LARAVEL NOVA

- Installazione
Laravel Nova permette di creare un *pannello di controllo* utilizzabile dall'Admin.
A tale scopo dobbiamo specificare che ogni utente che verrà creato sarà di ruolo *User*.

- Nel file *Migration* create_user_table.php:

```
public function up(){
	Schema::create('users', function(Blueprint $table){
		...
		$table->string('email')->unique();
		$table->string('role')->default('user');
	}
}
```

- Possiamo dare al nostro Utente Admin il ruolo Admin dal *Tinker*:

```
php artisan tinker
$u = App\Models\User::first();
$u->role = 'admin';
$u->save();
```

NB: *get()* restituisce una *Collection*, *first()* un elemento singolo.

- Scarichiamo la cartella di Laravel Nova dal sito Link - https://nova.laravel.com/ in ***Releases***.
Per lavorare in locale si può utilizzare una sola *licenza*, mentre per ogni lavoro che faremo per un cliente dovremo comprare una ***licenza singola***.

- Facciamo l'***unzip*** della cartella scaricata, e ***rinominiamola*** 'nova' e la trasciniamo nella cartella del nostro ***progetto.***

- Guardiamo la ***documentazione*** di Laravel Nova per l'installazione.
- Nel *composer.json* inseriamo ***dopo*** la chiusura di **"**require-dev":

```
"repositories":[{
	"type": "path",
	"url": "./nova"
}],
```

- Nel *composer.json* inseriamo nel "require":

```
"laravel/nova": "*"
```

- Nella *bash* installiamo e aggiungiamo la tabella *action events* :

```
composer update
php artisan nova:install
php artisan migrate
php artisan nova:publish //Per aggiungere le views del pannello in resources
```

In *resources\views\vendor/nova/partials* abbiamo accesso alle *views* del pannello di controllo.
In ***meta*** possiamo inserire tutti i riferimenti a link esterni essendo la *head* del nostro pannello.

Possiamo ora visualizzare Nova sul nostro sito se tutto e' andato a buon fine:

```
localhost:8000/nova
```

- Cambiamo il *path* di *nova* da *config\nova.php*:

```
'path' => '/admin',
```

- Nel file***.env*** controlliamo che siamo in modalità *production*:

```
APP_ENV=production //local puo' entrare chiunque per testing
APP_URL=http://localhost:8000
```

- In *app\Providers\NovaServiceProvider.php* e' contenuta la *logica* del pannello.
Creiamo una funzione che accetti un utente solamente se e' un *Admin*:

```
protected function gate(){
	Gate::define('viewNova', function ($user){
		return $user->isAdmin();
	});
}
```

- Nel *app\Models**User.php*** aggiungiamo la funzione. Se e' *admin* restituisce *true*:

```
public function isAdmin(){
	return $this->role == 'admin';
}
```

NB: Aggiungiamo ***'role'*** tra i *$fillable.*

- Nella ***resource*** *user* in *app\Nova\User.php* specifichiamo nei ***fields*** che esiste un campo di tipo ***select*** chiamato *'role'.* Ora appare nella tabella con la lista degli utenti nel pannello:

```
use Laravel\Nova\Fields\Select;

public function fields(Request $request){
	return [
		...
		Select::make('Role'),
			->options([
				'user' => "Utente",
				'admin' => "Amministratore"
			])
			->displayUsingLabels()
			->sortable()
	];
}
```

displayUsingLabels . Mostriamo i campi in ***options*** con il *value* che abbiamo assegnato.
sortable . Rende ordinabile per Label la colonna
hideFromIndex . Nasconde la colonna dalla *index* ma la lascia visibile nel *detail*
hideFromDetail . Nasconde la colonna del *detail* ma la lascia visibile in *index*
exceptOnForm . Lo vedo nell'*index* e nel *detail* ma non nell'*edit form*

- Creiamo un *Model* Article ma non creiamo il *Controller* dedicato visto che la logica della costruzione di un Article avverrà attraverso la ***resource*** Nova e non da un *user* che visita il sito:

```
php artisan make:model Article -m
```

- Costruiamo la *Migration* dell'*Article* nel solito modo:

```
public function up(){
	Schema::create('articles', function *Blueprint $table){
		$table->id();
		$table->string('title');
		$table->string('img')->nullable(); //Anche se non usiamo questo metodo
		$table->text('description')->nullable();
		$table->boolean('published')->default(false);
		$table->timestamps();
	});
}
```

- Dopo aver dato forma alla *route* e alla *view* che conterrà gli articoli del nostro Blog, nel Tinker ne creiamo uno per controllare che tutto funzioni correttamente:

```
php artisan tinker
$a = App\Models\Article::create(['title' => 'Titolo di prova',
                                 'description' => '<p>Descrizione di prova</p>']);
```

- Dopo aver creato il *Model* Article, creiamo la ***Resource*** Article di ***Nova***, sempre nella *Bash:*

```
php artisan nova:resource Article
```

- La *resource* appena creata non ha alcuna indicazione di quello che contiene Article.
In *app\Nova\Article.php* la istruiamo:

```
use Laravel\Nova\Fields\Text;
use Laravel\Nova\Fields\Textarea;
...

public function fields(Request $request){
	public static $title = 'title';
	public static $search = [ 'id', 'title' ];

	return [
		ID::make(__('ID'), 'id')->sortable(),
		Text::make(__('Titolo'), 'title')
			->rules('required'),
		Textarea::make(__('Descrizione'), 'description')
			->alwaysShow(),
		Image::make(__('Immagine'), 'img')
			->disk('public')
		//	->path('/blog')
			->prunable()
		Boolean::make(__('Pubblicato'), 'published'),
		Date::make(__('Creato il'), 'created_at')
			->format('DD/MM/YYYY'),
		DateTime::make(__('Modificato il'), 'updated_at')
			->format('DD/MM/YYYY, HH:mm:ss'),
		// Computed field
		Text::make(__('Data'), function(){
			return optional($this->created_at)->format('d/m/Y H:m:s');
		}),
	];
}
```

$title . Quale campo vuoi che compaia nella ricerca quando cerchi un articolo.
$search . Cosa vuoi *indicizzare* nella ricerca.
fields . Sono i campi della *resource*.
NB: I *parametri* dopo il ***make*** sono il nome che la colonna avrà, e il nome della colonna nel *database*.
alwaysShow . Mostra le Textarea nel *detail* al completo, e non in parte
disk . Indica lo *storage* dove le immagini che carichiamo da Nova verranno salvate, eseguire prima *php artisan storage:link*
path . Indichiamo il *percorso* di dove andranno salvate le immagini.
prunable . Cancella l'immagine dallo *storage* se l'Article viene cancellato.
Date . Campo che permette di modificare la data nel *detail*.
DateTime . Campo che permette di modificare data e ora nel *detail*.
format . Modifica del formato
rules . Rende il dato *required* e restituisce errore *frontend* nel form se non compilato.
Computed Field . Campo di sola lettura che richiede una funzione anonima.