KIV/WEB

WEBOVÉ STRÁNKY KONFERENčNÍHO SYSTÉMU

Patrik Janoušek - A17B0231P

6. ledna 2019

Obsah

1 Instalace **3**

2 Použité technologie **3**

3 Adresářové struktura **4**

4 Architektura aplikace **5**
4.1 Průchod HTTP požadavku aplikací 5
4.2 Definice cesty v aplikaci 6
4.3 Šablony 7

5 Uživatelské rozhraní **7**

6 Závěr **8**

1 Instalace

Se zprovozněním aplikace by nikdo neměl mít žádný problém. Aplikace neklade na softwarovou výbavu systému žádné speciální požadavky.

K běhu aplikace je pořeba webový server (např. Apache) a PHP ve verzi 7.3. Pravděpodobně bude fungovat i PHP 7.1, nicméně podpora této verze nebyla otestována.

VirtualHost ve webovém serveru apache stačí nakonfigurovat obdobně podle následující konfigurace:

```
<VirtualHost *:80>
    ServerName kiv-web.localhost
    DocumentRoot "/var/www/kiv-web/public"

    <Directory "/var/www/kiv-web/public">
        Options +FollowSymLinks
            Require all granted
            RewriteEngine On

            RewriteCond %{REQUEST_FILENAME} !-d
            RewriteCond %{REQUEST_FILENAME} !-f
            RewriteRule ^ index.php [L]
    </Directory>
</VirtualHost>
```

Dále je potřeba vytvořit symbolický odkaz `public/storage` na `storage` pomocí příkazu `ln -s $PWD/storage $PWD/public/storage`. Aby příkaz fungoval správně, tak je nutné být v kořenovém adresáři projektu.

Poté stačí nakonfigurovat proměnné v souboru `.env`, naimportovat databázi do databázového systému MySQL (nebo některého z jeho forků). Ve výchozím stavu existuje v databázi administrátorský účet se jménem i heslem `admin`.

2 Použité technologie

Aplikace byla postavena na technologiích PHP 7.3, MySQL (konkrétně MariaDB), a dodávaná konfigurace je pro webový server Apache. Neměl by však být žádný problém tuto aplikaci spustit na nginxu nebo jiném webovém serveru.

Pro lepší správu knihoven byl použit balíčkovací nástroj composer, který se stará i o autoloading podle standardu PSR-4, kterého aplikace hojně využívá.

Jako šablonovací systém byl použit Twig, který je vyvíjen jako jedna z komponent Symfony frameworku. Tento systém se stará o sestavení výsledného HTML dokumentu z několika různých částí. Tento šablonovací systém je také další vrstvou ochrany před XSS útoky.

Pro sestavení JavaScript a CSS souborů je použit nástroj gulp. Tento nástroj se stará o kompilaci a minifikaci kaskádových stylů z SCSS do CSS, aby jim rozumněl prohlížeč. Stejně tak probíhá kompilace a minifikace JavaScript souborů, které jsou psány podle standardu ES6. Výstupem z tohoto nástroje je pak jeden JavaScript a jeden CSS soubor. Díky tomu je možné efektivně využít cache prohlížeče. A vzhledem k tomu, že tyto soubory nejsou nějak velké, tak to nijak negativně neovlivňuje ani první načtení webové aplikace.

Všechny frontend knihovny jsou pak importování přes balíčkovací systém npm, který obdobně jako composer umožňuje snadnou správu a používání externích knihoven a frameworků.

3 Adresářové struktura

- app - Hlavní logika aplikace
  - Http - Jedná se o soubory, které se nějak přímo podílí na zpracování HTTP požadavku.
    - Controllers - Složka s controllery aplikace
  - Models - Modelová vrstva aplikace
  - Validators - Validátory pro jednotlivé formuláře aplikace
- bootstrap - Skripty, které se starají o inicializační nastartování aplikace
- core - Jádro aplikace. Velmi zjednodušeně řečeno se zde nachází *micro-framework*, který umožňuje celou aplikaci vyvíjet pohodlněji.
- public - Veřejná část webu. Zde se také nachází `index.php` jako jediný PHP soubor. Díky tomu není možné spouštět z veřejné části webu jiné PHP soubory než právě zmíněný `index.php`, což přináší větší bezpečnost celé aplikace.
  - dist - Zkompilované CSS a JavaScript soubory
- resources - Složka se skripty, styly, šablonami a daty pro validátor formulářů
  - scripts - Zdrojové JavaScript soubory před kompilací

  - styles - SCSS soubory
  - templates - Šablony pro šablonovací systém Twig
  - validation - Data pro validační systém. Chybové hlášky a překlady názvů jednotlivých polí.

- routes - Jednotlivé cesty v aplikaci. Zde se nachází mapování URI adres na konkrétní metodu v konkrétním controlleru.
- storage - Úložiště aplikace
- .babelrc - Informace pro převedení JavaScriptu do srozumitelné podoby pro prohlížeč.
- .env - Proměnné prostředí. Slouží ke konfiguraci aplikace.
- .env.example - *Šablona* pro `.env`. Pokud soubor `.env` neexistuje, tak je nutné ho vytvořit jako kopii tohoto souboru. Tento soubor aplikace ignoruje.
- composer.json - Konfigurace aplikace pro balíčkovací systém composer.
- gulpfile.js - Postupy kompilace pro jednotlivé části aplikace pro nástroj gulp.
- package.json - Konfigurace pro balíčkovací systém npm.
- README.md - Návod pro instalaci aplikace

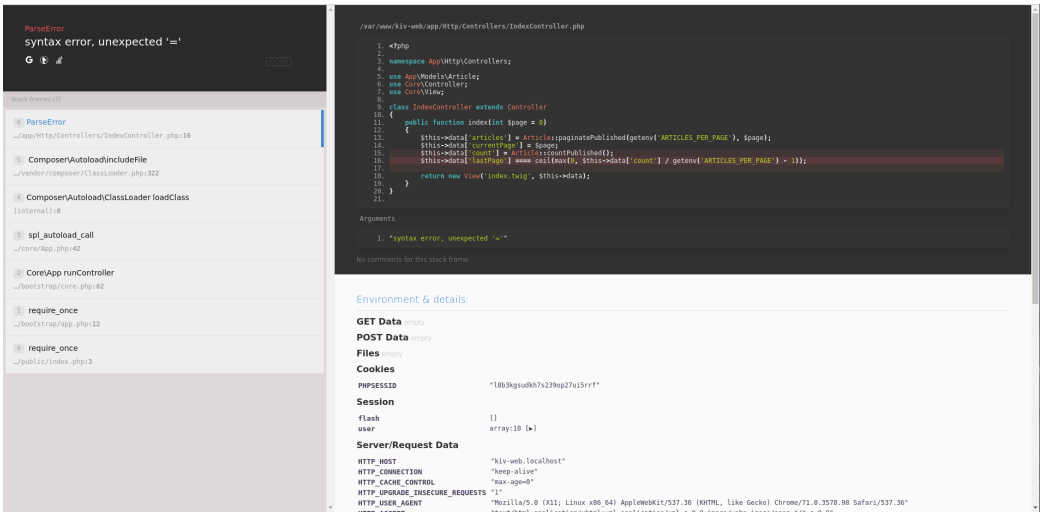
4 Architektura aplikace

Aplikace využívá obecně uznávaný návrhový model MVC. Díky tomu je oddělana aplikační logika, komunikace s databází a šablony pro šablonovací systém.

4.1 Průchod HTTP požadavku aplikací

Vše začíná u samotného souboru `index.php`, který obsahuje pouze nezbytně nutné množství kódu. Stará se tedy o načtení autoloaderu z composeru a také souboru `bootstrap/app.php`, který je v podstatě agregátorem souborů pro další průběh startu aplikace.

V souboru `bootstrap/app.php` se pak provádí definování *helper* funkcí, které zjednodušují používání aplikace a zkracují některé syntaktické zápisy,



Obrázek 1: Příklad chyby zobrazené Whoops frameworkem

jelikož jsou často obalem pro vytváření objektů. Dále se provádí načtení proměnného prostředí ze souboru `.env`, inicializace whoops frameworku, což je error handler framework, který se stará o hezčí a hlavně sdílnější chybové hlášky (viz obr. 1). Nakonec už je pak inicializováno samotné jádro aplikace, které se opět stará o inicializaci šablonovacího systému Twig, routovací knihovny Symfony Routing a vytvoření objektu aplikace, kde už začíná „skutečný“ život našeho HTTP požadavku. Pomocí symfony routingu se zjistí která metoda ve kterém controlleru se má zavolat, a také jeké parametry se jí mají předat. Z této metody je pak vrácen objekt `Response`, `View` nebo `Redirect`. Na základě tohoto vráceného objektu se pak jádro aplikace rozhodna, zda prohlížeči vrátí vykreslenou HTML šablonu, holý text, JSON nebo ho přesměruje na jinou stránku.

Co se týče přesměrování, tak má aplikace podporu pro takzvané flash messages, díky kterým je možná přeposlat nějaká data mezi stránkami. To se hodí právě v případě, kdy uživatele po odeslání formuláře přesměrujeme na jinou stránku a chceme mu zobrazit nějakou informaci o tom, zda se jeho operace zdařila.

4.2 Definice cesty v aplikaci

Cesty se v aplikaci definují v souboru `routes/web.yaml` ve formátu YAML, jak lze usoudit z přípony souboru. Díky tomuto je struktura velmi čistá a přehledná.

Následující příklad ukazuje jak lze definovat cestu v aplikaci

```
article:
    controller: App\Http\Controllers\ArticleController::showArticle
    path:
        cool:   /article/{id}
        ugly:   /?page=article&id={id}
```

Lze si povšimnout, že cesta má nadefinováné 2 cesty. Jsou jimi `cool` a `ugly`. Aplikace totiž podporuje oba styly URI a využívá se k tomu podpora Symfony Routingu pro lokalizování odkazů v aplikaci. Jedná se sice o menší zneužití této vlastnosti Symfony Routingu, ale je to jedním z nejjednodušších způsobů jak mít podporu pro oba styly URI. Přepnutí stylu URI se dá provést v `.env` souboru.

4.3 Šablony

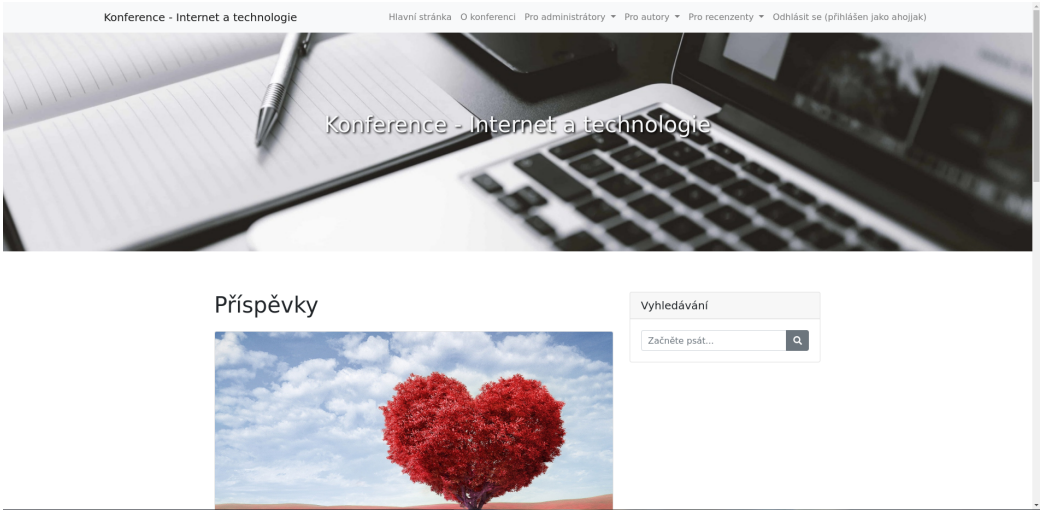
Jak již bylo zmíněno, tak šablony se nachází v adresářové struktuře nachází v `resources/templates`. Tyto šablony jsou patřičně dekomponovány tak, aby bylo možné je udržitelně skládat do komplexnějších a větších šablon.

Za zmínku stojí složka `base`, ve které se nachází základní šablony pro každou část webu. Díky tomu je možné, že administrátor bude mít mírně odlišný layout než třeba autor. Soubor `base.twig` je pak základním stavebním kamenem celé architektury šablon. Stanovuje obecný layout pro všechny šablony a také definuje některé klíčové proměnné pro celou aplikaci.

5 Uživatelské rozhraní

Celá webová aplikace používá velmi jednoduchý a čistý design, který je plně responzivní. Webovou aplikaci je tudíž možné používat jak z počítače, tak i tabletu nebo mobilního telefonu.

V horní části aplikace se nachází menu, které neustále zůstává na svojí pozici. A to i tehdy, pokud se uživatel pohybuje po stránce.



Obrázek 2: Úvodní stránka aplikace

6 Závěr

Aplikace podle mého názoru splňuje zadání v plném rozsahu. Ačkoliv jsem při vývoji nenarazil na větší překážky, tak byl velice zdlouhavý. Příčinou je pravděpodobně to, že jsem pro vypracování této semestrální práce nomohl použít PHP framework ani ORM. Díky tomu se vývoj stal mnohem pomalejším, jelikož jsem strávil spoustu hodin jenom vymýšlením rozumného jádra celé aplikace. Nicméně to určitě mělo smysl a posunulo mě to dál jako programátora. Kdybych použil například laravel, tak by to byl vzhledem k jednoduchosti aplikace jenom další web v řadě, který bych udělal za dva večery.