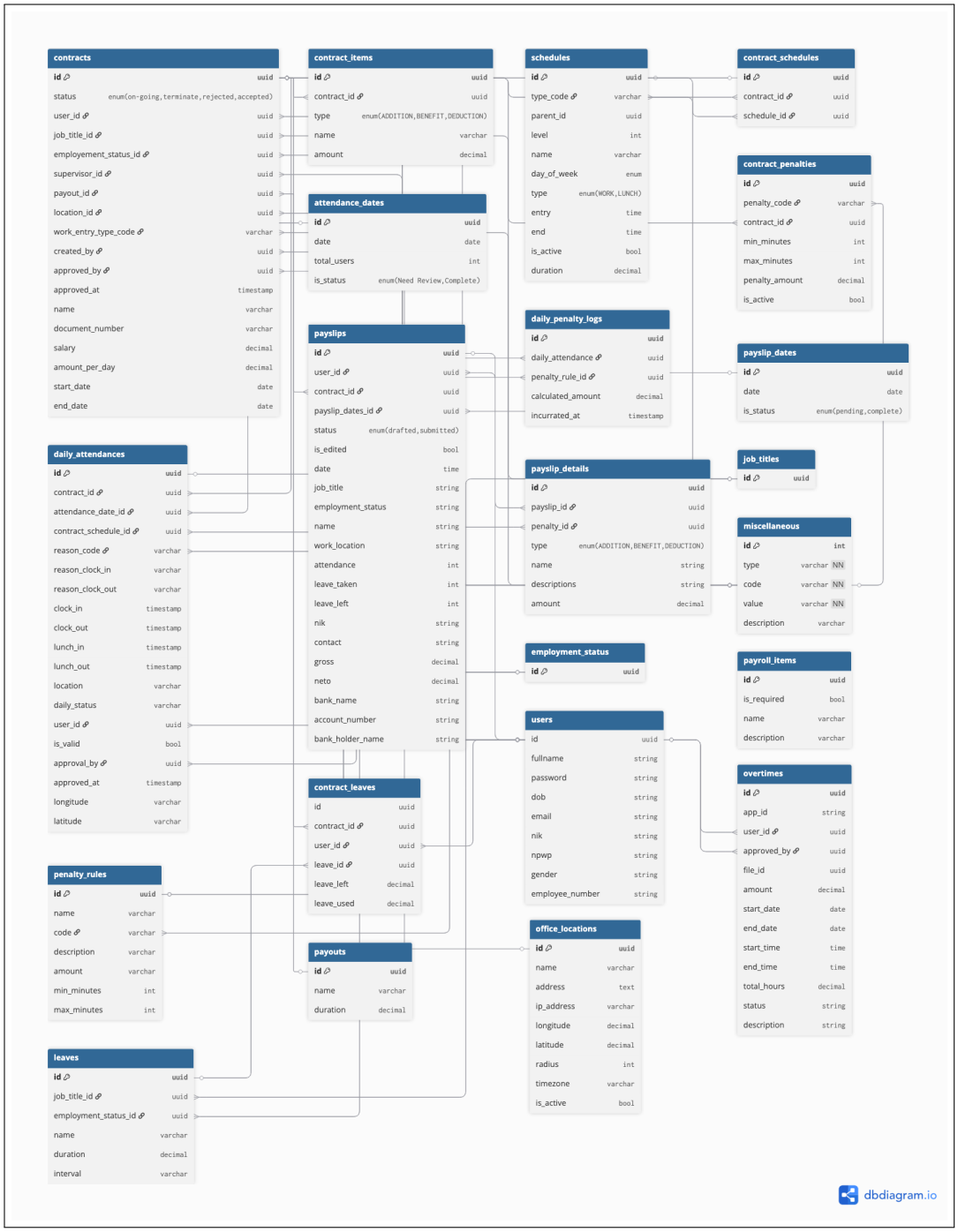
seluruh pengguna telah di *approve* maka *attendance date* status akan menjadi *complete*. Dan sistem akan mengembalikan kode 201 beserta pesan maupun datanya.

3.4.2 Perancangan Database

Perancangan *database* merupakan tahapan krusial dalam pengembangan sistem backend HRIS PT Visi Karya Nusantara. Pada tahap ini, struktur *database* dirancang untuk memastikan kekonsistenan data, menghindari redudansi data, dan mengoptimalkan performa kueri agar akses data menjadi lebih efisien [9]. Untuk memudahkan pemahaman terhadap struktur data yang kompleks, pembahasan skema *database* dibagi menjadi dua bagian: skema relasi global (keseluruhan) dan detail skema per modul.

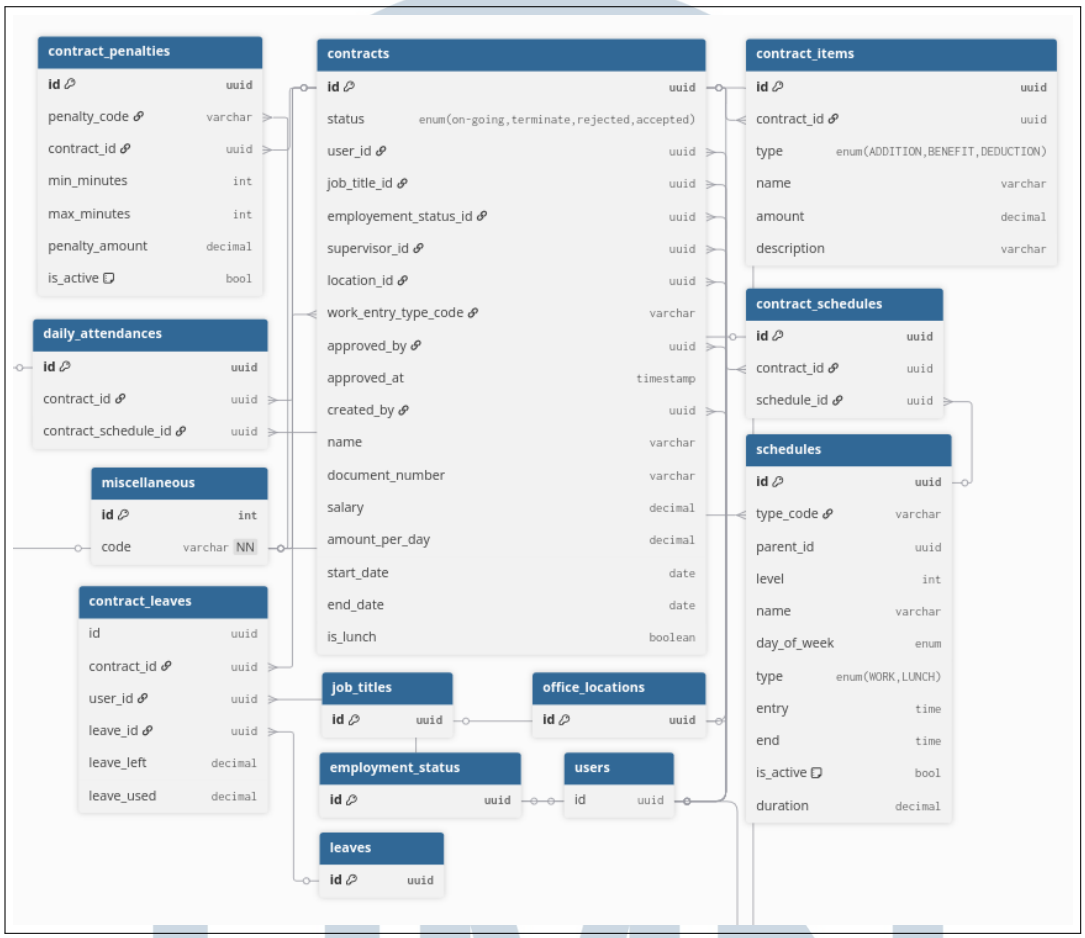
A Skema Relasi Global

Gambar 3.10 menampilkan relasi antar seluruh entitas dalam sistem. Pada skema ini, tabel `users` bertindak sebagai entitas pusat yang menghubungkan modul *Contract*, *Attendance*, dan *Payslip*. Hal ini memastikan bahwa setiap data transaksi (seperti gaji atau absen) selalu tervalidasi kepemilikannya terhadap satu karyawan yang unik.



Gambar 3.10. Skema Relasi Global

B Detail Skema Contract Modul



Gambar 3.11. Detail Relasi Contract

Pada Gambar 3.11 merupakan skema dari database *contract* dan terdiri dari tabel-tabel:

1. contracts: tabel ini merupakan tabel utama dari fitur *contract*, yang digunakan untuk menyimpan data-data kontrak untuk setiap karyawan seperti gaji, durasi kontrak, status kontrak, dan lain-lain. Yang di mana memiliki beberapa relasi antar table lain, seperti:
   - *Many-to-One*: terhubung ke tabel `users`, `job_titles`, `employment_statuses`, `office_locations`, dan `miscellaneous`.
   - *One-to-Many*: terhubung ke tabel `contract_items`, `contract_leaves`, dan `contract_penalties`.

- *Many-to-Many*: terhubung ke tabel `contract_schedules` yang berelasi dengan tabel `schedules`.

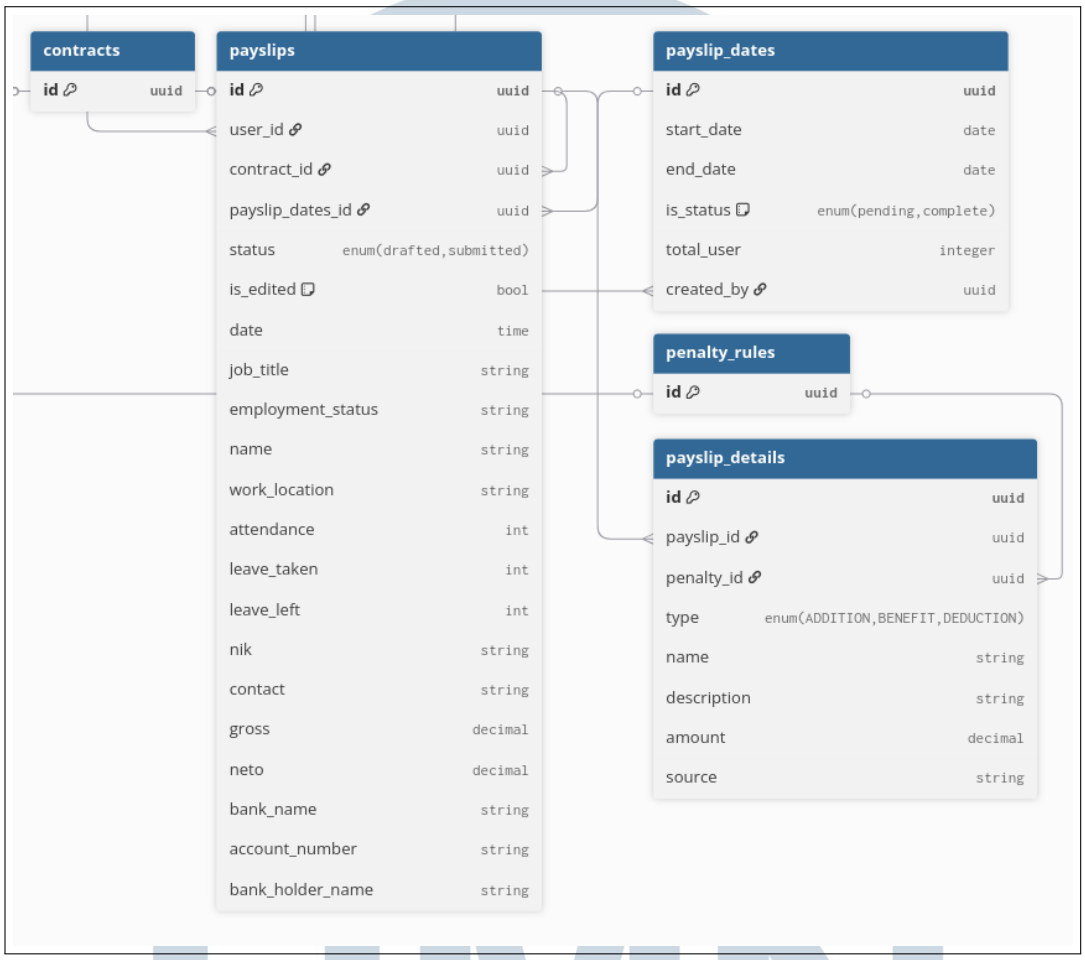
2. contract_items: digunakan untuk menyimpan kebutuhan seperti *benefit*, *addition*, *deduction*. Yang dimana data-data ini akan digunakan pada perhitungan gaji bulanan karyawan.

3. contract_schedules: digunakan untuk mengaitkan jadwal kerja karyawan dengan kontrak dipilih, dimana jadwal kerja yang terpilih akan dijadikan sebagai acuan sistem untuk karyawan melakukan absen setiap hari.

4. contract_penalties: digunakan untuk menyimpan data-data penalti karyawan seperti denda keterlambatan, denda jika tidak masuk.

5. contract_leaves: digunakan untuk menyimpan data perizinan untuk karyawan jika ingin mengajukan hal seperti mengambil cuti, sakit, *work from home*, dan lainnya.

C Detail Skema Payslip Date Modul



Gambar 3.12. Detail Relasi Payslip Date

Pada Gambar 3.12 merupakan skema dari database *payslip date* dan terdiri dari tabel-tabel:

1. payslips: tabel ini merupakan tabel utama untuk menyimpan data-data payslip, dimana data-datanya didapat dari berbagai macam tabel salah satunya tabel `users`. pada tabel ini memiliki relasi `One-to-Many` pada tabel `payslip_dates`, `contracts`, `users`.
2. payslip_dates: tabel ini merupakan tahap sebelum pengguna dapat melihat maupun membuat payslip, sehingga harus memilih tanggal paylip date yang diinginkan. pada tabel ini memiliki relasi `Many-to-One` pada tabel `users`

3. payslip_details: tabel ini merupakan penyimpanan data-data tambahan dari tabel lain untuk tabel payslip, seperti: `addition`, `benefit`, maupun `deduction`, dan lainnya. Dengan jumlah dan sumbernya tabel berasal. pada tabel ini memiliki relasi `Many-to-One` pada tabel `payslips`

D Detail Skema Attendance Dates Modul

Gambar 3.13. Detail Relasi Attendance Dates

Pada Gambar 3.13 merupakan skema dari database *attendance date* dan terdiri dari tabel-tabel:

1. daily_attendances: tabel ini merupakan tabel utama untuk menyimpan data-data absen pengguna, dimana data-datanya akan terbuat jika pengguna melakukan *clock-in, clock-out, lunch-in*, dan *lunch-out*.

2. attendance_dates: tabel ini bertujuan untuk membuat tanggal yang dimana ini merupakan tahap sebelum melakukan review absensi pada karyawan yang telah melakukan absen.

3. daily_penalty_logs: tabel ini bertujuan untuk menyimpan nilai dari denda yang dilakukan oleh karyawan, ketika tahap review absensi oleh *supervisor*.

4. penalty_rules: tabel ini merupakan tabel yang telah diinputkan nilainya dari awal, yang bertujuan ketika `reason_code` pada tabel daily attendance dipilih dengan data yang sama pada penalty rules. Maka, nilai `amount` akan dimasukkan ke tabel daily penalty logs

3.4.3 Spefisikasi API

Pada pengembangan sistem HRIS PT Visi Karya Nusantara, komunikasi antara sisi server dan aplikasi klien dilakukan melalui *Application Programming Interface* (API). Spesifikasi API ini dirancang menggunakan arsitektur REST yang memanfaatkan metode HTTP standar (GET, POST, PUT, DELETE) untuk menjalankan operasi CRUD. Bagian ini menjabarkan seluruh *endpoint-endpoint* yang telah dirancang selama magang berlangsung.

A Spesifikasi API Contract

Berikut merupakan *endpoint-endpoint* yang dirancang untuk *Contract* pada tabel 3.2.

Tabel 3.2. 7 Endpoint yang digunakan untuk Fitur Contract

No	Endpoint	Method	Request	Response
1	`/contract/`	GET	–	`status code`, `message`, `data`
2	`/contract/options`	GET	–	`status code`, `message`, `data`
3	`/contract/`	POST	`data`	`status code`, `message`
4	`/contract/:id`	PUT	`data`	`status code`, `message`
5	`/contract/user`	GET	–	`status code`, `message`, `data`

Keterangan dari masing-masing endpoint sebagai berikut:

1. **/contract/ (GET)** – Menampilkan seluruh data kontrak yang ada.
2. **/contract/options (GET)** – Mengambil aset untuk pembuatan kontrak.
3. **/contract/ (POST)** – Membuat data kontrak baru ke dalam sistem.
4. **/contract/:id (PUT)** – Memngupdate data kontrak yang sudah ada berdasarkan `id`.

5. **/contract/user (GET)** – Menampilkan kontrak yang terkait dengan pengguna.

Endpoint yang memerlukan parameter `id`, nilai `id` akan dikirim melalui parameter URL dengan nilai yang berupa `uuid` dari tabel *contracts*. Lalu pada kolom body (request), yang memiliki `request data` akan mengirimkan atribut-atribut diantaranya: `user_id`, `name`, `job_title_id`, dan lainnya untuk melihat lebih detail atribut yang digunakan dapat dilihat pada hasil implementasi dari contract. Lalu untuk response data, akan dilakukan penggabungan data dari tabel-tabel relasi yang telah dibuat pada gambar 3.11 skema database contract.

B Spesifikasi API Payslip Date

Berikut merupakan *endpoint-endpoint* yang dirancang untuk *payslip date* pada tabel 3.3.

Tabel 3.3. 5 Endpoint yang digunakan untuk Fitur payslip date

No	Endpoint	Method	Body (Request)	Response (body)
1	`/payslip`	GET	–	`status code, message, data`
2	`/payslip/date/`	POST	`request data`	`status code, message`
3	`/payslip/date/:id`	DELETE	–	`status code, message`
4	`/payslip/date/:id`	PATCH	`request data`	`status code, message`
5	`/payslip/date/:id`	GET	–	`status code, message, data`

Keterangan dari masing-masing endpoint sebagai berikut:

1. **/payslip (GET)** – Menampilkan seluruh data tanggal payslip date yang ada.
2. **/payslip/date/ (POST)** – Membuat data payslip date dari rentang tanggal.
3. **/payslip/date/:id (DELETE)** – Menghapus payslip date.
4. **/payslip/date/:id (PATCH)** – Mengupdate data payslip date dan payslip.

5. **/payslip/date/:id (GET)** – Menampilkan detail data payslip date.

Pada *endpoint* yang memerlukan parameter `id`, nilai `id` akan dikirimkan melalui parameter URL dengan nilai berupa `uuid` dari tabel *payslip_dates*.

C Spesifikasi API Attendance Dates

Berikut merupakan *endpoint-endpoint* yang dirancang untuk *attendance date* pada tabel 3.4.

Tabel 3.4. 5 Endpoint yang digunakan untuk fitur attendance date

No	Endpoint	Method	Body (Request)	Response (body)
1	`/attendance-dates/`	POST	data	`status code, message`
2	`/attendance-dates/`	GET	–	`status code, message, data`
3	`/attendance-dates/:id`	DELETE	–	`status code, message`
4	`/daily-attendances/ detail/:id`	GET	–	`status code, message`
5	`/daily-attendances/ detail/approve/:id`	POST	`data`	`status code, message`
6	`/daily-attendances/ attendances/`	POST	`data`	`status code, message`

Keterangan dari masing-masing endpoint sebagai berikut:

1. **/attendance-dates/ (POST)** – Membuat data attendance date dari rentang tanggal
2. **/attendance-dates/ (GET)** – Menampilkan seluruh data tanggal attendance date yang ada.
3. **/attendance-dates/:id (DELETE)** – Menghapus attendance date, dengan data daily attendance yang terhubung ke id tersebut.
4. **/daily-attendances/detail/:id (GET)** – Menampilkan detail data daily attendance dari attendance date id yang dipilih.

5. **/daily-attendances/detail/approve/:id (POST)** – Mengupdate data daily attendance, attendance date, dan data penalty setiap karyawan.

Pada *endpoint* yang memerlukan parameter `id`, nilai `id` akan dikirimkan melalui parameter URL dengan nilai berupa `uuid` dari tabel *attendance_dates*.

3.4.4 Pengembangan Daily Attendance

Endpoint `/daily-attendances/attendances/` digunakan oleh karyawan untuk melakukan proses *clock-in*, *clock-out*, *lunch-in*, dan *lunch-out* secara mandiri. Pada endpoint ini sistem akan mencari kontrak aktif karyawan yang bertujuan untuk melihat jadwal kerja karyawan pada hari tersebut.

Di mana fitur ini juga akan mengubah status kontrak kerja karyawan dari *approved* menjadi *on-going*. Endpoint ini terintegrasi secara langsung dengan fitur attendance date karna data yang telah tercatat akan di review yang telah dijelaskan di sub bab sebelumnya.

```
attendDaily = async (userInfo: any, body: IDailyAttendance) => {
    const nowUtc = new Date();

    const t = await sequelize.transaction();
    try {
        this.validateCheckInRequest(body);

        //ambil schedules dari relasi belongsToMany dengan filter hari ini
        const getContract = await this.ContractDao.findActiveContract(userInfo.id, {
            include: [
                {
                    model: models.schedule,
                    as: 'schedules',
                    through: { attributes: [] },
                },
                {
                    model: models.office_location,
                    attributes: ['timezone'],
                    required: true,
                },
            ],
        });

        if (!getContract) {
            throw new ApiError(httpStatus.BAD_REQUEST, 'Active contract not found for user');
        }
```

Gambar 3.14. Potongan kode fungsi inisialisasi attendDaily

Pada gambar 3.14 merupakan potongan kode dari attendDaily di mana setiap karyawan atau pengguna, yang akan menggunakan endpoint ini. Sistem akan selalu mengecek kontrak mereka untuk mengambil jadwal serta lokasi kerjanya.

```
case SCHEDULE.ACTION.CLOCK_IN: {
    const existingAttendance = attendance;
    if (existingAttendance?.clock_in) {
        throw new ApiError(httpStatus.CONFLICT, 'Already clocked in');
    }

    if (!selectedSchedule) {
        throw new ApiError(httpStatus.NOT_FOUND, 'Schedule not found for today');
    }

    // hitung waktu mulai dan selesai sesuai jadwal
    const { entryDateTime, endDateTime } = this.buildScheduleWindow(
        now,
        selectedSchedule
    );

    // validasi jam clock-in
    this.validateClockInWindow(
        now,
        entryDateTime,
        endDateTime,
        body.reason_clock_in
    );

    // simpan atau update data absensi karyawan
    const data = await this.saveClockInAttendance({
        existingAttendance,
        nowUtc,
        body,
        userInfo,
        entryDateTime,
        transaction: t,
        contractId: getContract.id,
    });

    // ubah status kontrak jika masih dalam tahap approved
    await this.ensureContractOnGoing(getContract, t);

    result = {
        statusCode: httpStatus.OK,
        message: 'You have successfully clocked in.',
        data,
    };
    break;
}
```

Gambar 3.15. Potongan kode fungsi utama attendDaily

Pada gambar 3.15 merupakan bagian utama dari logika *clock-in* yang dipendekkan dalam fungsi `attendDaily` dimana logika ini juga berlaku untuk `lunch-in`, `lunch-out`, dan `clock-out`. Dimana sistem menghitung waktu

sekarang dengan jadwal yang dipilih dari kontrak berdasarkan harinya. Lalu memvalidasi apakah karyawan telat, absen terlalu awal, ataupun absen setelah jam kerja berakhir. Jika validasi berhasil, data akan disimpan pada tabel `daily_attendances` dan jika karyawan baru memiliki kontrak dengan status *approved* kontrak akan di update juga agar statusnya menjadi *on-going*.

3.4.5 Implementasi Socket

Pada backend, telah mengimpelementasikan socket yang bertujuan untuk komunikasi *real-time* antar client dan server menggunakan Socket.IO [10]. Kode 3.1 dapat dilihat dibawah ini.

```
export const initSocket = (server: HttpServer): Server => {
    io = new Server(server, {
        cors: {
            origin: '*',
        },
    });

    io.use(async (socket: Socket, next) => {
        try {
            let token = socket.handshake.auth.token; //access token dikirim dari Frontend

            if (!token) { //for backend testing in apidog
                const authHeader = socket.handshake.headers.authorization || '';
                if (authHeader.startsWith('Bearer ')) {
                    token = authHeader.substring(7);
                }
            }

            if (!token) {
                return next(new ApiError(httpStatus.FORBIDDEN, 'Token not provided'));
            }

            const payload: any = jwt.verify(token, config.jwt.secret); //memverifikasi token menggunakan jwt secret key

            //memastikan token adalah access token
            if (payload.type !== tokenTypes.ACCESS) {
```

```
                return next(new ApiError(httpStatus.BAD_GATEWAY, '
    Invalid token type'));
            }

            const user = await userDao.findOne({
                where: { id: payload.sub },
                attributes: ['id'],
            });

            if (!user) {
                return next(new ApiError(httpStatus.NOT_FOUND, '
    User not found'));
            }

            //simpan data user ke objek socket
            socket.data.user = { id: user.id } as
    AuthenticatedUser;

            next();
        } catch (e: any) {
            console.error('Socket authentication failed:', e.
    message);
            return next(new ApiError(httpStatus.BAD_GATEWAY, '
    Authentication Error:'));
        }
    });

    io.on('connection', (socket: Socket) => {
        const user = socket.data.user as AuthenticatedUser;
        socket.join(`user:${user.id}`); //bisa join ke socket
        socket.on('disconnect', () => {
            console.log(`User disconnected: ${user.id}`);
        });
    });

    return io;
};
```

Kode 3.1: "Inisialisasi socket pada backend"

Pada kode diatas, sebelum koneksi socket diterima, sistem akan melalukan proses autentikasi menggunakan `access token` yang dimana token tersebut diverifikasi menggunakan `secret key` yang ada diserver. Jika, token valid berasal

dari server dan akun dari pengguna ada. maka socket akan terhubung ke user tersebut.

Salah satu fitur yang telah megimplementasikan socket adalah fitur event, dimana ketika pembuatan event telah berhasil maka socket akan mengirimkannya ke seluruh client yang telah konek ke dalam socket. Pada kode 3.2 merupakan potongan kode dari event yang telah mengimplementasikan socket.

```
await t.commit();

const result = await this.EventDao.findById(newEvent.id);

const payload = result?.toJSON === 'function' ? result.toJSON() :
    result;
getSocket().emit('event:created', payload);

return result;
```

Kode 3.2: "Implementasi socket pada event"

Pada kode diatas merupakan implementasi socket pada fungsi create event, dimana ketika seluruh proses pembuatan event berhasil dibuat. Maka socket akan mengirimkan sinyal bernama `event:created`, dimana sinyal ini nantinya akan ditangkap dari sisi client. Agar pembuatan event yang telah berhasil dapat masuk secara real-time.

3.5 Hasil Implementasi

Pada hasil implementasi fitur-fitur yang sudah di rancang maupun dikembangkan akan menunjukan *request body* serta *response body* dari API-API yang telah di buat.

3.5.1 Contract

Berikut ini merupakan request body untuk membuat *contract* ataupun melakukan update pada *contract*, yang dapat dilihat pada request di bawah ini atau pada gambar 3.16:

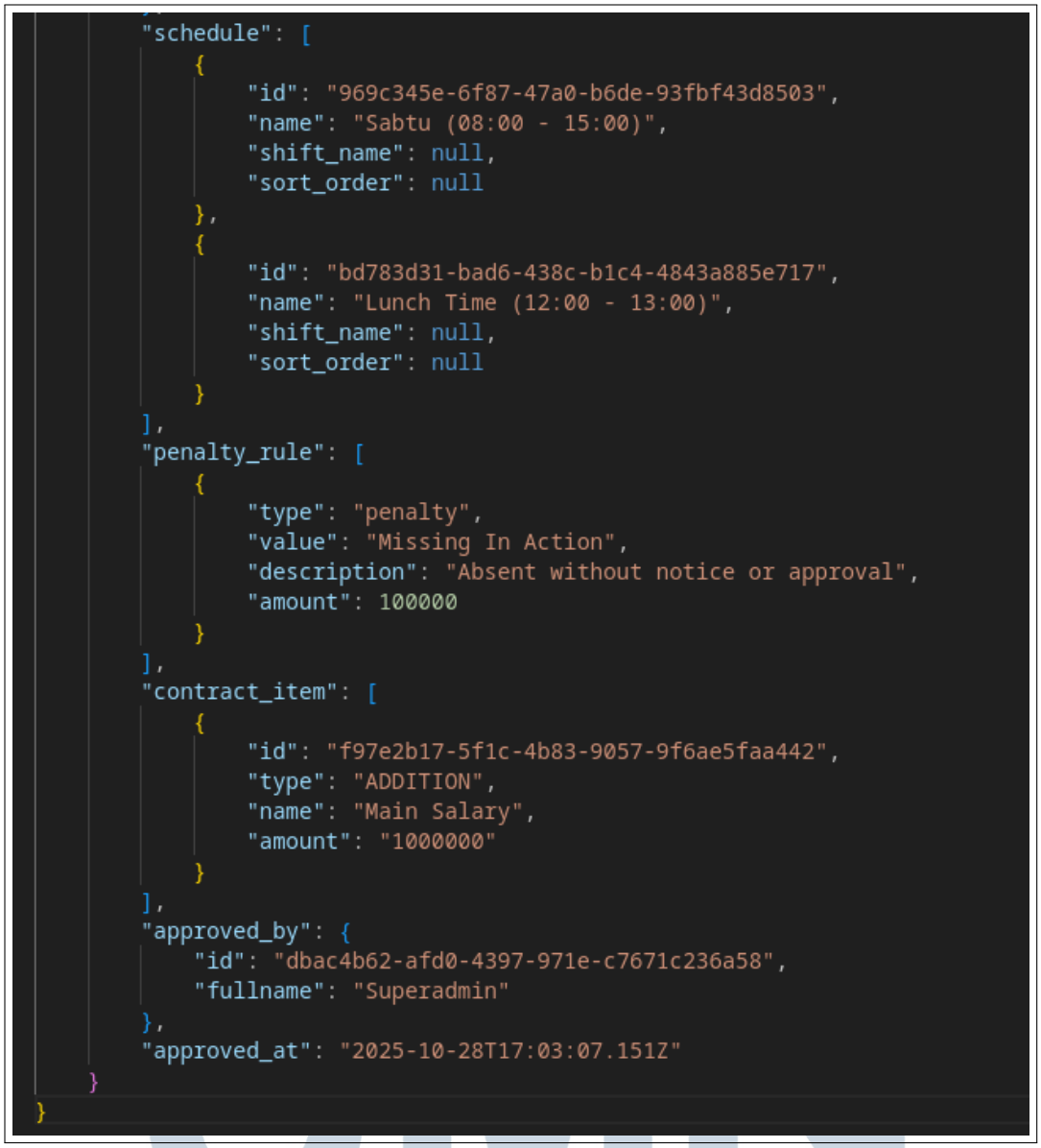
```
{
    "user_id": "e4f12883-5e11-4603-b3e5-c4aa1c49a171",
    "name": "Contract 4",
    "job_title_id": "b7745777-25b9-41ec-8e92-2102af97ae6d",
    "employment_status_id": "4f49db89-eb39-4db0-b6ee-38ba1d6d38a9",
    "supervisor_id": "0cb0b66a-e4d2-44f9-806c-a324f46a580d",
    "location_id": "e1b67c0c-f90c-49f0-937a-50c75984fd29",
    "start_date": "2026-11-04",
    "end_date": "2026-11-10",
    "leave_type": [
        {
            "id": "9f4947d5-7db9-43cc-9c9e-be85560bdacf"
        },
        {
            "id": "a9461bd2-ea26-4798-a647-67b883c3bc19"
        }
    ],
    "work_entry_type_code": "SHIFT",
    "work_day": [
        {
            "id": "782e0c41-87d9-4040-a55d-a1650262fa57"
        }
    ],
    "penalty_rule": [
        {
            "penalty_code": "LATE_UNDER_2HRS",
            "penalty_amount": 600000
        }
    ],
    "contract_item": [
        {
            "type": "ADDITION",
            "name": "Basic salary",
            "amount": 5000000
        }
    ]
}
```


Gambar 3.16. Request body pada API create/update

Sementara itu untuk *endpoint* detail seperti `/contract/:id`, `/contract/user/:id` akan memiliki respons body di bawah ini atau pada gambar 3.17, 3.18. Di mana, *endpoint* yang terdapat kata *user* hanya akan dapat diakses oleh pengguna itu sendiri.

```
{
    "code": 200,
    "message": "Success",
    "data": {
        "status": "ON-GOING",
        "employee_name": {
            "id": "2f30e240-b40e-11f0-ac07-b38b8a706955",
            "fullname": "Hiroshi Tanaka"
        },
        "name": "hiroshi1",
        "job_title": {
            "id": "a2992ce8-ce56-4cff-a336-4e56a3c15126",
            "name": "Board of Director"
        },
        "employment_status": {
            "id": "ac347adc-e9a4-4d09-afa7-9eb1dbd3131f",
            "name": "Part Time"
        },
        "supervisor": {
            "id": "a7822f27-8feb-4583-9971-b21c3a1649db",
            "fullname": "Timothy Baker"
        },
        "location": {
            "id": "c458d72c-1806-4fa8-a080-ed5ac8d9331a",
            "name": "Tangerang",
            "address": "Office"
        },
        "start_date": "2025-10-29T00:00:00.000Z",
        "end_date": "2026-10-01T00:00:00.000Z",
        "leave_type": [
            {
                "id": "51e6dc79-81b6-4716-8698-5b0555434b2d",
                "name": "WFH"
            },
            {
                "id": "9593ce0c-8602-4456-a59d-561abe96e18e",
                "name": "Force Majure"
            }
        ],
        "work_entry_type": {
            "id": 1,
            "type": "schedule",
            "code": "FIXED",
            "value": "Fixed",
            "description": ""
        },
```


Gambar 3.17. Contoh Respons JSON pada API detail



Gambar 3.18. Contoh Respons JSON pada API detail

Endpoint-endpoint seperti `/contract/` dan `/contract/user` akan menampilkan respons API seperti di bawah ini atau pada gambar 3.19, yang membedakan dari kedua endpoint ini adalah untuk `/contract/` akan menampilkan seluruh kontrak pengguna yang hanya bisa di akses oleh role *supervisor*, *superuser*, dan *superadmin*. Sebaliknya, untuk `/contract/user` hanya akan menampilkan kontrak dari pengguna itu sendiri.

```
{
    "code": 200,
    "message": "Success",
    "data": {
        "count": 16,
        "rows": [
            {
                "id": "0944dd7e-718f-4e2b-9a2d-cef88b31664d",
                "status": "ON-GOING",
                "name": "Kevin Contract",
                "start_date": "2025-11-01T00:00:00.000Z",
                "end_date": "2026-11-01T00:00:00.000Z",
                "created_at": "2025-11-01T08:43:37.712Z",
                "updated_at": "2025-11-01T08:44:57.207Z",
                "is_lunch": false,
                "employee_name": {
                    "id": "fb3ab8bb-c382-4ff2-8074-6d2c42163f27",
                    "fullname": "Kevin Ken"
                },
                "job_title": {
                    "id": "2c57c610-8393-4d53-ba55-d1a2180f8955",
                    "name": "Software Engineer"
                },
                "employment_status": {
                    "id": "9dcca920-e266-4fb7-aa09-b77a347c04df",
                    "name": "Internship"
                },
                "created": {
                    "id": "dbac4b62-afd0-4397-971e-c7671c236a58",
                    "fullname": "Superadmin"
                }
            },
        ]
    }
}
```


Gambar 3.19. API contract list

Dalam pembuatan kontrak pada sisi Frontend akan mengambil endpoint `/contract/options` di mana digunakan untuk mengisi setiap input, untuk respons dari *endpoint* tersebut dapat dilihat pada gambar 3.20, 3.21, 3.22.

```
{
    "code": 200,
    "message": "Success",
    "data": {
        "user": [
            {
                "id": "2f30e240-b40e-11f0-ac07-b38b8a706955",
                "fullname": "Hiroshi Tanaka"
            },
            {
                "id": "043bc970-b309-11f0-bb2b-a1d83e0e13df",
                "fullname": "Nana"
            }
        ],
        "job_title": [
            {
                "id": "a2992ce8-ce56-4cff-a336-4e56a3c15126",
                "name": "Board of Director"
            },
            {
                "id": "2c57c610-8393-4d53-ba55-d1a2180f8955",
                "name": "Software Engineer"
            }
        ],
        "employment_status": [
            {
                "id": "9dcca920-e266-4fb7-aa09-b77a347c04df",
                "name": "Internship"
            },
            {
                "id": "bf46b811-bec1-42aa-b500-068e93d81482",
                "name": "Probation"
            }
        ],
        "supervisor": [
            {
                "id": "a7822f27-8feb-4583-9971-b21c3a1649db",
                "fullname": "Timothy Baker"
            },
            {
                "id": "162bd469-d076-49ac-9703-3d5aa939ced8",
                "fullname": "Steven Adams"
            }
        ],
```


Gambar 3.20. Endpoint pre assets untuk create contract

Gambar 3.21. Endpoint pre assets untuk create contract

```
        "penalty_rule": [
            {
                "type": "penalty",
                "code": "MIA",
                "value": "MIA (Missing In Action)",
                "description": "Absent without notice or approval"
            },
            {
                "type": "penalty",
                "code": "LATE_UNDER_2HRS",
                "value": "Late Arrival (Less than 2 hours)",
                "description": "Penalty applied for minor tardiness below 2 hours."
            }
        ],
        "payroll_item": [
            {
                "id": "99cd5b73-8724-4284-b6a4-27fd838af00c",
                "name": "Tunjangan BPJS",
                "description": "Tunjangan yang diberikan untuk BPJS karyawan."
            }
        ],
        "unit_item": [
            {
                "type": "unit",
                "code": "ADDITION",
                "value": "Addition",
                "description": ""
            },
            {
                "type": "unit",
                "code": "BENEFIT",
                "value": "Benefit",
                "description": ""
            }
        ]
    }
}
```


Gambar 3.22. Endpoint pre assets untuk create contract

3.5.2 Payslip Date

Adapun request body untuk membuat *payslip_dates* yang dapat dilihat pada request di bawah ini atau pada gambar 3.23:


```
{
    "start_date": "2025-10-03",
    "end_date": "2025-11-03"
}
```


Gambar 3.23. Request body pada API create

Sementara itu untuk *endpoint* detail seperti `/payslip/date/payslip_date_id`, akan memiliki respons body di bawah ini atau pada gambar 3.24.

Gambar 3.24. Contoh Respons JSON pada API detail

Endpoint seperti `/payslip` akan menampilkan respons API seperti di bawah ini atau pada gambar 3.25.

```
{
    "code": 200,
    "message": "Success",
    "data": {
        "count": 1,
        "rows": [
            {
                "id": "c3d56b9e-6e27-43f9-8484-84831e450ad6",
                "start_date": "2025-10-28T00:00:00.000Z",
                "end_date": "2025-10-28T23:59:59.999Z",
                "is_status": "PENDING",
                "created_at": "2025-10-28T15:17:36.726Z",
                "updated_at": "2025-10-28T15:17:36.726Z",
                "total_user": 11,
                "created_by": "dbac4b62-afd0-4397-971e-c7671c236a58"
            }
        ]
    }
}
```


Gambar 3.25. API payslip date list

3.5.3 Attendance Date

Adapun request body untuk membuat *attendance_dates* pada *endpoint* `/attendance-dates/` yang dapat dilihat pada request di bawah ini atau pada gambar 3.26:


```
{
    "date": "2025-10-03"
}
```


Gambar 3.26. Request body pada API create

Lalu, untuk mengupdate data menggunakan endpoint `/daily-attendances/detail/approve/:attendance_date_id` memiliki request dibawah ini atau gambar 3.27:

Gambar 3.27. Request body pada API update

Sementara itu untuk *endpoint* list seperti `/attendance-dates/`, akan memiliki respons body di bawah ini atau pada gambar 3.28. yang tujuannya

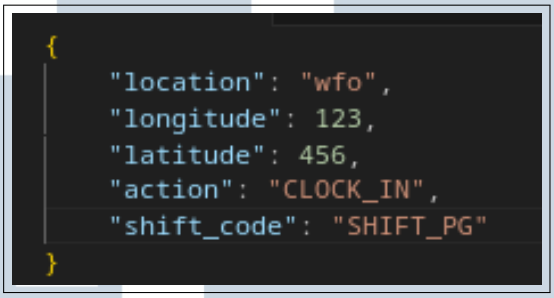
Gambar 3.28. Contoh Respons JSON pada API list

Endpoint seperti `/daily-attendances/detail/:attendance_date_id` akan menampilkan respons API seperti di bawah ini atau pada gambar 3.29. Dimana respon ini akan memberikan detail terkait karyawan yang telah melakukan absen, mulai dari statusnya, jam masuk, jam keluar, dan sebagainya. Respons ini akan berubah-ubah setiap harinya.


```
{
    "code": 200,
    "message": "Success",
    "data": {
        "count": 2,
        "rows": [
            {
                "id": "7553770f-21b5-4819-8dbe-f63ca3e9accb",
                "flow_status": "PENDING",
                "attendace_status": "NEED REVIEW",
                "user_id": "2f30e240-b40e-11f0-ac07-b38b8a706955",
                "name": "Hiroshi Tanaka",
                "job_name": "Board of Director",
                "clock_in": null,
                "clock_out": null,
                "lunch_in": null,
                "lunch_out": null,
                "reason_code": null,
                "reason_clock_in": null,
                "reason_lunch_end": null,
                "reason_clock_out": null,
                "clock_in_late": null,
                "clock_out_late": null,
                "lunch_start_late": null,
                "lunch_out_late": null,
                "attendance_date": "2025-11-09T00:00:00.000Z"
            },
            {
                "id": "3e64d187-a3b4-4e5d-92dc-0a1a6cbb5608",
                "flow_status": "PENDING",
                "attendace_status": "NEED REVIEW",
                "user_id": "fb3ab8bb-c382-4ff2-8074-6d2c42163f27",
                "name": "Kevin Ken",
                "job_name": "Software Engineer",
                "clock_in": null,
                "clock_out": null,
                "lunch_in": null,
                "lunch_out": null,
                "reason_code": null,
                "reason_clock_in": null,
                "reason_lunch_end": null,
                "reason_clock_out": null,
                "clock_in_late": null,
                "clock_out_late": null,
                "lunch_start_late": null,
                "lunch_out_late": null,
                "attendance_date": "2025-11-09T00:00:00.000Z"
            }
        ]
    }
}
```

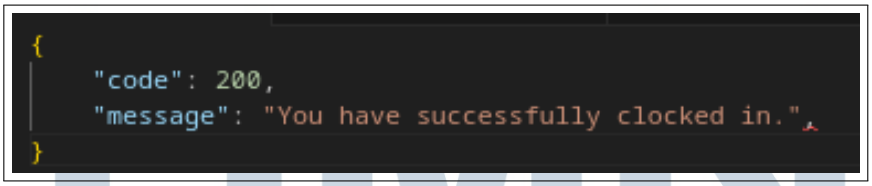

Gambar 3.29. API Attendance Date detail

Endpoint `/daily-attendances/attendance` memiliki request yang harus diberikan ketika ingin menggunakan endpoint ini untuk fungsi `attendDaily` atau fungsi *attendance*. Di mana *request* dapat dilihat pada gambar 3.30. *request* itu sendiri berupa, `location`, `longitude`, `latitude`, hingga `shift_code`.



Gambar 3.30. Request body API Daily Attendance (attendDaily)

Request-request tersebut merupakan wajib yang harus di kirim untuk menggunakan endpoint ini. Dan juga memiliki beberapa request opsional lainnya seperti, `reason_clock_in`, `reason_clock_out`, dan lain-lain. Untuk `shift_code` sendiri hanya berlaku untuk karyawan yang bekerja dengan tipe shif, sehingga jadwal akan dipilih sesuai `shift_code` yang dikirimkan pada hari tersebut untuk menentukan jam masuk dan jam selesainya. Dibawah ini merupakan salah satu respon jika karyawan telah berhasil melakukan absen pada gambar 3.31.



Gambar 3.31. Respon daily attendance

3.5.4 Socket

Gambar 3.32. Demo socket untuk event

Pada gambar 3.32 adalah hasil dari implementasi *socket* pada fitur `create event`. Gambar tersebut menampilkan pesan yang berhasil diterima dari sisi *client* melalui koneksi Socket setelah proses pembuatan event selesai dilakukan. Sehingga data event dapat langsung muncul tanpa perlu melakukan refresh atau pemanggilan ulang API. Mekanisme ini menunjukkan bahwa komunikasi dua arah antara *client* dan *server* melalui *Socket.IO* telah berjalan dengan baik yang bertujuan untuk mengambil data secara dinamis di aplikasi.

3.6 Kendala dan Solusi yang Ditemukan

Adapun kendala yang dihadapi selama magang di PT Visi Karya Nusantara:

1. Kurangnya komunikasi dalam pihak *Front-end* dan *Back-end* yang membuat alur pengerjaan menjadi terhambat.
2. Kurangnya pengetahuan terhadap metode *best practices* yang digunakan dalam industri teknologi dalam pengkodean.

Dan berikut solusi yang dilakukan untuk menangani kendala-kendala yang terjadi:

1. Membuat alur pengerjaan yang akan dilakukan dengan pihak *Front-end* dan *Back-end* selama seminggu dari hari senin sampai dengan jum'at.
2. Mencari informasi melalui *Stack Overflow* atau forum-forum terkenal serta blog yang membahas mengenai *best practice* kode.