# Pesanan Resource

## Membuat Pesanan Resource

### Mengubah Bahasa

Silahkan buka file **config/app.php** dan silahkan modifikasi bagian **locale** nya menjadi **id** seperti berikut :

*‘locale’ => ‘id’,*

Tujuannya adalah membuat semua bahasa menjadi bahasa Indonesia, dan bahkan ini juga menghilangkan plural yang ada secara default, namun jika Kamu tetap ingin menggunakan bahasa inggris itu sama sekali tidak masalah.

### Pesanan Observer

Dalam hal ini, kita ingin field/kolom **user\_id** itu terisi otomatis berdasarkan pengguna yang saat ini login. Untuk itu kita bisa masukan dalam method **booted()** tepat di bawah **$casts** seperti berikut:

Kemudian setelah itu, kita juga ingin membuat kolom nomor pesanan menjadi ORD00000001, kemudian jika ada data selanjutnya akan menjadi ORD00000002, itu sebenarnya bisa kita buat pada method booted() ini. **Namun**, itu akan menjadi membingungkan terhadap pengguna nantinya. Sehingga kita perlu menunjukkan ini pada saat penggunaan membuat order baru.

### Helper Function

Buat file baru dengan nama **helpers.php** tepat pada folder **app**, dan isikan seperti berikut:

Setelah itu, silahkan buka .composer.json untuk mendaftarkan fungsi ini. Pada bagian “**autoload-dev**”, silahkan tambahkan key baru dengan nama “**files**” dan masukkan helpersnya tepat seperti berikut:

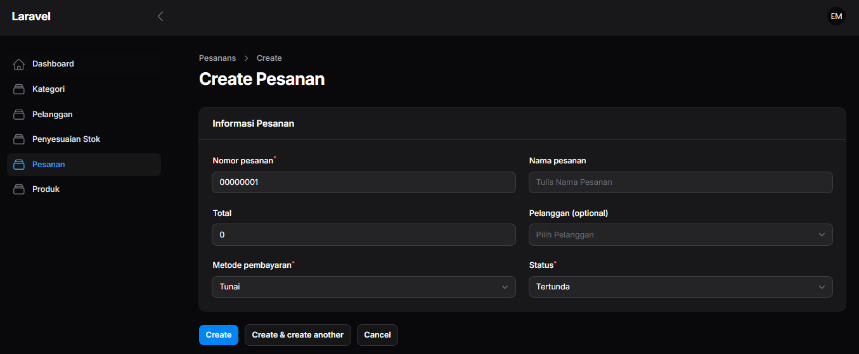
Lalu, buka terminal kamu dan jalankan perintah **dump-auto**:

*composer dump-auto*

Dengan begitu sekarang kamu sudah bisa menggunakan fungsi ini tepat di dalam form yang ada di pesanan resource.

### Membuat Form

Selanjutnya kita akan memodifikasi form pada **PesananResource.php**, tuliskan isi nya seperti berikut:

Maka tampilan yang default diberikan akan berubah seperti berikut:

Beberapa perubahan tersebut adalah saya menghilangkan beberapa field/kolom dan juga membuat beberapa field/kolom menjadi memiliki default value, jika Kamu perhatikan field total itu otomatis menjadi 0 dan readonly pada saat membuat pesanan. Tujuannya agar nanti field tersebut akan di isi otomatis pada saat kita melakukan transaksi.

### Custom Page

Kita juga butuh halaman khusus untuk melakukan transaksi ini, karena setelah kita membuat pesanan ini, kita ingin otomatis mengarahkan user/pengguna ke halaman transaksi, secara bawaan ketika membuat pesanan, kita akan otomatis pergi ke halaman edit pesanan. Yang mana kita tidak mau itu terjadi, karena kita ingin langsung di arahkan ke halaman transaksi.

Maka untuk itu, silahkan buka terminal Kamu dan buat 1 halaman dengan nama BuatTransaksi seperti berikut:

*php artisan make:filament-page BuatTransaksi --type=custom --resource=PesananResource*

dengan begitu, kkamu akan mendapatkan 2 file sekaligus yaitu view dan juga class/model nya.

1. app/Filament/Resources/PesananResource/Pages/BuatTransaksi.php
2. resources/views/filament/resources/pesanan-resource/pages/buat-transaksi.blade.php

silahkan buka kembali file PesananResource.php untuk menambahkan halaman yang baru saja kamu buat. Dan tepat pada bagian **getPages()** silahkan tambahkan ‘buat-transaksi’ seperti berikut:

setelah itu, buka file **CreatePesanan.php** untuk mengganti redirectnya seperti berikut:

jika Kamu bertanya dari mana saya tahu route itu, saya mengetahuinya dari route:list, kamu bisa gunakan perintah artisan pada terminal:

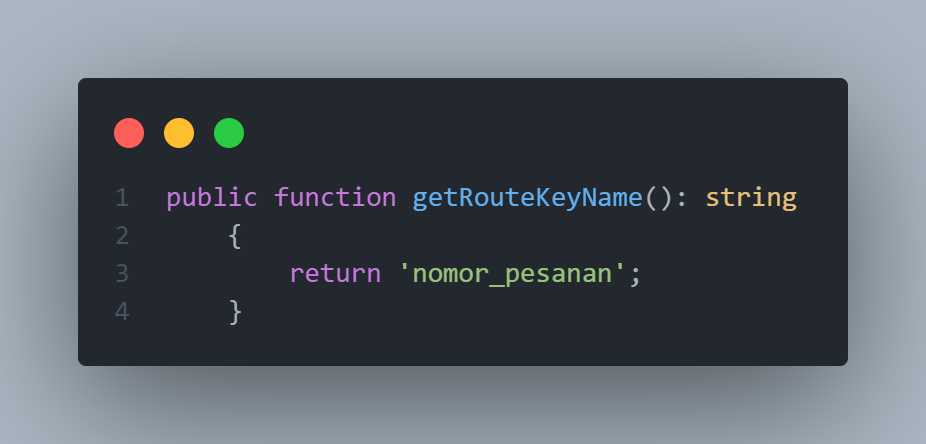
*php artisan route:list*

dan nanti kita ingin mhalaman buat transaksi tersebut menggunakan title yang berbeda, artisan alih-alih menggunakan ‘buat transaksi’ , kita ingin ganti jadi ‘Order: nomor\_pesanan’. Maka untuk itu,kamu perlu membuat semacam pengenalan terhadap halaman ini dengan membuat inisial pada pesanan nya.

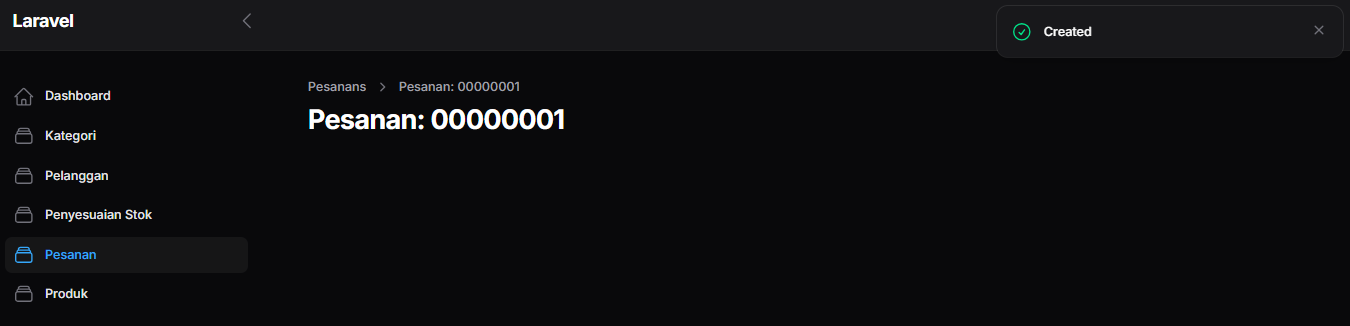
Buka file **BuatTransaksi** dan silahkan modifikasi menjadi seperti berikut:



Jelas di sini bahwa saya juga mengimplementasikan interface dari **HasForms**, agar nanti kita bisa menggunakan form pada halaman ini.

Sekarang buka model **Pesanan.php** dan silahkan tambahkan 1 method untuk memperkenalkan key dari route nya. Kamu bisa menambahkan itu tepat di bawah method **booted()** seperti berikut:

Sekarang kamu bisa coba langsung tekan tombol create/buat pada halaman buat pesanan untuk membuat pesanan baru, dan harusnya jika berhasil akan langsung di arahkan kehalaman seperti berikut:



## Pembuatan Transaksi

Dalam bagian ini, kita akan fokus untuk membahas bagaimana cara kita implementasi transaksi ini. Untuk memulai pelajaran selanjutnya saya ingin menjelaskan bahwa pada halaman ini, kita akan membuat opsi untuk produk, kemudian ketika kita pilih, maka otomatis kita akan insert record baru ke dalam database dan itu akan di masukkan ke tabel detail\_pesanans.

Karena kita butuh nanti membuat style tambahan, maka kita perlu membuat yang namanya theme sendiri. Sehingga untuk itu, silakan buka terminal Kamu dan jalankan perintah berikut:

*php artisan make:filament-theme*

Dengan begitu, maka akan otomatis mengunduh menggunakan npm yang ada di sistem Kamu, jadi pastikan node js sudah ada di sistem Anda saat ini. Setelah unduhan selesai, maka sekarang kita bisa memodifikasi file vite.config.js,

Buka file nya dan silakan modifikasi menjadi seperti berikut:



Setelah itu, silakan buka file **app/Providers/Filament/AppPanelProvider.php**, dan silakan chain method **->viteTheme('resources/css/filament/app/theme.css')** tepat di bawah **->sidebarCollapsibleOnDesktop()** seperti:

Setelah itu, silakan buka terminal Anda di tab baru, dan silakan jalankan

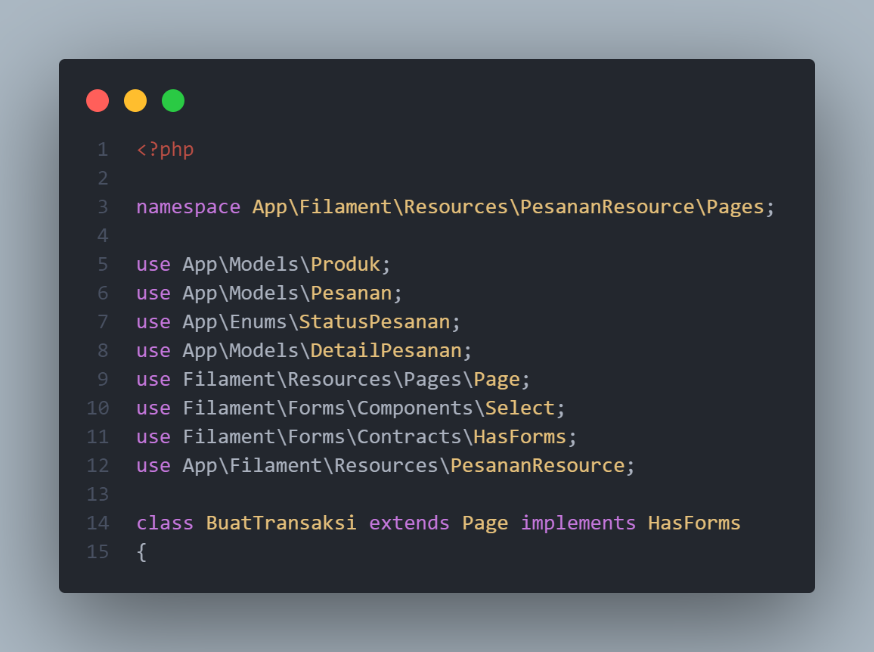
*npm run dev*

agar nanti css kita terbaca oleh tailwind nya.

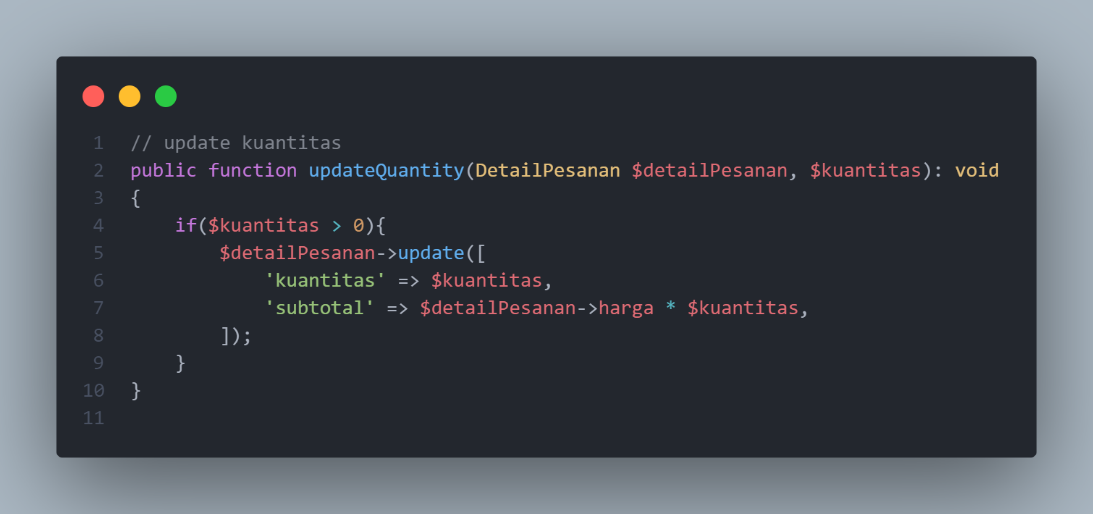
### Membuat Class Transaksi

1. Form Schema

Sebelum kita melanjut lebih jauh, kita akan fokus di **app/Filament/Resources/OrderResource/Pages/CreateTransaction.php** dulu, mari kita buat fungsi baru dengan nama getFormSchema. Saya akan menambahkan ini di bawah method **getTitle()** seperti berikut:

 Jangan lupa untuk menambahkan class yang dibutuhkan yaitu:

1. Update Quantity

Kemudian setelah getFormSchema(), kita juga butuh nantinya method untuk memperbarui quantity nya. Oleh karena itu silakan tambahkan method nya tepat di bawah **getFormSchema()** ini seperti:

1. Remove Produk

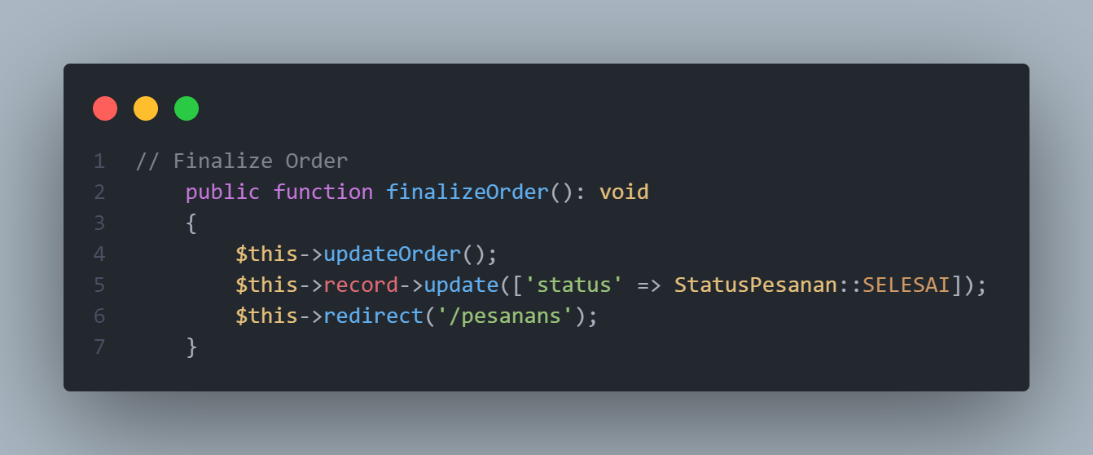
Selain memberikan penggunakan fitur untuk update quantity, tentu juga bisa menghapus order nya, entah itu salah klik atau mungkin entah itu batal, tulis kode nya seperti berikut:

1. Update Order

Setelah itu, kita akan membuat method dengan nama **updateOrder()**, method ini nantinya akan di panggil oleh 2 method yang akan kita buat selanjutnya.



1. Finalize Order

Kemudian, ketika nanti tombol simpan, atau order di klik, maka method ini yang akan kita panggil.

1. Draft

Tentu kita butuh draft, kita ingin pengguna bisa membuat orderan lain walaupun orderan yang saat ini belum selesai, maka oleh karena itu kita butuh method **saveAsDraft()** untuk menyimpan semua pesanan tetapi tetap dalam status **pending**.

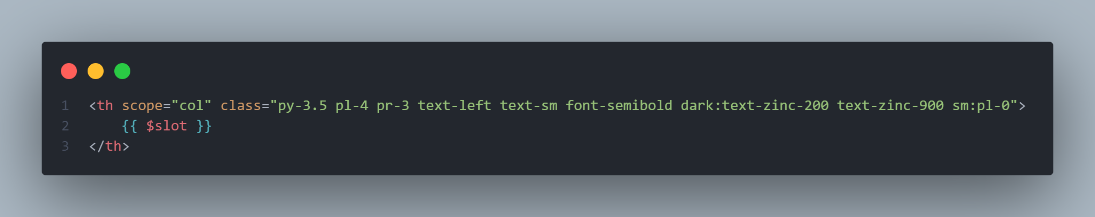
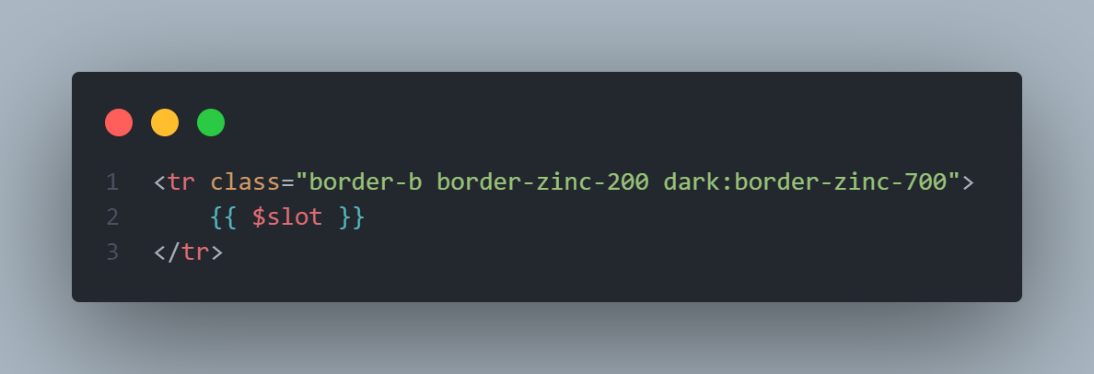
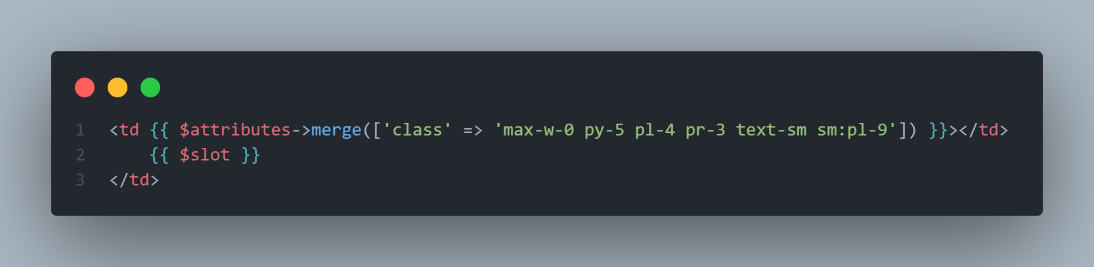


### Komponen Tabel

Sekarang, kita butuh komponen untuk table, untuk itu. Silakan buat 1 folder dengan nama **table** tepat di dalam folder **resources/views/components**, kemudian di dalamnya silakan buat 5 file yaitu:

1. index.blade.php
2. thead.blade.php



1. th.blade.php
2. tr.blade.php
3. td.blade.php

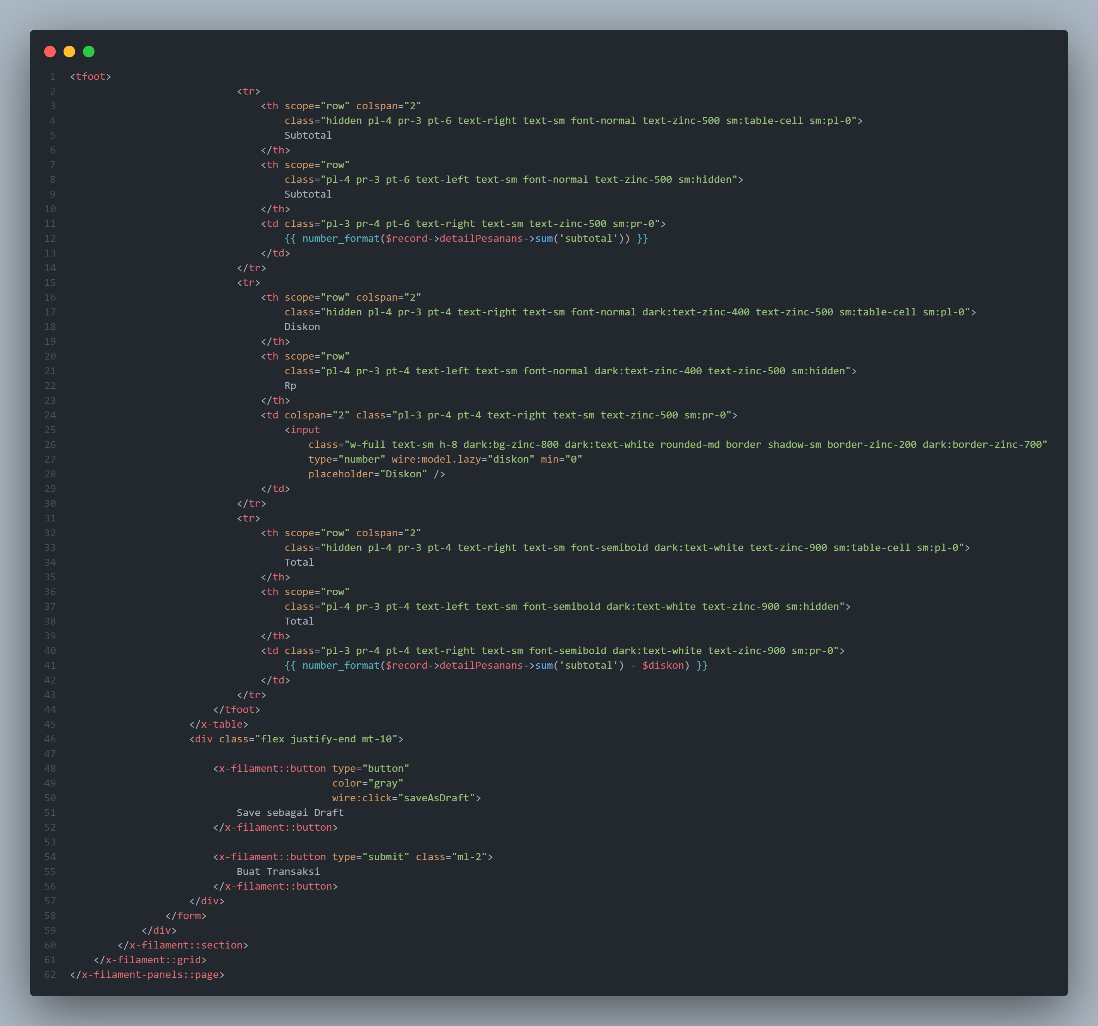
Setelah itu, maka kita bisa lanjut untuk memodifikasi file **resources/views/filament/resources/pesanan-resource/pages/buat-transaksi.blade.php**. Silakan buka file nya dan tambahkan markup berikut ini:

Dikarenakan penulisan kode pada bagian buat-transaksi.blade.php sedikit panjang maka akan saya tampilkan per section seperti berikut:

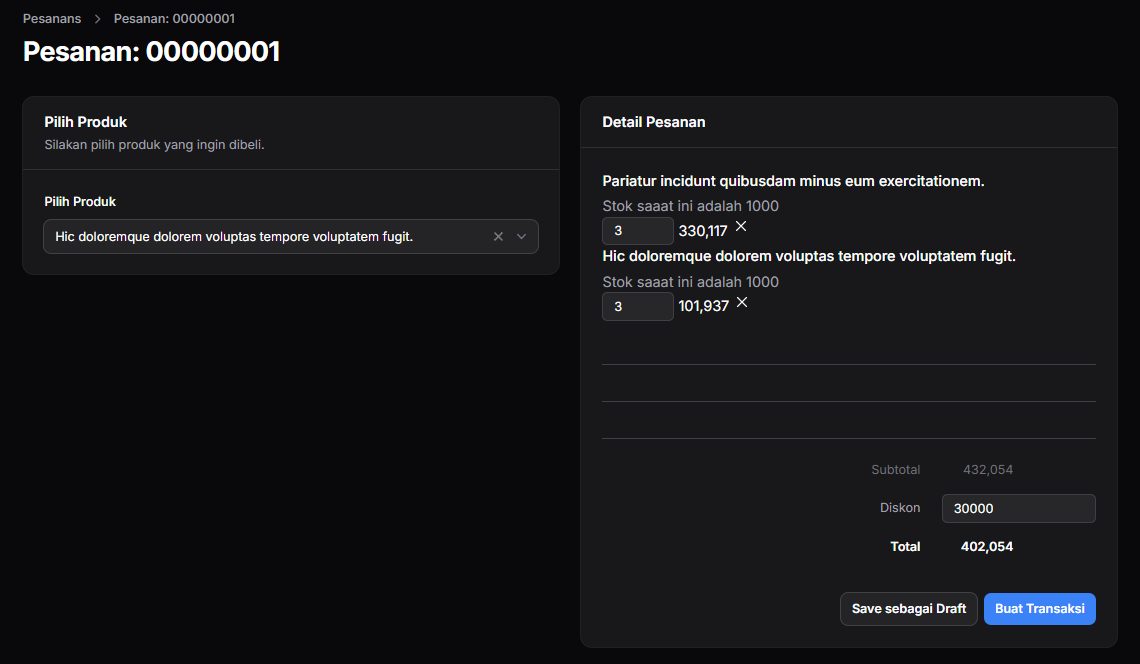
1. 



1. 



### Overview

Sekarang kamu bisa membuat orderan baru, dan harusnya tampilan nya akan seperti berikut:

*“#****note****: jika kamu sebelumnya telah membuat pesanan namun belum menekan tombol simpan sebagai draft/buat transaksi, maka pesanan tersebut sebenarnya sudah tersimpan dan tidak bisa di hapus, kamu bisa menghapus data detail pesanan sebelumnya di tabel detail\_pesanans basis data nya secara manual terlebih dahulu”*

## Modifikasi Daftar Pesanan

Pada bagian ini kita akan fokus pada table yang ada di **app/Filament/Resources/PesananResource.php**, ada beberapa yang ingin kita lakukan di sini, kita ingin mengetahui jumlah total dan juga profit yang kita dapatkan. Dalam hal ini, kita masih akan bekerja pada tabel nya saja, sehingga untuk *statistic* bisa kita lanjutkan nanti.

### Pesanan Factory

Untuk melihat ini bekerja atau tidak, tentu kita butuh factory, karena gk mungkin juga kita buat data order satu persatu. Untuk itu, silakan buka file **database/factories/OrderFactory.php** dan silahkan modifikasi method **definition()** dan method **configure()**. Berikut adalah penulisan kode nya:

|  |
| --- |
| public function definition(): array      {          Pesanan::unsetEventDispatcher();          return [              'user\_id' => 1,              'pelanggan\_id' => rand(1,50),              'nomor\_pesanan' => $this->faker->unique()->bothify('ORD########'),              'nama\_pesanan' => ucfirst($this->faker->word()),              'diskon' => $this->faker->numberBetween(5000, 10000),              'total' => 0,              'metode\_pembayaran' => collect(MetodePembayaran::cases())->random(),              'status' => collect(StatusPesanan::cases())->random(),          ];      }      public function configure()      {          return $this->afterMaking(function (Pesanan $pesanan){          })->afterCreating(function (Pesanan $pesanan){              $produkId = Produk::query()->inRandomOrder()->take(rand(1,5))->pluck('id');              $detailPesanan = $produkId->map(function($produkId) use ($pesanan){                  $kuantitas = rand(1,10);                  $harga = Produk::find($produkId)->harga;                  $subtotal = $kuantitas \* $harga;                  return [                      'pesanan\_id' => $pesanan->id,                      'produk\_id' => $produkId,                      'kuantitas' => $kuantitas,                      'harga' => $harga,                     'subtotal' => $subtotal,                  ];              });              DetailPesanan::insert($detailPesanan->toArray());              $total = $detailPesanan->sum('subtotal') - $pesanan->diskon;              $pesanan->total = $total;              $pesanan->keuntungan = $total \* 0.1;              $pesanan->save();          });      } |

Jangan lupa juga untuk import class yang di butuhkan seperti berikut:

|  |
| --- |
| *database/factories/PesananFactory.php* |
| use App\Models\DetailPesanan;  use App\Models\Produk;  use App\Models\Pesanan;  use App\Enums\MetodePembayaran;  use App\Enums\StatusPesanan;  use Illuminate\Database\Eloquent\Factories\Factory; |

Setelah itu, silakan modifikasi file **database/seeders/DatabaseSeeder.php** untuk menambahkan PesananFactory nya seperti:

|  |
| --- |
| *database/seeders/DatabaseSeeder.php* |
| public function run(): void  {          $this->call([              UserSeeder::class,              KategoriSeeder::class,          ]);          Pelanggan::factory(50)->create();          Produk::factory(50)->create();          PenyesuaianStok::factory(50)->create();          Pesanan::factory(50)->create();  } |

Setelah itu, silakan buka kembali terminal Kamu dan lakukan fresh seed seperti:

|  |
| --- |
| php artisan migrate:fresh --seed |

Dengan begitu, sekarang Kamu bisa lihat pada halaman Pesanan, seharusnya sudah ada beberapa data yang baru.

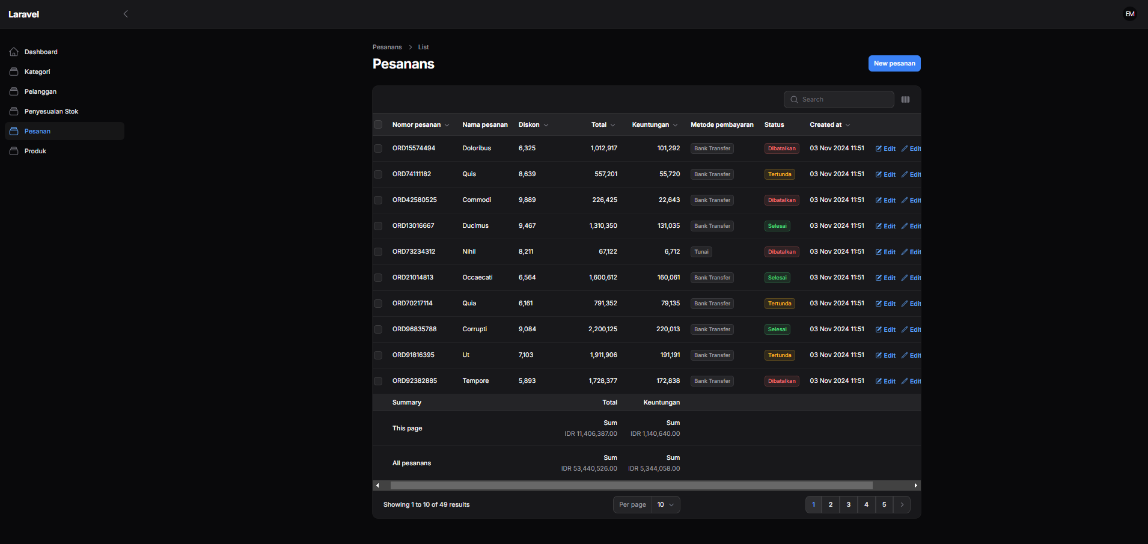
### Modifikasi Tabel

Ada beberapa penyesuaian yang akan kita lakukan pada tabel pesanan ini, termasuk mengganti status menjadi menggunakan pesanan, sekaligus menghitung total dan profit kita lakukan juga pada **table()** ini. Buka file nya, dan silakan ubah dengan kode berikut:

|  |
| --- |
| *app/Filament/Resources/PesananResource.php* |
| public static function table(Table $table): Table  {          return $table              ->columns(self::getTableColumns())              ->filters([                  //              ])              ->actions([                  EditAction::make(),              ])              ->bulkActions([                  Tables\Actions\BulkActionGroup::make([                      Tables\Actions\DeleteBulkAction::make(),                  ]),              ]);  } |

Dan di bagian paling bawah buat function baru dengan nama **getTableColumns()**, seperti berikut:

|  |
| --- |
| *app/Filament/Resources/PesananResource.php* |
| public static function getTableColumns(): array      {          return [              TextColumn::make('nomor\_pesanan')                      ->searchable()                      ->sortable(),                  TextColumn::make('nama\_pesanan')                      ->searchable(),                  TextColumn::make('diskon')                      ->numeric()                      ->sortable(),                  TextColumn::make('total')                      ->numeric()                      ->alignEnd()                      ->sortable()                      ->summarize(                          Sum::make('total')                              ->money('IDR'),                      ),                  TextColumn::make('keuntungan')                      ->numeric()                      ->alignEnd()                      ->sortable()                      ->summarize(                          Sum::make('total')                              ->money('IDR'),                      )                      ->sortable(),                  TextColumn::make('metode\_pembayaran')                      ->badge()                      ->color('gray'),                  TextColumn::make('status')                      ->badge()                      ->color(fn ($state) => $state->getColor()),                  TextColumn::make('user.name')                      ->numeric()                      ->toggleable(isToggledHiddenByDefault:true),                  TextColumn::make('pelanggan.nama')                      ->numeric()                      ->toggleable(isToggledHiddenByDefault:true),                  TextColumn::make('created\_at')                      ->dateTime()                      ->sortable()                      ->formatStateUsing(fn($state) => $state->format('d M Y H:i')),                  TextColumn::make('updated\_at')                      ->dateTime()                      ->sortable()                      ->formatStateUsing(fn($state) => $state->format('d M Y H:i'))                      ->toggleable(isToggledHiddenByDefault:true),                  ];      } |

Sekarang, jika kamu refresh maka tampilan sudah akan menjadi seperti berikut:

### Update Transaksi

Pada dasarnya, order ini telah mempunyai edit page, namun kita hanya akan memperbarui pelanggan atau pun metode pembayaran nya saja. Untuk yang lain, itu tidak perlu di lakukan. Nah untuk itu, kita butuh juga tombol untuk mengubah transaksi ini, maka itu bisa lakukan dengan menambahkan action baru tepat di samping edit. Jadi untuk itu, silakan fokus pada **chain method ->actions()** dan silakan perbarui menjadi seperti:

|  |
| --- |
| *app/Filament/Resources/PesananResource.php* |
| public static function table(Table $table): Table  {          return $table  ->defaultSort('created\_at', 'desc')              ->columns(self::getTableColumns())              ->filters([                  //              ])              ->actions([                  EditAction::make(),                  Action::make('ubah-transaksi')                      ->label('Edit Transaksi')                      ->icon('heroicon-o-pencil')  ->url(fn($record) => "/pesanans/{record->nomor\_pesanan}"),              ])              ->bulkActions([                  Tables\Actions\BulkActionGroup::make([                      Tables\Actions\DeleteBulkAction::make(),                  ]),              ]);  } |

## Implementasi Tab dan Filter

### Filters

Pada bagian ini, saya akan menunjukkan cara untuk filter order tersebut, baik itu dari payment method, status, dan juga tanggalnya. Kita tetap akan bekerja pada **app/Filament/Resources/PesananResource.php**, tepat pada method **table()**, kita akan modifikasi chain method yang namanya **filters()**, Buka file nya dan silakan tambahkan filter seperti:

|  |
| --- |
| *app/Filament/Resources/PesananResource.php* |
| public static function table(Table $table): Table      {          return $table              ->defaultSort('created\_at', 'desc')              ->columns(self::getTableColumns())              ->filters([                  SelectFilter::make('status')                      ->options(StatusPesanan::class),                  SelectFilter::make('metode\_pembayaran')                      ->multiple()                      ->options(MetodePembayaran::class),                  Filter::make('created\_at')                      ->form([                          DatePicker::make('created\_from')                              ->maxDate(fn (Forms\Get $get) => $get('end\_date') ?: now())                              ->native(false),                          DatePicker::make('created\_until')                              ->native(false)                              ->maxDate(now()),                      ])                      ->query(function (Builder $query, array $data): Builder {                          return $query                              ->when(                                  $data['created\_from'],                                  fn (Builder $query, $date): Builder => $query->whereDate('created\_at', '>=', $date),                              )                              ->when(                                  $data['created\_until'],                                  fn (Builder $query, $date): Builder => $query->whereDate('created\_at', '<=', $date),                              );                      }),              ])              ->actions([                  EditAction::make(),                  Action::make('ubah-transaksi')                      ->label('Edit Transaksi')                      ->icon('heroicon-o-pencil')                      ->url(fn($record) => "/pesanans/{record->nomor\_pesanan}"),              ])              ->bulkActions([                  Tables\Actions\BulkActionGroup::make([                      Tables\Actions\DeleteBulkAction::make(),                  ]),              ]);      } |

Jika Anda mendapatkan error, itu wajar, karena mungkin Builder class tidak ditemukan. Maka oleh karena itu, silakan import class-class yang mungkin tidak ada di dalam PesananResource ini.

|  |
| --- |
| *app/Filament/Resources/PesananResource.php* |
| use Filament\Forms;  use Filament\Tables;  use App\Models\Pesanan;  use Filament\Forms\Form;  use Filament\Tables\Table;  use App\Enums\StatusPesanan;  use App\Enums\MetodePembayaran;  use Filament\Resources\Resource;  use Filament\Tables\Actions\Action;  use Filament\Tables\Filters\Filter;  use Filament\Forms\Components\Group;  use Filament\Forms\Components\Select;  use Filament\Forms\Components\Section;  use Filament\Tables\Actions\EditAction;  use Filament\Tables\Columns\TextColumn;  use Filament\Forms\Components\TextInput;  use Filament\Forms\Components\DatePicker;  use Filament\Tables\Filters\SelectFilter;  use Illuminate\Database\Eloquent\Builder;  use Filament\Tables\Columns\Summarizers\Sum;  use App\Filament\Resources\PesananResource\Pages;  use Exception; |

Dengan begitu, maka sekarang Anda harusnya akan sudah punya filter tepat pada atas kanan tabel ini.

### Tabs

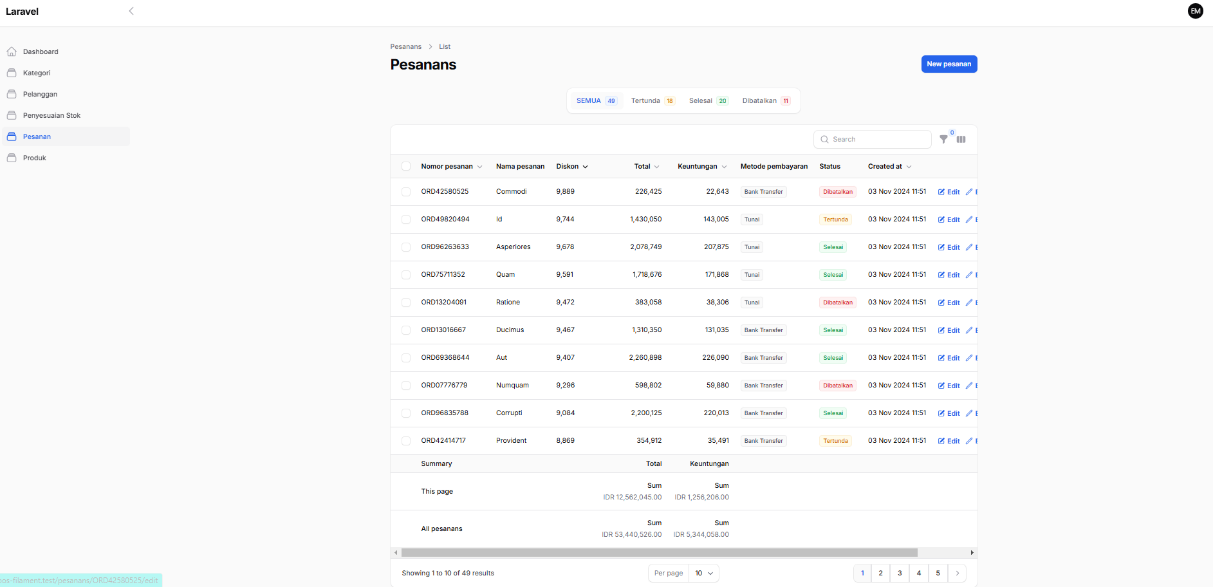
Selain filter, kita juga implementasi tab di sini, saya akan mencontohkan untuk menunjukkan tab dari status yang telah ada. Buka file **app/Filament/Resources/PesananResource/Pages/ListPesanans.php**, dan silakan tambahkan method baru dengan nama **getTabs()**, dan implementasikan tab nya seperti berikut:

|  |
| --- |
| *app/Filament/Resources/PesananResource/Pages/ListPesanans.php* |
| public function getTabs(): array      {          $statuses = collect([              'semua' => ['label' => 'SEMUA', 'badgeColor' => 'primary', 'status' => null],              StatusPesanan::TERTUNDA->name => ['label' => Tertunda, 'badgeColor' => 'warning', 'status' => StatusPesanan::TERTUNDA],              StatusPesanan::SELESAI->name => ['label' => 'Selesai', 'badgeColor' => 'success', 'status' => StatusPesanan::SELESAI],              StatusPesanan::DIBATALKAN->name => ['label' => 'Dibatalkan', 'badgeColor' => 'danger', 'status' => StatusPesanan::DIBATALKAN],          ]);          return $statuses->mapWithKeys(function ($data, $key){              $badgeCount = is\_null($data['status'])                  ? Pesanan::count()                  : Pesanan::where('status', $data['status'])->count();              return [$key => Tab::make($data['label'])                  ->badge($badgeCount)                  ->modifyQueryUsing(fn ($query) => is\_null($data['status']) ? $query : $query->where('status', $data['status']))                  ->badgeColor($data['badgeColor'])];          })->toArray();      } |

Tentu ada beberapa class yang harus di import tepat di atas class nya, beberapa class tersebut bisa di lihat di bawah:

|  |
| --- |
| *app/Filament/Resources/PesananResource/Pages/ListPesanans.php* |
| use Filament\Actions;  use App\Models\Pesanan;  use App\Enums\StatusPesanan;  use Filament\Resources\Components\Tab;  use Filament\Resources\Pages\ListRecords;  use App\Filament\Resources\PesananResource; |

Setelah semua nya Anda lakukan, sekarang harusnya kita mendapatkan hasil yang kurang lebih seperti:



### Default Pesanan

Pada saat ini, daftar pesanan yang di tampilkan itu tidak dari yang terakhir di input, sehingga jika Anda perhatikan baik-baik, ketika Anda buat pesanan baru, dia tidak akan otomatis berada di paling atas, untuk itu kita perlu menambahkan chain method pada table tersebut dengan nama **defaultSort()** . Anda bisa lihat implementasi nya di bawah:

|  |
| --- |
| *app/Filament/Resources/PesananResource/Pages/ListPesanans.php* |
| public static function table(Table $table): Table      {          return $table              ->defaultSort('created\_at', 'desc')              ->columns(self::getTableColumns())              ->filters([...])              ->actions([...])              ->bulkActions([...]);      } |

Perhatikan bahwa di table ini, saya membuat sort mereka default menggunakan created\_at dengan descending order. Sehingga data yang muncul pertama kali adalah data yang terakhir kali di input.

## Custom Action

Pada bagian ini kita akan belajar bagaimana cara memodifikasi aksi pada daftar order ini, tidak hanya kita akan belajar bagaimana cara membuat print, tapi kita juga akan merubah semua aksi yang ada di kanan pada setiap record nya.

### Laravel DOM PDF

buka terminal Anda dan silakan install package nya.

|  |
| --- |
| *Terminal* |
| **composer require barryvdh/laravel-dompdf** |

Selanjutnya adalah optional, bisa di publish, bisa juga tidak.

|  |
| --- |
| *Terminal* |
| **php artisan vendor:publish --provider="Barryvdh\DomPDF\ServiceProvider"** |

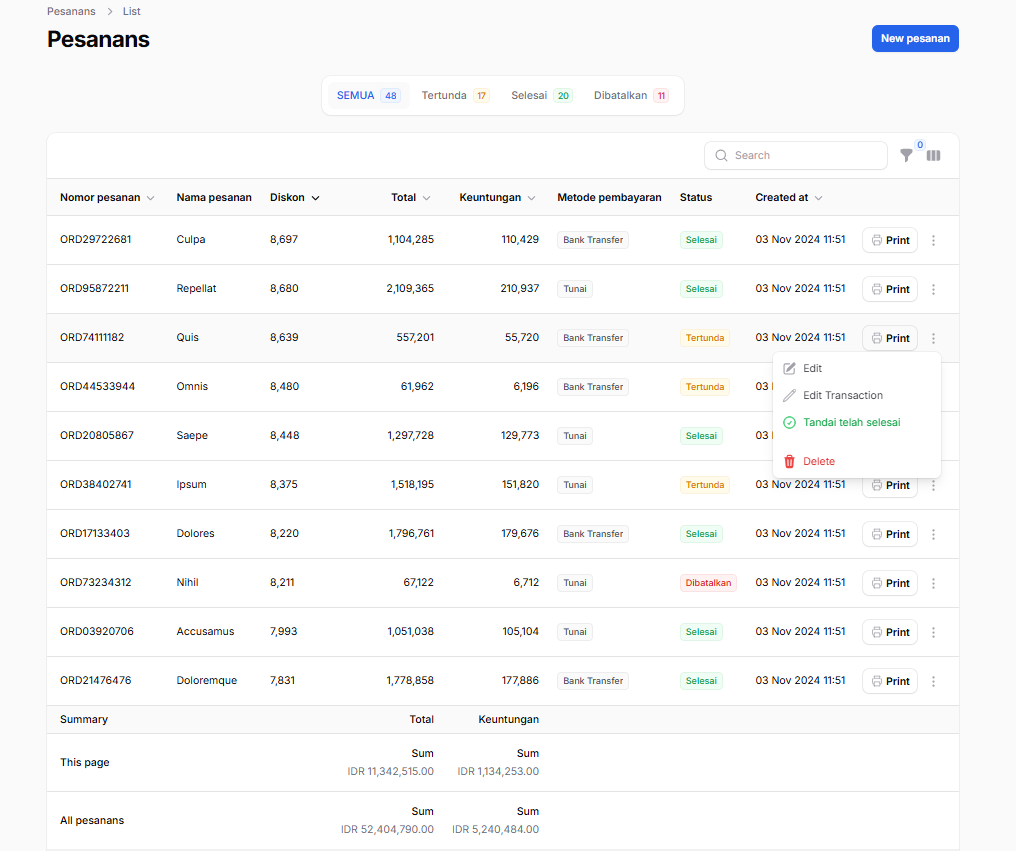
Setelah itu, sekarang bisa lanjut untuk membuat print action yang ada pada **app/Filament/Resources/PesananResource.php**. Buka filenya dan silakan tambahkan kode ini tepat pada method **table()** seperti:

|  |
| --- |
| *app/Filament/Resources/PesananResource.php* |
| public static function table(Table $table): Table      {          return $table              ->defaultSort('created\_at', 'desc')              ->columns(self::getTableColumns())              ->filters([                  SelectFilter::make('status')                      ->options(StatusPesanan::class),                  SelectFilter::make('metode\_pembayaran')                      ->multiple()                      ->options(MetodePembayaran::class),                  Filter::make('created\_at')                      ->form([                          DatePicker::make('created\_from')                              ->maxDate(fn(Forms\Get $get) => $get('end\_date') ?: now())                              ->native(false),                          DatePicker::make('created\_until')                              ->native(false)                              ->maxDate(now()),                      ])                      ->query(function (Builder $query, array $data): Builder {                          return $query                              ->when(                                  $data['created\_from'],                                  fn(Builder $query, $date): Builder => $query->whereDate('created\_at', '>=', $date),                              )                              ->when(                                  $data['created\_until'],                                  fn(Builder $query, $date): Builder => $query->whereDate('created\_at', '<=', $date),                              );                      }),              ])              ->actions([                  Action::make('print')                      ->button()                      ->color('gray')                      ->icon('heroicon-o-printer')                      ->action(function (Pesanan $record) {                          $pdf = Pdf::loadView('pdf.print-pesanan', [                              'pesanan' => $record,                          ]);                          return response()->streamDownload(function () use ($pdf) {                              echo $pdf->stream();                          }, 'receipt-' . $record->nomor\_pesanan . '.pdf');                      }),                  ActionGroup::make([                      EditAction::make()                          ->color('gray'),                      Action::make('edit-transaksi')                          ->visible(fn(Pesanan $record) => $record->status === StatusPesanan::TERTUNDA)                          ->label('Edit Transaction')                          ->icon('heroicon-o-pencil')                          ->url(fn($record) => "/pesanans/{$record->nomor\_pesanan}"),                      Action::make('tandai-telah-dibaca')                          ->visible(fn(Pesanan $record) => $record->status === StatusPesanan::TERTUNDA)                          ->requiresConfirmation()                          ->icon('heroicon-o-check-circle')                          ->color('success')                          ->action(fn(Pesanan $record) => $record->markAsComplete())                          ->label('Tandai telah selesai'),                      Action::make('divider')->label('')->disabled(),                      DeleteAction::make()                          ->before(function (Pesanan $pesanan) {                              $pesanan->detailPesanans()->delete();                              $pesanan->delete();                          }),                  ])                      ->color('gray'),              ])              ->bulkActions([                  BulkActionGroup::make([                      DeleteBulkAction::make()                          ->before(function (\Illuminate\Support\Collection $records) {                              $records->each(fn(Pesanan $pesanan) => $pesanan->detailPesanans()->delete());                          }),                  ]),              ]);      } |

Setelah itu, silakan buat 1 folder tepat di dalam folder **resources/views** dengan nama **pdf** dan di dalamnya silakan buat file dengan nama **print-pesanan.blade.php**. Silakan buka filenya dan buat print markup nya seperti:

|  |
| --- |
| *resources/views/pdf/print-pesanan.blade.php* |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <meta http-equiv="X-UA-Compatible" content="ie=edge">      <link rel="stylesheet" href="style.css">      <title>Receipt example</title>      <style>          @page {              size: 3.15in;              margin: 0;          }          body {              margin: 0;              padding: 0.2in;              width: auto;              line-height: 1.2;              font-size: 7pt;              font-family: Arial, Helvetica Neue, Helvetica, sans-serif;          }          table {              width: 100%;              table-layout: fixed;          }          th, td {              padding: 2px 0;              text-align: left          }          .title {              font-size: 10pt;              font-weight: bold;              text-align: center;              margin-bottom: -5px;          }          .subtitle {              text-align: center;              color: #71717a;          }      </style>  </head>  <body>  <div>      <h1 class="title">{{ config('app.name') }}</h1>      <p class="subtitle">{{ now()->format('d F Y H:i') }}</p>      <table>          <thead>          <tr>              <th style="width: 6%;">#</th>              <th style="width: 60%">Nama</th>              <th style="text-align: right;width: 40%">Kuantitas</th>              <th style="text-align: right">Harga</th>          </tr>          </thead>          <tbody>          @foreach ($pesanan->detailPesanans as $detail)              <tr>                  <td style="width: 6%; font-family: monospace">{{ $loop->iteration }}.</td>                  <td style="width: 60%">{{ $detail->produk->nama }}</td>                  <td style="width: 40%; letter-spacing: -1.5px; text-align: right;font-family: 'monospace'">{{ $detail->kuantitas }} x {{ number\_format($detail->harga, 0, '.', '.') }}</td>                  <td style="text-align: right;font-family: 'monospace'">{{ number\_format($detail->harga \* $detail->kuantitas, 0, '.', '.') }}</td>              </tr>          @endforeach          <tr><td/><td/><td/><td/></tr><tr><td/><td/><td/><td/></tr>          <tr><td/><td/><td/><td/></tr><tr><td/><td/><td/><td/></tr>          <tr>              <td colspan="3" style="text-align: right;">Total:</td>              <td style="text-align: right; font-weight: bold; font-family: 'monospace'">Rp {{ number\_format($pesanan->total, 0, '.', '.') }}</td>          </tr>          <tr>              <td colspan="3" style="text-align: right;">Diskon:</td>              <td style="text-align: right; font-weight: bold; font-family: 'monospace'">Rp {{ number\_format($pesanan->diskon ?? '0', 0, '.', '.') }}</td>          </tr>          <tr>              <td colspan="3" style="text-align: right;">Total Setelah Diskon:</td>              <td style="text-align: right; width: 40%; font-weight: bold; font-family: 'monospace', sans-serif">                  Rp {{ number\_format($pesanan->total - $pesanan->diskon, 0, '.', '.') }}              </td>          </tr>          </tbody>      </table>  </div>  </body>  </html> |

Dengan begitu, maka harusnya tampilan nya kurang lebih akan seperti ini:



## Ekspor Data

Dalam bagian ini kita akan belajar tentang bagaimana melakukan export menggunakan fitur yang diberikan oleh filament itu sendiri.

Untuk melakukan itu semua, ada beberapa yang mesti kita publish terlebih dahulu. Pertama sekali, karena export ini memerlukan yang namanya notification, tentu kita butuh notification table. Tidak hanya itu tentu nya, kita juga butuh yang namanya *batches-table*.

### Generate Table

buka terminal Anda dan silakan jalankan perintah berikut:

|  |
| --- |
| *Terminal* |
| **php artisan queue:batches-table**  **php artisan notifications:table**  **php artisan queue:table**  **php artisan vendor:publish --tag=filament-actions-migrations** |

Saya sengaja juga menpublish table untuk queue nya, agar nanti jika Kamu berfikir ingin menggunakan queue, tinggal ganti QUEUE\_CONNECTION nya saja menjadi database instead of sync.

Setelah itu, silakan lakukan migrate dengan menjalankan perintah berikut:

|  |
| --- |
| *Terminal* |
| **php artisan migrate** |

### Generate Ekspor

sekarang silakan jalankan perintah untuk membuat sebuah exporter nya dengan perintah berikut:

|  |
| --- |
| *Terminal* |
| **php artisan make:filament-exporter Pesanan** |

Dengan begitu, maka kita akan mendapatkan file baru di \*\***app/Filament/Exports/PesananExporter.php**.

### Modifikasi Panel

Karena sekarang kita butuh notification, jadi kita perlu mengaktifkan nya pada file **app/Providers/Filament/AppPanelProvider.php**, dan silakan tambahkan / chain method **->databaseNotifications()** pada panel ini. Di sini saya akan menambahkan nya tepat dibawah ->default() seperti:

|  |
| --- |
| *app/Providers/Filament/AppPanelProvider.php* |
| class AppPanelProvider extends PanelProvider  {      public function panel(Panel $panel): Panel      {          return $panel              ->default()              ->databaseNotifications()              ->id('app')              ->path('/')              ->login()              ->colors([                  'primary' => Color::Blue,              ])              ->discoverResources(in: app\_path('Filament/Resources'), for: 'App\\Filament\\Resources')              ->discoverPages(in: app\_path('Filament/Pages'), for: 'App\\Filament\\Pages')              ->pages([                  Pages\Dashboard::class,              ])              ->discoverWidgets(in: app\_path('Filament/Widgets'), for: 'App\\Filament\\Widgets')              ->widgets([                  Widgets\AccountWidget::class,                  Widgets\FilamentInfoWidget::class,              ])              ->middleware([                  EncryptCookies::class,                  AddQueuedCookiesToResponse::class,                  StartSession::class,                  AuthenticateSession::class,                  ShareErrorsFromSession::class,                  VerifyCsrfToken::class,                  SubstituteBindings::class,                  DisableBladeIconComponents::class,                  DispatchServingFilamentEvent::class,              ])              ->authMiddleware([                  Authenticate::class,              ])              ->sidebarCollapsibleOnDesktop()              ->viteTheme('resources/css/filament/app/theme.css');      }  } |

Dengan begitu, jika Kamu muat ulang di browser, akan ada tombol / icon notifikasi tepat di samping avatar profil kita (pojok kanan atas).

### Sediakan Kolom

Secara Bawaan, file ini akan menghasilkan 2 method yaitu getColumns() dan getCompletedNotificationBody(), dan yang pertama saya akan menunjukkan Kamu bagian dari method **getColumns():**

|  |
| --- |
| *app/Filament/Exports/PesananExporter.php* |
| public static function getColumns(): array      {          return [              ExportColumn::make('pelanggan.nama'),              ExportColumn::make('nomor\_pesanan'),              ExportColumn::make('nama\_pesanan'),              ExportColumn::make('diskon'),              ExportColumn::make('total'),              ExportColumn::make('metode\_pembayaran')->formatStateUsing(fn($state) => $state->value),              ExportColumn::make('status')->formatStateUsing(fn($state) => $state->value),              ExportColumn::make('created\_at'),          ];      } |

Perhatikan, bahwa di sini saya sudah memodifikasi metode\_pembayaran dan juga status karena pada dasarnya pada field ini, kita menggunakan enum, sehingga filament export akan menganggap itu tidak string, karena semua format yang di harapkan itu adalah string.

Untuk method getCompletedNotificationBody tentu tidak perlu diganti unless Anda butuh memodifikasi isi nya. Untuk kasus kita, saat ini tidak perlu.

### Tentukan Format

Secara bawaan ekspor akan menghasilkan 2 format yaitu Csv dan Juga Xlsx, namun untuk kasus ini, sepertinya kita tidak butuh Csv, jika memang Anda mau, tentu Anda tidak perlu membuat function yang akan kita buat ini.

Silakan buat 1 function dengan nama getFormats() dan isinya silakan buat seperti:

|  |
| --- |
| *app/Filament/Exports/PesananExporter.php* |
| public function getFormats(): array      {          return [              ExportFormat::Xlsx,          ];      } |

Jadi untuk keseluruhan class ini kurang lebih seperti berikut

|  |
| --- |
| *app/Filament/Exports/PesananExporter.php* |
| <?php  namespace App\Filament\Exports;  use App\Models\Pesanan;  use Filament\Actions\Exports\Exporter;  use Filament\Actions\Exports\ExportColumn;  use Filament\Actions\Exports\Models\Export;  use Filament\Actions\Exports\Enums\ExportFormat;  class PesananExporter extends Exporter  {      protected static ?string $model = Pesanan::class;      public static function getColumns(): array      {          return [              ExportColumn::make('pelanggan.nama'),              ExportColumn::make('nomor\_pesanan'),              ExportColumn::make('nama\_pesanan'),              ExportColumn::make('diskon'),              ExportColumn::make('total'),              ExportColumn::make('metode\_pembayaran')->formatStateUsing(fn($state) => $state->value),              ExportColumn::make('status')->formatStateUsing(fn($state) => $state->value),              ExportColumn::make('created\_at'),          ];      }      public static function getCompletedNotificationBody(Export $export): string      {          $body = 'Pesanan kamu sudah berhasil di ekspor dan ' . number\_format($export->successful\_rows) . ' ' . str('row')->plural($export->successful\_rows) . ' exported.';          if ($failedRowsCount = $export->getFailedRowsCount()) {              $body .= ' ' . number\_format($failedRowsCount) . ' ' . str('row')->plural($failedRowsCount) . ' Gagal di ekspor.';          }          return $body;      }      public function getFormats(): array      {          return [              ExportFormat::Xlsx,          ];      }  } |

***#catatan:*** *Hapus method getFormats jika Anda ingin export 2 format.*

### Ekspor Action

Setelah semua setup, waktunya kita menambahkan yang namanya ekspor actionnya. Buka kembali file  **app/Filament/Resources/PesananResource.php**, kita perlu menambah 1 method lagi tepat di bawah bulkActions dengan nama headerActions, dan isinya kurang lebih seperti:

|  |
| --- |
| *app/Filament/Resources/PesananResource.php* |
| ->headerActions([                  ExportAction::make()                      ->label('Ekspor Excel')                      ->fileDisk('public')                      ->color('success')                      ->icon('heroicon-o-document-text')                      ->exporter(PesananExporter::class),              ]); |

### Queue in Action

Bayangkan jika data yang ingin di export itu ribuan, tentu untuk aksi yang tergantung pada request dari server itu tidak akan mampu mengatasi itu, terlebih lagi itu akan bergantung dengan kesanggupan dari device masing-masing. Maka untuk itu, peran dari queue (work behind the scene) sangat berguna di sini. Ingat, pada saat kita generate table saya juga membuat yang nama nya **queue:table**, itu artinya kita memang berencana untuk memungsikan queue dalam hal ini. Untuk melakukan itu, silakan buka file .env untuk mengubah default dari **QUEUE\_CONNECTION=sync** menjadi **QUEUE\_CONNECTION=database**. Sehingga dengan begini, kita memberi tahu bahwa untuk menjalankan export ini itu menggunakan **queue**. Buka terminal Anda dan silakan jalankan perintah berikut:

|  |
| --- |
| *Terminal* |
| **php artisan queue:work** |

Tidak hanya dia queue export kita, namun dia juga akan bekerja sekalian untuk notifikasi database nya, sehingga aplikasi kita akan berjalan sangat smooth tanpa di bebani oleh request server.

## View Detail Pesanan

Kita memang telah membuat beberapa tombol pada table pesanan ini, seperti tombol edit order, edit transaksi, menandai status komplit, dan delete. Namun, kita belum mengimplementasikan view terhadap pesanan ini, ya kita bisa melihat transaksi dari tombol edit, namun jika transaksi berstatus selesai, tombol itu akan hilang. Maka oleh karena itu, pada bagian ini kita akan belajar bagaimana cara membuatnya.