This is your regular app structure. Each class type has a separate folder. Nothing unexpected so far.

Dealing with User model

If we use domains to structure our code, where to put the User model? Right into the Shared domain. Every single domain uses it, so it's shared in the application. To do that, we have to change Laravel's User provider configuration in the config/auth.php:

```
'providers' => [
  'users' => [
    'driver' => 'eloquent',
    'model' => \Domain\Shared\Models\User::class,
  ],
],
```

The *model* was changed from the default `App\Models\User::class`. After that, we can move the user into the `Shared\Models` namespace.

BaseModel and factories

In most applications, I use a `BaseModel` that extends Eloquent's model class and is extended by all models. It's also a class that can be placed inside the Shared domain. When using domains, you have to override a method on the model called `newFactory`. This method will return a new factory instance associated with the model. By default, Laravel tries to resolve the factory based on the model's namespace. But in our case, it tries to instantiate a class like `Database\Factories\Domain\Subscriber\Models\SubscriberFactory`. It's clearly not what we want.

First things first, I also namespace the `database/factories` folder by domain: