Webové Aplikace – Školní Webová Stránka

Dominik Rataj

# Anotace

Maturitní projekt jejímž zadáním byl školní web obsahuje postup pro vytvoření webových stránek se správou databáze. Umožňuje editace stránek, redakční systém, překlad, nahrávání souborů a kalendář akcí.

Cílem maturitní práce bylo vytvořit funkční školní web, který je podobný stávajícímu školnímu webu.

Prohlašuji, že dokumentaci maturitní práce jsem zpracoval samostatně podle svých znalostí, pokynů a rad vedoucího práce. Všechny použité zdroje informací a obrázků jsou uvedeny ve zdrojích.

V Orlové dne:

# Zadání

# Konfigurace

Konfigurace webové stránky je nutná při úvodním spuštění. Určuje nám přihlašovací údaje do databáze MySQL, heslo a jméno administrátorského účtu a nastavení jazyků lokalizace.

Konfigurační soubor se nachází ve složce **vendor/config.json** a je ve formátu JSON. Ve stejnojmenné složce je i soubor **config.template.json** ve které je předloha v případě vyššího řádu nastavení pro lokální vývoj.

## Jednotlivá Nastavení

|  |  |  |
| --- | --- | --- |
| Název | Výchozí Hodnota | Detail Nastavení |
| mode | release | Nastavuje PHP prostředí jestli se jedná o web v lokálním prostředí (hodnota „development“) nebo je web dostupný online (hodnota „release“) |
| sitename | nosch.cz | HTTP název webové stránky |
| databaseIP | localhost | IP adresa (nebo DNS název) MySQL databázového serveru |
| databasePort | 3306 | Číslo INET portu pro MySQL databázový server |
| databaseUser | root | Jméno uživatele pro přihlášení na MySQL server |
| databasePassword |  | Heslo uživatele pro přihlášení na MySQL server |
| databaseName | netoffice | Výchozí název databáze v MySQL serveru ve kterém budou data ukládány |
| attemptToCreateDatabase | false | V případě hodnoty „true“ se aplikace pokusí automaticky vytvořit databázi při prvním spuštění PHP skriptu. |
| forceDatabaseTableRecreation | false | **Nebezpečné.** V případě hodnoty „true“ se tabulky v databází znovu vytvoří i v případě, že již existují (dojde ke smazání předchozích dat) Používá se pouze v případě chyby v databází a v realném provozu je hodnota vždy „false“ |
| rootUsername | root | Přihlašovací jméno administrátorského účtu pro správu webu |
| rootPassword | root | Přihlašovací heslo administrátorského účtu pro správu webu |
| mapsURL | [LINK](https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d392.12179124407055!2d18.427985469931993!3d49.87663367639296!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x0%3A0xc1f2526c69235f38!2sSecondary%20School%20of%20NET%20Office%20Orlova!5e1!3m2!1sen!2scz!4v1581006568479!5m2!1sen!2scz) | Embed URL pro Google Maps v patičce stránky |
| defaultLanguage | 0 | Index základního jazyka užit při prvním načtení stránky |
| languages | czech,english | Názvy souborů v **vendor/usercfg/lang/XXX.yml** klasifikačních lokalizací. |

# Software

Software nutný pro funkci webu

|  |  |  |
| --- | --- | --- |
| Název | Link | Detail |
| Apache | <https://httpd.apache.org/> | Spouštěč PHP skriptů a sloužení dokumentů. Lze použít i jakýkoliv další PHP/HTML software s podporou PHP 7, webová stránka ale byla napsána speciálně pro software Apache |
| MySQL | <https://www.mysql.com/> | Databázový server pro ukládání dat |

Knihovny a nástroje použité ve vývoji

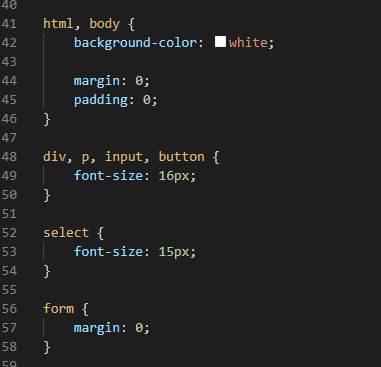
|  |  |  |
| --- | --- | --- |
| Název | Link | Detail |
| PHP 7+ | <https://www.php.net/> | PHP 7+ s podporou statických typů a mysqli |
| Twig, Twig Utils | <https://twig.symfony.com/> | Návrhy HTML dokumentů v PHP skriptu |
| Nodejs | <https://nodejs.org> | NodeJS 12+ s podporou npm |
| Webpack | <https://webpack.js.org/> | Knihovna NodeJS použitá pro support JavaScriptu pomocí TypeScript a minifikace výsledných JS a CSS kodů |
| TypeScript | <https://www.typescriptlang.org/> | Microsoft Superset pro JavaScript s podporou statických datových typů |
| JQuery | <https://jquery.com/> | JavaScript knihovna pro zjednodušené ovládání DOM pomocí JS |
| SASS | <https://sass-lang.com/> | CSS superset s vylepšeným syntaxem a podporou proměnných v kompilačním čase |

# Grafické rozvržení stránky

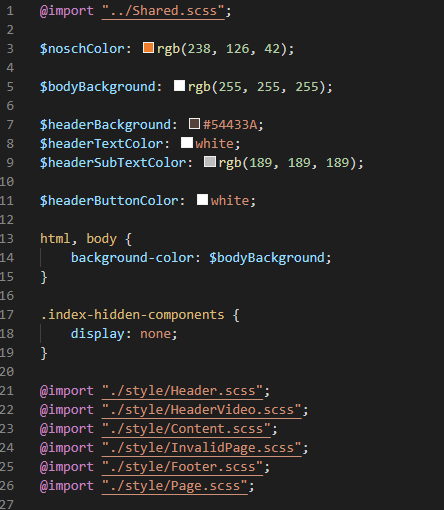
Grafický vzhled stránky je inspirován modernějšími weby při dodržování standardu pro školní weby. Web je grafický spíše světlý a grafický design je více zaměřen na přehlednost. Styl je programován v jazyce SASS, který se kompiluje do CSS stylu. Výhoda SASS oproti CSS jsou proměnné (viz. Ukázka). Pomocí NodeJS společně s Webpack jsou pak všechny SASS styly zkompilovány do jednoho zdrojového CSS souboru, který je individuální pro každou stránku (*bundle.css* pro **index** a *admin.css* pro **admin.php**)

Při lokálním vývoji je styl pro rychlý vývoj připojen pomocí JavaScriptu. Při finální produkci stránky je styl připojen klasickým *linkem*.

Ukázka SASS kódu, který je sdílen pro všechny stránky (**webpack/src/Shared.scss**)



Ukázka SASS kódu pro hlavní stránku (**webpack/src/index/Index.scss**) s využitím proměnných.

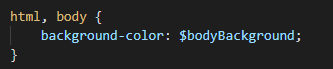


Lze vidět i použití direktivu *@import* který nám pomáhá rozdělit stylový kód do více souborů aby byla jednodušší správa stylového programování.

Využití proměnných lze vidět hned na začátku, kde se proměnné nastavují pomocí dolaru ($) a následné dvojtečky:



Použití proměnné přímo v kódu je jednoduchý opis názvu proměnné s prefixem



## Styl zápisu stylového kódu

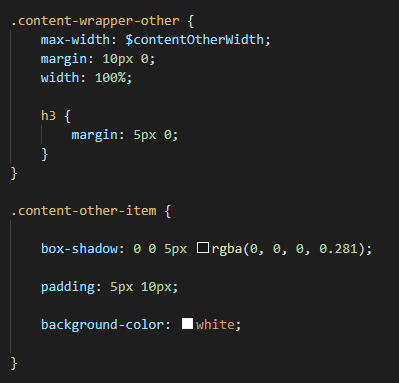
Proměnné, třídy a komentáře jsou prioritně psány v anglickém jazyce. Anglický jazyk je použit v celém projektu k jednoduššímu navigování a generalizace kódu. Komentáře a dokumentace je taktéž napsána v anglickém jazyce.

Stylisticky jsou mezery vždy odděleny pomlčkou, například:



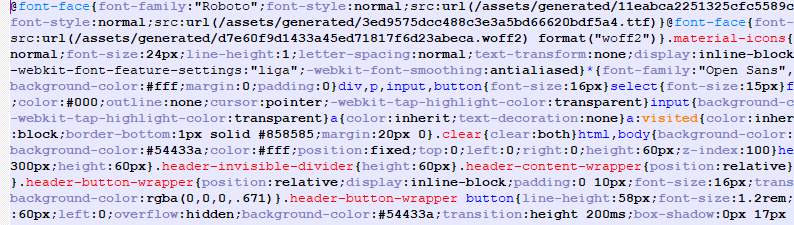
V příkladu je možné vidět i umístění složených závorek, které jsou vždy na konci názvu třídy nebo klíčového slova a poté na prázdný řádek při ukončení. Jednotlivé CSS atributy mají každá svůj dedikovaný řádek oddělen od startu řádku tabulátorem o velikosti 4.

Každé další sekce (třídy, ID) jsou vždy od sebe odděleny jedním řádkem.



SASS má výhodu takzvaných vložených klíčových slov. V příkladu je vidět *content-wrapper-other* obsahující *h3.* SASS při kompilaci jednotlivé značky oddělí do validního CSS stylu.

## Výsledek grafického kódu



Výsledný kód je minifikován a zároveň co nejvíc zjednodušen kompilátorem jazyku SASS. Z optimalizačních důvodů jsou i obrázky a ostatní soubory, které jsou přímo vloženy do webu (jako například obrázky a písma) minifikovány a názvy jsou změněny. Speciální atributy jsou pomoci Webpack SASS modulu také prefixovány (*-webkit, -moz*) pro co největší kompatibilitou mezi platformami.

# Funkčnost webu pomocí PHP

Při designování backend systému webu jsem se rozhodl použít PHP místo mé preferované volby NodeJS + express. Jelikož je PHP dostupné všude a je méně náročné na správu než NodeJS. Oproti TypeScriptu (NodeJS) nebo Javě má PHP velký problém ve svém jazyce. Designové prvky dynamických typů a malé podporu nápověď mě vedly k rozhodnutí použití PHP 7, kde je podpora statických typů mnohem lepší. Ačkoliv je PHP 7 nejnovější verze, neměl by být žádný problém sehnat potřebný software. Apache a nginx již dávno adaptovali novou PHP 7. Nová verze PHP je také více stabilní a rychlejší než předchozí verze 5, která se používala nejdéle.

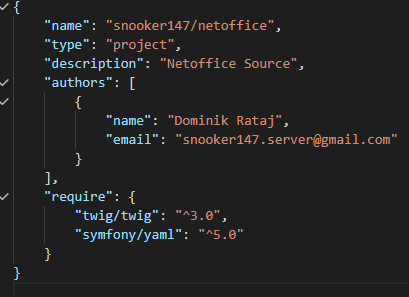
## Stylistická pravidla kódu PHP

V projektu je zásadním způsobem dodržován styl PHP kódu. Zpravidla se uplatňuje objektivita, a to i ve statických funkcích, a to z důvodu, aby nedošlo ke jmenným konfliktům. Záměrně ale kód nevyužívá PHP jmenné prostory a to kvůli tomu, že se jedná o projekt lokální, který nebude sloužit jako knihovna. Nemá tedy smysl jmenné prostory používat. Toto rozhodnutí je inspirováno známým stylem kódu praktikován např. ve společnosti Valve a jejich [Source Engine](https://github.com/ValveSoftware/source-sdk-2013).

Styl kódu téměř kopíruje styl CSS. Všechny třídy musí začínat velkým písmenem (jediná výjimka). Jakýkoliv další kód používá styl [Camel Case](https://en.wikipedia.org/wiki/Camel_case), který jsem odchytil od stylu programovacího jazyku Java. Výjimkou nejsou ani statické funkce, které musí navzdory konfliktu se standartním PHP stylem (snake case) používat styl Camel Case.

## Lokalita kódu a knihovny třetích stran

Vlastnoručně napsán PHP kód je umístěn ve složce **vendor/src.** Pro knihovny třetích stran jsem použil nástroj [composer](https://getcomposer.org). Jedná se o nástroj, který zjednodušuje stažení a správu nainstalovaných knihoven. Klíčovým souborem pro composer je *composer.json* umístěný ve složce **vendor.**

****

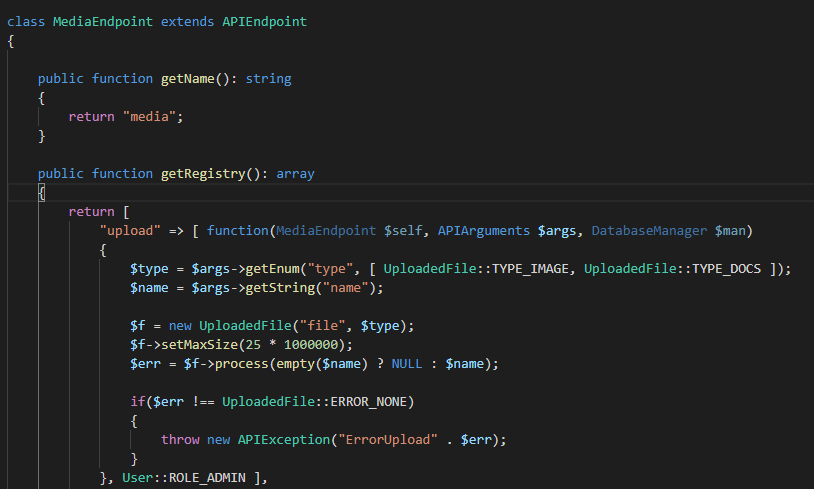
*Celá ukázka kontrolního souboru composeru*

Lze vidět, jak jsem se již zmiňoval, že používám Twig (twig/twig) a Twig Utils (symfony/yaml).

## Propojení Frontendu a Backendu

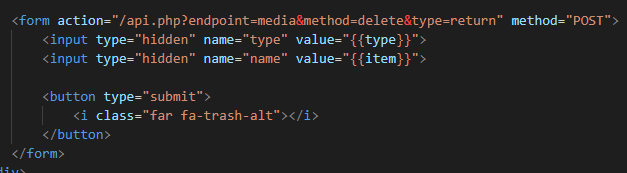
Na rozdíl od klasického stylu programování webových stránek pomocí PHP jsem se rozhodl použít PHP co nejvíce odděleně co se týče backend a frontendu. Pokud bychom například chtěli udělat v budoucnu mobilní aplikaci museli bychom použít nějaké API pomocí kterého propojíme mobilní aplikaci s naším PHP backendem. Toto již je možné v tomto projektu. Jakékoliv funkce s databází, přihlášení, načítání apod. jsou definována v API tohoto projektu. Propojení Backendu s Frontendem je nejvíce vidět ve složce **vendor/src/api** která obsahuje právě tuto komunikaci.

Nejjednodušší případ na popsání je například *MediaEndpoint*. Všechny koncové komunikační kanály pro API jsou nazývané *Endpoint* (koncový bod). *MediaEndpoint* se nám stará o nahrání medií jako třeba dokumenty nebo obrázky.



Vidíme, že *MediaEndpoint* dědí z třídy *APIEndpoint.* Zde je vidět styl kódu, který jsem popisoval předtím. Každá funkce webu má svoje třídy a vlastnosti. K tomu se nejlépe hodí objektivně orientované programování, které je v PHP dostupné. V metodě *getRegistry* pak vidíme kompletní implementace jednotlivých koncových bodů. V ukázce je možné vidět *upload* tedy nahrání souboru.

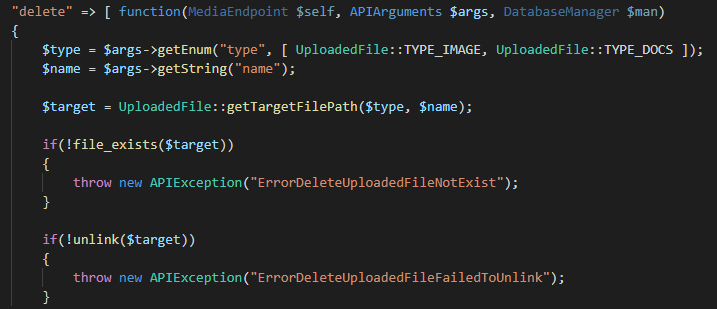
Každý koncový bod se identifikuje pomocí *metody* a názvu koncového bodu. Ty jsou obsaženy v **GET** parametrech.



V ukázce Twig kódu lze vidět formulář pro odeslání na API koncového bodu *media* a metody *delete*. Jedná se tedy o operaci, která vymaže položku médii. Vidíte, že URL destinace formuláře je:

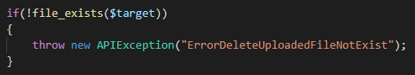
**api.php?endpoint=media&method=delete**

Zeleně jsou vyznačeny kontrolní parametry *endpoint* a *method* ty nám ukazují o které API se konkrétně jedná. Všechny ostatní parametry jsou poté připojeny pomocí typu **POST** jakým je formulář nastaven. Lze například vidět dva parametry (input) **type** a **name**.



Když se podíváme na implementaci API *delete* hned na prvních řádcích vidíme, že vyžadujeme tyto dva parametry a následně je ukládáme do PHP proměnných. Ke kolekci parametrů ze superglobálního pole **$\_GET** a **$\_POST** využíváme třídu *APIArguments* která je univerzálně napsána a je možné ji využít s jakýmkoliv typem pole. *APIArguments* automaticky vyhodí výjimku při špatném parametru.

Výjimky vyhozené v implementaci API jsou následně konvertovány do chyb, které jsou lokalizované. Lze to například vidět při zjišťování zdali soubor existuje či nikoliv.

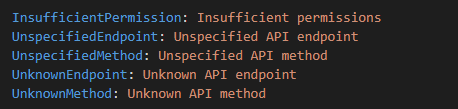


## Lokalizace

Abych zajistil co největší flexibilitu webové aplikace, rozhodl jsem se, že vytvořím vlastní systém lokalizace. Celou implementace lze vidět v souboru **vendor/src/lang/Lang.php.**



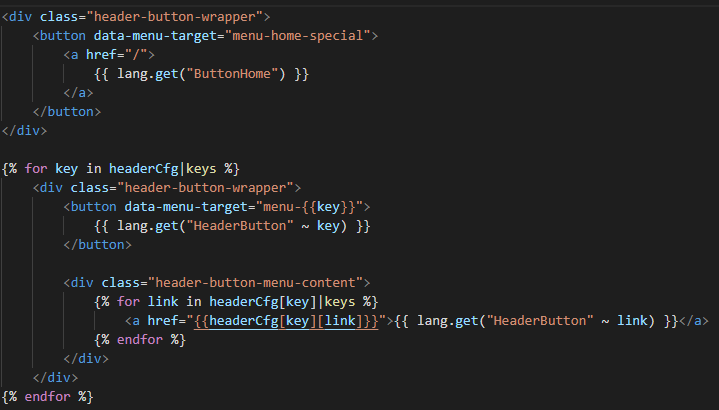
V ukázce lze vidět implementaci vrácení lokalizovaného klíče. Každý lokalizační klíč má svojí hodnotu uloženou v asocialním poli **$registry**, které je načteno s konfiguračních souborů ze složky **vendor/usercfg/lang**.



Ukázka ilustruje jak je lokalizace jednoduchá k nastavení. Mohou být definované jakékoliv další jazyky. Ukázka ilustruje jazyk *english.yml*

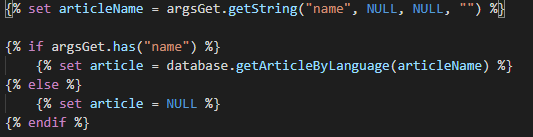
## Prezentační HTML kód stránky

Pomocí knihovny třetí strany Twig Renderer. Tato knihovna nám umožnuje vkládat PHP kód do HTML kódu bez nutnosti zdlouhavého použití funkcí *echo* a jíné.



V ukázce jde vidět lokalizace v tlačítku na domovskou stránku. Pomocí dvou složených závorek indikujeme Twig Renderu, že chceme vnořit proměnnou z PHP. Například lze použít lokalizace, jehož instance třídy je uložená v proměnné **$lang** a poté užití funkce **get** (popsána výše) k vrácení lokalizačního řetězce pomocí klíče (v tomto případe *ButtonHome*)

Níže lze naopak vidět užití cyklu **for**, který je velmi podobný cyklu v PHP. Cyklus **for** ve Twig Rendereru opakuje HTML kód pro každou hodnotu obsaženou v poli *headerCfg*.

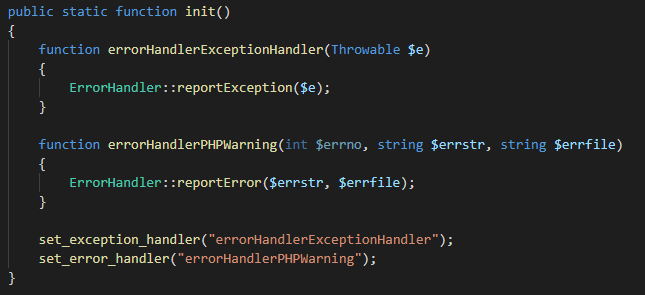


Twig umožňuje i jednoduché podmínky jakoby vývojář psal logický kód přímo v PHP. Obrázek výše ukazuje jak se jednoduše dají ve Twig dělat podmínky a příkazy pomocí složené závorky a procenta. Tento styl psaní mě osobně velice připomněl *ReactJS* ve kterém jsem rád pracoval.

## Hlášení a vypsání chybových hlášek PHP

Na rozdíl od ostatních PHP aplikací jsem se rozhodl použít vlastní výpis chybových hlášek PHP skriptů. To je kvůli tomu, aby se chyby v produkci dali jednoduše najít a opravit jelikož jakákoliv chyba je poté vypsána do textového souboru ve formátu **JSON** a jsou uloženy ve **vendor/errors**.

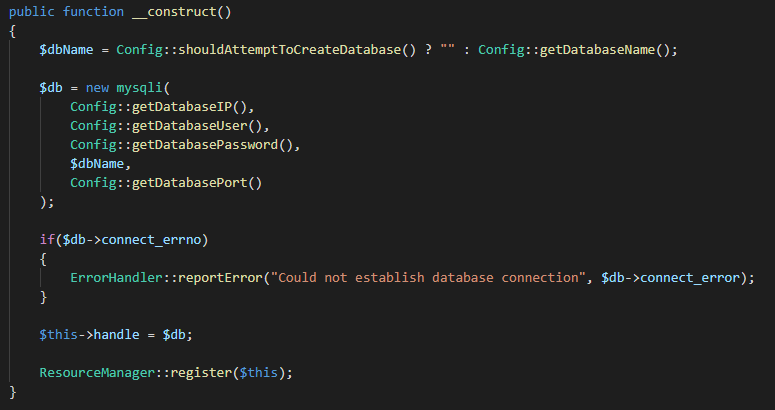
Logika zajištění chybových hlášek a chyb z PHP kódu v kódu **vendor/src/ErrorHandler.php**, který je vždy prvním souborem vložen do skriptu.



Lze vidět použití dvou metod z PHP *set\_exception\_handler* a *set\_error\_handler*, které se zavolají při nezajištěné výjimce nebo chybu v PHP. Tímto způsobem je možné chyby zachytit na velké úrovni nebo například kdykoliv zajišťovat vyhození neočekávaných chyb. V klasickém PHP chybovém emitoru je také možné vidět názvy souborů apod. Tyto informace by uživatel nikdy neměl vědět, můj vlastní systém definován v *ErrorHandler* zobrazí pouze hlášku a ostatní informace bezpečně uloží do souboru.

# Databáze

Databázový systém je založen na jazyku SQL. Projekt byl navržen na databázový software MySQL ale měl by být kompatibilní i s jinými databázovými systémy podporující SQL s výjimkou Microsoft SQL serveru. V kódu kvůli zjednodušení nelze nalézt přímý zápis příkazu SQL. Všechny operace s Databází jsou definovány a implementovány v třídě *Database* a *SimpleDatabaseManager* ve složce **vendor/src/database**. Třída *Database* se stará o připojení a konektivitě databáze podle parametrů v konfiguraci projektu.



K připojení a správě databáze používám modul *mysqli*, který je dostupný téměř v jakékoliv PHP platformě. Lze vidět přístup statických metod třídy *Config*, která se stará o vrácení hodnot z konfiguračního souboru. Nedojde tedy nikdy k tomu, aby bylo např. heslo nebo uživatel přímo napsán v kódu, tím pádem je jednodušší i nastavování služby pro administrátory.

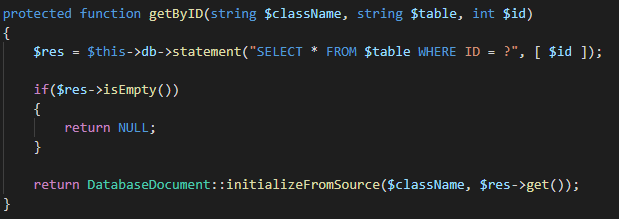
V ukázce je možno vidět i použití předešlého PHP kódu v *ErrorHandler*. Všechny moduly jsou mezi sebou opravdu velice propojené.

## Zabezpečení Databáze

Abstraktní třída *SimpleDatabaseManager* se stará o důkladnější sestavování a zjednodušení práce s databází. Nejdůležitější metodou je *statement*, který nám vytvoří parametrický dotaz. Parametrické dotazy jsou nejbezpečnější pro web, jelikož neumožní útočníkům vkládat vlastní SQL dotazy. Parametrický dotaz vypadá takto:



Na konci dotazu lze vidět otazník. Ten nám určuje, že na jeho místo přijde parametr, *mysqli* poté parametry vhodí do ovladače spojení MySQL a nemůže tak dojít ke [SQL Injection](https://www.w3schools.com/sql/sql_injection.asp).



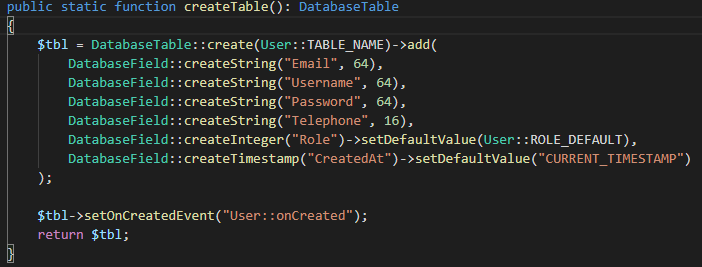
Implementace metody *getByID* v třídě *SimpleDatabaseManager*. Tato metoda je jednoduchá možnost jak získat dokument z tabulky pomocí ID. Sestavení dotazu je pak parametricky vidět na třetím řádku:

**SELECT \* FROM $table WHERE ID = ?**

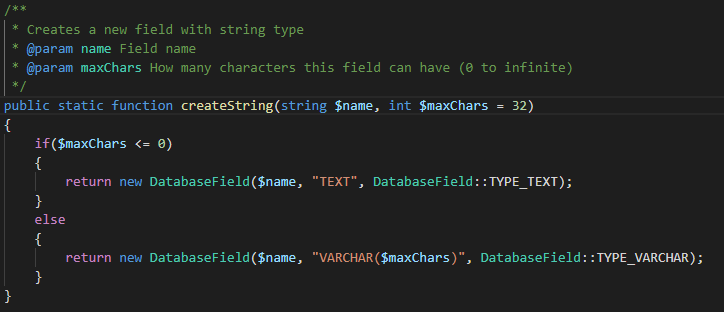
Jako druhý argument funkce je poté pole s parametry, je jasně vidět, že místo otazníku vejde ID záznamu, který chceme vrátit. Při parametrických dotazů se nemusíme ani zaobírat uvozovkami apod. ovladač *mysqli* je nám automaticky přidá podle datového typu. Implementace je specifikována v *SimpleDatabaseManager::statement*.

Dokumenty databázové struktury jsou poté definovány v kódu PHP jako statická funkce třídy implementující třídu *DatabaseDocument*.

## Navržení tabulek pomocí PHP kódu



Ukázka ilustruje, jak se vytváří struktura tabulky pro uživatele. Jako první se vytvoří instance třídy *DatabaseTable*, která definuje atributy tabulky. Poté pomocí metody *add* pouze přidáme všechny atributy pomocí statické metody *DatabaseField::createXXX*. První máme například *createString*. První argument metody *create* je vždy název atributy. Zde se dodržuje stylový zápis [Pascal Case](https://techterms.com/definition/pascalcase). Jako druhý parametr je velikost řetězce protože se jedná o datový typ **VARCHAR**. V případě, že je druhý parametr **0** nebude se jednat o **VARCHAR** ale **TEXT**.

**

Výše je vidět implantace metody *createString* užitá v předchozím odstavci. Můžeme jasně vidět záměnu datového typu v závislosti na velikosti řetězce. Je možné zde vidět i příklad dokumentace přímo v kódu. Ostatní metody pro vytváření atribut tabulky (*createInteger* apod.) jsou definovány podobně jako ukázka. Klíčová je instance třídy *DatabaseField*, která mimo jiné může nabývat i základních (defaultních) hodnot pomocí funkce *setDefaultValue*, může také kontrolovat či může atribut nabývat hodnot **NULL** apod.

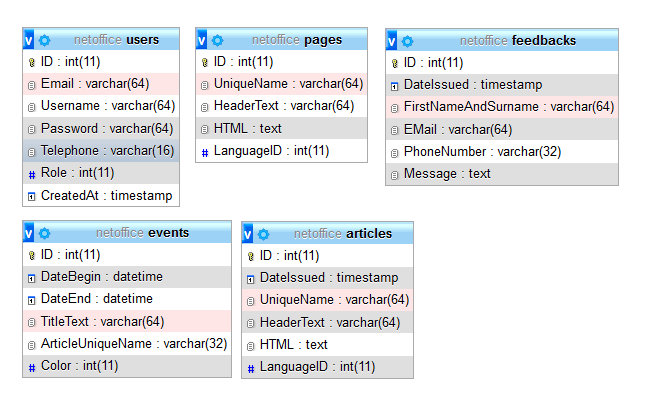
V tomto projektu sice nevyužito, ale je možné nastavit atributem, zdali se jedná o primární nebo cizí klič a na jakou tabulku referuje.

## Struktura Databáze

Pro strukturu databáze jsem se rozhodl použít co nejjednodušší systém, a to z toho důvodu, aby i psaní kódu PHP či HTML bylo co nejjednodušší a nezávazné. Lze to vidět na implementaci událostí v kalendáři a jejich propojení pomocí názvu článku místo toho aby byli propojené pomocí cizího klíče. Jedná se tedy o styl databáze z jednoduchých vztahů a databáze je použitá čistě pro uložení dat. Je zde také nutno poznamenat, že kvůli lokalizaci veškerého obsahu v aplikaci by bylo řešení databázové struktury mnohem složitější a nebylo by vůbec intuitivní.

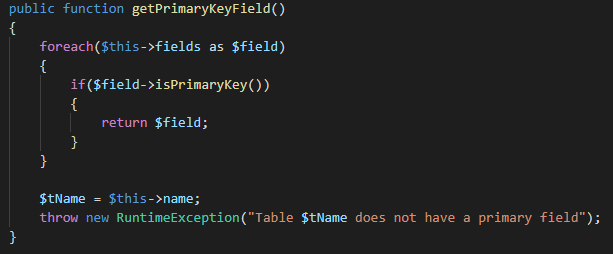
Databáze v podstatě obsahuje pouze pět tabulek. Z nichž pouze čtyři jsou zobrazovány administrátorovi stránky. Jsou to následující tabulky

|  |  |
| --- | --- |
| **Název** | **Detail** |
| Articles | Obsahuje informace o článcích. |
| Events | Událost v kalendáři |
| Feedbacks | Všechny zpětné vazby uživatelů |
| Pages | Obsahuje všechny stránky vytvořené administrátorem |
| Users | Obsahuje uživatele webu. V současné implementaci pouze jeden administrátorský účet. |

****

Všechny tabulky musí obsahovat primární klíč s názvem **ID** pro jednodušší identifikaci v PHP kódu.

Díky takovému sestavování Databáze v kódu PHP je také možné jednoduše zjistit například primární klíč (viz. Ukázka níže) nebo například jednoduše vytvořit instance objektu jednotlivých záznámů.



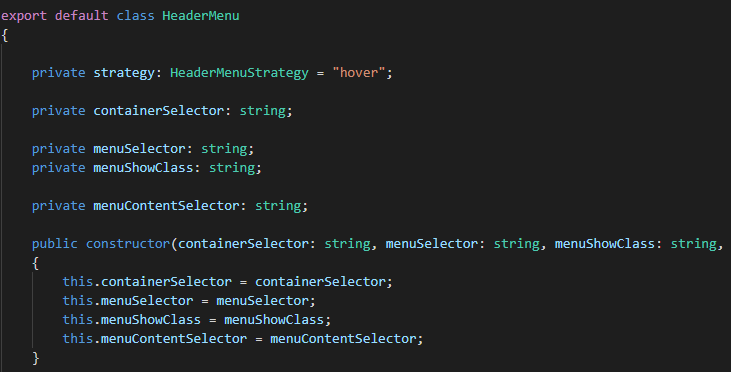
# JavaScript

V projektu se uplatňuje velké množství použití JavaScriptu. Nejvíce je používán v administrátorské části webu, tak aby prostředí pro klasické uživatele bylo co nejjednodušší a kompatibilní s nejvíce prohlížeči. Hojně se v administrátorském prostředí používá *Ajax* pro volání API metod (výše v sekci Propojení Frontendu a Backendu). Celý JavaScript kód je rozdělen do několika souborů pro přehlednost a využívá *NodeJS* společně s knihovnou *webpack* pro správu a kompilaci JavaScript kódu.

## TypeScript

Pro psaní kódu jsem zvolil superset JavaScriptu vyvíjen Microsoftem **TypeScript**. TypeScript je v podstatě JavaScript s datovými typy, což je hlavní důvod proč jsem se rozhodl jej použít. Jakýkoliv kód napsán v JavaScriptu je automaticky validním TypeScript kódem, takže jakékoliv knihovny nebo starší kód samozřejmě bude fungovat v TypeScriptu. Na rozdíl od JS jsou TypeScript soubory označovány koncovkou **ts**.

Zdrojový kód TypeScriptu stránkyspolečně se stylem je dostupný ve složce **webpack/src**.



Ukázkový kód TypeScriptu ukazuje, že je jednoduché například i definování vlastní třídy. V případě, že bychom nechtěli použít ES6 styl třídy (který není dostupný ve všech prohlížečích) můžeme v nastavení TypeScript (soubor **webpack/tsconfig.json**) nastavit aby výsledný JavaScript kód měl nižší verzi JavaScriptu. Toto je neskutečně užitečné, jelikož se nemusíme starat, jestli je daný styl kódování podporován všude, TypeScript to udělá za nás.

V deklaraci třídy můžeme vidět i nastavení datového typu jednotlivých proměnných. Například:



Vidíme, že proměnná je privátní a její datový typ musí být **string**. Takové striktní nastavování typů v klasickém JavaScriptu nenalezneme a pro programování složitějších aplikací je taková schopnost velmi užitečná, jelikož mnoho chyb nám objeví sám kompilátor a v případě neshody datových typů nebo jiných chyb vyhodí chybu ještě předtím, než se kód užije na stránce.

## Webpack

Webpack nám umožňuje jednodušší správu JavaScript a CSS kódu. V případě JavaScriptu nám také umožňuje importovat soubory podobně jako v PHP příkaz *include* nebo *require*. V případě webpacku se spíše jedná o příkaz *require*.

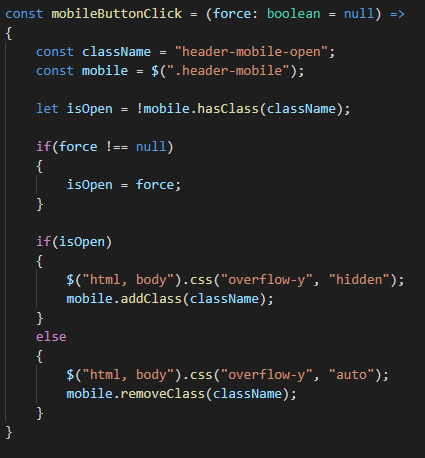


Zde například vidíme i import knihovny *JQuery* pomocí syntaxe ***import as***, ten nám umožnuje přejmenovat i výslednou proměnnou, která je poté v kontextu JavaScriptu využita. Zde jsem použil standartní dolar ($) pro import *JQuery* ale mohl jsem si to klidně pojmenovat jakkoliv bych chtěl.

Webpack se nám také může starat o soubory obrázkové nebo jakékoliv další média, to lze vidět ve složce **assets/generated**, o kterou se právě stará webpack a automaticky vygeneruje soubory podle toho, jak jsou užity v kódu.

## Užití JavaScriptu v projektu

Nejvíce je JavaScript použit na dynamické části webu jako například vysouvací nabídka v hlavním menu nebo nastavení hlavního menu podle velikosti okna.



Ukázka ukazuje implementaci metody při kliku na tlačítko v mobilním režimu. Lze vidět využití klíčových slov *const* a *let* které nahrazují použití klíčového slova *var*.

# Uživatelské prostředí

Uživatelské prostředí (UI) je dostupné ve dvojím provedení. To je standartní uživatelské pro návštěvníky webu a administrátorské.

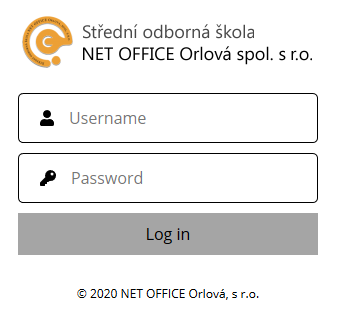
## Administrátorské prostředí

Předtím, než se uživatel může dostat do administrátorského prostředí, musí se přihlásit pomocí administrátorského účtu definovaného v konfiguraci.



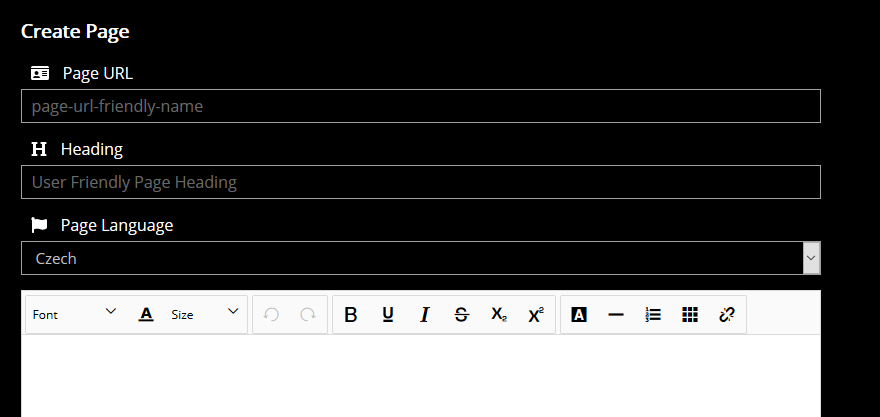
Kód přihlašovacího formuláře stránky pomocí Twig. Formulář odkazuje na API metodu popsanou v dřívějších sekcích o propojení Backendu s Frontendem.

Výsledný vzhled a struktura formuláře



### Přidání stránky

Administrátor může přidat stránku pomocí administrátorského režimu a následné otevření sekce *Pages*.



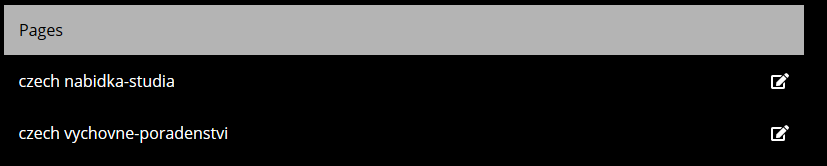
První je **URL** adresa nové stránky. Ta musí být unikátní a v souladu s pravidly **URL** odkazů. Podle nápovědy lze vidět, že by mezery měli být zaměněny za pomlčky a celý zápis by měl být uveden v malém písmenu.

Druhé je nastavení nadpisu stránky. To už by mělo být více pohledné po uživatele. Je zobrazeno jako titulek stránky a jako první řádek zobrazen na obrazovce.

Třetí je nastavení jazyka dané stránky. Jelikož je veškerý obsah v aplikaci lokalizován, i stránky vytvořené administrátorem je možno překládat. Při vytváření překladu stránky je pouze nutno uvést stejný název **URL** a poté zadat o jaký jazyk se jedná. Překládání stránek administrátorem není nutné, ale je užitečné pro uživatele z jiných regionů, v případě, že zobrazená stránka není v uživatelském nastaveném jazyce, stránka je poté zobrazena v jazyce základním podle konfigurace.

Čtvrtý je už samotný obsah dané stránky. Je zde vidět *wysiwyg* editor pro vzhled stránky.

Pod editorem je možno vidět list všech doposud přidaných stránek. Kliknutím na editovací tlačítko jej můžeme editovat. Jakoukoliv stránku je tedy možno kdykoliv změnit.

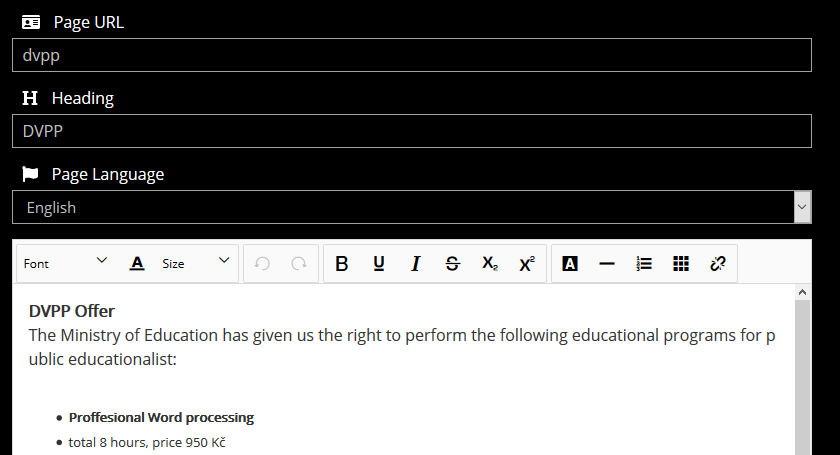


*První je vždy jazyk dané stránky, poté její* ***URL*** *adresa.*

#### Ilustrační vklad stránky



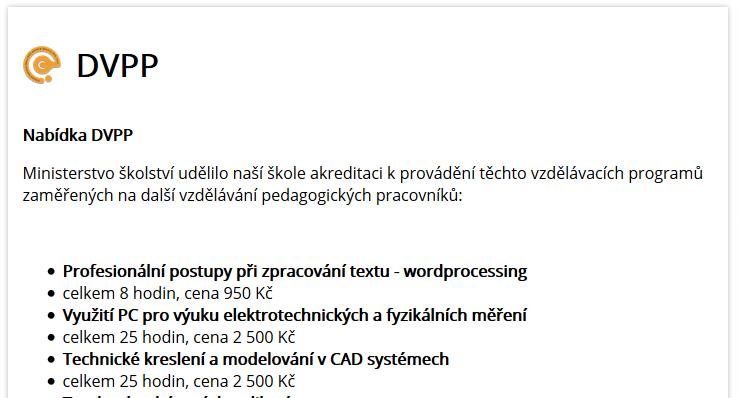
*Český základní vzhled.*



*Anglický překlad pro vytvořenou stránku.*

Vidíme, že jazyk stránky v kolonce *PageLanguage* se změnil na *English* ale **URL** zůstalo stejné. Právě díky **URL** aplikace ví, jakou stránku v jakém jazyce má zobrazit.

#### Ilustrační vzhled vložené stránky

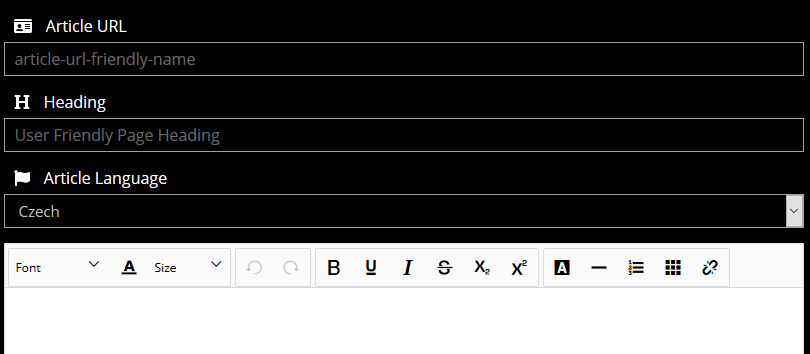


*Vzhled vložené stránky na* ***URL****: http(s)://domena/dvpp*

Vidíme, že vzhled se opravdu zachoval podle toho jak si administrátor přál.

### Přidání nového článku

Podobně jako stránky se články (též novinky) vytvářejí v administrátorském prostředí pomocí vyplnění kolonek.



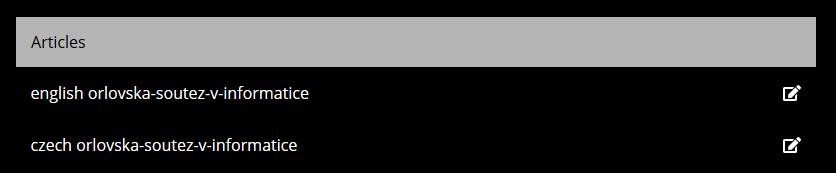
První je **URL** adresa nové novinky. Ta musí být unikátní a v souladu s pravidly **URL** odkazů. Podle nápovědy lze vidět, že by mezery měli být zaměněny za pomlčky a celý zápis by měl být uveden v malém písmenu.

Druhé je nastavení nadpisu novinky. To už by mělo být více pohledné po uživatele. Je zobrazeno jako titulek článku a jako první řádek zobrazen na obrazovce.

Třetí je nastavení jazyka dané novinky. Jelikož je veškerý obsah v aplikaci lokalizován, i novinky vytvořené administrátorem je možno překládat. Při vytváření překladu novinky je pouze nutno uvést stejný název **URL** a poté zadat o jaký jazyk se jedná. Překládání novinek administrátorem není nutné, ale je užitečné pro uživatele z jiných regionů, v případě, že zobrazená novinka není v uživatelském nastaveném jazyce, novinka je poté zobrazena v jazyce základním podle konfigurace.

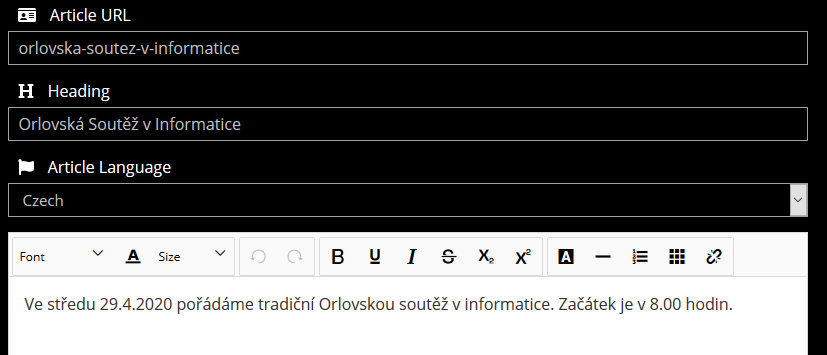
Čtvrtý je už samotný obsah daného článku. Je zde vidět *wysiwyg* editor pro vzhled stránky.

Pod editorem je možno vidět list všech doposud přidaných novinek. Kliknutím na editovací tlačítko jej můžeme editovat. Jakoukoliv novinku je tedy možno kdykoliv změnit.

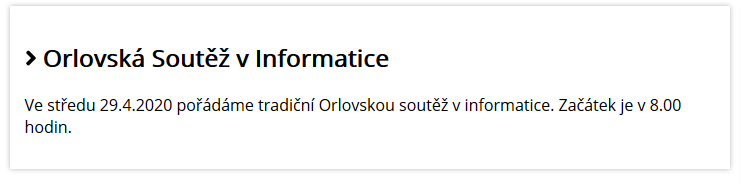


*Stejně jako u stránek. První je vždy jazyk novinky a poté* ***URL*** *novinky.*

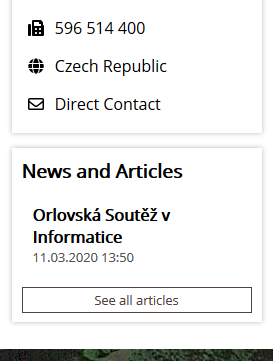
#### Ilustrační vklad novinky



#### Ilustrační vzhled vložené novinky



Novinky je také možné vidět v pravé straně webové stránky.



### Přidání nové události



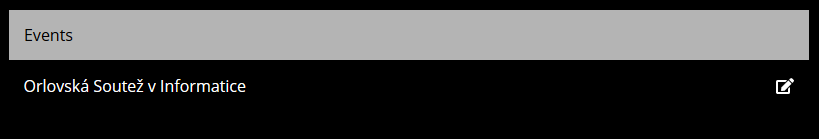
První nastavení je začátek události ve formátu *DEN.MĚSÍC.ROK HODINA:MINUTA*.

Druhé je poté konec události opět ve stejném formátu jako v nastavení předchozím.

Třetí je poté titulek události. Měl by jednoduše identifikovat, o jakou událost se jedná.

Čtvrtý je název **URL** článku, jehož se tato událost týká a na který je poté uživatel odkázán při kliku na více informací o události.

Poslední je barva události. Barva pomáhá v kalendáři rozlišit jednotlivé události.



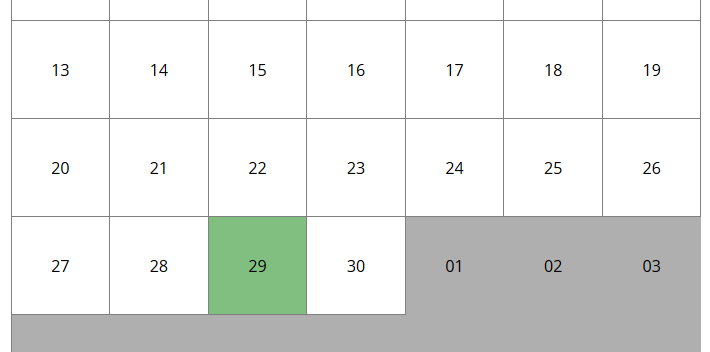
*Opět jako u předchozích prostředí, je možné vidět všechny dosavadně vytvořené události dole na stránce. Jelikož se tentokrát nejedná o lokalizovaný obsah, není nutné jej rozlišovat pomocí jazyku.*

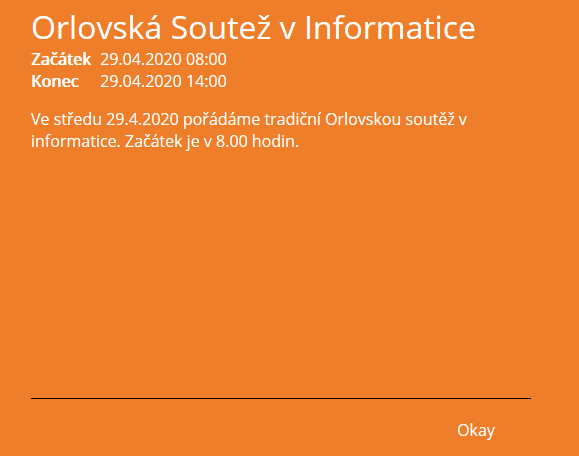
#### Ilustrační vklad události



*Vidíme, že událost začína 29.4 v 8:00 a končí ve stejný den v 14:00, v kalendáři bude mít barvu červenou a odkazuje na článek orlovska-soutez-v-informatice*

#### Ilustrační vzhled vložené události v kalendáři



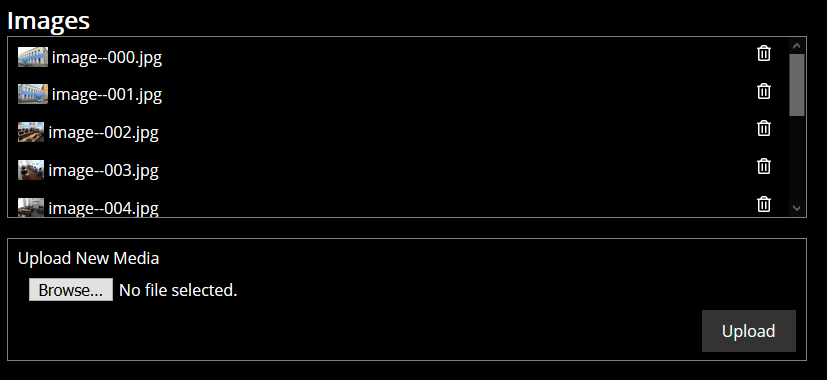


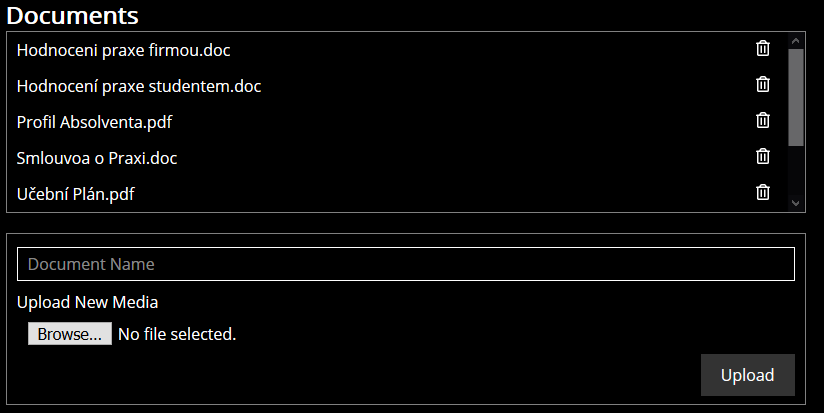
*Dialogové okno po kliknutí na událost.*

### Přidání médii

Média jsou soubory nebo obrázky dostupné pro návštěvníky webu. Obrázky jsou zobrazené v galerii, zatímco dokumenty a další soubory v listu dokumentů ke stažení.

#### Ilustrační vklad obrázku a dokumentu



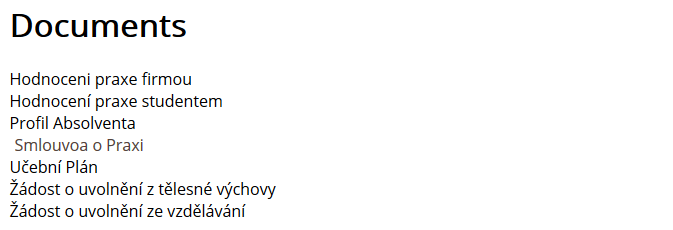


Dokumenty na rozdíl od obrázků musí mít jasně zvoleno jméno, aby uživatel jasně věděl o jaký dokument se jedná, a jeho jméno je pak následně ukázáno na stránce s dokumenty.

#### Ilustrační vzhled vložených obrázků v galerii



#### Ilustrační vzhled vložených dokumentů ke stažení

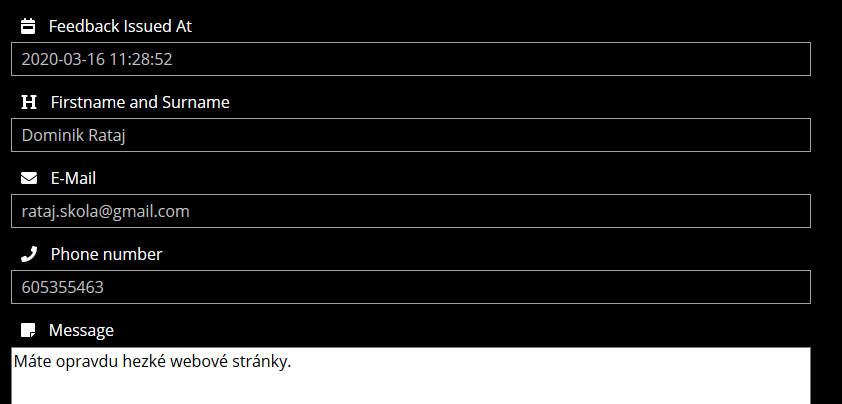


### Zobrazení zpětné vazby návštěvníků aplikace

Zpětnou vazbu mohou uživatelé zaslat podle formuláře uvedeného na stránce jako přímý kontakt. Administrátor jej pak může zobrazit, smazat ale nelze je upravit.

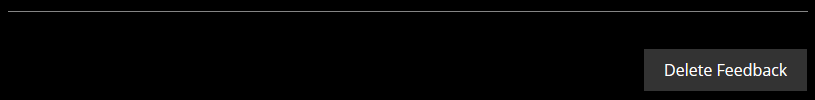


Ilustrační list všech odeslaných zpětných vazeb. Uživatel je povinen uvést svoji emailovou adresu. První v listu je ID zpětné vazby a další je kontaktní email.



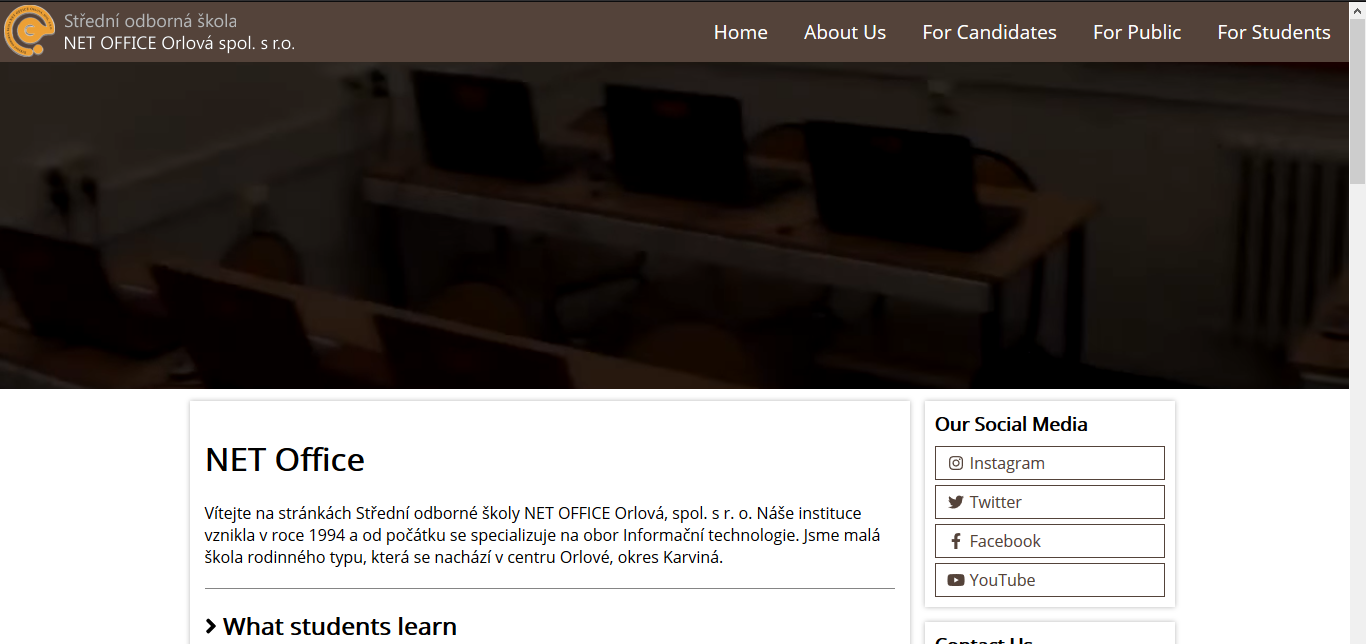
Zobrazení zpětné vazby. Je možné vidět, kdy byla zpětná vazba poslána, jméno a příjmení (pokud bylo zadáno), e-mail a telefonní číslo (pokud bylo uvedeno).

Poslední je poté samotné znění zpětné vazby.

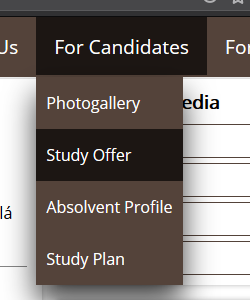


Hned po zprávě je tlačítko na vymazání zpětné vazby jeli to nutné.

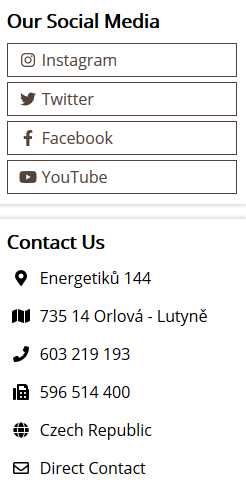
## Uživatelské prostředí návštěvníka



Všechny stránky zobrazují hlavičku s odkazy na jednotlivé stránky a logem školy. V horní liště (header) je možné vidět všechny odkazy. Horní lišta je vždy (kromě mobilního režimu) vidět a je možné s ní provádět interakce. Dále pokračuje ilustrační video ze školy, tento styl podporuje moderní design stránek a hezky ukazuje, že je stránka dynamická a neobsahuje jen jednoduchý styl.



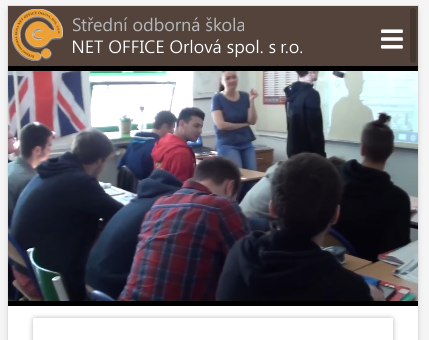
Při najetí na jednotlivé sekce v hlavičce se zobrazí více možnosti vázané k dané sekci.



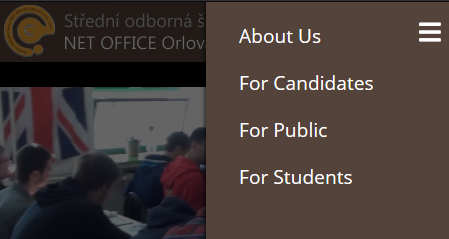
V pravé části stránky vidíme sociální média školy, kontakt a také kde se škola nachází.

### Responsivita

Responsivita je zajištěna stylovým programování podle *media query* a JavaScript kódu, který změní nastavení hlavičky podle velikosti okna.



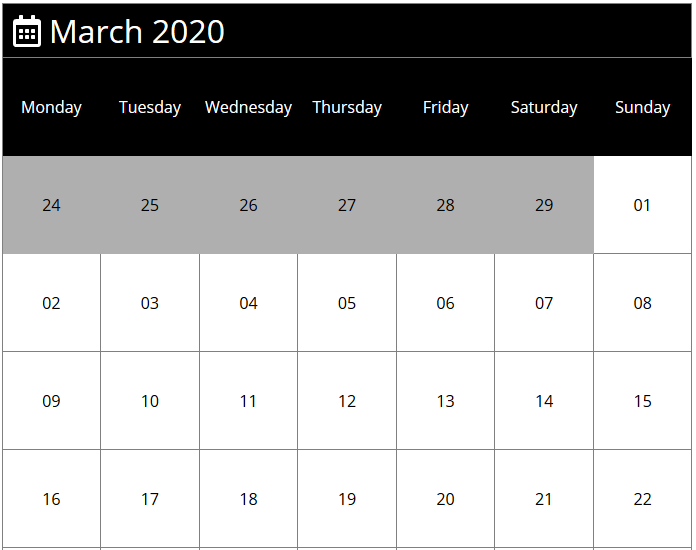
Zobrazení webu pro mobilní zařízení.



Při kliku na ikonku menu v mobilním režimu se objeví jednodušší zobrazení ve vertikální podobě. Animace jsou zde taky přítomny a připomínají moderní styl designu podle Microsoft UWP.

### Kalendář

Speciální části aplikace je kalendář akcí.



Kalendář používá měsíce a rok pro zobrazení událostí, o kterých jsem již výše psal.



Kalendář se poté dá dole posouvat na další nebo předchozí měsíce.

### Kontaktní Formulář