# Vložení/úpravy článku

Úloha je řešena jako specializovaný **applet jedné z implementací šablony jednotného administračního rozhraní** systému DOTAZ (viz dokumentace jednotného administračního rozhraní).

Applet pro tento účel má název **a\_article\_author**.php a je jediným výhradním apletem rozhraní **article\_author**.php. (=rozhraní autora článku, kopie aplikační šablony)

V souladu se šablonou rozhraní je aplet implementován jako dynamický formulář (generovaný php skriptem) pro přidání nebo úpravu článku. Obsahuje různé prvky pro správu obsahu článku, přidělování autorů, nahrávání souborů a další úpravy. Kombinuje HTML šablonu, Bootstrap stylování a jQuery/AJAX pro dynamické načítání dat a spolupracuje s backendovými php skripty, z nichž některé jsou výhradní.

Rozhraní appletu při vkládání (volba Nový článek z menu): Obsah obrázku text, software, Počítačová ikona, Operační systém

Popis byl vytvořen automaticky

Varianta appletu pro editaci (výběrem z nabídky dostupných článků):Obsah obrázku text, snímek obrazovky, software, Počítačová ikona

Popis byl vytvořen automaticky

# Popis HTML kódu

* **Nadpis (<h5 id="edit\_header">):**

Zobrazuje kontext formuláře (např. „Vložení nového článku“ nebo „Úprava článku“); volba provedena skriptem při načtení dat/vymazání dat

* **Formulář (<form>):**

Obsahuje jednotlivé sekce:

1. Název článku (input)
2. Správa autorů (výběrové seznamy)
3. Textová pole (abstrakt, poznámka pro redakci)
4. Nahrávání souborů (dokument a obrázek)
5. Výběr edice (select)

## Podrobný popis HTML sekce Správa autorů

Obsahuje podsekce „Autoři článku“ a „Registrované osoby“ s vstupními poli

* Výběrové seznamy:

1. **#selectedAuthors**: Autoři přiřazení k článku
2. **#registeredAuthors**: Všichni registrovaní autoři/osoby

* Šipky pro přesun autorů mezi seznamy (tlačítka):

1. **authorAdd()**: Přidá autora z „Registrované osoby“ do „Autoři článku“
2. **authorRemove()**: Odstraní autora z „Autoři článku“

* Filtr (checkbox) „jen autoři“ (script filterAuthors()):

Zajišťuje filtrování registrovaných osob:

* + Když je checkbox „jen autoři“ zaškrtnut, zobrazují se v **#registeredAuthors** jen uživatelé definovaní jako autoři (volá se authorsLoad(2,0))
  + Jinak se zobrazují všichni uživatelé (authorsLoad(1,0))

## Ostatní prvky

* **Nahrávání souborů:**

Input pro nahrávání dokumentu (#document) a obrázku (#image)

Používá skryté <input> elementy a viditelné tlačítko

* **Tlačítka:**

Uložit: Odesílá formulář k uložení článku

## Subformulář registrace autora

Slouží k doplnění autora do databáze, pokud ještě není registrován

TODO

# Funkce JavaScriptu

## Scripty sloužící k dynamickému načítání dat pomocí AJAX

**Načítání edic:**

Funkce editionsLoad(type, index)

Volá API getEditions.php a naplní <select id="edice"> seznamem edic

**Načítání autorů:**

Funkce authorsLoad(type, index)

Volá API getUsers.php a naplní seznam registrovaných osob

**Načítání detailů článku:**

Funkce aFormLoad(index)

Načte informace o článku pomocí API getArticle.php

Vyplní formulář údaji o článku: Název článku, seznam autorů, abstrakt atd

**Vyprázdnění formuláře:**

Funkce aFormEmpty()

Vymaže veškeré hodnoty formuláře

Připraví formulář pro přidání nového článku

## Scripty pro manipulaci s formulářem

**Přidání autora z výběru „Registrované osoby“ do „Autoři článku“:**

Funkce authorAdd()

Ověřuje:

* Že autor již není v seznamu „Autoři článku“
* Že výběr není prázdný

**Odstranění vybraného autora ze seznamu „Autoři článku“**

Funkce authorRemove()

**Filtrování registrovaných osob na základě stavu checkboxu „jen autoři“**

Funkce filterAuthors()

**Reset údajů existujícího článku**

Funkce aFormEmpty()

## Scripty pro odesílání dat

**Odeslání dat hlavního formuláře:**

Funkce aPost()

Vzhledem k nutnosti odeslat i soubory není napsána jako standardní AJAX post, ale jako obecný ajax script. Využívá API setArticleAuthor.php

**Odeslání dat subformuláře registrace uživatelů:**

Funkce RegisterAuthors()

Využívá API RegisterAuthors.php

# Funkce API backendu

Dynamické filtrování dat - getEditions.php (sdílený)

**Načtení vstupních parametrů**

Získávají se dva parametry z URL:

* + id (uloženo jako $FilterID) – slouží k filtrování podle konkrétního ID edice.
  + typ (uloženo jako $FilterType) – určuje, jaký typ edicí bude vrácen (např. publikované, nepublikované).
* Pokud jsou parametry přítomné, jsou zpracovány bezpečně pomocí htmlentities.

**Generování SQL dotazu**

Dotaz se dynamicky upravuje podle hodnot parametrů:

* + **Výchozí stav**: Pokud není zadán filtr, vybere všechny záznamy z tabulky RSP\_EDITION.
  + **Nepublikované edice (typ=1)**: Pokud je typ=1, vyberou se pouze edice, které nemají vyplněnou hodnotu Published.
  + **Konkrétní edice (id>0)**: Pokud je zadáno id větší než 0, vybere se pouze edice s tímto ID. Tento podmíněný dotaz přepíše předchozí filtry.

**Spuštění SQL dotazu**

Dotaz se spustí pomocí $conn->query($sql) a výsledky se iterativně načítají.

**Načtení dat do pole**

Záznamy získané z databáze jsou iterativně ukládány do pole $data jako objekty pomocí fetchObject.

**Vrácení dat ve formátu JSON**

Obsah pole $data se převede na formát JSON s použitím JSON\_PRETTY\_PRINT pro čitelný výstup a odešle se jako odpověď.

Filtrování uživatelských dat - getUsers.php (sdílený)

**Načtení vstupních parametrů:**

* id (uloženo jako $FilterID) – číselný filtr, slouží například pro omezení na uživatele s určitým stavem nebo konkrétním ID.
* typ (uloženo jako $FilterType) – určuje způsob filtrování uživatelů podle rolí nebo jiných vlastností.

**Generování SQL dotazu:**

* **Výchozí dotaz (0,0)**: Vybírá všechny uživatele (Active >= $FilterID), seřazené podle příjmení a jména.
* **Filtr aktivních uživatelů (typ=1)**: Vybírá uživatele s aktivním stavem (Active > 0) a rolí vyšší než 10, což vylučuje administrátory.
* **Filtr autorů a oponentů (typ=2)**: Vybírá aktivní uživatele s konkrétními rolemi (21, 22, 24), což odpovídá dříve publikujícím autorům a oponentům.
* **Filtr oponentů a vyšších (typ=3)**: Vybírá aktivní uživatele s rolí mezi 10 a 21 (včetně), vylučující administrátory.
* **Konkrétní uživatel (typ=4)**: Vybírá uživatele s přesným ID (ID = $FilterID).

**Zpracování výsledků:**

Dotaz se spouští pomocí $conn->query($sql).

Výsledky jsou iterativně načítány pomocí fetchObject a ukládány do pole $data.

**Vrácení dat ve formátu JSON:**

Pole $data se převede na JSON s formátováním JSON\_PRETTY\_PRINT a odešle jako odpověď.

Získání údajů o článku - getArticle.php

**Načtení ID filtru z parametru URL**

Proměnná $FilterID je nastavena na hodnotu parametru id z URL, pokud je přítomen a bezpečně převeden na entitu pomocí htmlentities. Pokud není zadán, zůstává výchozí hodnota 0.

**Příprava datové struktury**

Proměnná $data je inicializována jako prázdné pole. Tato proměnná bude obsahovat informace o článku a přidružených autorech a oponentech.

**Získání údajů o článku**

* + - Pokud je $FilterID číselné a různé od nuly:
    - Načte se článek z tabulky RSP\_ARTICLE, kde ID odpovídá $FilterID.
    - Výsledky dotazu jsou přidány do $data jako první prvek.

**Získání seznamu autorů**

Pro stejný článek (Article=$FilterID) se dotazem na tabulku RSP\_ARTICLE\_ROLE získají osoby s rolí 24 (autoři), jejichž role je platná (tj. Active\_to je buď null, nebo v budoucnosti).

Osoby jsou spojeny do jednoho řetězce odděleného čárkami (GROUP\_CONCAT) a tento řetězec je přidán jako vlastnost authors k prvnímu prvku $data.

**Získání seznamu oponentů**

Podobně se z tabulky RSP\_ARTICLE\_ROLE získají osoby s rolí 21 (oponenti) pro tentýž článek s kontrolou platnosti (Active\_to).

Tento seznam je přidán jako vlastnost opponents k prvnímu prvku $data.

**Výstup dat v JSON**

Obsah $data

Filtr načítání článků z databáze - getArticles.php

**Načítání vstupních parametrů:**

* **id**: Určuje konkrétní kritéria filtrování, například edici, stav článku, nebo role uživatele.
* **typ**: Specifikuje typ výběru, například všechny články, články přiřazené konkrétním rolím, nebo články podle redaktora.

**Generování SQL dotazu na základě parametrů:**

Výchozí dotaz načítá všechny články s jejich stavy a odpovídajícími barvami.

Pokud je zadán konkrétní typ, upravuje se filtrování tak, aby:

* + Zahrnovalo jen články, u kterých má uživatel odpovídající role.
  + Vrátí pouze články z konkrétní edice.
  + Filtrovalo články bez přiřazené edice nebo ty, které nejsou ve finálních stavech.

**Zpracování výsledků:**

Dotaz se provede nad databází, a výsledky se ukládají do pole.

Každý článek obsahuje informace jako ID, edici, název, stav a příslušnou barvu.

**Formátování výstupu:**

Načtená data se převedou do formátu JSON, který je čitelný a snadno zpracovatelný pro front-end.

Stahování aktuální verze dokumentu - getDocument.php

**Ověření platnosti parametru:**

Pokud je parametr id neplatný (není číselný nebo je 0), skript ukončí a vrátí zprávu noID.

**Dotaz do databáze:**

Na základě zadaného ID článku vyhledá:

* + **Aktivní verzi** dokumentu.
  + Ověří přístup uživatele k článku pomocí funkce hasAccess.

**Ověření existence dokumentu:**

Pokud není nalezena žádná verze, skript ukončí a vrátí zprávu noFile.

**Příprava stahování:**

Zjistí příponu souboru a připraví odpověď HTTP s hlavičkami:

* **Content-Type**: Nastaveno na obecný typ application/octet-stream pro stahování.
* **Content-Disposition**: Nastavuje jméno souboru, které se zobrazí při stahování.

Soubor je přečten z adresáře a poslán uživateli prostřednictvím funkce readfile.

**Zpracování chyb:**

Jakákoli jiná chyba (např. selhání databázového dotazu) je zachycena a skript vrátí zprávu Other error: [chyba].

Filtr seznamu edic-getEditions.php

**Získání parametrů z URL:**

Parametr id: Určuje konkrétní edici (pokud je zadáno).

Parametr typ: Určuje typ filtrace:

* + 0: Všechny edice.
  + 1: Jen nepublikované edice.

V**ytváření SQL dotazu:**

Bez filtru (typ = 0): Vrací všechny edice.

Pouze nepublikované edice (typ = 1): Vrací edice, kde hodnota Published je null.

Specifická edice (id > 0): Vrací konkrétní edici dle zadaného ID.

**Vykonání SQL dotazu:**

Dotaz je vykonán pomocí objektu $conn.

Výsledky jsou načteny a uloženy jako objekty do pole $data.

**Výstup dat ve formátu JSON:**

Pole $data je převedeno na čitelný JSON pomocí funkce json\_encode s volbou JSON\_PRETTY\_PRINT.

Filtr seznamu událostí - getMessages.php

**Příjem a zpracování parametrů:**

* id: Identifikátor (např. článku, uživatele nebo edice) pro filtrování.
* typ: Typ filtrace, který určuje povahu událostí, které mají být zobrazeny (např. všechny, události k článku, autorům, oponentům apod.).

**Vytváření SQL dotazů:**

Dotazy se liší podle hodnot parametru typ:

* + **0 (výchozí):** Získání všech událostí bez ohledu na typ.
  + **1:** Získání událostí spojených s konkrétním článkem (id > 0).
  + **2:** Zobrazení událostí autora, buď všech, nebo specifických pro článek (id).
  + **3:** Zobrazení událostí oponenta, buď všech, nebo specifických pro článek.
  + **4:** Události spojené se správou uživatelů (id pro konkrétního uživatele, nebo všechny).
  + **5:** Události související s edicemi (id pro konkrétní edici, nebo všechny).

**Propojení tabulek:** Události (RSP\_EVENT) jsou spojeny s typy událostí (RSP\_CC\_EVENT\_Type), což umožňuje vrátit i popis typu události (TypeText).

**Filtrace podle rolí a přístupu:**

* **Role autora (22, 24):** Filtruje události typu 3, 6, 7.
* **Role oponenta (21):** Filtruje události typu 4, 5, 11, 12, 13.
* **Správa uživatelů (type=10):** Události týkající se specifických uživatelů.

**Zpracování výsledků:**

Výsledky dotazu jsou načteny do pole $data a převedeny na formát JSON pomocí json\_encode s volbou JSON\_PRETTY\_PRINT.

**Výstup:**

JSON obsahuje všechny relevantní informace o událostech, včetně textového popisu typu (TypeText), ID, přidruženého článku/edice a dalších detailů.

Výběr a získání hodnoty proměnné-getMyID.php

Vybere a zobrazí hodnotu proměnné $myID, která je definována v souboru session\_open.php. Tento soubor zajišťuje správu uživatelských session, a proměnná $myID by měla obsahovat ID aktuálně přihlášeného uživatele.

Vytvoření nebo úprava uživatelských záznamů v databázi - RegisterAuthors.php

**Kontrola POST požadavku:**

Skript provádí kontrolu, zda je požadavek typu POST. Pokud není, vrací chybu.

**Validace vstupních údajů:**

Získá a ověřuje údaje jako Login, FirstName, LastName, Mail, passwd1, passwd2, ID, a další.

Provádí kontrolu povinných polí a jejich formátů (např. validace emailu, shoda hesel).

**Kontrola duplicity uživatelského jména a e-mailu:**

Pokud je uživatel aktivní, provádí kontrolu, zda není již existující aktivní záznam s daným uživatelským jménem nebo e-mailem.

**Kontrola oprávnění:**

Zajišťuje, že uživatel má dostatečná oprávnění k provedení požadované operace (např. změny funkcí nebo údajů jiného uživatele).

**Vkládání nebo aktualizace uživatele:**

Pokud je ID uživatele 0 (nový uživatel), provede se vkládání nového záznamu do tabulky RSP\_USER.

Pokud je ID uživatele jiné než 0 (existující uživatel), provede se aktualizace záznamu.

**Úprava hesla:**

Pokud je zadáno nové heslo, provádí se jeho aktualizace pomocí hashování.

**Vytváření záznamu o události:**

Po vytvoření nebo úpravě uživatele se zaznamená událost do tabulky RSP\_EVENT, což je pravděpodobně součástí logování operací v systému.

# Popis jednotného administračního rozhraní

V průběhu návrhu projektu se vyčlenily dvě části: veřejná publikační a privátní administrační. Zatímco veřejná část jsou standardní dynamicky generované webové stránky, privátní část je ve své podstatě webovou aplikací.   
Při návrhu aplikace je vhodné definovat rozhraní globálně, aby byl zajištěn jednotný vzhled i chování rozhraní, a také efektivní využití jak frontendových partů (HTML, javascript) tak backendových API scriptů (php), které mohou být sdíleny a využívány napříč aplikačními rozhraními. Celá aplikace je postavena na kombinaci HTML šablony, Bootstrap stylování a jQuery/AJAX pro dynamické načítání dat a spolupracuje s backendových php skriptů.

Návrh jednotného rozhraní je specifikován jako HTML šablona (template.php), včetně separátní definice použitých css stylů (style.css). Šablona je navržena jako integrační (obalové) rozhraní pro čtyři applety: menu, list, main a messages. Tři jsou napsány jako sdílené a jsou součástí šablony (i když i ty je možné v případě potřeby nahradit), a čtvrtý aplet main není součástí šablony a vždy se doplní podle požadované funkce. Šablona obsahuje i sdílené javascripty, které v každé instanci musí být dodefinovány a využity k provázání funkce napříč aplety – naplnění apletů daty dle potřeby implementace, zpracování události apletu (typicky kliknutí uživatele na objekt sdíleného appletu), …

Šablona se používá tak, že pro každé rozhraní je vytvořena její kopie, do ní dosazeny požadované separátně definované aplety – včetně vlastních scriptů, a dodefinovány sdílené funkce.

# Popis HTML kodu

Šablona obsahuje čtyři (pojmenované divy stylu bootsrap „col“. V každé z nich je includován aplet (ze sekce include/applet). Standardně jsou použity aplety a\_menu, a\_article, a\_message, a čtvrtý je ponechán pro samotnou implementaci. Funkce sdílených apletů bude popsána níže.

# Funkce JavaScriptu

Sekce <script> obsahuje:

* **Volání $document.ready()** pro inicializaci prostředí
* **Šablonu funkce menuClick(item)** pro definici reakcí na položky menu,

s předvyplněnými vazbami mezi aplikačními interface (bylo a případně i bude doplňováno průběžně, tak jak se aplikace bude rozvíjet)

Sekce menu volá tuto funkci při kliknutí na jednotlivé položky s definovaným parametrem typu řetězec = identifikace položky

* **Šablonu (prázdnou) funkce articleClick(article,version)** pro definici reakcí na výběr (článku) z apletu articles

Sekce articles volá tuto funkci při kliknutí na vybraný článek z nabídky a předává ID článku a doplňující parametr – stav článku

* **Šablonu (prázdnou) funkce messageClick(message,version)** pro definici reakcí na výběr (článku) z apletu messages

Sekce messages volá tuto funkci při kliknutí na vybranou zprávu z nabídky a předává ID zprávy a doplňující parametr – druh zprávy

* **Funkci setLayout(styl)** pro změnu rozložení stránky

Funkce nastaví jedno ze čtyř předdefinovaných rozložení (změní šířky, skryje část main, skryje zprávy) nastavením (bootstrap) stylů k pojmenovaným apletovým divům (viz popis HTML) Je možné dodefinovat i další rozložení.

K nastavení apletů se používají povinné funkce těchto apletů (sdílené aplety tyto funkce obsahují a případný uživatelský aplet nahrazující menu, articles nebo messages je také musí obsahovat)

* **Funkci messagesLoad(styl,parm)** pro nahrání zpráv

Funkce naplní aplet messages požadovanými zprávami. Parametry se standardně předávají API getMessages.php „as is“), význam je tedy specifikován v rámci API.

* **Funkci articlesLoad(styl,parm)** pro nahrání seznamu článků

Funkce naplní aplet articles požadovanými zprávami. Parametry se standardně předávají API getArticles.php „as is“), význam je tedy specifikován v rámci API.

# Funkce API backendu

Šablona sama nevyužívá backend, k datům přistupuje pouze prostřednictvím sdílených appletů. Popis API tedy následuje u popisu těchto komponent.

# Applet a\_articles.php - Seznam článků

Tento skript je součástí webové aplikace a jeho hlavním úkolem je dynamicky načítat a zobrazovat seznam článků na základě typu a indexu. Umožňuje interaktivní zobrazení článků v uživatelském rozhraní a zpracovává data získaná prostřednictvím AJAX požadavků.

Obsah obrázku text, snímek obrazovky, software, Počítačová ikona

Popis byl vytvořen automaticky

**Popis funkcionalit**

**HTML část:**

Hlavní nadpis Moje články zobrazuje uživateli seznam jeho článků.

Prázdný seznam s ID articleList, který je dynamicky plněn položkami pomocí JavaScriptu.

**JavaScript část:**

**Funkce articlesLoad(type, index)**:

* + - Posílá AJAX požadavek na server prostřednictvím metody GET
    - Požadavek se směřuje na skript include/ajax/getArticles.php, který vrací seznam článků ve formátu JSON
    - Data obdržená ze serveru se iterují pomocí metody $.each a výsledky se dynamicky přidávají do seznamu articleList

**Interakce s uživatelem:**

Každý článek v seznamu se zobrazí jako položka (li) se svým ID a názvem.

Na položky lze kliknout, přičemž volání funkce articleClick(ID, 0) umožňuje další akce, jako například otevření detailu článku.

**Použití a konfigurace**

**Zobrazení seznamu článků:**

Funkce articlesLoad přijímá dva parametry:

* + - type: Typ článku, který bude načten (např. schválené, čekající na schválení).
    - index: ID uživatele nebo jiné identifikační číslo, které určuje, pro koho budou články načteny.

Funkce je volána například při načtení stránky nebo změně typu zobrazení článků.

**Integrace do HTML:**

Element <ul> s ID articleList musí být obsažen na stránce. Funkce JavaScriptu do něj vloží dynamický obsah.

Zajistěte, že knihovna **jQuery** je správně nainstalována a připojena k projektu.

**Známé problémy a jejich řešení**

1. **Prázdný seznam:**

Pokud se seznam článků nenačte, ověřte, zda:

* + - Skript getArticles.php vrací data ve správném formátu JSON
    - Parametry type a id jsou správně předávány

1. **Neplatná data:**

Pokud skript vrací chybu, zkontrolujte, zda databáze obsahuje požadovaná data a připojení k databázi je správně nakonfigurováno.

# Applet a\_menu.php - Menu s navigací

Tento kód představuje část uživatelského rozhraní, které obsahuje navigační menu. Menu umožňuje uživatelům snadný přístup k funkcím, jako je vytvoření nového článku, zobrazení vlastních článků nebo přístup ke zprávám.

**Popis funkcionalit**

**Navigace v menu**:

Kliknutí na položku menu (<li>) vyvolá JavaScriptovou funkci menuItemClick(), která provede konkrétní akci v závislosti na předaném parametru.

* + - **Parametr 1**: Otevře sekci *Nový článek*
    - **Parametr 2**: Otevře sekci *Autorské články*
    - **Parametr 3**: Otevře sekci *Zprávy*

**Zobrazení ikon**:

Obrázky (art\_new.png, art\_my.png, msg.png) reprezentují vizuální indikátory pro jednotlivé položky menu.

Dynamická cesta k obrázkům ({$img\_dir}) zajišťuje, že jsou obrázky načteny správně i při změně umístění nebo struktury složek.

**Použití a konfigurace**

**Správné nastavení proměnné $img\_dir:**

Proměnná $img\_dir by měla obsahovat relativní nebo absolutní cestu ke složce, kde jsou obrázky uloženy.

**Přidání dalších položek menu:**

Nové položky lze přidat jednoduchým zkopírováním jednoho z existujících <li> elementů a úpravou parametrů:

* + - Obrázku (src)
    - Textu (popis akce)
    - Parametru funkce menuItemClick

**Styling menu:**

Stylování je zajištěno kombinací Bootstrap tříd a inline stylů.

Pro přizpůsobení je možné přidat vlastní třídu CSS nebo upravit existující styly.

**Známé problémy a jejich řešení**

1. **Chybějící obrázky:**

Pokud se obrázky nezobrazují:

* + - Ověřte, zda proměnná $img\_dir ukazuje na správnou složku
    - Zkontrolujte, zda jsou obrázky ve složce správně pojmenovány

1. **Funkce menuItemClick nefunguje:**

Zajistěte, že JavaScriptová funkce menuItemClick je správně implementována a dostupná na stránce

# Applet a\_messages.php – „**modální“ okno Zprávy**

Tento kód představuje implementaci modálního okna Bootstrap, které slouží k zobrazení seznamu zpráv. Kód obsahuje HTML šablonu a připojený JavaScript pro načtení a zobrazení dynamických dat.

Obsah obrázku text, snímek obrazovky, software, Počítačová ikona

Popis byl vytvořen automaticky

**JavaScript**

**Funkce messagesLoad(type, index)**:

Slouží k načtení seznamu zpráv na základě zadaného **typu** a **indexu**.

**Pracovní postup funkce:**

**Volání AJAX pomocí $.getJSON**:

URL: include/ajax/getMessages.php – skript na serveru vrací seznam zpráv ve formátu JSON.

Parametry:

* + - type: Typ zpráv (např. podle kategorie).
    - index: Index pro stránkování nebo další filtr.

**Zpracování přijatých dat**:

Každá zpráva je iterována pomocí $.each().

Z každé zprávy (e) jsou použity tyto atributy:

* + - **ID**: Unikátní identifikátor zprávy.
    - **Message**: Obsah zprávy.
    - **Article**: ID článku, na který zpráva odkazuje.
    - **Type**: Typ zprávy.

# Analýza společného aplikačního rozhraní

Celá aplikace je díky jednotné koncepci a centrální šabloně vzhledově i funkčně kompaktní, přičemž jsou zdůrazněny následující principy:

## 1. Modularita a oddělení komponent

Všechny implementace vycházejí z modulární struktury, kterou původní šablona template.php nastavuje. Šablona definuje několik samostatných částí, které jsou využity napříč celou aplikací:

* Menu (levý panel), který je vždy společný pro všechny typy administrace a umožňuje přechod mezi různými moduly (např. administrace uživatelů, článků, vydání).
* Seznam objektů, ze kterého se vybírá entita ke zpracování (obvykle článek ale i vydání nebo registrovaný uživatel), má jednotné pozicování a podobný formát.
* Hlavní obsah a Formuláře jsou přizpůsobeny pro různé administrativní úkoly a aplikují podobné techniky jako v šabloně, tedy načítání dat a zobrazování formulářů, které se mění na základě kontextu (např. editační formuláře pro články nebo uživatele).
* Zprávy a notifikace jsou zajišťovány podobně ve všech implementacích, vždy na základě AJAXových volání pro komunikaci se serverem a uživatelem.

## 2. Adaptace na specifické administrativní úkoly

Původní šablona poskytuje základní strukturu a mechanismy pro administraci, ale každá implementace přizpůsobuje tuto strukturu specifickým úkolům:

article\_author: Zaměřuje se na administraci článků autora, což zahrnuje jak zobrazení seznamu článků, tak jejich tvorbu a editaci.

article\_opponent: Zde je zaměření na přezkoumání článků a podání oponentury.

article\_redactor: Zde je zaměření na administraci článků z pozice redaktora / administrátora

*připraveny a implementovány budou i další administrativní šablony:*

user\_admin: Umožňuje správu uživatelů, kde je k dispozici formulář pro přidání a úpravu uživatelských účtů a přidělování funkcí.

edition\_admin: Zaměřuje se na správu vydání, což je jedinečný administrativní úkol zahrnující přidávání a úpravu informací o publikovaných vydáních.

V každé z těchto implementací je základní struktura menu, formuláře a zobrazení zpráv převzata ze šablony, ale logika a obsah se mění podle konkrétního zaměření (správa uživatelů, článků, vydání).

## 3. Dynamické načítání a zpracování dat

Všechny implementace využívají podobný mechanismus pro práci s dynamickými daty:

AJAX a JSON pro načítání dat ze serveru bez nutnosti obnovování stránky (např. pro načítání seznamu článků, uživatelů, vydání nebo editovaných dat).

Formuláře jsou předvyplněny dynamicky (např. když je editován existující článek, uživatel nebo vydání), což přispívá k uživatelsky přívětivému zážitku.

Po úpravách se data odesílají zpět na server a podle výsledku je uživatel informován o úspěchu nebo chybě (pomocí AJAXu), což zajišťuje hladkou interakci.

## 4. Oprávnění a bezpečnost

Každá implementace obsahuje podobné mechanismy pro kontrolu oprávnění a zajištění bezpečného přístupu k administrativním funkcím:

Kontrola oprávnění uživatele pomocí proměnné $myFunc, která určuje, zda má uživatel dostatečné privilegium pro přístup k určitému modulu. Pokud není splněno požadované oprávnění (např. funkce nižší než určité hodnoty), dojde k přesměrování uživatele. Stejné ověřování pak provádí i backendová API.

Bezpečné zpracování formulářových dat je zajištěno ověřováním a validací uživatelských vstupů už před jejich odesláním na server.

## 5. Použití společných funkcí pro správu formulářů

Všechny implementace používají podobné funkce pro práci s formuláři:

Funkce pro vymazání formulářů (aFormEmpty), která zajišťuje, že při přechodu mezi různými formuláři jsou všechna pole resetována na prázdné hodnoty.

Funkce pro načítání existujících dat do formulářů (aFormLoad), která zajistí, že při úpravách již existujících položek (např. článků, uživatelů, vydání) jsou všechna pole předvyplněná s aktuálními hodnotami.

## 6. Interaktivita a uživatelsky přívětivý design

Ve všech implementacích je kladen důraz na intuitivní a interaktivní uživatelské rozhraní:

Použití dinamických funkcí jako je změna vzhledu tlačítek, aktivace a deaktivace polí na základě uživatelských akcí (např. změna stavu "aktivní" u uživatele nebo přidání nové položky).

Vizuální zpětná vazba pro uživatele (např. změna barvy polí ve formulářích při chybě, indikace úspěchu nebo chyby při odeslání formuláře).

## 7. Opětovné využití šablon a komponent

Opětovné použití šablon a komponent je silným principem. Implementace jako article\_author, article\_opponent, user\_admin, a edition\_admin se opírají o společné části šablony (menu, hlavičky, zápatí, apod.), které jsou používány ve všech implementacích, což zajišťuje konzistenci a snazší správu aplikace.

Modulární kód: Všechny implementace zavádějí nové specifické součásti (např. formuláře pro různé administrativní úkoly), ale základní struktura šablony a její opětovné použití v nových kontextech zůstává nezměněna.

## Shrnutí

Všechny implementace rozvíjejí původní šablonu template.php tím, že ji adaptují na konkrétní administrativní úkoly, ale stále zachovávají její základní strukturu. Klíčovým principem je modularita, opětovné použití komponent a flexibilita pro různé administrativní funkce. Společnými rysy jsou dynamické zpracování dat, bezpečnostní prvky a zajištění hladké uživatelské interakce. Implementace se vždy přizpůsobují konkrétním požadavkům administrace (správa článků, uživatelů, vydání), ale všechny vycházejí z jednotné šablony a zajišťují jednotný vzhled a chování administrativního rozhraní.