**| Vytvorenie Laravel projektu**

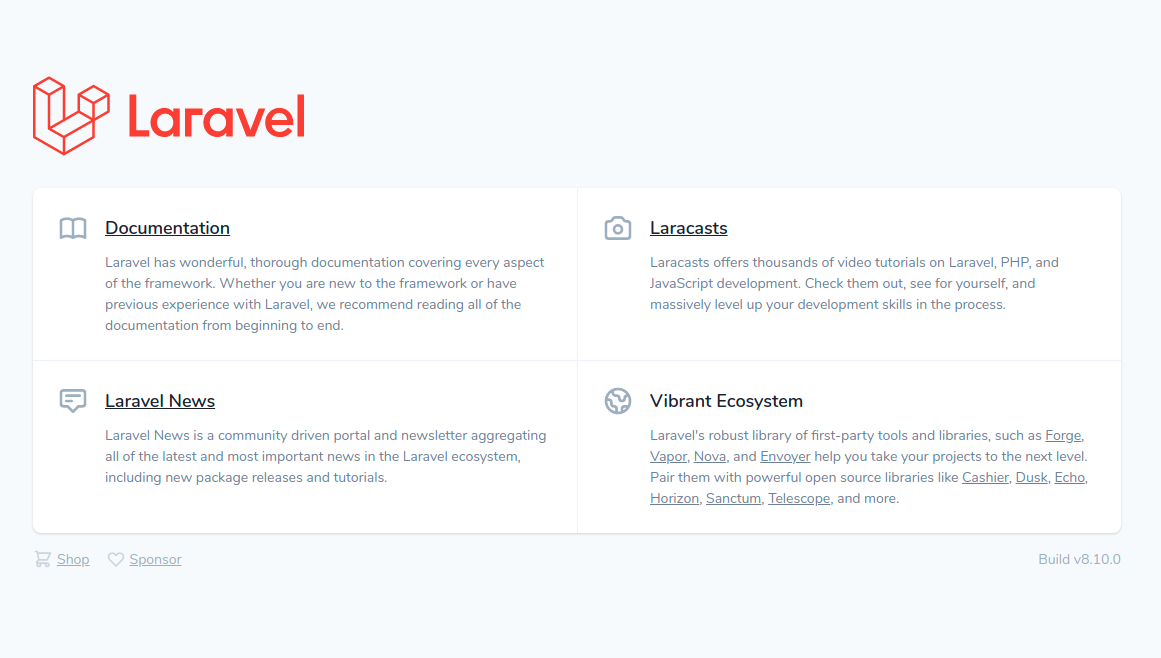
Pre projekt je potrebné vytvoriť novú zložku v **XAMP --> HTDOCS** **--> názov**, a následne pomocou príkazového riadka spustiť príkaz pre vytvorenie štruktúry aplikácie:

composer create-project laravel/laravel **názov**

**Laravel** projekt spustíme pomocou nasledujúceho príkazu v priečinku projektu vždy, keď budeme chcieť pracovať na webovej aplikácií:

php artisan serve

Po spustení projektu, sa webová aplikácia zobrazuje na URL adrese v rámci webového prehliadača <http://localhost:8000/>



**| Konfigurovanie projektu**

Pre pripojenie a prácu s databázou je potrebné skontrolovať konfiguráciu projektu v súbore s názvom **.env**. V tomto súbore je potrebné zmeniť premennú **DB\_DATABASE** na vlastnú hodnotu.

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

**DB\_DATABASE=laravel\_cms**

DB\_USERNAME=root

DB\_PASSWORD=password

...

**| Generovanie databázovej štruktúry**

Laravel je v základe postavený na [MVC architektúre](https://www.itnetwork.sk/php/mvc). Tvorba webovej aplikácie sa preto začína od modelu, pre vytvorenie dát pre pohľad a metódy, ktoré môže kontrolér potom ihneď využiť. Pre zobrazenie všetkých možností, ktoré umožňujú vygenerovať potrebné súbory, cez príkazový riadok/terminál v zložke s projektom je možné použiť nasledujúci príkaz:

php artisan make:model

Pre vygenerovanie všetkých potrebných súborov stačí do príkazového riadka vložiť nasledujúci príkaz:

php artisan make:model **názov** -a

*V prípade, že táto možnosť nie je k dispozícií, je potrebné všetky súbory vygenerovať zvlášť cez nasledujúce príkazy.*

php artisan make:model **názov**

php artisan make:controller **názov**

php artisan make:seeder **názov**

php artisan make:factory **názov**

**| Migrácia a model tabuľky**

Vygenerované migračné súbory obsahujú štandardne stĺpce ID, created\_at a updated\_at definujúce časy vytvorenia a posledné úpravy. Nachádzajú sa v **database/migrations** a je potrebné ich doplniť o vlastné hodnoty podľa potreby. Po migračných súboroch je potrebné sa sústrediť na modely, ktoré sa nachádzajú v priečinku **app/models.** V modeli je možné definovať polia vlastností prostredníctvom premennej $fillable, ktorá umožňuje odovzdávanie tzv. hromadným spôsobom.Taktiež sa v rámci modelov definujú Eloquent vzťahy. Každý vzťah musí byť definovaný metódou v modeli. Tá následne vnútorne volá metódu Eloquent triedy model, kedy je jej názov odvodený od typu vzťahu. Všetky vzťahy sú dostupné v rámci Laravel dokumentácie (*https://laravel.com/docs/8.x/eloquent-relationships*).

Pre vytvorenie migračnej databázovej tabuľky, ktorá si uchováva všetky spustené migrácie, je potrebné spustiť príkaz:

php artisan migrate

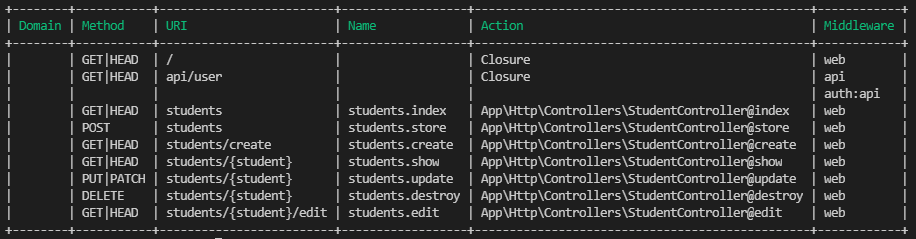
V prípade akýchkoľvek zmien, chýb atď. je potrebné pred opätovným zadaním príkazu migrate použiť príkaz:

php artisan db:wipe

**| Routovanie**

Pre využívanie CRUD operácií v rámci webovej aplikácie je potrebné mať pre každú operáciu spôsob akým ju zavolať. Metóda resource() v rámci vygenerovaných kontrolérov umožňuje definovanie ciest pre všetky dostupné operácie. Jedná sa o pridanie, zobrazenie, editáciu a odstránenie. Definujú sa v súbore, ktorý sa nachádza **routes/web.php.** Pre zobrazenie všetkých dostupných akcií je možné použiť píkaz:

php artisan route:list



Tabuľka ciest nám opisuje tieto vlastnosti:

* Domain - pre ktorú doménu je daná cesta určená. Ak je hodnota prázdna, platí východisková doména.
* Method - HTTP metóda akcie. Keďže HTML formuláre nepodporujú metódy PUT, PATCH ani DELETE, je potrebné ich trochu modifikovať.
* URI - URI akcie (v tomto prípade časť v URL adrese za doménou).
* Name - Názov, ktorý sa používa v kóde pre vytvorenie odkazu cez helper funkciu route() (napríklad route('students.index') vygeneruje http://localhost:8000/students).
* Action - Metóda kontroléru spracovávajúci danú akciu.
* Middleware - Výpis Middleware, cez ktoré požiadavka prejde.

### 