FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Rozšíření Pico CMS pro autentizaci
Student:	Pavel Tůma
Vedoucí:	Ing. Jiří Dostál
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Seznamte se s minimalistickým systémem Pico CMS a možnostmi tvorby rozšíření pro tento systém.
Analyzujte současná řešení autentizace a autorizace uživatelů dostupná pro tento systém.
Prostudujte možnosti realizace Single Sign-On (SSO) přihlašování s využitím protokolu OAuth.
Pro systém Pico implementujte rozšíření, které zajistí řízení přístupu ke stránkám.
Autentizaci bude zajišťovat vlastními uživatelskými účty nebo pomocí technologie SSO.
Výsledné rozšíření otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 12. prosince 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Rozšíření Pico CMS pro autentizaci

Pavel Tůma

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Dostál

14. května 2018

Poděkování

Děkuji vedoucímu své bakalářské práce Ing. Jiřímu Dostálovi za návrh zajímavého tématu, vstřícnost při konzultacích a cenné rady při tvorbě práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 14. května 2018 ……

Odkaz na tuto práci

Tůma, Pavel. *Rozšíření Pico CMS pro autentizaci*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce se zabývá analýzou, návrhem a implementací rozšíření minimalistického systému Pico CMS, navrženého pro správu webových stránek. Rozšíření umožní autentizaci uživatelů a konfigurovatelné omezení jejich přístupu ke stránkám. Přihlašování do systému bude umožněno vlastními uživatelskými účty nebo pomocí technologie SSO (Single Sign-On – jednotné přihlášení pomocí zavedeného účtu u důvěryhodného poskytovatele). Práce dále popisuje možnosti použití protokolu OAuth 2.0 pro autentizaci a při návrhu uvažuje možná bezpečnostní rizika.

Výsledné rozšíření může využít komunita uživatelů systému Pico pro zpřístupnění určitých částí jejich webové prezentace pouze autorizovaným uživatelům. V příloze práce lze nalézt zdrojový kód rozšíření a jeho uživatelskou dokumentaci.

Klíčová slova Rozšíření pro Pico CMS, plugin, autentizace, autorizace, uživatelské účty, návrh, implementace, bezpečnost, Single Sign-On, protokol OAuth 2.0

Abstract

This thesis deals with the analysis, design and implementation of a plugin for Pico CMS – simple, flat file web content management system. The plugin will provide a user authentication and a configurable access restriction to pages. Authentication will be achieved via local user accounts managed by the plugin or via a Single Sign-On (SSO) solution. This thesis also describes the options for authentication using the OAuth 2.0 protocol and discusses potential security risks of the final implementation.

The resulting plugin can be used by the community of Pico CMS users for allowing access to some parts of their web presentation only to the authorized users. The attachment contains a source code of the plugin and a documentation reference.

Keywords Pico CMS plugin, authentication, authorization, user accounts, design, implementation, security, Single Sign-On, OAuth 2.0 protocol

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Webové systémy pro správu obsahu jsou v dnešní době populárními prostředky pro majitele internetových prezentací. Pico CMS (Content Management System – systém pro správu obsahu) je minimalistický systém pro správu webového obsahu, ve kterém jsou jednotlivé stránky reprezentovány jako prosté textové soubory a k jejich formátování je použit jednoduchý jazyk Markdown [1].

Jedním z požadavků majitelů stránek může být potřeba omezení jistých částí jejich webové prezentace pouze určité skupině návštěvníků. Proces ověření identity takových subjektů se nazývá autentizace a následný proces určení přístupu autorizace. V současné době je pro přihlašování do různých webových služeb často nabízena možnost takzvaného jednotného přihlášení SSO (Single Sign-On). Její výhoda spočívá v tom, že si uživatel nemusí vytvářet nové přihlašovací údaje a k přihlášení použije svůj stávající účet na některé jiné službě. Dochází tak ke zjednodušení a následně i zrychlení procesu autentizace. Samotný systém Pico CMS tuto možnost neobsahuje, ale umožňuje rozšíření svých funkcionalit prostřednictvím pluginů.

Výsledek práce může být využít komunitou uživatelů publikačního systému Pico CMS v případě, že nastane potřeba ověřit identitu návštěvníků jejich stránek a na jejím základě umožnit nebo omezit jejich přístup k jednotlivým stránkám. Téma jsem si zvolil, neboť v současné době není k dispozici plugin pro aktuální verzi systému Pico CMS, který by potřebné funkcionality zajistil.

Cílem analytické části práce je detailní seznámení se s publikačním systémem Pico CMS a s možnostmi jeho rozšíření, dále analýza v současné době dostupných řešení autentizace a autorizace uživatelů pro tento systém a následně analýza možností realizace jednotného přihlašování se zaměřením na protokol OAuth 2.0. Cílem praktické části práce je návrh a implementace rozšíření pro systém Pico CMS, které umožní autentizaci jak prostřednictvím vlastních uživatelských účtů, tak pomocí technologie jednotného přihlášení a poskytne konfigurovatelné řízení přístupu ke stránkám vedeným v systému.

Kapitola 1

Analýza

1.1 Pico CMS

Systém Pico CMS (dále jen systém Pico) je minimalistický systém s otevřeným zdrojovým kódem pro správu webového obsahu, napsaný v jazyce PHP a dostupný pod licencí MIT. Jeho předností je zejména malá velikost a jednoduchost použití. Pro jeho instalaci stačí na webovém serveru pouze přítomnost PHP ve verzi nejméně 5.3.6 – nevyžaduje žádné další závislosti [2]. Protože jako úložiště veškerých dat využívá soubory prostého textu v souborovém systému webového serveru, není na rozdíl od většiny běžně používaných systémů pro správu obsahu závislý na relační databázi [3].

Popis hlavních aspektů tohoto systému je obsažen v několika následujících podsekcích. Pokud není uvedeno jinak, jde o vlastnosti v současné době nejnovější verze 2.0, nicméně většina uvedených vlastností je platná i pro předchozí verze systému.

1.1.1 Editace stránek

Pro formátování stránek je použit jednoduchý značkovací jazyk Markdown. Jde o formát prostého textu určený k tvorbě strukturovaných dokumentů, který byl v posledních letech adaptován mnoha webovými službami pro převod formátovaného prostého textu na HTML a jiné formáty [4]. Dle autora tohoto značkovacího jazyka bylo cílem vytvořit formátovací syntaxi takovou, aby byla co nejlépe čitelná. Soubor formátovaný jazykem Markdown by měl být čitelný, aniž by bylo příliš zjevné, že obsahuje formátovací instrukce a tagy [5].

Systém Pico neposkytuje pro editaci vytvářených stránek žádné uživatelské rozhraní. Vzhledem k tomu, že obsah každé stránky je uložen v samostatném textovém souboru a zároveň je formátován jednoduchým způsobem, editace je prováděna přímo běžným textovým editorem. Při zobrazení webové stránky pak systém Pico provede převod syntaxe jazyka Markdown do HTML

a webový prohlížeč obsah zobrazí dle stylů aktuálně nastavené šablony vzhledu stránky [2]. Celou instalaci lze tak i snadno přesunout na jiný server.

Na začátek každého souboru stránky systém Pico umožňuje uvést sekci s meta informacemi. Od samotného obsahu stránky je oddělena sekvencí pomlček a k její reprezentaci je použita syntaxe YAML – snadno čitelný formát pro serializaci dat [6]. Mezi meta informace lze uvést údaje jako je název stránky, její popis, autor, datum, šablona, která má být použita k jejímu zobrazení, pravidla pro indexaci stránky ve vyhledávačích a případně další uživatelsky definované parametry [2].

1.1.2 Architektura systému

Všechny HTTP požadavky na stránky vedené v systému Pico jsou směřovány na hlavní spouštěcí skript `index.php`. Ten zajistí zavedení automatického nahrávání tříd (*autoloader*) a inicializaci instance třídy Pico, která vykoná zpracování požadavku. Adresářová struktura celého systému je velmi jednoduchá a je znázorněna na obrázku 1.1.

```
config ..... Konfigurační adresář
    config.yml ..... Konfigurační soubor systému
content ..... Adresář se soubory stránek
    index.md ..... Hlavní stránka
lib ..... Zdrojové kódy systému
    Pico.php ..... Hlavní třída systému Pico
    PicoPluginInterface.php ..... Rozhraní pluginů
    AbstractPicoPlugin.php ..... Abstraktní třída pluginů
plugins ..... Adresář s pluginy
themes ..... Adresář se šablonami vzhledu
index.php ..... Zaváděcí skript
```

Obrázek 1.1: Základ adresářové struktury systému Pico [2]

Nevýhodou takové adresářové struktury je, že ve veřejném adresáři webového serveru se nacházejí i soubory, které by neměly být veřejně přístupné, jako je zejména konfigurační soubor systému nebo textové soubory jednotlivých stránek. Zakázání přístupu k těmto souborům tak musí být řešeno nastavením webového serveru. Systém Pico ve výchozím stavu obsahuje soubor `.htaccess` [2], kterým lze na úrovni adresářů upravit některá nastavení serveru Apache [7]. Tento soubor je předkonfigurován tak, aby všechny požadavky na neveřejné adresáře webový server přesměroval na zaváděcí skript `index.php`. V případě, že je ale systém Pico nainstalován na jiný webový server, který konfiguraci v souboru `.htaccess` nerespektuje, adresáře budou přístupné a uživatel musí provést dodatečnou konfiguraci pro zakázání jejich zobrazení.

Alternativně lze provést změnu souboru `index.php`, který obsahuje inicializaci systému Pico, jejímiž parametry jsou cesty ke konfiguračnímu adresáři, adresáři s pluginy a adresáři se šablonami vzhledu. Cestu k umístění souborů stránek je možné změnit přímo v konfiguračním souboru systému Pico. Tyto cesty je vhodné upravit tak, aby odkazovaly na umístění mimo veřejný prostor webového serveru. Dojde tak ke snížení rizika jejich kompromitace nezávisle na správné či chybné konfiguraci webového serveru.

1.1.3 Formát URL adres

Výsledná URL adresa každé stránky v systému je určena názvem textového souboru s jejím obsahem a jeho umístěním v případných podadresářích adresáře `content`. Například identifikátor stránky s umístěním `content/sub/page.md` bude `sub/page`. Je-li v adresáři obsažena stránka s názvem `index.md`, část se slovem `index` nemusí být uvedena. Tento identifikátor stránky je předán jako GET parametr v HTTP požadavku na `index.php`. Výsledná URL adresa, pod kterou je stránka přístupná, je `/?sub/page`. Pokud je na webovém serveru aktivován přepisovací modul, může být adresa stránky převedena na lépe vypadající tvar `/sub/page`. Toto nastavení přepisovacího modulu je pro webový server Apache obsaženo v souboru `.htaccess` dodávaném s instalací systému Pico [2].

1.1.4 Způsob instalace

Systém Pico umožňuje dva způsoby instalace. Doporučovaný způsob využívá nástroj Composer – správce závislostí pro PHP. Instalace spočívá ve spuštění příkazu `composer create-project picocms/pico-composer` a uvedení adresáře, kam má být systém nainstalován. Výhodou tohoto způsobu je snadné provedení případných aktualizací samotného systému, jeho závislostí, nebo i pluginů a šablon. Jedná se však o nástroj, který je spouštěn z příkazové řádky a nemusí tak být dostupný pro všechny uživatele. Proto je k dispozici i druhý způsob instalace, který spočívá ve stažení kompletního archivu s instalací systému Pico včetně předem stažených závislostí a následném rozbalení tohoto archivu do veřejného prostoru webového serveru [2].

1.1.5 Možnosti konfigurace

Určité vlastnosti systému jsou konfigurovatelné prostřednictvím konfiguračního souboru `config.yml`. Mezi nastavitelné hodnoty patří například název webové stránky, volba šablony vzhledu, možnosti řazení stránek při jejich výpisu, umístění adresáře se stránkami nebo parametry analyzátoru syntaxe Markdown. Stejně jako pro meta informace v úvodní části souborů stránek je pro tento soubor zvolen formát YAML. V konfiguračním souboru mohou být kromě obecných nastavení systému specifikována i nastavení jednotlivých instalovaných pluginů a šablon vzhledů [2].

1.1.6 Možnosti tvorby pluginů

Systém Pico umožňuje rozšíření svých funkcionalit prostřednictvím pluginů. Pluginy jsou třídy jazyka PHP umístěné v adresáři `plugins` v adresářové struktuře systému, viz obrázek 1.1. Pluginy mají definované metody, jejichž vykonání je spouštěné událostmi, které nastávají v průběhu zpracování stránky systémem Pico. Než systém Pico začne stránku zpracovávat, je provedeno načtení obsahu adresáře, v němž jsou pluginy obsaženy. Systém si pluginy zaregistruje a na události, které jsou vyvolávány během zpracování stránky, pak volá příslušné metody těch pluginů, které mají na dané události definované obsloužení. Pluginy tak získávají možnost napojení na zpracovávání požadavku a mohou jej ovlivňovat. V tabulce C.2 jsou uvedeny všechny dostupné události verze Pico 2.0 [1].

1.1.7 Změny uvedené ve verzi 2.0

Pico 2.0 je v době tvorby této práce nejnovější verzí tohoto systému. K jejímu prvnímu vydání došlo na konci roku 2017, dva roky po prvním vydání verze 1.0 [8]. Ačkoliv tato verze uvádí větší množství změn, následujících několik odstavců zmiňuje ty, které se týkají tvorby pluginů, a jsou proto podstatné pro návrh. Kompletní seznam změn je uveden v [8].

Došlo k vylepšení procesu instalace pluginů a šablon vzhledů. Ty je nyní možné instalovat příkazem nástroje Composer, který následně umožňuje i jejich snadnou aktualizaci. Uživatel tak například nemusí kontrolovat, zda nebyla pro některý z použitých pluginů vydána nová verze, ale spustí pouze obecný příkaz pro aktualizaci, který nahraje nové verze u těch částí, u kterých jsou k dispozici.

Konfigurační soubor systému Pico nyní využívá syntaxi YAML, jejíž cílem je snadná čitelnost pro uživatele [6]. V předchozí verzi byl pro konfiguraci používán PHP soubor s nastaveními jednotlivých konfiguračních parametrů do globální proměnné `$config`. Takové řešení mělo výhodu, že pokud webový server dovolil jeho načtení, obsah byl vyhodnocen jako PHP soubor bez výstupu a nedošlo k odeslání konfigurace v odpovědi.

Pluginy mohou definovat závislosti na jiných pluginech. Systém Pico při své inicializaci provede topologické seřazení všech aktivních pluginů dle seznamu jejich závislostí tak, aby závislosti byly splněny. Systém Pico tak zajistí, aby každý plugin závislý na vykonání metody jiného pluginu byl odbaven následně po metodách, na nichž závisí.

Dále tato verze upravuje specifikaci některých událostí a přidává i nové, čímž rozšiřuje možnosti pluginů. Tato změna však způsobuje nekompatibilitu pluginů vytvořených pro nižší verze systému Pico. V určitých případech je možné pluginy ze starších verzí použít pomocí pluginu PicoDeprecated.

1.1.8 Plugin PicoDeprecated pro kompatibilitu

PicoDeprecated je standardní plugin pro systém Pico 1.0 a 2.0, který umožňuje kompatibilitu pro pluginy vytvořené pro starší verze systému Pico. Rozpozná pluginy určené pro starší verze a funguje jako adaptér pro rozhraní verze, ve které je nainstalován, do verze, ze které pochází daný plugin [9]. Nelze jej obecně použít pro všechny pluginy, protože ne pro všechny změny bylo možné zachovat zpětnou kompatibilitu – například pokud starší rozhraní předávalo parametr referencí a nové již předává pouze hodnotou, nebo pokud pluginy přistupují přímo k atributům instance systému Pico, které mohou v závislosti na verzi obsahovat odlišné typy hodnot.

1.2 Analýza současného řešení

Dle seznamu všech dostupných pluginů pro systém Pico [10] jsou k dispozici dva pluginy zajišťující restrikci přístupu ke stránkám. Jedná se o Pico Private a Pico Users. Jejich analýzou se zabývá tato sekce.

1.2.1 Plugin Pico Private

Pico Private umožňuje autentizaci uživatelů uživatelským jménem a heslem prostřednictvím přihlašovacího formuláře. Plugin má přihlašovací údaje uživatelů definovány ve vlastním konfiguračním souboru ve formátu PHP, kde jsou reprezentovány jako pole hodnot. Konfigurační soubor také umožňuje volbu hashovacího algoritmu pro reprezentaci hesel – výchozím nastavením je SHA1, což není vhodný algoritmus pro hashování hesla [11]. Je však možné vhodnou konfigurací nastavit i Bcrypt, jehož bezpečnost je dostačující [11].

Plugin lze pro autorizaci využít dvěma způsoby. První způsob, který se v konfiguraci pluginu aktivuje nastavením parametru `private` na `all`, vyžaduje autentizaci pro přístup k libovolné stránce. Druhý způsob, `meta`, vyžaduje přihlášení pouze u stránek, jejichž meta informace uvedené na začátku každého souboru s obsahem stránky zahrnují parametr `Private: true` [12].

Tento plugin je určen pro systém Pico ve verzi 0.9. Ačkoliv je v novějších verzích systému Pico kompatibilita s pluginy určenými pro starší verze umožněna pluginem PicoDeprecated [9], plugin Pico Private není v jeho současném stavu možné použít ve vyšší verzi systému Pico než 0.9 kvůli způsobu, který používá pro přesměrování. Použitý způsob nepočítá s tím, že systém Pico od verze 1.0 v konfigurační hodnotě `base_url` obsahuje URL adresu aktuální instalace systému Pico, která je, na rozdíl od předchozích verzí, vždy zakončena lomítkem. Dojde tak k přesměrování na stránku `//login`, plugin ale provede přesměrování vždy, není-li aktuální stránka právě `/login` [12], to ale nenastane a dojde tak k zacyklení při přesměrování. Jde o nedostatek, který lze snadno odstranit, nicméně funkcionality tohoto pluginu jsou z velké části nahrazeny novějším pluginem Pico Users.

1.2.2 Plugin Pico Users

Plugin Pico Users umožňuje přihlašování uživatelů do systému, členění uživatelů do hierarchických skupin a jednoduchou správu práv přístupu jednotlivých uživatelů či skupin ke stránkám v systému [13].

Seznam uživatelů a přístupových práv je definován v konfiguračním souboru systému Pico. Jde o indexy `users` a `rights` globálního konfiguračního pole. Index `users` obsahuje přihlašovací údaje uživatelů (hesla jsou hashována algoritmem Bcrypt) a jejich přiřazení do skupin. Pokud se místo řetězce nachází v hodnotě další pole, jde o skupinu, která obsahuje další definice uživatelů nebo skupin (lze definovat mnohonásobně zanořené podskupiny). Je tak možné specifikovat i více uživatelů se stejným přihlašovacím jménem, pokud se nacházejí v různých skupinách. Při přihlášení je pak verifikace zadaného hesla aplikována na všechny uživatele s odpovídajícím jménem.

Index `rights` obsahuje přiřazení přístupových práv ke stránkám v systému Pico jednotlivým uživatelům či skupinám. Stránky se adresují pomocí umístění, na kterém se nacházejí. Pokud je na konci cesty uveden znak `/`, definované oprávnění je aplikováno rekurzivně na všechny stránky v tomto adresáři.

Informace o přihlášeném uživateli jsou do proměnné `$_SESSION` ukládány pod indexem unikátního identifikátoru aktuálního klienta, což je hash ze spojení řetězců IP adresy klienta, HTTP hlavičky User-Agent, jména prováděcího skriptu, aktuálního identifikátoru relace a slova `pico`. Tím je zajištěno, že i pokud bude identifikátor *session* přihlášeného uživatele kompromitován, útočník nebude přihlášen, pokud se mu nepovede simulovat stejný otisk klienta jako přihlášený uživatel. Jde o techniku zabezpečení, jejíž nevýhodou je, že uživatel nebude moci zůstat přihlášen v případě, že při každém požadavku změní IP adresu (např. použije-li anonymizační proxy).

Při analýze tohoto pluginu byl zjištěn nedostatek, který v současné verzi umožňuje přistoupit k chráněnému obsahu i bez přihlášení. Chyba spočívá v tom, že plugin vyhodnocuje přístupová oprávnění ke stránkám na základě parametru `$url` získaného z události `onRequestUrl`, což je obsah GET parametru, na základě něhož systém Pico určí cestu k souboru stránky pro zobrazení (popsáno v 1.1.3). Tento parametr před předáním do události `onRequestUrl` není převáděn do podoby, která by zaručila jeho jedinečnost pro zobrazení konkrétní stránky. Může tak existovat více různých hodnot tohoto parametru, které povedou k zobrazení jedné stránky. Například `/?page` i `/?./page` povedou k zobrazení stránky `page.md`. Je-li v nastavení pluginu přístup k této stránce omezen, dojde k omezení jen na první zmíněné variantě URL. Tento problém [14] identifikuje jako CWE-41 – chybné rozlišení ekvivalence cesty.

Plugin je určen pro systém Pico ve verzi 1.0. Nativní verze pro verzi 2.0 je připravována [13], v době tvorby této práce však ještě není funkční – plugin stále využívá rozhraní pluginů verze 1.0 a tak je nadále spouštěn přes PicoDeprecated.

1.2.3 Shrnutí současného stavu

Obě existující řešení poskytují autentizaci prostřednictvím vlastních uživatelských účtů vedených v konfiguračním souboru pluginu. Ani jeden z dostupných autentizačních pluginů nemá nativní podporu pro nejnovější verzi systému Pico – verzi 2.0. Pro použití těchto pluginů s touto verzí musí být aktivován dodatečný plugin PicoDeprecated, který zajistí kompatibilitu rozhraní starších pluginů. Ani v tomto režimu kompatibility však bez úprav ve zdrojovém kódu nepracují správně.

Ve verzích systému, pro které byly vytvořeny fungují správně, nicméně neimplementují ochrany proti běžným útokům na webové autentizační mechanismy [15]. Na oba pluginy je tak možné provést tzv. útok hrubou silou, protože neomezují počet pokusů na neplatné přihlášení. Není implementována ani ochrana proti útoku *Cross-site Request Forgery*. První z popisovaných pluginů je navíc náchylný k útoku využívajícím *Session fixation*. U druhého pluginu je možné obejít autorizaci. Při návrhu nového řešení proto bylo dbáno na to, aby implementace neobsahovala stejné nedostatky.

1.3 OAuth 2.0

OAuth 2.0 je autorizační protokol, který umožňuje aplikaci třetí strany omezeně přistupovat k HTTP službě poskytující informace. Jeden ze způsobů zajištění tohoto přístupu je získání souhlasné interakce od uživatele oprávněného k této službě přistupovat. Přihlašovací údaje uživatele nejsou při tomto procesu viditelné pro aplikaci třetí strany [16].

V této sekci bude nejdříve popsán protokol OAuth 2.0 a následně možnosti jeho využití pro realizaci jednotného přihlášení.

1.3.1 Definice rolí

Pro další popis protokolu je nezbytné uvést popis rolí účastníků figurujících v procesu autorizace. Tyto role definuje specifikace protokolu [16]. Definice používá termín zdroj (*resource*), což je chráněný objekt, ke kterému je přistupováno. Typicky jde o data (např. dokumenty, fotografie, informace o uživateli), služby (např. vložení, editace či mazání záznamů) nebo libovolné jiné zdroje vyžadující kontrolu práv [17].

Vlastník zdroje (*Resource owner*) Subjekt s právem přístupu ke zdroji. Obvykle jde o uživatele, který přistupuje prostřednictvím klientské aplikace k serveru se zdroji.

Server se zdrojem (*Resource server*) Server, ze kterého jsou přístupné chráněné zdroje jako odpovědi na požadavky obsahující přístupové tokeny.

Klient (*Client*) Klientská aplikace, která je se souhlasem vlastníka zdroje delegována k odesílání požadavků na chráněný zdroj.

Autorizační server (*Authorization server*) Server určený k výdeji přístupových tokenů klientovi poté, co je vlastník zdroje autentizován a autorizován.

1.3.2 Obecný princip protokolu

Před tím, než klient začne s využitím protokolu přistupovat k serveru se zdroji, musí být zaregistrován u autorizačního serveru. Tato registrace je zpravidla provedena vývojářem klientské aplikace a často je realizována vyplněním HTTP formuláře poskytnutého autorizačním serverem. Uváděné údaje o aplikaci zahrnují URI, na kterou budou přesměrovávány odpovědi autorizačního serveru, typ klienta a případné další informace jako název a popis. Typ klienta může být důvěryhodný – je schopen bezpečného uložení svých přihlašovacích údajů k autorizačnímu serveru (např. webová aplikace) a veřejný – (např. nativní aplikace spuštěna na zařízení vlastníka zdroje). Po registraci klient obdrží unikátní řetězec, který je veřejný a bude jej identifikovat u autorizačního serveru. Pokud je klient důvěryhodný, obdrží navíc i tajný údaj (např. heslo, nebo pár klíčů), kterým se bude autentizovat [16].

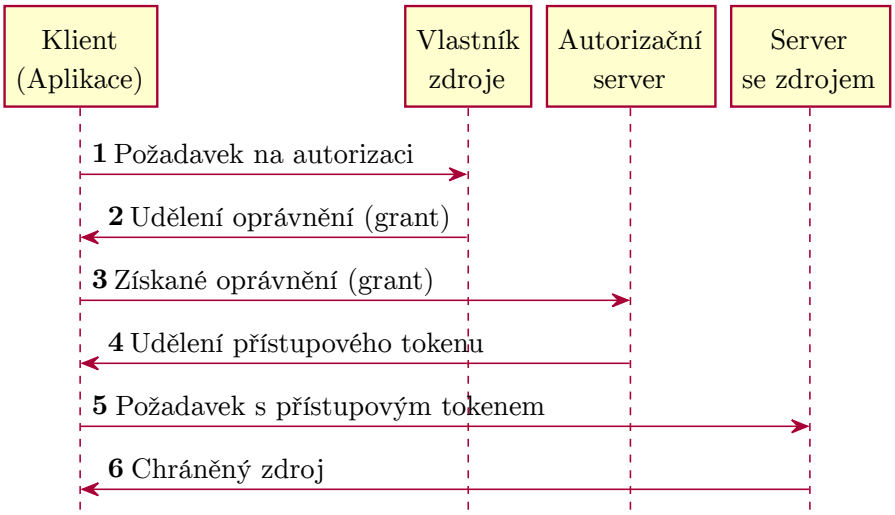
Nyní již klient může zahájit přístup k chráněnému zdroji. Jde o sekvenci událostí vyobrazenou na obrázku 1.2. Klient si vyžádá autorizaci od vlastníka zdroje (1). To je obvykle realizováno prostřednictvím autorizačního serveru. Klient je vlastníkem zdroje autorizován k přístupu ke zdroji (2). Obdržené oprávnění může mít různou podobu v závislosti na použitém způsobu vyžádání autorizace (*grant*) – specifikace definuje 4 možné. Klient pošle požadavek na autorizační server pro vydání přístupového tokenu (3). Autorizační server provede validaci požadavku a v případě úspěchu vydá klientovi přístupový token (4). Klient provede požadavek na server se zdrojem a uvede získaný přístupový token (5). Server se zdrojem validuje přístupový token a poskytne klientovi chráněný zdroj (6). Přesný způsob validace tokenu specifikace nepopisuje, typicky je provedena dotazem na autorizační server [16].

1.3.3 Způsob autorizace vlastníkem zdroje

Protokol umožňuje různé způsoby autorizace klienta vlastníkem zdroje a následného získání přístupového tokenu. Specifikace definuje tyto způsoby [16]:

Autorizační kód (*Authorization code grant*) Přístupový token je vydán na základě autorizačního kódu. Vhodný způsob pro důvěryhodné aplikace.

Implicitní (*Implicit grant*) Určen pro veřejné klienty (např. aplikace v Javascriptu spuštěná v prohlížeči). Neprovádí autentizaci klienta a přístupový token je viditelný pro vlastníka zdroje a případné další aplikace spuštěné v prohlížeči.



Obrázek 1.2: Abstraktní sekvence událostí OAuth 2.0 [16]

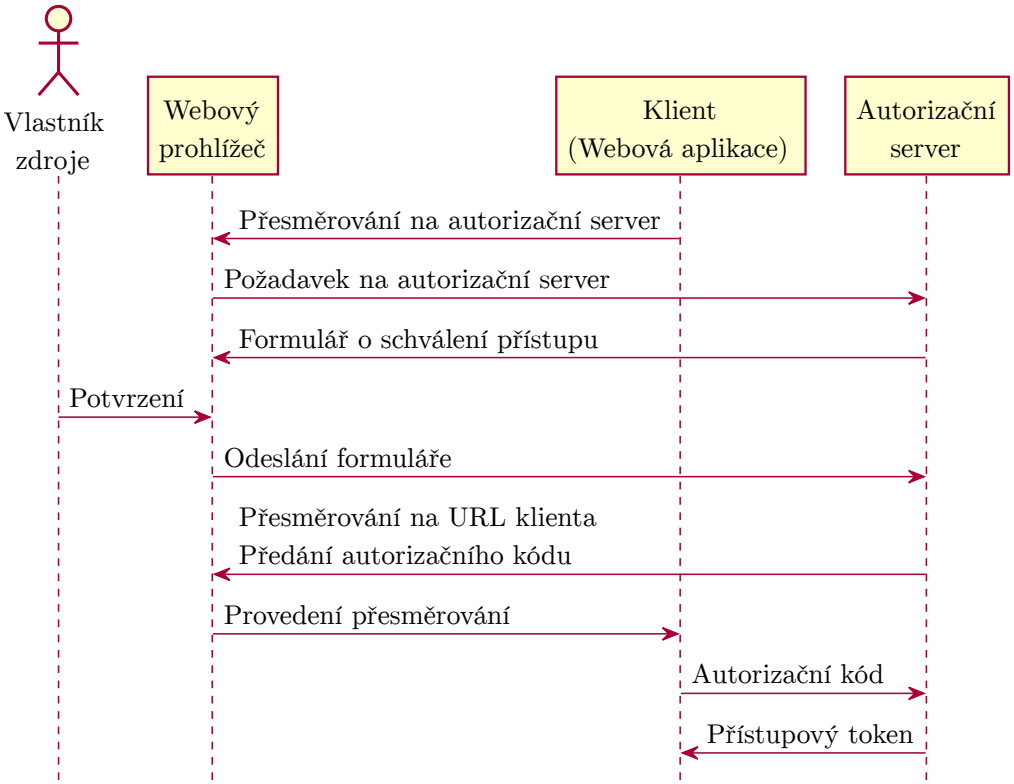
Přihlašovací údaje vlastníka zdroje (*Resource Owner Password Credentials Grant*) Vlastník zdroje zadává přihlašovací údaje přímo do klienta.

Přihlašovací údaje klienta (*Client Credentials Grant*) Určeno pro přístup ke zdroji jménem klienta.

Pro aplikaci běžící na webovém serveru nejvíce vyhovuje způsob autorizačního kódu. Sekvence událostí je při použití tohoto způsobu znázorněna na obrázku 1.3. Jde o konkrétní příklad, kde vlastník zdroje je uživatel přistupující ke klientovi přes webový prohlížeč. V průběhu procesu je uživatel požádán o autorizaci klienta pro přístup ke zdroji. Pokud uživatel na autorizačním serveru ještě není autentizován, zobrazí se nejdříve okno pro přihlášení a až poté dotaz na autorizaci. Ten typicky obsahuje název aplikace a seznam oprávnění, která si klientská aplikace nárokuje. Po potvrzení dotazu vrátí autorizační server v URL parametru autorizační kód, který klientská aplikace následně u autorizačního serveru zamění za přístupový token. Při této výměně se klientská aplikace autentizuje svým unikátním identifikátorem a heslem. Po získání přístupového tokenu klient přistupuje ke zdroji stejně jako na obrázku 1.2. Výhodou tohoto typu je, že získaný přístupový token není zasílán přes webový prohlížeč uživatele, nemůže tak dojít k jeho zcizení (např. zákeřným skriptem), jako je tomu u implicitního typu autorizace.

1.3.4 Bezpečnost protokolu

Bezpečnost protokolu OAuth 2.0 závisí na mnoha faktorech. Prvním předpokladem je správná implementace klienta a autorizačního serveru. Specifikace protokolu obsahuje samostatnou kapitolu s bezpečnostními doporuče-



Obrázek 1.3: Authorization code grant [18]

ními. K bezpečnostnímu incidentu však může dojít i v případě, že implementace protokolu vyhovuje nutným požadavkům specifikace. V následujících odstavcích jsou uvedeny jen subjektivně nejběžnější způsoby útoků. Více možných bezpečnostních rizik při použití protokolu je uvedeno v [16] a [19].

Specifikace vyžaduje užití TLS pro jakoukoliv komunikaci obsahující přístupový token a pro jakýkoliv požadavek odeslaný autorizačnímu serveru na endpoint pro autorizaci nebo výdej tokenů. Vydává však pouze doporučení nasazení TLS na klientovo URI pro přesměrování, kam autorizační server zasílá autorizační kód (*Authorization code grant*) nebo přístupový token (*Implicit grant*). Jako důvod uvádí, že v době tvorby protokolu by byl požadavek TLS pro vývojáře klientských aplikací příliš vysoký.

V případě odposlechnutí přístupového tokenu odeslaného po nezabezpečené transportní vrstvě však potenciální útočník okamžitě získá přístup ke chráněnému zdroji. Pro získání přístupu při odposlechnutí autorizačního kódu musí útočník navíc odposlechnutou odpověď blokovat, aby zabránil klientovi v použití tohoto kódu, což by jej učinilo neplatným – specifikace vyžaduje použití každého kódu maximálně jednou. Následně útočník může odposlechnutý kód vhodně sestaveným požadavkem odeslat do klientské aplikace a přistoupit tak přes ní ke chráněným zdrojům původního uživatele [19].

Další technikou použitelnou pro útok na proces autorizace je *Cross-site Request Forgery*. Při přesměrování na autorizační server by klient měl do cílové URI přidat parametr `state`, nastavený na dostatečně náhodnou neodhadnutelnou hodnotu. Autorizační server má v takovém případě povinnost vrátit nezměněnou hodnotu tohoto parametru ve své odpovědi na klientovo URI pro přesměrování. Klient tak získává možnost ověřit, zda požadavek opravdu odeslal uživatel (srovnáním hodnoty parametru z odpovědi s dříve vygenerovanou hodnotou). V případě neshody jde o požadavek z nelegitimního zdroje (technika útoku CSRF). Specifikace protokolu tento parametr doporučuje použít, ale označuje jej jako nepovinný. Dále ale uvádí, že ochrana proti CSRF musí být v klientské aplikaci implementována, což dává možnost využít alternativní techniky ochrany, které však nemusí být tak efektivní [20].

Phishing je další technika, která může být použita pro zneužití tohoto protokolu. Útočník může různými technikami, jako je ovlivnění DNS cache [19] nebo využití XSS (*Cross-site Scripting*) zranitelnosti v klientské aplikaci, přimět vlastníka zdroje navštívit zákeřnou verzi autorizačního dialogu. Pokud ji vlastník zdroje nebude schopen odhalit, může zadat své přihlašovací údaje, ke kterým tak získá přístup útočník [16].

1.3.5 Použití pro autentizaci

OAuth je autorizační protokol, jehož použitím lze autentizaci realizovat. Uživatel poskytne klientské aplikaci prostřednictvím protokolu přístup ke své identitě v aplikaci, přes kterou se chce přihlásit. Informace o identitě jsou zpřístupněny jako chráněný zdroj a obsahují informace o uživateli, který autorizoval vydání přístupového tokenu. Následně dochází k předpokladu, že pokud byl klient schopen získat informace o uživateli, může jej autentizovat. K tomu je typicky použit způsob autorizace autorizačním kódem [21].

Výhodou této realizace je, že uživatel může klientské aplikaci kromě informace o své identitě zároveň umožnit přístup i k jiným chráněným zdrojům. Nevýhodou je, že specifikace protokolu nevyžaduje, aby byl získaný autorizační kód platný pouze u klienta, který inicioval autorizaci. Pokud tak má útočník možnost zaregistrovat si vlastní klientskou aplikaci a podaří se mu přimět uživatele tuto zákeřnou aplikaci autorizovat, útočník může získat uživatelův autorizační kód, který může využít pro impersonaci uživatele v legitimní klientské aplikaci. Autorizační servery by tak měly provádět kontrolu původu autorizačního kódu [21].

Dalším nedostatkem je nadefinovaný formát chráněného endpointu s informacemi o uživateli. Každý server se zdroji může poskytovat různé typy informací dostupné pod různými označeními. Klientská aplikace musí mít tyto parametry předdefinované, aby mohla získat potřebné informace.

Zmíněné nevýhody řeší protokol OpenID Connect, nadstavba nad OAuth 2.0 určená pro autentizaci. K předání identity využívá tzv. identifikační token s přesně definovanou strukturou, který je předán jako JWT (JSON Web To-

ken) a může využít JWS (JSON Web Signature) pro podpis a JWE (JSON Web Encryption) pro šifrování obsahu. Dále protokol specifikuje strukturu dokumentu pro objevení (*Discovery document*), který poskytovatel identity může použít pro zveřejnění všech parametrů nutných pro autentizaci [22].

1.4 Stanovení cíle práce

Výsledné rozšíření bude implementováno formou pluginu. Při jiném způsobu realizace rozšíření (jako je například úprava stávajícího systému) by vznikly problémy s jeho údržbou v případě nových vydání systému Pico a se způsobem instalace pro stávající uživatele systému Pico. Navrhovaný plugin bude určen pro doposud nejnovější verzi systému Pico – verzi 2.0. Tuto verzi jsem zvolil, protože v oblasti tvorby pluginů uvádí oproti předchozím verzím několik vylepšení, které jsem popsal v sekci 1.1.7. Na základě provedené analýzy stávajících řešení a požadavků vyplývajících ze zadání práce jsem stanovil následující souhrn požadavků na výsledný plugin.

1.4.1 Funkční požadavky

- Možnost přihlášení vlastními uživatelskými účty.
- Možnost přihlášení prostřednictvím protokolu OAuth 2.0 způsobem popsaným v 1.3.5.
- Konfigurovatelné omezení přístupu ke stránkám v systému Pico na základě práv přihlášeného uživatele.
- Jednoduchý způsob umožnění přístupu k vybraným stránkám až po zadání sdíleného hesla.
- Konfigurace všech zmíněných vlastností prostřednictvím konfiguračních souborů.

1.4.2 Nefunkční požadavky

- Datová vrstva realizovaná prostřednictvím souborů prostého textu. Uvedení závislosti na databázové technologii třetí strany by nebylo v souladu s filozofií minimalistického systému Pico [1]. Plugin musí mít co nejnižší požadavky na systém, aby jej bylo možné použít i na webových serverech s omezeným počtem PHP rozšíření a aby byl snadno přenositelný.
- Rozšiřitelnost pluginu o další autentizační a autorizační metody.
- Bezpečnost systému a ochrana před nejběžnějšími útoky na webové aplikace.

Kapitola 2

Návrh

Při návrhu pluginu jsem identifikoval 4 základní skupiny funkcionalit, které jsem vyčlenil do samostatných modulů. Ty budou na sobě nezávislé, každý bude mít vlastní konfiguraci a uživatelé pluginu budou mít možnost v závislosti na jejich konkrétních požadavcích aktivovat jen takovou podmnožinu vlastností, kterou potřebují. Modulárním návrhem pluginu bude navíc zajištěna rozšiřitelnost o další vlastnosti. Budou implementovány následující:

- Modul pro autentizaci uživatelskými účty vedenými pluginem.

  Uživatelské účty budou podobně jako v současných řešeních uchovávány v textovém konfiguračním souboru, aby šlo jednoduchý systém Pico i nadále používat bez nutnosti externí relační databáze. Bude implementována i možnost návštěvníků registrovat si nový účet, možnost přihlášených uživatelů provést změnu hesla a možnost zaslání odkazu pro nastavení nového hesla na email.

- Modul pro autentizaci prostřednictvím služby s podporou protokolu OAuth 2.0.

  Návštěvníci budou mít možnost se do systému Pico přihlásit způsobem jednotného přihlášení. Uživatel pluginu může snadným způsobem v příslušném konfiguračním souboru definovat libovolné množství služeb, kterými se návštěvníci mohou přihlašovat. Dále může nastavit mapování atributů získaných od poskytovatele identity na atributy uživatele v pluginu (např. pro zobrazení profilové fotografie).

- Modul pro omezení přístupu na základě práv přihlášeného uživatele.

  Bude umožňovat definovat přístupová pravidla pro jednotlivé stránky. Pravidlo bude obsahovat buď seznam uživatelů nebo skupin, kterým má být umožněn přístup. U každého bude možné také specifikovat, zda se má aplikovat rekurzivně na všechny podstránky, či nikoliv.

- Modul pro zamknutí stránky jednoduchým heslem.

  Jde o jednoduchý způsob pro umožnění přístupu pouze skupině subjektů se znalostí sdíleného hesla. Tento způsob lze využít v případě, že nároky na utajení chráněného obsahu nejsou vysoké a využití restrikce na základě uživatelských účtů by nebylo rentabilní.

2.1 Architektura pluginu

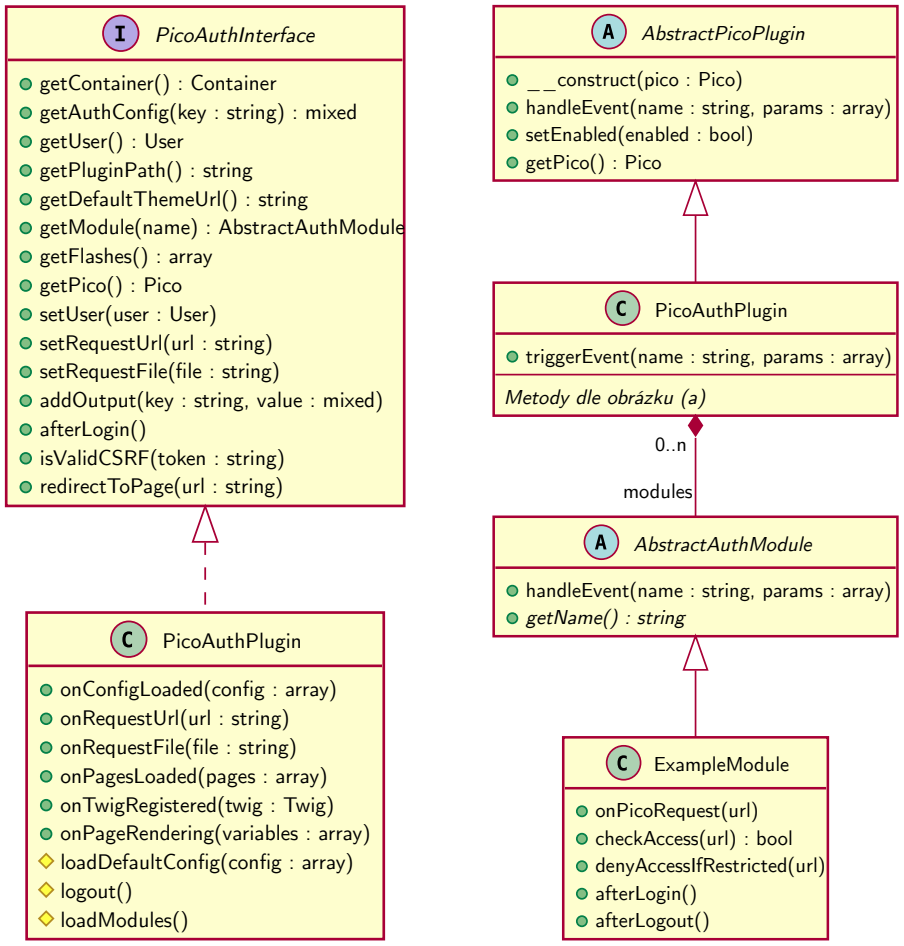
Pluginy v systému Pico jsou potomci třídy `AbstractPicoPlugin`, která je součástí každé instalace systému Pico. Poskytuje základní operace jako aktivaci a deaktivaci pluginu a přístup k instanci hlavní třídy systému Pico. Především však obsahuje metodu `handleEvent()`, která provádí volání registrovaných metod na instanci pluginu jako reakci na události přicházející ze systému Pico.

Hlavní třídou pluginu bude `PicoAuthPlugin`, která bude obsahovat implementace metod spouštěných pluginem. Ačkoliv Pico 2.0 nabízí 28 různých událostí [2], při návrhu jsem určil pouze 6 nutných pro realizaci všech požadavků na plugin. Jejich seznam je na diagramu 2.1a.

2.1.1 Moduly pluginu

Návrh jednotlivých modulů bude realizován obdobným způsobem, jako pluginy v systému Pico. Hlavní třída pluginu bude obsahovat pole instancí aktivních modulů, které budou reagovat na odesílané události. Tento způsob považuji za ideální, protože neklade požadavky na to, které metody musí být v modulu implementovány, což je závislé až na jeho účelu. Typicky půjde o moduly autentizační a autorizační. Cílem autentizačních modulů bude ověření identity uživatele. Po úspěšném dokončení autentizačního procesu konkrétním modulem dojde k předání informací o uživateli hlavní třídě `PicoAuthPlugin`, která následně zajistí uložení této informace do PHP *session* a tím dojde k přihlášení. Předání informací bude realizováno prostřednictvím instance třídy `User`, která bude do *session* ukládána v serializovaném formátu. Cílem autorizačních modulů je regulace přístupu návštěvníků k jednotlivým stránkám dostupným v systému Pico, typicky na základě oprávnění přihlášeného uživatele.

Seznam dostupných událostí, které mohou moduly pluginu využít je vypsán v třídě `ExampleModule` na diagramu 2.1b. Všechny autentizační a autorizační moduly, které budou implementovány, budou ale vyžadovat i zpětnou interakci s hlavní třídou pluginu. Sadu metod, která bude pro tento účel modulům k dispozici, jsem vyčlenil do rozhraní `PicoAuthInterface` (vyobrazeno na diagramu 2.1a). V budoucím vývoji pluginu by specifikace tohoto rozhraní měla zůstat neměnná, aby byla zaručena kompatibilita s existujícími moduly. Uvedením tohoto rozhraní nedojde k vytvoření závislosti na konkrétní verzi hlavní třídy pluginu a jednotlivé moduly budou také lépe testovatelné.



(a) Třída `PicoAuthPlugin` s rozhraním dostupným pro moduly

(b) Vztah třídy `PicoAuthPlugin` a ukázkového modulu

Obrázek 2.1: Zjednodušený třídní diagram hlavní třídy pluginu

Pro správu závislostí u jednotlivých pluginů bude použit tzv. poskytovatel závislostí (*dependency container*), což je třída zodpovědná za sestavení stromu závislostí za použití vhodných implementací, zvolených v jeho nastavení. U povinných závislostí tříd bude používán typ vkládání závislostí konstruktorem [23]. Většinu součástí pluginu tak bude možné deklarativně zaměnit za alternativní implementace v příslušném konfiguračním souboru.

Aktivaci jednotlivých modulů pluginu musí být uživatel schopen provést deklarativně v konfiguračním souboru. Jedním z hlavních cílů členění funkcionalit pluginu do modulů je, aby si uživatel mohl zvolit pouze takovou podmnožinu vlastností, kterou od pluginu potřebuje a neměl ve své instalaci zbytečné množství aktivních funkcí, které neplánuje využívat.

2.1.2 Konfigurace pluginu

Hlavní část konfigurace pluginu bude umístěna v konfiguračním souboru `config.yml` systému Pico, který umožňuje specifikovat i konfigurační parametry nainstalovaných pluginů. Tento soubor bude obsahovat všechna obecná nastavení, jako je právě definice seznamu aktivovaných modulů. Příklad je uveden v ukázce D.2. Pro zajištění větší přehlednosti bude konfigurace jednotlivých modulů vyčleněna do samostatných konfiguračních souborů. Aby byla zajištěna konzistence formátů, budou i tyto konfigurační soubory ve formátu YAML.

V závislosti na konfiguraci bude plugin do některých z těchto konfiguračních souborů muset provádět i zápis. Detailnějším popisem návrhových rozhodnutí týkajících se datové vrstvy se zabývá následující sekce.

2.2 Datová vrstva

Všechna data pluginu budou ukládána do textových souborů ve formátu YAML. Uvažoval jsem i o možnosti řešení prostřednictvím bez-serverové relační databáze SQLite 3. Nebylo by tak nutné před použitím pluginu instalovat či konfigurovat externí databázový server. Možnost jejího použití je však podmíněna instalovaným a aktivovaným rozšířením PHP `sqlite3` [24], nemusí tak být k dispozici na všech webových serverech. Nelze předpokládat, že každý uživatel pluginu by měl na svém webovém serveru dostatečná práva k jejímu doinstalování. Při použití této technologie by byla data zapisována do jednoho binárního souboru, který by nebyl bez externích nástrojů dobře editovatelný [25]. Plugin by tak musel uživateli zpřístupnit ještě uživatelské rozhraní, kde by bylo možné všechna data snadno editovat.

Vzhledem k tomu, že všechny úpravy (konfigurace i obsahu stránek) se v systému Pico provádějí editací souborů prostého textu, volba stejného způsobu úložiště umožní snadnou editaci, na kterou jsou uživatelé systému Pico zvyklí. Se zmíněnou možností použití relační databáze jsem však již při návrhu počítal jako s alternativou – plugin tak umožňuje záměnu způsobu ukládání dat. Toho je dosaženo abstrakcí datové vrstvy od jednotlivých modulů prostřednictvím rozhraních, navržených dle potřeb konkrétního modulu.

Nevýhodou textového úložiště by mohla být nižší rychlost. Běžně používané operační systémy však pro zrychlení čtecích operací ze souborů využívají tzv. vyrovnávací paměť stránek (*Page cache*), která v ideálním případě často používaný soubor přečte přímo z paměti [26]. Vzhledem k tomu, že navrhovaný plugin bude ve většině případů provádět pouze čtecí operace z malých textových souborů, není předpokládáno znatelné zpomalení odezvy systému. Zápis do souboru již může být omezující, protože musí být zajištěn exkluzivní přístup (a ostatní požadavky nemohou ve stejnou chvíli ze souboru číst ani do něj zapisovat) [26]. Při návrhu tak budou operace zápisu eliminovány na minimum a co nejvíce vyčleněny do samostatných souborů.

Oproti transakční databázi, jejíž transakce splňují vlastnosti ACID[1], protože využívají transakčního logu [27], budou vlastnosti navrhovaného datového úložiště omezeny. Konkrétní aspekty budou zmíněny v implementační sekci. Možnosti indexace jsou také limitovány – v případě souborů lze pro rychlý přístup k záznamu použít jméno souboru, které je zpravidla implementováno jako záznam v hash tabulce (např. *The dentry hash table* v OS Linux [26]) nebo jako indexy pole po převedení textového souboru do interní reprezentace. Tyto skutečnosti budou muset být zohledněny při implementaci.

Příklad návrhu datové vrstvy modulu pro omezování přístupu (ACL) je znázorněn na obrázku 2.2. Třída modulu `ACL` vyjadřuje závislost na implementaci `ACLStorageInterface`, což může být `ACLFileStorage`, třída implementující úložiště jako textovou databázi s podporou validace. Pro příklad byl zvolen modul `ACL`, protože jeho rozhraní je nejjednodušší. Návrh u všech ostatních modulů je obdobný, pouze s více metodami v rozhraní. Kompletní návrh je vyobrazen na diagramu tříd pro celý plugin, který je zahrnut v elektronické příloze práce.

2.2.1 Validace konfiguračních souborů

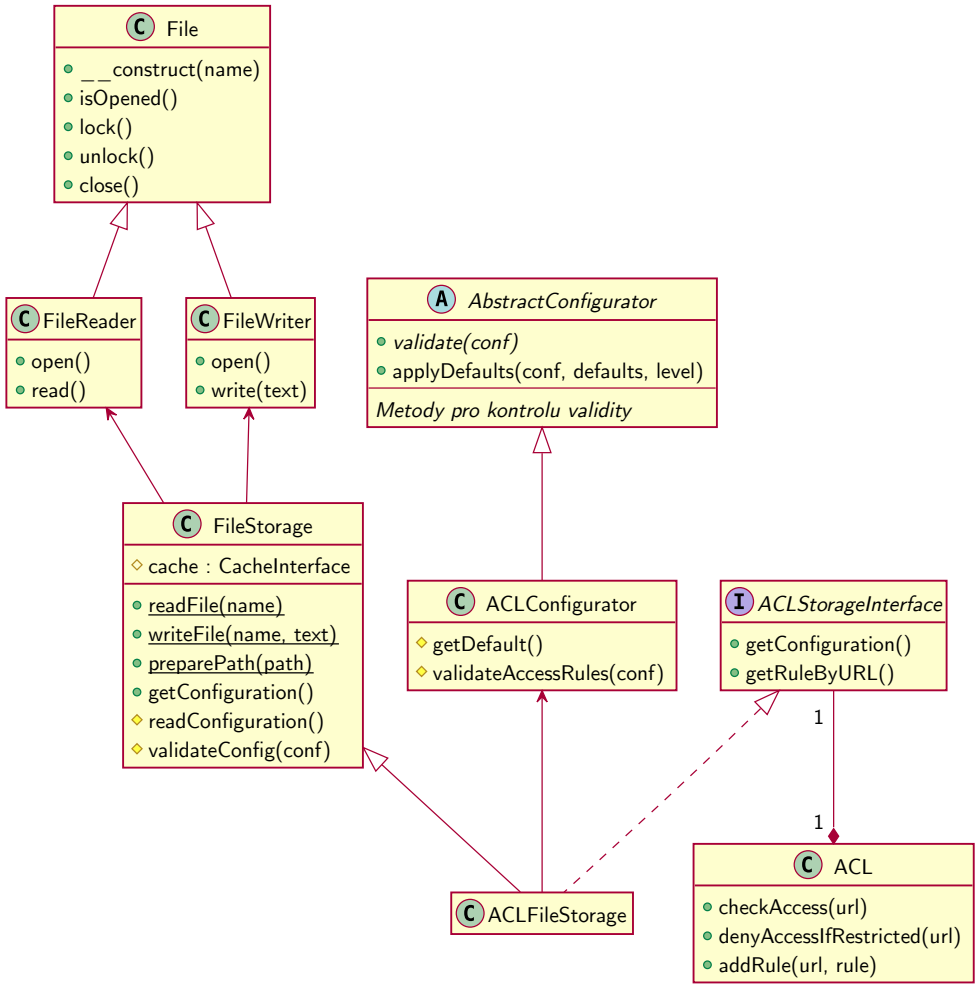
Editaci konfiguračních souborů provádí výhradně uživatel pluginu (správce konkrétní instalace systému Pico), od kterého se neočekává, že konfiguraci úmyslně vyplní chybnými vstupy. Nicméně i přesto musí být provedena validace všech hodnot konfiguračních souborů. V případě, že by například uživatel některou z hodnot neúmyslně vyplnil chybně a došlo by k její nesprávné interpretaci, výsledné nastavení by bylo jiné, než uživatel zamýšlel. V případě konfigurace funkcí pro omezení přístupu nebo nastavení bezpečnostních mechanismů, popisovaných dále v této kapitole, by to představovalo bezpečnostní riziko.

Pokud plugin detekuje chybu v nastavení (např. neočekávaný typ hodnoty nebo číselná hodnota mimo povolený rozsah), plugin na to musí reagovat chybovým stavem popsaným v 2.6.6.

2.2.2 Cache

Většina souborů, se kterými bude plugin pracovat, bude pouze pro čtení. Opakované parsování formátu YAML a provádění validace konfiguračních souborů je tak možné eliminovat pouze jedním provedením a následným čtením z cache. Po případné editaci konfigurace uživatelem bude obsah cache zaktualizován. Datová vrstva bude mít asociaci na libovolnou implementaci rozhraní `Psr\SimpleCache\CacheInterface` dle standardu PSR-16 [28]. Uživatel pluginu tak může zvolit a doinstalovat libovolnou implementaci v závislosti na možnostech jeho webového serveru. Její zavedení následně nastaví v konfiguračním souboru poskytovatele závislostí pro plugin. Efektivní volbou může být

[1] Atomicity, Consistency, Isolation, Durability

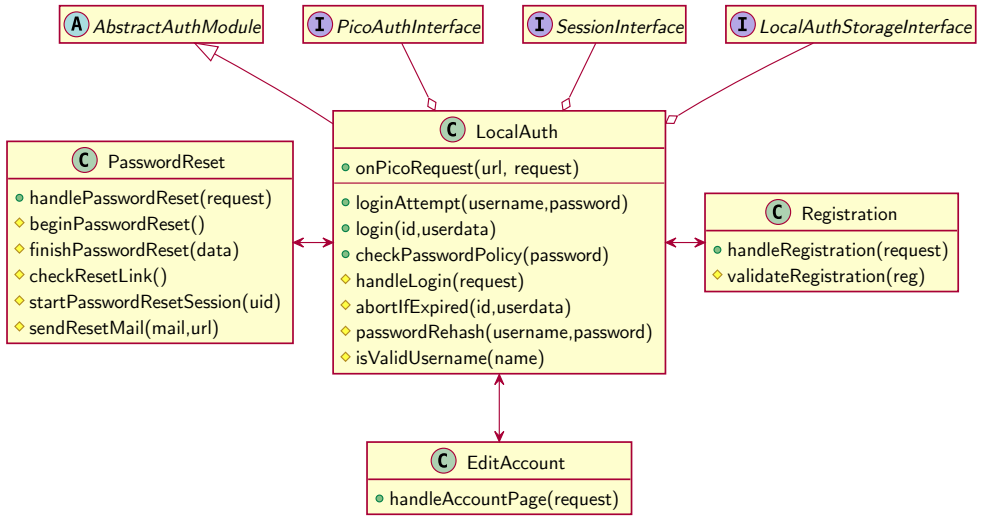


Obrázek 2.2: Zjednodušený diagram tříd datové vrstvy modulu ACL

například APCu cache – rozšíření pro PHP využívající rozsah sdílené paměti pro uložení cachovaných záznamů [29].

2.3 Lokální přihlášení

Plugin bude uživatelům umožňovat přihlášení pomocí vlastních uživatelských účtů. Autentizace bude založena na znalosti správné kombinace jména a hesla. Tato funkcionalita bude implementována v modulu nazvaném `LocalAuth`. Popisem jeho návrhu se zabývá tato sekce. Zjednodušený návrh tříd tohoto modulu je na obrázku 2.3.



Obrázek 2.3: Zjednodušený diagram tříd modulu LocalAuth

2.3.1 Správa uživatelských účtů

Uživatelské účty bude možné definovat v konfiguračním souboru modulu. Uživatelská jména budou indexy konfiguračního pole, příslušná hodnota bude obsahovat seznam atributů uživatelského účtu. Jediný povinný atribut bude hodnota `pwhash`, obsahující hash uživatelova hesla vzniklý aplikací některé z dostupných ověřených hashovacích funkcí pro odvozování klíčů.

Bcrypt je hashovací funkce založená na variaci dobře analyzované blokové šifry Blowfish. Jeho výpočetní náročnost lze specifikovat parametrem *cost*, jehož vliv je exponenciální. Vhodným nastavení tohoto parametru tak lze časově omezit proveditelnost útoku hrubou silou. Algoritmus také využívá kryptografickou sůl pro prevenci dohledání hesla ve vyhledávací tabulce [30].

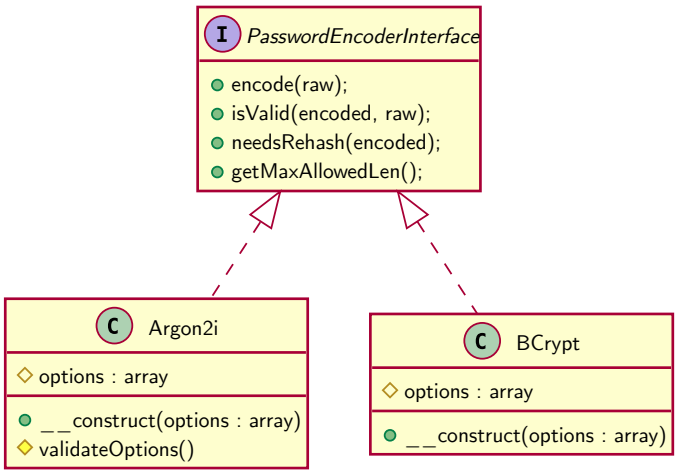
Je dostupný v API pro hashování hesel od PHP verze 5.5.0 [31]. Právě kvůli jeho nativní dostupnosti již od této verze PHP bude tento algoritmus v pluginu zvolen jako výchozí.

Argon2i je hashovací funkce doporučovaná na základě výhry v *The Password Hashing Competition* – veřejné soutěži, která si kladla za cíl nalézt standard pro bezpečné hashování hesel. Funkci lze parametrizovat specifikováním časové a paměťové náročnosti a stupně paralelismu [32].

Tento algoritmus má nativní podporu v PHP až od verze 7.2 [31], bude proto umožněn až jako alternativní volba pro uživatele pluginu s verzemi PHP, kde je podporován.

Volbu těchto funkcí lze provést buď na úrovni konfigurace modulu, nebo zvlášť u každého uživatele. Parametry jednotlivých hashovacích funkcí je možné specifikovat v konfiguraci poskytovatele závislostí. Zároveň je možné přidat další algoritmy implementací rozhraní `PasswordEncoderInterface` dle vyobrazení na obrázku 2.4.

Pokud bude při přihlášení uživatele zjištěno, že pro uložení reprezentace jeho hesla je použit algoritmus, který v době přihlášení není v nastavení pluginu zvolen jako výchozí, dojde k přehashování. Stejná situace nastane, i když je algoritmus stejný, ale některé jeho parametry náročnosti byly nastavením změněny.



Obrázek 2.4: Způsob implementace algoritmů pro hashování hesel

Jako jednu z nebytných vlastností při přihlašování uživatelským jménem a heslem jsem identifikoval možnost změnu hesla pro přihlášeného uživatele. Tuto možnost bude možné aktivovat v nastavení a plugin po její aktivaci umožní přihlášenému uživateli přístup k formuláři, kde může změnu hesla provést.

U každého uživatele bude možné specifikovat i další parametry, jako je seznam skupin (klíč `groups`) nebo sadu uživatelsky definovaných atributů (klíč `attributes`). Také bude k dispozici parametr `pwreset` jehož nastavením bude při prvním přihlášení uživatele vynucena změna hesla. To může být využito například v případě, že účet byl vytvořen manuálně s přednastaveným heslem a je nutné, aby bylo změněno. Kompletní příklad konfiguračního souboru je v příloze D.3.

Pokud jsou uživatelská data uložena v konfiguračním souboru modulu a dojde k jejich změně, jsou vyčleněna do samostatného datového souboru. Toto opatření bude implementováno proto, aby případné další změny uživatele vyžadovaly nutnost přepsání pouze jemu náležícího souboru, místo celého konfiguračního souboru modulu, ke kterému je přistupováno podstatně častěji.

Soubor s daty uživatele bude pojmenován jeho uživatelským jménem. Z tohoto důvodu budou na uživatelská jména kladeny požadavky pouze alfanumerických znaků. Uživatelská jména musí být také před uložením normalizována, aby byla zajištěna přenositelnost na souborový systém nerozlišující velikost znaků.

Způsob správy uživatelů editací konfiguračního souboru je přijatelný jen pro malé množství uživatelů. Například pokud správce systému vytvoří účet jen sobě pro přístup k neveřejným stránkám a neočekává v systému další uživatele. Nevýhoda spočívá v tom, že uživatel musí v konfiguraci poskytnout hash svého hesla, který si nejdříve musí vygenerovat voláním příslušné kryptografické funkce. Ačkoliv obě stávající řešení zkoumaná v analýze používají stejný způsob tvorby uživatelů, plugin by měl poskytnout i alternativní řešení. To může představovat registrace uživatelů.

2.3.2 Registrace

Pokud je funkce zapnutá, plugin umožní nepřihlášeným uživatelům přístup k registračnímu formuláři, kde si mohou založit účet. Pro registraci bude vyžadováno uvedení uživatelského jména, emailu a hesla. Po úspěšné registraci budou uživatelská data uložena do samostatného souboru. Všechny zadávané hodnoty musí podléhat přísným validačním pravidlům (délka a obsah zadávaného jména, korektnost emailové adresy, unikátnost identifikátorů a případně požadavky na minimální složitost hesla).

V případě automatizované registrace velkého množství účtů by mohlo dojít k vyčerpání prostředků (CWE-400: *Resource Exhaustion* [14]). Maximální počet registrací tak bude omezen konstantou v konfiguračním souboru a rychlost hromadné registrace omezena způsobem popsaným dále, v sekci 2.6.1.

2.3.3 Obnova hesla

Nelze vyloučit situaci, že některý z uživatelů své heslo zapomene. V takovém případě pak může situaci řešit se správcem stránky, který mu po ověření jeho identity nastaví jiné heslo nebo mu potřebná oprávnění nastaví na nově vytvořený účet a předchozí odstraní. V závislosti na počtu uživatelů je možné zvážit aktivaci funkce pro automatickou obnovu hesla prostřednictvím emailu. Uživatel tak získá možnost nechat si na email uvedený při registraci odeslat odkaz, který umožní nastavení nového hesla.

Značnou nevýhodou této možnosti je, že její bezpečnost závisí na bezpečnosti emailového protokolu. Protokol SMTP (Simple Mail Transfer Protocol), který doručení emailu zajišťuje, svým původním návrhem nezaručoval utajení přenášených informací. Tato možnost byla doplněna až dodatečně, rozšířením STARTTLS, které umožňuje využít TLS pro komunikaci mezi jednotlivými uzly na cestě od serveru odesílatele k serveru příjemce. Jeho použití však z důvodu kompatibility mezi jednotlivými emailovými servery většinou není

při doručování emailu vyžadováno. Při začátku komunikace je informace o jeho aktivaci odeslána po nezabezpečeném kanále, a je tak možné tuto komunikaci upravit tak, aby k jeho použití nedošlo. Často jsou akceptovány i neověřené certifikáty, což dává příležitost k odposlechu i šifrované komunikace (tzv. útok *Man-in-the-middle*). Bezpečnost emailu však nezávisí jen na transportní vrstvě – email může být přečten neoprávněným subjektem i kompromitací emailového serveru nebo schránky uživatele. Tyto i další bezpečnostní problémy emailové komunikace jsou podrobněji popsány v [33].

Předmět a formát obsahu odesílaného emailu bude možné upravit v konfiguraci. Odesílaný odkaz pro nastavení hesla bude obsahovat náhodný token, jehož platnost bude časově omezena konfigurovatelnou hodnotou. Po návštěvě odkazu s platným tokenem bude uživateli vytvořena časově omezená relace pro nastavení nového hesla a token bude pro další použití zneplatněn.

Volba způsobu odeslání emailu bude přenechána na uživateli pluginu. Ten v konfiguraci poskytovatele závislostí zvolí implementaci jednoduchého rozhraní `PicoAuth\Mail\MailerInterface` dle preferovaného způsobu pro odeslání emailu. Dokumentace pluginu poskytne ukázkové implementace pro často používané způsoby odeslání emailu.

2.4 SSO přihlášení

Plugin bude umožňovat možnost aktivace jednotného přihlášení. K tomu bude využit protokol OAuth 2.0 způsobem popsaným v sekci 1.3.5. Uživatel při přihlášení aplikaci autorizuje, ta získá autorizační token, který následně použije pro získání informací o uživateli na službě, přes kterou se přihlašuje. Na základě nich mu je vytvořena relace přihlášeného uživatele. Dle terminologie protokolu bude v textu používáno označení „autorizační server“, ačkoliv jeho použití v kontextu toho návrhu vede k autentizaci. Termín „poskytovatel“ je souhrnné označení pro autorizační server a server se zdroji poskytující informace o přihlášeném uživateli. Uživatel může mít při přihlášení na výběr z více možných poskytovatelů.

Protokol OAuth 2.0 nespecifikuje, jaké informace mají být poskytnuty na endpointu s informacemi o uživateli, jaký mají mít formát a na jaké URL adrese mají být dostupné. Aby bylo plugin možné použít s více poskytovateli, bylo nutné vyčlenit tyto variabilní faktory mimo zdrojový kód pluginu. To umožňuje existující PHP knihovna League OAuth 2.0 Client [34]. Té je možné poskytnout implementaci třídy `AbstractProvider` uzpůsobenou pro konkrétního poskytovatele. Implementace rozhraní `ResourceOwnerInterface` pak sjednocuje přístup k informacím o uživateli (vlastníkovi zdroje). Plugin tak může pracovat s jednotným rozhraním třídy `AbstractProvider` nezávisle na odlišnostech konkrétního autorizačního serveru. Knihovna navíc poskytuje implementace těchto tříd pro často používané poskytovatele [34], uživatel tak nemusí implementovat vlastní třídu, ale provede jen instalaci existující.

V konfiguračním souboru bude možné u každého autorizačního serveru specifikovat sadu atributů. Povinnými hodnotami bude identifikátor a klíč klientské aplikace získané při její registraci. Hodnota `provider` bude obsahovat plně kvalifikované jméno třídy poskytovatele. Pokud nebude uvedena, bude využita obecná implementace (`GenericProvider`), která bude vyžadovat specifikaci dalších parametrů, jako jsou URL adresy jednotlivých endpointů (pro zahájení autorizace, pro získání tokenu a pro získání informací o přihlášeném uživateli). Příklad obou způsobů konfigurace je uveden v příloze D.4.

Dále bude v konfiguraci poskytovatele k dispozici mapování atributů ze zdroje s informacemi o uživateli na atributy přihlášeného uživatele v systému Pico. Možným použitím je například získání URL adresy profilové fotografie od poskytovatele a její zobrazení u jména přihlášeného uživatele v systému Pico. Různí poskytovatelé budou mít typicky tuto informaci dostupnou pod různými názvy a někteří ji nemusí poskytovat vůbec. Mapování atributů umožní asociovat klíč v poli hodnot s informacemi o uživateli získaném od poskytovatele na název atributu přihlášeného uživatele v systému Pico. U každého poskytovatele bude možné specifikovat i výchozí hodnoty pro případ, že požadovaný atribut není k dispozici. V sekci výchozích hodnot bude specifikovatelné i výchozí členství ve skupinách, aby bylo možné provést autorizaci na základě přihlášení konkrétním poskytovatelem. Příklad popsaných možností je uveden v příloze D.4.

2.4.1 Přihlášení přes účet ČVUT

Ačkoli nejde o možnost vyžadovanou některým ze stanovených požadavků, jedním z aspektů při tvorbě zadání této práce byla možnost výsledný plugin použít pro umožnění přístupu ke stránkám v systému Pico pouze subjektům majícím platný účet ČVUT.

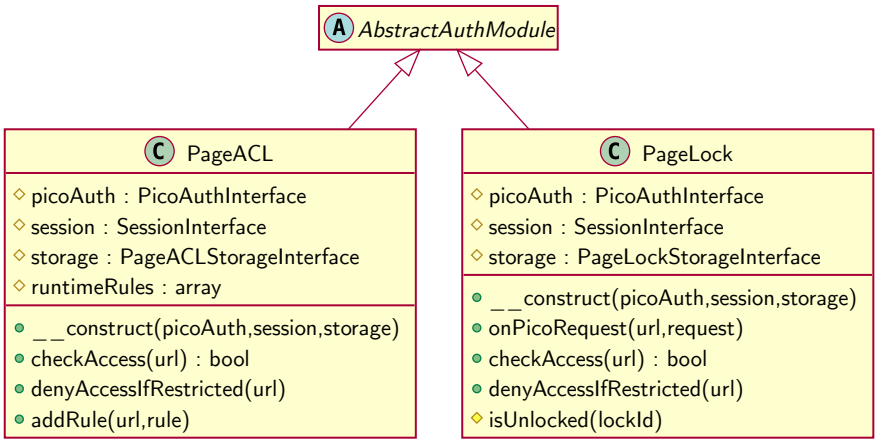
Na ČVUT je vyvíjen a provozován autorizační server Zuul OAAS (*OAuth 2.0 authorization server*). Jde o server s otevřeným zdrojovým kódem, založený na projektu Spring Security OAuth. Klientskou aplikaci u něj může zaregistrovat kdokoliv s účtem na ČVUT. K tomu je učena služba pro správu aplikací AppsManager [35]. Projekt Spring Security OAuth, na kterém je implementace tohoto serveru založena, implementuje nad rámec specifikace protokolu OAuth 2.0 [16] bezpečnostní opatření, které neumožňuje použít autorizační kód v jiné klientské aplikaci, než která proces autorizace iniciovala [36]. Při jeho použití tak nedochází k riziku popsanému v sekci 1.3.5.

Aby bylo výsledný plugin s tímto serverem možné snadno použít, kromě implementace samotného pluginu bude implementována i třída pro zmíněnou knihovnu League OAuth 2.0 Client, která bude obsahovat specifická nastavení pro použití zmíněné knihovny s autorizačním serverem ČVUT Zuul OAAS.

2.5 Omezení přístupu

Omezení přístupu ke stránkám dle práv přihlášeného uživatele bude realizováno autorizačním modulem PageACL na základě pravidel specifikovaných v souboru `PageACL.yml`. Každé pravidlo bude vztaženo na stránku v systému Pico. Ve výchozím stavu budou všechny stránky přístupné, přístup bude omezován specifikací pravidel s výčtem uživatelů a skupin s právem pro přístup k dané stránce. Pokud se na některou z takto konfigurovaných stránek pokusí přistoupit nepřihlášený uživatel, bude přesměrován na stránku s přihlášením. V případě úspěšného přihlášení bude přesměrován na původní požadovanou stránku. Má-li pro přístup nedostatečná oprávnění, bude zobrazena chybová stránka informující o odepření přístupu. Je-li pravidlo aplikováno na adresář, ve výchozím stavu bude aplikováno rekurzivně i na všechny podstránky. Toto nastavení bude možné upravit přidáním hodnoty `recursive: false` v parametrech konkrétního pravidla. Kompletní příklad konfigurace je uveden v D.5.

Druhý autorizační modul bude nezávislý na přihlášení uživatele a přístup k zabezpečeným stránkám umožní po zadání správného hesla ke stránce. Při pokusu o přístup ke stránce chráněné tímto způsobem se zobrazí textové pole pro zadání hesla k odemknutí stránky. Po zadání správné hodnoty bude příznak o jejím odemčení uložen do aktuální relace a při dalších požadavcích bude zobrazen již přímo obsah stránky. Tento způsob je možné použít v případech, kdy není nutné ověření identity přistupujících subjektů a časové nároky na zřízení jednotlivých účtů a nastavení odpovídajících práv by byly větší, než užitek, který by autorizace přinesla. Nastavení tohoto modulu bude v souboru `PageLock.yml`, kde bude uvedena definice jednotlivých zámků a jejich přiřazení ke stránkám v systému Pico. Pro uložení hesla ke stránkám je možné použít stejné hashovací funkce, je možné použít stejné hashovací funkce jako pro heslo autentizace. Výchozím nastavením je Bcrypt. Příklad konfigurace tohoto modulu je v příloze D.6.



Obrázek 2.5: Zjednodušený návrh tříd autorizačních modulů

2.6 Bezpečnostní mechanismy

Bezpečnost výsledného pluginu je nutné zohlednit již ve fázi návrhu. Plugin je navrhován pro obecné použití komunitou uživatelů systému Pico, proto je obtížné předem charakterizovat cílového uživatele a nároky na bezpečnost pro konkrétní použití. Stejně tak je problematické stanovit, jaká data se budou nacházet na stránkách systému Pico, ke kterým bude prostřednictvím navrhovaného pluginu omezován přístup, a jaké následky by měla jejich případná kompromitace. V následujících sekcích jsou popsány mechanismy, které budou implementovány pro zabezpečení pluginu.

2.6.1 Rate limiting

Jako podstatnou hrozbu funkce pro přihlášení lokálními uživatelskými účty jsem identifikoval tzv. útok hrubou silou (*brute force attack*). Jeho podstata spočívá systematickém provádění velkého množství pokusů o přihlášení s cílem uhádnutí správných přihlašovacích údajů uživatele a získání přístupu ke chráněným stránkám. Pro omezení rychlosti takto prováděného útoku a tím i snížení pravděpodobnosti jeho úspěchu v přijatelném čase bude stanoven limit na maximální počet chybných přihlášení. V kontextu navrhovaného pluginu bude pro tuto techniku používáno označení *rate limiting* (omezení počtu provedených akcí za daný časový úsek).

Protože bude tato technika v pluginu použita nejen na omezení chybných přihlášení, je pro vysvětlení jejího návrhu nejdříve nutné zavést obecná označení: kritická akce – jakákoliv operace podléhající omezením na počet provedení; typ akce – např. pokus o přihlášení, registrace účtu; entita – identifikovatelný objekt, na který je omezení aplikováno (např. uživatelský účet nebo IP adresa klienta).

Pro kombinace typu akce a entity bude možné v konfiguračním souboru specifikovat pravidlo pro omezení počtu akcí. Každé pravidlo musí obsahovat tyto 3 číselné parametry: maximální povolený počet akcí před blokací n_{max}, doba trvání blokace t_{bl} a doba od poslední akce, po které dojde k expiraci čítače akcí, t_{exp}. Pro každou kombinaci typu akce a entity bude vytvořen datový soubor, který bude pro každou entitu evidovat počet dosud provedených akcí a čas provedení poslední akce. Před každou kritickou akcí budou nejdříve tyto hodnoty přečteny a na jejich základě bude rozhodnuto buď o povolení nebo o blokaci akce. K povolení dojde, pouze pokud je počet dosud vykonaných akcí menší než povolený, nebo pokud doba od poslední akce již překročila dobu expirace čítače. Akce tak nebude možné provádět rychleji než

$$\min(\frac{n_{max}}{t_{bl}}, \frac{n_{max} - 1}{t_{exp}})$$

za jednotku času. Pro dobu expirace bude typicky platit $t_{exp} \geq t_{bl}$. Pokud bude v konfiguraci vynechána, použije se hodnota t_{bl}. Tento způsob jsem zvolil, protože vyžaduje uložení minimálního množství informací a není nutné

evidovat jednotlivé akce zvlášť, což by vedlo k problematické realizaci v textové databázi.

V případě blokace na IP adresu bude možné u každého pravidla specifikovat velikost podsítě, která bude pro účely omezování počtu akcí považována za jednu entitu. Hodnota bude nastavitelná jak pro adresy typu IPv4, tak IPv6. Toto nastavení je vhodné, pokud útočník použije pro provedení útoku souvislý rozsah adres. Nevýhodou může být, že příliš striktní nastavení způsobí, že uživatelé ze stejné sítě budou limit sdílet. Uživatel pluginu tak musí provést úpravu těchto parametrů dle konkrétního použití.

Výchozí nastavení pro všechny omezované akce je v tabulce 2.1. Dle doporučení [37] je omezení pro přihlašování aplikováno spíše na kratší časové úseky (zde 15 minut), aby byl snížen negativní dopad na uživatele a zároveň stále došlo k výraznému omezení rychlosti hádání hesel. Jakkoliv nastavený limit na uživatelské jméno je však zneužitelný pro útok odepření služby (*Denial of service*) [37], uživatel pluginu tak může zvážit jeho odstranění. Ve výchozím nastavení je pro omezení na IP adresu počítáno s 5× vyšším limitem než na jednoho uživatele.

Tabulka 2.1: Výchozí nastavení pro omezení akcí

Akce	Omezeno na	Max. rychlost	Max./24h
Chybný pokus o přihlášení	Uživ. jméno	10/15 min.	960
	IP adresa	50/15 min.	4800
Žádost o reset hesla	Email	2/24 hod.	2
	IP adresa	10/24 hod.	10
Registrace účtu	IP adresa	2/12 hod.	4
Pokus o odemknutí stránky	IP adresa	10/30 min.	480

2.6.2 Instalační modul

Jak bylo popsáno v sekci 1.1.2, ve výchozím stavu je adresářová struktura systému Pico celá umístěna ve veřejném prostoru webového serveru a zakázání přístupu ke konfiguračním souborům je závislé na nastavení webového serveru. Pokud nejde o server Apache, uživatel musí konfiguraci doplnit sám. Nelze spoléhat na to, že instalace systému Pico, do které bude plugin instalován bude konfigurována správně. Instalační modul se pokusí chybu v konfiguraci detekovat a zejména uživatele o rizicích chybné konfigurace informovat.

Instalační modul bude jako jediný aktivován již ve výchozí konfiguraci. Zpřístupní v systému Pico stránku `/PicoAuth`, na kterou bude uživatel nasměrován v instalační sekci dokumentace pluginu. Tato stránka provede v kontextu konkrétní instalace sytému Pico sekvenci požadavků na adresy, které musí být konfigurovány jako nepřístupné. V případě pozitivní odpovědi na

některý z požadavků zobrazí upozornění. Tato kontrola si však neklade za cíl ověření zcela bezchybné konfigurace, což by ve fázi instalace, kdy konfigurační soubory pluginu ještě neexistují, ani nebylo možné. Hlavním cílem je upozornit uživatele na tento problém a doporučit přesun vybraných adresářů do neveřejného prostoru webového serveru, což modul také zkontroluje. Uživatel má možnost doporučení instalátoru ignorovat, bude-li však plugin používán bez tohoto opatření, jde o slabinu identifikovanou jako CWE-219 (*Sensitive Data Under Web Root* [14]).

Po úvodní kontrole konfigurace instalátor nabídne uživateli volbu dostupných modulů pluginu a na základě výběru vygeneruje krátkou část konfigurace, kterou uživatel vloží do konfiguračního souboru systému Pico. Dojde tak k aktivaci zvolených funkcí a zároveň i k automatické deaktivaci instalačního modulu. Na závěrečné stránce instalátoru budou také uvedeny možné příklady konfigurace zvolených modulů.

2.6.3 Password policy

Ve výchozím stavu je na heslo zadávané při registraci nebo změně hesla kladen požadavek pouze na minimální délku 8 znaků, což je doporučení dle [38]. Uživatel pluginu má možnost na základě svých specifických požadavků minimální nároky na složitost hesla upravit. K tomu bude v pluginu k dispozici třída `PasswordPolicy`, jejíž konfigurací bude možné specifikovat kombinace požadavků jako je minimální počet velkých či malých písmen, číslic a speciálních znaků. Jednotlivá omezení uživatel definuje uvedením vybraných volání ověřovacích metod třídy `PasswordPolicy` v konfiguračním souboru správce závislostí. Má možnost specifikovat i alternativní implementaci rozhraní `PasswordPolicyInterface` a provádět další kontroly hesla, jako např. kontrolu, zda nejde o slovníkové heslo vyhledáním v dostupných databázích.

2.6.4 Vyhodnocení ekvivalentních cest

Systém Pico poskytuje pluginům událost `onRequest` obsahující argument určený pro identifikaci stránky, která bude zobrazena. Jde v podstatě o cestu k souboru stránky upravenou způsobem popsaným v 1.1.3. Tuto hodnotu nelze použít pro vyhodnocení autorizačních pravidel, protože existuje mnoho ekvivalentních zápisů, které povedou na zobrazení jedné stránky. Ty jsou navíc závislé na operačním systému webového serveru. Například jde o libovolné množství vhodně umístěných sekvencí znaků pro aktuální adresář (`./page` nebo `././page`), na systému Windows může jít o dodatečnou tečku za konci adresářů, názvy s různou velikostí znaků a zkrácené, tzv. „8.3 názvy“. Problém je podrobněji popsán v [14] pod označením CWE-41.

Navrhované řešení je využít událost `onRequestFile`, která poskytuje absolutní cestu souboru stránky, který bude nahráván. Tuto cestu plugin převede do jednotné reprezentace funkcí `realpath()` standardně dostupnou v PHP [39].

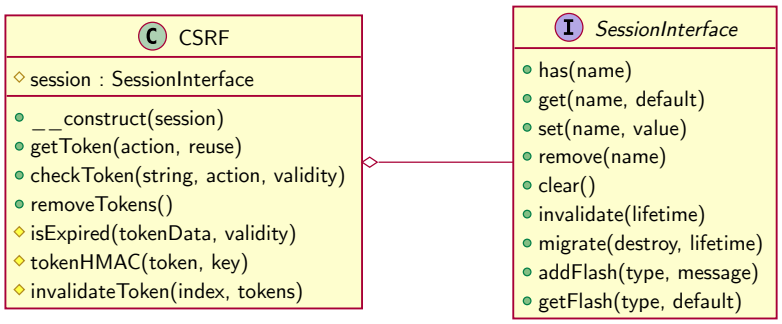
Ze získané cesty bude zpětně odvozen parametr URL, který již může být použit pro vyhodnocení autorizačních pravidel. Pokud soubor neexistuje, až tehdy bude pro autorizaci použit argument získaný z události `onRequest`, což umožní řízení přístupu i ke stránkám, které jsou generované dynamicky jinými pluginy a nemají ekvivalentní názvy.

2.6.5 Cross site request forgery

Cross site request forgery (zkráceně CSRF) je technika útoku, která umožňuje útočníkovi, typicky za využití škodlivého kódu, odeslat v kontextu relace přihlášeného uživatele nezamýšlený požadavek. Škodlivý kód může být buď HTML (např. `img` element obsažený v emailu odeslaném útočníkem), nebo JavaScript vložený útočníkem do webové stránky, kterou útočník přiměje uživatele navštívit [40].

Ochrana proti tomuto typu útoku bude v pluginu implementována vybavením všech formulářů náhodným synchronizačním tokenem. Při validaci formuláře bude nejdříve provedeno srovnání, zda nejde o token s expirovanou dobou platnosti a zda se jeho hodnota shoduje s hodnotou nastavenou již při generování formuláře. Aby pak bylo možné útok provést, útočník by musel předem uhádnout hodnotu náhodného tokenu ve formuláři oběti. Délka tokenu bude zvolena na dostatečně velkou hodnotu, aby uhádnutí nebylo pravděpodobné.

Implementovaná třída pro správu synchronizačních tokenů bude umožňovat generování tokenů s možností jejich přiřazení ke konkrétnímu formuláři (i v případě kompromitace takového tokenu nebude platný pro odeslání jiných formulářů, než pro který byl vystaven), možností kontroly jejich časové platnosti a možností jejich zneplatnění po úspěšném odeslání. Návrh této třídy je na obrázku 2.6.



Obrázek 2.6: Návrh třídy pro správu synchronizačních tokenů

Pokud by byl token vkládaný do formulářů statický, útočník by jej teoreticky mohl od uživatele získat technikou útoku zvanou BREACH – metodou útoku postranním kanálem na HTTPS komunikaci využívající analýzu délky odpovědí, kompresi na úrovni HTTP a uživatelský vstup reflektovaný v odpovědi k postupnému odhalení tajné statické informace obsažené v odpově-

dích [41]. Aby token nebyl statický, navrhovaná třída pro správu tokenů vydá při každém volání metody `getToken()` odlišný řetězec reprezentující platný token. Token tak nebude možné odcizit zmíněnou technikou, ani pokud nejsou použity jiné způsoby její prevence popsané v [41].

2.6.6 Chybový stav aplikace

Dostane-li se plugin do chybového stavu, ať už zjištěním chyby v konfiguraci, nebo zachycením jiné neočekávané výjimky z některého z aktivních modulů, musí dojít k přerušení nahrávání vyžádané stránky, protože nelze zaručit, že vzniklá chyba by nevedla k zobrazení stránky, ke které by byl v bezchybném stavu přístup omezen.

Jako odpověď na požadavek bude vrácena obecná chybová stránka bez bližších informací o chybě, jejichž zveřejnění by odhalilo informace o instalaci konkrétního systému. Zpráva o chybě bude zaznamenána do aplikačního logu, jež je popsán v další sekci.

2.6.7 Aplikační log

Použití aplikačního logu je doporučeno mj. pro identifikaci bezpečnostních incidentů, zajištění nepopiratelnosti akcí a monitorování stavu aplikace [42]. Seznam událostí, které budou zaznamenávány do logu, je v tabulce C.1.

Modulům s podporou pro tvorbu logu bude možné předat libovolnou implementaci rozhraní `Psr\Log\LoggerInterface`, která je součástí standardu PSR-3 [43]. Uživatel pluginu si tak může zvolit libovolnou implementaci v závislosti na požadavcích na úložiště a zpracování logovaných událostí. Každý modul navíc umožňuje specifikovat jinou instanci, aby bylo možné události z různých částí systému zpracovávat jiným způsobem. Toto nastavení je provedeno editací konfiguračního souboru poskytovatele závislostí pro plugin.

Možnou implementací uvedeného rozhraní je např. nástroj Monolog. Umožňuje více způsobů pro ukládání logu do souborů, databází i externích služeb. Je také možné nastavit dodatečné akce v závislosti na závažnosti logované události (např. zaslání emailu při vzniku kritické chyby). Lze definovat i různé úpravy formátů, které budou na log před jeho zapsáním aplikovány (např. převod do formátu specifického pro zvolený způsob úložiště) nebo způsoby pro rozšíření logu o další údaje (vytížení systému v době události nebo přidání IP adresy uživatele) [44].

Pokud je log ukládán do souboru umístěném na stejném webovém serveru, kde je plugin nainstalován, musí být umístěn mimo veřejný prostor webového serveru a musí mít nastavena přísná přístupová oprávnění. V případě ukládání logu na vzdálené umístění je nutné použití bezpečných protokolů a mohou být použity specializované systémy, jako je CLFS (Common Log File System) [42].

2.6.8 Správa session

Session je relace mezi webovým serverem a uživatelem, umožňující udržovat informace (např. o přihlášeném uživateli) mezi jednotlivými HTTP požadavky [45]. Na straně serveru jsou atributy aktuální relace dostupné v proměnné `$_SESSION`. Všechny části pluginu k ní budou přistupovat přes rozhraní `SessionInterface`, jehož návrh je na obrázku 2.6. Ve výchozím stavu bude jako implementace tohoto rozhraní použita třída `SymfonySession` využívající komponentu HttpFoundation z frameworku Symfony.

Přístupem přes rozhraní bude umožněn snadný přechod na jiného správce *session*. Jde například o plugin PicoSession [46], připravovaný oficiální plugin pro systém Pico určený pro zprostředkování jednotného přístupu k *session* proměnným všem pluginům v systému Pico. Plugin zatím není dokončen, jeho rozhraní je však již stanoveno [46]. Ve svém pluginu proto poskytnu i implementaci `SessionInterface` pracující s tímto pluginem, aby bylo možné na tuto preferovanou možnost správy přejít, jakmile dojde k jejímu vydání.

Identifikátor *session* bude obnovován v konfigurovatelných intervalech a při důležitých změnách, jako je přihlášení uživatele nebo počátek relace pro změnu hesla; předchozí identifikátor bude zneplatněn. Tím dojde k odstranění hrozby *session fixation* (CWE-384 [14]) [45]. Konfigurace pluginu dále umožní nastavení maximální doby platnosti *session* a maximální povolené doby neaktivity, po které dojde ke zrušení *session*. Příklad takové konfigurace je v D.2. Další důležité parametry pro bezpečnost *session*, jako jsou např. vlastnosti příslušného souboru cookie [47], budou nastavitelné v konfiguračním souboru poskytovatele závislostí, nejsou-li již nastaveny v konfiguraci PHP.

2.6.9 Reakce na další hrozby

Při chybném přihlášení nesmí dojít ke zveřejnění informace o tom, zda bylo neplatné uživatelské jméno, nebo heslo (CWE-203 [14]). To by dalo potenciálnímu útočníkovi informaci o existenci konkrétního uživatelského jména, na které by mohl cílit další útoky. Tato informace by neměla být dostupná ani jinými způsoby. Například pokud uživatelské jméno neexistuje, verifikace zadaného hesla musí být vždy provedena, aby rozdíl mezi provedením a neprovedením časově náročné hashovací funkce nebyl znatelný v době odpovědi webového serveru (CWE-208 [14]). Stejně tak technika *rate limiting* musí být při přihlášení aplikována i na neexistující uživatelská jména.

Jedním z podstatných bezpečnostních rizik pro webové aplikace je *Cross-Site Scripting* (XSS) – technika, která kvůli špatnému ošetření uživatelských vstupů v aplikaci umožní útočníkovi spouštět zákeřné skripty v prohlížeči oběti [48]. Systém Pico, a tedy i navrhovaný plugin, používá pro sestavení stránek šablonovací systém Twig, který umožňuje na výstup aplikovat filtry pro *escaping* – převod speciálních znaků na HTML entity [49]. Jejich vhodným použitím se eliminuje možnost spuštění skriptu využívající výstup z pluginu.

Kapitola 3

Realizace

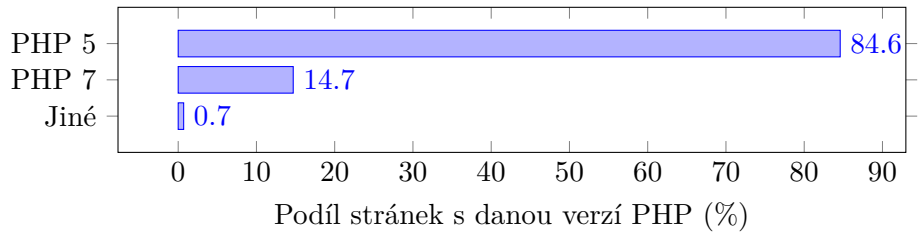
Implementaci pluginu jsem provedl dle návrhu popsaného v předchozí kapitole. Zdrojový kód pluginu a uživatelská dokumentace jsou v příloze této práce. Vybranými aspekty realizace se zabývám v této kapitole.

3.1 Použité technologie

3.1.1 Volba verze PHP

Požadavky na plugin nespecifikují, jaké verze PHP by měl podporovat. Samotný systém Pico vyžaduje PHP verze 5.3.6 a vyšší [2]. Tato verze však byla vydána již v roce 2011 a bezpečnostní podpora pro ni skončila v roce 2014 [50]. Využití verze, pro které již nejsou vydávány aktualizace by uživatele vystavilo známým zranitelnostem, které jsou v novějších verzích již opraveny.

Verze 5.6 je poslední vydání PHP 5 a je k ní poskytována prodloužená podpora až do roku 2019. Navzdory blížícímu se konci této podpory používá verzi 5 v současné době dle [51] stále více jak 80 % webových serverů s PHP. Jako minimální podporovanou verzi PHP jsem proto stanovil verzi 5.6.



Obrázek 3.1: Rozložení současného zastoupení verzí PHP [51]

3.1.2 Použité knihovny

Pro správu závislostí pluginu jsem využil nástroj Composer, který umožňuje automatické stažení a instalaci závislostí ve specifikovaných verzích dle konfiguračního souboru `composer.json` [52]. Při implementaci jsem se snažil vytvořit závislosti na co nejmenším množství knihoven třetích stran, aby byl plugin, stejně jako systém Pico, co nejvíce minimalistický. Při implementaci pluginu jsem využil následující knihovny (uvedené názvy odpovídají jejich označení v nástroji Composer – tj. vydavatel/název):

symfony/yaml Zajišťuje oboustranný převod mezi textem ve formátu YAML a PHP polem [53]. V pluginu je tento formát použit pro interpretaci konfiguračních souborů. Stejnou knihovnu využívá i systém Pico [2], při instalaci přes Composer tak nedojde k jejímu opětovnému stažení.

symfony/http-foundation Knihovna poskytující objektově orientovanou implementaci pro práci s HTTP specifikací. V PHP je HTTP požadavek reprezentován jako sada speciálních globálních proměnných (`$_GET`, `$_POST`, `$_SESSION`, …). Tato knihovna nahrazuje tyto proměnné objektově orientovanou vrstvou, což je vhodné pro testování [53].

paragonie/random_compat Knihovna pro využití kryptograficky bezpečného pseudo-náhodného generátoru nutného např. pro tvorbu CSRF tokenů, tokenů pro reset hesla, aj. To od PHP 7 umožňuje nativní funkce `random_bytes()`, kterou knihovna emulací zpřístupňuje i v nižších verzích PHP. Toho je docíleno přístupem ke zdroji náhodnosti příslušného operačního systému [54].

Ačkoliv je v PHP 5 dostupná funkce `openssl_random_pseudo_bytes()`, která má poskytnout totéž, její použití jsem nezvolil kvůli chybě, která v určitých verzích vede k nedostatečně náhodným hodnotám (bezpečnostní problém s označením CVE-2015-8867 [55]).

league/oauth2-client Knihovna pro implementaci klientských aplikací protokolu OAuth 2.0. Umožňuje použití protokolu s jakýmkoliv autorizačním serverem vyhovujícím standardu OAuth 2.0 (RFC 6749) [34]. Zdůvodnění jejího výběru jsem popsal již v sekci 2.4.

league/container Kontejner pro vkládání závislostí, jehož použití umožní rozčlenění aplikace do samostatných komponent, což vede k lepší testovatelnosti [56]. Ukázka jeho použití je v příloze D.1.

psr/log, psr/simple-cache Jde o standardizovaná rozhraní pro práci s aplikačním logem a s cache. Použití standardizovaných rozhraní umožní uživateli pluginu široké možnosti pro výběr implementace.

3.2 Implementace jednotlivých částí

3.2.1 Datová vrstva

Pro řízení přístupu k datovým souborům jsem využil mechanismus zámků dostupný v sadě funkcí pro práci se souborovým systémem v PHP, konkrétně ve funkci `flock()` [57]. Před čtením libovolného souboru plugin získá voláním této funkce s parametrem `LOCK_SH` sdílený zámek a při zápisu parametrem `LOCK_EX` exkluzivní zámek pro zápis. Tím je zaručeno, že nedojde k čtení souboru, do kterého zrovna probíhá zápis. Stejně tak je vyloučen paralelní zápis do jednoho souboru. Při požadavku na exkluzivní přístup dojde k blokujícímu čekání, dokud soubor nemá žádné jiné zámky, což může vést k odepření služby (DoS) v případě mnoha požadavků na zápis do jednoho souboru. Úložiště s možností častého zápisu jsem proto rozčlenil do více souborů (např. uživatelské účty), aby jejich uzamknutí neomezovalo provádění ostatních požadavků, které tento soubor nepotřebují.

Atomicitu operace zápisu (provede se buď celý, nebo vůbec) jsem původně zamýšlel realizovat zápisem do dočasného souboru a až po jeho úspěšném zapsání přejmenováním na skutečný název zapisovaného souboru. Pokud by došlo během zápisu k náhlému ukončení PHP procesu (např. z důvodu vypnutí serveru), v datovém adresáři by sice zůstal nekompletní soubor, konzistence originálního souboru by ale zůstala nenarušena. Samotné přejmenování je na unixových systémech implementováno jako atomická operace [58]. Tento způsob jsem však nakonec nezvolil, protože není slučitelný se synchronizačním mechanismem popsaným výše. Seznam zámků je uložen ve struktuře *inode* a při přejmenování by došlo ke změně asociace mezi názvem souboru a strukturou *inode*. Ostatní požadavky, které o exkluzivní přístup pro zápis požádaly před přejmenováním souboru, by po jeho získání zapsaly do již neodkazovaného souboru, čímž by došlo ke ztrátě některých zápisů.

U zápisů kritických na ztrátu dat (v pluginu pouze zápis do konfigurace se seznamem uživatelů) je nejdříve provedena kopie původního souboru, který je až poté přepisován novým obsahem. Je-li tento zápis úspěšný, soubor s kopií je následně smazán, v opačném případě je k dispozici pro případnou obnovu a nedojde k nenávratné ztrátě původních dat. Vzhledem k tomu, že zapisované soubory jsou velmi malé, a jejich zápis tak bude rychlý, pravděpodobnost selhání systému právě v okamžiku zápisu považuji za velmi malou. I tak však jde o nedostatek, jehož odstranění je možné použitím alternativní datové vrstvy, což návrh pluginu umožňuje.

3.2.1.1 Úložiště pro rate limiting

Inkrementace čítače provedených akcí je ve třídě `RateLimit` realizována nejdříve získáním aktuální hodnoty z datové vrstvy, následnou kontrolou dosavadní blokace a kontrolou expirace čítače a případným vynulováním, samotnou inkrementací a uložením nového záznamu. Celý popsaný proces musí být

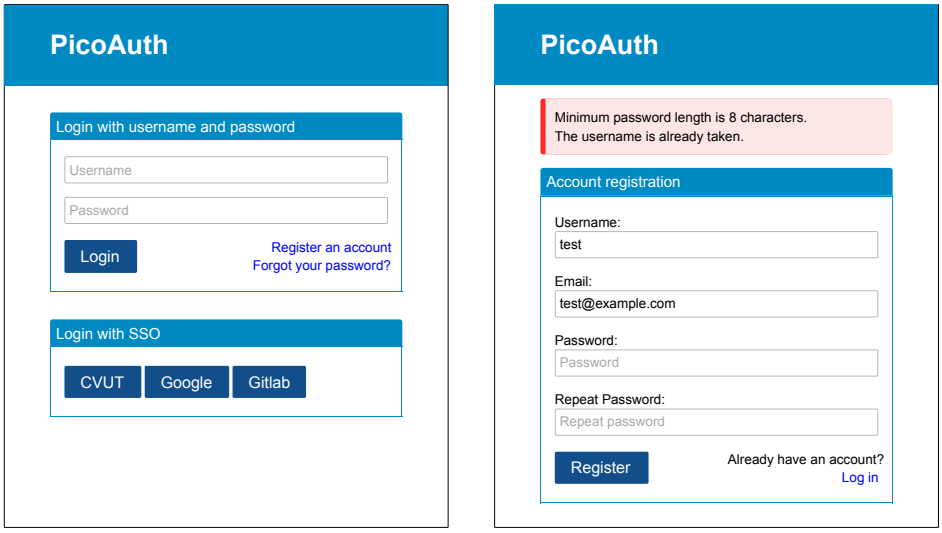
proveden jako atomická operace, aby jiný požadavek v průběhu jejího provádění neprovedl změnu čítače, čímž by došlo ke ztrátě informace a možnosti provedení značně vyššího počtu akcí, než je povoleno.

Atomicitu jsem zajistil přidáním metody `transaction()` do rozhraní úložiště `RateLimitStorageInterface`. Datové operace provedené mezi začátkem a koncem jedné transakce musejí být provedeny s jedním zámkem. V implementaci textového úložiště je tak v tomto případě exkluzivní zámek vystaven již na začátku transakce a je držen, dokud nejsou provedeny všechny operace.

Do úložiště pro *rate limiting* může být zapisováno i větší množství záznamů. Jejich počet závisí na konfiguraci limitů, velikosti sjednocujících IP podsítí a počtu adres, odkud je útok prováděn. Výkon textové databáze tak nemusí být pro toto použití dostačující a může dojít ke zpomalení nahrávání stránky a v mezních případech až k úplnému odepření služby. Proto jsem pro `RateLimitStorageInterface` implementoval i úložiště založené na databázi SQLite 3, které požadavky dokáže zpracovat rychleji, a dokáže tak reagovat i na podstatně větší útok. Toto úložiště si uživatel pluginu může aktivovat, má-li k dispozici příslušné rozšíření PHP. Implementaci výše zmíněné metody `transaction()` jsem v kontextu databázového úložiště implementoval příslušnými SQL příkazy pro kontrolu transakcí. V případě SQLite jde o volání `BEGIN EXCLUSIVE TRANSACTION`. Během takto zahájené transakce je databáze pro všechny ostatní požadavky zamknuta a dochází k blokujícímu čekání [25].

3.2.2 Implementace SSO přihlášení

Výchozí přihlašovací formulář automaticky zobrazí volby pro nakonfigurované poskytovatele, které lze použít pro jednotné přihlášení. Uživatel přihlášení zahájí volbou poskytovatele, což je realizováno jako POST požadavek s identifikátorem poskytovatele. Plugin přečte konfiguraci vybraného poskytovatele, jeho identifikátor uloží do *session* a zahájí tok protokolu OAuth přesměrováním na autorizační endpoint poskytovatele. Uživatel aplikaci autorizuje a plugin získá na svém endpointu pro odpověď autorizační kód a stavový řetězec, který se musí shodovat s řetězcem odeslaným na autorizační endpoint při prvním přesměrování. Plugin ze *session* získá identifikátor poskytovatele a přečte jeho konfiguraci, na základě které pak využije autorizační kód pro získání přístupového tokenu. Voláním metody `getResourceOwner($accessToken)` na instanci třídy pro obsluhu zvoleného poskytovatele jsou získány informace o uživateli. Na základě konfigurace mapování atributů jsou z nich vybrané atributy (vždy alespoň identifikátor uživatele) použity k vytvoření instance třídy `User`. Vyplněnou instanci třídy `User` předá modul instanci pluginu (`PicoAuthPlugin`), na které také provede volání `afterLogin()`. Zde jsou již provedeny operace společné pro všechny způsoby přihlášení, jako je změna *session* identifikátoru, uložení instance `User` do *session*, záznam o přihlášení do logu a přesměrování přihlášeného uživatele na stránku po přihlášení.



(a) Přihlašovací formulář

(b) Registrační formulář a způsob prezentace chybně zadaných údajů

Obrázek 3.2: Výchozí uživatelské rozhraní pluginu

3.2.3 Výchozí vzhled

Pro přizpůsobení vzhledu stránek využívá systém Pico šablonovací systém Twig. Pro použití pluginu je nutné, aby šablona obsahovala nezbytné prvky jako je zobrazení jména přihlášeného uživatele, tlačítko pro odhlášení, formulář pro přihlášení aj. Systém Pico pluginům neumožňuje rozšiřitelnost aktuálně nastavené šablony. Aby uživatel mohl plugin integrovat do své stávající šablony, musí provést její úpravu.

Tato úprava spočívá v přidání souboru `picoAuth_base.twig` do adresáře s použitou šablonou. Jde o soubor se základem vzhledu, do kterého plugin vloží formuláře sestavené výchozím způsobem. Rozšířená možnost úpravy pak spočívá v poskytnutí šablony pro jednotlivé stránky (`login.twig`, `register.twig`, `logout.twig`, …).

Aby šlo plugin použít i bez úprav šablony, integroval jsem do pluginu výchozí vzhled, který bude použit, nejsou-li potřebné soubory nalezeny v adresáři použité šablony systému Pico. K tomu jsem použil volání `$twig->getLoader()->addPath()` pro přidání alternativního zdroje šablon. Výchozí vzhled vybraných stránek pluginu je na obrázku 3.2. Jde o záměrně jednoduchý návrh, aby bylo snadné jej použít jako předlohu pro tvorbu upravených vzhledů. Pokud uživatel nechce provádět žádné úpravy vzhledů, může si v systému Pico aktivovat vzhled `picoauth-theme`, který stejný motiv aplikuje na celý systém Pico a zároveň poskytne lištu pro přihlášení/odhlášení uživatele (obrázek E.1). Výchozí šablona pluginu je responsivní a pro sestavení jejích stylů jsem využil CSS preprocesor Sass.

3.2.4 Výchozí konfigurace

Aby se zjednodušil proces konfigurace, jednotlivé konfigurační soubory nemusí obsahovat hodnoty pro všechny dostupné parametry. V takovém případě jsou doplněny o výchozí hodnoty. To je realizováno rekursivním sjednocením pole výchozích hodnot a pole přečteného z konfigurace. Sjednocení je provedeno do stanovené hloubky dle charakteru konfiguračního souboru. To je implementováno v metodě `AbstractConfigurator::applyDefaults($config, array $defaults, $depth = 1)`, která je volána ještě před validací obsahu konfigurace. Ve výchozím stavu jsou deaktivovány části pluginu provádějící operace zápisu, aby plugin bylo možné nainstalovat i bez nutnosti konfigurace oprávnění pro zápis v souborovém systému.

Výchozí konfigurace poskytovatele závislostí je umístěna v souboru `src/container.php` v adresáři pluginu. Jeho editace na tomto umístění by nebyla ideální, protože při případné aktualizaci by došlo ke konfliktu. Umístí-li uživatel upravenou verzi tohoto souboru do adresáře s konfigurací systému Pico, je tato verze upřednostněna.

3.3 Instalace

Instalace pluginu bude prováděna nástrojem Composer, což je preferovaný způsob instalace pluginů v systému Pico 2.0. Závislosti všech pluginů, šablon i samotného systému Pico jsou na jednom místě a jsou spravovány jednotně. Jde však o nástroj spouštěný z příkazové řádky a např. uživatelé využívající služby sdíleného hostingu jej nemusejí mít k dispozici.

Plugin jsem proto implementoval tak, aby bylo možné jej nainstalovat i bez nástroje Composer. Instalace pak spočívá v extrakci archivu pluginu s předem staženými závislostmi do adresáře `plugins` systému Pico. Při použití tohoto způsobu jsou závislosti pluginu na jiném umístění, než závislosti systému Pico, a tak musí plugin před svým zavedením provést registraci autoloaderu. To jsem implementoval ve třídě `PicoAuth`, která se pro systém Pico jeví jako hlavní třída pluginu, jejím úkolem je však pouze provést registraci závislostí a přicházející události pouze předat skutečné třídě pluginu `PicoAuthPlugin`.

3.3.1 Dokumentace

K pluginu jsem poskytl návod pro instalaci a referenční dokumentaci. Pro formátování těchto dokumentů jsem použil jazyk Markdown, který je podporován pro tvorbu dokumentační sekce (wiki) projektů na službách jako jsou GitLab nebo GitHub. Zároveň je tak možné dokumentaci zobrazit jako stránky v systému Pico.

K dokumentaci zdrojového kódu jsem použil syntaxi PHPDoc. Aplikace phpDocumentor pak umožňuje vygenerovat dokumentaci z komentářů ve zdrojových souborech. Oba typy dokumentace jsou zahrnuty v příloze práce.

Kapitola 4

Testování

4.1 Jednotkové testy

Jednotkové testy ověřují funkčnost jednotlivých částí pluginu. Pro realizaci jednotkových testů jsem použil framework PHPUnit, jehož struktura odpovídá architektuře xUnit [59]. Pro automatické spouštění testů jsem využil nástroj pro průběžnou integraci GitLab CI dostupný na školním serveru GitLab. Ten při každé změně provede stažení závislostí pluginu nástrojem Composer a následné spuštění testů. To je provedeno v izolovaném systému, kontejneru nástroje Docker. Testy jsou spouštěny paralelně ve dvou kontejnerech s odlišnými verzemi PHP – verzí 5.6 (nejnižší podporovanou) a verzí 7.2 (aktuální nejvyšší dostupná).

Při psaní testů jsem se zaměřil zejména na ty části pluginu, které jsou kritické pro bezpečnost. Aktuální sada testů se skládá ze 465 kontrol v 304 testovacích případech, což odpovídá nadpolovičnímu pokrytí všech metod pluginu. Výstup z testovacího nástroje s informacemi o pokrytí jednotlivých tříd je v elektronické příloze práce. Stoprocentní pokrytí je předmětem pro další aktualizace pluginu.

4.2 Statická analýza kódu

Statická analýza kódu slouží k auditu zdrojového kódu bez jeho spuštění. Na základě různých indikátorů dokáže odhalit problémové části kódu, jejichž opravu by měl programátor zvážit při manuální diagnostice [60]. Na zdrojový kód implementovaného pluginu jsem aplikoval 3 statické analyzátory s různým zaměřením. Detailní výstupy analyzátorů jsou v příloze práce.

Nástroj Exakat [60] je syntaktický analyzátor pro PHP s možností generování výstupních zpráv se zaměřením na bezpečnost, optimalizaci rychlosti a kvalitu kódu. Jeho analýza nenalezla v kódu pluginu žádná místa, která by byla označena jako kritická. V kategorii analýzy bezpečnosti nalezl 10 varo-

vání. V konkrétním kontextu jejich výskytu však žádné z nich nepovažuji za rizikové. Na základě některých doporučení z ostatních kategoriích jsem provedl příslušné úpravy kódu. Úpravu méně podstatných doporučení jsem přenechal na případné další aktualizace pluginu. Jde například o možné zrychlení použitím plně kvalifikovaných jmen použitých globálních funkcí PHP volaných z neglobálních jmenných prostorů.

Analyzátor PHPStan [61] kromě kódu analyzuje i komentáře ve formátu phpDoc. Poskytuje tak i kontrolu, zda typy definované v phpDoc komentářích souhlasí se skutečnými hodnotami v kódu. Tím může nalézt problémy vedoucí k použití neočekávaného typu, které by se mohly v neočekávaných situacích projevit až za běhu kvůli dynamickému typování v PHP. Při prvním spuštění na kód pluginu tento analyzátor nalezl 56 chyb, zejména v nesouladu typů. Ty jsem ve všech výskytech opravil.

Nástroj PHP CodeSniffer [62] je zaměřen na kontrolu, zda zdrojový kód odpovídá zvoleným standardům. Tím je zajištěno, že kód neobsahuje nepřehledné formulace a jeho formátování je konzistentní. Použil jsem stejný standard pro styl kódu, jakým je formátován kód systému Pico, který tento analyzátor používá také. Jde o standard s označením PSR-2. Spouštění tohoto analyzátoru jsem přidal k testům do nástroje pro průběžnou integraci, aby kontrola kvality kódu byla prováděna průběžně.

4.3 Test bezpečnosti

Pro automatizovaný test bezpečnosti systému Pico s aktivovaným pluginem jsem použil nástroj OWASP ZAP [63] určený pro analýzu bezpečnosti webových aplikací. Nalezl 3 problémy související s nastavením testovacího webového serveru, jejichž náprava není v kompetenci pluginu. Zbývající nalezený nedostatek spočíval v chybějícím atributu `autocomplete="off"` na HTML elementu pro zadání hesla, což prohlížečům umožní heslo uložit a může dojít k jeho kompromitaci (CWE-525 [14]). Atribut jsem do výchozí šablony doplnil.

Test ochrany proti systematickému hádání hesel jsem provedl nástrojem THC Hydra [64], běžně používaným nástrojem pro útoky na vzdálené autentizační systémy. Program na základě parametrů z příkazové řádky odesílá vhodně sestavené POST požadavky na pokus o přihlášení. Zkoušená hesla a uživatelská jména umožňuje číst ze slovníkového souboru, nebo generovat všechny možnosti. Příklad použití na implementovaný plugin a uživatelské jméno `user` je v ukázce 4.1. Přihlášení je v tomto příkladu považováno za neúspěšné, obsahuje-li odpověď řetězec `Wrong`. Příklad pro zjednodušení neobsahuje CSRF token. Test prokázal správné fungování omezení počtu pokusů a běh programu Hydra se po překročení limitu předčasně ukončil. Výchozí slovník slabých hesel dodávaný k distribuci Kali Linux určené pro testy bezpečnosti má 14 mil. záznamů. Jeho vyzkoušení na jeden uživatelský účet by při výchozím nastavení omezení trvalo 41 let.

```
$ hydra -I -l user -P wordlist.txt -F -t 16 -w 10 -V -s 8080 \
  10.0.2.2 http-post-form \
  "/index.php?login:username=\^USER\^\&password=\^PASS\^:Wrong"
```

Ukázka 4.1: Použití nástroje THC Hydra na implementovaný plugin

```
$ dirb http://10.0.2.2/Pico/ -w -X ,.yml,.md

---- Scanning URL: http://10.0.2.2/Pico/ ----
==> DIRECTORY: http://10.0.2.2/Pico/config/
==> DIRECTORY: http://10.0.2.2/Pico/content/
---- Entering directory: http://10.0.2.2/Pico/config/ ----
+ http://10.0.2.2/Pico/config/config.yml (CODE:200|SIZE:2134)
---- Entering directory: http://10.0.2.2/Pico/content/ ----
+ http://10.0.2.2/Pico/content/index.md (CODE:200|SIZE:16573)
```

Ukázka 4.2: Výstup nástroje dirb při chybně konfigurovaném systému Pico

Významnou hrozbou při použití pluginu je chybné nastavení webového serveru s instalací systému Pico, které může vést k úniku konfiguračních souborů a textů stránek. Na chybně konfigurovanou stránku se systémem Pico jsem použil nástroj dirb, určený pro systematické procházení adresářového prostoru webového serveru za využití slovníku častých názvů adresářů [65]. Úspěch programu i při použití výchozího slovníku potvrdil závažnost problému. Ukázka použití a zkrácený výstup programu je v ukázce 4.2. Pokud útočník disponuje znalostí, že cílová stránka využívá systém Pico, pro vyzkoušení dostupnosti těchto cest ani specializovaný nástroj není nutný. Instalátor pluginu však dokáže uživatele na nebezpečné nastavení upozornit, viz obrázek E.2.

Správné fungování synchronizačních mechanizmů pro zápis do souborů jsem ověřil odesláním velkého množství paralelních požadavků na akci podléhající omezením na počet provedení. Uložený počet akcí se shodoval s počtem odeslaných požadavků. Zároveň jsem tímto způsobem provedl srovnání rychlosti datové vrstvy. Při 100 000 záznamech v úložišti trvá 1 aktualizace záznamu 80 ms v textové databázi a 5 ms v SQLite3, při 20 paralelních požadavcích je již rozdíl 300 ms vs 20 ms. Pro použití SQLite3 s mnoha paralelními požadavky bylo nutné nastavit vyšší hodnotu pro maximální dobu čekání na přístup do uzamknuté databáze (`SQLite3::busyTimeout`). Výchozí hodnota tohoto nastavení je 0 [66], což způsobuje vysokou chybovost i při nízkém zatížení. Tuto hodnotu jsem umožnil uživateli pluginu při použití úložiště `RateLimitSqliteStorage` specifikovat. Testovací skript k výše zmíněnému testu je v elektronické příloze práce.

Závěr

Cílem práce bylo navrhnout a implementovat rozšíření umožňující autentizaci a autorizaci uživatelů pro minimalistický webový systém pro správu obsahu Pico. Provedenou analýzou jsem zjistil, že v současné době dostupná řešení nemají nativní podporu v nejnovější verzi systému Pico a zároveň jsem v jejich implementaci nalezl několik bezpečnostních nedostatků. Provedl jsem implementaci pluginu, který umožňuje přihlašování uživatelů vlastními uživatelskými účty, nebo způsobem jednotného přihlášení za využití protokolu OAuth 2.0. Plugin poskytuje konfigurovatelné omezení přístupu ke stránkám na základě seznamu uživatelů či uživatelských skupin. Cíl práce tak byl splněn.

Plugin jsem navrhl s ohledem na rozšiřitelnost. Autentizační a autorizační moduly jsou implementovány jako samostatné a na sobě nezávislé moduly. Ty jsou v pluginu registrovány deklarativním způsobem v konfiguračním souboru a komunikují přes pevně stanovené rozhraní. Rozšiřující modul je tak možné zavést bez nutnosti změn zdrojového kódu. V této fázi je plugin použitelný, pokud by byl rozšiřován, bylo by výhodou doplnit autentizační metodu využívající protokol OpenID Connect, který umožňuje automatickou konfiguraci poskytovatelů identity a poskytuje dodatečné bezpečnostní mechanismy oproti současné implementaci využívající OAuth 2.0.

Při návrhu jsem uvažoval i různou úroveň náročnosti uživatelů. Plugin je použitelný hned po instalaci, umožňuje však i široké možnosti konfigurace. Implementace jednotlivých částí pluginu lze konfiguračně zaměnit. Je možné provést výběr třídy pro realizaci úložiště pluginu, ukládání aplikačního logu, tvorby cache, odesílání emailu nebo správy *session* proměnných.

Výsledný plugin poskytnu komunitě uživatelů systému Pico zveřejněním na webu GitHub, kde jej plánuji udržovat a dále rozšiřovat. Také jej registruji do služby Packagist, což je repozitář pro správce závislostí Composer. Instalace a následné aktualizace tak bude možné provádět jednoduše přes spuštění příkazu ve správci závislostí.

Bibliografie

1. THE PICO COMMUNITY. *All About Pico: What is a "stupidly simple, blazing fast, flat file CMS" anyway?* [online]. 2017 [cit. 2018-04-15]. Dostupné z: `http://picocms.org/about/`.
2. PELLEGROM, G. et al. *Pico* [online]. 2018. Verze 2.0 [cit. 2018-04-19]. Dostupné z: `https://github.com/picocms/Pico`.
3. SOCIALCOMPARE. *PHP CMS comparison* [online]. 2018 [cit. 2018-04-19]. Dostupné z: `http://socialcompare.com/en/comparison/php-cms-comparison-content-management-system`.
4. MACFARLANE, John. *CommonMark Spec* [online]. 2017. Verze 0.28 [cit. 2018-04-19]. Dostupné z: `http://spec.commonmark.org/0.28/`.
5. GRUBER, John. *Markdown* [online]. 2004 [cit. 2018-04-19]. Dostupné z: `https://daringfireball.net/projects/markdown/`.
6. BEN-KIKI, Oren; EVANS, Clark; NET, Ingy döt. *YAML Ain't Markup Language (YAML™)* [online]. 2009. Verze 1.2 [cit. 2018-04-19]. Dostupné z: `http://yaml.org/spec/1.2/spec.html`.
7. THE APACHE SOFTWARE FOUNDATION. *Apache HTTP Server Tutorial: .htaccess files* [online]. 2018 [cit. 2018-04-19]. Dostupné z: `https://httpd.apache.org/docs/2.4/howto/htaccess.html`.
8. THE PICO COMMUNITY. *Pico Changelog* [online]. 2018 [cit. 2018-04-19]. Dostupné z: `https://github.com/picocms/Pico/blob/pico-1.1/CHANGELOG.md`.
9. RUDOLF, Daniel. *Pico Deprecated Plugin* [online]. 2018 [cit. 2018-03-20]. Dostupné z: `https://github.com/picocms/pico-deprecated`.
10. THE PICO COMMUNITY. *Pico Plugins* [online]. 2018 [cit. 2018-03-20]. Dostupné z: `https://github.com/picocms/Pico/wiki/Pico-Plugins`.

11. BHORKAR, Tejaswini. A Survey of Password Attacks and Safe Hashing Algorithms. *International Research Journal of Engineering and Technology*. 2017, roč. 4, č. 12, s. 1556. ISSN 2395-0056.

12. BLEUZEN, Johan. *Pico Private* [online]. 2015 [cit. 2018-03-20]. Dostupné z: `https://github.com/jbleuzen/Pico-Private`.

13. LIAUTAUD, Nicolas. *Pico Users* [online]. 2017 [cit. 2018-03-20]. Dostupné z: `https://github.com/nliautaud/pico-users`.

14. THE MITRE CORPORATION. *The Common Weakness Enumeration (CWE™) Initiative* [online]. 2018 [cit. 2018-04-21]. Dostupné z: `https://cwe.mitre.org/`.

15. OWASP FOUNDATION. *Authentication Cheat Sheet* [online]. 2017 [cit. 2018-04-17]. Dostupné z: `https://www.owasp.org/index.php/Authentication_Cheat_Sheet`.

16. HARDT, D. (ed.). *The OAuth 2.0 Authorization Framework* [Internet Requests for Comments]. RFC Editor, 2012. ISSN 2070-1721. Dostupné z DOI: `10.17487/RFC6749`. RFC. RFC Editor.

17. PARECKI, Aaron. *OAuth 2.0 Servers* [online]. 2016 [cit. 2018-04-19]. Dostupné z: `https://www.oauth.com/oauth2-servers/definitions/`.

18. ORACLE CORPORATION ®. *Oracle ® Fusion Middleware*. 2013. Dostupné také z: `https://docs.oracle.com/cd/E39820_01/doc.11121/gateway_docs/oracle_api_gateway.pdf`. 11g Release 2, Chapter: API Gateway OAuth 2.0 Authentication Flows.

19. YANG, F.; MANOHARAN, S. A security analysis of the OAuth protocol. In: *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. 2013, s. 271–276. ISSN 1555-5798. Dostupné z DOI: `10.1109/PACRIM.2013.6625487`.

20. LI, Wanpeng; MITCHELL, Chris J.; CHEN, Thomas. Mitigating CSRF attacks on OAuth 2.0 and OpenID Connect. *CoRR*. 2018, roč. 1801.07983. Dostupné z arXiv: `1801.07983`.

21. LODDERSTEDT, T; MCGLOIN, M; HUNT, P. *Oauth 2.0 threat model and security considerations* [Internet Requests for Comments]. RFC Editor, 2013. Dostupné z DOI: `10.17487/RFC6819`. RFC. RFC Editor.

22. SAKIMURA, Nat; BRADLEY, John; JONES, Mike; MEDEIROS, Breno de; MORTIMORE, Chuck. OpenID Connect Core 1.0 incorporating errata set 1. *The OpenID Foundation, specification*. 2014.

23. FOWLER, Martin. *Inversion of Control Containers and the Dependency Injection pattern* [online]. 2004 [cit. 2018-04-19]. Dostupné z: `https://www.martinfowler.com/articles/injection.html`.

24. THE PHP GROUP. *Vendor Specific Database Extensions* [online]. 2001–2018 [cit. 2018-04-20]. Dostupné z: `http://php.net/manual/en/refs.database.vendors.php`.

25. KREIBICH, J. *Using SQLite*. O'Reilly Media, 2010. O'Reilly Series. ISBN 9780596521189. Dostupné také z: `https://books.google.cz/books?id=HFIM47wp0X0C`.

26. BOVET, Daniel Pierre; CESATI, Marco. *Understanding the Linux Kernel: From I/O Ports to Process Management*. O'Reilly Media, 2005. ISBN 9780596554910.

27. LIU, Na; ZHOU, Jianfei. Relational Database's Transaction Operation and the Concurrent Control. *International Journal of Database Theory and Application*. 2015, roč. 8, č. 2, s. 259–266. Dostupné z DOI: `10.14257/ijdta.2015.8.2.24`.

28. DRAGOONIS, Paul (ed.). *PSR-16: Common Interface for Caching Libraries* [online]. PHP Framework Interop Group, 2018 [cit. 2018-04-25]. Dostupné z: `https://www.php-fig.org/psr/psr-16/`. Technická zpráva.

29. THE PHP GROUP. *APC User Cache* [online]. 2001–2018 [cit. 2018-04-28]. Dostupné z: `http://php.net/manual/en/book.apcu.php`.

30. PROVOS, Niels; MAZIERES, David. A Future-Adaptable Password Scheme. In: *USENIX Annual Technical Conference, FREENIX Track*. 1999, s. 81–91.

31. THE PHP GROUP. *Password Hashing* [online]. 2001–2018 [cit. 2018-04-17]. Dostupné z: `http://php.net/manual/en/book.password.php`.

32. AUMASSON, Jean-Philippe. *Password Hashing Competition and our recommendation for hashing passwords: Argon2* [online]. 2015 [cit. 2018-04-17]. Dostupné z: `https://password-hashing.net`.

33. MALATRAS, A; COISEL, I; SANCHEZ, I. Technical recommendations for improving security of email communications. In: *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2016, s. 1381–1386. Dostupné z DOI: `10.1109/MIPRO.2016.7522355`.

34. THE PHP LEAGUE. *OAuth 2 Client: Easy integration with OAuth2 service providers* [online]. The PHP League, 2017 [cit. 2018-04-28]. Dostupné z: `http://oauth2-client.thephpleague.com/`.

35. JIRŮTKA, Jakub. *OAuth 2.0* [online]. 2017 [cit. 2018-04-28]. Dostupné z: `https://rozvoj.fit.cvut.cz/Main/oauth2`.

36. SYER, Dave. *Spring Security OAuth* [online]. 2018 [cit. 2018-04-20]. Dostupné z: `https://github.com/spring-projects/spring-security-oauth`.

37. HITACHI ID SYSTEMS, INC. *Password Management Best Practices* [online]. 2016 [cit. 2018-03-11]. Dostupné z: https://hitachi-id.com/password-manager/docs/password-management-best-practices.pdf.

38. GRASSI, Paul A. et al. NIST Special Publication 800-63B. Digital Identity Guidelines: Authentication and Lifecycle Management. *National Institute of Standards and Technology, Los Altos, CA*. 2017.

39. THE PHP GROUP. *Filesystem Functions* [online]. 2001–2018 [cit. 2018-04-17]. Dostupné z: http://php.net/manual/en/ref.filesystem.php.

40. OWASP FOUNDATION. *Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet* [online]. 2018 [cit. 2018-03-11]. Dostupné z: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet.

41. GLUCK, Yoel; HARRIS, Neal; PRADO, Angelo. BREACH: reviving the CRIME attack. 2013. Dostupné také z: http://css.csail.mit.edu/6.858/2015/readings/breach.pdf.

42. OWASP FOUNDATION. *Logging Cheat Sheet* [online]. 2017 [cit. 2018-04-17]. Dostupné z: https://www.owasp.org/index.php/Logging_Cheat_Sheet.

43. BOGGIANO, Jordi (ed.). *PSR-3: Logger Interface* [online]. PHP Framework Interop Group, 2018 [cit. 2018-04-25]. Dostupné z: https://www.php-fig.org/psr/psr-3/. Technická zpráva.

44. BOGGIANO, Jordi. *Monolog - Logging for PHP* [online]. 2017 [cit. 2018-04-17]. Dostupné z: https://github.com/Seldaek/monolog.

45. OWASP FOUNDATION. *Session Management Cheat Sheet* [online]. 2018 [cit. 2018-03-12]. Dostupné z: https://www.owasp.org/index.php/Session_Management_Cheat_Sheet.

46. RUDOLF, Daniel. *Pico Session: Pico's official utility plugin for session management* [online]. 2018. Verze 1.0.0 [cit. 2018-03-05]. Dostupné z: https://github.com/PhrozenByte/pico-session.

47. THE PHP GROUP. *Securing Session INI Settings* [online]. 2001–2018 [cit. 2018-04-17]. Dostupné z: http://php.net/manual/en/session.security.ini.php.

48. OWASP FOUNDATION. *Cross-site Scripting (XSS)* [online]. 2018 [cit. 2018-03-12]. Dostupné z: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

49. SENSIOLABS. *Twig: Twig for Template Designers* [online]. 2010–2018 [cit. 2018-05-06]. Dostupné z: https://twig.symfony.com/doc/2.x/templates.html.

50. THE PHP GROUP. *Supported Versions* [online]. 2001–2018 [cit. 2018-04-28]. Dostupné z: http://php.net/supported-versions.php.

51. GELBMANN, M. *Usage statistics and market share of PHP for websites* [online]. W3Techs, 2018 [cit. 2018-04-28]. Dostupné z: https://w3techs.com/technologies/details/pl-php/all/all.

52. ADERMANN, Nils; BOGGIANO, Jordi et al. *Composer: Dependency Manager for PHP* [online]. 2018 [cit. 2018-04-28]. Dostupné z: https://getcomposer.org/doc/.

53. SENSIOLABS. *Symfony: The Components Book*. 2015. Dostupné také z: https://symfony.com/pdf/Symfony_components_2.6.pdf.

54. PARAGON INITIATIVE ENTERPRISES et al. *Random compat: PHP 5.x support for random_bytes() and random_int()* [online]. 2018 [cit. 2018-04-28]. Dostupné z: https://github.com/paragonie/random_compat.

55. THE MITRE CORPORATION. *Common Vulnerabilities and Exposures (CVE-2015-8867)* [online]. 2015 [cit. 2018-04-15]. Dostupné z: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-8867.

56. THE PHP LEAGUE. *The League of Extraordinary Packages: Our packages* [online]. 2018 [cit. 2018-04-28]. Dostupné z: https://thephpleague.com/.

57. THE PHP GROUP. *flock: Portable advisory file locking* [online]. 2001–2018 [cit. 2018-04-28]. Dostupné z: http://php.net/manual/en/function.flock.php.

58. KERRISK, Michael. *Linux Programmer's Manual: RENAME(2)* [online]. 2017 [cit. 2018-04-28]. Dostupné z: http://man7.org/linux/man-pages/man2/rename.2.html.

59. BERGMANN, Sebastian. *PHPUnit: The PHP Unit Testing framework* [online]. 2018 [cit. 2018-05-01]. Dostupné z: https://github.com/sebastianbergmann/phpunit.

60. EXAKAT LTD. *Exakat: What is static analysis* [online]. 2014–2018 [cit. 2018-05-03]. Dostupné z: https://www.exakat.io/what-is-static-analysis/.

61. MIRTES, Ondřej et al. *PHPStan: PHP Static Analysis Tool* [online]. 2018 [cit. 2018-05-02]. Dostupné z: https://github.com/phpstan/phpstan.

62. SHERWOOD, Greg et al. *PHP_CodeSniffer* [online]. 2018 [cit. 2018-05-02]. Dostupné z: https://github.com/squizlabs/PHP_CodeSniffer.

63. OWASP FOUNDATION. *OWASP Zed Attack Proxy Project* [online]. 2018 [cit. 2018-05-01]. Dostupné z: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.

64. HEUSE, Marc. *THC Hydra* [online]. 2001–2018 [cit. 2018-05-01]. Dostupné z: `https://github.com/vanhauser-thc/thc-hydra`.

65. OFFENSIVE SECURITY ®. *DIRB: DIRB Package Description* [online]. 2014 [cit. 2018-05-08]. Dostupné z: `https://tools.kali.org/web-applications/dirb`.

66. THE PHP GROUP. *SQLite3::busyTimeout* [online]. 2001–2018 [cit. 2018-05-08]. Dostupné z: `http://php.net/manual/en/sqlite3.busytimeout.php`.

Příloha A

Seznam použitých zkratek

ACID Atomicity, Consistency, Isolation, Durability

ACL Access Control List

CMS Content Management System

CSRF Cross-site Request Forgery

CSS Cascading Style Sheets

CVE Common Vulnerabilities and Exposures

CWE Common Weakness Enumeration

DNS Domain Name System

DoS Denial of Service

HTML Hypertext Markup Language

JWE JSON Web Encryption

JWS JSON Web Signature

JWT JSON Web Token

OAAS OAuth Authorization Server

PSR PHP Standards Recommendations

RFC Request for Comments

SSO Single Sign-On

TLS Transport Layer Security

URI Uniform Resource Identifier

URL Uniform Resource Locator

XSS Cross-site Scripting

YAML YAML Ain't Markup Language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `bundle` ..... instalace systému Pico s pluginem
- `docs` ..... dokumentace pluginu
- `extras` ..... doplňkové přílohy práce
  - `examples` ..... příklady konfigurace
  - `diagram` ..... digram návrhu tříd pluginu
  - `tests` ..... výstupy testovacích nástrojů
- `src`
  - `pico-auth` ..... zdrojové kódy pluginu
  - `pico-auth-theme` ..... šablona vzhledu pro systém Pico
  - `oauth2-cvut` ..... třída poskytovatele pro ČVUT Zuul OAAS
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF

Příloha C

Tabulky

Tabulka C.1: Seznam významných událostí pro zápis do aplikačního logu

Původ	**Závažnost**	**Popis**
LocalAuth	Notice	Neplatný pokus o přihlášení
PasswordReset	Info	Odeslán email pro reset hesla
PasswordReset	Info	Navštíven platný odkaz pro reset hesla
PasswordReset	Info	Dokončena změna hesla
PasswordReset	Warning	Pokus o uplatnění neplatného tokenu
PasswordReset	Critical	Není specifikována třída pro odeslání emailu
PasswordReset	Critical	Chyba při odesílání emailu
Registration	Info	Nová registrace
Registration	Warning	Každých 10 % max. počtu uživatelů
OAuth	Notice	OAuth 2.0 vrácen chybový kód
OAuth	Warning	OAuth 2.0 chyba kontroly stavového kódu
OAuth	Critical	OAuth 2.0 chyba při získávání dat o uživateli
RateLimit	Notice	Při každém dosažení limitu
PicoAuthPlugin	Info	Úspěšné přihlášení (libovolným způsobem)
PicoAuthPlugin	Warning	Neplatný CSRF token (libovolný formulář)
PicoAuthPlugin	Critical	Nezachycená výjimka z kteréhokoliv modulu

Tabulka C.2: Dostupné události v Pico 2.0 [2]

Název události	**Popis**
`onPluginsLoaded`	Registrace pluginů
`onConfigLoaded`	Načtení konfigurace
`onRequestUrl`	Požadavek na adresu stránky
`onRequestFile`	Požadavek na soubor stránky
`onContentLoading`	Nahrávání obsahu souboru stránky
`on404ContentLoading`	Soubor nenalezen
`on404ContentLoaded`	Nahrání stránky s kódem chyby
`onContentLoaded`	Obsah stránky nahrán
`onMetaParsing`	Interpretace meta informací
`onMetaParsed`	Meta informace přečteny
`onContentParsing`	Interpretace obsahu stránky
`onContentPrepared`	Obsah připraven pro interpretaci
`onContentParsed`	Obsah interpretován
`onPagesLoading`	Čtení seznamu stránek
`onPagesDiscovered`	Seznam stránek načten
`onPagesLoaded`	Seznam stránek seřazen
`onCurrentPageDiscovered`	Zařazení aktuální stránky do kontextu
`onPageTreeBuilt`	Strom stránek sestaven
`onPageRendering`	Příprava šablony stránky
`onPageRendered`	Zpracování požadavku dokončeno
`onPluginManuallyLoaded`	Manuální nahrání pluginu
`onMetaHeaders`	Výchozí meta informací připraveny
`onYamlParserRegistered`	Zaregistrován analyzátor YAML
`onParsedownRegistered`	Zaregistrován analyzátor Markdown
`onSinglePageLoading`	Nahrávání stránky
`onSinglePageContent`	Přečten obsah stránky
`onSinglePageLoaded`	Obsah stránky interpretován
`onTwigRegistered`	Nahrán šablonovací systém Twig

Příloha D

Ukázky konfigurace

```
$container = new League\Container\Container;

$container->share('OAuth', 'PicoAuth\Module\Authentication\OAuth')
    ->withArgument('PicoAuth')
    ->withArgument('session')
    ->withArgument('OAuth.storage')
    ->withMethodCall('setLogger', ['logger']);

$container->share('OAuth.storage', 'PicoAuth\Storage\OAuthFileStorage')
    ->withArgument('configDir')
    ->withArgument('cache');

$container->share('session', 'PicoAuth\Session\SymfonySession')
    ->withArgument('session.storage');
```

Ukázka D.1: Způsob deklarace závislostí modulu OAuth v `container.php`

```
PicoAuth:
  authModules: [ LocalAuth, OAuth, PageACL, PageLock ]
  afterLogin: 'index'       # Redirection after login
  afterLogout: 'index'      # Redirection after logout
  alterPageArray: false     # Removes unaccessible pages from menus
  sessionInterval: 900      # Session regeneration every 15 minutes
  sessionIdle: 3600         # Maximum 1 hour of inactivity
  sessionTimeout: 7200      # Absolute maximum 2 hours
  rateLimit: true           # Option to control rate limiting
  debug: false              # Displays details on the error page
```

Ukázka D.2: Konfigurace pluginu v `config.yml` systému Pico

```
# LocalAuth configuration
registration:
  enabled: true
  nameLenMin: 3
  nameLenMax: 16
  maxUsers: 10000
passwordReset:
  enabled: false
  tokenLen: 50          # The length of the reset token
  tokenValidity: 7200   # The reset link is valid for 2 hours
  resetTimeout: 1800    # The password reset session lasts for 30 min.
  emailSubject: 'site_title% - Password Reset'
  emailMessage: >
    Hello,\n\n
    Visit the link to reset your password to %site_title%.\n\n
    Reset URL:\n%url%
accountEdit:
  enabled: true
login:
  passwordRehash: true
encoder: 'bcrypt'

# User accounts
# The accounts can be also in separate files in the data directory.
users:
  user1:
    pwhash: '$2y$10$qi5Xz.39w3pvPw2k4BvanudOoelF/Li6necs1DoQUKFRLec...'
    encoder: 'bcrypt'
    email: 'user1@picoauth.org'
    attributes:
      img: '/profile_picture.png'
  user2:
    pwhash: '$2y$10$KMWh.9YkxhhNqoMRBmNU5OnlerVSIJbmIckenlW.Llsa8a9...'
    encoder: 'bcrypt'
    displayName: 'Test User 2'
    groups: [ test, group2 ]
    pwreset: true
```

Ukázka D.3: Konfigurace přihlašování uživatelskými účty v `LocalAuth.yml`

```yaml
callbackPage: 'oauth_callback'
providers:
  CVUT:
    # Using a provider class
    provider: '\Tp3\OAuth2\Client\Provider\CVUT'
    options:
      clientId: '922c64d9-b359-4d42'
      clientSecret: 'f71e20bb-a75a-45b7'
    default:
      groups: [ cvut ]
      attributes:
        img: '/default.png'
  Gitlab:
    # No provider specified, defaults to the GenericProvider
    options:
      clientId: '39bd1bbf-fd92-419b'
      clientSecret: 'edd2daa5-d6fb-49cc'
      urlAuthorize: 'https://gitlab.com/oauth/authorize'
      urlAccessToken: 'https://gitlab.com/oauth/token'
      urlResourceOwnerDetails: 'https://gitlab.com/api/v4/user'
    attributeMap:
      userId: 'username'
      displayName: 'name'
      img: 'avatar_url'
```

Ukázka D.4: Konfigurace modulu OAuth v `OAuth.yml`

```yaml
# List of access rules
access:
  /secret:
    groups: [ default ]      # Everyone authenticated is in this group
  /secret/cvut:
    groups: [ cvut ]         # All users that used the CVUT provider
  /secret/test:
    users: [ test ]          # List of user identifiers
  /nonrecursive:
    recursive: false         # Rules apply resursively, this disables it
    users: [ user1 ]
  /secret/page:              # Any combination of groups and users
    groups: [ test, group2 ]
    users: [ user1, user2 ]

  # The following rule would apply on all pages
  # that are not covered by the rules above
  /:
    groups: [ default ]
```

Ukázka D.5: Konfigurace autorizace v `PageACL.yml`

```yaml
# The default encoder to be used if not specified
encoder: 'bcrypt'

# A list of locks and their keys
locks:
  secret:
    key: '$2y$10$xwxnbUdgafIN33t7oHQGJ...'
  lock2:
    key: 'weakKey'
    encoder: 'plain'            # Can be used, but is not recommended
    file: 'locked_screen.md'    # A custom page to show before unlock

# An association of the locks to pages, in the same way as in PageACL
urls:
  /secret/locked:
    lock: 'secret'
  /locked_A:
    lock: 'lock2'
    recursive: false
```

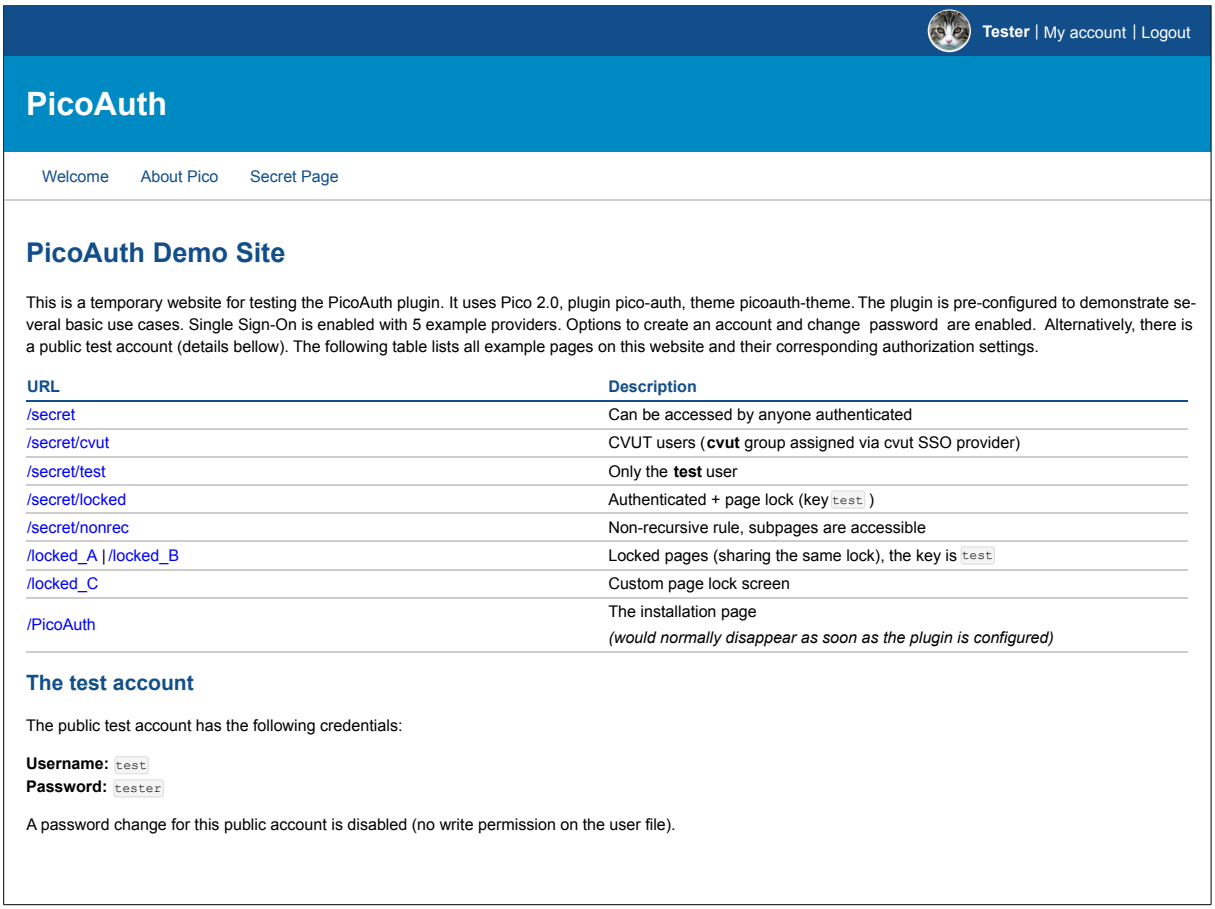
Ukázka D.6: Konfigurace zámků stránek v `PageLock.yml`

```yaml
# A list of actions to be rete limited (doesn't include all available)
actions:
  login:
    ip:
      count: 50
      counterTimeout: 43200   # Counter expiration after 12 hours
      blockDuration: 900      # Maximum speed 50 attempts / 15 minutes
      # A custom message to be shown if the limit is reached
      errorMsg: "Amount of failed attempts exceeded, \
                 wait %min% minutes."
      netmask_IPv4: 32  # Netmask 255.255.255.255
      netmask_IPv6: 48  # Netmask ffff:ffff:ffff::
    account:
      count: 10
      counterTimeout: 43200
      blockDuration: 900
  registration:
    ip:
      count: 10
      blockDuration: 86400
      errorMsg: "Amount of maximum submissions exceeded, \
                 wait %min% minutes."
```

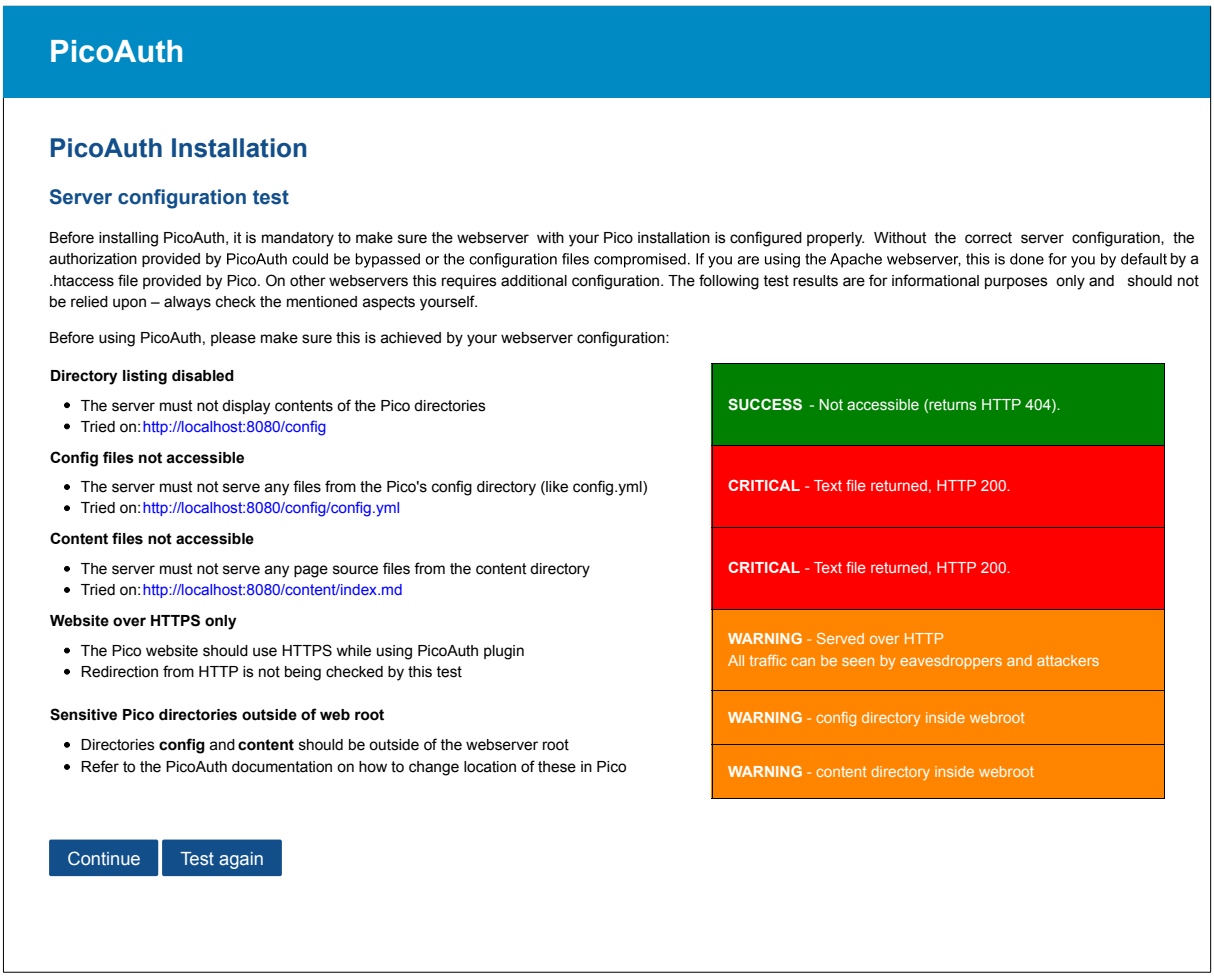
Ukázka D.7: Konfigurace *rate limiting* v `RateLimit.yml`

Příloha E

Obrazová příloha



Obrázek E.1: Šablona `picoauth-theme` optimalizována pro použití s pluginem



Obrázek E.2: Výstup instalátoru při chybné konfiguraci systému Pico