



Learnings from Laravel



Ronny Hartenstein
app dev team lead
Axilaris GmbH
@rhflow_de



Legacy Code ~~Here~~

Archeological
Excavation Leader

side projects as chance

Laravel, Symfony, Zend,
CakePHP, Slim, ...

pick one

PensionManager

Domain Model / Entities:

Mandant (client)

→ Arbeitgeber (employer)

→ Standort (location)

→ Arbeitnehmer (employee)

→ Zusage (contract)

→ Vertragssereignisse (events)

Dev requirements:

- Routing
- ORM
- Feature tests
- “shiny features for fun”

we choose Laravel

(5.1 LTS, start was late 2017)


for PensionManager


(sorry, not opensource)

lets have a look inside


- routing
- controllers and actions (CQRS like)
- template (Blade)


<https://pensionmanager.axilaris.de/> → Demo


 Demo Admin


 Dashboard

VERWALTUNG


 Arbeitgeber


 Standorte


 Arbeitnehmer

 Zusagen

ADMINISTRATION

 Nutzer

 Historie



















+ Hinzufügen

Suche nach... ▾

Demo Admin ▾

Sie befinden sich im Demo-Modus. Verwenden Sie keine sensiblen Daten, da diese von allen einsehbar sind. Alle Daten werden nächtlich zurückgesetzt.

 Arbeitnehmer

Personal-Nr.	Name	Arbeitgeber, Standort		Letzte Änderung	
AN2	Caro Ass Lohnunterbrechung	Demo GmbH 	Zwickau 	21.09.2018 00:00:01 Uhr	
AN5	Hans Dampf	Demo GmbH 	Detmold 	21.09.2018 00:00:01 Uhr	
AN3	Donna Keil	Demo GmbH 	Zwickau 	21.09.2018 00:00:01 Uhr	
AN1	Dean Orm Führungskraft	Demo GmbH 	Zwickau 	21.09.2018 00:00:01 Uhr	
AN4	Inge Wahrsam Neueinstellung	Demo GmbH 	Zwickau 	21.09.2018 00:00:01 Uhr	

Framework and MVC basics

lets explore Arbeitgeber..

// READ

```
Route::get('/arbeitgeber', 'ArbeitgeberController@index')->name('arbeitgeber');  
Route::get('/arbeitgeber/{id}', 'ArbeitgeberController@show')->name('arbeitgeber.show');
```

// CREATE

```
Route::get('/arbeitgeber/create', 'ArbeitgeberController@create')->name('arbeitgeber.create');  
Route::post('/arbeitgeber', 'ArbeitgeberController@store')->name('arbeitgeber.store');
```

// UPDATE

```
Route::get('/arbeitgeber/{id}/edit', 'ArbeitgeberController@edit')->name('arbeitgeber.edit');  
Route::put('/arbeitgeber/{id}', 'ArbeitgeberController@update')->name('arbeitgeber.update');
```

// DELETE

```
Route::delete('/arbeitgeber/{id}/soft', 'ArbeitgeberController@soft_delete')->name('arbeitgeber.del
```

```
public function index()
{
    $arbeitnehmers = Arbeitnehmer::orderBy('name', 'ASC')->orderBy('vorname', 'ASC')
        ->with(['standort', 'standort.arbeitgeber'])
        ->paginate(20);
    return view( view: 'verwaltung/arbeitnehmer/index', ['arbeitnehmers' => $arbeitnehmers]);
}
```

```
public function show(Request $request, int $id)
{
    $this->checkCurrentUserCan( rolle: 'lesen', className: Arbeitnehmer::class);

    $arbeitnehmer = Arbeitnehmer::with([
        'standort.arbeitgeber',
        'zusagen',
        'vertragsereignisse' => function($query) {
            $query->orderBy('datum', 'DESC');
        }
    ])->findOrFail($id);
    return view( view: 'verwaltung/arbeitnehmer/show', ['arbeitnehmer' => $arbeitnehmer]);
}
```

```

<tbody>
@foreach($arbeitnehmers as $arbeitnehmer)
<tr>
    <td>
        {{ $arbeitnehmer->personalnummer }}
    </td>
    <td>
        @include('verwaltung/arbeitsnehmer/_status', ['arbeitsnehmer' => $arbeitnehmer])
    </td>
    <td>
        @php
            $standort = $arbeitnehmer->standort;
            $arbeitsgeber = $arbeitnehmer->standort->arbeitsgeber;
        @endphp
        {{ $arbeitsgeber->name.' '.$arbeitsgeber->rechtsform }}
        <a href="{{ route('arbeitsgeber.show', $arbeitsgeber->id) }}" class="btn btn-xs btn-default"><
        {{ $standort->ort }}
        <a href="{{ route('standorte.show', $standort->id) }}" class="btn btn-xs btn-default"><i clas
    </td>
    <td>{{ Helper::localize($arbeitnehmer->updated_at) }} Uhr</td>
    <td class="text-right">
        @if (Helper::currentUserCan('lesen', \App\Models\Arbeitsnehmer::class))
            <a href="{{ route('arbeitsnehmer.show', $arbeitnehmer->id) }}" class="btn btn-xs btn-prim
        @endif
    </td>
</tr>
</tbody>
@endforeach
</tbody>

```



```
public function edit(Request $request, int $id)
{
    $this->checkCurrentUserCan( rolle: 'schreiben', className: Arbeitnehmer::class);
    $arbeitnehmer = Arbeitnehmer::with( relations: 'standort.arbeitgeber')->findOrFail($id);
    $standorte = Standort::with( relations: 'arbeitgeber')->get();
    return view( view: 'verwaltung/arbeitnehmer/edit', ['arbeitnehmer' => $arbeitnehmer, 'standorte' => $standorte]);
}
```

```
public function update(Request $request, int $id)
{
    $this->checkCurrentUserCan( rolle: 'schreiben', className: Arbeitnehmer::class);
    $validated_data = $this->execute_validations($request, $id);
    $arbeitnehmer = Arbeitnehmer::findOrFail($id);
    $arbeitnehmer->fill($validated_data);
    $arbeitnehmer->standort_id = $validated_data['standort_id'];
    if ($arbeitnehmer->save()) {
        $flash = ['flash_notice' => 'Der Arbeitnehmer wurde erfolgreich gespeichert.'];
    } else {
        $flash = ['flash_error' => 'Der Arbeitnehmer konnte nicht gespeichert werden.'];
    }
    return redirect()->route( route: 'arbeitnehmer.show', [ id' => $arbeitnehmer->id])->with($flash);
}
```

and now ...

special fancy stuff



DB: Eloquent ORM, Query Builder

Model

```
class Arbeitnehmer extends Model
{
    use SoftDeletes;
    use \Znck\Eloquent\Traits\BelongsToThrough;

    protected static function boot() {
        parent::boot();
        static::addGlobalScope(new MandantScope());
        static::addGlobalScope(new UserStandortScope());
    }
    public function mandant() {
        return $this->belongsTo( related: 'App\Models\Mandant');
    }
    public function standort() {
        return $this->belongsTo( related: 'App\Models\Standort');
    }
    public function arbeitgeber() {
        return $this->belongsToThrough( related: 'App\Models\Arbeitgeber', through: 'App\Models\Standort');
    }
    public function zusagen() {
        return $this->hasMany( related: 'App\Models\Zusage');
    }
    public function historien() {
        return $this->morphMany( related: 'App\Models\Historie', name: 'owner');
    }
}
```

Migrations

```
class CreateArbeitnehmerTable extends Migration
{
    public function up() {
        Schema::create( table: 'arbeitnehmer', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->increments( column: 'id');
            $table->integer( column: 'mandant_id')->unsigned();
            $table->integer( column: 'standort_id')->unsigned();
            $table->string( column: 'name');
            $table->string( column: 'vorname');
            $table->date( column: 'diensteintritt');
            $table->date( column: 'geburtstag');
            $table->boolean( column: 'status_neueinstellung')->default(false);
            $table->boolean( column: 'status_lohnunterbrechung')->default(false);
            $table->boolean( column: 'status_fuehrungskraft')->default(false);
            $table->boolean( column: 'status_zweites_dienstverhaeltnis')->default(false);
            $table->boolean( column: 'status_minijobber')->default(false);
            $table->boolean( column: 'status_rentenversicherungspflichtig')->default(false);
            $table->timestamps();
            $table->softDeletes();
            $table->foreign( columns: 'mandant_id')->references('id')->on('mandanten');
            $table->foreign( columns: 'standort_id')->references('id')->on('standorte');
        });
    }

    public function down() {
        Schema::dropIfExists( table: 'arbeitnehmer');
    }
}
```

Query Builder

einfache SQL als Beispiel

```

public static function anzahlArbeitnehmerMitZusage($mandant_id)
{
    $result = DB::select('
        SELECT COUNT(`an`.`aggregate`) AS `anzahl` from (
            SELECT `arbeitnehmer`.`id` as `aggregate`
            FROM `arbeitnehmer`
            LEFT JOIN `zusagen` ON `arbeitnehmer`.`id` = `zusagen`.`arbeitnehmer_id`
            WHERE `arbeitnehmer`.`mandant_id` = ?
                AND `zusagen`.`mandant_id` = ?
                AND `zusagen`.`id` IS NOT NULL
            GROUP BY `arbeitnehmer`.`id`) an
        ', [$mandant_id, $mandant_id]);
    return $result[0]->anzahl;
}

```

complex static SQLs? don't write OOP-code

Transactions

```
DB::beginTransaction();

$newGesellschaft = Gesellschaft::create($validated_data['gesellschaft']);

$gesellschaft1->nachfolgegesellschaft_id = $newGesellschaft->id;
$gesellschaft1->save();
$gesellschaft2->nachfolgegesellschaft_id = $newGesellschaft->id;
$gesellschaft2->save();

$zusagenCount = $this->zusagenUebernehmen($gesellschaft1, $gesellschaft2, $newGesellschaft);

$deleter1 = new GesellschaftDeleter($gesellschaft1);
$deleter1->run();
$errors = $deleter1->errors();

$deleter2 = new GesellschaftDeleter($gesellschaft2);
$deleter2->run();
$errors = array_merge($errors, $deleter2->errors());

$this->fusionsHistorySpeichern($gesellschaft1, $gesellschaft2, $newGesellschaft);

if (count($errors) > 0) {
    DB::rollBack();
} else {
    DB::commit();
}
```


ORM

```
$arbeitnehmers = Arbeitnehmer::orderBy('name', 'ASC')->orderBy('vorname', 'ASC')  
->with(['standort', 'standort.arbeitgeber'])  
->paginate(20);
```

```
$arbeitnehmer = Arbeitnehmer::with([  
    'standort.arbeitgeber',  
    'zusagen',  
    'vertragsereignisse' => function($query) {  
        $query->orderBy('datum', 'DESC');  
    }  
])->findOrFail($id);
```

```
$arbeitnehmer = Arbeitnehmer::with( relations: 'standort.arbeitgeber')->findOrFail($id);  
$standorte = Standort::with( relations: 'arbeitgeber')->get();
```

use “artisan tinker” (PsySh) to play with it

```
>>> Mandant::get()
=> Illuminate\Database\Eloquent\Collection {#3226
  all: [
    App\Models\Mandant {#3227
      id: 3,
      name: "Demo Admin",
      lizenzmodell: "premium",
      iban: "DE12 1234 5678 90",
      bic: "ABCDE99XXX",
      rechnungsadresse: ""
        Demo Adminz.Hd. Hr. Mustermann\n
        Altersruhe 67\n
        12345 Musterstadt
        "",
      created_at: "2018-09-23 15:09:07",
      updated_at: "2018-09-23 15:09:07",
      deleted_at: null,
    },
  ],
}
```

```
>>> Arbeitnehmer::where('name','=','Keil')->get()
=> Illuminate\Database\Eloquent\Collection {#3213
  all: [
    App\Models\Arbeitnehmer {#3204
      id: 3,
      mandant_id: 3,
      standort_id: 3,
      personalnummer: "AN3",
      name: "Keil",
      vorname: "Donna",
      diensteintritt: "2008-04-01",
      geburtstag: "1976-07-19",
      status_neueinstellung: 0,
      status_lohnunterbrechung: 0,
      status_fuehrungskraft: 0,
      status_zweites_dienstverhaeltnis: 0,
      status_minijobber: 0,
      status_rentenversicherungspflichtig: 0,
      created_at: "2018-09-23 15:09:07",
      updated_at: "2018-09-23 15:09:07",
      deleted_at: null,
    },
  ],
}
```


external init

“composer require illuminate/database”

```
$capsule = new Illuminate\Database\Capsule\Manager();

touch( filename: __DIR__.'/../events.sqlite');
$capsule->addConnection([
    'driver'      => 'sqlite',
    'database'    => __DIR__.'/../events.sqlite',
]);
$capsule->setAsGlobal();
$capsule->bootEloquent();
```

Scope



```
class Arbeitnehmer extends Model
{
    protected static function boot() {
        parent::boot();
        static::addGlobalScope(new MandantScope());
        static::addGlobalScope(new UserStandortScope());
    }
}
```

```
class MandantScope implements Scope
{
    public function apply(Builder $builder, Model $model)
    {
        if (!empty(Auth::id())) {
            $builder->where(
                column: $model->getTable().'.mandant_id',
                operator: '=',
                Auth::user()->mandant_id
            );
        }
    }
}
```

Auth



```
→ laravel-test-auth git:(master) php artisan make:auth
```

```
Authentication scaffolding generated successfully.
```

```
→ laravel-test-auth git:(master) X gst
```

```
On branch master
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git checkout -- <file>..." to discard changes in working directory)
```

```
modified: routes/web.php
```

```
Untracked files:
```

```
(use "git add <file>..." to include in what will be committed)
```

```
app/Http/Controllers/HomeController.php
```

```
resources/views/auth/
```

```
resources/views/home.blade.php
```

```
resources/views/layouts/
```



Controller Middleware

```
class ArbeitnehmerController extends Controller
{
    // ..

    public function __construct()
    {
        $this->middleware( middleware: 'auth');
        $this->middleware( middleware: 'auth_no_role:backend');
    }
}
```

```
class MandantenController extends Controller
{
    public function __construct()
    {
        $this->middleware( middleware: 'auth');
        $this->middleware( middleware: 'auth_role:backend');
    }
}
```

```
class UsersController extends Controller
{
    public function __construct()
    {
        $this->middleware( middleware: 'auth');
        $this->middleware( middleware: 'auth_role:admin');
    }
}
```



```
class AuthRole
{
    public function handle($request, Closure $next, $role = '')
    {
        if ($request->user()->hasRole($role) === false) {
            return redirect()->route( route: 'home');
        }

        return $next($request);
    }
}
```

```
class AuthNoRole
{
    public function handle($request, Closure $next, $role = '')
    {
        if ($request->user()->hasRole($role) === true) {
            return redirect()->route( route: 'home');
        }

        return $next($request);
    }
}
```

“but D.R.Y.!”

i don't care (here)



Feature Tests + Faker test data

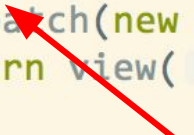
```
public function testShowsArbeitnehmerListOfTheUsersMandant()  
{  
    $arbeiter = factory(Arbeitnehmer::class)->create();  
    $user = factory(User::class)->create([  
        'mandant_id' => $arbeiter->mandant_id,  
    ]);  
  
    $this->assertEquals($user->mandant_id, $arbeiter->mandant_id);  
  
    $this->actingAs($user)  
        ->from(route( name: 'home'))  
        ->get(route( name: 'arbeiter'))  
        ->assertStatus( status: 200)  
        ->assertSee($arbeiter->name)  
    ;  
}
```

```
$factory->define(App\Models\Arbeitnehmer::class, function (Faker $faker) {  
    static $personalnummer = 1;  
  
    $standort = factory('App\Models\Standort')->create();  
  
    return [  
        'mandant_id' => $standort->mandant_id,  
        'standort_id' => $standort->id,  
  
        'personalnummer' => $personalnummer++,  
        'name' => $faker->regexify( regex: '[A-Z][a-z]{5,10}'),  
        'vorname' => $faker->regexify( regex: '[A-Z][a-z]{5,10}'),  
        'diensteintritt' => $faker->dateTimeBetween('-10 years', '-3 years')->format( format: 'Y-m-d'),  
        'geburtstag' => $faker->dateTimeBetween('-60 years', '-18 years')->format( format: 'Y-m-d'),  
        'status_neueinstellung' => false,  
        'status_lohnunterbrechung' => false,  
        'status_fuehrungskraft' => false,  
        'status_zweites_dienstverhaeltnis' => false,  
        'status_rentenversicherungspflichtig' => false,  
        'status_minijobber' => false,  
    ];  
});
```



Events & Listener

```
public function print(Zusage $zusage) {  
    event(new ZusagePrintEvent($zusage));  
    dispatch(new PrintZusageJob($zusage));  
    return view( view: 'zusage.show', ['data' => $zusage, 'print' => true]);  
}
```



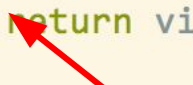
```
class EventServiceProvider extends ServiceProvider  
{  
    protected $listen = [  
        'App\Events\ZusagePrint' => [  
            'App\Listeners\SendZusagePrintingEmail',  
        ],  
    ];  
}
```

```
class SendZusagePrintingEmail  
{  
    public function handle(ZusagePrint $event)  
    {  
        print "\nListener: got event Zusage {$event->zusage->id}";  
    }  
}
```



Job Queue


```
public function print(Zusage $zusage) {  
    event(new ZusagePrintEvent($zusage));  
    dispatch(new PrintZusageJob($zusage));  
    return view( view: 'zusage.show', ['data' => $zusage, 'print' => true]);  
}
```



```
class PrintZusage implements ShouldQueue  
{  
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;  
    protected $zusage;  
  
    public function __construct(Zusage $zusage) {  
        $this->zusage = $zusage;  
    }  
  
    public function handle() {  
        print "\ndrucke Zusage {$this->zusage->id} ..";  
    }  
}
```

php artisan queue:work

there's more:

Validation, Facades, Services,
Emails, Mocking, Encrypter,
Support helper functions

trending: framework less

just use components
using PSR standards

<https://kevinsmith.io/modern-php-without-a-framework>

```
composer create-project \  
    laravel/laravel my-laravel-sandbox
```

or

```
composer require illuminate/database  
composer require illuminate/support  
composer require illuminate/view  
composer require illuminate/http
```



start your side project
now!

learn what and how to
improve your legacy
big-ball-of-mud

thanks for having me :)

feedback:



<https://joind.in/talk/e3803>



Ronny Hartenstein
app dev team lead
Axilaris GmbH
@rhflow_de