Backend Assessment III

PT. Kledo Berhati Nyaman

Syarat Teknis

Gunakan Laravel 8 sebagai framework, dengan syarat sebagai berikut:

* Webserver menggunakan `artisan serve`.
* Validasi menggunakan Form Request Validation.
* Tidak ada algoritma alur proses di Controller.
* Transaksi basis data menggunakan Eloquent (MySQL/MariaDB).
* Dokumentasi REST-API menggunakan Swagger (l5-swagger).
* Testing menggunakan PHPUnit.
* Sertakan README sebagai petunjuk penggunaan.

Kriteria Penilaian

* Aplikasi berjalan sesuai spesifikasi, seluruh syarat di atas wajib dipenuhi. README akan sangat berpengaruh dalam proses uji coba yang kami lakukan.
* Kualitas kode.
* Kecepatan. Maksimal waktu pengerjaan adalah 24 jam, terhitung mulai dokumen ini dikirim.

Berikut adalah penjelasan mengenai apa saja yang wajib Anda selesaikan dalam tugas ini.

Kebutuhan

Buat aplikasi dengan konsep REST-API pada sisi backend untuk memenuhi kebutuhan sistem perhitungan lembur.

Ada 2 metode perhitungan lembur yang digunakan.

1. Salary / 173
2. Fixed

Jika yang digunakan adalah metode pertama, maka semua upah lembur dari pegawai yang ada akan dihitung menggunakan metode tersebut. Begitu juga jika yang dipilih adalah metode kedua.

Khusus untuk pegawai dalam masa percobaan, upah lembur mulai dihitung saat durasi lembur lebih dari 1 jam. Saat lebih dari 1 jam, yang dihitung adalah durasi setelah 1 jam tersebut. Namun tidak akan dihitung jika durasi setelah 1 jam belum mencapai 1 jam, aturan ini berlaku kelipatan. Durasi lembur untuk pegawai dalam masa percobaan ini dihitung pada setiap lembur yang dilakukan, bukan dari akumulasi lembur per-bulan.

Contoh hasil perhitungan lembur pegawai dalam masa percobaan seperti berikut.

1. Lembur 2 jam, maka mendapatkan upah lembur 1 jam
2. Lembur 2,5 jam, maka mendapatkan upah lembur 1 jam
3. Lembur 1,5 jam, maka tidak mendapatkan upah lembur
4. Lembur 3,9 jam, maka mendapatkan upah lembur 2 jam

Tabel

references

* id
* code
* name
* expression

settings

* key
* value
* expression

employees

* id
* name
* status\_id
* salary

overtimes

* id
* employee\_id
* date
* time\_started
* time\_ended

Seeder

references

|  |  |  |  |
| --- | --- | --- | --- |
| id | code | name | expression |
| 1 | overtime\_method | Salary / 173 | (salary / 173) \* overtime\_duration\_total |
| 2 | overtime\_method | Fixed | 10000 \* overtime\_duration\_total |
| 3 | employee\_status | Tetap |  |
| 4 | employee\_status | Percobaan |  |

settings

|  |  |  |
| --- | --- | --- |
| key | value | expression |
| overtime\_method | 1 | (salary / 173) \* overtime\_duration\_total |

Endpoint

Setting

PATCH /settings

Mengubah data `settings`.

Request

* key
* value

Rule

* key
* Hanya bisa diisi `overtime\_method`.
* value
* Hanya bisa diisi oleh nilai dari `references`.`id` dengan kriteria `code` = `overtime\_method`.
* expression
* Otomatis didapat berdasarkan `value` yang diterima, diambil dari `references`.`expression`.

Employee

POST /employees

Membuat data `employees`.

Request

* name
* status\_id
* salary

Rule

* name
* String
* Minimal 2 karakter
* Harus unik
* status\_id
* Integer
* Sesuai dengan yang ada di `references`.`id` dengan kriteria `code` = `employee\_status`
* salary
* Integer
* Minimal 2 juta
* Maksimal 10 juta

GET /employees

Menampilkan seluruh data `employees`, dengan format pagination.

Request

* per\_page
* page
* order\_by
* order\_type

Rule

* per\_page
* Integer
* Default 10
* page
* Integer
* Default 1
* order\_by
* String
* Enum(name, salary)
* order\_type
* String
* Enum(ASC, DESC)

Response

* id
* name
* status\_id
* status
* name
* salary

Overtime

POST /overtimes

Membuat data `overtimes`.

Request

* employee\_id
* date
* time\_started
* time\_ended

Rule

* employee\_id
* Integer
* Sesuai dengan yang ada di `employees`.`id`
* date
* Date
* Tidak boleh ada `date` yang sama pada `employee\_id` tersebut
* time\_started
* Format HH:mm
* Tidak boleh lebih dari `time\_ended`
* time\_ended
* Format HH:mm
* Tidak boleh kurang dari `time\_started`

GET /overtimes

Menampilkan data `overtimes` berdasarkan rentang waktu yang ditentukan, tanpa format pagination.

Request

* date\_started
* date\_ended

Rule

* date\_started
* Date
* Tidak boleh lebih dari `date\_ended`
* date\_ended
* Date
* Tidak boleh kurang dari `date\_started`

Response

* id
* employee\_id
* employee
* name
* date
* time\_started
* time\_ended

Overtime Pay

GET /overtime-pays/calculate

Menampilkan hasil perhitungan dari `overtimes` yang ada pada setiap `employees`, berdasarkan bulan yang ditentukan, tanpa format pagination.

Request

* month

Rule

* month
* Format YYYY-MM

Response

* id
* name
* status\_id
* status
* name
* salary
* overtimes[]
* id
* date
* time\_started
* time\_ended
* overtime\_duration \*perhitungan durasi sesuai penjelasan di awal
* overtime\_duration\_total \*akumulasi `overtime\_duration`
* amount \*perhitungan upah yang diterima sesuai pengaturan `overtime\_method` di `settings`

Sesuai namanya, gunakan `settings`.`expression` sebagai rumus untuk menghasilkan upah lembur, bukan manual di kode.

Testing

Buat testing yang bersifat fungsional menggunakan PHPUnit untuk memastikan apa yang telah Anda kerjakan sesuai dengan aturan yang diberlakukan. Testing mencakup semua skenario di atas tanpa terkecuali. Pastikan testing dapat berjalan dengan satu perintah.

Checklist

Pastikan kriteria-kriteria di bawah terpenuhi

|  |  |  |
| --- | --- | --- |
| **Topik** | **Requirement** | **Nilai Tambah** |
| Validasi endpoint | Gunakan Form Request Validation | Menggunakan file class request terpisah |
| Controller | Alur logika kode diletakkan diluar file class Controller. Dari file class Controller hanya memanggil method dari class lain. | Menggunakan Repository Pattern |
| Dokumentasi | Gunakan dokumentasi swagger, di controller dan di model. Pastikan Swagger UI bisa diakses, dan bisa digunakan untuk mencoba endpoint. |  |
| Testing | Setiap endpoint di test untuk memastikan bisa berjalan dengan baik. | Tidak hanya test response code saja, tapi berbagai kondisi di test. Mulai dari test validasi-nya, serta test alur didalamnya. Semakin lengkap semakin bagus. |
| Readme | Pastikan tim penilai bisa menginstall script Anda dengan mudah. Bisa copy paste perintah yang Anda cantumkan di readme, dan bisa berjalan normal. | Semakin jelas readme nya semakin bagus. |