# Restricting Access with User Authorization

**Mateo Prigl**
Software developer

# Relationship Between Users and Posts

posts table

users table

| id | title | description | user_id |
|---|---|---|---|
| 1 | Hello there | Hello every... | 3 |
| ... | ... | ... | ... |

| id | name | email | ... |
|---|---|---|---|
| 3 | mateo | mateo@mail.com | ... |
| ... | ... | ... | ... |

# Summary

# Adding a Foreign Key Column

## Create a new migration

```
php artisan make:migration add_user_id_to_posts_table
```

## Create a new foreign key column

```
public function up()
{
     Schema::table('posts',
                    function (Blueprint $table)
     {
            $table->foreignId('user_id')
                ->nullable()
                ->constrained()
                ->onDelete('cascade');
     });
}
```

```
public function down()
{
     Schema::table('posts',
                    function (Blueprint $table)
     {
            $table->dropForeign(['user_id']);
            $table->dropColumn('user_id');
     });
}
```

# Recreate the Database

**Rollback all of the migrations and migrate them again**

```
php artisan migrate:refresh
```

# One-to-many Relationship Between Models

**Post belongs to a user**

```
public function user()
{
    return $this->belongsTo(User::class);
}
```

app/Models/Post.php

**User has many posts**

```
public function posts()
{
    return $this->hasMany(Post::class);
}
```

app/Models/User.php

# One-to-many Relationship Between Models

**Create a new post that belongs to user**

```
$validated = $request->validated();

$post = $request->user()->posts()->create($validated);
```

**Retrieve the user who created the post**

```
$post->user;
```

# Create Policies for Authorization

**Create a new policy**

```
php artisan make:policy PostPolicy
```

**Register the policy and assign it to an Eloquent model**

```
use App\Models\Post;
use App\Policies\PostPolicy;
...
    protected $policies = [
        Post::class => PostPolicy::class,
    ];
```

**app/Providers/AuthServiceProvider.php**

# Define Policies

```
use App\Models\Post;
use App\Models\User;
...
class PostPolicy
{
    ...
    public function update(User $user, Post $post)
    {
        return $user->id === $post->user_id;
    }

    public function delete(User $user, Post $post)
    {
        return $user->id === $post->user_id;
    }
}
```

# Authorization Blade Directives

**Render template components based on policy permissions**

```
@can('update', $post)
<a href="#">Edit link</a>
@endcan

@can('delete', $post)
<button>Delete</button>
@endcan
```

# Authorize Controller Actions

**Authorize controller actions with the controllers *authorize* method**

```
public function edit(Post $post)
{
    $this->authorize('update', $post);
    // this will not run if the authorization fails
    ...
}
```

# Authorization with Gates

**Register gates inside of the boot method**

```
public function boot()
{
    Gate::define('admin', function (User $user) {
        return $user->is_admin();
    });
}
```

**app/Providers/AuthServiceProvider.php**

**Use them to implement authorization with the *allow* method**

```
if (! Gate::allows('admin'))
{
    abort(403);
}
```

# Up Next:
# Deployment and Testing